

TỦ SÁCH TRI THỨC DUY TÂN

NGUYỄN XUÂN HUY

SÁNG TẠO TRONG THUẬT TOÁN VÀ LẬP TRÌNH

với ngôn ngữ Pascal và C#

Tập 1

Tuyển các bài toán Tin nâng cao
cho học sinh và sinh viên giỏi

M U C L U C

	Lời nói đầu	i
Chương I	GIẢI MỘT BÀI TOÁN TIN	1
<i>Bài 1.1.</i>	<i>Số thân thiện</i>	2
<i>Bài 1.2.</i>	<i>Số cấp cộng</i>	8
<i>Bài 1.3.</i>	<i>Số cấp nhân</i>	11
<i>Bài 1.4.</i>	<i>Mảng ngẫu nhiên</i>	13
<i>Bài 1.5.</i>	<i>Chia mảng tỉ lệ 1:1</i>	16
<i>Bài 1.6.</i>	<i>Chia mảng tỉ lệ 1:k</i>	21
Chương II	SINH DỮ LIỆU VÀO VÀ RA	27
<i>Bài 2.1.</i>	<i>Sinh ngẫu nhiên theo khoảng</i>	27
<i>Bài 2.2.</i>	<i>Sinh ngẫu nhiên tăng</i>	29
<i>Bài 2.3.</i>	<i>Sinh hoán vị ngẫu nhiên</i>	31
<i>Bài 2.4.</i>	<i>Sinh ngẫu nhiên đều</i>	33
<i>Bài 2.5.</i>	<i>Sinh ngẫu nhiên tỉ lệ</i>	36
<i>Bài 2.6.</i>	<i>Sinh ngẫu nhiên tập tăng</i>	40
<i>Bài 2.7.</i>	<i>Sinh ngẫu nhiên tập cấp số cộng</i>	42
<i>Bài 2.8.</i>	<i>Sinh ngẫu nhiên mảng đối xứng</i>	43
<i>Bài 2.9.</i>	<i>Số độ cao h</i>	46
<i>Bài 2.10.</i>	<i>Tập các hoán vị</i>	49
<i>Bài 2.11.</i>	<i>Đọc dữ liệu từ tập vào mảng biết hai kích thước</i>	53
<i>Bài 2.12.</i>	<i>Đọc dữ liệu từ tập vào mảng biết một kích thước</i>	56
<i>Bài 2.13.</i>	<i>Đọc dữ liệu từ tập vào mảng đối xứng</i>	60
<i>Bài 2.14.</i>	<i>Đếm tàu</i>	62
<i>Bài 2.15.</i>	<i>Sắp đoạn</i>	65
Chương III	BÀN PHÍM VÀ MÀN HÌNH	79
<i>Bài 3.1.</i>	<i>Bảng mã ASCII</i>	79
<i>Bài 3.2.</i>	<i>Bộ Tú lơ khơ</i>	80
<i>Bài 3.3.</i>	<i>Hàm GetKey</i>	88
<i>Bài 3.4.</i>	<i>Trò chơi 15</i>	90
<i>Bài 3.5.</i>	<i>Bảng nhảy</i>	95
Chương IV	TỔ CHỨC DỮ LIỆU	107
<i>Bài 4.1.</i>	<i>Cụm</i>	107
<i>Bài 4.2.</i>	<i>Bài gộp</i>	112
<i>Bài 4.3.</i>	<i>Chuỗi hạt</i>	120

<i>Bài 4.4.</i>	<i>Sắp mảng rồi ghi tệp</i>	129
<i>Bài 4.5.</i>	<i>abc - sắp theo chỉ dẫn</i>	133
<i>Bài 4.6.</i>	<i>Xâu mẫu</i>	141
Chương V	PHƯƠNG PHÁP THAM LAM	153
<i>Bài 5.1.</i>	<i>Băng nhạc</i>	153
<i>Bài 5.2.</i>	<i>Xếp việc</i>	158
<i>Bài 5.3.</i>	<i>Xếp ba lô</i>	165
<i>Bài 5.4.</i>	<i>Cây bao trùm ngắn nhất</i>	170
<i>Bài 5.5.</i>	<i>Trộn hai tệp</i>	177
Chương VI	PHƯƠNG PHÁP QUAY LUI	193
<i>Bài 6.1.</i>	<i>Tám Hậu</i>	195
<i>Bài 6.2.</i>	<i>Từ chuẩn</i>	207
<i>Bài 6.3.</i>	<i>Tìm đường trong mê cung</i>	216
Chương VII	QUY HOẠCH ĐỘNG	227
<i>Bài 7.1.</i>	<i>Chia thưởng</i>	228
<i>Bài 7. 2.</i>	<i>Palindrome</i>	235
<i>Bài 7.3.</i>	<i>Cắm hoa</i>	243
<i>Bài 7.4.</i>	<i>Tìm các đường ngắn nhất</i>	253
Chương VIII	SUY NGẪM	267
<i>Bài 8.1.</i>	<i>Lát nền</i>	267
<i>Bài 8.2.</i>	<i>Chữ số cuối khác 0</i>	276
<i>Bài 8.3.</i>	<i>Hình chữ nhật tối đại trong ma trận 0/1</i>	281
<i>Bài 8.4.</i>	<i>Ma phương</i>	291
<i>Bài 8.5.</i>	<i>Tháp Hà Nội cổ</i>	308
<i>Bài 8.6.</i>	<i>Tháp Hà Nội xuôi</i>	311
<i>Bài 8.7.</i>	<i>Tháp Hà Nội ngược</i>	316
<i>Bài 8.8.</i>	<i>Tháp Hà Nội thẳng</i>	321
<i>Bài 8.9.</i>	<i>Tháp Hà Nội sắc màu (Hà Nội Cầu vòng)</i>	325

Lời nói đầu

Thế theo yêu cầu của đồng đảo bạn đọc, chúng tôi biên soạn lại cuốn Sáng tạo trong Thuật toán và Lập trình với các bài Toán Tin nâng cao cho học sinh và sinh viên nhằm cung cấp những kỹ thuật lập trình cơ bản để giải những bài toán khó trên máy tính.

Một bài toán tin được hiểu là khó nếu ta sử dụng thuật giải mới nảy sinh trong đầu khi vừa biết nội dung bài toán thì hoặc là ta thu được kết quả sai hoặc là lời giải thu được sẽ không hữu hiệu theo nghĩa chương trình đòi hỏi quá nhiều bộ nhớ hoặc/và chạy quá lâu. Những thuật giải nảy sinh lập tức trong đầu như vậy thường được gọi là thuật giải tự nhiên. Dĩ nhiên, khái niệm này chỉ là tương đối. Nếu bạn đã nắm vững nhiều dạng thuật giải và đã từng thử sức với nhiều bài toán khó thì đến một lúc nào đó các thuật giải tự nhiên của bạn sẽ đáng tin cậy. Đó cũng chính là mục đích của sự học tập và rèn luyện và cũng là ước mơ của người viết tập sách này.

Để đọc sách không đòi hỏi bạn phải có tri thức gì đặc biệt. Để tiếp thu tốt và đóng góp cho việc hiệu chỉnh và cải tiến nội dung cuốn sách chỉ cần bạn biết sử dụng một trong các ngôn ngữ lập trình: Pascal trong môi trường Turbo hoặc Free Pascal hoặc C#.

Các kỹ thuật lập trình được minh họa qua những bài toán cụ thể tương đương với trình độ nâng cao của học sinh và sinh viên. Hình thức phát biểu bài toán suy cho cùng là không quan trọng. Các kỹ thuật lập trình và phương pháp xây dựng thuật giải cho những bài toán thường được dùng rộng rãi trong quá trình thiết kế và cài đặt các phần mềm ứng dụng trong thực tiễn, cho nên việc sớm làm chủ các tri thức này mới thật sự là cần thiết. Chính vì vậy mà chúng tôi cho rằng nội dung cuốn sách có thể phù hợp với các bạn học sinh, sinh viên các trường đại học và những bạn đọc muốn tự hoàn thiện tri thức trong lĩnh vực giải thuật và lập trình. Thiết nghĩ cuốn sách cũng có thể được dùng làm tài liệu tham khảo để dạy ở các lớp chuyên tin của các trường phổ thông. Nội dung sách gồm hai phần. Phần thứ nhất giới thiệu vắn tắt về bản chất các phương pháp và kỹ thuật lập trình và các đề toán để các bạn thử sức. Phần thứ hai trình bày và phân tích chi tiết lời giải cùng với những bình luận và xuất xứ của các bài toán.

Trong tập sách này cũng cung cấp toàn văn các chương trình viết bằng ngôn ngữ lập trình Pascal và C# để bạn đọc tiện so sánh với lời giải của mình. Cả hai phần đều đề cập đến nội dung của tám chương như sau.

Chương thứ nhất trình bày sơ đồ chung để giải một bài toán tin. Các bài tập ở chương này hầu hết thuộc loại dễ giải. Chương thứ hai giới thiệu các kỹ thuật sinh dữ liệu một cách tự động nhằm phục vụ cho việc kiểm thử (test) chương trình. Chương thứ ba trình bày các kỹ thuật quản lý bàn phím và màn hình. Chương thứ tư đề cập đến cách thức tổ chức dữ liệu cho một bài toán tin. Ba chương tiếp theo giới thiệu ba trong số các phương pháp khá phổ biến thường được vận dụng trong thiết kế thuật giải. Đó là phương pháp tham lam, phương pháp quay lui và quy hoạch động. Các phương pháp này đều là không vận năng theo nghĩa không thể dùng chúng để giải mọi bài toán tin. Trong thực

tế, một phương pháp vạn năng như vậy là không hữu hiệu. Tùy theo nội dung bài toán mà ta chọn phương pháp phù hợp. Đó cũng là điểm khó, đòi hỏi ở bạn đọc một quá trình tìm tòi và tích lũy kinh nghiệm.

Riêng chương cuối cùng của cuốn sách, chương thứ tám giới thiệu một số bài toán tin để bạn đọc tự phát hiện phương pháp giải.

Những nội dung trong tập sách này được tập hợp và chỉnh lí từ các bài giảng về thuật toán và lập trình, từ các cuốn sách Tìm đường trong mê cung, Bắn tàu trên biển và từ các bài viết của tác giả đăng trong tạp chí Tin học và nhà trường và một số lời giải hay của các bạn học sinh.

Lần xuất bản này chúng tôi trình bày thêm các bài giải viết trong môi trường ngôn ngữ C# để các bạn sinh viên cùng tham khảo. Hi vọng rằng trong các dịp khác chúng tôi sẽ cung cấp thêm các phương án giải với bạn đọc. Tuy nhiên, suy cho cùng, môi trường lập trình chỉ mang tính minh họa. Khi đã biết thuật toán, việc thể hiện thuật toán đó trong môi trường lập trình cụ thể chắc chắn là việc làm quen thuộc của bạn đọc.

Xin được chân thành cảm ơn các em học sinh, sinh viên, các thầy cô giáo, bạn bè và đồng nghiệp đã chia sẻ kinh nghiệm và trợ giúp tài liệu, nhận xét và bình luận để hình thành nội dung cơ bản của cuốn sách.

Chúng tôi hi vọng sẽ tiếp tục nhận được những ý kiến phê bình của bạn đọc về nội dung, chất lượng và hình thức trình bày để có thể định hướng cho các tập tiếp theo.

Hà Nội, Lễ Hội Đạp Thanh - 2008

N.X.H

CHƯƠNG 1

GIẢI MỘT BÀI TOÁN TIN

Phần này sẽ giới thiệu một số bước thường vận dụng trong quá trình giải các bài toán tin.

1. Bước đầu tiên và là bước quan trọng nhất là *hiểu rõ nội dung bài toán*. Đây là yêu cầu quen thuộc đối với những người làm toán. Để hiểu bài toán theo cách tiếp cận của tin học ta phải gắng xây dựng một số *thí dụ* phản ánh đúng các yêu cầu đề ra của đầu bài rồi thử giải các thí dụ đó để hình thành dần những *hướng đi* của thuật toán.
2. Bước thứ hai là dùng một ngôn ngữ quen thuộc, tốt nhất là ngôn ngữ toán học *đặc tả các đối tượng* cần xử lý ở mức độ trừu tượng, lập các tương quan, xây dựng các hệ thức thể hiện các quan hệ giữa các đại lượng cần xử lý.
3. Bước thứ ba là *xác định cấu trúc dữ liệu* để biểu diễn các đối tượng cần xử lý cho phù hợp với các thao tác của thuật toán. Trong những bước tiếp theo ta tiếp tục *làm mịn dần các đặc tả* theo trình tự từ trên xuống, từ trừu tượng đến cụ thể, từ đại thể đến chi tiết.
4. Bước cuối cùng là sử dụng ngôn ngữ lập trình đã chọn để viết *chương trình* hoàn chỉnh. Ở bước này ta tiến hành theo kỹ thuật đi từ dưới lên, từ những thao tác nhỏ đến các thao tác tổ hợp.

Sau khi nhận được chương trình ta cho chương trình chạy thử với các dữ liệu lấy từ các thí dụ đã xây dựng ở bước đầu tiên.

Điều quan trọng là *xây dựng các thủ tục* một cách khoa học và có chủ đích nhằm kiểm tra tính tin cậy của chương trình thu được và thực hiện một số cải tiến.

Chúng ta sẽ vận dụng cách tiếp cận trên để giải một số bài toán cụ thể. Những phần trình bày dưới đây có thể sử dụng một vài kí pháp quen thuộc của tin học, thí dụ:

$x = abc$ số tự nhiên x được tạo bởi ba chữ số a, b và c .
 $a, b = 0..9$ hai số a và b có thể nhận các giá trị từ 0 đến 9.

Sở dĩ ta không sử dụng các kí hiệu toán học vì trên bàn phím máy tính không có các kí hiệu đó. Chọn các kí hiệu có sẵn trong các ngôn ngữ lập trình giúp chúng ta có thể viết các chú thích ngay trong chương trình.

Bài 1.1. Số thân thiện

Tìm tất cả các số tự nhiên hai chữ số mà khi đảo trật tự của hai chữ số đó sẽ thu được một số nguyên tố cùng nhau với số đã cho.

Hiểu đầu bài

Ta kí hiệu (a, b) là ước chung lớn nhất (*ucln*) của hai số tự nhiên a và b . Hai số tự nhiên a và b được gọi là nguyên tố cùng nhau khi và chỉ khi $(a, b) = 1$. Khi đó, chẳng hạn:

a. $(23, 32) = 1$, vậy 23 là một số cần tìm. Theo tính chất *đối xứng*, ta có ngay 32 cũng là một số cần tìm.

b. $(12, 21) = 3$, vậy 12 và đồng thời 21 không phải là những số cần tìm.

Đặc tả: Gọi hai chữ số của số tự nhiên cần tìm x là a và b , ta có:

- (1) $x = ab$.
- (2) $a, b = 0..9$ (a và b biến thiên trong khoảng $0..9$).
- (3) $a > 0$ vì x là số có hai chữ số.
- (4) $(ab, ba) = 1$.

Ta kí hiệu x' là số đối xứng của số x theo nghĩa của đầu bài, khi đó ta có đặc tả như sau:

- (5) $x = 10..99$ (x biến thiên từ 10 đến 99, vì x là số có hai chữ số).
- (6) $(x, x') = 1$.

Nếu $x = ab$ thì $x' = ba$. Ta có thể tính giá trị của x' theo công thức:

$$x' = (\text{chữ số hàng đơn vị của } x) * 10 + (\text{chữ số hàng chục của } x).$$

Kí hiệu *Đơn*(x) là toán tử lấy chữ số hàng đơn vị của số tự nhiên x và kí hiệu *Chục*(x) là toán tử lấy chữ số hàng chục của x , ta có:

$$x' = \text{Đơn}(x) * 10 + \text{Chục}(x).$$

Tổng hợp lại ta có đặc tả:

Số cần tìm x phải thỏa các tính chất sau: $x = 10..99$ (x nằm trong khoảng từ 10 đến 99).

- (7) $x' = \text{Đơn}(x) * 10 + \text{Chục}(x)$.
- (8) $(x, x') = 1$ (ước chung lớn nhất của x và x' bằng 1).

Đặc tả trên được thể hiện qua ngôn ngữ phỏng trình tựa Pascal như sau:

```
(9) for x:=10 to 99 do
    if ucln(x, đơn(x)*10+Chục(x))=1 then Lấy(x);
```

trong đó, **ucln(a,b)** là hàm cho ước chung lớn nhất của hai số tự nhiên **a** và **b**; **Lấy(x)** là toán tử hiển thị **x** lên màn hình hoặc ghi **x** vào một mảng nào đó với mục đích sử dụng lại, nếu cần.

Ta làm mịn đặc tả (10):

uclid(a, b): Thuật toán Euclid là chia liên tiếp, thay số thứ nhất bằng dư của nó khi chia cho số thứ hai rồi hoán vị hai số.

```
(*-----*
  Tim uoc chung lon nhat cua hai so
    a va b. Thuat toan Euclid
-----*)
function Ucln(a,b: integer): integer;
```

```

var r: integer;
begin
  while b > 0 do
    begin
      r:= a mod b; a:= b; b:= r;
    end;
  Ucln:= a;
end;

```

Đơn(x) = (**x mod 10**): số dư của phép chia nguyên *x* cho 10, thí dụ:

Đơn(19) = 19 mod 10 = 9.

Chục(x) = (**x div 10**): thương nguyên của phép chia *x* cho 10, thí dụ:

Chục(19) = 19 div 10 = 1.

Lấy(x): **write(x)** hoặc nạp giá trị *x* vào mảng *s* theo các thao tác sau:

n := n + 1;

s[n] := x;

n đếm số phần tử hiện đã nạp trong mảng **s**.

Biểu diễn dữ liệu

Ta dùng mảng *s* để lưu các số tìm được. Dễ thấy *s* phải là một mảng nguyên chứa tối đa 90 phần tử vì các số cần khảo sát nằm trong khoảng từ 10 đến 99.

```
var s: array[1..90] of integer;
```

Phương án 1 của chương trình sẽ hoạt động theo hai bước như sau:

1. **n := Tim;**

2. **Xem(n);**

Bước 1. Tìm và ghi vào mảng **s** các số thoả điều kiện đầu bài, **n** là số lượng các số tìm được.

Bước 2. Hiện thị các phần tử của mảng **s[1..n]** chứa các số đã tìm được.

Toán tử *x'* được viết dưới dạng hàm cho ta số tạo bởi các chữ số của *x* theo trật tự ngược lại. Ta đặt tên cho hàm này là *SoDao* (số đảo). Hàm có thể nhận giá trị vào là một số tự nhiên có nhiều chữ số.

Để tạo số đảo *y* của số *x* cho trước, hàm *SoDao* lấy dần các chữ số hàng đơn vị của *x* để ghép vào bên phải số *y*:

y := y*10 + (x mod 10)

Sau mỗi bước, chữ số hàng đơn vị đã lấy được loại hẳn khỏi *x* bằng toán tử:

x := x div 10

Chỉ thị **{\$B-}** trong chương trình NTCN (nguyên tố cùng nhau) dưới đây đặt chế độ kiểm tra biểu thức logic vừa đủ. Khi đã xác định được giá trị chân lí cần thiết thì không tiến hành tính tiếp giá trị của biểu thức đó nữa. Thí dụ, với các lệnh

x := 1; y := 5;

if (x > 5) and (x + y < 7) then y := y + 1

else y := y-1;

trong chế độ **{\$B-}**, sau khi tính được giá trị chân lí **(x > 5) = false**, chương trình sẽ bỏ qua nhân tử logic **(x + y < 7)**, vì tích logic của false với giá trị tùy ý cho ta false. Trong trường hợp này lệnh **y := y - 1** sẽ được thực hiện. Ngược lại, nếu ta đặt chỉ thị **{\$B+}** thì chương trình, sau khi tính được **(x > 5) = false** vẫn tiếp tục tính giá trị của **(x + y < 7)** rồi lấy tích của hai giá trị tìm được (**false and true = false**) làm giá trị của biểu thức điều kiện trong cấu trúc rẽ nhánh nói

trên. Cuối cùng toán tử $y := y - 1$ cũng được thực hiện giống như trường hợp trên nhưng khối lượng tính toán lại nhiều hơn.

(* Pascal *)

```
(*-----*)
      So than thien (xy,yx) = 1
-----*)
program SoThanThien;
{$B-}
uses Crt;
const MN = 90;
var s: array[1..MN] of integer;
function Ucln(a,b: integer): integer; tự viết
function SoDao(x: integer): integer;
var y: integer;
begin
  y := 0;
  repeat
    { ghép chu so hang don cua x vao ben phai y }
    y := 10*y + (x mod 10);
    x := x div 10; { loai chu so hang don }
  until (x = 0);
  SoDao := y;
end;
(*-----*)
      Tim cac so thoa dieu kien dau bai
      ghi vao mang s.
      Output: so luong cac so tim duoc
-----*)
function Tim: integer;
var x,d: integer;
begin
  d := 0; {So luong cac so can tim }
  for x := 10 to 99 do
    if Ucln(x,SoDao(x)) = 1 then
      begin
        d := d + 1; s[d] := x;
      end;
  Tim := d;
end;
(*-----*)
      Hien thi mang s[1..n] tren man hinh.
-----*)
procedure Xem(n: integer);
var i: integer;
begin
  writeln;
  for i := 1 to n do write(s[i]:4);
  writeln;
end;
BEGIN
  n := Tim; Xem(n); writeln;
```

```

        write(' Tong cong ',n,' so'); readln;
    END.

// C#
using System;
namespace SangTao1
{
    /*****
        So Than Thien: (xy, yx) = 1
        *****/
    class SoThanThien
    {
        static int mn = 90;
        static int [] s = new int[mn];
        static void Main(string[] args)
        {
            Run();
            Console.ReadLine();
        }
        static void Run()
        {
            int n = Find();
            for (int i=0;i<n;++i)
                Console.Write(s[i] + " ");
            Console.WriteLine("\n Tong cong: "+n+" so");
        }
        static int Find()
        {
            int d = 0;
            for (int x = 10; x < 100; ++x)
                if (Ucln(x,SoDao(x))==1) s[d++] = x;
            return d;
        }
        static int Ucln(int a, int {}b)
        {
            int r;
            while (b != 0){ r = a%b;a = b;b = r; }
            return a;
        }
        static int SoDao(int x)
        {
            int y = 0;
            do { y = y*10+(x%10); x /= 10; } while (x!=0);
            return y;
        }
    } // SoThanThien
} // SangTao1

```

Cải tiến

Ta vận dụng tính đối xứng đã nhận xét ở phần trên để cải tiến chương trình. Như vậy chỉ cần khảo sát các số $x = ab$, với $a > b \geq 0$. Trường hợp $a = b$ ta không xét vì khi đó $x' = x$ và do đó $Ucln(x, x) = x \geq 10 \neq 1$.

Nếu $b = 0$ ta có $x = 10a$ và $x' = a$. Ta thấy $Ucln(10a, a) = a = 1$ khi và chỉ khi $a = 1$. Do đó ta xét riêng trường hợp này. Khi $ab = 10$ ta có $(10, 1) = 1$. Vậy 10 chính là một số cần tìm và là số đầu tiên.

Mỗi khi tìm được hai chữ số a và b thỏa điều kiện $a > b$ và $\text{Ucln}(a*10 + b, b*10 + a) = 1$ ta đưa $a*10 + b$ vào kết quả, nếu $b > 0$ ta đưa thêm số đảo $b*10 + a$ vào kết quả.

(* Pascal *)

```
(*-----
      So Than thien: Phuong an 2
-----*)
function Tim2: integer;
var a,b,d: integer;
begin
  d:= 1; {So luong cac so can tim}
  s[d] := 10;
  for a := 1 to 9 do
    for b := 1 to a-1 do
      if Ucln(a*10+b,b*10+a)=1 then
        begin
          d := d + 1; s[d] := a*10 + b;
          d := d + 1; s[d] := b*10 + a;
        end;
  Tim2 := d;
end;
```

// C#

```
// Phuong an 2
static int Find2()
{ int a,b, d = 0;
  s[d++] = 10;
  for (a = 1; a <= 9; ++a)
    for (b = 1; b < a; ++b)
      if (Ucln(10*a + b, 10*b + a) == 1)
        { s[d++]=10*a+b; s[d++]=10*b+a; }
  return d;
}
```

Bài 1.2. Số cấp cộng

Tìm các số tự nhiên lẻ có ba chữ số. Ba chữ số này, theo trật tự từ trái qua phải tạo thành một cấp số cộng.

Đặc tả

1. x là số tự nhiên có ba chữ số: $x = 100*a + 10*b + c$.
2. x là số lẻ nên chữ số hàng đơn vị c phải là số lẻ: $c = 1, 3, 5, 7, 9$.
3. Chữ số hàng trăm của x phải khác 0: $a = 1..9$.
4. Nếu dãy a, b, c lập thành một cấp số cộng thì số đứng giữa b là trung bình cộng của hai số đầu và cuối: $b = (a + c)/2$ hay $2b = a + c$.

Từ (4) ta suy ra $(a + c)$ là số chẵn. Do c lẻ, $(a + c)$ chẵn nên a lẻ.

Nếu biết a và c ta tính được $x = 100a + 10(a + c) / 2 + c$

$$= 100a + 5(a + c) + c = 105a + 6c.$$

Vì chỉ có 5 chữ số lẻ là 1, 3, 5, 7 và 9 nên tổ hợp của a và c sẽ cho ta 25 số.

Tổ chức dữ liệu

Ta tạo sẵn mảng nguyên 5 phần tử **ChuSoLe**[1..5] và gán trước các giá trị 1, 3, 5, 7, 9 cho mảng này. Trong Turbo Pascal (TP) việc này được thực hiện thông qua khai báo:

```
const ChuSoLe: array[1..5] of integer = (1,3,5,7,9);
```

Chú ý rằng khai báo này phải đặt trong mục **const** là nơi khai báo hằng.

Trong C# ta khai báo như sau:

```
int [] ChuSoLe = {1,3,5,7,9};
```

Ý nghĩa của dòng khai báo trên là như sau: Xin cấp phát một biến mảng kiểu nguyên có 5 phần tử với chỉ dẫn từ 1 đến 5, tên biến là **ChuSoLe**. 5 phần tử của biến được gán trước các trị 1, 3, 5, 7 và 9.

Sau đó, mỗi khi cần, ta chỉ việc duyệt mảng **ChuSoLe** là thu được toàn bộ các chữ số lẻ theo trật tự đã khai báo trước.

Chú ý

Thủ tục **inc(d)** trong chương trình TP dưới đây tăng giá trị của biến **d** lên thêm 1 đơn vị, tức là tương đương với câu lệnh **d := d + 1** và **++d (C#)**. Tương tự, thủ tục **dec(d)** sẽ giảm giá trị của biến **d** xuống 1 đơn vị, tương đương với câu lệnh **d := d - 1** và **--d (C#)**.

Tổng quát hơn, ta có thể viết:

inc(d,n) tương đương với **d := d + n** và

dec(d,n) tương đương với **d := d - n**.

Khi **n = 1** thì có thể bỏ qua tham số thứ hai.

(* Pascal *)

```
(-----
      Cac so tu nhien le 3 chu so
      lap thanh cap so cong
-----*)
program CapCong;
uses crt;
const
ChuSoLe: array [1..5] of integer = (1,3,5,7,9);
var s: array [1..25] of integer;
    n: integer;
(*-----
      Phat sinh cac so dang
      105a+6c; a,c = 1,3,5,7,9
-----*)
Function Tim: integer;
var a,c,d,x: integer;
begin
  d := 0;
  for a := 1 to 5 do
  begin
    x := 105*ChuSoLe[a];
    for c := 1 to 5 do
    begin
      inc(d); s[d] := x + 6*ChuSoLe[c];
    end;
  end;
  Tim := d;
end;
```

```

(*-----
  Hien thi mang s[1..n] moi dong 20 so
-----*)
procedure Xem(n: integer); tự viết
BEGIN
  n := Tim; Xem(n); writeln;
  write('Tong cong ',n,' so'); readln;
END.

// C#
using System;
namespace SangTao1
{
  class SoCapCong
  {
    static void Main(string[] args)
    {
      Show(Find());
      Console.WriteLine("\n fini");
      Console.ReadLine();
    }
    static int[] Find()
    {
      int d = 0;
      int [] ChuSoLe = {1,3,5,7,9};
      int []s = new int[25];
      int x;
      for (int i = 0; i < 5; ++i)
      {
        x = 105 * ChuSoLe[i];
        for (int j = 0; j < 5; ++j)
          s[d++] = x + 6 * ChuSoLe[j];
      }
      return s;
    }
    static void Show(int[] s)
    {
      foreach (int x in s)
        Console.Write(x + " ");
    }
  } // SoCapCong
} // SangTao1

```

Chú thích

1. Trong **C#** một hàm có thể cho ra giá trị là một mảng như hàm **Find** trong chương trình trên.
2. Lệnh **foreach (int x in a) P(x)** thực hiện thao tác **P(x)** trên mọi phần tử **x** của mảng, từ phần tử đầu tiên **a[0]** đến phần tử cuối cùng **a[a.Length]** với **a.Length** là chiều dài (số phần tử) của mảng **a**.

Chú ý

1. Dựa vào nhận xét: dãy ba số a, b, c tạo thành cấp số cộng khi và chỉ khi b là trung bình cộng của a và c , tức là $2b = a + c$ ta có thể giải bài toán trên bằng phương pháp vét cạn dùng ba vòng **for** như sau:

```
for a := 1 to 9 do
  for b := 0 to 9 do
    for c := 0 to 9 do
      if odd(c) and (2*b=a+c) then
        Ghi nhận số 100*a+10*b+c;
```

Hàm **odd(c)** kiểm tra tính lẻ của số nguyên c .

Phương pháp vét cạn đòi hỏi khoảng $10 \times 10 \times 10 = 1000$ lần duyệt trong khi chỉ có 25 số, tức là một phần bốn mươi các số thoả mãn điều kiện của đầu bài. Phương pháp mô tả trong chương trình được gọi là *phương pháp sinh*: nó sinh ra đúng 25 số cần tìm.

2. Ta cần ghi nhận *phương pháp sinh*

Phương pháp sinh

*Thay vì duyệt tìm các đối tượng
hãy sinh ra chúng.*

Bài 1.3. Số cấp nhân

Tìm các số tự nhiên có ba chữ số. Ba chữ số này, theo trật tự từ trái qua phải tạo thành một cấp số nhân với công bội là một số tự nhiên khác 0.

Đặc tả

Chú ý rằng ta chỉ xét các cấp số trên dãy số tự nhiên với công bội d là một số nguyên dương. Gọi x là số cần tìm, ta có:

$$1. \ x \text{ là số có ba chữ số: } x = 100*a + 10*b + c.$$

$$2. \ a = 1..9; b = a*d; 0 < c = a*d*d \leq 9.$$

Hệ thức 2 cho phép ta tính giới hạn trên của d :

$$ad^2 \leq 9$$

$$d \leq \sqrt{9/a}$$

Vì d là số nguyên nên ta phải có $d \leq \text{trunc}(\text{sqrt}(9 \text{ div } a))$, trong đó **sqrt** là hàm tính căn bậc hai, **trunc** là hàm lấy phần nguyên.

Ta cho a biến thiên trong khoảng $1..9$ rồi cho công bội d biến thiên trong khoảng từ 1 đến **trunc(sqrt(9 div a))**. Với mỗi cặp số a và d ta tính

$$x = 100*a + 10*a*d + a*d*d = a*(100 + 10*d + d*d)$$

Tuy nhiên, ta có thể nhẩm tính trước cận trên của d thì sẽ đỡ phải gọi các hàm **trunc** và **sqrt** là những hàm thao tác trên số thực do đó sẽ tốn thời gian.

a	1	2	3	4	5	6	7	8	9
Cận trên d	3	2	1	1	1	1	1	1	1

(* Pascal *)

(*-----*)

Cac so tu nhien 3 chu so
lap thanh cap nhan

```

-----*)
program CapNhan;
uses crt;
const MN = 30;
      cd: array[1..9] = (3,2,1,1,1,1,1,1,1);
var s: array [1..MN] of integer;
      n: integer;
function Tim: integer;
var a,d,n: integer;
begin
  n:= 0;
  for a:= 1 to 9 do
    for d:=1 to cd[a]do
      begin
        inc(n); s[n]:= a*(100+10*d+d*d);
      end;
    Tim:= n;
  end;
procedure Xem(n: integer): tự viết
BEGIN
  clrscr; n:= Tim; Xem(n);
  writeln; write('Tong cong ',n,' so'); readln;
END.

// C#
using System;
using System.Collections;
namespace SangTaol
{
  class SoCapNhan
  {
    static void Main(string[] args)
    {
      Show(Find());
      Console.WriteLine("\n fini");
      Console.ReadLine();
    }
    static ArrayList Find()
    {
      ArrayList s = new ArrayList();
      int[] cd = {0,3,2,1,1,1,1,1,1};
      for (int a = 1; a <= 9; ++a)
      {
        for (int d = 1; d <= cd[a]; ++d)
          s.Add(a * (100 + 10 * d + d * d));
      }
      return s;
    }
    static void Show(ArrayList s) tự viết
  } // SoCapNhan
} SangTaol

```

Chú thích

- Trong **C#** một hàm có thể cho ra giá trị là một mảng - danh sách kiểu **ArrayList** như hàm **Find** trong chương trình.
- Khi không biết có bao nhiêu phần tử được sinh ra trong quá trình tìm kiếm thì nên dùng kiểu mảng - danh sách để chứa kết quả.
- Mảng **cd** chứa các cận của d ứng với mỗi trị của a = 1..9, ta thêm cho **cd** phần tử 0 để tiện truy nhập.

Bài 1.4. Mảng ngẫu nhiên

Sinh ngẫu nhiên n số nguyên không âm cho mảng nguyên a.

Đặc tả

Trong **TP** hàm **random(n)** sinh một số ngẫu nhiên kiểu nguyên nằm trong khoảng từ 0 đến $n - 1$. Hãy tưởng tượng có một quân súc sắc n mặt mã số các mặt từ 0 đến $n - 1$. Khi ta gọi hàm **random(n)** thì máy tính sẽ gieo quân súc sắc đó và cho ta giá trị xuất hiện trên mặt ngửa.

Trong **C#** phương thức **Next(n)** của lớp **Random** hoạt động tương tự như **random(n)** của **TP**.

Chú ý

1. Trước khi gọi hàm **random** ta cần gọi thủ tục **randomize** để máy tính khởi động cơ chế phát sinh số ngẫu nhiên.
2. Thủ tục **Gen(m)** trong chương trình dưới đây sinh ngẫu nhiên m số nguyên trong khoảng từ 0 đến $m - 1$. Ta có thể cải tiến để viết thủ tục **Gen(n, d, c)** - sinh ngẫu nhiên n số nguyên trong khoảng từ d đến c ($d < c$) như sau.
Đề ý rằng **random(c-d+1)** biến thiên trong khoảng từ 0 đến $c-d$, do đó **d+random(c-d+1)** sẽ biến thiên trong khoảng từ d đến $d+c-d = c$.

(* Pascal *)

```
program RandomGen;
(*-----
    Sinh ngau nhien n so nguyen
    khong am cho mang a
    ----- *)
{$B-}
uses crt;
const MN = 500;
var
    a: array [1..MN] of integer;
    n: integer;
    Procedure Gen(m: integer);
    var i: integer;
    begin
        randomize; n := m;
        for i := 1 to n do a[i] := random(m);
    end;
    procedure Xem: tự viết;
    BEGIN
        Gen(200); Xem;
    END.
```

// C#

```

using System;
namespace SangTao1
{
    class RandomGen
    {
        static void Main(string[] args)
        {
            Show(Gen(200));
            Console.WriteLine("\n Fini ");
            Console.ReadLine();
        }
        static int [] Gen(int n)
        {
            int [] a = new int[n];
            Random r = new Random();
            for (int i = 0; i < n; ++i)
                a[i] = r.Next(n);
            return a;
        }
        static void Show(int [] s): tự viết
    } // RandomGen
} // SangTao1

```

Bài 1.5. Chia mảng tỉ lệ 1:1

Tìm cách chia dãy số nguyên không âm a_1, a_2, \dots, a_n , $n > 1$ cho trước thành hai đoạn có tổng các phần tử trong mỗi đoạn bằng nhau.

Đặc tả

Ta quy ước viết #E là "tồn tại" và #V là "với mọi". Kí hiệu $\text{sum}(a[d..c])$ là tổng các phần tử liên tiếp nhau từ $a[d]$ đến $a[c]$ của dãy a :

$$\text{sum}(a[d..c]) = a[d] + a[d+1] + \dots + a[c].$$

Gọi t là tổng các phần tử của mảng: $t = \text{sum}(a[1..n])$.

Muốn chia a thành hai đoạn $a[1..i]$ và $a[i+1..n]$ có tổng bằng nhau ta phải có:

1. t là số chẵn (t chia hết cho 2). Đặt $t2 = t \text{ div } 2$.
2. (#E i : $1 \leq i \leq n$): $\text{sum}(a[1..i]) = t2$.

Chương trình

Hàm **Chia** cho giá trị i nếu mảng a chia được thành $a[1..i]$ và $a[i+1..n]$. Trong trường hợp vô nghiệm **Chia** = -1. Ta gọi i là điểm chia và dùng biến **tr** (tổng riêng) để tích lũy tổng các phần tử của đoạn đang xét $a[1..i]$. Khi **tr** = **t2** bài toán có nghiệm i . Ngược lại, khi **tr** > **t2** bài toán vô nghiệm.

Ta khởi trị ngẫu nhiên cho mảng a . Tuy nhiên ta muốn số lần có nghiệm (mảng a chia được thành hai phần có tổng bằng nhau) xấp xỉ bằng số lần vô nghiệm. Ta sẽ thực hiện mục tiêu đề ra như sau:

Mỗi lần khởi trị ta tung đồng xu hai mặt. Nếu gặp mặt sấp (**random**(2)=0), ta sẽ khởi trị tùy ý cho mảng a , ngược lại, nếu gặp mặt ngửa (**random**(2)=1) ta khởi trị a là mảng có nghiệm.

Để khởi trị sao cho mảng a có nghiệm ta lại chọn ngẫu nhiên một điểm cắt d trong khoảng $1..(n/2)$. Sau đó ta khởi trị ngẫu nhiên cho các phần tử $a[1..d]$. Với các phần tử còn lại ta cũng khởi trị ngẫu nhiên trong khoảng hợp lí sao cho tổng các giá trị

của chúng đúng bằng tổng t của đoạn $a[1..d]$. Bạn đọc xem chi tiết thủ tục **Gen** trong chương trình.

(* **Pascal** *)

```
(*-----
          Chia mảng nguyên a thành 2 đoạn
          có tổng bằng nhau
----- *)
program ChiaTiLe11;
{$B-}
uses crt;
const MN = 500; Esc = #27;
var  a: array [1..MN] of integer;
     n: integer;
(*-----
          Sinh ngẫu nhiên n số nguyên không âm
          cho mảng a
----- *)
procedure Gen(m: integer);
  var i,d,t: integer;
begin
  randomize;  n := m;
  if random(2)=0 then
  begin {khởi tạo tùy ý}
    for i := 1 to n do a[i]:=random(m);
    exit;
  end;
  { Khởi tạo mảng có tổng d phần tử đầu
    bằng tổng các phần tử còn lại }
  d := random(n div 2)+ 1; { điểm chia }
  t := 0;
  for i := 1 to d do
  begin
    a[i] := random(n);
    t := t + a[i];
  end; { t = sum(a[1..d]) }
  for i := d+1 to n-1 do
  begin { sum(a[d+1..i]) + t = sum(a[1..d]) }
    a[i] := random(t);
    t := t-a[i];
  end;
  a[n] := t; { sum(a[1..d]) = sum(a[d+1..n]) }
end;
procedure Xem: Hiển thị mảng a, tự viết
function Chia: integer;
var i, t, t2, tr: integer;
begin
  Chia := -1;  t := 0;
  for i:=1 to n do t:=t+a[i]; {t=sum(a[1..n])}
  if Odd(t) then exit; { vô nghiệm }
  t2 := t div 2;  tr := 0;
  for i:=1 to n do
  begin
```

```

        tr := tr + a[i];
        if tr > t2 then exit; {vo nghiem }
        if tr = t2 then { co nghiem i }
            begin Chia:= i; exit; end;
        end;
    end;
procedure Test;
var i: integer;
begin
    repeat
        Gen(10); Xem; i := Chia;
        if i = -1 then writeln('Khong chia duoc')
        else
            begin
                writeln('Doan thu nhat: a[1..',i,']');
                writeln('Doan thu hai: a['',i+1,'...',n,']');
            end;
        until ReadKey=Esc;
    end;
BEGIN
    Test;
END.

```

Chú ý

1. Muốn dừng chương trình hãy nhấn phím ESC có mã ASCII là #27.
2. Nếu mảng a có chứa một số giá trị 0 thì bài toán có thể có nhiều nghiệm (nhiều cách chia).

// C#

```

using System;
namespace SangTao1
{
    class ChiaMangTiLe1_1
    {
        static void Main()
        {
            do {
                Run(20);
                Console.Write("\n Bam phim ENTER " +
                    "de tiep tuc, ");
                Console.Write("\n Bam phim T de thoat: ");
            } while (Console.ReadLine() == "");
        }
        static public void Run(int n)
        {
            int[] a = new int[n];
            Gen(a, n); // sinh ngau nhien 1 test
            Print(a, n);
            int t = 0, d = Chia(a, n, ref t);
            if (d < 0)
                Console.WriteLine("\n Khong chia duoc");
            else if (KiemTra(a, n, d))
            { Console.WriteLine("\n Doan thu nhat: 1..{0} ",d);
              Console.WriteLine("\n Doan thu hai: {0}..{1} ",

```

```

        d+1, n);
    Console.WriteLine("\n Tong moi doan: " + t);
}
else Console.WriteLine("\n Loi giai sai!");
} // end Run
// Kiem tra sum(a[1..d] == sum(a[d+1..n]) ?
static public bool KiemTra(int[] a, int n, int d)
{ if (d < 0 || d >= n) return false;
  int t = 0;
  for (int i = 0; i < d; ++i) t += a[i];
  for (int i = d; i < n; ++i) t -= a[i];
  return (t == 0) ? true : false;
}
static public int Chia(int[] a, int n, ref int t)
{ int sum = 0; // sum = tong(a[1..n])
  for (int i = 0; i < n; ++i) sum += a[i];
  if (sum % 2 != 0) return -1;
  t = sum / 2; // tong moi doan
  int tr = 0; // tong rieng
  // doan 1: tr = sum a[1..i]
  for (int i = 0; i < n; ++i)
  { tr += a[i];
    if (tr == t) return i+1;
  }
  return -1;
}
// sinh ngau nhien n so ghi vao mang a
static public void Gen(int[] a, int n)
{
    Random r = new Random();
    if (r.Next(2) == 0)
    { // 1/2 so test la vo nghiem
        for (int i = 0; i < n; ++i) a[i]=r.Next(n);
        return;
    }
    // sinh mang a: sum(a[0..d-1])=sum(a[d..n-1])
    int d = r.Next(n / 2) + 1; // diem chia
    int t = 0;
    // sinh doan a[0..d-1]
    for (int i = 0; i < d; ++i)
    { a[i] = r.Next(n); t += a[i]; }
    // sinh tiep doan a[d..n-1]
    int n1 = n-1;
    for (int i = d; i < n1; ++i)
    { a[i] = r.Next(t); t -= a[i]; }
    a[n-1] = t; // phan tu cuoi
}
static public void Print(int[] a, int n): tự viết
} // SoCapNhan
} // SangTao1

```

Bài 1.6. Chia mảng tỉ lệ 1:k

Tìm cách chia dãy số nguyên không âm a_1, a_2, \dots, a_n , $n > 1$ cho trước thành hai đoạn có tổng các phần tử trong một đoạn gấp k lần tổng các phần tử trong đoạn kia, k nguyên dương.

Đặc tả

Gọi t là tổng các phần tử của dãy a , $t = \text{sum}(a[1..n])$

Muốn chia a thành hai đoạn $a[1..i]$ và $a[i+1..n]$ có tổng gấp nhau k lần ta phải có:

1. t chia hết cho $(k+1)$. Đặt $t1 = t \text{ div } (k+1)$ và $tk = t - t1$.
2. ($\#E$ $i: 1 \leq i \leq n$): $\text{sum}(a[1..i]) = t1$ hoặc $\text{sum}(a[1..i]) = tk$.

Đề ý rằng nếu $k = 1$ thì $t1 = tk$; nếu $k > 1$ thì $t1 < tk$, do đó bài này là trường hợp riêng của bài trước khi $k = 1$.

Trong chương trình dưới đây, hàm Chia(k) cho giá trị i nếu mảng a chia được thành hai đoạn $a[1..i]$ và $a[(i+1)..n]$ có tổng gấp k lần nhau. Trong trường hợp vô nghiệm Chia = -1. Ta gọi i là điểm chia và dùng biến tr (tổng riêng) để tích lũy tổng các phần tử của đoạn đang xét $a[1..i]$. Khi $tr = t1$ bài toán có nghiệm I , ngược lại, khi $tr > t1$ ta chưa thể kết luận là bài toán vô nghiệm. Trường hợp này ta phải tiếp tục tích lũy tr để hi vọng đạt được tổng $tr = tk$. Nếu sau khi tích lũy ta thu được $tr = tk$ thì bài toán có nghiệm i , ngược lại, khi $tr > tk$ ta kết luận là bài toán vô nghiệm.

```
Function Chia(n,k: integer): integer;
var i: integer;
    t, t1, tk, tr: longint;
begin
    Chia := -1;
    t := 0; { t = sum(a[1..n]) }
    for i := 1 to n do t := t+a[i];
    if (t mod (k+1) <> 0) then exit; { vô nghiệm }
    { Xu li trường hợp có nghiệm }
    t1 := t div (k+1); { đoạn tổng nhỏ }
    tk := t - t1; { tk = k * t1 }
    tr := 0; { tổng riêng tr = sum(a[1..i]) }
    for i := 1 to n do
        begin
            tr := tr + a[i];
            if (tr = t1) or (tr = tk) then
                begin { lay nghiệm i }
                    Chia:= i; exit;
                end;
        end;
    end;
```

Ta gọi thủ tục Gen để sinh dữ liệu kiểm thử. Cũng giống như bài trước, ta sẽ sinh ngẫu nhiên dữ liệu kiểm thử cho hai trường hợp: chắc chắn có nghiệm và có thể vô nghiệm. Với trường hợp có thể vô nghiệm ta sinh ngẫu nhiên như bình thường,

```
for i := 1 to n do a[i] := random(n);
```

Với trường hợp có nghiệm, ta sinh ngẫu nhiên mảng a gồm hai đoạn:

Đoạn thứ nhất $a[1..d]$ và đoạn thứ hai $a[d+1..n]$ trong đó d là một điểm chia được sinh ngẫu nhiên

```
d := random(n div 2)+1; {diem chia}
```

Ta lại chọn ngẫu nhiên một trong hai trường hợp:

Trường hợp thứ nhất: đoạn thứ nhất gấp k lần đoạn thứ hai.
 Trường hợp thứ hai: đoạn thứ hai gấp k lần đoạn thứ nhất.

(* Pascal *)

```
(*-----
      Chia mảng nguyên a thành 2 đoạn
      có tổng tỉ lệ 1:k
----- *)
{$B-}
uses Crt;
const MN = 500;
      Esc = #27; { dấu thoát }
      bl = #32; { dấu cách }
      nl = #13#10; { xuống dòng }
var a: array [0..MN] of integer;
    n: integer;

(*-----
Sinh ngẫu nhiên n số nguyên không âm cho mảng a
----- *)
Procedure Gen(m,k: integer);
  var i,d: integer; t: longint;
begin
  n := m; t := 0;
  if random(2) = 0 then { thử nghiệm }
  begin
    for i := 1 to n do a[i] := random(n);
    exit;
  end;
  { thử nghiệm }
  d := random(n div 2)+1; { điểm chia }
  for i := 1 to d do
  begin
    a[i] := random(n); t := t+a[i];
  end;
  if (random(2) = 0) then
  { đoạn a[1..d] gấp k lần đoạn cuối }
    a[d] := a[d]+(k-1)*t
  else { đoạn cuối gấp k lần đoạn a[1..d] }
    t := k*t;
  for i := d+1 to n-1 do
  begin
    a[i] := random(t); t := t-a[i];
  end;
  a[n] := t;
end;
Procedure Xem; Hiển thị mảng a, tự viết
Function Chia(n,k: integer): integer; Tự viết
Procedure Test;
  var j,i,k: integer; t: longint;
  begin
    randomize;
    repeat
```

```

n := 10 + random(10);
k := random(5)+1;
writeln(nl, ' n = ', n, ' k = ', k);
Gen(n,k); Xem; i := Chia(n,k);
if i < 0 then writeln('Khong chia duoc')
else
begin
  t := 0;
  for j := 1 to i do t := t+a[j];
  write('Doan 1: a[1..', i, '].');
  writeln(' Tong = ', t);
  t := 0;
  for j:=i+1 to n do t := t+a[j];
  write('Doan 2: a[' , i+1, '...', n, '].');
  writeln(' Tong = ', t);
end;
until ReadKey = Esc;
end;
BEGIN
  Test;
END.

```

// C#

```

using System;
using System.Collections.Generic;
using System.Text;
namespace SangTao1
{
  /*-----
  *      Chia Mang Ti Le 1:k
  * Chia mang nguyen khomng am a[1..] thanh
  *      hai doan ti le 1:k hoac k:1
  * -----*/
  class ChiaMangTiLe1_k
  {
    static void Main(string[] args)
    {
      do
      {
        Run(10, 3);
        Console.Write("\n Bam RETURN de tiep tục, ");
        Console.Write("\n Bam T de thoát: ");
      } while (Console.ReadLine() != "T");
    }
    static public void Run(int n, int k)
    {
      if (n < 0 || n > 1000000 || k < 1) return;
      int[] a = Gen(n, k);
      Print(a);
      int d = Chia(a, k);
      if (d < 0)
      {

```

```

        Console.WriteLine("\n Vo nghiem");
        return;
    }
    Console.WriteLine("\n "+ Test(a, d, k));
}
// Kiem tra k*Sum(a[1..d]) = Sum(a[d+1..n]) ?
// hoac Sum(a[1..d]) = k*Sum(a[d+1..n])
static public bool Test(int[] a, int d, int k)
{
    Console.WriteLine("\n\n Test, k = " + k);
    Console.WriteLine("    Diem Chia = " + d);
    int t1 = 0;
    for (int i = 0; i < d; ++i) t1 += a[i];
    int t2 = 0;
    for (int i = d; i < a.Length; ++i) t2 += a[i];
    Console.WriteLine("Sum1 = {0}, Sum2 = {1}",
        t1, t2);
    return (t1 == k * t2 || t2 == k * t1);
}
static public int Chia(int[] a, int k)
{
    int t = 0;
    foreach (int x in a) t += x;
    if (t % (k + 1) != 0) return -1;
    int t1 = t / (k + 1); // tong 1phan chia
    int t2 = t - t1; // tong phan con lai
    int tr = 0; // tong rieng
    for (int i = 0; i < a.Length; ++i)
    {
        tr += a[i];
        if (tr == t1 || tr == t2) return i+1;
    }
    return -1;
}
static public int[] Gen(int n, int k)
{
    Random r = new Random();
    int[] a = new int[n];
    if (r.Next(2) == 0)
    { // khoang 1/2 so test la vo nghiem
        for (int i = 0; i < n; ++i)
            a[i] = r.Next(n);
        return a;
    }
    int d = r.Next(n / 2) + 1; //diem chia
    int t = 0;
    int d1 = d - 1;
    for (int i = 0; i < d1; ++i)
        { a[i] = r.Next(n); t += a[i]; }
    if (r.Next(2) == 0)
        // doan dau a[1..d]
        // gap k lan doan cuoi a[d+1..n]
        a[d1] += (k - 1) * t;
}

```

```
        else t *= k; // doan cuoi gap k lan doan dau
        int n1 = n - 1;
        for (int i = d; i < n1; ++i)
            { a[i] = r.Next(t); t -= a[i]; }
        a[n1] = t;
        return a;
    }
    static public void Print(int[] a) tự viết
} // ChiaMangTiLe1_k
} // SangTao1
```

CHƯƠNG 2

SINH DỮ LIỆU VÀO VÀ RA

Hầu hết các bài toán tin đều đòi hỏi *dữ liệu vào và ra*. Người ta thường dùng ba phương thức sinh và nạp dữ liệu sau đây:

1. *Nạp dữ liệu trực tiếp từ bàn phím*. Phương thức này được dùng khi dữ liệu không nhiều.

2. *Sinh dữ liệu nhờ hàm random* (xem chương 1). Phương thức này nhanh chóng và tiện lợi, nếu khéo tổ chức có thể sinh ngẫu nhiên được các dữ liệu đáp ứng được một số điều kiện định trước.

3. *Đọc dữ liệu từ một tệp, thường là tệp văn bản*. Phương thức này khá tiện lợi khi phải chuẩn bị trước những tệp dữ liệu phức tạp.

Kết quả thực hiện chương trình cũng thường được thông báo trực tiếp trên màn hình hoặc ghi vào một tệp văn bản.

Bài 2.1. Sinh ngẫu nhiên theo khoảng

Sinh ngẫu nhiên cho mảng nguyên a n phần tử trong khoảng $-M..M$; $M > 0$.

Đặc tả

Ta viết thủ tục tổng quát **Gen(n,d,c)** - sinh ngẫu nhiên n số nguyên trong khoảng từ d đến c ($d < c$) (xem bài giải 1.4). Để giải bài 2.1 ta chỉ cần gọi **Gen(n,-M,M)**.

Để ý rằng **random(c-d+1)** biến thiên trong khoảng từ 0 đến c-d do đó **d+random(c-d+1)** sẽ biến thiên trong khoảng từ d đến **d+c-d = c**.

```
(*-----
sinh ngẫu nhiên n số nguyên trong khoảng
d đến c và ghi vào mảng a
-----*)

Procedure Gen(n,d,c: integer);
var i,len: integer;
begin
  randomize;
  len := c-d+1;
  for i:=1 to n do a[i]:= d+random(len);
end;
```

(* Pascal *)

```

(*-----
      Sinh ngẫu nhiên cho mảng nguyên a
      n phần tử trong khoảng -M..M; M > 0.
-----*)
program RGen;
uses crt;
const MN = 1000;
var a: array[1..MN] of integer;
(*-----
      sinh ngẫu nhiên n số nguyên trong khoảng
      d đến c và ghi vào mảng a
----- *)
Procedure Gen(n,d,c: integer); tự viết
procedure Xem(n: integer); Hiển thị mảng a, tự viết
procedure Test;
var n: integer;
begin
  n := 20;
  { sinh ngẫu nhiên 20 số trong khoảng -8..8 }
  Gen(n,-8,8);
  Xem(n);
  readln;
end;
BEGIN
  Test;
END.

```

// C#

```

using System;
using System.Collections.Generic;
using System.Text;
namespace SangTao1
{
    /*-----
    *      Sinh ngẫu nhiên n số
    *      trong khoảng d..c
    * -----*/
    class RGen
    {
        static void Main(string[] args)
        {
            Print(Gen(20, -8, 8));
            Console.ReadLine();
        }
        static public int[] Gen(int n, int d, int c)
        {
            Random r = new Random();
            int len = c-d+1;
            int [] a = new int[n];
            for (int i = 0; i < n; ++i)
                a[i] = d + r.Next(len);
        }
    }
}

```

```

        return a;
    }
    static public void Print(int [] a)
    {
        Console.WriteLine();
        foreach (int x in a)
            Console.Write(x + " ");
        Console.WriteLine();
    }
} // RGen
} // SangTao1

```

Bài 2.2. Sinh ngẫu nhiên tăng

Sinh ngẫu nhiên n phần tử được sắp không giảm cho mảng nguyên a .

Thuật toán

1. Sinh ngẫu nhiên phần tử đầu tiên: $a[1] := \text{random}(n)$;
2. Từ phần tử thứ hai trở đi, trị được sinh bằng trị của phần tử sát trước nó cộng thêm một đại lượng ngẫu nhiên:
 $(i = 2..n): a[i] := a[i - 1] + \text{random}(n)$, do đó $a[i] \geq a[i - 1]$.

(* Pascal *)

```

(*-----
    Sinh ngau nhien cho mang nguyen a
    n phan tu sap khong giam
-----*)
program IncGen;
uses crt;
const MN = 1000;
var a: array [1..MN] of integer;
(*-----
    Sinh ngau nhien day tang gom n phan tu
-----*)
procedure Gen(n: integer);
var i: integer;
begin
    randomize;
    a[1]:= random(5); {khoi tao phan tu dau tien }
    for i:= 2 to n do a[i]:= a[i-1]+random(10);
end;
procedure Xem(n: integer); tự viết
procedure Test;
var n: integer;
begin
    n := 200; { test voi 200 phan tu }
    Gen(n); Xem(n); readln;
end;
BEGIN
    Test;
END.

```

// C#

```

using System;
using System.Collections.Generic;
using System.Text;
namespace SangTao1
{
    /*-----
    *      Sinh ngẫu nhiên n số
    *      tạo thành dãy không giảm
    * -----*/
    class IncGen
    {
        static void Main(string[] args)
        {
            Print(Gen(200));
            Console.ReadLine();
        }
        static public int[] Gen(int n)
        {
            Random r = new Random();
            int [] a = new int[n];
            a[0] = r.Next(5);
            for (int i = 1; i < n; ++i)
                a[i] = a[i-1] + r.Next(10);
            return a;
        }
        static public void Print(int [] a) tự viết
    } // IncGen
} // SangTao1

```

Bài 2.3. Sinh hoán vị ngẫu nhiên

Sinh ngẫu nhiên cho mảng nguyên a một hoán vị của 1..n.

Đặc tả

Xuất phát từ hoán vị đơn vị $a = (1, 2, \dots, n)$ ta đổi chỗ $a[1]$ với một phần tử tùy ý (được chọn ngẫu nhiên) $a[j]$ sẽ được một hoán vị. Ta có thể thực hiện việc đổi chỗ nhiều lần.

(* Pascal *)

```

(*-----
    Sinh ngẫu nhiên cho mảng nguyên a
    một hoán vị của 1..n
    -----*)
program GenPer;
const MN = 1000; { số lượng tối đa }
      Esc = #27; { dấu thoát }
      BL = #32; { dấu cách }
var a: array[1..MN] of integer;
(*-----
    Sinh dữ liệu
    -----*)
procedure Gen(n: integer);
var i, j, x: integer;
begin
    { Khởi tạo hoán vị đơn vị }

```

```

for i:= 1 to n do a[i]:= i;
for i:= 1 to n do
begin
  j := random(n)+1;
  x := a[1]; a[1] := a[j]; a[j] := x;
end;
end;
procedure Xem(n: integer); tự viết
procedure Test;
var n: integer;
begin
  randomize;
  repeat {chon ngau nhien kich thuoc n = 10..39}
    n := random(30)+10; Gen(n); Xem(n);
  until ReadKey = Esc; { Nhan ESC de thoat }
end;
BEGIN
  Test;
END.

```

// C#

```

using System;
using System.Collections.Generic;
using System.Text;
namespace SangTao1
{
  /*-----
  *   Sinh ngau nhien hoan vi
  *           1..n
  * -----*/
  class GenPer
  {
    static void Main(string[] args)
    {
      Print(Gen(20));
      Console.ReadLine();
    }
    static public int[] Gen(int n)
    {
      Random r = new Random();
      int[] a = new int[n];
      for (int i = 0; i < n; ++i)
        a[i] = i+1;
      for (int i = 0; i < n; ++i)
      {
        int j = r.Next(n);
        int t = a[0];
        a[0] = a[j]; a[j] = t;
      }
      return a;
    }
    static public void Print(int [] a) tự viết
  }
}

```

```

    } // IncGen
} // SangTao1

```

Bài 2.4. Sinh ngẫu nhiên đều

Sinh ngẫu nhiên n phần tử cho mảng nguyên a thoả điều kiện n phần tử tạo thành k đoạn liên tiếp có tổng các phần tử trong mỗi đoạn bằng nhau và bằng giá trị t cho trước.

Thuật toán

1. Chọn số lượng các phần tử trong mỗi đoạn là $\text{random}(n \text{ div } k) + 1$, khi đó số lượng các phần tử được phát sinh ngẫu nhiên sẽ không vượt quá

$$k * (n \text{ div } k) \leq n$$

Sau đó ta sẽ chỉnh sao cho số lượng các phần tử đúng bằng n .

2. Giả sử $a[d..c]$ là đoạn thứ j cần được sinh ngẫu nhiên sao cho

$$a[d] + a[d + 1] + \dots + a[c] = t$$

Ta sinh đoạn này như sau:

2.1. Gán $tr := t$; { tr - giá trị còn lại của tổng }.

2.2. Gán trị ngẫu nhiên $0..tr-1$ cho các phần tử $a[d..(c - 1)]$

($i = d..c$): $a[i] := \text{random}(tr)$

2.3. Đồng thời chỉnh giá trị còn lại của tr :

$tr := tr - a[i]$

Ta có:

$a[d] < t$

$a[d+1] < t - a[d]$

$a[d+2] < t - a[d+1] - a[d]$

...

$a[c - 1] < t - a[d] - a[d + 1] - \dots - a[c - 2]$

Chuyển về các phần tử $a[*]$ trong biểu thức cuối cùng, ta thu được

$$a[d] + a[d + 1] + \dots + a[c - 1] < t$$

2.4. Ta đặt giá trị còn lại của tổng riêng vào phần tử cuối đoạn: $a[c] := tr$ sẽ thu được $a[d] + a[d + 1] + \dots + a[c] = t$.

(* Pascal *)

```

(*-----*)
    Sinh ngau nhien cho mang nguyen a
    n phan tu tao thanh k doan lien tiep
    co tong bang nhau
(*-----*)
program KGen;
uses crt;
const MN = 1000; {kich thuoc toi da cua mang a}
      Esc = #27; {dau thoat}
      BL = #32; {dau cach}
var a: array[1..MN] of integer;
(*-----*)
    Sinh du lieu
(*-----*)
procedure Gen(n,k,t: integer);
var i,j,p,tr,s: integer;

```

```

begin
  if (k < 1) or (k > n) then exit;
  s := n div k; {s - số tối đa phần tử trong mỗi đoạn}
  i := 0; {chỉ dẫn liên tục cho các phần tử mỗi sinh}
  for j := 1 to k do {sinh đoạn thứ j}
    begin
      tr := t;
      for p := 1 to random(s) do
        { random(s)+1 = số phần tử trong 1 đoạn }
        begin
          inc(i);
          a[i] := random(tr);
          tr := tr - a[i]; {giá trị còn lại của tổng}
        end;
      inc(i); {i phần tử cuối cùng của đoạn j}
      a[i] := tr;
    end;
    {bù 0 cho các phần tử còn lại}
    for i := i+1 to n do a[i] := 0;
  end;
  procedure Xem(n: integer); Hiển thị mảng a, tự viết
  procedure Test;
  var n,k: integer;
  begin
    randomize;
    repeat
      n := random(30) + 1;
      k := random(8) + 1;
      t := random(30)+10;
      writeln('n = ',n,' k = ',k,' t = ',t);
      Gen(n,k,t); Xem(n);
    until ReadKey = Esc;
  end;
BEGIN
  Test;
END.

// C#
using System;
using System.Collections.Generic;
using System.Text;
using System;
namespace SangTaol
{
  class KGen
  {
    static void Main(string[] args)
    {
      Random r = new Random();
      int n, k, t;
      do
      {
        n = r.Next(30) + 1;

```

```

        t = r.Next(30) + 1;
        k = r.Next(8) + 1;
        Console.WriteLine("\n n = " + n +
            " k = " + k + " t = " + t);
        Print(Gen(n, k, t));
        Console.Write("\n Bam RETURN de tiep tục: ");
    } while (Console.ReadLine() == "");
}
// sinh n phần tử chia thành k đoạn,
// mỗi đoạn có tổng t
static public int[] Gen(int n, int k, int t)
{
    if (k < 1 || k > n) return new int[0];
    Random r = new Random();
    int[] a = new int[n];
    int s = n / k; // số phần tử trong 1 đoạn
    int i = 0;
    for (int j = 0; j < k; ++j)
    { // sinh đoạn thứ j
        int tr = t;
        int endp = r.Next(s);
        for (int p = 0; p < endp; ++p, ++i)
        { a[i] = r.Next(tr); tr -= a[i]; }
        a[i++] = tr;
    }
    // điền 0 cho đủ n phần tử
    for (; i < n; ++i) a[i] = 0;
    return a;
}
static public void Print(int[] a) tự viết
} // KGen
} // SangTao1

```

Bài 2.5. Sinh ngẫu nhiên tỉ lệ

Sinh ngẫu nhiên cho mảng nguyên a có n phần tử tạo thành hai đoạn liên tiếp có tổng các phần tử trong một đoạn gấp k lần tổng các phần tử của đoạn kia.

Thuật toán

1. Sinh ngẫu nhiên tổng **t1** := **random(n) + 1**.
2. Tính **t2** := **k*t1**.
3. Gieo đồng xu bằng cách gọi **random(2)** để xác định tổng nào trong số **t1** và **t2** được chọn trước.
4. Sinh **random(n div 2)+1** phần tử cho đoạn thứ nhất sao cho tổng các phần tử của đoạn này bằng **t1** (xem bài 2.4).
5. Sinh nốt các phần tử cho đoạn thứ hai sao cho tổng các phần tử của đoạn này bằng **t2**.

(* Pascal *)

```

program K2gen;
uses crt;
const MN = 1000;

```

```

var a: array[1..MN] of integer;
(*-----
      Sinh du lieu
-----*)
procedure Gen(n,k:integer);
var i,j,t1,t2:integer;
begin
  if (k < 1) OR (k > n) then exit;
  t1 := random(n) + 1;
  {tong mot doan; tong doan con lai = k*t1 }
  {chon ngau nhien doan co tong lon dat truoac hay sau
}

  if random(2)= 0 then t2 := k*t1
  else
begin
  t2 := t1; t1 := k*t2;
end;
  i := 0; {sinh doan thu nhat}
  for j := 1 to random(n div 2) do
begin
  inc(i); a[i] := random(t1);
  t1 := t1 - a[i];
end;
  inc(i); a[i] := t1;
  while i < n do {sinh doan thu hai }
begin
  inc(i); a[i]:= random(t2);
  t2 := t2 - a[i];
end;
  a[n] := a[n] + t2;
end;
procedure Xem(n: integer); tự viết
procedure Test;
var n,k: integer;
begin
  randomize;
  repeat
    n := random(30) + 1;
    k := random(8) + 1;
    write(' n = ',n,' k = ',k);
    Gen(n,k); Xem(n);
  until ReadKey = #27;
end;
BEGIN
Test;

```

END.

// C#

```
using System;
using System.Collections.Generic;
using System.Text;
namespace SangTao1
{
    class K2Gen
    {
        static void Main(string[] args)
        {
            Random r = new Random();
            int n, k;
            do
            {
                n = r.Next(30) + 2;
                k = r.Next(8) + 1;
                Console.WriteLine("\n n = " + n +
                    " k = " + k);
                int [] a = new int [n];
                int n1 = Gen(a,n,k);
                Print(a);
                Test(a, n1, k);
                Console.Write("\n Bam RETURN " +
                    " de tiep tuc: ");
            } while (Console.ReadLine() == "");
        }
        // Kiem tra ket qua
        static void Test(int[] a, int n1, int k)
        {
            int t1 = 0;
            for (int i = 0; i < n1; ++i)
                t1 += a[i];
            Console.WriteLine("\n Doan thu nhât: " +
                "sum(a[0.." + (n1 - 1) + "]) = " + t1);
            int t2 = 0;
            for (int i = n1; i < a.Length; ++i)
                t2 += a[i];
            Console.WriteLine("\n Doan thu hai: " +
                "sum(a["+n1+".." + (a.Length - 1) + "]) = "+t2);
            if ((t1 == k * t2) || (t2 == k * t1))
                Console.WriteLine("\n DUNG");
            else Console.WriteLine("\n SAI");
        }
    }
}
```

```

static public int Gen(int [] a, int n, int k)
{
    Random r = new Random();
    int i = 0; // phan tu thu i trong a
    // n1 - so phan tu trong doan 1
    int n1 = r.Next(n / 2) + 1;
    int t1 = 0; // tong doan 1
    // sinh doan thu 1
    for (; i < n1; ++i) //
    {
        a[i] = r.Next(10); t1 += a[i];
    }
    int t2 = k* t1;
    int tt = t1;
    // xac dinh ngau nhien
    // 0. t2 gap k lan t1, hoac
    // 1. t1 gap k lan t2
    if (r.Next(2)==1)
    { // t1 gap k lan t2
        t1 = t2; t2 = tt; a[i-1] += (t1-t2);
    }
    // sinh doan 2
    for (; i < n; ++i) //
    {
        a[i] = r.Next(t2); t2 -= a[i];
    }
    a[n-1] += t2;
    return n1;
}

static public void Print(int[] a)
{
    Console.WriteLine();
    foreach (int x in a)
        Console.Write(x + " ");
    Console.WriteLine();
}
} // K2Gen
} // SangTao1

```

Bài 2.6. Sinh ngẫu nhiên tệp tăng

Sinh ngẫu nhiên n số tự nhiên sắp tăng và ghi vào một tệp văn bản có tên cho trước.

Thuật toán

Bạn đọc xem trực tiếp chương trình và giải thích cách làm.

(* Pascal *)

```
(*-----
    Sinh ngẫu nhiên n số tự nhiên sắp tăng
    và ghi vào tệp văn bản có tên cho trước
-----*)
program FincGen;
uses crt;
const BL = #32; { dấu cách }
(*-----
    Sinh dữ liệu
-----*)
procedure Gen(fn: string; n: integer);
var   f: text; i: integer; x: longint;
begin
    assign(f,fn); {fn: file name (tên tệp)}
    rewrite(f); randomize;
    x := 0;
    for i:= 1 to n do
        begin
            x := x+random(10); write(f,x,BL);
            { mỗi dòng trong file chứa 20 số }
            if i mod 20 = 0 then writeln(f);
        end;
    if i mod 20 <> 0 then writeln(f);
    close(f);
end;
procedure Test;
begin
    Gen('DATA.INP',200);
    write('Ket'); readln;
end;
BEGIN
    Test;
END.
```

// C#

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
namespace SangTaoT1
{
    class FincGen
    {
        static void Main(string[] args)
        {
            string fn = "Data.txt";
            GenToFile(fn, 81);
            Show(fn); Console.ReadLine();
        }
        static public void GenToFile(string fn, int n)
```

```

    {
        StreamWriter f = File.CreateText(fn);
        Random r = new Random();
        int x = r.Next(10);
        for (int i = 0; i < n; ++i)
        {
            f.Write(x + " ");
            // Mỗi dòng 20 số
            if (i % 20 == 19) f.WriteLine();
            x += r.Next(10);
        }
        if (n % 20 != 19) f.WriteLine();
        f.Close();
    }
    static public void Show(string fn)
    {
        Console.WriteLine(File.ReadAllText(fn));
    }
} // FincGen
} // SangTaol

```

Bài 2.7. Sinh ngẫu nhiên tệp cấp số cộng

Sinh ngẫu nhiên một cấp số cộng có n số hạng và ghi vào một tệp văn bản có tên cho trước.

Thuật toán

1. Sinh ngẫu nhiên số hạng thứ nhất **a[1]** và công sai **d**.
 2. Sinh các phần tử **a[i]**, $i = 2..n$
for $i:=2$ **to** n **do** $a[i]:=a[i-1]+d$;
 3. Ghi file
- Độ phức tạp: n .

(* Pascal *)

```

program FCapCong;
uses crt;
const BL = #32;
procedure Gen(fn: string; n: integer);
var f: text; i,d: integer; x: longint;
begin
    assign(f,fn); rewrite(f);
    randomize;
    d := random(n div 4)+1; {cong sai }
    x := random(20); write(f,x,BL);
    for i:= 2 to n do
        begin { mỗi dòng ghi 20 số }
            x:= x + d; write(f,x,BL);
            if i mod 20 = 0 then writeln(f);
        end;
    if i mod 20 <> 0 then writeln(f);
    close(f);
end;
BEGIN

```

```
Gen('DATA.INP',200); write('Ket'); readln;
END.
```

```
// C#
```

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
namespace SangTaol
{
    class FCapCong
    {
        static void Main(string[] args)
        {
            string fn = "Data.txt";
            GenToFile(fn, 81);
            Show(fn); Console.ReadLine();
        }
        static public void GenToFile(string fn, int n)
        {
            StreamWriter f = File.CreateText(fn);
            Random r = new Random();
            int s = r.Next(n), d = r.Next(10)+1;
            for (int i = 0; i < n; ++i)
            {
                f.Write(s + " ");
                if (i % 20 == 19) f.WriteLine();
                s += d;
            }
            if (n % 20 != 19) f.WriteLine();
            f.Close();
        }
        static public void Show(string fn)
        {
            Console.WriteLine(File.ReadAllText(fn));
        }
    } // FcapCong
} // SangTaol
```

Bài 2.8. Sinh ngẫu nhiên mảng đối xứng

Sinh ngẫu nhiên các giá trị để ghi vào một mảng hai chiều $a[1..n, 1..n]$ sao cho các phần tử đối xứng nhau qua đường chéo chính, tức là

$$a[i, j] = a[j, i], 1 \leq i, j \leq N.$$

Thuật toán

1. Sinh ngẫu nhiên các phần tử trên đường chéo chính $a[i, i], i=1..n$.
 2. Sinh ngẫu nhiên các phần tử nằm phía trên đường chéo chính $a[i, j], i=1..n, j=i+1..n$ rồi lấy đối xứng: $a[j, i] := a[i, j]$.
- Độ phức tạp: n^2 .

(* Pascal *)

```

program GenMatSym;
uses crt;
const MN = 100;
var a = array[1..MN,1..MN] of integer;
procedure Gen(n: integer);
var i, j: integer;
begin
    randomize;
    for I := 1 to n do
        begin
            a[i,i] := random(n);
            for j := i+1 to n do
                begin
                    a[i,j]:=random(n); a[j,i]:=a[i,j];
                end;
            end;
        end;
    end;
procedure Xem(n: integer);
var i, j: integer;
begin
    writeln;
    for i:= 1 to n do
        begin
            writeln;
            for j:= 1 to n do write(a[i,j]:4);
        end;
    end;
end;
BEGIN
    Gen(20); Xem(20); readln;
END.

```

// C#

```

using System;
using System.Collections.Generic;
using System.Text;
namespace SangTaoI
{
    class GenMatSym
    {
        static void Main(string[] args)
        {
            int n = 20;
            int[,] a = Gen(n);
            Print(a, n);
            Console.WriteLine("\n Fini ");
            Console.ReadLine();
        }
        static public int [,] Gen(int n)
        {
            int[,] a = new int[n, n];
            Random r = new Random();
            for (int i = 0; i < n; ++i)

```

```

        {
            a[i, i] = r.Next(100);
            for (int j=i+1; j<n; ++j)
            { a[i, j] = r.Next(100);
              a[j, i] = a[i, j];
            }
        }
        return a;
    }
// hiển thị mảng a[0..(n-1)]
    static public void Print(int [,] a,int n) tự
viết
    } // GenMatSym
} // SangTao1

```

Bài 2.9. Số độ cao h

Độ cao của một số tự nhiên là tổng các chữ số của số đó. Sinh toàn bộ các số tự nhiên có tối đa ba chữ số và có độ cao h cho trước. Ghi kết quả vào một tệp văn bản có tên cho trước.

Thuật toán

Bài toán này có cách phát biểu khác và tổng quát như sau: *có n cốc nước dung tích 9 thìa mỗi cốc. Cho một bình đựng h thìa nước. Hãy xác định mọi phương án chia nước vào các cốc.*

Ta xét lời giải với $n = 3$. Ta có $h = 0..27$.

1. Các số cần tìm y có dạng $y = abc$, $a + b + c = h$ và a biến thiên từ $mina$ đến $maxa$, trong đó $mina$ là lượng nước ít nhất trong cốc đầu tiên a , $maxa$ là lượng nước lớn nhất trong cốc a . Nếu đổ đầy hai cốc b và c , mỗi cốc 9 thìa nước thì lượng nước còn lại sẽ là tối thiểu cho cốc a . Ngược lại, nếu tổng cộng chỉ có $h < 9$ thìa nước thì lượng nước tối đa trong cốc a phải là h . Ta có

```

if h ≤ 18 then mina := 0 else mina := h-18;
if h ≥ 9 then maxa := 9 else maxa := h;

```

2. Với mỗi $a = mina..maxa$ ta tính

2.1. $bc = h - a$ (biến bc chứa tổng các chữ số b và c).

2.2. Giải bài toán nhỏ với $n = 2$.

```

if bc ≤ 9 then minb := 0 else minb := bc-9;

```

```

if bc ≥ 9 then maxb := 9 else maxb := bc;

```

2.3. Với mỗi $b = minb..maxb$ ta tính

```

y = 100*a + 10*b + (bc - b) .

```

Ghi số này vào tệp.

(* Pascal *)

```

(*-----
Sinh cac so khong qua 3 chu so
co do cao h va ghi vao tep fn
-----*)
program HGen;
uses crt;

```

```

function Gen(fn:string;h:integer): integer;
var f: text;
a,b,bc,mina,maxa,minb,maxb: integer;
x,y,d: integer;
begin {tong 3 chu so toi da la 27, toi thieu la 0 }
    if (h < 0) OR (h > 27) then exit;
    assign(f,fn); rewrite(f);
    d:= 0; {dem so luong cac so do cao h}
    if h <= 18 then mina := 0 else mina := h-18;
    if h >= 9 then maxa := 9 else maxa := h;
    for a := mina to maxa do
    begin
        x := 100*a;
        bc := h-a;
        if bc <= 9 then minb := 0 else minb := bc-9;
        if bc >= 9 then maxb := 9 else maxb := bc;
        for b := minb to maxb do
        begin
            y := x + 10*b + (bc - b);
            write(f,y:4);
            inc(d); { Ghi moi dong 10 so }
            if d mod 10 = 0 then writeln(f);
        end;
    end;
    close(f);
    Gen := d;
end;
procedure Test;
var n: integer;
begin
    n := Gen('HEIGHT.NUM',10);
    write('Tong cong ',n,' so');
    readln;
end;
BEGIN
    Test;
END.

```

// C#

```

using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
namespace SangTaol
{
    class HGen
    {
        static void Main(string[] args)
        {
            string fn = "HGen.txt";
            int h = 10;
            Console.WriteLine("\n File " + fn);
        }
    }
}

```

```

        Console.WriteLine(" H = " + h);
        int d = Gen(fn,10);
        Test(fn,d);
        Console.WriteLine("\n Fini ");
        Console.ReadLine();
    }
    static public int Gen(string fn, int h)
    {
        int a, b, mina, maxa, minb, maxb,
        bc, x, y, d = 0;
        StreamWriter f = File.CreateText(fn);
        mina = (h <= 18) ? 0 : h-18;
        maxa = (h <= 9) ? h : 9;
        for (a = mina; a <= maxa; ++a)
        {
            x = 100*a; bc = h - a;
            minb = (bc <= 9) ? 0 : bc-9;
            maxb = (bc >= 9) ? 9 : bc;
            for (b = minb; b <= maxb; ++b)
            {
                ++d; y=x+10*b+(bc-b); f.Write(y+" ");
                if (d % 10 == 0) f.WriteLine();
            }
        }
        f.Close();
        return d;
    }
    // Doc lai file de kiem tra
    static public void Test(string fn,int d)
    {
        Console.WriteLine("\n Tong cong " +
                           d + " so");
        Console.WriteLine(File.ReadAllText(fn));
    }
} // HGen
} // SangTao1

```

Chú ý

1. Có thể giải bài toán trên bằng phương pháp vét cạn dùng ba vòng for như sau:

```

for a := 0 to 9 do
    for b := 0 to 9 do
        for c := 0 to 9 do
            if a+b+c = h then
                write(f,a,b,c,#32);

```

2. Phương pháp vét cạn đòi hỏi $10 \times 10 \times 10 = 1000$ lần duyệt trong khi với $h = 10$ chỉ có 63 số thoả mãn điều kiện của đầu bài. Dùng phương pháp sinh ta nhận được đúng 63 số cần tìm, không phải duyệt thừa số nào.

Bài 2.10. Tập các hoán vị

Với mỗi số n cho trước trong khoảng $1..9$, ghi vào một tệp văn bản có tên cho trước toàn bộ các hoán vị của $1..n$. Hoán vị được sắp xếp tăng theo thứ tự từ điển, thí dụ $21345 < 21354$.

Thuật toán

1. Khởi tạo và ghi hoán vị nhỏ nhất là hoán vị đơn vị $s[1..n] = (1, 2, \dots, n)$.
2. Giả sử ta đã ghi được hoán vị $s[1..n]$ vào tệp. Hoán vị tiếp theo được tạo từ s thông qua hàm **Next** như sau:
 - 2.1 **Tìm điểm gãy**: Tìm ngược từ $s[n]$ trở về trước đến vị trí i đầu tiên thỏa điều kiện $s[i] < s[i + 1]$.
 - Nếu không tìm được i tức là s là hoán vị lớn nhất, $s[1..n] = (n, n-1, \dots, 1)$. Đặt trị **false** cho hàm **Next** và dừng thuật toán. **Next = false** có nghĩa là không tồn tại hoán vị sát sau hoán vị s hay s là hoán vị lớn nhất.
 - Nếu tìm được: thực hiện bước 2.2.
 - 2.2 **Tìm điểm vượt**: Tìm ngược từ $s[n]$ trở về trước đến vị trí j đầu tiên thỏa điều kiện $s[j] > s[i]$.
 - 2.3 **Đổi chỗ** $s[i]$ với $s[j]$.
 - 2.4 **Lật**: Đảo lại trật tự của dãy $s[i + 1..n]$ ta sẽ thu được hoán vị đứng sát sau hoán vị s .
3. Đặt trị **true** cho hàm **Next**. **Next = true** có nghĩa là tìm được hoán vị sát sau hoán vị s .

Chú ý

Khi khởi tạo hoán vị đơn vị ta sử dụng phần tử $s[0] = 0$ làm lính canh. Nhờ vậy, khi duyệt ngược để tìm điểm gãy ta không phải kiểm tra giới hạn mảng. Thay vì viết

```
i := n-1;
while (i > 0) and (s[i] > s[i+1]) do i := i-1;
```

ta chỉ cần viết

```
i := n-1;
while (s[i] > s[i+1]) do i := i-1;
Hàm Next được mô tả như sau:
function Next(n: integer): Boolean;
var i, j, t: integer;
begin
  Next := false; i := n-1;
  while (s[i] > s[i+1]) do i := i-1;
  if i = 0 then exit;
  { s[i] < s[i+1], i là điểm gãy }
  j := n; { Tìm điểm vượt a[j] > a[i] }
  while (s[j] < s[i]) do j := j-1;
  { Đổi chỗ s[i], s[j] }
  t := s[i]; s[i] := s[j]; s[j] := t;
  { Lật s[i+1..n] } i := i+1; j := n;
  while i < j do
begin
  t := s[i]; s[i] := s[j]; s[j] := t;
  i := i+1; j := j-1;
```

```

end;
Next:= true;
end;

```

Thí dụ, với $n = 8$, giả sử ta đã ghi được hoán vị $s = 74286531$, khi đó hoán vị sát sau s sẽ được xây dựng như sau:

	①	②	③	④	⑤	⑥	⑦	⑧
S	7	4	2	8	6	5	3	1
Tìm điểm gãy: $i = 3$, vì $s[3] < s[4]$	7	4	<u>2</u>	8	6	5	3	1
Tìm điểm vượt: $j = 7$, vì $s[7] > s[3]$	7	4	2	8	6	5	<u>3</u>	1
Đổi chỗ điểm gãy và điểm vượt: $s[3] \leftrightarrow s[7]$	7	4	<u>3</u>	8	6	5	<u>2</u>	1
Lật đoạn $s[4..8]$	7	4	3	<u>1</u>	<u>2</u>	<u>5</u>	<u>6</u>	<u>8</u>

Quy trình hoạt động của hàm Next

$74286531 \Rightarrow 74312568$

(* Pascal *)

```

program GenAllPer;
{$B-}
uses crt;
const MN = 9; {max n} BL = #32; {dau cach}
var s: array[0..MN] of integer;
function Next(n: integer): Boolean; {tự viết}
procedure Gen(n: integer);
const
    fn = 'HoanVi.dat'; {ten tep ket qua}
var
    d: longint; {dem so luong hoan vi}
    i: integer;
    f: text; {tep ket qua}
begin
    if (n < 1) or (n > MN) then exit;
    assign(f,fn); rewrite(f);
    d := 0; {dem so hoan vi}
    {Sinh hoán vị đơn vị. Đặt lính canh s[0] = 0}
    for i := 0 to n do s[i] := i;
    repeat
        d := d+1; {Ghi hoan vi thu d, s vao tep}
        for i:= 1 to n do write(f, s[i], BL);
        writeln(f);
    until not (next(n));
    writeln(f, ' Tong cong ',d, ' hoan vi');
    close(f);
end;
BEGIN
    Gen(5); write('fini'); readln;
END.

```

// C#

```

using System;
using System.Collections.Generic;
using System.Text;

```

```

using System.IO;
namespace SangTaol
{
    class GenAllPer
    {
        static void Main(string[] args)
        {
            string fn = "HoanVi.txt";
            int d = Gen(fn,5);
            Test(fn,d); // Xem kết quả
            Console.WriteLine("\n Fini ");
            Console.ReadLine();
        }
        // Sinh các hoán vị, ghi file fn
        static public int Gen(string fn, int n)
        {
            if (n < 1 | n > 9) return 0;
            int d = 0; // dem so hoan vi d = n!
            StreamWriter f = File.CreateText(fn);
            int[] a = new int[n + 1];
            for (int i=0; i <= n; ++i) a[i]=i;
            do { // Ghi file
                for (int i=1; i <= n; ++i)
                    f.Write(a[i] + " ");
                f.WriteLine(); ++d;
            } while (Next(a));
            f.Close();
            return d;
        }
        static bool Next(int[] a)//Hoán vị tiếp
        {
            int i, j, t, n = a.Length-1;
            for (i=n-1; a[i] > a[i+1];--i) ;
            if (i == 0) return false;
            for (j = n; a[j] < a[i]; --j) ;
            t = a[i]; a[i] = a[j]; a[j] = t;
            for (++i, j = n; i < j; ++i, --j)
                { t = a[i]; a[i] = a[j]; a[j] = t; }
            return true;
        }
        static public void Test(string fn, int d)
        {
            Console.WriteLine("\n Tong cong "
                               + d + " so");
            Console.WriteLine(File.ReadAllText(fn));
        }
    } // GenAllPer
} // SangTaol

```

Bài 2.11. Đọc dữ liệu từ tệp vào mảng biết hai kích thước

Đọc dữ liệu kiểu nguyên từ một tệp văn bản vào một mảng hai chiều.

Tệp có cấu trúc như sau:

- Hai số đầu tiên, m là kích thước của mảng gồm n dòng và m cột.

- Tiếp đến là các dữ liệu ghi liên tiếp nhau theo từng dòng của mảng.
- Các số cách nhau ít nhất một dấu cách.

Thí dụ:

2 3 -1 4 5 3 7 1

cho biết mảng có $n = 2$ dòng và $m = 3$ cột với dữ liệu như sau:

-1	4	5
3	7	1

Đặc tả

Ta viết hàm **Doc** cho giá trị **true** nếu đọc được dữ liệu. Chú ý rằng dữ liệu vào là đúng do đó không cần kiểm tra tính đúng đắn của chúng. Như vậy **Doc** sẽ cho giá trị **false** trong trường hợp không mở được **file**, do ghi sai đường dẫn hoặc **file** không được tạo lập từ trước.

Chỉ thị **{ \$I- }** yêu cầu hệ thống chỉ ghi nhận chứ không bắt các lỗi vào/ra, tức là không dừng sự thực hiện chương trình. Biến hệ thống **IORESULT** sẽ ghi nhận số hiệu lỗi. Nếu **IORESULT=0** thì thao tác vào ra không sinh lỗi, ngược lại, nếu **IORESULT \neq 0** tức là đã có lỗi.

Chỉ thị **{ \$I+ }** yêu cầu hệ thống bắt mọi lỗi vào/ra. Như vậy, dòng lệnh

{ \$I- } reset(f) ; { \$I+ }

sẽ được hiểu như sau:

Thoạt tiên ta yêu cầu hệ thống bỏ chế độ bắt lỗi vào/ra **{ \$I- }**. Sau đó thực hiện lệnh mở tệp để đọc **reset(f)**. Tiếp đến đặt lại chế độ bắt lỗi **{ \$I+ }**.

(* Pascal *)

```
uses crt;
const MN = 100;
var a: array[1..MN,1..MN] of integer;
m,n: integer;
Function Doc(fn: string): Boolean;
var
    f: text;
    i, j: integer;
begin
    Doc := false; assign(f,fn);
    { $I- } reset(f); { $I+ }
    if IORRESULT <> 0 then exit; {không mở được file}
    read(f,n,m); {doc kích thước n và m của mảng }
    for i := 1 to n do
        for j:= 1 to m do
            read(f,a[i, j]);
    close(f);
    Doc := true;
end;
procedure Xem(n,m: integer); Hiển thị mảng 2 chiều,
tự viết
BEGIN
    if Doc('DATA.INP') then Xem(n,m)
    else write('Không mở được tệp ');
    readln;
```

END.

// C#

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
namespace sabgTao1
{
    class DocMang2Chieu
    {
        static void Main(string[] args)
        {
            string fn = "Data.inp";
            int n = 0, m = 0;
            int [,] a = Doc(fn, ref n, ref m);
            if (a != null)
            {
                PrintInput(fn);
                Print(a, n, m);
            }
            else
                Console.WriteLine("\n " +
                    " Khong mo duoc file " +fn);
            Console.WriteLine("\n Fini ");
            Console.ReadLine();
        }

        static public int[,] Doc(string fn,
            ref int n, ref int m)
        {
            if (!File.Exists(fn)) return null;
            // Cac dau ngan
            char[] cc = new char[]
            { ' ', '\n', '\t', '\r' };
            // Mo tep ten fn doc, toch, dong tep
            string[] ss = (File.ReadAllText(fn)).
                Split(cc,
                    StringSplitOptions.RemoveEmptyEntries);
            // Chuyển sang mảng 1 chiều int [] c
            int [] c = Array.ConvertAll(ss,
                new Converter<string,int>(int.Parse));
            n = c[0]; m = c[1];
            int[,] a = new int[n, m];
            int k = 2;
            for (int i = 0; i < n; ++i)
                for (int j = 0; j < m; ++j)
                    a[i,j] = c[k++];
            return a;
        }

        static void Print(int[,] a, int n, int m)
        {
            Hiển thị mảng 2 chiều a, tự viết
        }
    }
}
```

```

        static public void PrintInput(string fn)
        {
            Đọc lại file fn, tự viết
        } // DocMang2Chieu
    } // sangTao1

```

Giải thích

Trong các máy tính hiện đại, bộ nhớ trong RAM đủ lớn để có thể chứa toàn bộ dữ liệu trong hầu hết các file input vì thế với môi trường **C# .NET** bạn nên đọc một lần dữ liệu từ các file này. Hàm **Doc** cho ra mảng nguyên hai chiều. Nếu file không tồn tại, hàm cho ra giá trị null. Bạn cần chuẩn bị trước file input với tên **Data.inp** và ghi vào thư mục **BIN\DEBUG** trong Project hiện hành. Nếu ghi file vào thư mục khác thì trong tham biến **fn** phải ghi chi tiết đường dẫn, thí dụ **"D:\MyDIR\Data.inp"**. Khi viết đường dẫn, thay vì viết dấu **"\"** ta phải viết hai dấu đó, tức là **"\\"** vì bản thân dấu **"\"** trong đóng vai trò báo hiệu kí tự đứng sát sau nó là kí tự điều khiển, thí dụ, **"\n"** biểu thị dấu xuống dòng. Bạn cũng có thể viết *dấu đối mức* @ cạnh đường dẫn để chỉ thị rằng bạn muốn dùng một dấu **"\"** thay vì hai dấu, thí dụ,

```
@ "D:\MyDIR\Data.inp"
```

Lệnh **File.ReadAllText(fn)** mở file với đường dẫn **fn** đọc toàn bộ dữ liệu một lần vào một biến string sau đó tự động đóng file.

Lệnh **Split(cc, StringSplitOptions.RemoveEmptyEntries)**

tách các đơn vị trong biến string để ghi vào biến **string[] ss** đồng thời bỏ đi các *dấu trắng* mô tả trong biến **cc**, bao gồm dấu cách **' '**, dấu xuống dòng **'\n'**, dấu tab **'\t'** và dấu kết **RETURN '\r'**, cuối cùng loại bỏ các đơn vị rỗng, tức là các string không chứa kí tự nào (**Length = 0**).

Lệnh **int[] c = Array.ConvertAll(ss, New Converter<string,int>(int.Parse));**

chuyển các string trong **ss** sang dạng số nguyên và ghi vào mảng nguyên (một chiều) **c**. Đến đây toàn bộ dữ liệu trong file input **fn** đã được đọc và ghi vào mảng nguyên **c**. Các mảng trong **C#** được đánh chỉ dẫn từ **0** đến **Length-1**. Theo điều kiện của đầu bài **c[0]** chứa giá trị **n**, **c[1]** chứa giá trị **m**, từ **c[2]** trở đi chứa lần lượt các giá trị trên các dòng của mảng hai chiều.

Bài 2.12. Đọc dữ liệu từ tệp vào mảng biết một kích thước

Đọc dữ liệu kiểu nguyên từ một tệp văn bản vào một mảng hai chiều $a[n,m]$ cho biết một kích thước m (số cột).

Tệp có cấu trúc như sau:

- Số đầu tiên ghi số lượng cột **m** của mảng tức là số phần tử trên một dòng.
- Tiếp đến là các dữ liệu ghi liên tiếp nhau theo từng dòng của mảng.
- Các số cách nhau ít nhất một dấu cách.

Thí dụ:

3 -1 4 5 3 7 1

sẽ được bố trí vào mảng **n = 3** dòng, **m = 3** cột như sau:

-1	4	5
3	7	1

Thuật toán

1. Mở tệp.
2. Đọc giá trị đầu tiên vào biến m : số lượng cột của ma trận.
3. Mỗi lần đọc xong một dòng ta tăng con đếm dòng (n) thêm 1.

Chú ý

Do có thể gặp dòng trống nên ta cần sử dụng hàm `SeekEof`. Hàm `SeekEof` duyệt tiếp từ vị trí hiện thời của con trỏ tệp, bỏ qua các dấu trắng (gồm dấu cách, dấu kết thúc dòng, dấu đầu dòng, dấu nhảy TAB), nếu gặp dấu hết tệp thì cho giá trị `true`, ngược lại, nếu sau khi đã bỏ qua các dấu trắng mà chưa gặp dấu hết tệp thì cho giá trị `false`.

(* Pascal *)

```
uses crt;
const MN = 100;
var   a: array[1..MN,1..MN] of integer;
      m,n: integer;
Function Doc(fn: string): Boolean;
var f: text; j: integer;
begin
  Doc := FALSE; assign(f,fn);
  {$I-} reset(f); {$I+}
  if IORESULT <> 0 then exit;
  read(f,m); {m: so luong cot}
  n := 0; {n: so luong dong}
  while NOT SeekEof(f) do
    begin
      inc(n);
      for j := 1 to m do read(f,a[n,j]);
    end;
  close(f);
  Doc := TRUE;
end;
procedure Xem(n,m: integer); tự viết
BEGIN
  if Doc('DATA.INP') then Xem(n,m)
  else write('Khong mo duoc tep ');
  readln;
END.
```

Chú ý

Cần chuẩn bị trước dữ liệu và ghi trong tệp văn bản `DATA.INP`, thí dụ:

DATA.INP						
3	-1	4	5	3	7	1

// C#

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
```

```

namespace SangTao1
{
    class DocMang2
    {
        static void Main(string[] args)
        {
            string fn = "Data.inp";
            int n = 0, m = 0;
            int [,] a = Doc(fn, ref n, ref m);
            if (a != null)
            {
                PrintInput(fn);
                Print(a, n, m);
            }
            else Console.WriteLine("\n " +
                " Khong mo duoc file " + fn);
            Console.WriteLine("\n Fini ");
            Console.ReadLine();
        }

        static public int[,] Doc(string fn,
            ref int n, ref int m)
        {
            if (!File.Exists(fn)) return null;
            int [] c = Array.ConvertAll(
                File.ReadAllText(fn).
                    Split(new char[] { ' ', '\n', '\t', '\r' },
                        StringSplitOptions.RemoveEmptyEntries),
                new Converter<string,int>(int.Parse));
            int k = 0;
            m = c[k++]; n = (c.Length-1)/m;
            int[,] a = new int[n, m];
            for (int i = 0; i < n; ++i)
                for (int j = 0; j < m; ++j)
                    a[i,j] = c[k++];
            return a;
        }

        static void Print(int [,] a, int n, int m)
            Hiển thị mảng 2 chiều a[n,m], tự viết
        static public void PrintInput(string fn)
            Hiển thị file fn; tự viết
    } // DocMang2
} // SangTao1

```

Giải thích

Biết số cột của mảng là m ta có thể tính ra số dòng n của mảng theo công thức $n = (c.Length-1) / m$, trong đó $c.Length$ chứa số lượng các giá trị đã đọc từ file input, bao gồm giá trị m và $n.m$ giá trị của mảng, tức là $c.Length = n.m+1$.

Bài 2.13. Đọc dữ liệu từ tệp vào mảng đối xứng

Đọc dữ liệu kiểu nguyên từ một tệp văn bản có tên fn vào một mảng hai chiều đối xứng.

Tệp có cấu trúc như sau:

- Số đầu tiên ghi số lượng cột (và đồng thời là số lượng dòng) của mảng.
- Tiếp đến là các dữ liệu ghi liên tiếp nhau theo nửa tam giác trên tính từ đường chéo chính.
- Các số cùng dòng cách nhau ít nhất một dấu cách.

Thí dụ: 3 1 2 3 4 6 8 sẽ được bố trí vào mảng 3×3 như sau:

<u>1</u>	<u>2</u>	<u>3</u>
2	<u>4</u>	<u>6</u>
3	6	<u>8</u>

Thuật toán

1. Mở tệp.
2. Đọc giá trị đầu tiên vào biến n : số lượng cột và dòng của ma trận vuông đối xứng.
3. Với mỗi dòng i ta đọc phần tử trên đường chéo chính của dòng đó $a[i, i]$, sau đó ta đọc các phần tử nằm ở bên phải $a[i, i]$, tức là $a[i, j]$ với $j = i + 1..n$ rồi lấy đối xứng bằng phép gán $a[j, i] := a[i, j]$.

(* Pascal *)

```
uses crt;
const MN = 100;
var   a: array[1..MN,1..MN] of integer;
      n: integer; { kích thước mảng }
Function Doc(fn: string): Boolean;
var   f: text; i, j: integer;
begin
  Doc := FALSE; assign(f,fn);
  {$I-} reset(f); {$I+}
  if IORESULT <> 0 then exit;
  read(f,n);
  for i := 1 to n do
    begin
      read(f,a[i,i]);
      for j := i+1 to n do
        begin
          read(f,a[i,j]); a[j,i] := a[i,j];
        end;
    end;
  close(f); Doc:= TRUE;
end;
procedure Xem(n,m: integer); tự viết
BEGIN
  if Doc('DATA.INP') then Xem(n,n)
  else write('Khong mo duoc tep ');
  readln;
END.
```

// C#

```

using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
namespace SangTao1
{
    class MangDoiXung
    {
        static void Main(string[] args)
        {
            string fn = "Data.inp";
            int n = 0;
            int [,] a = Doc(fn, ref n);
            if (a != null)
            {
                PrintInput(fn);
                Print(a, n);
            }
            else Console.WriteLine("\n " +
                " Khong mo duoc file "+fn);
            Console.WriteLine("\n Fini ");
            Console.ReadLine();
        }
        static public int[, ] Doc(string fn,
            ref int n)
        {
            if (!File.Exists(fn)) return null;
            int [] c = Array.ConvertAll(
                File.ReadAllText(fn).
                Split(new char[] { ' ', '\n', '\t', '\r' },
                    StringSplitOptions.RemoveEmptyEntries),
                new Converter<string,int>(int.Parse));
            int k = 0; n = c[k++];
            int[, ] a = new int[n, n];
            for (int i = 0; i < n; ++i)
                for (int j = i; j < n; ++j)
                    a[i,j] = a[j,i] = c[k++];
            return a;
        }
        static void
        Print(int [,] a, int n)
        {
            Hiển thị mảng 2
            chiều a[n,n], tự viết
            static public void PrintInput(string fn)
            {
                Hiển thị file fn, tự viết
            } // MangDoiXung
        } // SangTao1
    }

```

1	1	1	1	0	0	1	1	1
0	0	0	0	0	0	1	1	1

Bài 2.14. Đếm tàu

Một tệp văn bản có tên *fn* có ghi sơ đồ một vùng biển hình chữ nhật chiều ngang 250 kí tự, chiều dọc (số dòng) không hạn chế. Trên biển có các con tàu hình chữ nhật chứa các kí tự 1, vùng nước được biểu thị qua các kí tự 0. Biết rằng các con tàu không dính nhau. Hãy đếm số lượng tàu.

1	1	0	0	0	0	0	0	0
1	1	0	0	1	1	0	0	1

5 tàu

Ví dụ, hình bên có 5 tàu.

Thuật toán

Vì các tàu không dính nhau nên ta phân biệt các tàu qua mũi tàu, tức là góc A - góc Tây-Bắc của tàu. Ta có,

$$\text{số lượng tàu} = \text{số lượng mũi tàu}$$

Mũi tàu là điểm nhận giá trị 1 và nếu bước một bước sang trái hoặc lên trên sẽ lên bờ hoặc rơi xuống biển.

Sau khi mở tệp ta đọc và xử lí từng dòng văn bản *y* và so sánh nó với dòng *x* đã xử lí trước đó. Nếu *y* là dòng đầu tiên, tức là dòng nằm sát bờ Bắc, ta khởi trị cho *x* với 250 ks tự 0 tức là ta loại trừ trường hợp bước lên bờ Bắc. Khi xử lí *y*, ta chú ý tách riêng trường hợp tàu nằm sát bờ Tây, tức là xét riêng *y*[1]. Sau mỗi lần xử lí dòng *y* ta copy dòng *y* sang *x* và luôn giữ cho *x* có chiều dài tối đa 250 kí tự như yêu cầu của đầu bài.

(* Pascal *)

```

program Ships;
{$B-}
uses crt;
const MN = 250;
boong = '1'; nuoc = '0';
Function Dem(fn: string):
integer;
var
    f: text; d,i: integer;
    x,y: string; {x:dong tren, y:dong duoi }
begin
    Dem := 0; assign(f,fn);
    {$I-} reset(f); {$I+}
    if IORESULT <> 0 then exit;
    x := nuoc;
    for i := 1 to 8 do x:= x+x; {x = '00...0'}
    d := 0;
    while NOT EOF(f) do
    begin
        readln(f,y);
        if (y[1]=boong)AND(x[1]=nuoc) then d:=d+1;
        for i:=2 to length(y) do
            if (y[i]= boong) AND (y[i-1]= nuoc)
                AND (x[i]= nuoc) then d:=d+1;
    
```

A	0	0	0	0	0	0	B
0	1	1	1	1	1		
0	1	1	1	1	1		
D	0	0	0	0	0	0	C

Con tàu ABCD

```

        x := y;
    end;
    Dem := d;
end;
BEGIN
    n:= Dem('TAU.INP');
    if n=0 then
        write('Khong mo duoc tep/khong co tau')
    else write('Tong so tau: ',n);
    readln;
END.

```

// C#

```

using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
namespace SangTao1
{
    class Ships
    {
        static public string fn = "Tau.inp";
        static public string gn = "Tau.out";
        static public char boong = '1';
        static public char nuoc = '0';
        static void Main(string[] args)
        {
            Save(Count());
            Test();
            Console.WriteLine("\n Fini ");
            Console.ReadLine();
        }
        static public int Count()// dem tau
        {
            StreamReader f = File.OpenText(fn);
            string x = new string(nuoc,251);
            string y;
            string empty = "";
            int d = 0;
            while ((y=(f.ReadLine()).Trim())
                    != empty)
            {
                d += Scan(x, y); x = y;
            }
            f.Close(); return d;
        }
    }
    // Sánh dòng trên x với dòng dưới y
    static public int Scan(string x, string y)
    {
        int d = 0;
        if ((y[0]==boong)&&(x[0]==nuoc)) ++d;
        for (int i = 1; i < y.Length; ++i)

```

```

        if ((y[i]==boong) && (y[i-1]==nuoc)
            && (x[i]==nuoc)) ++d;
        return d;
    }
    static public void Save(int d) // ghi file
    { File.WriteAllText(gn, d.ToString()); }
    static public void Test()
    {
        Console.WriteLine("\n" +
            File.ReadAllText(fn) + "\n");
        Console.WriteLine("\n" +
            File.ReadAllText(gn) + "\n");
    }
} // Ships
} // SangTaol

```

Bài 2.15. Sắp đoạn

Trong một tệp văn bản chứa những đoạn cắt ra từ một trục số. Mỗi đoạn có dạng $\langle d, c \rangle$ trong đó " $<$ " có thể là một trong hai kí tự (hoặc $[$, $>$ có thể là một trong hai kí tự) hoặc $]$, d và c là các biểu thức dạng x hoặc $x + y$ hoặc $x * y$ với x và y là những số tự nhiên. Ta luôn có $d \leq c$. Chiều dài của đoạn $\langle d, c \rangle$ là hiệu $c - d$. Hãy sắp xếp các đoạn tăng theo chiều dài và ghi chúng vào một tệp văn bản theo đúng dạng thức đọc được của mỗi đoạn. Có thể thêm, bớt một số dấu cách trong và ngoài các đoạn. Trên mỗi dòng của tệp luôn luôn chứa trọn một số đoạn.

Thí dụ cho dữ liệu vào trong file input "Doan.inp" là:

[2+1, 7) (4, 4*3) (5, 6]

Sau khi sắp ta được kết quả sau trong file output "Doan.out":

(5, 6] [2+1, 7) (4, 4*3)

Thuật toán

Ta mô tả cấu trúc của mỗi đoạn như sau:

mo so1[toan1 so2], so3[toan2 so4]

trong đó:

- ♦ mo là một trong hai dấu mở ngoặc: (hoặc [.
- ♦ so1, so2, so3 và so4 là các số tự nhiên xuất hiện trong thành phần của đoạn.
- ♦ toan1 và toan2 là dấu các phép toán (+, *), nếu có trong thành phần của đoạn.
- ♦ dong là một trong hai dấu đóng ngoặc:) hoặc].

Trong mô tả trên, chúng ta sử dụng kí pháp $[*]$ để chỉ ra thành phần $*$ có thể bỏ qua.

Nếu thành phần thứ i ($i = 1..2$) của đoạn không có dấu phép toán, thì cũng không có toán hạng thứ hai, tức là thành phần đó có dạng là một số tự nhiên thì ta đặt $\text{toan}[i] = \text{BL}$ (dấu cách).

Nếu số thứ i không xuất hiện trong đoạn, ta đặt $\text{so}[i] = 0$.

Thí dụ:

Đoạn	mo	so1	Toan1	so2	so3	Toan2	so4	dong
------	----	-----	-------	-----	-----	-------	-----	------

[2+10, 7*6)	[2	+	10	7	*	6)
[2+10, 7)	[2	+	10	7	BL	0)
(2, 7+5]	(2	BL	0	7	+	5]

Ngoài ra ta thêm một thành phần len để xác định chiều dài của đoạn. len của mỗi đoạn được tính theo công thức sau

len = TriCuoi-TriDau

TriCuoi = so3 Toan2 so4, nếu Toan2 là dấu '+' hoặc '*' và

TriCuoi = so3, nếu Toan2 = BL.

Tương tự,

TriDau = so1 Toan1 so2, nếu Toan1 là dấu '+' hoặc '*' và

TriDau = so1, nếu Toan1 = BL.

Ta sử dụng cấu trúc bản ghi để biểu diễn dữ liệu cho mỗi đoạn:

type

MangSo = array[1..4] of integer; {4 toan hang}

MangToan = array[1..2] of char; {2 toan tu +,*}

KieuDoan = record

mo: char;

dong: char;

so: MangSo;

Toan: MangToan;

len: integer;

end;

Các đoạn đọc được sẽ được ghi dần vào mảng a với biến đếm số phần tử n:

type MangDoan = array[0..1000] of KieuDoan;

var a: MangDoan; n: integer;

Khi đó thủ tục tính chiều dài len của mỗi đoạn sẽ được cài đặt như sau:

procedure LenSeg(i: integer);

var dau, cuoi: integer;

begin

with a[i] do

begin

dau := so[1];

if Toan[1]='+' then dau := dau+so[2]

else if Toan[1]='*' then dau:=dau*so[2];

cuoi:=so[3];

if Toan[2]='+' then cuoi:=cuoi+so[2]

else if Toan[2]='*' then cuoi:=cuoi*so[2];

end;

len := cuoi-dau;

end;

Cấu trúc **with x do T** cho phép ta thực hiện thao tác **T** trên các thành phần của bản ghi **x** mà không phải viết lại phần tiếp đầu **x**.

Để đọc các đoạn từ tệp ta sử dụng một máy trạng thái như sau. Hãy tưởng tượng mắt bạn bị bịt kín, do đó bạn phải dùng tay để nhận biết từng kí tự trong tệp văn bản.

Mỗi lần bạn sờ một kí tự c nào đó rồi dựa vào kí tự đó bạn xác định các thủ tục cần thực hiện để nhận biết từng đối tượng. Muốn vậy ta sử dụng một biến gọi là biến trạng thái q với mục đích ghi nhận các tình huống đã gặp và trên cơ sở đó xác định các thao tác cần thiết. Gọi q là biến trạng thái. Trong quá trình đọc và xử lí tệp input ta có thể gặp năm trạng thái như sau:

$q = 0$: Trạng thái dò tìm đầu đoạn: Nếu gặp kí tự mở đầu một đoạn, cụ thể là nếu gặp kí tự $c = '('$ hoặc $c = '['$ thì cần tạo một đoạn mới như sau:

- Tăng chỉ dẫn ghi nhận đoạn mới: $n := n + 1$;
- Ghi nhận kí tự mở đầu đoạn: $a[n].mo := c$;
- Khởi trị mảng số: $a[n].so := (0, 0, 0, 0)$;
- Khởi trị mảng dấu các phép toán: $a[n].Toan := (BL, BL)$;
- Chuyển qua trạng thái $q := 1$ là trạng thái tìm đọc $so[1]$.

```
0: if c in '(' , '[' then
    begin
        n:=n+1; a[n].mo:=c;
        a[n].so:=KhoiTriSo;
        a[n].Toan:=KhoiTriToan;
        q:= 1;
    end;
```

Các biến **KhoiTriSo** và **KhoiTriToan** được khai báo và gán trị khởi đầu như sau:

```
const
    KhoiTriSo: MangSo = (0,0,0,0);
    KhoiTriToan: MangToan = (BL,BL);
```

$q = 1$: Trạng thái tìm đọc số thứ nhất, $so[1]$: Ở trạng thái này, nếu gặp chữ số thì ta ghép thêm chữ số đó vào $so[1]$, nếu gặp dấu phép toán thì ta hiểu là thành phần thứ nhất của đoạn là một biểu thức dạng:

$so[1] \text{ Toan}[1] \text{ so}[2]$

Ta ghi nhận dấu phép toán vào trường **Toan[1]** và chuyển qua trạng thái **$q = 2$** để đọc số thứ hai. Nếu gặp dấu phẩy (,) là dấu ngăn giữa hai thành phần của đoạn ta chuyển qua trạng thái **$q = 3$** để đọc số đầu tiên của thành phần thứ hai, tức là đọc $so[3]$.

```
1: if c in ChuSo then DocSo(n,1)
    else if c in PhepToan then
        begin a[n].Toan[1]:=c; q:=2; end
    else if c=',' then q:=3;
```

Thủ tục **DocSo(i,j)** nhận thêm 1 chữ số để ghép vào biến $a[i].so[j]$.

$q = 2$: Đọc số thứ hai, $so[2]$: Ở trạng thái này, nếu gặp chữ số thì ta ghép thêm chữ số đó vào $so[2]$, nếu gặp dấu phẩy là dấu ngăn giữa hai thành phần của đoạn ta chuyển qua trạng thái **$q = 3$** để đọc số đầu tiên của thành phần thứ hai, tức là đọc $so[3]$.

```
2: if c in ChuSo then DocSo(n,2)
    else if c=',' then q:=3;
```

$q = 3$: Đọc số thứ ba, $so[3]$: Ở trạng thái này, nếu gặp chữ số thì ta ghép thêm chữ số đó vào $so[3]$, nếu gặp dấu phép toán thì ta hiểu là thành phần thứ hai của đoạn là một biểu thức dạng:

so[3] Toan[2] so[4]

Ta ghi nhận dấu phép toán vào trường Toan[2] và chuyển qua trạng thái $q = 4$ để đọc số thứ tư, **so[4]**, nếu gặp kí tự $c = ')'$ hoặc $c = ']'$ thì ta hiểu là đã kết thúc một đoạn, ta gọi thủ tục **KetDoan** để thực hiện các thao tác sau:

- Ghi nhận kí tự đóng đoạn: **a[n].dong := c**.
- Tính chiều dài của đoạn: **LenSeg(n)** ;
- Chuyển qua trạng thái $q = 0$ để tiếp tục với đoạn tiếp theo, nếu còn.

```
procedure KetDoan;
begin
    a[n].dong:=c; LenSeg(n); q:=0;
end;
```

Đoạn chương trình thể hiện trạng thái $q = 3$ khi đó sẽ như sau:

```
3: if c in ChuSo then DocSo(n,3)
    else if c in PhepToan then
        begin a[n].Toan[2]:=c; q:=4 end
    else if c in[')',''] then KetDoan;
```

q = 4: Đọc số thứ tư, so[4]: Ở trạng thái này, nếu gặp chữ số thì ta ghép thêm chữ số đó vào **so[4]**, nếu gặp kí tự $c = ')'$ hoặc $c = ']'$ thì ta hiểu là đã kết thúc một đoạn, ta gọi thủ tục **KetDoan**.

```
4: if c in ChuSo then DocSo(n,4)
    else if c in[')',''] then KetDoan;
```

Đọc tệp xong ta dùng thủ tục **qsort** sắp các đoạn tăng dần theo chiều dài. Sau khi sắp ta ghi các đoạn vào tệp **gn** theo các trường.

(* Pascal *)

```
{ $B- }
program Segments;
uses crt;
const
    fn = 'DOAN.INP'; {Tep input}
    gn = 'DOAN.OUT'; {Tep output}
    MN = 1000; {So luong toi da cac doan}
    BL = #32; {Dau cach}
    ChuSo = ['0'..'9'];
    PhepToan = ['+', '*'];
type
    MangSo = array[1..4] of integer;
    MangToan = array[1..2] of char;
    KieuDoan = record
        mo: char; {dau mo ngoac}
        dong: char; {dau dong ngoac}
        so: MangSo; {4 so trong doan}
        Toan: MangToan; {2 phép toán}
        len: integer; {chieu dai doan}
```

```

        end;
        MangDoan = array[0..MN] of KieuDoan;
const
    KhoiTriSo: MangSo = (0,0,0,0);
    KhoiTriToan: MangToan = (BL,BL);
var
    f,g:text;
    a: MangDoan;
    c: char;{ky tu dang xet}
    n: integer;{chi so doan dang xet}
    q: integer;{bien trang thai}
(*-----*)
    Cac trang thai q = 0: do tim dau doan
    1: doc so[1]
    2: doc so[2]
    3: doc so[3]
    4: doc so[4]
-----*)
procedure LenSeg(i: integer); tự viết
procedure KetDoan; tự viết
(*-----*)
    Them 1 chu so vao so thu j cua doan i
-----*)
procedure DocSo(i,j: integer);
begin
    a[i].so[j]:=a[i].so[j]*10+(ord(c)-ord('0'))
end;
(*-----*)
    Doc cac doan
-----*)
procedure doc;
begin
    assign(f,fn); reset(f);
    q:=0; n:=0;
    while not eof(f) do
    begin
        read(f,c);
        case q of
        0: if c in['(',')'] then
            begin
                n:=n+1; a[n].mo:=c;
                a[n].so:=KhoiTriSo;
                a[n].Toan:=KhoiTriToan;
                q:=1;
            end;

```

```

1: if c in ChuSo then DocSo(n,1)
   else if c in PhepToan then
     begin a[n].Toan[1]:=c; q:=2 end
   else if c=', ' then q:=3;
2: if c in ChuSo then DocSo(n,2)
   else if c=', ' then q:=3;
3: if c in ChuSo then DocSo(n,3)
   else if c in PhepToan then
     begin
       a[n].Toan[2]:=c; q:=4;
     end
     else if c in['(',')','(',')'] then KetDoan;
4: if c in ChuSo then DocSo(n,4)
   else if c in ['(',')','(',')'] then KetDoan;
end; { case }
end; { while }
close(f);
end;
procedure qsort(d,c:integer);
var i,j,m: integer;
    x: KieuDoan;
begin
  i:=d; j:=c; m:=a[(i+j) div 2].len;
  while i<=j do
    begin
      while a[i].len < m do i:=i+1;
      while a[j].len > m do j:=j-1;
      if i<=j then
        begin
          x:=a[i]; a[i]:=a[j]; a[j]:= x;
          i:=i+1; j:=j-1;
        end;
      end;
    end;
  if d < j then qsort(d,j);
  if i < c then qsort(i,c);
end;
procedure Ghi;
var i: integer;
begin
  assign(g,gn); rewrite(g);
  for i:=1 to n do
    with a[i] do
      begin
        if Toan[1]<>BL then
          write(g,mo, so[1],Toan[1],so[2])

```

```

        else write(g,mo, so[1]);
        if Toan[2]<>BL then
            write(g,' ',so[3],Toan[2],so[4],dong,BL)
        else write(g,' ',so[3],dong,BL);
        { moi dong viet 10 doan }
        if i mod 10 = 0 then writeln(g);
    end;
    close(g);
end;
BEGIN
    Doc; qsort(1,n); Ghi;
END.

```

// C#

```

using System;
using System.IO;
using System.Collections;
namespace SangTao1
{
    class SapDoan
    {
        static public string fn = "Doan.inp";
        static public string gn = "Doan.out";
        static public string s; // du lieu vao
        static public Doan[] d = new Doan[5000]; // cac doan
        static public int n = 0; // so luong doan
        static public char Ket = '#';
        static void Main(string[] args)
        {
            s = File.ReadAllText(fn) + Ket.ToString();
            Console.WriteLine("\n Du lieu " +
                "truoc khi xu li:\n " + s);
            n = DocDoan();
            Console.WriteLine("\n Tong cong " + n + " Doan");
            Console.WriteLine(n); Printd();
            QSort(d, 0, n-1);
            Console.WriteLine("\n Da sap: "); Printd();
            Ghi(); XemLai();
            Console.WriteLine("\n Fini ");
            Console.ReadLine();
        }
        static public void XemLai()
        {
            Console.WriteLine("\n Kiem tra lai:\n");
            Console.WriteLine("\n Input:\n" +
                File.ReadAllText(fn));
            Console.WriteLine("\n Output:\n" +
                File.ReadAllText(gn));
        }
        static public int DocDoan()
        {

```

```

int n = -1;
int q = 0; // trạng thái
int i = 0; // biến trỏ trong s
int dau, cuoi;
for (; s[i] != Ket; ++i)
{
    switch(q)
    {
        case 0: // Tìm dấu đoạn
            if (GapMo(s[i]))
            {
                ++n; d[n] = new Doan();
                d[n].Mo = s[i]; q = 1;
            }
            break;
        case 1: // Doc so1
            if (GapSo(s[i]))
                d[n].Sol = DocSo(ref i);
            else if (GapToan(s[i]))
                {d[n].Toan1 = s[i]; q = 2;}
            else if (s[i]=='(',')' q = 3;
            break;
        case 2: // Doc so2
            if (GapSo(s[i]))
                d[n].So2 = DocSo(ref i);
            else if (s[i]=='(',')' q = 3;
            break;
        case 3: // Doc so3
            if (GapSo(s[i]))
                d[n].So3 = DocSo(ref i);
            else if (GapToan(s[i]))
            {
                d[n].Toan2 = s[i]; q = 4;
            }
            else if (GapDong(s[i]))
                q = 5;
            break;
        case 4: // Doc so4
            if (GapSo(s[i]))
                d[n].So4 = DocSo(ref i);
            else if (GapDong(s[i]))
                q = 5;
            break;
        case 5: // Xong 1 đoạn
            d[n].Dong = s[--i];
            dau = d[n].Sol;
            if (d[n].Toan1 == '+')
                dau += d[n].So2;
            else if (d[n].Toan1 == '*')
                dau *= d[n].So2;
            cuoi = d[n].So3;
            if (d[n].Toan2 == '+')
                cuoi += d[n].So4;
    }
}

```

```

        else if (d[n].Toan2 == '*')
            cuoi *= d[n].So4;
        d[n].Len = cuoi-dau;
        Console.WriteLine("\n Doan "+
                           n + ". ");
        d[n].Print(); q = 0; break;
    }
} // endfor
return (++n);
}
static public bool GapSo(char c)
{ return (c >= '0' && c <= '9'); }
static public bool GapMo(char c)
{ return (c == '(' || c == '['); }
static public bool GapDong(char c)
{ return (c == ')' || c == ']'); }
static public bool GapToan(char c)
{ return (c == '+' || c == '*'); }
static public int DocSo(ref int i)
{
    int m = 0;
    do
    { m = m * 10 + s[i++] - '0';
    } while (GapSo(s[i]));
    --i;
    return m;
}
static public void Printd()
{ for (int i = 0; i < n; ++i) d[i].Print(); }
static public void Ghi()
{
    StreamWriter g = File.CreateText(gn);
    for (int i = 0; i < n; ++i) d[i].FWrite(g);
    g.Close();
}
static public void QSort(Doan[] d, int s, int e)
{
    int i = s, j = e, m = d[(i+j)/2].Len;
    Doan t;
    while (i <= j)
    {
        while (d[i].Len < m) ++i;
        while (d[j].Len > m) --j;
        if (i <= j)
        {
            t = d[i]; d[i] = d[j]; d[j] = t;
            ++i; --j;
        }
    }
    if (s < j) QSort(d, s, j);
    if (i < e) QSort(d, i, e);
}
} // SapDoan

```

```

public class Doan
{
    public char Mo;
    public char Dong;
    public int So1;
    public int So2;
    public int So3;
    public int So4;
    public char Toan1;
    public char Toan2;
    public int Len;
    public Doan()
    {
        Mo = '<'; Dong = '>';
        So1 = So2 = So3 = So4 = 0;
        Toan1 = Toan2 = '#';
        Len = 0;
    }
    public void FWrite(StreamWriter g)
    {
        g.Write(Mo.ToString());
        if (Toan1 != '#')
            g.Write(So1 + Toan1.ToString() + So2);
        else g.Write(So1);
        g.Write(",");
        if (Toan2 != '#')
            g.Write(So3 + Toan2.ToString() + So4);
        else g.Write(So3);
        g.WriteLine(Dong.ToString());
    }
    public void Print()
    {
        Console.Write(Mo.ToString());
        if(Toan1!='#')
            Console.Write(So1+Toan1.ToString()+So2);
        else Console.Write(So1);
        Console.Write(",");
        if(Toan2!='#')
            Console.Write(So3+Toan2.ToString()+So4);
        else Console.Write(So3);
        Console.WriteLine(Dong.ToString()+
            " Len = "+Len);
    } // Print
} // Doan
} // SangTao1

```

CHƯƠNG 3

BÀN PHÍM VÀ MÀN HÌNH

Bài 3.1. Bảng mã ASCII

Sinh tệp có tên ASCII.DAT chứa mã ASCII để tiện dùng.

Chú ý

ASCII (đọc là a-ski) là bộ *mã chuẩn* dùng trong trao đổi thông tin của Mĩ và đầu tiên được cài đặt trong các máy tính sử dụng hệ điều hành MS-DOS. Trong bảng mã này, mỗi kí tự có một mã số riêng biệt chiếm 1 byte. Trong TP Ta viết 65 là để biểu thị mã số 65, viết #65 là để biểu thị kí tự có mã số 65, tức là chữ 'A'. Các kí tự mang mã số từ 0 đến 31 là các kí tự điều khiển, thí dụ, kí tự #13 điều khiển con trỏ văn bản xuống dòng mới, kí tự #10 điều khiển con trỏ văn bản về đầu dòng. Như vậy, xâu kí tự #13#10 sẽ điều khiển con trỏ về đầu dòng mới và do đó lệnh **write(#13#10)** sẽ tương đương với lệnh **writeln**. Lệnh **writeln(#13#10)** sẽ tương đương với hai lệnh **writeln**;

Chương trình dưới đây ghi vào tệp văn bản có tên ASCII.DAT các kí tự và mã của chúng. Tất cả có 256 kí tự chia làm hai phần. 128 kí tự đầu tiên mã số từ 0 đến 127 là *các kí tự cơ sở*, 128 kí tự còn lại, mã số từ 128 đến 255 là *các kí tự mở rộng*.

Sau khi thực hiện chương trình, bạn có thể mở tệp ASCII.DAT để xem từng kí tự và mã của chúng. Lưu ý rằng có kí tự hiển thị được và có kí tự không hiển thị được trên màn hình, chẳng hạn như các kí tự điều khiển.

(* Pascal *)

```
program ASCII;  
uses crt;  
procedure ASCII;  
var f: text; i: byte;  
begin  
  assign(f, 'ASCII.DAT');  
  rewrite(f);  
  for i := 0 to 255 do
```

```

begin
    write(f,chr(i),': ',i:3,' ');
    if i mod 5 = 0 then writeln(f);
end;
close(f);
writeln('OK'); readln;
end;
BEGIN
    ASCII;
END.

```

// C#

Chương trình dưới đây lưu lại mã của 128 kí tự đầu tiên ứng với phần cơ sở của bảng mã ASCII. Các kí tự phân mở rộng phụ thuộc vào từng phiên bản cài đặt của các hệ điều hành.

```

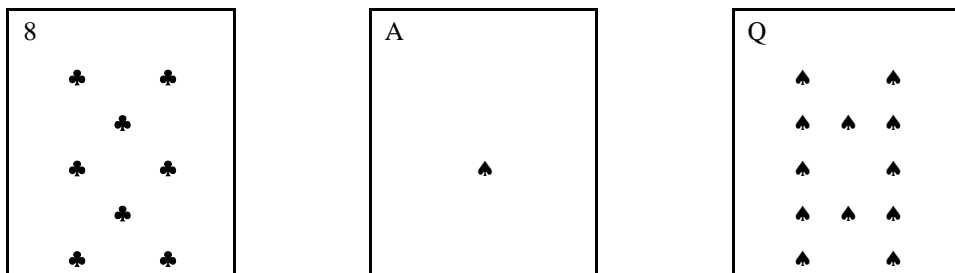
using System;
using System.IO;
namespace SangTao1
{
    class ASCII
    {
        static void Main()
        {
            string fn = "ASCII.TXT";
            StreamWriter g = File.CreateText(fn);
            for (int i = 0; i < 128; ++i)
                g.WriteLine("{0}: {1}", i, (char)i);
            g.Close();
            Console.WriteLine(File.
                ReadAllText(fn)); // Doc lai
            Console.ReadLine();
        }
    } // class
} // space

```

Bài 3.2. Bộ Tú lơ khơ

Lập chương trình hiển thị trên màn hình các quân bài Tú lơ khơ gồm Rô, Cơ, Pích, Nhép theo quy định quân A mang mã số 1 và có 1 hình đơn vị, các quân mã số i từ 2 đến 10 có i hình đơn vị, các quân J, Q và K lần lượt có 11, 12 và 13 hình đơn vị tương ứng. Hình đơn vị gồm bốn loại kí tự có mã ASCII tương ứng như sau:

♦ (Rô) : #4, ♥ (Cơ) : #3, ♠ (Pích) : #6, ♣ (Nhép) : #5.



8	A	Q
---	---	---

Ba quân bài Tú lơ khơ

Gợi ý

Trước hết ta cần thống nhất một số quy định sau:

- ♦ Quân bài được vẽ bằng một màu M tùy chọn.
- ♦ Nếu là quân Rô hoặc Cơ ta đặt màu chữ là đỏ (RED), với các quân Pích và Nhép ta đặt màu chữ là đen (BLACK).
- ♦ Mỗi quân bài có hai thuộc tính là loại (Rô, Cơ, Pích hoặc Nhép) và mã số. Mã số của quân A là 1, J là 11, Q là 12 và K là 13. Các quân còn lại mang mã số từ 2 đến 10 ứng với số ghi trên quân bài đó.
- ♦ Trên nền các quân bài J, Q và K không vẽ hình người mà vẽ số lượng hình đơn vị (Rô, Cơ, Pích hoặc Nhép) tương ứng với mã số của quân đó.

Để bố trí số lượng hình đơn vị trên mỗi quân bài cho cân đối ta cần 5 dòng. Thủ tục Dong (q:char; s:string; x,y:byte) vẽ 5 dòng chứa hình đơn vị loại q, bắt đầu tính từ tọa độ (x, y) ứng với vị trí góc trên trái của quân bài trên màn hình, theo dấu hiệu ghi trong xâu mẫu s. Thí dụ, lời gọi với xâu mẫu s = '20302' sẽ vẽ 5 dòng thể hiện cho quân mang mã số 7 thuộc loại v (Rô, Cơ, Pích hoặc Nhép) như sau:

1. Dòng thứ nhất có 2 kí tự v.
2. Dòng thứ hai có 0 kí tự v tức là để trống.
3. Dòng thứ ba có 3 kí tự v.
4. Dòng thứ tư có 0 kí tự v tức là để trống.
5. Dòng thứ năm có 2 kí tự v.

Vì trong xâu mẫu s tổng cộng có $2 + 3 + 2 = 7$ kí tự v nên quân bài mang mã số 7.

```

procedure Dong(v: char; s: string; x,y: byte);
var i: byte;
begin
  x := x+3; y := y+TY;
  for i := 1 to 5 do
    begin
      gotoxy(x,y);
      case s[i] of
        '1': write(BL,BL,v,BL,BL);
        '2': write(v,BL,BL,BL,v);
        '3': write(v,BL,v,BL,v);
      end;
      y := y+TY;
    end;
  end;

```

Các mẫu dòng s được tính toán trước và khởi trị như sau:

```

MauDong: array[1..13] of string[5] =
  ('00100', '01010', '10101', '20002', '20102',
   '20202', '20302', '21212', '30303', '22222',
   '22322', '23232', '23332');

```

Ta dễ dàng nhận ra có tất cả 13 mẫu dòng ứng với 13 mã số 1(A), 2,..., 10, 11(J), 12(Q) và 13(K). Tóm lại mẫu dòng thứ i cho ta phương thức vẽ i hình đơn vị trên quân bài mang mã số i. Mỗi mẫu dòng được biểu diễn qua một xâu 5 kí tự.

Các thủ tục điều khiển màn hình có ý nghĩa như sau:

gotoxy(x,y) : Chuyển con trỏ màn hình đến cột x dòng y .

TextColor(c) : Đặt màu c cho nét chữ. Thí dụ, kể từ sau khi gặp lệnh **TextColor(BLACK)** các kí tự xuất hiện trên màn hình sẽ có nét màu đen,

TextBackground(m) : Đặt màu m cho nền chữ. Thí dụ, kể từ sau khi gặp lệnh **TextBackground(WHITE)** các kí tự sẽ được viết trên nền trắng.

textattr: Biến hệ thống có giá trị 1 byte, tính từ phải qua trái, 4 bit đầu tiên (gọi là các bit thấp) tạo thành một số nguyên thể hiện màu cho nét chữ, 4 bit tiếp theo (gọi là các bit cao) thể hiện màu cho nền chữ. Thí dụ phép gán **textattr:=7** sẽ được nhận giá trị nhị phân là (0000)(0111) và do đó hệ thống sẽ đặt màu nét chữ là 7 (màu trắng) và màu nền chữ là 0 (màu đen). Như vậy phép gán trên tương đương với tổ hợp của hai lệnh **TextColor** và **TextBackground**.

Lệnh **write(a:m)** hiển thị đơn vị dữ liệu a với độ rộng m vị trí. Nếu chiều dài dữ liệu của a nhỏ hơn m thì hệ thống tự động điền thêm dấu cách cho đủ m vị trí. Nếu chiều dài dữ liệu của a lớn hơn m thì hệ thống hiển thị đủ vị trí cho a . Thí dụ, lệnh **write(BL:20)** sẽ hiển thị 20 dấu cách trên màn hình.

Vì màn hình trong hệ điều hành Windows có độ phân giải cao, khác với màn hình văn bản trong DOS nên thủ tục **VeBai** được cài đặt với tham số điều khiển **Kieu** quy định kiểu của hệ điều hành. **Kieu = Wind** sẽ hiển thị bộ bài trong chế độ Windows, **Kieu = DOS** sẽ hiển thị bộ bài trong chế độ màn hình DOS. Hai kiểu chỉ khác nhau ở một giá trị cần khởi trị cho vài tham số, cụ thể là:

Kích thước quân bài. Nếu coi mỗi quân bài như một hình chữ nhật thì **DX** là chiều rộng, **DY** là chiều dài.

Độ giãn dòng **TX**. Khi hiển thị trên màn hình Windows thì ta để cách hai dòng, **TX = 2**, ngược lại, trên màn hình DOS ta đặt **TX = 1**.

Bảng dưới đây mô tả các tham số cần khởi trị cho hai môi trường WINDOWS và DOS.

WINDOWS		DOS
DX	9	9
DY	12	6
TX	2	1

*Các tham số kích thước quân bài $DX \times DY$ và độ giãn cách dòng **TX** trong môi trường WINDOWS và DOS.*

(* Pascal *)

```
uses crt;
const
    CO = #3; RO = #4; NHEP = #5; PIC = #6;
    WIND = 1; DOS = 2; BL = #32;
    DX: byte = 9;
    DY: byte = 12; {kích thước quân bài}
    TY: byte = 2;
    MauDong: array[1..13] of string[5] = ('00100',
                                           '01010',
                                           '10101',
                                           '20002',
                                           '20102',
```

```

'20202',
'20302',
'21212',
'30303',
'22222',
'22322',
'23232',
'23332');
Nhan: array[1..13] of string[2]
      = ('A', '2', '3', '4', '5',
         '6', '7', '8', '9', '10',
         'J', 'Q', 'K');
procedure Dong(q: char; s: string; x, y: byte); tự viết
{-----}
    Ve nen mau M cho quan bai
    tai vi tri goc tren trai (x,y)
    -----}
procedure Nen(M, x, y: byte);
var i: byte;
begin
    TextBackGround(M);
    for i:= 0 to DY do
        begin
            gotoxy(x+1, y+i);
            write(BL:DX);
        end;
    end;
{-----}
Ve 1 quan bai kieu q (ro, co, bich, nhiep);
so n (2 ... 10; 1 = A; 11 = J; 12 = Q; 13 = K)
goc Tay-Bac tai cot x, dong y cua man hinh,
-----}
procedure VeQuanBai(q: char; n, x, y: byte);
var i, j: byte;
begin {VeQuanBai}
    if (q = RO) OR (q = CO) then TextColor(RED)
    else TextColor(BLACK);
    Nen(WHITE, x, y);
    Dong(q, MauDong[n], x, y);
    {viet so}
    gotoxy(x+1, y+1); write(Nhan[n]:2);
    gotoxy(x+DX-1, y+DY-1); write(Nhan[n]);
end;
Procedure VeBai(Kieu: byte);
var i: integer;
begin
    if Kieu = DOS then
        begin
            DY:= 6;
            TY:= 1;
        end else
        if Kieu = WIND then
            begin

```

```

        DY:= 12;
        TY:= 2;
    end else
    begin
        writeln('Dat kieu khong dung');
        write('Cach goi thu tuc: ');
        writeln('VeBai(WIND) hoac VeBai(DOS)');
        readln; halt;
    end;
textbackground(BLUE); clrscr;
for i := 1 to 13 do {Ve bo Tu lo kho}
    begin
        VeQuanBai(RO,i,5,10);
        VeQuanBai(CO,i,20,10);
        VeQuanBai(PIC,i,35,10);
        VeQuanBai(NHEP,i,50,10);
        if ReadKey=#27 then halt;
    end;
textattr:= 7; clrscr;
end;
BEGIN
    VeBai(WIND);
END.

```

// C#

```

using System;
using System.IO;
namespace SangTao1
{
    /*-----
    *           Bo bai Tulokho
    * -----*/
    class TuLoKho
    {
        static void Main()
        {
            BoBai b = new BoBai();
            b.Draw(6, 4);
            Console.ReadLine();
        }
    } // Class

    /*-----
    *           Mo ta bo bai Tulokho
    * -----*/
    class BoBai
    {
        private char CO = (char)3;
        private char RO = (char)4;
        private char NHEP = (char)5;
        private char PIC = (char)6;
    }
}

```

```

const int DX = 9;
const int DY = 6; // kích thước quan bài
// 1 khoảng cách giữa 2 dòng
const int TY = 1;
const int SOQUAN = 13;
private string[] MauDong = {"00100",
                             "01010",
                             "10101",
                             "20002",
                             "20102",
                             "20202",
                             "20302",
                             "21212",
                             "30303",
                             "22222",
                             "22322",
                             "23232",
                             "23332" };
private string[] Nhan = {"A", "2", "3", "4", "5", "6", "7",
                         "8", "9", "10", "J", "Q", "K"};
// Dữ liệu màu nền và text cho màn hình
private void SetColors(ConsoleColor bg, ConsoleColor fg)
{
    Console.BackgroundColor = bg;
    Console.ForegroundColor = fg;
}
// Viết s tại cột x, dòng y
private void WriteAt(string s, int x, int y)
{
    Console.SetCursorPosition(x, y);
    Console.Write(s);
} // WriteAt
// Vẽ bộ bài tại vị trí x, y
public void Draw(int x, int y)
{
    int DD = DX + 10;
    Console.BackgroundColor =
        ConsoleColor.Blue;
    Console.Clear();
    for (int i = 0; i < SOQUAN; ++i)
    {
        VeQuanBai(RO, i, x, y);
        VeQuanBai(CO, i, x + DD, y);
        VeQuanBai(PIC, i, x + 2 * DD, y);
        VeQuanBai(NHEP, i, x + 3 * DD, y);
        Console.ReadLine();
    }
    Console.ResetColor(); // trả lại nền cũ
} // Draw
/*-----
                        Vẽ 5 dòng trong quan bài
-----*/
private void Lines(char q, string s,

```

```

                                int x, int y)
{
    const string BL = " ";
    string qs = q.ToString();
    x += 3;
    for (int i = 0; i < 5; ++i)
    {
        y += TY;
        Console.SetCursorPosition(x, y);
        switch (s[i])
        {
            case '1':
                Console.WriteLine(BL + BL + qs + BL + BL);
                break;
            case '2':
                Console.WriteLine(qs + BL + BL + BL + qs);
                break;
            case '3':
                Console.WriteLine(qs + BL + qs + BL + qs);
                break;
        } // switch
    } // for
}
// Dat mau nen cho quan bai
private void Nen(ConsoleColor m,
                 int x, int y)
{
    string s = new string(' ', DX);
    Console.BackgroundColor = m;
    for (int i = 0; i <= DY; ++i)
        WriteAt(s, x + 1, y + i);
}

/*-----
Ve 1 quan bai kieu q (ro, co, bich, nhiep);
so n (1 ... 10; 0 = A; 10 = J; 11 = Q; 12 = K)
goc Tay-Bac tai cot x, dong y cua man hinh,
-----*/
private void VeQuanBai(char q, int n, int x, int
y)
{
    // Chon mau chu RO, CO: mau do
    // PIC, NHEP: mau den
    Console.ForegroundColor =
        (q == RO || q == CO)
        ? ConsoleColor.Red
        : ConsoleColor.Black;
    // Dat nen quan bai mau trang
    Nen(ConsoleColor.White, x, y);
    // Ve 5 dong
    Lines(q, MauDong[n], x, y);
    // Viet so o goc tren-tra

```

```

WriteAt(Nhan[n], x + 2, y); // + 1);
// Viet so o goc duoi-phai
if (n == 9)
WriteAt(Nhan[n], x + DX - 2, y + DY );
else WriteAt(Nhan[n], x + DX - 1, y + DY );
} // VeQuanBai
} // TuLoKho
} // SangTao1

```

Chú thích

Các tham số **x**, **y** và **DX**, **DY** phụ thuộc vào độ phân giải màn hình. Bạn cần điều chỉnh các tham số này cho phù hợp với chế độ phân giải màn hình đã chọn.

Bài 3.3. Hàm *GetKey*

Mỗi khi ta nhấn một phím, trong vùng đệm 2 byte sẽ được nạp 1 hoặc 2 byte tùy theo kiểu phím đã nhấn. Nếu là phím thường như a, b, c, %, \$,... trong vùng đệm sẽ được nạp 1 byte chứa mã ASCII của kí tự tương ứng. Nếu ta nhấn phím mở rộng như F1,..., F10, các phím dịch chuyển con trỏ, ↑, →, ←, ↓, Ins (chèn), Del (xoá), PageUp/PgUp (lên một trang), PageDown/PgDn (xuống một trang),... trong vùng đệm sẽ được nạp hai byte, byte thứ nhất có giá trị 0, byte thứ hai chứa mã riêng của phím đã nhấn. Mã riêng này có thể trùng với mã của các kí tự thường. Thí dụ, khi ta nhấn phím mở rộng F10 trong vùng đệm sẽ được nạp 2 byte (0, 68). Mã riêng 68 trùng với mã của kí tự D. Hàm *ReadKey* cho ta kí tự của phím đã nhấn và không hiển thị kí tự đó (trên màn hình), ta gọi là hàm nhận thắm một kí tự. *ReadKey* trước hết kiểm tra vùng đệm bàn phím xem còn byte nào chưa được đọc không. Nếu còn, *ReadKey* sẽ đọc byte đó. Ngược lại, nếu vùng đệm trống, *ReadKey* sẽ chờ để ta nhấn một phím rồi sau đó đọc 1 byte từ vùng đệm.

Hãy viết hàm *GetKey* cho ra mã ASCII của phím thường đã nhấn và cho ra mã riêng của phím mở rộng cộng thêm 128 nhằm phân biệt được phím thường với phím mở rộng.

Chú ý: Hàm **GetKey** ở bài 3.3 cho mã của một số phím mở rộng dùng để điều khiển con trỏ màn hình như sau:

LEN:	200	Mũi tên trở lên ↑
XUONG:	208	Mũi tên trở xuống ↓
PHAI:	205	Mũi tên trở qua phải →
TRAI:	203	Mũi tên trở qua trái ←

ESC (27) và **ENTER/RETURN (13)** là những phím thường.

Gợi ý

Trước hết gọi hàm **c := ReadKey** rồi kiểm tra giá trị của kí tự **c**. Nếu **c** có mã 0 tức là đã nhấn phím mở rộng, ta cần đọc tiếp byte thứ hai và gán cho hàm giá trị của byte đó cộng thêm dấu hiệu nhận biết phím mở rộng là 128. Nếu **c** có mã khác 0, ta gán cho hàm giá trị đó.

```

(* Pascal *)
(* -----
      Ham GetKey
-----*)
program Conio;

```

```

uses crt;
const Esc = 27;
Function GetKey: integer;
var c: char;
begin
    c:= ReadKey;
    if c <> #0 then GetKey := Ord(c)
    else GetKey := Ord(ReadKey) + 128;
end;
Procedure Test;
var k: integer;
begin
    repeat
        write(' Nhan Phim (Bam ESC de thoat): ');
        k:= GetKey;
        if k > 128 then
            writeln(' Phim mo rong (0, ',k-128,') ==> ',k)
        else
            writeln(' Phim thuong ',chr(k), '(',k,')');
        until k = Esc;
    readln;
end;
BEGIN
    Test;
END.

```

Bài 3.4. Trò chơi 15

Có 15 quân cờ được đánh mã số từ 1 đến 15 được đặt trong một bàn cờ hình vuông 4×4 ô theo hình trạng ban đầu như trong hình. Mỗi bước đi, ta được phép di chuyển một quân nằm cạnh ô trống vào trong ô trống.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Trò chơi 15

Viết chương trình thực hiện hai chức năng sau đây:

- Đào ngẫu nhiên các quân cờ để chuyển từ hình trạng ban đầu về một hình trạng H nào đó.
- Nhận phím điều khiển của người chơi rồi di chuyển quân cờ theo phím đó. Khi nào người chơi đạt được hình trạng ban đầu thì kết thúc một ván.

Trò chơi này có tên là Trò chơi 15, từng nổi tiếng ở thế kỉ XIX như trò chơi Rubic ở thời đại chúng ta vậy.

Gợi ý

Trò chơi này khá dễ lập trình. Bạn cần lưu ý sự khác biệt giữa vị trí của phần tử $a[i, j]$ trong ma trận a với vị trí hiển thị của nó trên màn hình, vì thủ tục $gotoxy(i, j)$ đưa con trỏ màn hình đến cột i , dòng j trong khi $a[i, j]$ lại truy cập tới dòng (i) và cột (j). Sự khác biệt này được tính đến trong thủ tục Den (đến - chuyển con trỏ đến vị trí thích hợp để hiển thị phần tử $a[i, j]$).

Mảng b chứa hình trạng ban đầu của bàn cờ và dùng để kiểm tra xem mảng a có trùng với hình trạng này không (xem hàm logic $Dung$).

Hai thủ tục $DaoNgauNhiem$ và $Game15$ đều cùng gọi đến thủ tục $Chuyen(k)$ - dịch chuyển một trong bốn quân sát với ô trống vào ô trống. Ta quy ước chọn các giá trị của k là:

0: Lên - chuyển quân nằm sát dưới ô trống vào ô trống.

1: Xuống - chuyển quân nằm sát trên ô trống vào ô trống.

2: Phải - chuyển quân nằm sát trái ô trống vào ô trống.

3: Trái - chuyển quân nằm sát phải ô trống vào ô trống.

Đương nhiên, nếu ô trống nằm sát đường biên thì có thể có trường hợp không chuyển được.

Ta phân biệt phần tử (i, j) của mảng a với vị trí hiển thị giá trị $a[i, j]$ của phần tử đó trên màn hình như sau. Gọi (x, y) là vị trí góc trên trái của vùng hiển thị, dx và dy là chiều dài hai cạnh của ô sẽ hiển thị giá trị $a[i, j]$, khi đó thủ tục $Den(i, j)$ dưới đây chuyển con trỏ màn hình đến vị trí hiển thị được mô tả như sau:

```
procedure Den(i, j: byte);
begin
  Gotoxy(y+(j-1)*dy, x+(i-1)*dx);
end;
```

Khi đó thủ tục $Viet(i, j: byte)$ viết giá trị $a[i, j]$ vào ô tương ứng trên màn hình sẽ thực hiện các thao tác sau. Trước hết cần chuyển con trỏ màn hình đến vị trí cần viết bằng thủ tục $Den(i, j)$. Sau đó xét ô $a[i, j]$. Nếu đó là ô trống ($a[i, j] = 0$) thì xóa ô đó, ngược lại ta ghi giá trị của $a[i, j]$ vào ô cần hiển thị.

```
procedure Viet(i, j: byte);
begin
  Den(i, j);
  if a[i, j] = 0 then
    begin
      TextBackground(YELLOW);
      write(BL:2);
      TextBackground(BLUE);
    end
  else write(a[i, j]:2);
end;
```

Khi khởi động chương trình sẽ hiển thị trò chơi và tiến hành đảo ngẫu nhiên các quân cờ cho đến khi người chơi nhấn một phím tùy ý. Từ thời điểm đó, người chơi sẽ tìm cách di chuyển các quân cờ để đạt tới hình trạng ban đầu.

```
(* Pascal *)
(*-----
                Game 15
-----*)
uses crt;
const
```

```

BL = #32;
DD = 4;
x = 2; y = 3; {Goc Tay-Bac cua ban co}
dx = 2; dy = 3; {Khoang cach giua cac o}
{cac ma dich chuyen con tro}
LEN = 200;
XUONG = 208;
PHAI = 205;
TRAI = 203;
var
a, b: array [1..DD,1..DD] of byte; {ban co}
xx, yy: byte; {Toa do cua con tro}

{-----
  Nhan tham 1 ki tu
-----}
Function GetKey: integer;
var c: char;
begin
  c:= ReadKey;
  if c <> #0 then GetKey:= Ord(c)
  else
    begin
      c:= ReadKey;
      GetKey:= Ord(c) + 128;
    end;
end;

{-----
Chuyen con tro man hinh
den vi tri hien thi
quan co (i,j)
-----}
procedure Den(i,j: byte);
begin
  Gotoxy(y+(j-1)*dy,x+(i-1)*dx);
end;

{-----
Viet gia tri cua quan co
a[i,j] vao o tuong ung
-----}
procedure Viet(i,j: byte);
begin
  Den(i,j);
  if a[i,j] = 0 then
    begin
      TextBackGround(YELLOW);
      write(BL:2);
      TextBackGround(BLUE);
    end
  else write(a[i,j]:2);
end;

{-----
Khoi tri:
1. Dat mau chu, mau nen

```

```

2. Ve ban co
-----}
procedure Init;
var i, j, k: byte;
begin k := 0;
    {nen ngoai mau vang}
    TextBackGround(YELLOW);
    Gotoxy(1,1);
    { Ve cac o trong }
    for i:= 1 to dx*DD+1 do
        begin
            for j := 1 to dy*DD+3 do write(BL);
            writeln;
        end;
    TextBackGround(BLUE); {nen ban co mau xanh}
    TextColor(WHITE); {chu trang}
    {Khoi tri va hien thi cac quan co}
    for i:= 1 to DD do
        for j:= 1 to DD do
            begin
                inc(k); a[i,j]:= k;
                b[i,j]:= k; Viet(i,j);
            end;
    a[DD,DD]:= 0; b[DD,DD]:= 0;
    Viet(DD,DD); Den(DD,DD);
    xx:= DD; yy:= DD;
end;
{-----}
Di chuyen quan co a[xx,yy]
vao o trong ke no
-----}
procedure Chuyen(k: integer);
begin
    case k of
    0: {LEN}
        if xx < DD then
            begin
                a[xx,yy]:= a[xx+1,yy];
                Viet(xx,yy);
                inc(xx); a[xx,yy]:= 0;
                Viet(xx,yy);
            end;
    1: {XUONG}
        if xx > 1 then
            begin
                a[xx,yy]:= a[xx-1,yy];
                Viet(xx,yy);
                dec(xx); a[xx,yy]:= 0;
                Viet(xx,yy);
            end;
    2: {PHAI}
        if yy > 1 then
            begin

```

```

        a[xx,yy] := a[xx,yy-1];
        Viet(xx,yy);
        dec(yy); a[xx,yy] := 0;
        Viet(xx,yy);
    end;
3: {TRAI}
    if yy < DD then
    begin
        a[xx,yy] := a[xx,yy+1];
        Viet(xx,yy);
        inc(yy); a[xx,yy] := 0;
        Viet(xx,yy);
    end;
end;
{-----}
    Dao ngau nhien cac quan co
{-----}
procedure DaoNgauNhien;
var c: integer;
begin
    repeat
        Chuyen(random(4));
        Delay(50); {Dat do tre de kip quan sat}
    until KeyPressed;
    c:= GetKey;
end;
{-----}
    Kiem tra ban co a co
    trung voi cau hinh chuan ?
{-----}
function Dung: Boolean;
var i, j: byte;
begin
    Dung:= FALSE;
    for i:= 1 to DD do
        for j:= 1 to DD do
            if (a[i,j] <> b[i,j]) then exit;
        Dung:= TRUE;
    end;
procedure Game15;
var    k: integer;
        d, v: longint;
        sx, sy, ex, ey: byte;
begin
    Randomize;
    ClrScr; d:= 0; v:= 1;
    TextBackGround(BLUE);
    TextBackGround(BLUE);
    TextColor(WHITE);
    Init;
    gotoxy(5,25); clreol;
    write(' Van: ');

```

```

sx:= Wherex; sy:= Wherey;
write(v); write(' Tong so buoc: ');
ex:= Wherex; ey:= Wherey;
DaoNgauNhien;
repeat
  gotoxy(sx,sy); write(BL:10);
  gotoxy(sx,sy); write(v);
  gotoxy(ex,ey); clreol; {xoa cho den het dong}
  gotoxy(ex,ey); write(d);
  Den(xx,yy);
  k:= GetKey;
  case k of
    LEN: k:= 0;
    XUONG: k:= 1;
    PHAI: k:= 2;
    TRAI: k:= 3;
  end;
  Chuyen(k); inc(d);
  if Dung then
    begin
      DaoNgauNhien;
      inc(v); d:= 0;
      gotoxy(sx,sy); write(BL:10);
      gotoxy(sx,sy); write(v);
    end;
  until UpCase(chr(k)) in [#27, 'Q'];
  Textattr := 7; clrscr;
end;
BEGIN
  Game15;
END.

```

Bài 3.5. Bảng nhảy

Bảng nhảy bước b , bậc k là một tám bảng có đặc tính kì lạ sau đây: nếu bạn viết lần lượt lên bảng n số nguyên thì sau khi viết số thứ i , số thứ $(i - b)$ đã viết trước đó sẽ được tăng thêm k đơn vị mà ta gọi là nhảy số.

Với mỗi cặp số nguyên dương b và k cho trước hãy lập trình để biến màn hình máy tính của bạn thành một bảng nhảy sau đó thử viết lên tám bảng đó để nhận được dãy N số tự nhiên đầu tiên $1\ 2\ \dots\ N$ với mỗi N cho trước.

Thí dụ, để thu được dãy số $1\ 2\ \dots\ 10$ trên bảng nhảy bước $b = 3$ bậc $k = 6$ bạn cần viết dãy số sau:

-5 -4 -3 -2 -1 0 1 8 9 10

Gợi ý

Với mỗi bước b ta cần lưu lại b giá trị nạp trước đó trong đoạn $a[0..b-1]$ của mảng a đồng thời với vị trí hiện thị trên màn hình của các giá trị đó trong các mảng tương ứng x và y . Ta sử dụng số học đồng dư cho việc lưu trữ này, cụ thể là khi cần nạp phần tử thứ i trước hết ta nạp vào một biến đệm z . Sau đó ta tăng phần tử $a[i \bmod b]$ thêm k đơn vị và tìm đến cột $x[i \bmod b]$, dòng $y[i \bmod b]$ để cập nhật lại giá trị này. Cuối cùng ta chuyển giá trị z vào $a[i \bmod b]$. Nói cách khác ta xử lí vùng đệm $a[0..(b-1)]$ theo nguyên tắc vòng tròn. Các chi tiết xử lí màn hình trong trường hợp chuyển dòng và cuộn màn hình khi thao tác ở dòng cuối màn hình là đơn giản và được chỉ rõ trong chương trình

(* Pascal *)

```

uses crt;
const
  MN = 50;
  d = 6; {chieu dai cua moi so}
  ML = 12; {so luong tren mot dong}
  LIM = d*ML; BL = #32;
  W = 500; {kich thuoc toi da cua bang nhay}
var
  a: array [0..MN] of integer; {vung dem}
  {toa do con tro man hinh}
  x, y: array [0..MN] of byte;
  {-----}
  Viet n so tren bang nhay bac k buoc b
  {-----}
procedure BangNhay(b,k,n: integer);
var   i, j, z, t: integer;
      xx, yy: byte; {vi tri con tro}
begin
  textattr := 7; clrscr;
  writeln('Bang nhay bac ',k,' buoc ',b);
  writeln(' gom ',n,' so');
  writeln('Bat dau nap day ',n,' so. ');
  writeln('Sau moi so bam ENTER');
  xx:= wherex; yy:= wherey;
  for i := 0 to n-1 do
  begin
    gotoxy(xx,yy); readln(z); {nap 1 so}
    if i < b then t := i
    else
    begin
      t:= i MOD b;
      for j:= 1 to 5 do
      begin {sua lai so truoc do b buoc}
        gotoxy(x[t],y[t]); write(BL:d);
        gotoxy(x[t],y[t]); write(a[t]);
        delay(W);
        gotoxy(x[t],y[t]); write(BL:d);
        gotoxy(x[t],y[t]); write(a[t]+k);
        delay(W);
      end;
    end;
    x[t] := xx; y[t]:= yy;
    a[t] := z; xx:= xx + d;
    if xx > LIM then

```

```

begin
  xx:= 1;
  if yy < 24 then inc(yy)
  else
    begin
      gotoxy(1,25); writeln;
      for j := 0 to b do dec(y[j]);
    end;
  end;
  gotoxy(xx,yy); write(' KET ');
  readln;
end;
BEGIN
  BangNhay(3,6,10);
  (* Loi giai: -5 -4 -3 -2 -1 0 1 8 9 10 *)
END.

```

// C#

Các bài 3.3, 3.4 và 3.5 là đơn giản. Trong C# có hàm `ReadKey()` cho ra giá trị kiểu `ConsoleKeyInfo`. Dựa theo giá trị này ta có thể xác định phím nào đã được bấm. Đoạn trình dưới đây minh họa khá chi tiết việc nhận biết các phím.

```

// Minh hoa ham Console.ReadKey()
using System;
using System.Text;
class Sample
{
  public static void Main()
  {
    Test();
  }
  public static void Test()
  {
    ConsoleKeyInfo k;
    do {
      Console.WriteLine("\n Bam phim ESC de thoat: ");
      k = Console.ReadKey(true);
      char c = k.KeyChar;
      Console.Write(c);
      if (char.IsLetter(c))
        Console.WriteLine(" Chu cai");
      else if (char.IsNumber(c))
        Console.WriteLine(" Chu so");
      else if (char.IsControl(c))
      {
        Console.WriteLine(" Phim dieu khien ");
        switch (k.Key) {
          case ConsoleKey.F1: Console.WriteLine(" F1");
            break;

```

```
case ConsoleKey.F2: Console.Write(" F2");
    break;
case ConsoleKey.F3: Console.Write(" F3");
    break;
case ConsoleKey.F4: Console.Write(" F4");
    break;
case ConsoleKey.F5: Console.Write(" F5");
    break;
case ConsoleKey.F6: Console.Write(" F6");
    break;
case ConsoleKey.F7: Console.Write(" F7");
    break;
case ConsoleKey.F8: Console.Write(" F8");
    break;
case ConsoleKey.F9: Console.Write(" F9");
    break;
case ConsoleKey.F10: Console.Write(" F10");
    break;
case ConsoleKey.F11: Console.Write(" F11");
    break;
case ConsoleKey.F12: Console.Write(" F12");
    break;
case ConsoleKey.Enter: Console.Write(" ENTER");
    break;
case ConsoleKey.Backspace:
    Console.Write(" Lui (Backspace)");
    break;
case ConsoleKey.Home:
    Console.Write(" Home");
    break;
case ConsoleKey.Insert:
    Console.Write(" Ins");
    break;
case ConsoleKey.Delete:
    Console.Write(" Del");
    break;
case ConsoleKey.PageDown:
    Console.Write(" PgDn");
    break;
case ConsoleKey.PageUp:
    Console.Write(" PgUp");
    break;
case ConsoleKey.Pause:
    Console.Write(" Pause - Break");
    break;
case ConsoleKey.RightArrow:
    Console.Write(" Mui ten phai");
    break;
case ConsoleKey.LeftArrow:
    Console.Write(" Mui ten trai");
    break;
case ConsoleKey.DownArrow:
```

```

        Console.Write(" Mui ten xuong");
        break;
    case ConsoleKey.UpArrow:
        Console.Write(" Mui ten len");
        break;
    case ConsoleKey.End: Console.Write(" End");
        break;
    case ConsoleKey.Escape: Console.Write(" ESC");
        Console.ReadKey(true);
        break;
    }
}
} while (k.Key != ConsoleKey.Escape);
}
} // Sample

```

Chương trình C# dưới đây thể hiện trò chơi Gam15 nhằm minh họa một số tính năng quản lý bàn phím và màn hình trong môi trường DOT.NET.

Hàm GetKey() nhận một phím và chuyển các giá trị điều khiển sang *nội mã*, tức là mã do chúng ta tự đặt, thí dụ, ta có thể gán mã cho phím mũi tên trái là 4.

Để khởi tạo cấu hình ban đầu ta sẽ lặp ngẫu nhiên maxk lần với giá trị maxk do người chơi tự chọn.

Các ô trong bàn cờ được thể hiện dưới dạng [xx], trong đó xx là trị số của quân cờ. Ô trống được thể hiện là [] với màu xanh. Thí dụ, cấu hình ban đầu (a) và cấu hình (b) nhận được sau khi đảo ngẫu nhiên một số lần có thể như sau:

[1]	[2]	[3]	[4]
[5]	[6]	[7]	[8]
[9]	[10]	[11]	[12]
[13]	[14]	[15]	[]

(a) Cấu hình ban đầu

[1]	[6]	[2]	[4]
[5]	[10]	[3]	[8]
[]	[14]	[7]	[11]
[9]	[13]	[15]	[12]

(b) Cấu hình sau khi đảo

```

// C#
using System;
using System.IO;
namespace SangTao1
{
    /*-----
    *                               Game15
    * -----*/

    class Game15
    {
        const int scot = 20; // Toa do cot goc
        const int sdong = 8; // Toa do dong goc
        const int dcot = 5; // Kh cach giua 2 o tren dong
        const int ddong = 2; // Khoang cach dong
    }
}

```

```

const int dd = 4; // ban co kich thuoc 4 X 4
const int ddl = dd - 1;
const int LEN = 1;
const int XUONG = 2;
const int PHAI = 3;
const int TRAI = 4;
const int END = 5;
const int ESC = 6;
const string BL = " "; // dau cach
static public int [,] a = new int [dd, dd];
static public int [,] b = new int[dd, dd];
static public int cot = ddl; // toa do o trong
static public int dong = ddl;
static void Main()
{
    Init(); Play();
}
static public void Play()
{
    int k = 0;
    int d = 0;
    if (Console.CursorVisible)
        Console.CursorVisible =
            !Console.CursorVisible;
    Console.SetCursorPosition(1, 1);
    Console.Write("So buoc: ");
    int coty = Console.CursorTop;
    int dongx = Console.CursorLeft;
    Console.SetCursorPosition(dongx, coty);
    Console.Write(d);
    do
    {
        VeO(dong, cot);
        if (Sanhab())
        {
            Console.SetCursorPosition(dongx+2,
                                       coty);
            Console.Write(" Chuc mung " +
                          "Thanh Cong !!!");
            Console.ReadLine();
            return;
        }
        k = GetKey(); ++d;
        Console.SetCursorPosition(dongx, coty);
        Console.Write(" ");
        Console.SetCursorPosition(dongx, coty);
        Console.Write(d);
        switch (k)
        {
            case LEN: // Day quan duoi o trong LEN
                if (dong < ddl)
                {
                    a[dong,cot]=a[dong+1,cot];

```

```

        VeO(dong, cot);
        ++dong; a[dong, cot] = 0;
        VeO(dong, cot);
    }
    break;
case XUONG://Day quan tren o trong XUONG
    if (dong > 0)
    {
        a[dong,cot]=a[dong-1,cot];
        VeO(dong, cot);
        --dong; a[dong, cot] = 0;
        VeO(dong, cot);
    }
    break;
case PHAI: // Day quan TRAI o trong sang
    if (cot > 0)
    {
        a[dong, cot]=a[dong,cot-1];
        VeO(dong, cot);
        --cot; a[dong, cot] = 0;
        VeO(dong, cot);
    }
    break;
case TRAI: // Day quan PHAI o trong sang
    if (cot < ddl)
    {
        a[dong,cot]=a[dong,cot+1];
        VeO(dong, cot);
        ++cot; a[dong, cot] = 0;
        VeO(dong, cot);
    }
    break;
    }
} while (k != ESC);
}
static public void Gotoij(int i,int j)
{
    Console.SetCursorPosition(scot+dcot*j,
                               sdong+ddong*i);
}
static public void VeO(int i, int j)
{
    int dong = sdong + ddong*i;
    int cot = scot + dcot * j;
    Console.SetCursorPosition(cot,dong);
    Console.Write("  ");
    Console.SetCursorPosition(cot, dong);
    if (a[i,j] == 0)
    {
        Console.BackgroundColor =
            ConsoleColor.Blue;
        Console.Write("[ "+BL+BL+"");
        Console.BackgroundColor =

```

```

        ConsoleColor.Black;
    }
    else if (a[i,j] < 10)
        Console.Write "[" + BL + a[i,j] + "];
        else Console.Write "[" + a[i,j] + "];
    }
    // so sanh hai cau hinh a va b
    static public bool Sanhab()
    {
        for (int i = 0; i < dd; ++i)
            for (int j = 0; j < dd; ++j)
                if (a[i,j] != b[i,j]) return false;
        return true;
    }
    // Dao ngau nhien maxk lan
    static public void DaoNgauNhien(int maxk)
    {
        Random r = new Random();
        for (; Sanhab(); ) // Dao den khi a != b
        {
            for (int k = 0; k < maxk; ++k)
            {
                switch (r.Next(4) + 1)
                {
                    case LEN:
                        // Day quan duoi o trong LEN
                        if (dong < dd1)
                        {
                            a[dong,cot]=a[dong+1,cot];
                            ++dong; a[dong, cot] = 0;
                        }
                        break;
                    case XUONG:
                        // Day quan tren o trong XUONG
                        if (dong > 0)
                        {
                            a[dong,cot]=a[dong-1,cot];
                            --dong; a[dong, cot] = 0;
                        }
                        break;
                    case PHAI:
                        // Day quan TRAI o trong sang
                        if (cot > 0)
                        {
                            a[dong,cot]=a[dong,cot-1];
                            --cot; a[dong, cot] = 0;
                        }
                        break;
                    case TRAI: // Day quan PHAI o trong sang
                        if (cot < dd1)
                        {
                            a[dong, cot] = a[dong, cot + 1];
                            ++cot; a[dong, cot] = 0;
                        }
                }
            }
        }
    }

```

```

        }
        break;
    } // switch
    } // for k
} // for sanh
}
static public void VeBanCo()
{
    for (int i = 0; i < dd; ++i)
        for (int j = 0; j < dd; ++j)
            VeO(i,j);
}
static public void Init()
{
    // Khoi tri ban co a.
    // Ban co b dung de doi sanh.
    int k = 1;
    for (int i = 0; i < dd; ++i)
        for (int j = 0; j < dd; ++j)
            b[i,j] = a[i, j] = k++;
    b[ddl,ddl] = a[ddl, ddl] = 0;
    dong = ddl; cot = ddl;
    DaoNgauNhiem(200);
    VeBanCo();
}
static public int GetKey()
{
    ConsoleKeyInfo k;
    k = Console.ReadKey(true);
    char c = k.KeyChar;
    if (char.IsControl(c))
    {
        switch (k.Key)
        {
            case ConsoleKey.RightArrow: return PHAI;
            case ConsoleKey.LeftArrow:  return TRAI;
            case ConsoleKey.DownArrow:  return XUONG;
            case ConsoleKey.UpArrow:     return LEN;
            case ConsoleKey.End:         return END;
            case ConsoleKey.Escape:      return ESC;
        }
    }
    return 0;
}
} // Game15
} // space

```

CHƯƠNG 4

TỔ CHỨC DỮ LIỆU

Bài 4.1. Cụm

Một cụm trong một biểu thức toán học là đoạn nằm giữa hai dấu đóng và mở ngoặc đơn (). Với mỗi biểu thức cho trước hãy tách các cụm của biểu thức đó.

Dữ liệu vào: Tập văn bản **CUM.INP** chứa một dòng kiểu xâu kí tự (string) là biểu thức cần xử lí.

Dữ liệu ra: Tập văn bản **CUM.OUT** dòng đầu tiên ghi *d* là số lượng cụm. Tiếp đến là *d* dòng, mỗi dòng ghi một cụm được tách từ biểu thức. Trường hợp gặp lỗi cú pháp ghi số -1 .

Thí dụ:

CUM.INP	CUM.OUT
$x * (a + 1) * ((b - 2) / (c + 3))$	4 (a+1) (b-2) (c+3) ((b-2) / (c+3))

Gợi ý

Giả sử xâu *s* chứa biểu thức cần xử lí. Ta duyệt lần lượt từ đầu đến cuối xâu *s*, với mỗi kí tự *s*[*i*] ta xét hai trường hợp:

- Trường hợp thứ nhất: **s**[*i*] là dấu mở ngoặc ' (': ta ghi nhận *i* là vị trí xuất hiện đầu cụm vào một ngăn xếp (stack) **st**:
 $inc(p); st[p] := i;$
trong đó **p** là con trỏ ngăn xếp. **p** luôn luôn trỏ đến ngọn, tức là phần tử cuối cùng của ngăn xếp. Thủ tục này gọi là nạp vào ngăn xếp: **NapST**.

- ♦ Trường hợp thứ hai: **s[i]** là dấu đóng ngoặc **)**: ta lấy phần tử ngọn ra khỏi ngăn xếp kết hợp với vị trí **i** để ghi nhận các vị trí đầu và cuối cụm trong **s**. Hàm này gọi là lấy phần tử ra khỏi ngăn xếp: **LaySt**. Khi lấy một phần tử ra khỏi ngăn xếp ta giảm con trỏ ngăn xếp 1 đơn vị.

j := st[p]; dec(p);

Có hai trường hợp gây ra lỗi cú pháp đơn giản như sau:

1. Gặp **)** mà trước đó chưa gặp **(**: Lỗi "*chưa mở đã đóng*". Lỗi này được phát hiện khi xảy ra tình huống **s[i] =)** và stack rỗng (**p = 0**).
2. Đã gặp **(** mà sau đó không gặp **)**: Lỗi "*mở rồi mà không đóng*". Lỗi này được phát hiện khi xảy ra tình huống đã duyệt hết biểu thức **s** nhưng trong stack vẫn còn phần tử (**p > 0**). Lưu ý rằng stack là nơi ghi nhận các dấu mở ngoặc **(**.

Ta dùng biến **SoCum** để đếm và ghi nhận các cụm xuất hiện trong quá trình duyệt biểu thức. Trường hợp gặp lỗi ta đặt **SoCum := -1**. Hai mảng **dau** và **cuoi** ghi nhận vị trí xuất hiện của kí tự đầu cụm và kí tự cuối cụm. Khi tổng hợp kết quả, ta sẽ dùng đến thông tin của hai mảng này.

```
(*-----
      Xử lý biểu thức trong s
-----*)
procedure BieuThuc;
var i: byte;
begin
  KhoiTriSt; {Khởi trị cho stack}
  SoCum := 0; {Khởi trị con đếm cụm}
  for i := 1 to length(s) do
    case s[i] of
      '(': NapSt(i);
      ')': if StackRong then
        begin SoCum := -1; exit; end
        else
        begin
          inc(SoCum);
          dau[SoCum] := LaySt;
          cuoi[SoCum] := i;
        end;
    end {case};
  if p > 0 then
    begin SoCum := -1; exit; end;
end;
```

Sau khi duyệt xong chuỗi **s** ta ghi kết quả vào tệp **output g**. Hàm **copy(s, i, d)** cho chuỗi gồm **d** kí tự được cắt từ chuỗi **s** kể từ kí tự thứ **i** trong **s**. Thí dụ **copy('12345678', 4, 3) = '456'**.

(* Pascal *)

```
program Cum;
{$B-}
uses crt;
const
  fn = 'CUM.INP'; gn = 'CUM.OUT';
  type mb1 = array[1..255] of byte;
```

```

var s: string; {chua bieu thuc can xu li}
    st: mbl; {stack}
    {stack luu vi tri xuat hien dau ( trong xau s}
    p: integer; {con tro stack}
    dau,cuoi: mbl; {vi tri dau, cuoi cua 1 cum}
    SoCum: integer;
    f,g: text;
(*-----
        Khoi tri stack st
-----*)
procedure KhoiTriSt;
begin p := 0; end;
(*-----
        Nap tri i vao stack st
-----*)
procedure NapSt(i: byte);
begin inc(p); st[p] := i; end;
(*-----
        Lay ra 1 tri tu ngon stack st
-----*)
function LaySt: byte;
begin LaySt := st[p]; dec(p); end;
(*-----
        Kiem tra Stack St rong ?
-----*)
function StackRong: Boolean;
begin StackRong := (p=0); end;
(*-----
        Xu ly bieu thuc trong s
-----*)
procedure BieuThuc; tự viết
(*-----
Doc du lieu tu tep input vao xau s
-----*)
procedure Doc;
begin
    s := ''; {gan tri rong cho s}
    assign(f,fn); reset(f);
    if not seekeof(f) then readln(f,s);
    close(f);
end;
(*-----
Ghi ket qua vao tep output
-----*)
procedure Ghi;
var i: byte;
begin
    assign(g,gn); rewrite(g); writeln(g,SoCum);
    for i := 1 to SoCum do
        writeln(g,copy(s,dau[i],cuoi[i]-dau[i]+1));
    close(g);
end;
BEGIN

```

```

Doc; BieuThuc; Ghi;
END.

```

```
// C#
```

```

using System;
using System.IO;
namespace SangTao1
{
    /*-----
    *           Tach cum trong bieu thuc
    * -----*/
    class Cum
    {
        const string fn = "cum.inp";
        const string gn = "cum.out";
        static void Main()
        {
            if (XuLi()) KiemTra();
            Console.ReadLine();
        }

        static public bool XuLi()
        {
            // Doc du lieu vao string s,
            // bo cac dau cach dau va cuoi s
            string s = (File.ReadAllText(fn)).Trim();
            int[] st = new int[s.Length]; //stack
            int p = 0; // con tro stack
            int sc = 0; // Dem so cum
            String ss = ""; // ket qua
            for (int i = 0; i < s.Length; ++i)
            {
                switch (s[i])
                {
                    case '(': st[++p] = i; break;
                    case ')':
                        if (p == 0)
                        {
                            Console.WriteLine("\nLOI:" +
                                " Thieu (");
                            return false;
                        }
                        ++sc;
                        for (int j = st[p]; j <= i; ++j)
                            ss += s[j];
                        ss += "\n"; // ket dong
                        --p;
                        break;
                } // switch
            } // for
            if (p > 0)
            {

```

```

        Console.WriteLine("\n LOI: Thua (");
        return false;
    }
    // Ghi file ket qua
    File.WriteAllText(gn, (sc.ToString() + "\n"
        + ss));
    return true;
}
// Doc lai 2 file inp va out de kiem tra
static public void KiemTra()
{
    Console.WriteLine("\n Input file " + fn);
    Console.WriteLine(File.ReadAllText(fn));
    Console.WriteLine("\n Output file " + gn);
    Console.WriteLine(File.ReadAllText(gn));
}
} // Cum
} // SangTao1

```

Bài 4.2. Bài gộp

Bộ bài bao gồm n quân, được gán mã số từ 1 đến n . Lúc đầu bộ bài được chia cho n người, mỗi người nhận 1 quân. Mỗi lượt chơi, trọng tài chọn ngẫu nhiên hai số x và y trong khoảng $1..n$. Nếu có hai người khác nhau, một người có trong tay quân bài x và người kia có quân bài y thì một trong hai người đó phải trao toàn bộ số bài của mình cho người kia theo nguyên tắc sau: mỗi người trong số hai người đó trình ra một quân bài tùy chọn của mình, Ai có quân bài mang mã số nhỏ hơn sẽ được nhận bài của người kia. Trò chơi kết thúc khi có một người cầm trong tay cả bộ bài. Biết số quân bài n và các quân bài trọng tài chọn ngẫu nhiên sau m lượt chơi, hãy cho biết số lượng người còn có bài trên tay.

Dữ liệu vào: Tập văn bản **BAIGOP.INP**.

- Dòng đầu tiên: hai số n và m , trong đó n là số lượng quân bài trong bộ bài, m là số lần trọng tài chọn ngẫu nhiên hai số x và y . Các quân bài được gán mã số từ 1 đến n . Mã số này được ghi trên quân bài.

- Tiếp đến là m dòng, mỗi dòng ghi hai số tự nhiên x và y do trọng tài cung cấp. Các số trên cùng một dòng cách nhau qua dấu cách.

Dữ liệu ra: Hiển thị trên màn hình số lượng người còn có bài trên tay.

Thí dụ:

Dữ liệu vào: BAIGOP.INP	Kết quả hiển thị trên màn hình
10 6 2 5 3 3 4 7 1 5 2 8 9 3	5

ý nghĩa: bộ bài có 10 quân mã số lần lượt 1, 2,..., 10 và có 10 người chơi. Sáu lần trọng tài chọn ngẫu nhiên các cặp số (x, y) là (2, 5), (3, 3), (4, 7), (1, 5), (2, 8) và (9, 3). Cuối ván chơi còn lại 5 người có bài trên tay: {1, 2, 5, 8}, {3, 9}, {4, 7}, {6}, {10}.

Thuật toán

Đây là bài toán có nhiều ứng dụng hữu hiệu nên bạn đọc cần tìm hiểu kĩ và cố gắng cài đặt cho nhuần nhuyễn. Như sau này sẽ thấy, nhiều thuật toán xử lí đồ thị như tìm

cây khung, xác định thành phần liên thông, xác định chu trình... sẽ phải vận dụng cách tổ chức dữ liệu tương tự như thuật toán sẽ trình bày dưới đây.

Bài này đòi hỏi tổ chức các tập quân bài sao cho thực hiện nhanh nhất các thao tác sau đây:

Find(x): cho biết tên của tập chứa phần tử x .

Union(x, y): hợp tập chứa x với tập chứa y .

Mỗi tập là nhóm các quân bài có trong tay một người chơi. Như vậy mỗi tập là một tập con của bộ bài $\{1, 2, \dots, n\}$. Ta gọi bộ bài là *tập chủ* hay *tập nền*. Do tính chất của trò chơi, ta có hai nhận xét quan trọng sau đây:

1. Hợp của tất cả các tập con (mỗi tập con này do một người đang chơi quản lí) đúng bằng tập chủ.
2. Hai tập con khác nhau không giao nhau: tại mỗi thời điểm của cuộc chơi, mỗi quân bài nằm trong tay đúng một người.

Họ các tập con thỏa hai tính chất nói trên được gọi là một *phân hoạch* của tập chủ.

Các thao tác nói trên phục vụ trực tiếp cho việc tổ chức trò chơi theo sơ đồ sau:

Khởi trị:

for $i := 1$ **to** n **do**

begin

Trọng tài sinh ngẫu nhiên hai số x và y

trong khoảng $1..n$:

Hợp tập chứa x với tập chứa y : $\text{Union}(x, y)$;

end;

Để thực hiện thủ tục **Union(x, y)** trước hết ta cần biết quân bài x và quân bài y đang ở trong tay ai? Sau đó ta cần biết người giữ quân bài x (hoặc y) có quân bài nhỏ nhất là gì? Quân bài nhỏ nhất được xác định trong toàn bộ các quân bài mà người đó có trong tay. Đây chính là điểm dễ nhầm lẫn. Thí dụ, người chơi A đang giữ trong tay các quân bài 3, 4 và 7, $A = \{3, 4, 7\}$; người chơi B đang giữ các quân bài 2, 5, 9 và 11, $B = \{2, 5, 9, 11\}$. Các số gạch chân là số hiệu của quân bài nhỏ nhất trong tay mỗi người. Nếu $x = 9$ và $y = 7$ thì A (đang giữ quân $y = 7$) và B (đang giữ quân $x = 9$) sẽ phải đấu với nhau. Vì trong tay A có quân nhỏ nhất là 3 và trong tay B có quân nhỏ nhất là 2 nên A sẽ phải nộp bài cho B và ra khỏi cuộc chơi. Ta có, $B = \{2, 3, 4, 5, 7, 9, 11\}$. Ta kết hợp việc xác định quân bài x trong tay ai và người đó có quân bài nhỏ nhất là bao nhiêu làm một để xây dựng hàm **Find(x)**. Cụ thể là hàm **Find(x)** sẽ cho ta *quân bài nhỏ nhất có trong tay người giữ quân bài x* . Trong thí dụ trên ta có:

Find(x) = Find(9) = 2 và Find(y) = Find(7) = 3

Lưu ý rằng hàm **Find(x)** không chỉ rõ ai là người đang giữ quân bài x mà cho biết quân bài có số hiệu *nhỏ nhất* có trong tay người đang giữ quân bài x , nghĩa là **Find(9)=2** chứ không phải **Find(9) = B**. Để giải quyết sự khác biệt này ta hãy chọn phần tử có số hiệu nhỏ nhất trong tập các quân bài có trong tay một người làm *phần tử đại diện của tập đó*. Ta cũng đồng nhất phần tử đại diện với *mã số của người giữ tập quân bài*. Theo quy định này thì biểu thức **Find(9)=2** có thể được hiểu theo một trong hai nghĩa tương đương như sau:

- ♦ Người số 2 đang giữ quân bài 9.
- ♦ Tập số 2 chứa phần tử 9.

Tổ chức hàm **Find** như trên có lợi là sau khi gọi **i:=Find(x)** và **j:=Find(y)** ta xác định ngay được ai phải nộp bài cho ai. Nếu $i < j$ thì j phải nộp bài cho i , ngược

lại, nếu $i > j$ thì i phải nộp bài cho j . Trường hợp $i = j$ cho biết hai quân bài x và y đang có trong tay một người, ta không phải làm gì.

Tóm lại ta đặt ra các nguyên tắc sau:

a) Lấy phần tử nhỏ nhất trong mỗi tập làm tên riêng cho tập đó.

b) Phần tử có giá trị nhỏ quản lý các phần tử có giá trị lớn hơn nó theo phương thức: mỗi phần tử trong một tập đều trở trực tiếp đến một phần tử nhỏ hơn nó và có trong tập đó. Phần tử nhỏ nhất trong tập trở tới chính nó.

Trong thí dụ trên ta có $A = \{3, 4, 7\}$, $B = \{2, 5, 9, 11\}$, $x = 9$ và $y = 7$.

Như vậy, tập A có phần tử đại diện là 3 và tập B có phần tử đại diện là 2.

Dữ liệu của tập A khi đó sẽ được tổ chức như sau:

$A = \{3 \rightarrow 3, 4 \rightarrow 3, 7 \rightarrow 3\}$. Như vậy 3 là phần tử đại diện của tập này, do đó ta không cần dùng biến A để biểu thị nó nữa mà có thể viết:

$\{3 \rightarrow 3, 4 \rightarrow 3, 7 \rightarrow 3\}$ hoặc gọn hơn $\{3, 4, 7\} \rightarrow 3$.

Tương tự, dữ liệu của tập B sẽ có dạng:

$\{2 \rightarrow 2, 5 \rightarrow 2, 9 \rightarrow 2, 11 \rightarrow 2\}$ hoặc gọn hơn $\{2, 5, 9, 11\} \rightarrow 2$.

Khi đó **Find(9) = 2** và **Find(7) = 3**, và do đó, tập 3 phải được gộp vào tập 2. Phép **Union(9, 7)** sẽ tạo ra tập sau đây:

$\{3 \rightarrow 2, 4 \rightarrow 3, 7 \rightarrow 3, 2 \rightarrow 2, 5 \rightarrow 2, 9 \rightarrow 2, 11 \rightarrow 2\}$,

tức là ta thực hiện đúng một thao tác sửa $3 \rightarrow 3$ thành $3 \rightarrow 2$: để hợp hai tập ta chỉ việc đối sánh hai phần tử đại diện i và j của chúng:

- ♦ Nếu $i > j$ thì cho tập i phụ thuộc vào tập j .
- ♦ Nếu $j > i$ thì cho tập j phụ thuộc vào tập i .
- ♦ Nếu $i = j$ thì không làm gì.

Kĩ thuật nói trên được gọi là *hợp các tập con rời nhau*.

Ta dùng một mảng nguyên a thể hiện tất cả các tập. Khi đó hai tập A và B nói trên được thể hiện trong a như sau:

$a[3] = 3; a[4] = 3; a[7] = 3;$

{tập A: phần tử đại diện là 3, các phần tử 3, 4 và 7 đều trỏ đến 3}

$a[2] = 2; a[5] = 2; a[9] = 2; a[11] = 2;$

{tập B: phần tử đại diện là 2, các phần tử 2, 5, 9 và 11 đều trỏ đến 2}

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
	2	3	3	2		3		2		2				
	(B)	(A)	(A)	(B)		(A)		(B)		(B)				

Sau khi hợp nhất A với B ta được:

$a[3] = 2; \{\text{chỗ sửa duy nhất}\}$

$a[4] = 3; a[7] = 3; a[2] = 2; a[5] = 2; a[9] = 2; a[11] = 2;$

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
	2	2	3	2		3		2		2				

Theo các nguyên tắc trên ta suy ra phần tử x là phần tử đại diện của tập khi và chỉ khi $a[x] = x$. Dựa vào đây ta tổ chức hàm **Find(x)**: xác định phần tử đại diện của tập chứa x .

```
function Find(x: integer): integer;
begin
```

```

    while (x <> a[x]) do x := a[x];
    Find := x;
end;

```

Khi đó thủ tục **Union** được triển khai đơn giản như sau:

```

procedure Union(x, y: integer);
begin
    x := Find(x);
    y := Find(y);
    if x = y then exit else
        if x < y then a[y] := x
        else {x > y} a[x] := y;
end;

```

Lúc bắt đầu chơi, mỗi người giữ một quân bài, ta khởi trị $a[i] := i$ cho mọi $i = 1..n$ với ý nghĩa: tập có đúng một phần tử thì nó là đại diện của chính nó.

Để đếm số tập còn lại sau mỗi bước chơi ta có thể thực hiện theo hai cách.

Ta thấy, nếu $\text{Find}(x) = \text{Find}(y)$ thì không xảy ra việc gộp bài vì x và y cùng nằm trong một tập. Ngược lại, nếu $\text{Find}(x) \neq \text{Find}(y)$ thì do gộp bài nên số người chơi sẽ giảm đi 1 tức là số lượng tập giảm theo. Ta dùng một biến c đếm số lượng tập. Lúc đầu khởi trị $c := n$ (có n người chơi). Mỗi khi xảy ra điều kiện $\text{Find}(x) \neq \text{Find}(y)$ ta giảm c 1 đơn vị: $\text{dec}(c)$.

Theo cách thứ hai ta viết hàm **Dem** để đếm số lượng tập sau khi kết thúc một lượt chơi. Ta có đặc tả sau đây:

số lượng tập = số lượng đại diện của tập.

Phần tử i là đại diện của một tập khi và chỉ khi $a[i] = i$.

Đặc tả trên cho ta:

```

function Dem: integer;
var d, i: integer;
begin
    d := 0;
    for i := 1 to n do
        if a[i] = i then inc(d);
    Dem := d;
end;

```

Dĩ nhiên trong bài này phương pháp thứ nhất sẽ hiệu quả hơn, tuy nhiên ta thực hiện cả hai phương pháp vì hàm **Dem** là một tiện ích trong loại hình tổ chức dữ liệu theo tiếp cận **Find-Union**.

Ta cũng sẽ cài đặt **Union(x, y)** dưới dạng hàm với giá trị ra là 1 nếu trước thời điểm hợp nhất x và y thuộc về hai tập phân biệt và là 0 nếu trước đó x và y đã thực sự có trong cùng một tập. Nói cách khác **Union(x, y)** cho biết phép hợp nhất có thực sự xảy ra (1) hay không (0).

Trong chương trình dưới đây tệp **BAIGOP.INP** chứa dữ liệu vào có cấu trúc như sau:

- Dòng đầu tiên chứa hai số nguyên dương n và m , trong đó n là số lượng quân bài, m là số lần trọng tài phát sinh ra hai số ngẫu nhiên.

- Tiếp đến là m dòng, mỗi dòng chứa hai số do trọng tài phát sinh. Các số được viết cách nhau bởi dấu cách.

Kĩ thuật trên có tên gọi là *Find-Union* đóng vai trò quan trọng trong các thủ tục xử lí hợp các tập rời nhau. Trước khi xem chương trình chúng ta hãy thử làm một bài tập nhỏ sau đây:

Với mảng a được tổ chức theo kĩ thuật *Find-Union* dưới đây hãy cho biết có mấy tập con và hãy liệt kê từng tập một.

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
1	2	2	3	2	6	3	1	2	6	2	10	3	8	12

Đáp số: ba tập.

Tập với đại diện 1: {1, 8, 14}.

Tập với đại diện 2: {2, 3, 4, 5, 7, 9, 11, 13}.

Tập với đại diện 6: {6, 10, 12, 15}.

(* Pascal *)

```
(*-----
                                BAI GOP
-----*)
program BaiGop;
uses crt;
const
  MN = 5000;
  fn = 'BAIGOP.INP';
var
  a: array[1..MN] of integer;
  c,n,m: integer;
  {c: dem so tap
   n: so quan bai = so nguoi choi
   m: so lan trong tai sinh 2 so}
  f: text;
{-----
  Xac dinh tap chua phan tu x
-----}
function Find(x: integer): integer;
begin
  while (x <> a[x]) do x:= a[x];
  Find := x;
end;
{-----
  Hop cua tap chua x voi tap chua y.
  Union = 1: co hop nhat
  Union = 0: khong hop nhat vi x va y
               thuoc cung 1 tap
-----}
function Union(x,y: integer): integer;
begin
  Union := 0;
  x := Find(x);
  y := Find(y);
```

```

        if x = y then exit
        else if x < y then a[y] := x
        else a[x] := y;
    Union := 1;
end;
{-----
  Dem so luong tap con roi nhau (so nguoi choi)
-----}
function Dem: integer;
var d,i: integer;
begin
    d := 0;
    for i := 1 to n do
        if a[i] = i then inc(d);
    Dem:= d;
end;
procedure BaiGop;
var i,j,k,x,y: integer;
begin
    assign(f,fn); reset(f);
    readln(f,n,m);
    {n - so quan bai; m - so lan chon x,y}
    c := n; {c: so nguoi choi}
    if (n < 1) or (n > MN) then exit;
    for i := 1 to n do a[i] := i;
    for i := 1 to m do
        begin
            readln(f,x,y);
            c := c-Union(x,y);
        end;
    writeln(c);
    close(f);
end;
BEGIN
    BaiGop;
END.

```

// C#

```

using System;
using System.IO;
namespace SangTao1
/*-----
*                               Bai gop
* -----*/
class BaiGop
{
    const string fn = "baigop.inp";
    static int[] a; // quan li cac tap roi nhau
    static int n = 0; // so quan bai
    static int m = 0; // so lan trong tai
                    // chon 2 quan bai
    static void Main()
    {

```

```

        Run();
        Console.ReadLine();
    } // Main
    // Xác định tập chưa phân tử x
    static int Find(int x) tự viết
    static int Union(int x, int y) tự viết
    static void Run()
    {
        int [] b = ReadInput();
        n = b[0];
        m = b[1];
        a = new int[n + 1];
        for (int i = 1; i <= n; ++i)
            a[i] = i;
        int j = 2, d = n;
        for (int i = 1; i <= m; ++i)
        {
            d -= Union(b[j], b[j+1]);
            j += 2;
        }
        Console.WriteLine("\n " + d + "\n");
    }
    static int[] ReadInput()
    {
        char[] cc = new char[]
        { ' ', '\n', '\t', '\r' };
        string[] ss = (File.ReadAllText(fn)).
            Split(cc, StringSplitOptions.
                RemoveEmptyEntries);
        return Array.ConvertAll(ss,
            new Converter<string, int>(int.Parse));
    }
} // BaiGop
} // SangTao1

```

Bài 4.3. Chuỗi hạt

Trong một tệp văn bản tên **CHUOI.DAT** có biểu diễn một chuỗi hạt, mỗi hạt có thể nhận một trong số các màu mã số từ 1 đến 30.

Lập trình thực hiện các việc sau:

- Đọc chuỗi hạt từ tệp vào mảng nguyên dương a.
- Hiển thị số màu có trong chuỗi.
- Tìm một điểm để cắt chuỗi rồi căng thẳng ra sao cho tổng số các hạt cùng màu ở hai đầu là lớn nhất.

Chuỗi được thể hiện trong tệp dưới dạng hình thoi, dòng đầu tiên và dòng cuối cùng mỗi dòng có một hạt.

Mỗi dòng còn lại có hai hạt (xem hình).

Các hạt của chuỗi được đánh số bắt đầu từ hạt trên cùng và theo chiều kim đồng hồ.

CHUOI.DAT

Trong thí dụ này, các thông báo trên màn hình sẽ là:

4		Số màu trong chuỗi: 5
4	7	Cắt giữa hạt thứ 7 và thứ 8, tổng số lớn nhất là 7.
1	4	
5	8	
5	8	
5	8	
8		
Chuỗi hạt		

Thuật toán

Khung chương trình được phác thảo như sau:

```

procedure run;
var i: integer;
begin
    Đọc dữ liệu;
    Tính và thông báo số màu
    Xử lý để tìm điểm cắt;
    Thông báo điểm cắt
end;
  
```

Để đọc chuỗi từ tệp vào mảng a ta dùng thêm một mảng phụ b có cùng cấu trúc như mảng a . Mảng b sẽ chứa các hạt ở nửa trái chuỗi, trong khi mảng a chứa các hạt ở nửa phải. Lúc đầu, do chỉ có 1 hạt tại dòng đầu tiên nên ta đọc hạt đó vào $a[1]$. Tại các dòng tiếp theo, mỗi dòng $n = 2, \dots$ ta đọc số hiệu màu của 2 hạt, hạt trái vào $b[n]$ và hạt phải vào $a[n]$. Dấu hiệu kết thúc chuỗi là 1 hạt. Hạt này được đọc vào $b[n]$. Ta để ý rằng, theo cấu trúc của chuỗi hạt thì số hạt trong chuỗi luôn luôn là một số chẵn.

Thí dụ dưới đây minh hoạ giai đoạn đầu của thủ tục đọc dữ liệu. Khi kết thúc giai đoạn này ta thu được $n = 7$ và nửa phải của chuỗi hạt (số có gạch dưới) được ghi trong $a[1..(n-1)]$, nửa trái được ghi trong $b[2..n]$. Tổng số hạt trong chuỗi khi đó sẽ là $2 \cdot (n-1)$.

CHUOI . DAT			
4		<u>4</u>	a[1]
4	7	b[2]	<u>7</u> a[2]
1	4	b[3]	<u>4</u> a[3]
5	8	b[4]	<u>8</u> a[4]
5	8	b[5]	<u>8</u> a[5]
5	8	b[6]	<u>8</u> a[6]
8		b[7]	8

*Đọc dữ liệu của chuỗi hạt vào
hai mảng a và b*

$a[1..6] = (4, 7, 4, 8, 8, 8)$

$b[2..7] = (4, 1, 5, 5, 5, 8)$

Sau khi đọc xong ta duyệt ngược mảng b để nối nửa trái của chuỗi hạt vào sau nửa phải a .

```

(*-----
      Doc du lieu tu file CHUOI.DAT
      ghi vao mang a
-----*)
procedure Doc;
var
  f: text;
  i: integer;
begin
  assign(f,fn); reset(f);
  n := 1; read(f,a[n]);
  while NOT SeekEof(f) do
    begin
      inc(n); read(f,b[n]);
      if NOT SeekEof(f) then read(f,a[n]);
    end;
    {noi nua trai b (duyet nguoc) vao nua phai a}
    for i:= 0 to n-2 do a[n+i]:= b[n-i];
    n := 2*(n-1); close(f);
  end;

```

Theo thí dụ trên, sau khi nối **b[2..n]** vào sau **a[1..(n - 1)]** ta thu được

a[1..12] = (4,7,4,8,8,8,8,5,5,5,1,4)

Để đếm số màu trong chuỗi ta dùng phương pháp đánh dấu. Ta sử dụng mảng **b** với ý nghĩa như sau:

- **b[i] = 0**: màu *i* chưa xuất hiện trong chuỗi hạt;
- **b[i] = 1**: màu *i* đã xuất hiện trong chuỗi hạt.

Lần lượt duyệt các phần tử **a[i]**, **i = 1..n** trong chuỗi. Nếu màu **a[i]** chưa xuất hiện ta tăng trị của con đếm màu **d** thêm 1, **inc(d)** và đánh dấu màu **a[i]** đó trong **b** bằng phép gán **b[a[i]] := 1**.

```

(*-----
      Dem so mau trong chuoai
-----*)
function Dem: integer;
var i,d: integer;
begin
  d := 0;
  fillchar(b,sizeof(b),0);
  for i := 1 to n do
    if b[a[i]] = 0 then
      begin
        inc(d);
        b[a[i]] := 1;
      end;
  Dem := d;
end;

```

Để tìm điểm cắt với tổng chiều dài hai đầu lớn nhất ta thực hiện như sau. Trước hết ta định nghĩa điểm đổi màu trên chuỗi hạt là hạt (chỉ số) mà màu của nó khác với màu của hạt đứng sát nó (sát phải hay sát trái, tùy theo chiều duyệt xuôi từ trái qua phải hay duyệt ngược từ phải qua trái). Ta cũng định nghĩa một đoạn trong chuỗi hạt là một dãy liên tiếp các hạt cùng màu với chiều dài tối đa. Mỗi đoạn đều có điểm đầu và điểm cuối.

Vì điểm cuối của mỗi đoạn chỉ lệch 1 đơn vị so với điểm đầu của đoạn tiếp theo, cho nên với mỗi đoạn ta chỉ cần quản lý một trong hai điểm: điểm đầu hoặc điểm cuối của đoạn đó. Ta chọn điểm đầu. Kỹ thuật này được gọi là quản lý theo đoạn.

Thí dụ, chuỗi hạt a với $n = 12$ hạt màu như trong thí dụ đã cho:

$$a[1..12] = (4, 7, 4, 8, 8, 8, 8, 5, 5, 5, 1, 4)$$

mới xem tương như được tạo từ bảy đoạn là:

$$\begin{aligned} a[1..1] &= (4) \\ a[2..2] &= (7) \\ a[3..3] &= (4) \\ a[4..7] &= (8, 8, 8, 8) \\ a[8..10] &= (5, 5, 5) \\ a[11..11] &= (1) \\ a[12..12] &= (4) \end{aligned}$$

Tuy nhiên, do chuỗi là một dãy hạt khép kín và các hạt được bố trí theo chiều quay của kim đồng hồ nên thực chất a chỉ gồm sáu đoạn:

$$\begin{aligned} a[2..2] &= (\underline{7}) \\ a[3..3] &= (\underline{4}) \\ a[4..7] &= (\underline{8}, 8, 8, 8) \\ a[8..10] &= (\underline{5}, 5, 5) \\ a[11..11] &= (\underline{1}) \\ a[12..1] &= (\underline{4}, 4) \end{aligned}$$

trong đó $a[x..y]$ cho biết chỉ số đầu đoạn là x , cuối đoạn là y . Nếu $x \leq y$ thì các hạt trong đoạn được duyệt theo chiều kim đồng hồ từ chỉ số nhỏ đến chỉ số lớn, ngược lại, nếu $x > y$ thì các hạt trong đoạn cũng được duyệt theo chiều kim đồng hồ từ chỉ số lớn đến chỉ số nhỏ. Các phần tử đầu mỗi đoạn được gạch chân. Có thể có những đoạn chứa cả hạt cuối cùng $a[n]$ và hạt đầu tiên $a[1]$ nên ta cần xét riêng trường hợp này.

Đoạn trình dưới đây xác định các điểm đầu của mỗi đoạn và ghi vào mảng $b[1..sdc]$. Giá trị sdc cho biết số lượng các đoạn.

```

sdc := 0;
if a[1]<>a[n] then
  begin
    sdc := 1;
    b[sdc] := 1;
  end;
for i := 1 to n-1 do
  if a[i] <> a[i+1] then
    begin
      inc(sdc);
      b[sdc] := i+1;
    end;
end;

```

Gọi điểm đầu của ba đoạn liên tiếp là $d1$, $d2$ và $d3$. Ta thấy, nếu chọn điểm cắt sát trái hạt $d2$ thì hiệu $d3 - d1$ chính là tổng số hạt đồng màu tại hai đầu của chuỗi hạt được căng ra. Từ đó ta phác thảo được sơ đồ cho thủ tục xử lý để tìm điểm cắt **DiemCat** với chiều dài lớn nhất **Lmax** như sau:

```

Khởi trị;
Duyệt từ bộ ba điểm đầu của
ba đoạn liên tiếp d1, d2, d3

```

Nếu $d3-d1 > L_{max}$ thì

Đặt lại $L_{max} := d3-d1$

Đặt lại $DiemCat := d2$

xong nếu

Giả sử chuỗi hạt có m đoạn. Theo phương thức duyệt chuỗi hạt vòng tròn theo chiều kim đồng hồ, ta cần xét riêng hai đoạn đầu và cuối, cụ thể là:

- ♦ Với đoạn 1 ta phải xét hai đoạn đứng trước và sau đoạn đó là đoạn m và đoạn 2.
- ♦ Với đoạn m ta phải xét hai đoạn đứng trước và sau đoạn đó là đoạn $m-1$ và đoạn 1.

Ta xử lý riêng hai đoạn này ở bước khởi trị như sau:

{xử lý điểm cat đầu}

$L_{max} := (b[1] + (n - b[sdc])) + (b[2] - b[1]);$

$DiemCat := b[1];$

{xử lý điểm cat cuối}

if $(b[1] + (n - b[sdc])) + (b[sdc] - b[sdc-1]) > L_{max}$ then

begin

$L_{max} := (b[1] + (n - b[sdc])) + (b[sdc] - b[sdc-1]);$

$DiemCat := b[sdc];$

end;

Phương án cuối cùng của thủ tục xuly sẽ như sau:

procedure xuly;

var i, sdc: integer; {sdc: số điểm cat}

begin

sdc:=0;

if $a[1] \neq a[n]$ then

begin

sdc := 1;

$b[sdc] := 1;$

end;

for i:=1 to n-1 do

if $a[i] \neq a[i+1]$ then

begin

inc(sdc);

$b[sdc] := i+1;$

end;

if sdc=0 then

begin

$DiemCat:=0;$

$L_{max}:=n;$

exit;

end;

{xử lý điểm cat đầu}

$L_{max} := (b[1] + (n - b[sdc])) + (b[2] - b[1]);$

$DiemCat := b[1];$

{xử lý điểm cat cuối}

if $(b[1] + (n - b[sdc])) + (b[sdc] - b[sdc-1]) > L_{max}$ then

begin

$L_{max} := (b[1] + (n - b[sdc])) + (b[sdc] - b[sdc-1]);$

$DiemCat := b[sdc];$

end;

```

    {xu li cac diem cat giua}
    for i:= 2 to sdc-1 do
    if b[i+1]-b[i-1] > Lmax then
    begin
        Lmax := b[i+1]-b[i-1];
        DiemCat := b[i];
    end;
end;

```

(* Pascal *)

```

(*-----
      CHUOI HAT
-----*)
program Chuoi;
{$B-}
uses crt;
const
    MN = 500; {So luong hat toi da trong chuoi}
    MC = 30; {So luong mau}
    fn = 'CHUOI.DAT';
    BL = #32;
var
    a,b,len: array[0..MN] of byte;
    n: integer; {So luong phan tu thuc co trong chuoi hat}
    DiemCat: integer; {diem cat}
    Lmax: integer; {Chieu dai toi da}
(*-----
    Doc du lieu tu tep CHUOI.DAT ghi
                                vao mang a
-----*)
procedure Doc;
var
    f: text;
    i: integer;
begin
    assign(f,fn);
    reset(f);
    n:= 1;
    read(f,a[1]);
    while not SeekEof(f) do
    begin
        inc(n);
        read(f,b[n]);
        if not SeekEof(f) then read(f,a[n]);
    end;
    for i:=0 to n-2 do a[n+i]:= b[n-i];
    n:= 2*(n-1);
    close(f);
end;
(*-----
      Hien thi chuoi tren man hinh
      de kiem tra ket qua doc
-----*)

```

```

-----*)
procedure Xem;
var i: integer;
begin
  writeln;
  writeln('Tong so hat: ',n);
  for i:= 1 to n do
    write(a[i],b1);
  end;
  (*-----
    Dem so mau trong chuoai
  -----*)
function Dem: integer;
var i,d: integer;
begin
  d:=0;
  fillchar(b,sizeof(b),0);
  for i:= 1 to n do
    if b[a[i]] = 0 then
      begin
        inc(d);
        b[a[i]]:=1;
      end;
  Dem:= d;
end;
procedure xuly;
var i,sdc: integer; {sdc: so diem cat}
begin
  sdc:=0;
  if a[1]<>a[n] then
    begin
      sdc:=1;
      b[sdc]:=1;
    end;
  for i:=1 to n-1 do
    if a[i] <> a[i+1] then
      begin
        inc(sdc);
        b[sdc]:=i+1;
      end;
  if sdc=0 then
    begin
      DiemCat:=0;
      Lmax:=n;
      exit;
    end;
  {xu li diem cat dau}
  Lmax:= (b[1]+(n-b[sdc]))+(b[2]-b[1]);
  DiemCat:=b[1];
  {xu li diem cat cuoi}
  if (b[1]+(n-b[sdc]))+(b[sdc]-b[sdc-1]) > Lmax
  then
    begin

```

```

        Lmax:= (b[1]+(n-b[sdc]))+(b[sdc]-b[sdc-1]);
        DiemCat:=b[sdc];
    end;
    {xu li cac diem cat giua}
    for i:=2 to sdc-1 do
        if b[i+1]-b[i-1] > Lmax then
            begin
                Lmax:= b[i+1]-b[i-1];
                DiemCat:=b[i];
            end;
    end;
end;
procedure run;
var i: integer;
begin
    Doc; Xem; writeln;
    writeln('So mau trong chuoai: ',Dem);
    xuly;
    writeln;
    if DiemCat=0 then
        writeln(' Chuoi dong mau, cat tai diem tuy y')
    else
        begin
            if DiemCat = 1 then i :=n
            else i:=DiemCat-1;
            writeln('Cat giua hat ',i,
                ' va hat ',DiemCat);
        end;
    writeln(' Chieu dai max: ',Lmax);
    readln;
end;
BEGIN
    run;
END.

```

Dữ liệu kiểm thử CHUOI.DAT	Kết quả dự kiến
4 4 7 1 4 5 8 5 8 5 8 8	Cắt giữa hạt: 7 và 8 Chiều dài max: 7

// C#

```

using System;
using System.IO;
namespace SangTao1
{
    class ChuoiHat
    {

```

```

const string fn = "chuoit.dat";
static int[] a; // chuoit hat
static int n; // so phan tu trong input file
static void Main()
{
    Run();
    Console.ReadLine();
} // Main
static void Run()
{
    int[] b = ReadInput();
    n = b.Length;
    a = new int[n];
    int t = 0; // nua trai
    int p = n - 1; // nua phai
    int n2 = n / 2;
    for (int i = 0; i < n2; ++i)
    {
        a[t++] = b[2 * i];
        a[p--] = b[2 * i + 1];
    }
    Console.WriteLine();
    for (int i = 0; i < n; ++i)
        Console.Write(a[i] + " ");
    Console.WriteLine("\n\n Chuoi hat co "
        + Dem() + " mau \n");
    DiemCat();
}
static int[] ReadInput()
{
    char[] cc = new char[] { ' ', '\n', '\t', '\r' };
    return Array.ConvertAll(
        (File.ReadAllText(fn)).Split(cc,
            StringSplitOptions.RemoveEmptyEntries,
            new Converter<string, int>(int.Parse));
    );
}
/*-----
 * Dem so mau trong chuoit
 * -----*/
static int Dem()
{
    int[] b = new int[31];
    for (int i = 1; i <= 30; ++i) b[i] = 0;
    for (int i = 0; i < n; ++i) b[a[i]] = 1;
    int d = 0;
    for (int i = 1; i <= 30; ++i) d += b[i];
    return d;
}
static void DiemCat()
{
    int sdc = 0; // dem so diem cat
    int[] b = new int[n];

```

```

        for (int i = 1; i < n ; ++i)
            if (a[i] != a[i-1]) b[sdc++] = i-1;
    // xet diem dau a[0] va diem cuoi a[n-1]
        if (a[n-1] != a[0]) b[sdc++] = n-1;
        int DiemCat = 0;
        int Lmax = 0;
        if (sdc == 0) // chuoi hat dong mau
        {
            Lmax = n;
            Console.WriteLine("Chuoi hat dong mau. ");
            Console.WriteLine("Chon diem cat tuy y. ");
            Console.WriteLine("Chieu dai max = " + Lmax);
            return;
        }
        if (sdc == 2) // 2 mau
        {
            Lmax = n;
            Console.WriteLine("\n Cat giữa hat " + b[0]);
            Console.WriteLine("va hat " + (b[0] + 1) % n);
            Console.WriteLine("Chieu dai max = " + Lmax);
            return;
        }
        for (int i = 0; i < sdc; ++i)
            if ((b[(i + 2) % sdc] + n - b[i]) % n > Lmax)
            {
                Lmax = (b[(i + 2) % sdc] + n - b[i]) % n;
                DiemCat = b[(i + 1) % sdc];
            }
        Console.WriteLine("\n Cat giữa hat thu " +
                           (DiemCat + 1));
        Console.WriteLine(" va hat thu " +
                           ((DiemCat + 1) % n + 1));
        Console.WriteLine(" Chieu dai max = " + Lmax);
    }
} // class
} // SangTaol

```

Bài 4.4. Sắp mảng rồi ghi tệp

Sinh ngẫu nhiên n phần tử cho mảng nguyên a . Sắp a theo trật tự tăng dần rồi ghi vào một tệp văn bản có tên tùy đặt.

Gợi ý

Chương trình giới thiệu hai thủ tục sắp mảng là MinSort và QuickSort. Theo phương pháp MinSort với mỗi i ta tìm phần tử nhỏ nhất $a[j]$ trong đoạn $a[i..n]$ sau đó ta đổi chỗ phần tử này với phần tử $a[i]$. Như vậy mảng được chia thành hai đoạn: đoạn trái, $a[1..i]$ được sắp tăng, còn đoạn phải $a[i + 1..n]$ chưa xử lý. Mỗi bước ta thu hẹp đoạn phải cho đến khi còn một phần tử là xong.

Theo phương pháp QuickSort ta lấy một phần tử x nằm giữa đoạn mảng cần sắp làm mẫu rồi chia mảng thành hai đoạn. Đoạn trái $a[1..i]$ bao gồm các giá trị không lớn hơn x và đoạn phải $a[j..n]$ bao gồm các giá trị không nhỏ hơn x . Tiếp đến ta lặp lại thủ tục này với hai đoạn thu được nếu chúng chứa nhiều hơn một phần tử.

(* Pascal *)

```

(*-----
      SAPTANG: Sap mang, ghi tep
-----*)
program SapTang;
{$B-}
uses crt;
const
  MN = 5000;
  fn = 'saptang.out';
var
  f: text;
  a: array[1..MN] of integer;
  n: integer;
(*-----
sinh ngau nhien m phan tu
cho mang nguyen a
-----*)
procedure Init(m: integer);
var i: integer;
begin
  if (m < 0) or (m > MN) then exit;
  n:= m;
  randomize;
  for i:= 1 to n do a[i]:= random(MN);
end;
(*-----
      Sap nhanh doan a[d..c]
-----*)
procedure QuickSort(d, c: integer);
var i, j, x, y: integer;
begin
  i:= d; j:= c;
  x:= a[(i+j) div 2]; {lay tri mau x}
  while i <= j do
    begin
      while a[i] < x do inc(i); {to chuc doan trai}
      while a[j] > x do dec(j); {to chuc doan phai}
      if (i <= j) then {doi cho neu can}
        begin
          y:= a[i];
          a[i]:= a[j];
          a[j]:= y;
          inc(i); dec(j);
        end;
    end;
  if (d < j) then QuickSort(d,j); {sap tiep doan trai}
  if (i < c) then QuickSort(i,c); {sap tiep doan phai}
end;
(*-----
Tim chi dan m trong khoang d..c
thoa a[m] = min(a[d..c])
-----*)
function Min(d, c: integer): integer;

```

```

var i: integer;
begin
  for i:= d+1 to c do
    if a[i] < a[d] then d:= i;
  Min:= d;
end;
procedure MinSort;
var i, j, y: integer;
begin
  for i:= 1 to n-1 do
    begin
      j:= Min(i,n);
      {doi cho a[i] va a[j]}
      y:= a[i];
      a[i]:= a[j];
      a[j]:= y;
    end;
end;
(*-----
      Ghi tep, moi dong khong qua 20 so
-----*)
procedure Ghi;
var i: integer;
begin
  assign(f,fn); rewrite(f);
  writeln(f,n);
  for i:= 1 to n do
    begin
      write(f,a[i]:5);
      if i mod 20 = 0 then writeln(f);
    end;
  close(f);
end;
procedure TestQuickSort;
begin
  Init(MN);
  QuickSort(1,n);
  Ghi;
  write('Fini Quick Sort'); readln;
end;
procedure TestMinSort;
begin
  Init(MN);
  MinSort;
  Ghi;
  write('Fini Min Sort'); readln;
end;
BEGIN
  TestQuickSort;
  {TestMinSort;}
END.

```

// C#

```

using System;
using System.IO;
namespace SangTao1
{
    /*-----
    *          Sinh du lieu
    *          Sap tang
    *          Ghi file
    * -----*/
    class SapTang
    {
        const int mn = 50000;
        const string fn = "SapTang.dat";
        static int[] a = new int[mn];
    static int n = 0; // so phan tu trong input file

        static void Main()
        {
            Run(150);
            Console.ReadLine();
        } // Main

        static void Run(int nn) // sinh nn phan tu
        {
            n = nn;
            Console.WriteLine("\n Sinh ngau nhien " + n);
            Console.WriteLine("\n phan tu cho mang
                               a[0.." + (n - 1) +
                               "]"");
            Gen();
            Console.WriteLine("\n Quik Sort...");
            QSort(0, n - 1);
            Console.WriteLine("\n Ghi file " + fn + "...");
            Ghi();
            Console.WriteLine("\n Kiem tra lai file " +
                               fn + "\n\n");
            ShowFile();
        }
        /*-----
        * Sinh ngau nhien n so nguyen
        * cho mang a[0..n-1]
        * -----*/
        static void Gen()
        {
            Random r = new Random();
            for (int i = 0; i < n; ++i) a[i] = r.Next(100);
        }
        /*-----
        * Giai thuat sap (tang) nhanh
        * Quick Sort (Hoare T.)
        * -----*/
        static void QSort(int d, int c)
        {

```

```

        int i = d;
        int j = c;
        int m = a[(i + j) / 2];
        int t;
        while (i <= j)
        {
            while (a[i] < m) ++i;
            while (m < a[j]) --j;
            if (i <= j)
            {
                t = a[i]; a[i] = a[j]; a[j] = t;
                ++i; --j;
            }
        }
        if (d < j) QSort(d, j);
        if (i < c) QSort(i, c);
    }
    static void MinSort()
    {
        int i = 0;
        int j = 0;
        int t = 0;
        for (i = 0; i < n; ++i)
            for (j = i + 1; j < n; ++j)
                if (a[j] < a[i])
                { t = a[i]; a[i] = a[j]; a[j] = t; }
    }
    /*-----
    * Ghi du lieu vao file SapTang.Dat
    * -----*/
    static void Ghi()
    {
        StreamWriter f = File.CreateText(fn);
        f.WriteLine(n);
        for (int i = 0; i < n; ++i)
            f.Write(a[i] + ((i % 10 == 9) ? "\n" : " "));
        f.Close();
    }
    /*-----
    * Doc lai du lieu tu file
    * SapTang.dat de kiem tra
    * -----*/
    static void ShowFile()
    {
        Console.WriteLine(File.ReadAllText(fn));
    }
} // class
} // SangTao1

```

Bài 4.5. abc - sắp theo chỉ dẫn

Cho chuỗi S gồm N ký tự tạo từ các chữ cái 'a'..'z'. ta gọi S là chuỗi mẫu. Từ chuỗi mẫu S này người ta tạo ra N chuỗi thứ cấp bằng cách dịch chuỗi S qua trái i vị trí theo dạng vòng tròn, tức là i ký tự đầu chuỗi lần lượt được chuyển về cuối chuỗi, i

$= 0, 1, \dots, N - 1$. Như vậy *xâu thứ cấp* với $i = 0$ sẽ trùng với *xâu mẫu* S . Giả sử ta đã sắp tăng N *xâu* thu được theo trật tự từ điển. Hãy tìm *xâu thứ k* trong dãy.

Tên chương trình: `abc.pas`.

Dữ liệu vào: tệp văn bản `abc.inp` có cấu trúc như sau:

- Dòng thứ nhất chứa hai số tự nhiên N và k cách nhau qua dấu cách, $6 \leq N \leq 500$, $1 \leq k \leq N$. N cho biết chiều dài *xâu* S , k cho biết vị trí của *xâu thứ cấp* trong dãy được sắp tăng theo thứ tự từ điển.
- Dòng thứ hai: *xâu mẫu* S .

Dữ liệu ra: tệp văn bản `abc.out` gồm một dòng chứa *xâu thứ k* trong dãy được sắp.

Thí dụ:

<code>abc.inp</code>	<code>abc.out</code>
6 3 dabdec	cdabde

Bài giải

Ta gọi *xâu s* ban đầu là *xâu mẫu*, các *xâu* được sinh ra bởi phép quay là *xâu thứ cấp*. Để ý rằng các *xâu mẫu* cũng là một *xâu thứ cấp* ứng với phép quay 0 vị trí. Ta có thể nhận được *xâu thứ cấp thứ i* bằng cách duyệt *xâu mẫu* theo vòng tròn kể từ vị trí thứ i về cuối, đến vị trí thứ n . Sau đó duyệt tiếp tục từ vị trí thứ 1 đến vị trí thứ $i - 1$. Ta kí hiệu *xâu thứ cấp thứ i* là $[i]$. Thí dụ, nếu *xâu mẫu* $s = \text{'dabdec'}$ thì *xâu thứ cấp thứ 2* sẽ là $[2] = \text{'abdec d'}$. Để tìm *xâu thứ k* trong dãy được sắp, trước hết ta cần sắp tăng các *xâu* đó theo trật tự từ điển sau đó lấy *xâu thứ k* trong dãy được sắp làm kết quả. Để sắp tăng được các *xâu* này mà không phải sinh ra các *xâu* mới ta dùng một mảng phụ `id` gọi là mảng chỉ dẫn. Trước khi sắp ta gán $id[i] := i$. Sau khi sắp thì $id[i]$ sẽ cho biết tại vị trí thứ i trong dãy được sắp là *xâu thứ cấp* nào. Trong thí dụ trên, $id[3] = 6$ là *xâu thứ cấp* $[6]$. Giá trị 3 cho biết cần tìm *xâu thứ $k = 3$* trong dãy sắp tăng các *xâu thứ cấp*. Giá trị 6 cho biết *xâu* cần tìm là *xâu thứ 6* trong dãy các *xâu thứ cấp* được sinh ra lúc đầu, tức là lúc chưa sắp.

		Xâu thứ cấp	Sắp tăng	$id[i] = ?$
①	$[1] = s$	dabdec	$[2]$	2
②	$[2]$	abdec d	$[3]$	3
③	$[3]$	bdecda	$[6]$	6
④	$[4]$	dec dab	$[1]$	1
⑤	$[5]$	ecdabd	$[4]$	4
⑥	$[6]$	cdabde	$[5]$	5

*Sắp chỉ dẫn các *xâu thứ cấp**

Thuật toán *QuickSort* sắp nhanh các *xâu thứ cấp* do Hoare đề xuất có độ phức tạp $N \cdot \log_2 N$ được trình bày như sau:

```

Init: for i:=1 to n do id[i]:=i;
procedure idquicksort(d,c: integer);
var i, j, m, y: integer;
begin
  i:=d;
  j:=c;

```

```

m:=id[(i+j) div 2]; {phan tu giua}
while i <= j do
begin
  while Sanh(id[i],m)<0 do inc(i); {doan trai}
  while Sanh(m,id[j])<0 do dec(j); {doan phai}
  {doi cho neu can}
  if (i <= j) then
  begin
    y:= id[i];
    id[i]:= id[j];
    id[j]:= y;
    inc(i); dec(j);
  end;
end;
if d < j then idquicksort(d,j);
if i < c then idquicksort(i,c);
end;

```

Hàm $\text{Sanh}(i, j)$ so sánh hai xâu thứ cấp $[i]$ và $[j]$ theo thứ tự từ điển và cho giá trị -1 nếu xâu thứ cấp $[i]$ nhỏ hơn xâu thứ cấp $[j]$, cho giá trị 1 nếu xâu thứ cấp $[i]$ lớn hơn xâu thứ cấp $[j]$ và 0 nếu hai xâu này giống nhau. Để so sánh hai xâu theo trật tự từ điển ta lần lượt duyệt từng cặp kí tự của mỗi xâu. Nếu hai xâu giống nhau tại mọi vị trí thì ta gán trị 0 cho hàm Sanh . Ngược lại, nếu tìm được vị trí khác nhau đầu tiên thì ta so sánh hai kí tự $s[i]$ và $s[j]$ tại vị trí đó và gán cho hàm Sanh giá trị -1 nếu $s[i] < s[j]$, ngược lại, tức là khi $s[i] > s[j]$ thì gán giá trị 1 cho hàm Sanh .

Ta chỉ cần lưu ý là việc duyệt xâu phải thực hiện trên vòng tròn theo chiều quay của kim đồng hồ.

```

function Sanh(i,j: integer): integer;
var k: integer;
begin
  for k:=1 to n do
  begin
    if s[i] <> s[j] then
    begin
      if s[i] < s[j] then Sanh:=-1
      else Sanh:=1;
      exit;
    end;
    if i=n then i:=1 else inc(i);
    if j=n then j:=1 else inc(j);
  end;
  Sanh:=0;
end;

```

Hoare cũng cung cấp thêm thuật toán tìm phần tử thứ k trong dãy được sắp với độ phức tạp $2N$. Ta vận dụng thuật toán này cho bài toán abc . Bản chất thuật toán này là như sau. Ta cũng sắp tăng các xâu thứ cấp theo thuật toán QuickSort nhưng không sắp hết mà chỉ quan tâm đến đoạn dữ liệu trong mảng có chứa phần tử thứ k . Lưu ý rằng sau một lần chia dữ liệu của đoạn $id[d..c]$ ta thu được ba đoạn: đoạn $id[d..j]$, $id[j+1..i-1]$ và $id[i..c]$, trong đó đoạn giữa là $id[j+1..i-1]$ đã được sắp. Nếu k rơi vào đoạn thứ nhất là $id[d..j]$ hoặc đoạn thứ ba là $id[i..c]$ thì ta chỉ cần sắp tiếp đoạn đó. Hàm $\text{Find}(k)$ cho ra vị trí gốc của xâu thứ cấp sẽ đứng thứ k trong dãy đã sắp. Trong thí dụ trên $\text{Find}(3)=6$.

```

(*-----
                        Tim phan tu thu k
-----*)
function Find(k: integer):integer;
var d, c, i, j, m, y: integer;
begin
  d:=1 ;
  c:=n;
  while d <= c do
    begin
      i:=d;
      j:=c;
      m:=id[(i+j) div 2]; {phan tu giua}
      while i <= j do
        begin
          while Sanh(id[i],m)<0 do inc(i); {doan trai}
          while Sanh(m,id[j])<0 do dec(j); {doan phai}
          {doi cho neu can}
          if (i <= j) then
            begin
              y:= id[i];
              id[i]:= id[j];
              id[j]:= y;
              inc(i); dec(j);
            end;
          end;
          if j < k then d:=i;
          if k < i then c:=j;
        end;
      find:=id[k];
    end;
end;

(* Pascal *)

(*-----
                        ABC: Tim phan tu thu k
-----*)
program ABC;
{$B-}
uses crt;
const
  MN = 501;
  nl = #13#10; {xuong dong}
  bl = #32; {dau cach}
  fn = 'abc.inp';
  gn = 'abc.out';
type
  MI = array[0..MN] of integer;
  MC = array[0..MN] of char;
var
  f,g: text;
  s: MC;
  id: MI;

```

```

n,k: integer;
(*-----
      Doc du lieu:
      n: chieu dai xau s,
      k: vi tri xau thu cap trong day da sap
-----*)
procedure Doc;
var i: integer;
begin
  assign(f,fn); reset(f);
  readln(f,n,k);
  for i:=1 to n do read(f,s[i]);
  close(f);
end;
(*-----
      So sanh 2 xau thu cap [i] va [j].
      Sanh(i,j)
      = 0: neu [i] = [j]
      = -1: neu [i] < [j]
      = 1 neu [i] > [j]
-----*)
function Sanh(i,j: integer): integer;
var k: integer;
begin
  for k:=1 to n do
  begin
    if s[i] <> s[j] then
      begin
        if s[i] < s[j] then Sanh:=-1
          else Sanh:=1;
        exit;
      end;
    if i=n then i:=1 else inc(i);
    if j=n then j:=1 else inc(j);
  end;
  Sanh:=0;
end;
(*-----
      Tim phan tu thu k
-----*)
function Find(k: integer):integer;
var d, c, i, j, m, y: integer;
begin
  d:=1 ;
  c:=n;
  while d <= c do
  begin
    i:=d;
    j:=c;
    m:=id[(i+j) div 2]; {phan tu giua}
    while i <= j do
      begin
        while Sanh(id[i],m)<0 do inc(i); {doan trai}

```

```

        while Sanh(m,id[j])<0 do dec(j); {doan phai}
        {doi cho neu can}
        if (i <= j) then
            begin
                y:= id[i];
                id[i]:= id[j];
                id[j]:= y;
                inc(i); dec(j);
            end;
        end;
        if j < k then d:=i;
        if k < i then c:=j;
    end;
    find:=id[k];
end;
{-----}
        Ghi ket qua vao tep
{-----}
procedure Ghi(k: integer);
var i: integer;
begin
    assign(g,gn); rewrite(g);
    for i:=1 to n do
        begin
            write(g,s[k]);
            if k=n then k:=1 else inc(k);
        end;
    close(g);
end;
procedure run;
var i:integer;
begin
    Doc;
    for i:=1 to n do id[i]:=i;
    Ghi(find(k));
end;
BEGIN
    run;
END.

```

// C#

```

using System;
using System.IO;
namespace SangTao1
{
    /*-----
    *   Tim xau mau thu k voi do phuc tap 2N
    *   -----*/
    class abc
    {
        const int mn = 500;
        const string fn = "abc.inp";
    }
}

```

```

const string gn = "abc.out";
static string s;
static int n = 0; // chieu dai xau mau
static int k = 0; // xau thu k
static int[] id;

static void Main()
{
    Run();
    Console.ReadLine();
} // Main

static void Run()
{
    Doc();
    Console.WriteLine(n + " " + k + " " + s);
    id = new int[n];
    for (int i = 0; i < n; ++i) id[i] = i;
    PhanTuGiua();
    Ghi();
    Test();
    Console.WriteLine("\n Fini");
}
/*-----
 * Ghi dong thu k trong
 * day da sap vao file gn
 * -----*/
static void Ghi()
{
    StreamWriter g = File.CreateText(gn);
    int j = id[k-1];
    for (int i = 0; i < n; ++i)
        g.Write(s[(j + i) % n]);
    g.Close();
}
/*-----
 * Hien thi dong thu s[j...]
 * -----*/
static void PrintLine(int j)
{
    for (int i = 0; i < n; ++i)
        Console.Write(s[(j + i) % n]);
    Console.WriteLine();
}
static void Doc()
{
    char[] cc = new char[] { ' ', '\n', '\t', '\r' };
    string [] ss = (File.ReadAllText(fn)).Split(cc,
        StringSplitOptions.RemoveEmptyEntries);
    n = int.Parse(ss[0].Trim()); // do dai xau mau
    k = int.Parse(ss[1].Trim()); // xau thu k
    s = ss[2]; // xau mau
}

```

```

static void PhanTuGiua() // Tim phan tu thu k
{
    int m;
    int d = 0;
    int c = n - 1;
    int i = 0;
    int j = 0;
    int t = 0;
    int k0 = k - 1; // xau thu k tinh tu 1
    while (d <= c)
    {
        i = d;
        j = c;
        m = id[(i + j) / 2];
        while (Sanh(id[i], m) < 0) ++i;
        while (Sanh(m, id[j]) < 0) --j;
        if (i <= j)
        {
            t = id[i]; id[i] = id[j]; id[j] = t;
            ++i; --j;
        }
        if (j < k0) d = i;
        if (k < i) c = j;
    }
}
// so sanh 2 xau thu cap s[x...] va s[y...]
static int Sanh(int x, int y)
{
    int ix = 0;
    int iy = 0;
    for (int i = 0; i < n; ++i)
    {
        ix = (x + i) % n;
        iy = (y + i) % n;
        if (s[ix] != s[iy])
            return (s[ix] < s[iy]) ? -1 : 1;
    }
    return 0;
}
static void IdQSort(int d, int c) // sap theo chi dan
{
    int i = d;
    int j = c;
    int m = id[(i + j) / 2];
    int t = 0;
    while (i <= j)
    {
        while (Sanh(id[i], m) < 0) ++i;
        while (Sanh(m, id[j]) < 0) --j;
        if (i <= j)
        {
            t = id[i]; id[i] = id[j]; id[j] = t;
            ++i; --j;
        }
    }
}

```

```

    }
}
    if (d < j) IdQSort(d, i);
    if (i < c) IdQSort(i, c);
}
/*-----
 * Kiểm tra lại bằng cách dùng
 * thuật toán QSort theo chỉ dẫn
 * -----*/
static void Test()
{
    Console.WriteLine("\n Xau mau: " + s);
    for (int i = 0; i < n; ++i) id[i] = i;
    IdQSort(0, n - 1);
    Console.WriteLine("\n Day sap tang: \n");
    for (int i = 0; i < n; ++i)
    {
        Console.Write((i + 1) + ". ");
        PrintLine(id[i]);
    }
    Console.WriteLine();
    Console.WriteLine("\n Xau thu " + k);
    PrintLine(id[k-1]);
    Console.WriteLine("\n Xau ghi trong file " + gn);
    Console.WriteLine(File.ReadAllText(gn));
}
} // class
} // SangTao1

```

Bài 4.6. Xâu mẫu

Một tệp văn bản f có tên `STRINGS.INP` chứa các xâu kí tự, mỗi dòng ghi một xâu có chiều dài tối đa 250 kí tự. Xâu đầu tiên được gọi là xâu mẫu s . Lập trình:

Đọc xâu mẫu s từ tệp f , ghi vào tệp văn bản g có tên `STRINGS.OUT`. Sau đó đọc từng xâu x còn lại của f , với mỗi xâu x cần ghi vào g các thông tin sau:

- nội dung xâu x ;
- hai số v và d cách nhau qua dấu cách, trong đó v là vị trí xuất hiện và d là chiều dài lớn nhất của khúc đầu của x trong xâu mẫu s . Nếu vô nghiệm thì ghi -1 0.

Thí dụ:

STRINGS.INP	STRINGS.OUT
cabxabc dab	cabxabc dab
abcd	abcd
cdaeh	5 4
	cdaeh
	7 3

Thuật toán

Với mỗi xâu kí tự w ta kí hiệu $w[i..j]$, $i \leq j$, và gọi là đoạn, là xâu gồm dãy kí tự liên tiếp từ $w[i]$ đến $w[j]$ trong xâu w . Thí dụ, nếu $w = \text{'cabxabc dab'}$ thì $w[5..8] = \text{'abcd'}$. Gọi

s là chuỗi mẫu, x là chuỗi cần khảo sát. Nhiệm vụ của ta là tìm vị trí v và chiều dài lớn nhất d để $x[1..d] = s[v..(v + d - 1)]$. Ta vận dụng kỹ thuật tổ chức hậu tố như sau.

Hậu tố của một chuỗi là đoạn cuối của chuỗi đó. Như vậy một chuỗi có chiều dài n sẽ có n hậu tố. Thí dụ, với chuỗi mẫu $s[1..10] = \text{'cabxabcddab'}$ ta có 10 hậu tố sau đây:

```

s[1..10] = 'cabxabcddab'
s[2..10] = 'abxabcddab'
s[3..10] = 'bxabcddab'
s[4..10] = 'xabcddab'
s[5..10] = 'abcddab'
s[6..10] = 'bcdab'
s[7..10] = 'cdab'
s[8..10] = 'dab'
s[9..10] = 'ab'
s[10..10] = 'b'

```

Như vậy, hậu tố sau sẽ nhận được từ hậu tố sát trước nó bằng cách bỏ đi ký tự đầu tiên.

Trước hết ta sắp tăng các hậu tố của chuỗi mẫu s theo trật tự từ điển. Sử dụng một mảng chỉ dẫn id , trong đó $id[i]$ trỏ đến vị trí đầu tiên của hậu tố trong chuỗi mẫu. Cụ thể là, nếu $id[i] = k$ thì hậu tố tương ứng sẽ là $s[k..n]$. Sau khi sắp tăng các hậu tố của chuỗi mẫu $s[1..10] = \text{'cabxabcddab'}$ ta thu được:

i	id[i]	Hậu tố	Chuỗi
1	9	$s[9..10]$	ab
2	5	$s[5..10]$	abcddab
3	2	$s[2..10]$	abxabcddab
4	10	$s[10..10]$	b
5	6	$s[6..10]$	bcdab
6	3	$s[3..10]$	bxabcddab
7	1	$s[1..10]$	cabxabcddab
8	7	$s[7..10]$	cdab
9	8	$s[8..10]$	dab
10	4	$s[4..10]$	xabcddab

Sắp tăng theo chỉ dẫn các hậu tố của chuỗi

$s[1..10] = \text{'cabxabcddab'}$

Việc còn lại là so sánh chuỗi x với các hậu tố $s[i..j]$ để tìm khúc đầu chung dài nhất giữa chúng. Thí dụ, với $x[1..4] = \text{'abcd'}$ thì khúc đầu chung dài nhất tìm được với hậu tố $s[5..10]$ do $id[2]$ trỏ tới. Vị trí v tìm được sẽ là 5 và chiều dài lớn nhất d sẽ là 4.

Phần chính của chương trình sẽ như sau:

```

procedure Run;
begin
  ...
  n := length(s);

```

```

for i:=1 to n do id[i]:=i;
IdQuikSort(1,n);
while not seekeof(f) do
begin
  readln(f,x);
  writeln(g,x);
  Tim; {xac dinh v và d}
  writeln(g,v,BL,d);
end;
end;

```

Đề ý rằng với mỗi xâu x , nếu phần tử đầu tiên của x là $x[1]$ không trùng với phần tử đầu tiên của hậu tố h thì chiều dài của khúc đầu chung của chúng sẽ bằng 0. Nhờ nhận xét này và do dãy các hậu tố đã được sắp tăng nên với mỗi xâu x , trước hết ta gọi hàm `Binsearch` để thực hiện tìm kiếm nhị phân phần tử $x[1]$ trong dãy gồm các phần tử đầu tiên của các hậu tố, sau đó ta thực hiện việc duyệt tìm.

```

procedure Tim;
var
  i,Len: integer;
begin
  v:=BinSearch; d := 0;
  if v=0 then exit;
  Maxlen:=0;
  for i:=v to n do
  begin
    if s[id[i]]<>x[1] then exit;
    Len:=Eqsx(id[i]);
    if Len > d then
      begin
        d:=Len;
        v:=id[i];
      end;
  end;
end;

```

Hàm `BinSearch` sẽ cho ra chỉ dẫn tới hậu tố h đầu tiên thoả điều kiện $h[1] = x[1]$.

```

(*-----
  Tim xuất hiện của x[1] trong dãy
  đã sắp các hậu tố
-----*)
function BinSearch:integer;
var
  d,c,m: integer;
begin
  d:=1;
  c:=n;
  repeat
    m:=(d+c) div 2;
    if x[1]>s[id[m]] then d:=m+1
    else c:=m;
  until d=c;

```

```

        if x[1]<>s[id[d]] then Binsearch := -1
        else BinSearch := d;
    end;

```

Hàm Eqsx(i) cho ta chiều dài lớn nhất của khúc đầu chung giữa hậu tố s[i..n] và xâu x.

```

    (*-----
        Khúc dau dai nhat giua
        hau to s[i..n] va x
    -----*)
    function Eqsx(i:integer): integer;
    var
        k,m:integer;
    begin
        m:=min(length(x),n-i+1);
        for k:=1 to m do
            if s[i+k-1]<>x[k] then
                begin
                    Eqsx:=k-1;
                    exit;
                end;
            Eqsx:=m;
        end;
    end;

(* Pascal *)

    (*-----
        STRINGS: Xau mau
    -----*)
    program Strings;
    {$B-}
    uses crt;
    const
        MN = 255;
        cd = #0; {ki tu trong}
        cc = #255; {ki tu cuoi cua bang ma ASCII}
        BL=#32; {dau cach}
        fn = 'strings.inp'; {tep vao}
        gn = 'strings.out'; {tep ra}
    type
        mbl = array[0..MN] of integer;
    var
        f,g: text;
        s,x: string; {s: xau mau}
        id: mbl; {chi dan}
        n: integer; {chieu dai xau mau s}
        v,d: integer;
        {v: vi tri xuat hien khuc dau cua x trong xau mau s}
        {d: maxlen}
    (*-----
        min cua 2 phan tu
    -----*)

```

```

function min(a,b:integer):integer;
begin
    if a<=b then min:=a
    else min:=b;
end;
(*-----
Tim xuat hien cua x[1] trong day da sap cac hau to
-----*)
function BinSearch:integer;
var
    d,c,m: integer;
begin
    d:=1;
    c:=n;
    repeat
        m:=(d+c) div 2;
        if x[1]>s[id[m]] then d:=m+1
        else c:=m;
    until d=c;
    if x[1]<>s[id[d]] then Binsearch:=0
    else BinSearch:=d;
end;
(*-----
so sanh 2 hau to trong s:
s[i..n] va s[j..n]
-----*)
function sanh(i,j:integer):integer;
var k:integer;
begin
    for k:=0 to min(n-i,n-j) do
        if s[i+k]<>s[j+k] then
            begin
                if s[i+k]<s[j+k] then sanh:=-1
                else sanh:=1;
                exit;
            end;
    if i=j then sanh:=0
    else if i<j then sanh:=1
    else sanh:=-1;
end;
(*-----
Quick sort cac hau to theo chi dan
-----*)
procedure IdQuickSort(d,c: integer);
var i,j,m,t: integer;
begin
    i:=d; {dau}
    j:=c; {cuoi}
    m:=id[(i+j) div 2]; {phan tu giua}
    while i<=j do
        begin

```

```

        while sanh(id[i],m)<0 do inc(i);
        while sanh(id[j],m)>0 do dec(j);
        if (i<=j) then
            begin
                t:=id[i];
                id[i]:=id[j];
                id[j]:=t;
                inc(i); dec(j);
            end;
        end;
        if d<j then IdQuickSort(d,j);
        if i<c then IdQuickSort(i,c);
    end;
    (*-----
    Khuc dau dai nhat giua hau to s[i..n] va x
    -----*)
    function Eqsx(i:integer): integer;
    var k,m:integer;
    begin
        m:=min(length(x),n-i+1);
        for k:=1 to m do
            if s[i+k-1]<>x[k] then
                begin
                    Eqsx:=k-1;
                    exit;
                end;
        Eqsx:=m;
    end;
    (*-----
    Tim vi tri va chieu dai lon nhat
    MaxLen giua cac hau to cua xau mau s va xau x
    -----*)
    procedure Tim;
    var i,Len: integer;
    begin
        v:=BinSearch;
        d:=0;
        if v=0 then exit;
        for i:=v to n do
            begin
                if s[id[i]]<>x[1] then exit;
                Len:=Eqsx(id[i]);
                if Len > d then
                    begin
                        d:=Len;
                        v:=id[i];
                    end;
            end;
    end;
    procedure Run;

```

```
var i:integer;
begin
  assign(f,fn);
  reset(f);
  assign(g,gn);
  rewrite(g);
  readln(f, s);
  writeln(g,s);
  n:= length(s);
  for i:=1 to n do id[i]:=i;
  IdQuickSort(1,n);
  while not seekeof(f) do
    begin
      readln(f,x);
      writeln(g,x);
      Tim;
      writeln(g,v,BL,d);
    end;
  close(f);
  close(g);
end;
BEGIN
  Run;
END.
```

Dữ liệu kiểm thử STRINGS.INP	Kết quả dự kiến STRINGS.OUT
cabxabcdab abcd cdaeh	cabxabcdab abcd 5 4 cdaeh 7 3

```
// C#
using System;
using System.IO;
namespace SangTao1
{
  /*-----
  *      Xau mau
  * -----*/
  class Strings
  {
    const string fn = "strings.inp";
    const string gn = "strings.out";
    static string s; // xau mau
    static string x; // xau can xu ly
```

```

        static int[] id;
static int v = 0; // vị trí xuất hiện khúc đầu x trg s
        static int d = 0; // chiều dài x trong s
        static int n = 0; // chiều dài xau mau
        static void Main()
        {
            Run();
            Test();
            Console.ReadLine();
        } // Main

        // Đọc lại file gn để kiểm tra kết quả
        static void Test()
        {
            Console.WriteLine("\n Kiểm tra lại kết quả \n\n");
            Console.WriteLine("\n Input: " +
                File.ReadAllText(fn));
            Console.WriteLine("\n Output: " +
                File.ReadAllText(gn));
        }
        static void Run()
        {
            StreamReader f = File.OpenText(fn);
            StreamWriter g = File.CreateText(gn);
            // Bỏ qua các dòng trong đầu tiên
            while ((s = (f.ReadLine()).Trim()) == "") ;
            n = s.Length; // Chiều dài xau mau
            id = new int[n];
            // Khởi tạo cho index
            for (int i = 0; i < n; ++i) id[i] = i;
            IdQSort(0, n - 1);
            Console.WriteLine(" Xau mau: " + s + "\n\n");
            SPrint();
            g.WriteLine(s);
            while ((x = f.ReadLine()) != null)
            {
                x = x.Trim();
                if (x != "")
                {
                    Console.WriteLine(x);
                    g.WriteLine(x);
                    Find();
                    g.WriteLine(v + " " + d);
                }
            }
        }
    }

```

```

        f.Close(); g.Close();
    }
    static void Find()
    {
        v = BinSearch(x[0]); // hau to co ki tu dau la x[0]
        d = 0;
        if (v < 0) return;
        for (int i = v; i < n; ++i)
        {
            int j = id[i];
            if (s[j] != x[0]) return;
            int k = ComLen(x, j);
            if (d < k) { v = j + 1; d = k; }
        }
    }
    // MaxLen khuc dau cua x va hau to s[j...
    static int ComLen(string x, int j)
    {
        int minlen = Min(x.Length, n - j);
        for (int i = 0; i < minlen; ++i)
            if (x[i] != s[j + i]) return i;
        return minlen;
    }
    static int Min(int a, int b)
    { return (a < b) ? a : b; }
    static int BinSearch(char c)
    {
        int i = 0;
        int j = n - 1;
        int m = 0;
        while (i < j)
        {
            m = (i + j) / 2;
            if (s[id[m]] < c) i = m + 1;
            else j = m;
        }
        return (s[id[i]] == c) ? i : -1;
    }
    // Hien thi day duoc sap cac hau to
    // cua s de kiem tra
    static void SPrint()
    {
        Console.WriteLine("\n Cac hau to sap tang: \n");
        for (int i = 0; i < n; ++i)
            Console.WriteLine(s.Substring(id[i], n - id[i]));
    }
}

```

```

    }
    static void IdQSort(int d, int c)
    {
        int i = d;
        int j = c;
        int m = id[(i + j) / 2];
        int t = 0;
        while (i <= j)
        {
            while (Sanh(id[i], m) < 0) ++i;
            while (Sanh(m, id[j]) < 0) --j;
            if (i <= j)
            {
                t = id[i]; id[i] = id[j]; id[j] = t;
                ++i; --j;
            }
        }
        if (d < j) IdQSort(d, j);
        if (i < c) IdQSort(i, c);
    }
    static int Sanh(int x, int y)
    {
        int ix = 0;
        int iy = 0;
        for (int i = 0; i < n; ++i)
        {
            ix = (x + i) % n;
            iy = (y + i) % n;
            if (s[ix] != s[iy])
                return (s[ix] < s[iy]) ? -1 : 1;
        }
        return 0;
    }
} // Strings
} // SangTao1

```

CHƯƠNG 5

PHƯƠNG PHÁP THAM LAM

Phương pháp tham lam gợi ý chúng ta tìm một trật tự hợp lý để duyệt dữ liệu nhằm đạt được mục tiêu một cách chắc chắn và nhanh chóng. Thông thường, dữ liệu được duyệt theo một trong hai trật tự là tăng hoặc giảm dần theo một chỉ tiêu nào đó. Một số bài toán đòi hỏi những dạng thức cải biên của hai dạng nói trên.

Bài 5.1. Băng nhạc

Người ta cần ghi N bài hát, được mã số từ 1 đến N , vào một băng nhạc có thời lượng tính theo phút đủ chứa toàn bộ các bài đã cho. Với mỗi bài hát ta biết thời lượng phát của bài đó. Băng sẽ được lắp vào một máy phát nhạc đặt trong một siêu thị. Khách hàng muốn nghe bài hát nào chỉ việc nhấn phím ứng với bài đó. Để tìm và phát bài thứ i trên băng, máy xuất phát từ đầu cuộn băng, quay băng để bỏ qua $i - 1$ bài ghi trước bài đó. Thời gian quay băng bỏ qua mỗi bài và thời gian phát bài đó được tính là như nhau. Tính trung bình, các bài hát trong một ngày được khách hàng lựa chọn với số lần (tần suất) như nhau. Hãy tìm cách ghi các bài trên băng sao cho tổng thời gian quay băng trong mỗi ngày là ít nhất.

Dữ liệu vào được ghi trong tệp văn bản tên **BANGNHAC . INP**.

- Dòng đầu tiên là số tự nhiên N cho biết số lượng bài hát.
- Tiếp đến là N số nguyên dương thể hiện dung lượng tính theo phút của mỗi bài. Mỗi đơn vị dữ liệu cách nhau qua dấu cách.

Thí dụ dưới đây cho biết có $N = 3$ bài hát:

- Bài 1 phát trong thời gian 7 phút.
- Bài 2 phát trong thời gian 2 phút.
- Bài 3 phát trong thời gian 3 phút.

BANGNHAC . INP

3
7 2 3
1 12
19

BANGNHAC . OUT

Dữ liệu ra được ghi trong tệp văn bản

BANGNHAC . OUT theo dạng thức sau:

- N dòng đầu tiên thể hiện trật tự ghi bài hát trên băng: mỗi dòng gồm hai số nguyên dương j và d cách nhau bởi dấu cách, trong đó j là mã số của bài hát cần ghi, d là thời gian tìm và phát bài đó theo trật tự ghi này.
- Dòng thứ $n + 1$ ghi tổng số thời gian quay băng nếu mỗi bài hát được phát một lần trong ngày.

Với thí dụ trên, kết quả thu được sẽ như sau:

- Cần ghi lần lượt trên băng các bài theo trật tự : bài 2, bài 3, bài 1;
- Để tìm và phát bài 2 cần 2 phút;
- Để tìm và phát bài 3 cần 5 phút;
- Để tìm và phát bài 1 cần 12 phút;
- Tổng thời gian để tìm và phát mỗi bài một lần là: 19 phút.

Thuật toán

Giả sử ta có ba bài hát với số phút lần lượt như sau:

Mã số bài hát	①	②	③
Thời gian phát	7	2	3

Ta xét vài tình huống ghi băng để rút ra kết luận cần thiết.

Trật tự ghi trên băng	Thời gian phát
(x, y, z)	$t(x) + t(y) + t(z)$; t(i): thời gian tìm và phát bài i
(①, ②, ③)	$(7) + (7 + 2) + (7 + 2 + 3) = 28$ phút
(①, ③, ②)	$(7) + (7 + 3) + (7 + 3 + 2) = 29$ phút
(②, ①, ③)	$(2) + (2 + 7) + (2 + 7 + 3) = 23$ phút
(②, ③, ①)	$(2) + (2 + 3) + (2 + 3 + 7) = 19$ phút (phương án tối ưu)
(③, ①, ②)	$(3) + (3 + 7) + (3 + 7 + 2) = 25$ phút
(③, ②, ①)	$(3) + (3 + 2) + (3 + 2 + 7) = 20$ phút

Vậy phương án tối ưu sẽ là (②, ③, ①): ghi bài 2 rồi đến bài 3, cuối cùng ghi bài 1. Tổng thời gian theo phương án này là 19 phút.

Để có phương án tối ưu ta chỉ cần ghi băng theo trật tự tăng dần của thời lượng. Bài toán được cho với giả thiết băng đủ lớn để ghi được toàn bộ các bài. Dễ dàng sửa chương trình để vận dụng cho trường hợp dung lượng băng hạn chế trong M phút. Chương trình sắp xếp dữ liệu theo chỉ dẫn.

(* Pascal *)

```
(*-----
      BangNhac.pas
-----*)
program BangNhac;
uses crt;
const
  MN = 200; BL = #32; {dau cach}
  fn = 'Bangnhac.inp'; gn = 'Bangnhac.out';
var
  a,id: array[1..MN] of integer;
  f, g: text;
  n: integer;
{-----
  Doc du lieu tu input file vao mang a
  -----}
procedure Doc;
var i,k: integer;
begin
  assign(f,fn); reset(f); read(f,n);
  for i := 1 to n do read(f,a[i]);
  close(f);
end;
{-----
  Khoi tri mang chi dan id
  quan li sap tang theo chi dan
  -----}
procedure InitID;
var i: integer;
begin
  for i := 1 to n do id[i] := i;
end;
{-----
      Sap tang theo chi dan
  -----}
```

```

procedure IDQuickSort(d,c: integer);
var i, j, m, x: integer;
begin
  i := d;
  j := c;
  m := a[id[(i+j) div 2]]; {phan tu giua}
  while i <= j do
    begin
      while a[id[i]] < m do inc(i);
      while a[id[j]] > m do dec(j);
      if i <= j then
        begin
          x := id[i];
          id[i] := id[j];
          id[j] := x;
          inc(i); dec(j);
        end;
      end;
    if d < j then IDQuickSort(d,j);
    if i < c then IDQuickSort(i,c);
  end;
  {-----
  Ghi ket qua vao output file
  -----}
procedure Ghi;
var i, t, tt: longint;
begin
  assign(g,gn); rewrite(g);
  t := 0; {thoi gian tim va phat 1 bai}
  tt := 0; {tong thoi gian cho n bai}
  for i := 1 to n do
    begin
      t := t + a[id[i]];
      tt := tt + t;
      writeln(g,id[i],BL,t);
    end;
  writeln(g,tt); close(g);
end;
BEGIN
  Doc; InitID; IDQuickSort(1,n); Ghi;
END.

```

// C#

```

using System;
using System.IO;
namespace SangTao1
{
  /*-----
  *           Bang nhac
  * -----*/
  class BangNhac
  {
    const string fn = "BangNhac.inp";

```

```

const string gn = "BangNhac.out";
static public Bang[] b;
static public int n = 0; // so bai hat
static void Main()
{
    Doc(); QSort(0, n-1);
    Ghi(); Test();
    Console.WriteLine("\n Fini ");
    Console.ReadLine();
}
static void Ghi()
{
    StreamWriter g = File.CreateText(gn);
    int t = 0; // tg tim va phat 1 bai
    int tt = 0; // tong tg tim va phat n bai
    for (int i = 0; i < n; ++i)
    {
        t += b[i].len;
        tt += t;
        g.WriteLine(b[i].id + " " + t);
    }
    g.WriteLine(tt); g.Close();
}
static void QSort(int d, int c)
{
    int i = d, j = c, m = b[(i+j)/2].len;
    Bang t = new Bang(0,0);
    while (i <= j)
    {
        while (b[i].len < m) ++i;
        while (m < b[j].len) --j;
        if (i <= j)
        {
            t = b[i]; b[i] = b[j];
            b[j] = t; ++i; --j;
        }
    }
    if (d < j) QSort(d, j);
    if (i < c) QSort(i, c);
}
// Doc lai file gn de kiem tra ket qua
static void Test() tự viết
static void Doc()
{
    int [] a = Array.ConvertAll(
        (File.ReadAllText(fn)).Split(
            new char[] { '\n', ' ', '\t', '\0', '\r'},
            StringSplitOptions.RemoveEmptyEntries),
        new Converter<string, int>(int.Parse));
    n = a[0];
}

```

```

        b = new Bang[n];
        for (int i = 1; i <= n; ++i)
            b[i-1] = new Bang(a[i], i);
    }
    public struct Bang
    {
        public int len; // thời lượng
        public int id; // số hiệu 1, 2, ...
        public Bang(int t, int nn)
        { len = t; id = nn; }
    }
} // BangNhac
} // SangTao1

```

Bài 5.2. Xếp việc

Có N công việc cần thực hiện trên một máy tính, mỗi việc đòi hỏi đúng 1 giờ máy. Với mỗi việc ta biết thời hạn phải nộp kết quả thực hiện sau khi hoàn thành việc đó và tiền thưởng thu được nếu nộp kết quả trước hoặc đúng thời điểm quy định. Chỉ có một máy tính trong tay, hãy lập lịch thực hiện đủ N công việc trên máy tính sao cho tổng số tiền thưởng thu được là lớn nhất và thời gian hoạt động của máy là nhỏ nhất. Giả thiết rằng máy được khởi động vào đầu ca, thời điểm $t = 0$ và chỉ tắt máy sau khi đã hoàn thành đủ N công việc.

Dữ liệu vào: tệp văn bản **viiec.inp**:

- Dòng đầu tiên là số N .
- N dòng tiếp theo: mỗi việc được mô tả bằng hai số tự nhiên, số thứ nhất là thời hạn giao nộp, số thứ hai là tiền thưởng. Các số cách nhau bởi dấu cách.

Thí dụ:

viiec.inp

```

4
1 15
3 10
5 100
1 27

```

Ý nghĩa: Cho biết có 4 việc với các thông tin sau:

- Việc thứ nhất phải nộp không muộn hơn thời điểm 1 (giờ) với tiền thưởng 15 (ngàn đồng);
- Việc thứ hai phải nộp không muộn hơn thời điểm 3 (giờ) với tiền thưởng 10 (ngàn đồng);
- Việc thứ ba phải nộp không muộn hơn thời điểm 5 (giờ) với tiền thưởng 100 (ngàn đồng);
- Việc thứ tư phải nộp không muộn hơn thời điểm 1 (giờ) với tiền thưởng 27 (ngàn đồng).

Dữ liệu ra: tệp văn bản **viiec.out**:

- N dòng đầu tiên, dòng thứ t ghi một số tự nhiên i cho biết việc thứ i được làm trong giờ t .
- Dòng cuối cùng ghi tổng số tiền thu được.

Với thí dụ trên, tệp **viiec.out** sẽ như sau:

viiec.out

```

4
2
3
1
137

```

Ý nghĩa:

- Giờ thứ 1 thực hiện việc 4 và nộp đúng hạn nên được thưởng 27;
- Giờ thứ 2 thực hiện việc 2 và nộp trước hạn nên được thưởng 10;
- Giờ thứ 3 thực hiện việc 3 và nộp trước hạn nên được



thường 100;

- Giờ thứ 4 thực hiện việc 1;
- Tổng tiền thưởng thu được do đã hoàn thành đúng hạn ba việc 4, 2 và 3 là $27 + 10 + 100 = 137$.

Thuật toán

Ta ưu tiên cho những việc có tiền thưởng cao, do đó ta sắp các việc giảm dần theo tiền thưởng. Với mỗi việc k ta đã biết thời hạn giao nộp việc đó là $h = t[k]$. Ta xét trục thời gian b . Nếu giờ h trên trục đó đã bận do việc khác thì ta tìm từ thời điểm h trở về trước một thời điểm có thể thực hiện được việc k đó. Nếu tìm được một thời điểm m như vậy, ta đánh dấu bằng mã số của việc đó trên trục thời gian b , $b[m] := k$. Sau khi xếp việc xong, có thể trên trục thời gian còn những thời điểm rỗi, ta dồn các việc đã xếp về phía trước nhằm thu được một lịch làm việc trù mật, tức là không có giờ trống. Cuối cùng ta xếp tiếp những việc trước đó đã xét nhưng không xếp được. Đây là những việc phải làm nhưng không thể nộp đúng hạn nên sẽ không có tiền thưởng. Với thí dụ đã cho, $N = 4$, thời hạn giao nộp $t = (1, 3, 5, 1)$ và tiền thưởng $a = (15, 10, 100, 27)$ ta tính toán như sau:

- Khởi trị: trục thời gian với 5 thời điểm ứng với $T_{\max} = 5$ là thời điểm muộn nhất phải nộp kết quả, $T_{\max} = \max \{ \text{thời hạn giao nộp} \}$, $b = (0, 0, 0, 0, 0)$.
- Chọn việc 3 có tiền thưởng lớn nhất là 100. Xếp việc 3 với thời hạn $t[3] = 5$ vào h : $h[5] = 3$. Ta thu được $h = (0, 0, 0, 0, 3)$.
- Chọn tiếp việc 4 có tiền thưởng 27. Xếp việc 4 với thời hạn $t[4] = 1$ vào h : $h[1] = 4$. Ta thu được $h = (4, 0, 0, 0, 3)$.
- Chọn tiếp việc 1 có tiền thưởng 15. Xếp việc 1 với thời hạn $t[1] = 1$ vào h : Không xếp được vì từ thời điểm 1 trở về trước trục thời gian $h[1..1]$ đã kín. Ta thu được $h = (4, 0, 0, 0, 3)$.
- Chọn nốt việc 2 có tiền thưởng 10. Xếp việc 2 với thời hạn $t[2] = 3$ vào h : $h[3] = 2$.
- Ta thu được $h = (4, 0, 2, 0, 3)$.
- Dồn việc trên trục thời gian h , ta thu được $h = (4, 2, 3, 0, 0)$.
- Xếp nốt việc phải làm mà không có thưởng, ta thu được $h = (4, 2, 3, 1)$.
- Ca làm việc kéo dài đúng $N = 4$ giờ.
- Nếu không muốn sắp giảm mảng tiền thưởng a theo chỉ dẫn ta có thể sắp song song a và id như mô tả trong chương trình.

Trong chương trình dưới đây ta sử dụng mảng id với hai mục đích: $id[i] = v > 0$ cho biết việc v đứng thứ i trong dãy được sắp giảm theo giá trị tiền thưởng và việc v chưa được xếp. $id[i] = v < 0$ cho biết việc v đã xếp xong trong lần duyệt đầu tiên.

(* Pascal *)

```
(*-----
  VIEC.PAS Chon viec
-----*)
program viec;
uses crt;
const
  MN = 200; bl = #32; {dau cach}
  nl = #13#10; {xuong dong}
  fn = 'viec.inp'; {input file}
```

```

    gn = 'viec.out'; {output file}
var
  a,id,t: array[1..MN] of integer;
  {a: tien thuong, t: thoi han giao nop}
  {id: chi dan}
  h: array[0..MN] of integer; {truc thoi gian}
  N: integer; {so luong viec}
  f,g: text;
  M: integer; {so viec da xep}
  tt: longint; {tong so tien thuong}
(*-----
  Doc du lieu tu input file
-----*)
procedure Doc;
var i,k: integer;
begin
  assign(f,fn); reset(f);
  readln(f,N);
  for i := 1 to N do
    readln(f,t[i],a[i]);
  close(f);
end;
(*-----
  Khoi tri cho mang chi dan id
-----*)
procedure InitID;
var i: integer;
begin
  for i := 1 to N do id[i] := i;
end;
(*-----
  Sap giam a[1..N] theo chi dan
-----*)
procedure IDQuickSort(d,c: integer);
var i, j, m, k: integer;
begin
  i := d; j := c;
  m := a[id[(i+j) div 2]]; {phan tu giua}
  while i <= j do
    begin
      while a[id[i]] > m do inc(i);
      while a[id[j]] < m do dec(j);
      if i <= j then
        begin
          k := id[i];
          id[i] := id[j];
          id[j] := k;
          inc(i); dec(j);
        end;
    end;
  if d < j then IDQuickSort(d,j);
  if i < c then IDQuickSort(i,c);
end;

```

```

(*-----
Xep viec theo giai thuat tham lam
-----*)
procedure XepViec;
var i,k,v: integer;
begin
  fillchar(h,sizeof(h),0);
  for i := 1 to N do
    begin
      v := id[i]; {viec nao}
      for k := t[v] downto 1 do
        if h[k] = 0 then
          begin
            {xep duoc viec v tai thoi diem k}
            h[k] := v;
            id[i] := -v;
            break;
          end;
        end;
      end;
    end;
  end;
  (*-----
Don cac viec da xep trong h len phia truoc
va tinh tong tien thuong
-----*)
procedure DonViec;
var i: integer;
begin
  tt := 0;
  {tim gio trong dau tien trong h}
  for i := 1 to MN do
    if h[i] = 0 then
      begin
        M := i;
        break;
      end
    else tt := tt + a[h[i]];
  if M > N then exit;
  for i := M + 1 to MN do
    if h[i] > 0 then
      begin
        h[M] := h[i];
        tt := tt + a[h[i]];
        inc(M);
        if M > N then exit;
      end;
    end;
  end;
  end;
  (*-----
Xep not cac viec con lai
-----*)
procedure XepTiep;
var i: integer;
begin
  for i := 1 to N do

```

```

        if id[i] > 0 then
        begin
            h[M] := id[i];
            inc(M);
        end;
    end;
    (*-----
        Ghi ket qua
    -----*)
    procedure GhiTep;
    var i: integer;
    begin
        assign(g,gn); rewrite(g);
        for i := 1 to N do
            writeln(g,h[i]);
            writeln(g,tt); close(g);
        end;
    BEGIN
        Doc; InitID; IDQuickSort(1,n);
        XepViec; DonViec; XepTiep; GhiTep;

    END.

```

// C#

```

using System;
using System.IO;
namespace SangTao1
{
    /*-----
    *           Xep viec
    * -----*/
    class XepViec
    {
        const int mn = 280;
        const string fn = "Viec.inp";
        const string gn = "Viec.out";
        static public Viec [] v; // cac viec
        static public int n = 0; // so luong viec
        static public int tong = 0;
        static public int[] h;
        static public int k = 0;
        static void Main()
        {
            Doc(); QSort(0, n-1);
            Xep(); Ghi(); Test();
            Console.ReadLine();
        } // Main

        static void Xep()
        {
            // Tim Tmax

```

```

int tmax = 0;
for (int i = 0; i < n; ++i)
    if (v[i].t > tmax) tmax = v[i].t;
int tt = tmax + n + 1;
h = new int[tt];
// Khoi tri cho h
for (int i = 0; i < tt; ++i) h[i] = 0;
tong = 0;
for (int i = 0; i < n; ++i)
{
    int td = v[i].t;
    while (h[td] > 0) --td;
    if (td == 0)
        h[++tmax] = v[i].id; //viec ko thg
    else
    {
        h[td] = v[i].id;
        tong += v[i].thuong;
    }
}
// Dich cac viec len phia truoc
k = 0;
for (k = 1; k <= tmax; ++k)
    if (h[k] == 0) break;
for (int i = k + 1; i <= tmax; ++i)
    if (h[i] > 0)
        h[k++] = h[i];
}
static void Ghi() // Ghi file
{
    StreamWriter g = File.CreateText(gn);
    for (int i = 1; i < k; ++i)
        g.WriteLine(h[i]);
    g.WriteLine(tong); g.Close();
}
// Sap cac viec giam theo tien thuong
static void QSort(int d, int c)
{
    int i = d;
    int j = c;
    int m = v[(d + c) / 2].thuong;
    Viec t = new Viec(0, 0, 0);
    while (i <= j)
    {
        while (v[i].thuong > m) ++i;
        while (m > v[j].thuong) --j;
    }
}

```

```

        if (i <= j)
        {
            t = v[i]; v[i] = v[j]; v[j] = t;
            ++i; --j;
        }
    }
    if (d < j) QSort(d, j);
    if (i < c) QSort(i, c);
}
// Doc lai file gn de kiem tra ket qua
static void Test() tự viết
static void Doc()
{
    int [] a = Array.ConvertAll(
        (File.ReadAllText(fn)).Split(
            new char[] { '\n', ' ', '\t', '\0', '\r' },
            StringSplitOptions.RemoveEmptyEntries),
        new Converter<string, int>(int.Parse));
    n = a[0];
    v = new Viec[n];
    Console.WriteLine(" n = " + n);
    int k = 1;
    for (int i = 0; i < n; ++i)
        v[i] = new Viec(a[k++], a[k++], i+1);
}
public struct Viec
{
    public int t; // Thoi han giao nop
    public int thuong; // Tien thuong
    public int id; // Ma so
    public Viec(int th, int thg, int nn)
    { t = th; thuong = thg; id = nn; }
}
} // XepViec
} // SangTao1

```

Bài 5.3. Xếp ba lô

Có N vật (mặt hàng), với mỗi vật ta biết trọng lượng và giá trị của nó. Hãy xác định trọng lượng cần lấy ở một số vật để xếp vào một ba lô có sức chứa tối đa là M sao cho giá trị chứa trong ba lô là lớn nhất. Giả thiết là có thể lấy một tỉ lệ tùy ý ở mỗi vật.

Dữ liệu vào: Tập văn bản **balo.inp**:

- Dòng đầu tiên: hai giá trị nguyên dương N và M .
- N dòng tiếp theo, mỗi dòng chứa hai giá trị nguyên dương d và v cho mỗi vật, trong đó d là trọng lượng, v là giá trị tính theo một đơn vị trọng lượng của vật đó (đơn giá). Các số cách nhau qua dấu cách.

BALO.INP		BALO.OUT
5 30	Có $N = 5$ vật và sức chứa tối đa của ba lô là $M = 30$ (kg).	8
8 5	- Vật thứ nhất có trọng lượng 8, đơn giá 5	3
5 4	tr/kg,	0
4 2	- Vật thứ hai có trọng lượng 5, đơn giá 4,	3


```

-----*)
procedure IDQuickSort(d,c: integer);
var i, j, k, x: integer;
begin
  i := d; j := c;
  x := gdv[id[(i+j) div 2]]; {phantu giua}
  while i <= j do
    begin
      while gdv[id[i]] > x do inc(i);
      while gdv[id[j]] < x do dec(j);
      if i <= j then
        begin
          k := id[i];
          id[i] := id[j];
          id[j] := k;
          inc(i); dec(j);
        end;
      end;
    if d < j then IDQuickSort(d,j);
    if i < c then IDQuickSort(i,c);
  end;
procedure XepBaLo;
var i: integer;
begin
  tt := 0; {tong gia tri }
  t := m; {trong luong con lai cua balo }
  for i :=1 to n do
    if t >= a[id[i]] then
      begin { lay tron vat id[i] }
        t := t-a[id[i]];
        tt := tt + (a[id[i]]*gdv[id[i]]);
      end
    else { lay cho day balo }
      begin
        tt := tt+(t*gdv[id[i]]); {lay vua du }
        a[id[i]] := t; {chinh lai vat cuoi }
        t := 0;
      end;
  end;
end;
procedure Ghi;
var i: integer;
begin
  assign(g,gn); rewrite(g);
  for i := 1 to n do writeln(g,a[i]);
  writeln(g,tt); close(g);
end;
BEGIN
  Doc; InitID; IDQuickSort(1,n);
  XepBaLo; Ghi;
END.

// C#
using System;

```

```

using System.IO;
namespace SangTao1
{
    /*-----
    *           Xep BaLo
    * -----*/
    class BaLo
    {
        const string fn = "BaLo.inp";
        const string gn = "BaLo.out";
        static public Item[] Items;
        static public int[] t;
        static public int n = 0; // so luong vat
        static public int m = 0; // suc chua cua Ba lo
        static public int vh = 0; // Gia tri cua balo
        static void Main()
        {
            Doc(); QSort(0, n-1);
            Xep(); Ghi();
            Test();
            Console.WriteLine("\n Fini");
            Console.ReadLine();
        } // Main
        static public void Xep()
        {
            int th = m; // tr lg con lai cua balo
            vh = 0;
            t = new int[n];
            for (int i = 0; i < n; ++i) t[i] = 0;
            for (int i = 0; i < n; ++i)
            {
                int j = Items[i].Id;
                t[j] = Min(th, Items[i].Weight);
                th -= t[j];
                vh += t[j]*Items[i].Price;
                if (th == 0) break;
            }
        }
        static public int Min(int a, int b)
        { return (a < b) ? a : b; }
        static public void Ghi()
        {
            StreamWriter g = File.CreateText(gn);
            for (int i = 0; i < n; ++i)
                g.WriteLine(t[i]);
            g.WriteLine(vh); g.Close();
        }
        // Sap cac BaLo giam theo tien thuong
        static public void QSort(int d, int c)
        {
            int i = d, j = c;
            int m = Items[(d + c) / 2].Price;
            Item t = new Item(0,0,0);

```

```

        while (i <= j)
        {
            while (Items[i].Price > m) ++i;
            while (m > Items[j].Price) --j;
            if (i <= j)
            {
                t = Items[i];
                Items[i] = Items[j];
                Items[j] = t;
                ++i; --j;
            }
        }
        if (d < j) QSort(d, j);
        if (i < c) QSort(i, c);
    }
    // Doc lai file gn de kiem tra ket qua
    static public void Test() tự viết
    /*-----
    * Doc du lieu vao mang a
    * -----*/
    static public void Doc()
    {
        char[] cc = new char[]
        { '\n', ' ', '\t', '\0', '\r' };
        int[] b = Array.ConvertAll ((
            File.ReadAllText(fn)).
            Split(cc, StringSplitOptions.
                RemoveEmptyEntries),
            new Converter<string, int>(int.Parse));
        n = b[0]; // so luong vat
        m = b[1]; // gioi han trong luong balo
        // Tach du lieu
        Items = new Item[n];
        for (int k = 2, i = 0; i < n; ++i, k+=2)
            Items[i] = new Item(b[k], b[k+1], i);
    }
    public struct Item // mo ta mot mat hang
    {
        public int Weight; // trong luong
        public int Price; // don gia
        public int Id; // ma so
        public Item(int w, int p, int i)
        { Weight = w; Price = p; Id = i; }
    }
    } // BaLo
} // SangTao1

```

Bài 5.4. Cây bao trùm ngắn nhất

Cho một đồ thị liên thông G vô hướng bao gồm n đỉnh, mã số từ 1 đến n , và m cạnh nối hai đỉnh với nhau. Mỗi cạnh có chiều dài cho trước. Tìm liên thông của đồ thị cho biết với hai đỉnh cho trước tùy ý ta luôn tìm được các cạnh nối nhau để đi từ đỉnh này đến đỉnh kia. Hãy chỉ ra một phân P của đồ thị thỏa các tính chất sau:

- (i) P chứa tất cả các đỉnh của G ;
- (ii) P chứa một số ít nhất các cạnh của G ;
- (iii) P là đồ thị liên thông;
- (iv) Tổng chiều dài các cạnh của P là ngắn nhất.

Đồ thị P thỏa ba tính chất (i), (ii) và (iii) được gọi là *cây bao trùm* của đồ thị G . Nếu P thỏa thêm tính chất (iv) thì P được gọi là *cây bao trùm ngắn nhất* của G . Một số tài liệu dùng thuật ngữ *cây khung* thay cho cây bao trùm và *cây khung cực tiểu* thay cho cây bao trùm ngắn nhất.

Bài toán trên có nhiều ứng dụng thực tiễn. Một trong số ứng dụng đó được mô tả thông qua thí dụ sau:

Có n máy tính được nối với nhau thành mạng bằng cáp quang là một loại dây truyền tin đắt tiền. Trong mạng này, hai máy tính bất kì đều có thể liên lạc được với nhau trực tiếp hoặc thông qua một vài máy trung gian. Ta gọi tính chất này là tính liên thông của mạng máy tính. Hãy bỏ bớt một số dây nối để n máy tính trên vẫn liên thông được với nhau. Mạng tối thiểu thu được chính là một cây bao trùm ngắn nhất của mạng ban đầu.

Dữ liệu vào: tệp văn bản tên **DOTHI . INP**.

- Dòng đầu tiên ghi hai số tự nhiên n và m cách nhau qua dấu cách, biểu thị số đỉnh (n) và số cạnh (m) của đồ thị.
- Mỗi dòng thứ $i = 1, 2, \dots, m$ trong số m dòng tiếp theo ghi ba giá trị x y và d cách nhau qua dấu cách với ý nghĩa cạnh (x, y) của đồ thị có chiều dài d .

Dữ liệu ra: tệp văn bản tên **DOTHI . OUT** bao gồm:

- Danh sách các cạnh được chọn.
- Dòng cuối cùng ghi tổng chiều dài tìm được.

Thuật toán

Ta dùng thuật giải Kruskal với kĩ thuật như sau. Duyệt các cạnh từ chiều dài nhỏ đến lớn. Cạnh được chọn sẽ là cạnh không tạo thành chu trình khi ghép nó vào đồ thị kết quả.

DOTHI . INP	Ý nghĩa: Đồ thị có 8 đỉnh và 17 cạnh.	DOTHI . OUT	Ý nghĩa: Cây bao trùm ngắn nhất của đồ thị đã cho gồm 8 đỉnh và 7 cạnh
8 17		T	
1 2 8	Cạnh (1, 2) dài 8,	1 5	cho gồm 8 đỉnh và 7 cạnh
1 3 4	cạnh (1, 3) dài 4,	4 8	là (chiều dài mỗi cạnh
1 4 6	cạnh (1, 4) dài 6, ...,	7 8	được ghi sau dấu
1 5 1	cạnh (7, 8) dài 1 đơn vị.	2 3	hai chấm):
1 6 2		1 6	cạnh 1. (1, 5): 1
2 3 2		3 8	cạnh 2. (4, 8): 1
2 4 7		1 3	cạnh 3. (7, 8): 1
3 4 9		14	cạnh 4. (2, 3): 2
3 7 4			cạnh 5. (1, 6): 2
3 8 3			cạnh 6. (3, 8): 3
4 5 5			cạnh 7. (1, 3): 4
4 6 5			Tổng chiều dài 7 cạnh đã
4 8 1			chọn là: 14.
5 6 6			
6 7 8			
6 8 7			
7 8 1			

Lưu ý rằng đồ thị kết quả thu được ở các bước trung gian có thể không liên thông mà bao gồm nhiều mảnh liên thông (cây con). Loại đồ thị này được gọi là rừng. Kết quả cuối cùng sẽ là cây vì nó liên thông và được tạo thành từ $n - 1$ cạnh. Ta vận dụng tổ chức find-union cho các tập đỉnh rời nhau để quản lí các tập đỉnh được chọn nhằm phát hiện chu trình. Cạnh (x, y) khi được ghép vào đồ thị trung gian sẽ tạo thành chu trình khi và chỉ khi các đỉnh x và y cùng nằm trong một cây của đồ thị (rừng) trung gian đó. Như vậy mỗi cây con của đồ thị trung gian được quản lí như một tập con của tập các đỉnh $1..n$ của đồ thị ban đầu. Tập con này có phần tử đại diện chính là gốc của cây tương ứng. Phần tử này được chọn theo mã số nhỏ nhất. Các đỉnh còn lại của cây con đều trở về gốc đó.

Dễ thấy cây bao trùm luôn luôn có n đỉnh và $n - 1$ cạnh.

(* Pascal *)

```
(*-----
  DOTH1.PAS Cay bao trum ngan nhat
    (thuat giai Kruskal)
-----*)
program DoThi;
uses crt;
const
  MN = 100; bl = #32; {dau cach}
  fn = 'DoThi.inp'; gn = 'DoThi.out';
  nl = #13#10; {xuong dong}
type { Mo ta mot canh cua do thi }
  CANH = record
    x,y: integer; {canh (x,y) }
    len: integer; { do dai canh }
  end;
var
  a: array[0..MN] of CANH; { Mang cac canh }
  d: array[1..MN] of integer; {dung cho find-union }
  n: integer; {n: so dinh }
  m: integer; {so canh }
  f,g: text;
procedure Doc; (* Doc du lieu *)
var i: integer;
begin
  assign(f,fn); reset(f);
  read(f,n,m);
  for i := 1 to m do
    read(f,a[i].x,a[i].y,a[i].len);
  close(f);
end;
(* Sap canh tang theo len *)
procedure qsort(d,c: integer);
var
  i,j,m: integer;
  t: CANH;
begin
  i := d;
  j := c;
  m := a[(i+j) div 2].len; {diem giua}
  while (i<=j) do
```

```

begin
  while a[i].len < m do i := i+1;
  while a[j].len > m do j := j-1;
  if i<=j then
    begin
      t := a[i];
      a[i] := a[j];
      a[j] := t;
      i := i+1; j := j-1;
    end;
  end;
  if d < j then qsort(d,j);
  if i < c then qsort(i,c);
end;
{Tim dai dien cua tap chua x }
function Find(x: integer): integer;
begin
  while x <> d[x] do x := d[x];
  Find := x;
end;
{-----
Hop nhât 2 tap dinh: tap chua dinh x va tap chua
dinh y.
Union = false: Neu canh (x,y) tao thanh chu trinh.
Union = true: Neu (x,y) khong tao thanh chu trinh.
-----}
function Union(x,y: integer): Boolean;
begin
  Union := false;
  x := Find(x);
  y := Find(y);
  if x = y then exit;
  if x < y then d[y] := x
  else d[x] := y;
  Union := true;
end;
procedure CBTNN; (* Cay bao trum ngan nhât *)
var
  i, t: integer;
  k: integer;
begin
  assign(g,gn); rewrite(g);
  for i := 1 to n do d[i] := i; {Khoi tri }
  k := 0;
  t := 0; {tong chieu dai cac canh}
  for i := 1 to m do
    {duyet cac canh theo chieu dai tang dan }
    if Union(a[i].x,a[i].y) then
      begin
        writeln(g,a[i].x,bl,a[i].y);
        t := t + a[i].len;
        inc(k);
        if k=n-1 then break;{chon duoc n-1 canh la du }
      end
    end;
  end;

```

```

        end;
        writeln(g,t);
        close(g);
    end;
BEGIN
    Doc; qsort(1,m);
    CBTNN; readln;
END.

// C#
using System;
using System.IO;
namespace SangTao1
{
    /*-----
    *          Cay khung (cay bao trum)
    *          ngan nhât
    * -----*/
    class CayKhung
    {
        const string fn = "DoThi.inp";
        const string gn = "DoThi.out";
        // do thi co n dinh
        // m canh (u,v)
        // u la dinh dau, v - dinh cuoi
        // Len - chieu dai canh
        static int[] d ; // to chuc Find-Union
        static int n = 0; // so dinh cua do thi
        static int m = 0; // so canh cua do thi
        static Canh[] cc; // Tap cac canh
        static int [] t; // canh duoc chon
        static int k; // so canh duoc chon
        static int sum = 0;
        static void Main()
        {
            Doc(); QSort(0, m-1);
            Ghi(); Test();
            Console.WriteLine("Fini");
            Console.ReadLine();
        } // Main
        static void Xep()
        {
            d = new int[n+1];
            t = new int [n];
            k = 0; sum = 0;
            // Khoi tri cho Find-Union
            for (int i = 1; i <= n; ++i) d[i] = i;
            for (int i = 0; i < m; ++i)
                if (Union(cc[i].U,cc[i].V))
                    { t[k++] = i; sum += cc[i].Len; }
        }
        static void Ghi()
        {

```

```

        StreamWriter g = File.CreateText(gn);
        for (int i = 0; i < k; ++i)
            cc[t[i]].FWrite(g);
        g.WriteLine(sum); g.Close();
    }
    static int Find(int x)
    {
        while (d[x] != x) x = d[x];
        return x;
    }
    // Hop nhat 2 tap dinh
    // tap chua dinh u va tap chua dinh v
    static bool Union(int u, int v)
    {
        u = Find(u); v = Find(v);
        if (u == v) return false;
        if (u < v) d[v] = u; else d[u] = v;
        return true;
    }
    // Sap cac canh tang dan theo Len
    static void QSort(int d, int c)
    {
        int i = d, j = c;
        int m = cc[(d + c) / 2].Len;
        Canh t = new Canh(0,0,0);
        while (i <= j)
        {
            while (cc[i].Len < m) ++i;
            while (m < cc[j].Len) --j;
            if (i <= j)
            {
                t = cc[i]; cc[i] = cc[j];
                cc[j] = t;
                ++i; --j;
            }
        }
        if (d < j) QSort(d, j);
        if (i < c) QSort(i, c);
    }
    // Doc lai file gn de kiem tra ket qua
    static void Test() tự viết
    static void Doc()
    {
        int[] b =

        Array.ConvertAll((File.ReadAllText(fn)).Split(
        new char[] { '\n', ' ', '\t', '\0', '\r' },
        StringSplitOptions.RemoveEmptyEntries),
        new Converter<string, int>(int.Parse));
        n = b[0]; // so dinh
        m = b[1]; // so canh
        // Tach du lieu
        cc = new Canh[m];
    }

```

```

    for (int k = 2, i = 0; i < m; ++i, k += 3)
        cc[i] = new Canh(b[k], b[k + 1], b[k + 2]);
    } // Doc
    public struct Canh // Mo ta canh
    {
        public int U; // dinh dau
        public int V; // dinh cuoi u < v
        public int Len;
        public Canh(int d1, int d2, int d)
        {
            if (d1 < d2) { U = d1; V = d2; }
            else { U = d2; V = d1; }
            Len = d;
        }
        public void Print()
        {
            Console.WriteLine(U + " " + V
                               + " " + Len);
        }
        public void Print2()
        { Console.WriteLine(U + " " + V); }
        public void FWrite(StreamWriter f)
        { f.WriteLine(U + " " + V); }
    }
} // CayKhung
} // SangTao1

```

Bài 5.5. Trộn hai tệp

Cho hai tệp văn bản *data1.inp* và *data2.inp* chứa các số nguyên được sắp tăng. Viết chương trình trộn hai dãy dữ liệu trong hai tệp này thành một dãy dữ liệu sắp tăng duy nhất và ghi trong tệp văn bản *data.out*.

Chú ý:

- Với dữ liệu đã cho trong tệp thứ nhất là 5 số, tệp thứ hai là 6 số thì tệp kết quả sẽ chứa 11 số.
- Số lượng các số trong mỗi tệp tối đa là 50 nghìn và không biết trước.
- Các số có giá trị kiểu nguyên, được tách nhau bởi dấu cách và có thể nằm trên nhiều dòng.
- Khi trộn hai tệp nói trên ta phải thực hiện tối thiểu 22 lần đọc-ghi bao gồm 11 lần đọc và 11 lần ghi.

Thí dụ:

<i>data1.inp</i>	<i>data2.inp</i>	<i>data.out</i>
2	3	2
3	3	3
5	4	3
5	7	3
10	12	4
	20	5
		5
		7
		10
		12
		20

Thuật toán

Ta dùng phương pháp cân. Gọi hai tệp chứa dữ liệu cần trộn là f và g , tệp chứa kết quả trộn là h . Hãy tưởng tượng, ta dùng tay trái lấy lần lượt, mỗi lần một phần tử của tệp f (ghi vào biến t) và dùng tay phải lấy lần lượt mỗi lần một phần tử của tệp g (ghi vào biến p). So sánh vật nặng trên hai tay t và p . Tay nào cầm phần tử nhẹ hơn thì đặt phần tử đó vào tệp kết quả h và do tay đó rồi nên lấy tiếp phần tử từ tệp tương ứng. Quá trình này kết thúc khi nào một trong hai tệp f hoặc g được duyệt xong. Cuối cùng ta chuyển nốt các phần tử còn lại của tệp chưa duyệt hết (tệp f hoặc g) vào tệp kết quả h .

Ta cần lưu ý mấy điểm sau đây:

Khi đọc xong phần tử cuối cùng của một tệp thì tệp đó chuyển sang trạng thái kết thúc (**EOF**), do đó nếu ta tổ chức vòng lặp **WHILE** trong thủ tục trộn hai tệp theo điều kiện **(NOT EOF(f)) AND (NOT EOF(g))** thì phần tử cuối của các tệp đó sẽ chưa kịp được so sánh, trong khi ta muốn tôn trọng nguyên tắc: sau khi so sánh t và p thì một trong hai biến, t hoặc p phải được giải phóng. Có thể thực hiện nguyên tắc này bằng kỹ thuật săn đuôi như sau: dùng biến logic **ef** ghi nhận trạng thái hết tệp f sớm hơn một nhịp. Điều đó có nghĩa khi **ef=FALSE** biến t vẫn đang chứa giá trị chưa xử lý (chưa so sánh với p và do đó chưa được ghi vào tệp h). Chú ý rằng dù **ef = FALSE** nhưng có thể ta vẫn có **EOF(f)=TRUE**. Một biến **eg** tương tự cũng được tạo cho tệp g . Về bản chất có thể hiểu các biến **ef** và **eg** khi chúng nhận giá trị **TRUE** là thực sự đã đọc được 1 đơn vị dữ liệu từ file.

(* Pascal *)

```
(*-----
      Merge Files
-----*)
program MergeFiles;
uses crt;
const
  BL = #32; MN = 12;
  fn = 'data1.inp'; gn = 'data2.inp';
  hn = 'data.out';
{-----
  Tron tep fn va gn ghi vao hn
-----}
procedure FMerge;
var
  f,g,h: text;
  ef, eg: Boolean;
  t, p: integer;
  d: longint; {sophan tu trong tep out}
begin
  assign(f,fn); reset(f);
  assign(g,gn); reset(g);
  assign(h,hn); rewrite(h);
  ef := SeekEof(f);
  if NOT eof then read(f,t);
  eg := SeekEof(g);
  if NOT eg then read(g,p);
  d := 0;
  while (NOT ef) AND (NOT eg) do
    if t < p then
```

```

        begin
            inc(d);
            write(h,t,BL);
            if d mod 10 = 0 then writeln(h);
            ef := SeekEof(f);
            if NOT ef then read(f,t);
        end else
        begin
            inc(d);
            write(h,p,BL);
            if d mod 10 = 0 then writeln(h);
            eg := SeekEof(g);
            if NOT eg then read(g,p);
        end;
    while (NOT ef) do
    begin
        inc(d);
        write(h,t,BL);
        if d mod 10 = 0 then writeln(h);
        ef := SeekEof(f);
        if NOT ef then read(f,t);
    end;
    while (NOT eg) do
    begin
        inc(d);
        write(h,p,BL);
        if d mod 10 = 0 then writeln(p);
        eg := SeekEof(g);
        if NOT eg then read(g,p);
    end;
    close(f); close(g); close(h);
end;
BEGIN
    FMerge;
END.
// C#
using System;
using System.IO;
namespace SangTao1
{
    /*-----
    *           Tron 2 files sap tang
    * -----*/
    class TronTep
    {
        static string gn = "Data.out"; // file ket qua
        static string fn1 = "data1.inp"; // input 1
        static string fn2 = "data2.inp"; // input 2
        static void Main()
        {
            Merge();
            Test();
        }
    }
}

```

```

        Console.WriteLine("\n Fini");
        Console.ReadLine();
    }
    // true neu ki tu c co phai la chu so
    static public bool IsDigit(char c)
    { return (c >= '0' && c <= '9'); }
    // true neu c la dau + hoac -
    static public bool IsPlusMin(char c)
    { return (c == '+' || c == '-'); }
    // true neu c la chu so hoac dau +, -
    static public bool Legal(char c)
    { return IsDigit(c) || IsPlusMin(c); }
    // true neu c la dau trang, bao gom
    // dau cach, xuong dong, tab, het dong, cuoi string
    static public bool IsWhite(char c)
    { return (c==' '||c=='\n' ||c=='\t' ||c=='\r' ||c=='\0'); }
    // doc 1 so nguyen co dau
    static public bool ReadInt(StreamReader f, ref int s)
    {
        s = 0;
        int sign = 1;
        char c = ' ';
        while (IsWhite(c) && !f.EndOfStream)
            c=(char)f.Read();
        if (!Legal(c)) return false;
        if (IsPlusMin(c))
        {
            if (c == '-') sign = -1;
            if (f.EndOfStream) return false;
            c = (char)f.Read();
            while (IsWhite(c) && !f.EndOfStream)
                c = (char)f.Read();
            if (!IsDigit(c)) return false;
        }
        while (IsDigit(c))
        {
            s = s * 10 + (int)(c - '0');
            if (f.EndOfStream) break;
            c = (char)f.Read();
        }
        s *= sign;
        return true;
    }
    static void Merge()
    {
        StreamWriter g = File.CreateText(gn);
        StreamReader f1 = File.OpenText(fn1);
        StreamReader f2 = File.OpenText(fn2);
        int x=0,y=0;
        bool b1 = ReadInt(f1, ref x);
        bool b2 = ReadInt(f2, ref y);
        while (b1 && b2)
        {

```

```

        if (x <= y)
        {
            g.WriteLine(x);
            b1 = ReadInt(f1, ref x);
        }
        else
        {
            g.WriteLine(y);
            b2 = ReadInt(f2, ref y);
        }
    }
    while (b1){g.WriteLine(x);b1=ReadInt(f1,ref x);}
    while (b2){g.WriteLine(y);b2=ReadInt(f2,ref y);}
    f1.Close(); f2.Close(); g.Close();
}
static void Test()
{
    Console.WriteLine("Inp1:\n"+File.ReadAllText(fn1));
    Console.WriteLine("Inp2:\n"+File.ReadAllText(fn2));
    Console.WriteLine("Out:\n"+File.ReadAllText(gn));
}
} // ChonTep
} // SangTao1

```

Chú thích Thực ra có thể đọc dữ liệu từ hai file vào hai mảng tương ứng rồi trộn hai mảng này. Tuy nhiên chúng ta muốn minh họa lời giải với hạn chế là mỗi lần chỉ được phép đọc một đơn vị dữ liệu từ mỗi file.

Hàm bool ReadInt(StreamReader f , ref int i) đọc mỗi lần một số nguyên i từ file text f . Nếu đọc được, hàm cho ra giá trị true và số i , ngược lại, nếu không gặp số nguyên, hàm cho ra giá trị false. Hàm hoạt động như sau. Trước hết bỏ qua các kí tự trắng. Nếu gặp dấu trừ ‘-’ hàm ghi nhận dấu đó, sau đó hàm đọc tiếp dãy số và tích lũy dần vào biến nguyên i .

Bài 5.6. Trộn nhiều tệp

Cho n tệp văn bản mã số từ 1 đến n . Tệp thứ i chứa d_i phần tử được sắp tăng. Hãy lập một lịch chỉ ra trình tự trộn mỗi lần hai tệp để cuối cùng thu được một tệp sắp tăng duy nhất với tổng số lần ghi dữ liệu vào tệp là nhỏ nhất. Biết rằng thủ tục trộn hai tệp chỉ có thể đọc tuần tự hoặc ghi tuần tự mỗi lần một phần tử.

Dữ liệu vào: Tệp văn bản MF.INP.

- Dòng đầu tiên là số lượng n các tệp chứa dữ liệu sắp tăng.
- Tiếp đến là n số tự nhiên $d_i, i = 1..n$ cho biết số phần tử trong tệp thứ i . Mỗi số ghi trên một dòng.

Dữ liệu ra: Tệp văn bản MF.OUT.

- Dòng đầu tiên: m là số lần thực hiện trộn hai tệp.
- Tiếp đến là m dòng, mỗi dòng chứa ba số tự nhiên i, j và k cho biết cần lấy tệp i trộn với tệp j và ghi kết quả vào tệp k . Các số trên cùng một dòng cách nhau qua dấu cách.

Tập chứa kết quả trung gian phải có mã số khác với mã số của các tập tạo lập trước đó.

Thí dụ:

MF . INP MF . OUT

5	4	<i>Ý nghĩa: Cho 5 tập sắp tăng với số phần tử lần lượt là 10, 5, 4, 4, 3. Cần thực hiện 4 lần trộn, mỗi lần 2 tập. Lần thứ nhất: trộn tập 5 với tập 3 ghi vào tập 6. Lần thứ hai: trộn tập 4 với tập 2 ghi vào tập 7. Lần thứ ba: trộn tập 6 với tập 7 ghi vào tập 8. Lần thứ tư: trộn tập 1 với tập 8 ghi vào tập 9. Tổng số lần ghi là 58.</i>
10	5 3 6	
5	4 2 7	
4	6 7 8	
4	1 8 9	
3	58	

Thuật toán

Trước hết đề ý rằng nếu trộn tập sắp tăng f gồm n phần tử với tập sắp tăng g gồm m phần tử để thu được tập sắp tăng h thì đối với các phần tử trong hai tập nguồn ta chỉ cần thực hiện thao tác đọc, còn thao tác ghi chỉ thực hiện đối với tập đích h . Kí hiệu $|f|$ là số phần tử trong tập f , ta có:

$$|f| = n, |g| = m$$

Do tổng số các phần tử của hai tập là $m + n$ nên số phần tử trong tập đích h sẽ là

$$|h| = n + m = |f| + |g|$$

và do đó số lần ghi (tối thiểu) các phần tử vào tập h sẽ là $n + m$.

Ta có nhận xét sau: Muốn xây dựng một quy trình trộn mỗi lần hai tập cho nhiều tập ban đầu với yêu cầu tổng số thao tác ghi tập là tối thiểu thì ta phải tạo ra các tập trung gian càng ít phần tử càng tốt.

Ta dùng kí hiệu $f \oplus g \rightarrow h$ với ý nghĩa là trộn hai tập nguồn f và g để thu được tập h . Ta có

$$\text{Nếu } f \oplus g \rightarrow h \text{ thì } |h| = |f| + |g|$$

Đề ý rằng trộn tập f với tập g hay trộn tập g với tập f thì số thao tác ghi tập như nhau và cùng bằng $|f| + |g|$. Giả sử ta có ba tập với số phần tử tương ứng là

$$s[1..3] = (5, 1, 2).$$

Giả sử ta thực hiện quy trình $(\textcircled{1} \oplus \textcircled{2}) \oplus \textcircled{3}$ như sau:

Bước 1: Trộn tập $\textcircled{1}$ với tập $\textcircled{2}$ ghi tạm vào tập $\textcircled{4}$. Số thao tác ghi sẽ là $(5 + 1) = 6$ và tập $\textcircled{4}$ có 6 phần tử.

Bước 2: Trộn tập $\textcircled{4}$ với tập $\textcircled{3}$ ghi vào tập $\textcircled{5}$. Số thao tác ghi sẽ là $6 + 2 = 8$ và tập $\textcircled{5}$ có 8 phần tử.

Kết quả thu được tập $\textcircled{5}$. Tổng số thao tác ghi trong cả hai bước trên sẽ là:

$$6 + 8 = 14.$$

Tổng quát, với ba tập a , b và c được trộn theo quy trình:

$$(a \oplus b) \oplus c$$

ta dễ dàng tính được tổng số thao tác ghi tập cho quy trình trên là

$$(|a| + |b|) + (|a| + |b|) + c = 2(|a| + |b|) + c.$$

Bảng dưới đây tính toán cho ba phương án để phát hiện ra phương án tối ưu.

Phương án	Quy trình thực hiện	Tổng số thao tác ghi tệp
1	$(① \oplus ②) \oplus ③$	$2(5 + 1) + 2 = 2.6 + 2 = 14$
2	$(① \oplus ③) \oplus ②$	$2(5 + 2) + 1 = 2.7 + 1 = 15$
3	$(② \oplus ③) \oplus ①$	$2(1 + 2) + 5 = 2.3 + 5 = 11$ (phương án tối ưu)

Khảo sát các quy trình trộn ba tệp

$$s[1..3] = (5, 1, 2)$$

Thuật toán tham lam khi đó sẽ như sau:

Thuật toán Huffman

Lặp (đến khi chỉ còn một tệp duy nhất)

- Lấy hai tệp u và v có số phần tử nhỏ nhất.
- Trộn $u \oplus v \rightarrow h$. Ta có $|h| = |u| + |v|$.
- Loại bỏ u và v khỏi danh sách các tệp cần xử lí.
- Kết nạp h vào danh sách các tệp cần xử lí

xong lặp

Với n tệp ban đầu, dễ thấy rằng mỗi lần lặp ta loại bỏ được hai tệp (u và v có số phần tử min) và thêm một tệp (h) tức là mỗi lần lặp ta loại bỏ được một tệp, do đó số lần lặp sẽ là $n - 1$.

Thuật toán trên mang tên nhà toán học Mĩ Huffman là người đầu tiên đề xuất.

Ta minh hoạ thuật toán trên với dữ liệu vào như sau:

$$s[1..5] = (10, 5, 4, 4, 3).$$

Ý nghĩa: Trộn 5 tệp sắp tăng với số phần tử lần lượt là 10, 5, 4, 4 và 3 để thu được một tệp sắp tăng duy nhất.

Lần lặp	Danh sách các tệp cần xử lí	Hai tệp có số phần tử min	Trộn	Số thao tác ghi tệp
1	(①:10, ②:5, ③:4, ④:4, ⑤:3)	⑤:3 , ③:4	⑤ \oplus ③ \rightarrow ⑥	7
2	(①:10, ②:5, ④:4, ⑥:7)	②:5 , ④:4	② \oplus ④ \rightarrow ⑦	9
3	(①:10, ⑥:7, ⑦:9)	⑥:7 , ⑦:9	⑥ \oplus ⑦ \rightarrow ⑧	16
4	(①:10, ⑧:16)	①:10 , ⑧:16	① \oplus ⑧ \rightarrow ⑨	26
Kết quả	(⑨:26)			58

Minh hoạ thuật toán Huffman với dữ liệu vào

$$(①:10, ②:5, ③:4, ④:4, ⑤:3)$$

Vì $n = 5$ nên số lần lặp sẽ là $n - 1 = 4$. Sau 4 lần lặp ta thu được tệp mã số 9 với 26 phần tử. Để tính tổng số thao tác ghi ta chỉ cần lấy tổng số phần tử của các tệp tham gia trong mỗi lần trộn hai tệp. Tổng đó là:

$$tt = (3 + 4) + (5 + 4) + (7 + 9) + (10 + 16) = 7 + 9 + 16 + 26 = 58.$$

Ta chọn phương án cài đặt sau đây cho thuật toán Huffman. Phương án này tỏ ra tiện lợi trong nhiều ứng dụng. Lợi thế của nó là không xoá đi các đối tượng (tập) đã xử lí mà chỉ đánh dấu chúng để khi cần có thể khôi phục lại và giải trình kết quả.

Cụ thể là ta sẽ xây dựng một cây nhị phân gồm $2n - 1$ đỉnh và gọi là cây Huffman như sau.

Các đỉnh được mã số từ $1..2n - 1$. Mỗi đỉnh nhận một giá trị nguyên dương gọi là trọng số của đỉnh đó.

Trên hình vẽ, đỉnh được thể hiện trong hình tròn, cạnh đó là giá trị của trọng số. Trong bài toán trộn tệp này mã số của đỉnh chính là mã số của tệp, trọng số của đỉnh chính là số phần tử có trong tệp tương ứng.

Thuật toán tạo cây Huffman

Khởi tạo: n đỉnh rời nhau $1..n$ có trọng số $s(1), \dots, s(n)$.

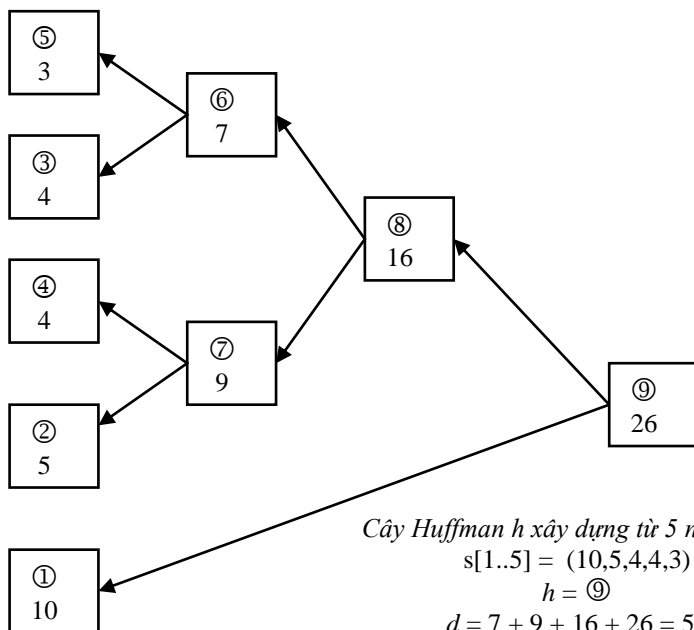
$h := n$;

Lặp $n - 1$ lần

- Lấy hai đỉnh u và v có $s(u)$ và $s(v)$ min.
- Đánh dấu u và v là đã xử lí.
- $h := h + 1$
- Tạo đỉnh mới h trở đến u và v và $s(h) = s(u) + s(v)$.

xong lặp

Để tổ chức dữ liệu cho cây Huffman chúng ta dùng ba mảng nguyên s , t và p kích thước $2n - 1$ phần tử. Với mỗi đỉnh i , $s[i]$ cho biết trọng số của đỉnh i , $t[i]$ trỏ đến con trái của đỉnh i , $p[i]$ trỏ đến con phải của đỉnh i . Hai con trái $t[i]$ và phải $p[i]$ chính là hai đỉnh đạt trọng số min trong mỗi lần lặp, h chính là đỉnh mới được tạo lập từ hai đỉnh có trọng số min. Ngoài ra ta dùng một mảng nguyên d để đánh dấu các đỉnh đã xử lí, $d[i] = 0$ cho biết đỉnh i chưa được xử lí, $d[i] = 1$ cho biết đỉnh i đã xử lí. Các đỉnh mới được tạo lập và thêm vào cây lần lượt nhận mã số là $n + 1, n + 2, \dots, 2n - 1$, do đó đỉnh cuối cùng sẽ có mã số là $h = 2n - 1$. Thủ tục tạo cây Huffman h khi đó sẽ như sau:



```

{-----
    Tao cay Huffman h = 2n-1
    tu cac trong so s[1..n]
-----}

procedure Huffman;
var i,u,v: integer;
begin
    fillchar(d,sizeof(d),0);
    fillchar(t,sizeof(t),0);
    fillchar(p,sizeof(p),0);
    h := n; tt := 0; {tong trong so}
    for i := 1 to n-1 do
    begin
        min2(u,v); {u,v dat trong so min }
        h := h+1; {ma so cua dinh moi}
        s[h] := s[u]+s[v]; {trong so cua dinh moi }
        tt := tt+s[h]; {tong trog so }
        t[h] := u; {tro toi con trai }
        p[h] := v; {tro toi con phai }
    end;
end;

```

Thủ tục `min2(u,v)` tìm trong số các đỉnh chưa xử lý hai đỉnh u và v đạt trọng số min. Thủ tục này gọi hai lần hàm `min1`, mỗi lần tìm một đỉnh đạt trọng số min trong số các đỉnh chưa xử lý và đánh dấu luôn đỉnh tìm được (là đã xử lý).

```

{-----
    Tim trong so cac dinh chua xu li
    hai dinh u va v dat trong so min.
-----}

procedure min2(var u,v: integer);
begin
    u := min1; v := min1;
end;

{-----
    Tim trong so cac dinh chua xu li
    mot dinh dat trong so min
    va danh dau dinh tim duoc.
-----}

function min1: integer;
var i, imin, vmin: integer;
begin
    vmin := MaxInt;

```

```

for i := 1 to h do
  if d[i]=0 then {dinh i chua xu li }
    if s[i] < vmin then
      begin
        vmin := s[i]; imin := i;
      end;
  d[imin] := 1; min1 := imin;
end;

```

Sau khi tạo xong cây Huffman, để ghi kết quả, ta chỉ cần duyệt các đỉnh được tạo mới, tức là các đỉnh có mã số từ $n + 1$ đến $h = 2n - 1$ để lấy hai con trái và phải của mỗi đỉnh.

```

{-----
Duyet cac dinh tu n+1    den 2n-1,
ghi thong tin vao tep.
-----}

procedure Ghi;
var i: integer;
begin
  assign(g,gn) ;
  rewrite(g) ;
  writeln(g,n-1) ;
  for i := n+1 to h do
    writeln(g,t[i],BL,p[i],BL,i) ;
  writeln(g,tt) ;
  close(g) ;
end;

```

(* Pascal *)

```

(*-----
Tron nhieu tep
-----*)

uses crt;
const
  fn = 'MF.INP' ;
  gn = 'MF.OUT' ;
  MN = 200;
  BL = #32; {Dau cach}
  NL = #13#10; {xuong dong}
type
  MI1 = array[0..MN] of integer;
  MB1 = array[0..MN] of byte;
var
  s,t,p: MI1;
  {s[i] - so phan tu trong tep i}
  {t[i] - tro trai}
  {p[i] - tro phai i}

```

```

d: MB1; {danh dau tep da xu li}
n: integer; {so luong tep ban dau}
h: integer; {cay Huffman}
f,g: text;
tt: longint;
procedure Doc;
var i: integer;
begin
  assign(f,fn); reset(f); read(f,n);
  for i := 1 to n do read(f,s[i]);
  close(f);
end;
{-----}
Tim trong so cac dinh chua xu li
mot dinh dat trong so min
va danh dau dinh tim duoc.
-----}
function min1: integer; tự viết
{-----}
Tim trong so cac dinh chua xu li
hai dinh u va v dat
trong so min,  $U < v$ .
-----}
procedure min2(var u,v: integer); tự viết
{-----}
Tao cay Huffman h =  $2n-1$ 
tu cac trong so s[1..n]
-----}
procedure Huffman; tự viết
{-----}
Duyet cac dinh tu n+1 den  $2n-1$ ,
ghi thong tin vao tep.
-----}
procedure Ghi; tự viết
BEGIN
  Doc; Huffman; Ghi;
END.

```

// C#

```

using System;
using System.IO;
namespace SangTao1
{
  /*-----
  *      Cay Huffman
  *      Tron n file sap tang
  * -----*/

```

```

class HuffmanTree
{
    static string fn = "MF.inp"; // file ket qua
    static string gn = "MF.out"; // file ket qua

    static int[] t; // tro trai
    static int[] p; // tro phai
    static int[] v; // trong so dinh
    static int[] d; // danh dau dinh da xu ly
    static int n = 0; // so phan tu
    static int n2; // n2 = 2*n
    static int h = 0; // Goc cua cay Huffman
    static int tt = 0; // tong trong so
    static void Main()
    {
        Doc(); Huffman(); Ghi(); Test();
        Console.WriteLine("\n Fini");
        Console.ReadLine();
    } // Main
    static void Ghi()
    {
        StreamWriter f = File.CreateText(gn);
        for (int i = n + 1; i <= h; ++i)
            f.WriteLine(t[i] + " " + p[i] + " " + i);
        f.WriteLine(tt);
        f.Close();
    }
    static void Huffman()
    {
        h = n; // goc cay Huffman
        tt = 0; // tong trong so
        int m1 = 0, m2 = 0;
        int x;
        for (int i = 1; i < n; ++i)
        {
            m1 = MinV(); m2 = MinV();
            if (m1 > m2)
                {x = m1; m1 = m2; m2 = x;}
            // m1 < m2
            ++h; // them dinh moi
            v[h] = v[m1] + v[m2];
            t[h] = m1; // tro trai
            p[h] = m2; // tro phai
            tt += v[h];
        }
    }
    // Tim dinh chua xu ly co trong so min
    static int MinV()
    {
        int imin = 0;
        for (int i = 1; i <= h; ++i)
            if (d[i] == 0) // dinh i chua x li
                if (v[i] < v[imin]) imin = i;
    }
}

```

```

        d[imin] = 1; // danh dau dinh i
        return imin;
    }
    static void Doc()
    {
        char[] cc = new char[] { '\n', ' ', '\t', '\0', '\r' };
        int [] a =
        Array.ConvertAll((File.ReadAllText(fn)).Split(cc,
            StringSplitOptions.RemoveEmptyEntries),
            new Converter<string,int>(int.Parse));
        n = a[0]; n2 = 2*n;
        v = new int[n2];
        t = new int[n2];
        p = new int[n2];
        d = new int[n2];
        v[0] = int.MaxValue; // linh canh
        // Khoi tri cac nut cua cay
        for (int i = 0; i < n2; ++i)
            t[i] = p[i] = d[i] = 0;
        for (int i = 1; i <= n; ++i)
            v[i] = a[i];
    }
    static void Print(int[] a, int n) tự viết
    static void Test() tự viết
    } // Huffmantree
} // SangTao1

```

Chú ý

Thuật ngữ *tham lam* không có nghĩa là lấy nhiều nhất mà chỉ là xác định một chiến lược xử lý dữ liệu sao cho có hiệu quả nhất.

CHƯƠNG 6

PHƯƠNG PHÁP QUAY LUI

Giả sử ta phải tìm trong một tập dữ liệu D cho trước một dãy dữ liệu:

$$v = (v[1], v[2], \dots, v[n])$$

thoả mãn đồng thời hai tính chất P và Q . Trước hết ta chọn một trong hai tính chất đã cho để làm nền, giả sử ta chọn tính chất P .

Sau đó ta thực hiện các bước sau đây:

Bước 1. (Khởi trị) Xuất phát từ một dãy ban đầu $v = (v[1], \dots, v[i])$ nào đó của các phần tử trong D sao cho v thoả P .

Bước 2. Nếu v thoả Q ta dừng thuật toán và thông báo kết quả là dãy v , ngược lại ta thực hiện Bước 3.

Bước 3. Tìm tiếp một phần tử $v[i+1]$ để bổ sung cho v sao cho

$$v = (v[1], \dots, v[i], v[i+1]) \text{ thoả } P.$$

Có thể xảy ra các trường hợp sau đây:

3.1. Tìm được phần tử $v[i+1]$: quay lại bước 2.

3.2. Không tìm được $v[i+1]$ như vậy, tức là với mọi $v[i+1]$ có thể lấy trong D , dãy $v = (v[1], \dots, v[i], v[i+1])$ không thoả P . Điều này có nghĩa là đi theo đường

$$v = (v[1], \dots, v[i])$$

sẽ không dẫn tới kết quả. Ta phải đổi hướng tại một vị trí nào đó. Để thoát khỏi ngõ cụt này, ta tìm cách thay $v[i]$ bằng một giá trị khác trong D . Nói cách khác, ta loại $v[i]$ khỏi dãy v , giảm i đi một đơn vị rồi quay lại Bước 3.

Cách làm như trên được gọi là quay lui: lùi lại một bước.

Dĩ nhiên ta phải đánh dấu $v[i]$ là phần tử đã loại tại vị trí i để sau đó không đặt lại phần tử đó vào vị trí i trong dãy v .

Khi nào thì có thể trả lời là không tồn tại dãy v thoả đồng thời hai tính chất P và Q ? Nói cách khác, khi nào thì ta có thể trả lời là bài toán vô nghiệm?

Dễ thấy, bài toán vô nghiệm khi ta đã duyệt hết mọi khả năng. Ta nói là đã vét cạn mọi khả năng. Chú ý rằng có thể đến một lúc nào đó ta phải lùi liên tiếp nhiều lần. Từ đó suy ra rằng, thông thường bài toán vô nghiệm khi ta không còn có thể lùi được nữa. Có nhiều sơ đồ giải các bài toán quay lui, dưới đây là hai sơ đồ khá đơn giản, không đệ quy.

Sơ đồ 1: Giải bài toán quay lui (tìm 1 nghiệm)

```
Khởi trị  $v$ :  $v$  thoả  $P$ ;  
repeat  
  if ( $v$  thoả  $Q$ ) then  
    begin  
      Ghi nhận nghiệm;  
    exit;
```

Sơ đồ 2: Giải bài toán quay lui (tìm 1 nghiệm)

```
Khởi trị  $v$ :  $v$  thoả  $P$ ;  
repeat  
  if ( $v$  thoả  $Q$ ) then  
    begin  
      Ghi nhận nghiệm;  
    exit;
```

```

end;
if (Tìm được 1 nước đi)
then Tiến
else
  if (có thể lùi được)
  then Lùi
  else
    begin
      Ghi nhận: vô nghiệm;
      exit;
    end;
until false;

```

```

end;
if (Hết khả năng duyệt)
then
  begin
    Ghi nhận vô nghiệm;
    exit;
  end;
if (Tìm được 1 nước đi)
then Tiến
else Lùi;
until false;

```

Thông thường ta khởi trị cho v là dãy rỗng (không chứa phần tử nào) hoặc dãy có một phần tử. Ta chỉ yêu cầu dãy v được khởi trị sao cho v thoả P . Lưu ý là cả dãy v thoả P chứ không phải từng phần tử trong v thoả P .

Có bài toán yêu cầu tìm toàn bộ (mọi nghiệm) các dãy v thoả đồng thời hai tính chất P và Q . Nếu biết cách tìm một nghiệm ta dễ dàng suy ra cách tìm mọi nghiệm như sau: mỗi khi tìm được một nghiệm, ta thông báo nghiệm đó trên màn hình hoặc ghi vào một tệp rồi thực hiện thao tác Lùi, tức là giả vờ như không công nhận nghiệm đó, do đó phải loại $v[i]$ cuối cùng trong dãy v để tiếp tục tìm hướng khác. Phương pháp này có tên là phương pháp giả sai. Hai sơ đồ trên sẽ được sửa một chút như sau để tìm mọi nghiệm.

Sơ đồ 3: Giải bài toán quay lui (tìm mọi nghiệm)

```

Khởi trị:  $v$  thoả  $P$ ;
 $d := 0$ ; {đếm số nghiệm}
repeat
  if ( $v$  thoả  $Q$ ) then
    begin
       $d := d+1$ ;
      Ghi nhận nghiệm thứ  $d$ ;
      Lùi; { giả sai }
    end;
  if (Tìm được 1 nước đi)
  then Tiến
  else if (có thể lùi được)
  then Lùi
else { hết khả năng }
begin
  if  $d = 0$  then
    Ghi nhận: vô nghiệm;
  else
    Ghi nhận:  $d$  nghiệm;
  exit;
end;
until false;

```

Sơ đồ 4: Giải bài toán quay lui (tìm mọi nghiệm)

```

Khởi trị:  $v$  thoả  $P$ ;
 $d := 0$ ; {đếm số nghiệm}
repeat
  if ( $v$  thoả  $Q$ ) then
    begin
       $d := d+1$ ;
      Ghi nhận nghiệm thứ  $d$ ;
      Lùi; { giả sai }
    end;
  if (Hết khả năng duyệt)
  then
    begin
      if  $d = 0$  then
        Ghi nhận: vô nghiệm;
      else
        Ghi nhận:  $d$  nghiệm;
      exit;
    end;
  if (Tìm được 1 nước đi)
  then Tiến
  else Lùi;
until false;

```

Bài 6.1. Các quân Hậu

Quân Hậu trên bàn cờ Vua có thể ăn theo hàng, theo cột chứa nó hoặc theo đường chéo của hình vuông nhận nó làm đỉnh.

a) Tìm một cách đặt N quân Hậu trên bàn cờ Vua kích thước $N \times N$ ô sao cho không quân nào ăn được quân nào.

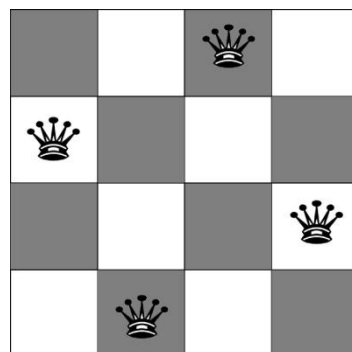
b) Tìm mọi cách đặt N quân Hậu theo điều kiện trên. Ghi kết quả vào một tệp văn bản tên $N_HAU.OUT$.

Thuật toán

Trước hết ta đặt các quân Hậu ở mép ngoài bàn cờ. Hậu thứ i sẽ đứng ở đầu cột thứ i . Sau đó ta dịch dần các Hậu vào trong các dòng của bàn cờ và ghi nhận vị trí của chúng vào một mảng v . Phần tử $v[i]$ của mảng v cho biết phải đặt Hậu thứ i , tức là Hậu chiếm cột i tại dòng $v[i]$.

Thí dụ, với bàn cờ 4×4 ta có lời giải $v = (2, 4, 1, 3)$ với ý nghĩa:

- Đặt Hậu thứ nhất tại (cột 1) dòng 2, Hậu thứ 2 tại (cột 2) dòng 4, Hậu thứ 3 tại (cột 3) dòng 1 và Hậu thứ 4 tại (cột 4) dòng 3.
- Mỗi khi đặt được Hậu thứ i ta chuyển qua Hậu tiếp theo $i + 1$. Điều kiện đặt được Hậu i trên dòng d của bàn cờ là nó không bị các Hậu đã đặt trước đó, tức là các Hậu $j = 1..(i - 1)$ chiếu. Đây chính là tính chất P.
- Hậu $j < i$ chiếu (đụng độ) Hậu i khi và chỉ khi $v[j] = v[i]$ (cùng hàng) hoặc $i - j = abs(v[i] - v[j])$ (Hậu i và Hậu j nằm trên hai đỉnh đối diện của hình vuông, do đó hai cạnh liên tiếp của hình vuông này phải bằng nhau).
- Tính chất Q khi đó sẽ là: đặt được đủ N Hậu.



Sơ đồ tìm một nghiệm XepHau1 như sau:

```
(*-----
      Tim 1 nghiệm: xep M quan hau tren
      ban co M X M
-----*)
procedure XepHau1(M: byte);
var i: byte;
begin
  if (M < 1) or (M > MN) then exit;
  {MN = 20 la gioi han kích thước bàn cờ}
  n := M;
  {Khởi trị: Đặt cốc hậu 1..N ngoài bàn cờ.
   Hậu i Đặt tại đầu cột i, i=1..N.}
  for i := 1 to n do v[i] := 0;
  i := 1; {Hậu đang xét}
  repeat
    if i > n then {có nghiệm v[1..n]}
    begin
      KetQual(n);
      exit;
    end;
    if i < 1 then {vô nghiệm}
    begin
      KetQual(0);
      exit;
    end;
  until true;
end;
```

```

        end;
    if Tim(i) {có cách đi }
    then inc(i) {Tiền}
    else
        begin {Lùi}
            v[i] := 0;
            dec(i);
        end;
    until false;
end;

```

Thủ tục có hai tình huống, KetQual (n) : hiển thị mảng $v[1..n]$, trong đó $v[i]$ là dòng đặt Hậu i , KetQual (0) : thông báo vô nghiệm.

Hàm Tim(i) thực hiện chức năng sau đây: xuất phát từ dòng Hậu i đang đứng là $v[i]$ đẩy tiếp Hậu i xuống các dòng dưới để tìm được một dòng đặt nó sao cho không bị các Hậu đặt trước đó, tức là không bị các Hậu $j = 1..(i-1)$ ăn.

Tim(i)=true: tìm được một vị trí (dòng) đặt Hậu i , ngược lại Tim=false.

```

(*-----
  Xuất phát từ dòng v[i]+1, tìm dòng mới
  có thể đặt được Hậu i
-----*)
function Tim(i: byte): Boolean;
begin
    Tim := true;
    while v[i] < n do
        begin
            inc(v[i]);
            if DatDuoc(i) then exit;
        end;
    Tim := false;
end;

```

Hàm Boolean DatDuoc(i) cho giá trị true nếu Hậu i không bị các Hậu $j = 1, 2, \dots, i-1$ đã đặt trước đó ăn. Ngược lại, nếu Hậu i bị một Hậu nào đó ăn thì hàm cho ra giá trị false.

```

(*-----
  Kiểm tra xem có đặt được Hậu i
  tại o (v[i],i) của bàn có không ?
-----*)
function DatDuoc(i: byte): Boolean;
var j: byte;
begin
    DatDuoc := false;
    for j := 1 to i-1 do
        if (v[i] = v[j]) or (i-j = abs(v[i]-v[j]))
            {Hậu j ăn được Hậu i}
        then exit;
    DatDuoc := true;
end;

```

Thao tác Tiến đơn giản là chuyển qua xét Hậu kế tiếp, Hậu $i+1$.

```

Tien: Chuyển qua Hậu tiếp theo
      inc(i);

```

Thao tác Lùi đưa Hậu ra ngoài bàn cờ, chuyển qua xét Hậu trước đó, Hậu $i - 1$.

Lui: Đưa Hậu ra ngoài bàn cờ, chuyển qua Hậu trước đó

`v[i] := 0; dec(i);`

Ta viết thủ tục XepHau để tìm mọi nghiệm của bài toán. Với bàn cờ 8×8 ta thu được 92 nghiệm. Với bàn cờ 10×10 ta thu được 724 nghiệm.

```
(*-----*
Tim moi cach dat M Hau tren ban co
                                     M X M
-----*)
procedure XepHau(M: byte);
var
    i: byte;
    d: integer; {dem so nghiem}
begin
    if (M < 1) or (M > MN) then exit;
    n := m;
    for i := 1 to n do v[i] := 0;
    assign(g,gn);
    rewrite(g);
    i := 1; {Hau dang xet}
    d := 0; {dem so nghiem}
    repeat
        if i > n then {Tim duoc 1 nghiem}
            begin
                inc(d);
                KetQua(d); {v[1..n] la nghiem thu d}
                i := n; {gia sai}
            end;
        if i < 1 then {Tim het cac nghiem}
            begin
                writeln(g,'Tong cong ',d,' nghiem ');
                close(g);
                writeln('Xem ket qua trong file ',gn);
                readln;
                exit;
            end;
        if Tim(i) then inc(i)
        else begin
            v[i] := 0;
            dec(i);
        end;
    until false;
end;

(* Pascal *)
(*=====
                                     N Hau
=====*)
{$B-}
uses crt;
const
    MN = 20;
```

```

gn = 'N_HAU.OUT';
BL = #32; {dau cach}
var
v: array[0..MN] of byte;
n: byte; {so quan hau, kích thước bàn cờ}
g: text; {tệp kết quả}
function DatDuoc(i: byte): Boolean; tự viết
function Tim(i: byte): Boolean; tự viết
(*-----*)
      Hien thi nghiem tren man hinh
      Cho bai toan tim 1 nghiem
      k=0: vo nghiem
      k=n: co nghiem v[1..n]
-----*)
procedure KetQual(k: byte);
var i: byte;
begin
  writeln;
  if k = 0 then write('Vo nghiem')
  else
    for i := 1 to k do write(v[i]:3);
  writeln;
end;
(*-----*)
Tim 1 nghiem: xep M quan hau tren
                ban co M X M
-----*)
procedure XepHau(M: byte); tự viết
(*-----*)
Ghi nghiem thu d vào tệp g 'N_Hau.out'
      Bai toan tim moi nghiem
-----*)
procedure KetQua(d: integer);
var i: byte;
begin
  write(g, 'Nghiem thu ', d, ': ');
  for i := 1 to n do write(g, v[i], BL);
  writeln(g);
end;
(*-----*)
Tim moi cach dat M Hau tren ban co M X M
-----*)
procedure XepHau(M: byte); tự viết
BEGIN
  XepHau(8); {tim 1 nghiem}
  XepHau(8); {tim du 92 nghiem}

END.

```

Phương án cải tiến

Ta xét một phương án cải tiến tập trung vào việc nâng cao tốc độ tính toán khi kiểm tra hai hậu dụng độ nhau. Mỗi khi tìm vị trí đặt hậu thứ i trên bàn cờ ta cần kiểm

tra xem hậu i đó có đụng độ với tất cả $(i-1)$ hậu đặt trước đó không. Thời gian chi phí tập trung ở chính điểm này.

Để cải tiến, ta sẽ sử dụng thêm 3 mảng đánh dấu các dòng và các đường chéo của các hậu đã đặt trên bàn cờ với ý nghĩa là sau đây:

- Mảng $dd[1..n]$ dùng để đánh dấu dòng. Nếu $dd[i] = 0$ tức là chưa có hậu nào chiếm dòng i , do đó có thể chọn dòng i này để đặt một hậu khác. Ngược lại, nếu $dd[i] = 1$ có nghĩa là đã có hậu nào đó được đặt trên dòng i . Các hậu khác không được phép chiếm dòng i đó nữa.
- Mảng $c1[-(n-1)..(n-1)]$ kiểm soát các đường chéo theo hướng Tây Bắc - Đông Nam. Ta tạm gọi là các đường chéo chính. Có cả thảy $2n-1$ đường chéo trong bàn cờ vuông cạnh n . Nếu hậu i đặt trên dòng j thì sẽ kiểm soát đường chéo chính $i-j$. Như vậy khi $c1[i-j] = 1$ có nghĩa là đã có hậu kiểm soát đường chéo này. Ngược lại, khi $c1[i-j] = 0$ thì đường chéo này rỗng và ta có thể đặt một quân hậu vào “ (x,y) ” của bàn cờ, nếu $y-x = i-j$, trong đó, x, i là các tọa độ dòng và y, j là các tọa độ cột.
- Mảng $c2[2..2n]$ kiểm soát các đường chéo theo hướng Đông Bắc - Tây Nam. Ta tạm gọi là các đường chéo phụ. Nếu hậu i đặt trên dòng j thì sẽ kiểm soát đường chéo phụ $i+j$. Như vậy khi $c1[i+j] = 1$ có nghĩa là đã có hậu kiểm soát đường chéo này. Ngược lại, khi $c1[i+j] = 0$ thì đường chéo này rỗng và ta có thể đặt một quân hậu vào “ (x,y) ” của bàn cờ, nếu $y+x = i+j$, trong đó, x, i là các tọa độ dòng và y, j là các tọa độ cột.

Điều kiện để hậu i có thể đặt trên dòng j khi đó sẽ là:

$$(\text{dd}[j] = 0) \text{ and } (\text{c1}[i-j] = 0) \text{ and } (\text{c2}[i+j] = 0), \text{ hay} \\ (\text{dd}[j] + \text{c1}[i-j] + \text{c2}[i+j] = 0)$$

(* Pascal *)

```
(*=====
                               N Hau
=====*)
{$B-}
uses crt;
const
  MN = 20;
  gn = 'N_HAU.OUT';
  BL = #32; {dau cach} nl = #13#10; { Chuyen dong }
type   mil = array[0..MN] of integer;
var
  v:   mil; { vi tri dat hau }
  c1:  array[-mn..mn] of integer; { cheo 1 }
  c2:  array[0..2*mn] of integer; { cheo 2 }
  dd:  mil;   { dong }
  n:   integer; { so quan hau, kich thuoc ban co }
  g:   text; { file ket qua }

(*-----
    Nhac Hau i khoi ban co
-----*)
procedure NhacHau(i: integer);
begin
  if v[i] = 0 then exit;
  c1[i-v[i]] := 0; c2[i+v[i]] := 0;
```

```

    dd[v[i]] := 0;
end;
(*-----
    Dat Hau i vao dong j
    -----*)
procedure DatHau(i,j: integer);
begin
    c1[i-j] := 1; c2[i+j] := 1;
    dd[j] := 1;
end;
(*-----
    Xuất phát từ dòng v[i]+1,
    tìm dòng j có thể đặt được Hậu i
    -----*)
function Tim(i: integer): integer;
    var j: integer;
begin
    Tim := 0;
    for j := v[i] + 1 to n do
        if ( c1[i-j] + c2[i+j] + dd[j] = 0 ) then
            begin
                Tim := j;
                exit;
            end;
    end;
end;
(*-----
    Hiện thí nghiệm trên màn hình
    Cho bài toán tìm 1 nghiệm
    k = 0: vô nghiệm
    k = n: có nghiệm v[1..n]
    -----*)
procedure Ket1(k: integer);
    var i: integer;
begin
    writeln;
    if k = 0 then write('Vô nghiệm')
    else for i := 1 to k do write(v[i]:3);
    writeln;
end;
(*-----
    Tìm 1 nghiệm: xếp M quân hậu trên
    bàn cờ M X M
    -----*)
procedure XepHau1(M: integer);
    var i,j: integer;
begin
    if (M < 1) or (M > MN) then exit;
    fillchar(c1,sizeof(c1),0);
    fillchar(c2,sizeof(c2),0);
    fillchar(dd,sizeof(dd),0);
    fillchar(v,sizeof(v),0);
    n := M; i := 1; { Đang xét Hậu i }

```

```

repeat
  if i > n then
    begin
      Ket1(n); { co nghiem v[1..n] }
      exit;
    end;
  if i < 1 then
    begin
      Ket1(0); {vo nghiem}
      exit;
    end;
  NhacHau(i); j := Tim(i);
  if j > 0 then
    begin { Tien: Dat Hau i tai dong j }
      DatHau(i,j);
      v[i] := j; inc(i); { Xet Hau i+1 }
    end
  else
    begin { Lui: Dat Hau i ra ngoai ban co }
      v[i] := 0; dec(i); { Xet Hau i-1 }
    end;
until false;
end;
(*-----
  Ghi nghiem thu d vao tep g 'N_Hau.out'
  Bai toan tim moi nghiem
-----*)
procedure Ket(d: integer);
  var i: integer;
begin
  write(g,'Nghiem thu ',d,': ');
  for i := 1 to n do write(g,v[i],BL);
  writeln(g);
end;
(*-----
  Tim moi cach dat M Hau
  tren ban co M X M
-----*)
procedure XepHau(M: integer);
  var i,j: integer;
      d: integer; { dem so nghiem }
begin
  if (M < 1) or (M > MN) then exit;
  n := m;
  fillchar(v,sizeof(v),0);
  fillchar(c1,sizeof(c1),0);
  fillchar(c2,sizeof(c2),0);
  fillchar(dd,sizeof(dd),0);
  assign(g,gn); rewrite(g);
  i := 1; {Hau dang xet}
  d := 0; {dem so nghiem}
  repeat
    if i > n then

```

```

begin
    inc(d);
    Ket(d); { v[1..n] la nghiem thu d }
    i := n;
end;
if i < 1 then
begin
    writeln(g, 'Tong cong ', d, ' nghiem ');
    close(g);
    writeln('Xem ket qua trong file ', gn);
    exit;
end;
NhacHau(i); j := Tim(i);
if j > 0 then
begin { Tien }
    DatHau(i, j); v[i] := j; inc(i);
end
else
begin { Lui }
    v[i] := 0; dec(i);
end;
until false;
end;
procedure Test;
begin
    XepHau1(8); { tim 1 nghiem }
    XepHau(8); { tim du 92 nghiem }
    readln;
end;
BEGIN
    Test;
END.

```

// C#

```

using System;
using System.IO;
namespace SangTao1
{
    /*-----
    *           Bai toan Tam Hau
    *   Phuong an tong quat cho N Hau
    * -----*/
    class TamHau
    {
        static int mn = 20;
        static int mn2 = 2 * mn;
        static int[] v = new int[mn + 1];
        // Vet timkiem, v[i] - dong dat Hau i
        static int[] dd = new int[mn + 1];
        // dd[i] = 1: dong i bi cam
        static int[] cl = new int[mn2 + 1];
        // cl[i] = 1 duong cheo chinh i bi cam
        static int[] c2 = new int[mn2 + 1];
    }
}

```

```

// c2[i] = 1  duong cheo phu i bi cam
static int n = 0; // kích thước ban đầu
static void Main()
{
    Console.WriteLine("\n Test 1: Tìm 1 nghiệm " +
        " với n = 1..10 ");
    Test1();
    Console.ReadLine();
    Console.WriteLine("\n Test 2: Tìm mọi nghiệm " +
        " với n = 8");
    Test2();
    Console.WriteLine("\n Fini ");
    Console.ReadLine();
} // Main
// Test 1: tìm 1 nghiệm với n = 1..10
static void Test1()
{
    for (int i = 1; i <= 10; ++i)
    {
        Console.WriteLine(" \n n = " + i + ": ");
        if (XepHau(i)) Print(v, n);
        else Console.WriteLine(" Vô Nghiệm");
    }
}
static bool XepHau(int SoHau)
{
    if (SoHau > mn || SoHau < 1) return false;
    n = SoHau;
    Array.Clear(v, 0, v.Length);
    Array.Clear(dd, 0, dd.Length);
    Array.Clear(c1, 0, c1.Length);
    Array.Clear(c2, 0, c2.Length);
    int k = 1; // Hạng đang xét
    int dong = 0; // dòng đang xét
    do
    {
        if (k > n) return true;
        if (k < 1) return false;
        NhacHau(k);
        if ((dong = TimNuocDi(k)) > 0)
        {
            DatHau(k, dong);
            v[k++] = dong; // tiến
        }
        else v[k--] = 0; // lùi
    } while (true);
}
// Nhac Hạng k khởi vị trí đang xét
static public void NhacHau(int k)
{
    if (v[k] == 0) return;
    dd[v[k]] = c1[n+(k-v[k])] = c2[k+v[k]] = 0;
}

```

```

// Dat Hau k tai dong i
static public void DatHau(int k, int i)
{
    dd[i] = c1[n+(k-i)] = c2[k+i] = 1;
}
// Test2: Tim moi nghiem
static void Test2()
{
    Console.WriteLine("\n Tong cong " +
                      XepHauNN(8) + " nghiem");
}
// Phuong an tim moi nghiem
// Phuong phap gia sai
static int XepHauNN(int SoHau)
{
    int soNghiem = 0;
    if (SoHau > mn || SoHau < 1) return 0;
    Array.Clear(v, 0, v.Length);
    Array.Clear(dd, 0, dd.Length);
    Array.Clear(c1, 0, c1.Length);
    Array.Clear(c2, 0, c2.Length);
    StreamWriter f =
        File.CreateText("N_HAU.OUT");
    n = SoHau;
    int k = 1;
    int dong = 0;
    do
    {
        if (k > n)
        {
            ++soNghiem;
            Console.Write("\n Nghiem thu "+
                          soNghiem + ": ");
            Print(v, n);
            for (int j = 1; j <= n; ++j)
                f.Write(v[j] + " ");
            f.WriteLine();
            k = n;
        }
        if (k < 1)
        {
            f.WriteLine(soNghiem);
            f.Close();
            return soNghiem;
        }
        NhacHau(k);
        if ((dong = TimNuocDi(k)) > 0)
        {
            DatHau(k, dong);
            v[k++] = dong; // tien
        }
        else v[k--] = 0; // lui
    } while (true);
}

```

```

    }
    // Dịch hau k tu vi tri hien tai v[k]
    // xuong den dong cuoi (n)
    // tim mot vi tri dat hau k
    static int TimNuocDi(int k)
    {
        for (int i = v[k] + 1; i <= n; ++i)
            if ((dd[i] + c1[n+(k-i)] + c2[k+i]) == 0)
                return i;
        return 0;
    }
    static void Print(int[] a, int n)
    {
        for (int i = 1; i <= n; ++i)
            Console.Write(a[i] + " ");
    }
} // TamHau
} // SangTao1

```

Bài 6.2. Từ chuẩn

Một từ loại M là một dãy các chữ số, mỗi chữ số nằm trong khoảng từ 1 đến M . Số lượng các chữ số có mặt trong một từ được gọi là chiều dài của từ đó. Từ loại M được gọi là từ chuẩn nếu nó không chứa hai khúc (từ con) liên nhau mà giống nhau.

a) Với giá trị N cho trước, hiển thị trên màn hình một từ chuẩn loại 3 có chiều dài N .

b) Với mỗi giá trị N cho trước, tìm và ghi vào tệp văn bản tên TUCHUAN.OUT mọi từ chuẩn loại 3 có chiều dài N .

$$1 \leq N \leq 40000.$$

Thí dụ:

1213123 là từ chuẩn loại 3, chiều dài 7.

1213213 không phải là từ chuẩn vì nó chứa liên tiếp hai từ con giống nhau là 213.

Tương tự, 12332 không phải là từ chuẩn vì chứa liên tiếp hai từ con giống nhau là 3.

Bài giải

Ta dùng mảng $v[1..n]$ để lưu từ cần tìm. Tại mỗi bước i ta xác định giá trị $v[i]$ trong khoảng $1..m$ sao cho $v[1..i]$ là từ chuẩn.

Điều kiện P: $v[1..i]$ là từ chuẩn.

Điều kiện Q: Dùng thuật toán theo một trong hai tình huống sau đây:

- nếu $i = n$ thì bài toán có nghiệm $v[1..n]$.
- nếu $i = 0$ thì bài toán vô nghiệm.

TimTul: Tìm một nghiệm.

```

{Khởi trị mọi vị trí bằng 0 }
for i := 1 to n do v[i] := 0;
i := 1;
repeat
    if i > n then {co nghiệm v[1..n]}
    begin
        KetQual(n); {in nghiệm v[1..n]}
    end

```

```

        exit;
    end;
    if i < 1 then {vo nghiem}
    begin
        KetQual(0);
        exit;
    end;
    j := Tim(i);
    if j > 0 then
    begin
        v[i] := j;
        inc(i) {tiến}
    end
    else
    begin {Lùi}
        v[i] := 0;
        dec(i);
    end;
until false;

```

Hàm Tim hoạt động như sau: duyệt các giá trị tại vị trí $v[i]$ của từ $v[1..i]$ kể từ $v[i] + 1$ đến m sao cho $v[1..i]$ là từ chuẩn.

Tim = true nếu tồn tại một giá trị $v[i]$ như vậy. Ngược lại, nếu với mọi $v[i] = v[i] + 1..m$ từ $v[1..i]$ đều không chuẩn thì Tim = false.

```

function Tim(i: integer): Boolean;
begin
    Tim := true;
    while v[i] < 3 do
    begin
        inc(v[i]);
        if Chuan(i) {v[1..i] là tu chuan}
        then exit;
    end;
    Tim := false;
end;

```

Để kiểm tra tính chuẩn của từ $v[1..i]$, ta lưu ý rằng từ $v[1..i-1]$ đã chuẩn (tính chất P), do đó chỉ cần khảo sát các cặp từ có chứa $v[i]$, cụ thể là khảo sát các cặp từ có chiều dài k đứng cuối từ v . Đó là các cặp từ $v[(i-k+1)..(i-k)]$ và $v[i-k+1..i]$ với $k = 1..(i \text{ div } 2)$. Nếu với mọi k như vậy hai từ đều khác nhau thì Chuan=true. Ngược lại, Chuan = false.

```

function Chuan(i: integer): Boolean;
var k: integer;
begin
    Chuan := false;
    for k := 1 to (i div 2) do
        if Bang(i,k) then exit;
    Chuan := true;
end;

```

Hàm Bang(i, k) kiểm tra xem hai từ kề nhau chiều dài k tính từ i trở về trước có bằng nhau hay không.

Hai từ được xem là khác nhau nếu chúng khác nhau tại một vị trí nào đó.

```

function Bang(i,k: integer): Boolean;

```

```

var j: integer;
begin
  Bang := false;
  for j := 0 to k-1 do
    if v[i-j] <> v[i-k-j] then exit;
  Bang := true;
end;

```

Thủ tục `TimTu` tìm mọi nghiệm của bài toán.

```

(* Pascal *)
(*-----*
          Tu chuan
-----*)
{$B- }
uses crt;
const
  MN = 40; {Cho cau b: tim moi nghiem }
  MN1 = 40000; {Cho cau a: tim 1 nghiem }
  gn = 'TuChuan.OUT';
var
  v: array[0..MN1] of byte; {chua nghiem }
  n: integer; {chieu dai tu: tinh chat Q }
  g: text; {output file }
(*-----*
Kiem tra hai tu ke nhau, chieu dai k
tinh tu vi tri i tro ve truoc co bang nhau ?
-----*)
function Bang(i,k: integer): Boolean; tự viết
(*-----*
Kiem tra tu v[1..i] co la tu chuan ?
-----*)
function Chuan(i: integer): Boolean; tự viết
(*-----*
Sua v[i] de thu duoc tu chuan
Tim = true: Thanh cong
Tim = false: That bai
-----*)
function Tim(i: integer): Boolean; tự viết
(*-----*
Hien thi ket qua, tu v[1..n]
(Cau a: tim 1 nghiem)
-----*)
procedure KetQual(k: integer);
var i: integer;
begin
  writeln;
  if k = 0 then write('Vo nghiem')
  else for i := 1 to k do write(v[i]);
  writeln;
end;
(*-----*
Quay lui: tim 1 nghiem cho bai toan
tu chuan chieu dai len, chi chua cac

```

```

chu so 1..lim
-----*)
procedure TimTu1(len: integer); tự viết
(*-----
Test cau a: Tu chuan dai 200
chi chua cac chu so 1, 2, 3
-----*)
procedure Test1;
begin
  clrscr;
  TimTu1(200);
  readln;
end;
(*-----
      Ghi mot nghiem vao file
-----*)
procedure KetQua(d: integer);
var i: integer;
begin
  if d = 0 then write(g, 'Vo nghiem')
  else
    begin
      write(g, 'Nghiem thu ', d, ': ');
      for i := 1 to n do write(g, v[i]);
      writeln(g);
    end;
end;
(*-----
      Cau b: Liet ke toan bo cac tu chuan
      chieu dai len, chi chua cac chu so 1, 2, 3
-----*)
procedure TimTu(len: integer);
var
  i: integer;
  d: longint;
begin
  if (len < 1) or (len > MN) then exit;
  n := len;
  for i := 1 to n do v[i] := 0;
  assign(g, gn);
  rewrite(g);
  i := 1;
  d := 0;
  repeat
    if i > n then {tim duoc 1 nghiem v[1..n]}
    begin
      inc(d);
      KetQua(d);
      i := n;
    end;
  if i < 1 then {da vet het}
  begin
    if d = 0 then KetQua(0);

```

```

        close(g) ;
        write('OK'); readln;
        exit;
    end;

    if Tim(i) then inc(i) {tiến }
    else {Lui }
    begin
        v[i] := 0;
        dec(i);
    end;
    until false;
end;
(*-----
Test cau b: Liet ke toan bo cac
tu dai 16, chi chua cac chu so 1, 2,3
Ket qua ghi trong tep TuChuan.out
-----*)
procedure Test;
begin
    clrscr;
    TimTu(16);
end;
BEGIN
    Test;
END.

```

Với $N = 16$, $M = 3$, có tổng cộng 798 nghiệm, tức là 798 từ chuẩn chiều dài 16 tạo từ các chữ số 1, 2 và 3. Dưới đây là 20 nghiệm đầu tiên tìm được theo thuật toán.

```

Nghiem thu 1: 1213123132123121
Nghiem thu 2: 1213123132123213
Nghiem thu 3: 1213123132131213
Nghiem thu 4: 1213123132131231
Nghiem thu 5: 1213123132131232
Nghiem thu 6: 1213123132312131
Nghiem thu 7: 1213123132312132
Nghiem thu 8: 1213123132312321
Nghiem thu 9: 1213123212312131
Nghiem thu 10: 1213123212312132
Nghiem thu 11: 1213123212313212
Nghiem thu 12: 1213123212313213
Nghiem thu 13: 1213123212313231
Nghiem thu 14: 1213123213121321
Nghiem thu 15: 1213123213121323
Nghiem thu 16: 1213123213231213
Nghiem thu 17: 1213123213231232
Nghiem thu 18: 1213123213231321
Nghiem thu 19: 1213212312131231
Nghiem thu 20: 1213212312131232

```

// C#

```

using System;
using System.IO;
namespace SangTao1
{
    /*-----
    *          Tu chuan
    * -----*/
    class TuChuan
    {
        static int mn = 500000;
        static string fn = "TuChuan.out";
        static int[] v = new int[mn + 1];
        static int n = 0; // kích thước ban đầu
        static int k = 0;
        static void Main()
        {
            int sl = 10;
            Console.WriteLine("Test 1: Tìm 1 nghiệm với n = "+sl);
            Test1(sl);
            Console.WriteLine("Test 2: Tìm mọi nghiệm với n "+sl);
            Test2(sl);
            Console.WriteLine("\n Đọc lại Kết quả:\n");
            Console.WriteLine(File.ReadAllText(fn));
            Console.WriteLine("\n Fini");
            Console.ReadLine();
        }
        // Test 2: tìm mọi nghiệm
        static void Test2(int sl)
        {
            Console.WriteLine(" Tổng cộng " +
                               TimMoiTu(sl) + "
                               nghiệm");
        }
        // Tìm mọi nghiệm. Phương pháp giả sai
        static int TimMoiTu(int len)
        {
            if (len > mn || len < 1) return 0;
            StreamWriter f = File.CreateText(fn);
            n = len;
            int soNghiem = 0;
            Array.Clear(v, 0, v.Length);
            k = 1;
            do
            {
                if (k > n)
                {
                    ++soNghiem;
                    for (int i = 1; i <= n; ++i)
                        f.Write(v[i]);
                    f.WriteLine();
                    k = n;
                }
            }
        }
    }
}

```

```

        if (k < 1)
        {
            f.WriteLine(soNghiem);
            f.Close();
            return soNghiem;
        }
        if (CoNuocDi()) k++; // tien
        else v[k--] = 0; // lui
    } while (true);
}
// Test 1: tim 1 nghiem
static void Test1(int sl)
{
    if (TimTu(sl)) Print(v, n);
    else Console.WriteLine("\n Vo Nghiem");
}
static bool TimTu(int len)
{
    if (len > mn || len < 1) return false;
    n = len;
    for (int i = 0; i <= n; ++i) v[i] = 0;
    k = 1;
    do
    {
        if (k > n) return true;
        if (k < 1) return false;
        if (CoNuocDi()) k++; // tien
        else v[k--] = 0; // lui
    } while (true);
}
static bool CoNuocDi()
{
    while (v[k] < 3)
    {
        ++v[k];
        if (Chuan()) return true;
    }
    return false;
}
// Kiem tra v[1..k] la tu chuan
static bool Chuan()
{
    int k2 = k / 2;
    for (int j = 1; j <= k2; ++j)
        if (Bang(j)) return false;
    return true;
}
// v[k-2d+1..k-d]==v[k-d+1..k] ?
static bool Bang(int d)
{
    int kd = k - d;
    for (int i = 0; i < d; ++i)
        if (v[k - i] != v[kd - i]) return false;
}

```

```

        return true;
    }
    static void Print(int[] a, int n)
    { // moi dong 50 ki tu
        for (int i = 1; i <= n; ++i)
            Console.Write(a[i]+
                ((i%50==0)? "\n": ""));
        Console.WriteLine();
    }
} // TuChuan
} // SangTao1

```

Bài 6.3. Tìm đường trong mê cung.

Mê cung là một đồ thị vô hướng bao gồm N đỉnh, được mã số từ 1 đến N , với các cạnh, mỗi cạnh nối hai đỉnh nào đó với nhau. Cho hai đỉnh S và T trong một mê cung. Hãy tìm một đường đi bao gồm các cạnh nối nhau liên tiếp bắt đầu từ đỉnh S , kết thúc tại đỉnh T sao cho không qua đỉnh nào quá một lần.

Dữ liệu vào: Tập văn bản tên **MECUNG . INP** với cấu trúc như sau:

- Dòng đầu tiên, được gọi là dòng 0, chứa ba số tự nhiên N , S và T ghi cách nhau bởi dấu cách, trong đó N là số lượng đỉnh của mê cung, S là đỉnh xuất phát, T là đỉnh kết thúc.
- Dòng thứ i , $i = 1..(N - 1)$ cho biết có hay không cạnh nối đỉnh i với đỉnh j , $j = (i + 1)..N$.

Thí dụ:

MECUNG . INP									
9	6	7							
1	0	1	1	1	0	0	0		
1	1	0	0	0	0	0			
0	0	0	1	0	0				
0	1	1	0	0					
0	0	0	0						
0	0	0							
0	0								
1									

cho biết:

- Dòng 0: 9 6 7 - mê cung gồm 9 đỉnh mã số 1..9, cần tìm đường đi từ đỉnh 6 đến đỉnh 7.

- Dòng 1: 1 0 1 1 1 0 0 0 - đỉnh 1 được nối với các đỉnh 2, 4, 5, và 6. Không có cạnh nối đỉnh 1 với các đỉnh 3, 7, 8 và 9.

- ...

- Dòng 8: 1 - đỉnh 8 có nối với đỉnh 9.

Vì đồ thị là vô hướng nên cạnh nối đỉnh x với đỉnh y cũng chính là cạnh nối đỉnh y với đỉnh x .

Thông tin về đỉnh N không cần thông báo, vì với

mỗi đỉnh i ta chỉ liệt kê các đỉnh $j > i$ tạo thành cạnh (i, j) .

Kết quả ra ghi trong tập văn bản **MECUNG . OUT**:

- Dòng đầu tiên ghi số tự nhiên k là số đỉnh trên đường đi từ s đến t , nếu vô nghiệm, ghi số 0.
- Từ dòng tiếp theo ghi lần lượt các đỉnh có trên đường đi.

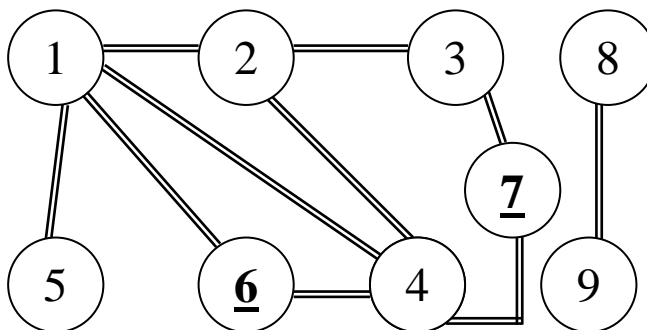
Với thí dụ đã cho kết quả có thể là:

MECUNG . OUT									
5									
6	4	2	3	7					

Từ đỉnh 6 có thể đến được đỉnh 7, qua 5 đỉnh theo đường bốn khúc:

$6 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 7$.

Với mê cung đã cho, nếu yêu cầu tìm đường đi từ đỉnh 6 đến đỉnh 9, tức là với dữ liệu vào như trên thì sẽ nhận được kết quả 0 với ý nghĩa là không có đường đi từ đỉnh 6 đến đỉnh 9, do mê cung đã cho không liên thông, đỉnh 6 và đỉnh 9 nằm trong hai vùng liên thông khác nhau.



Thuật toán

Xuất phát từ đỉnh $v[1] = s$, mỗi bước lặp i ta thực hiện các kiểm tra sau. Gọi k là số đỉnh đã đi qua và được tích lũy trong mảng giải trình đường đi v , cụ thể là xuất phát từ đỉnh $v[1] = s$, sau một số lần duyệt ta quyết định chọn đường đi qua các đỉnh $v[1], v[2], v[3], \dots, v[k]$. Có thể gặp các tình huống sau:

a) (Đến đích?) nếu $v[k] = t$ tức là đã đến được đỉnh t : thông báo kết quả, dừng thuật toán, ngược lại thực hiện b .

b) (Thất bại?) $k = 0$: nếu đã quay trở lại vị trí xuất phát $v[i] = s$ mà từ đó không còn đường đi nào khác thì phải lùi một bước nữa, do đó $k = 0$. Trường hợp này chứng tỏ bài toán vô nghiệm, tức là, do đồ thị không liên thông nên không có đường đi từ đỉnh s đến đỉnh t . Ta thông báo vô nghiệm và dừng thuật toán.

c) (Đi tiếp?) nếu từ đỉnh $v[k]$ tìm được một cạnh chưa đi qua và dẫn đến một đỉnh i nào đó thì tiến theo đường đó, nếu không: thực hiện bước d .

d) (Lùi một bước) Bỏ đỉnh $v[k]$, lùi lại đỉnh $v[k-1]$.

Thuật toán trên có tên là *sơ chỉ Arian* được phỏng theo một truyền thuyết cổ Hy Lạp sau đây. Anh hùng Te-dây phải tìm diệt con quái vật nhân ngư (đầu người, mình trâu) Minotav ẩn náu trong một phòng của mê cung có nhiều ngõ ngách rắc rối đã từng làm lạc bước nhiều dũng sĩ và những người này đều trở thành nạn nhân của Minotav. Người yêu của chàng Te-dây là công chúa của xứ Mino đã đưa cho chàng một cuộn chỉ và dặn chàng như sau: Chàng hãy buộc một đầu chỉ vào cửa mê cung (phòng xuất phát s), sau đó, tại mỗi phòng trong mê cung, chàng hãy tìm xem có Minotav ẩn trong đó không. Nếu có, chàng hãy chiến đấu dũng cảm để hạ thủ nó rồi cuộn chỉ quay ra cửa hang, nơi em trông ngóng chàng. Nếu chưa thấy Minotav tại phòng đó, chàng hãy kiểm tra xem chỉ có bị rối hay không. Cuộn chỉ bắt đầu rối khi nào từ phòng chàng đứng có hai sợi chỉ đi ra hai cửa khác nhau. Nếu chỉ rối như vậy, chàng hãy cuộn chỉ để lùi lại một phòng và nhớ đánh dấu đường đã đi để khỏi lạc bước vào đó lần thứ hai.

Nếu không gặp chỉ rối thì chàng hãy yên tâm dò tìm một cửa chưa đi để qua phòng khác. Đi đến đâu chàng nhớ nhả chỉ theo đến đó. Nếu không có cửa để đi tiếp hoặc từ phòng chàng đang đứng, mọi cửa ra đều đã được chàng đi qua rồi, thì chàng hãy cuộn chỉ để lùi lại một phòng rồi tiếp tục tìm cửa khác.

Ta xuất phát từ sơ đồ tổng quát cho lớp bài toán quay lui.

```

(*      Pascal      *)
(*-----
          MC - Tìm dương trong mê cung
          (Thuật toán Arian)
          s: dinh xuất phát
          t: dinh ket.
-----*)

procedure MC;
var i: byte;
begin
  Doc; {doc du lieu}
  {-----
    khai tao mang d,
    danh dau cac dinh da tham:
    d[i] = 1: dinh da tham
    d[i] = 0: dinh chua tham
    -----}
  fillchar(d,sizeof(d),0);
  k := 1; {k - dem so dinh da chon }
  v[k] := s; {dinh xuất phát }
  d[s] := 1; {da tham dinh s }
  repeat
    if v[k] = t then {den dich }
    begin
      ket(k); {ghi ket qua: giai trinh duong di }
      exit;
    end;
    if k < 1 then {vo nghiem }
    begin
      ket(0);
      exit;
    end;
    i := Tim;
    {tu dinh v[k] tim 1 dinh chua tham i }
    if i > 0 then
      {neu tim duoc, i > 0, di den dinh i }
      NhaChi(i)
    else CuonChi;
    {neu khong tim duoc, }
    { i = 0: lui 1 buoc - bo dinh v[k] }
  until false;
end;

Thủ tục Doc - đọc dữ liệu từ tệp MECUNG.INP vào mảng hai chiều a. Đây chính
là ma trận kề của đồ thị biểu diễn mê cung. Mảng a sẽ đối xứng vì mê cung là đồ thị vô
hướng. Đây cũng chính là lý do giải thích dữ liệu vào chỉ cho dưới dạng nửa trên của ma
trận kề.

(*-----
          Doc du lieu
-----*)

procedure Doc;
var i,j: byte;
begin

```

```

assign(f,fn);
reset(f);
read(f,n,s,t);
fillchar(a,sizeof(a),0);
if (n < 1) or (n > MN) then exit;
for i := 1 to n-1 do
  for j := i+1 to n do
    begin
      read(f,a[i,j]);
      a[j,i] := a[i,j]; {lay doi xung }
    end;
  close(f);
end;

```

Thủ tục Xem – hiển thị dữ liệu trên màn hình để kiểm tra việc đọc có đúng không. Với những người mới lập trình cần luôn luôn viết thủ tục Xem. Khi nộp bài thì có thể bỏ lời gọi thủ tục này. Các hằng kiểu **string** **bl = #32** là mã ASCII của dấu cách, hằng **nl = #13#10** là một xâu chứa hai kí tự điều khiển có mã ASCII là xuống dòng #13, tức là ứng với phím **RETURN** và đưa con trỏ màn hình về đầu dòng #10. Khi đó lệnh **writeln** sẽ tương đương với lệnh **write(nl)**.

```

(*-----
      Xem du lieu
-----*)
procedure xem;
var i,j: byte;
begin
  write(nl,n,bl,s,bl,t,nl);
  for i := 1 to n do
    begin
      for j := 1 to n do
        write(a[i,j],bl);
      write(nl);
    end;
end;

```

Thủ tục **Ket(k)** - ghi đường đi $v[1..k]$ từ s đến t tìm được vào tệp output.

Ket(0) : thông báo vô nghiệm.

```

(*-----
      Ghi ket qua.
      k = 0: vo nghiem
      k > 0: co duong tu s den t
              gom k canh
-----*)
procedure Ket(k: byte);
var i: byte;
begin
  assign(g,gn); rewrite(g);
  write(g,k,nl);
  if k > 0 then
    begin
      write(g,v[1]);
      for i := 2 to k do
        write(g,bl,v[i]);
    end;
end;

```

```

end;
close(g) ;
end;

```

Hàm **Tim** - từ đỉnh $v[k]$ tìm một bước đi đến đỉnh i . Điều kiện: i phải là đỉnh chưa thăm và đương nhiên có cạnh đi từ $v[k]$ đến i , nghĩa là giá trị $a[v[k], i]$ trong ma trận kề phải là 1. Ta dùng một mảng d đánh dấu đỉnh i đã thăm chưa. $d[i] = 0$ – đỉnh i chưa thăm, $d[i] = 1$ – đỉnh i đã thăm và đã từng được chọn để đưa vào mảng v là mảng giải trình đường đi. Nếu tìm kiếm thành công ta gán cho hàm **Tim** giá trị i , chính là đỉnh cần đến. Ngược lại, khi việc tìm kiếm thất bại, nghĩa là không tìm được đỉnh i để có thể đi từ đỉnh $v[k]$ đến đó, ta gán cho hàm **Tim** giá trị 0.

Ta lưu ý là mỗi đỉnh chỉ đi đến không quá một lần. Đương nhiên khi lùi thì ta buộc phải quay lại đỉnh đã đến, do đó, chính xác hơn ta phải gọi $d[i]=1$ là giá trị đánh dấu khi tiến đến đỉnh i .

```

(*-----
  Tu dinh v[k] tim duoc mot buoc di
  den dinh i. Dieu kien:
    d[i] = 0 - dinh i chua xuat hien
              trong lich trinh v
    d[i] = 1 - dinh i da xuat hien
              trong lich trinh v.
  -----*)
function Tim: byte;
var i: byte;
begin
  Tim := 0;
  for i := 1 to n do
    if d[i] = 0 then {dinh i chua tham }
      if a[v[k],i] = 1 {co duong tu v[k] den i }
      then
        begin
          Tim := i;
          exit;
        end;
  end;
end;

```

Nếu tìm được đỉnh chưa thăm thoả các điều kiện nói trên ta tiến thêm một bước theo cạnh $(v[k], i)$. Ta cũng đánh dấu đỉnh i là đã thăm bằng lệnh gán $d[i] := 1$. Đó là nội dung của thủ tục **NhaChi** (nhả chỉ).

```

(*-----
  Di 1 buoc tu v[k] den i
  -----*)
procedure NhaChi(i: byte);
begin
  inc(k) ;
  v[k] := i; {tien them 1 buoc }
  d[i] := 1; {danh dau dinh da qua }
end;

```

Nếu từ đỉnh $v[k]$ ta không tìm được đỉnh nào để đi tiếp thì ta phải thực hiện thủ tục **CuonChi** (cuộn chỉ) như dưới đây. Thủ tục này chỉ đơn giản là lùi một bước từ đỉnh $v[k]$ hiện đang đứng trở về đỉnh trước đó, nếu có, tức là $k \geq 1$, ta đánh dấu cạnh $(v[k-1], v[k])$ là đã đi hai lần. Ta nhận xét rằng, nếu không tính lần trở lại một đỉnh khi phải

lùi một bước thì mỗi đỉnh trong mê cung chỉ cần thăm tối đa là một lần, do đó thay vì đánh dấu cạnh ($v[k-1]$, $v[k]$) ta chỉ cần đánh dấu đỉnh $v[k]$ là đủ.

```
(*-----
Lui 1 buoc vi tu dinh v[k] khong
co kha nang nao dan den ket qua
-----*)

procedure CuonChi;
begin
    dec(k);
end;

(* Pascal *)

(*-----
MECUNG.PAS Tim duong trong me cung
-----*)

{$B-}
uses crt;
const
    MN = 100; {So dinh toi da }
    fn = 'MECUNG.INP'; {input file }
    gn = 'MECUNG.OUT'; {output file }
    nl = #13#10; {xuong dong moi }
    bl = #32; {dau cach }
type
    MB1 = array[0..MN] of byte;
    MB2 = array[0..MN] of MB1;
var
    a: MB2; {ma tran ke, doi xung }
    v: MB1; {vet tim kiem }
    d: MB1; {danh dau dinh da chon }
    n: byte; {so dinh }
    s: byte; {dinh xuat phat }
    t: byte; {dinh ket thuc }
    k: byte; {buoc duyett }
    f,g: text; {f: input file; g: output file}
(*-----
Doc du lieu
-----*)
procedure Doc; tự viết
(*-----
Xem du lieu
-----*)
procedure xem; tự viết
(*-----
Ghi ket qua.
    k = 0: vo nghiem
    k > 0: co duong tu s den t gom k canh
-----*)
procedure Ket(k: byte); tự viết
(*-----
Tu dinh v[k] tim duoc mot buoc di den dinh i.
Dieu kien:
```

```

        d[i] = 0 - dinh i chua xuat hien
                    trong lich trinh v
        d[i] = 1 - dinh i da xuat hien
                    trong lich trinh v,
        -----*)
function Tim: byte; tự viết
(*-----
    Di 1 buoc tu v[k] den i
    -----*)
procedure NhaChi(i: byte); tự viết
(*-----
    Lui 1 buoc vi tu dinh v[k] khong co kha nang nao
    dan den ket qua
    -----*)
procedure CuonChi; tự viết
(*-----
    Tim duong trong me cung
    (Thuat toan Soi chi Arian)
    s: dinh xuat phat
    t: dinh ket.
    -----*)
procedure MC; tự viết
BEGIN
    MC; write(nl, 'fini');
END.

```

Với thí dụ đã cho trong đề bài, bạn hãy chạy thử chương trình MECUNG.PAS với hai dữ liệu kiểm thử, một dữ liệu kiểm thử có nghiệm và một dữ liệu kiểm thử vô nghiệm.

Chú ý

Đường đi tìm được không phải là đường ngắn nhất. Trong chương 7 ta sẽ dùng thuật giải Dijkstra để tìm đường đi ngắn nhất.

// C#

```

using System;
using System.IO;
namespace SangTaoT1
{
    /*-----
    *          Tim duong trong me cung
    * -----*/
    class MeCung
    {
        static string fn = "MeCung.INP";
        static string gn = "MeCung.OUT";
        static int mn = 200; // So dinh toi da
        static int[] v; // vet duong di
        static int[] d; // dinh dang xet
        static int[,] c; // ma tran ke 0/1
        static int n = 0; // So dinh
        static int s = 0; // Dinh xuat phat
        static int t = 0; // Dinh ket
        static int k = 0; // buoc duyet
    }
}

```

```

static void Main()
{
    Doc(); Show(); Ghi(MC());
    // Doc lai de kiem tra
    Console.WriteLine("\n Kiem tra");
    Console.WriteLine("\n Input: \n");
    Console.WriteLine(File.ReadAllText(fn));
    Console.WriteLine("\n Output: ");
    Console.WriteLine(File.ReadAllText(gn));
    Console.WriteLine("\n Fini ");
    Console.ReadLine();
} // Main
static void Doc()
{
    int[] a = Array.ConvertAll(
        ((File.ReadAllText(fn)).Trim()).
Split(new char[] { ' ', '\n', '\r', '\t', '\0' },
StringSplitOptions.RemoveEmptyEntries),
new Converter<String, int>(int.Parse));
    n = a[0]; // so dinh
    s = a[1]; // dinh xuat phat
    t = a[2]; // dinh ket
    c = new int[n + 1, n + 1];
    // c ma tran ke
    v = new int[n + 1]; // vet duong di
    d = new int[n + 1];
    // d[i] = 1: da tham dinh i
    k = 2;
    for (int i = 1; i <= n; ++i)
    {
        c[i, i] = 0;
        for (int j = i + 1; j <= n; ++j)
            c[i, j] = c[j, i] = a[++k];
    }
}
// Hien thi de kiem tra
// thu tuc doc du lieu
static void Show()
{
    Console.WriteLine("\n" + n + " "
        + s + " " + t);
    for (int i = 1; i <= n; ++i)
    {
        Console.WriteLine();
        for (int j = 1; j <= n; ++j)
            Console.Write(c[i, j] + " ");
    }
}
static void Ghi(bool Ket)
{
    StreamWriter f = File.CreateText(gn);
    if (Ket) // co nghiem

```

```

        {
            f.WriteLine(k);
            for (int i = 1; i <= k; ++i)
                f.Write(v[i] + " ");
        }
        else f.WriteLine(0); // vo nghiem
        f.Close();
    }
    static bool MC()
    {
        Array.Clear(v, 0, v.Length);
        Array.Clear(d, 0, d.Length);
        k = 1; // Buoc duyot
        v[k] = s; d[s] = 1;
        // danh dau phong da den
        int phong = 0;
        do
        {
            if (v[k] == t)
                return true; // den dich
            if (k < 1)
                return false; // het cach
            if ((phong = Tim()) > 0)
            { // Tien them 1 buoc
                // nha chi, danh dau
                v[++k] = phong; d[phong] = 1;
            }
            else --k; // lui
        } while (true);
    }
    // Tu phong v[k] tim duoc
    //mot duong sang phong khac
    static int Tim()
    {
        for (int j = 1; j <= n; ++j)
            if (d[j] == 0) // phong j chua tham
                if (c[v[k], j] > 0)
                    //co hanh lang toi j
                    return j;
        return 0;
    }
} // MeCung
} // SangTao1

```

CHƯƠNG 7

QUY HOẠCH ĐỘNG

Các bài toán quy hoạch động chiếm một vị trí khá quan trọng trong tổ chức hoạt động và sản xuất. Chính vì lẽ đó mà trong các kì thi học sinh giỏi quốc gia và quốc tế chúng ta thường gặp loại toán này.

Thông thường những bạn nào dùng phương pháp quay lui, vét cạn cho các bài toán quy hoạch động thì chỉ có thể vét được các tập dữ liệu nhỏ, kích thước chừng vài chục byte. Nếu tìm được đúng hệ thức thể hiện bản chất quy hoạch động của bài toán và khéo tổ chức dữ liệu thì ta có thể xử lí được những tập dữ liệu khá lớn.

Có thể tóm lược nguyên lí quy hoạch động do Bellman phát biểu như sau:

Quy hoạch động

Quy hoạch động là lớp các bài toán mà quyết định ở bước thứ i phụ thuộc vào quyết định ở các bước đã xử lí trước hoặc sau đó.

Để giải các bài toán quy hoạch động, ta có thể theo sơ đồ sau đây:

Sơ đồ giải bài toán quy hoạch động

1. *Lập hệ thức*: Lập hệ thức biểu diễn tương quan quyết định của bước đang xử lí với các bước đã xử lí trước đó. Khi đã có hệ thức tương quan chúng ta đã có thể xây dựng ngay thuật giải, tuy nhiên các hệ thức này thường là các biểu thức đệ quy, do đó dễ gây ra hiện tượng tràn miền nhớ khi ta tổ chức chương trình trực tiếp bằng đệ quy.
2. *Tổ chức dữ liệu và chương trình*: Tổ chức dữ liệu tính toán dần theo từng bước. Nên tìm cách khử đệ quy. Trong các bài toán quy hoạch động thuộc chương trình phổ thông thường đòi hỏi một vài mảng hai chiều.
3. *Làm tốt*: Làm tốt thuật toán bằng cách thu gọn hệ thức quy hoạch động và giảm kích thước miền nhớ.

Bài 7.1. Chia thưởng

Cần chia hết m phần thưởng cho n học sinh sắp theo thứ tự từ giỏi trở xuống sao cho mỗi bạn không nhận ít phần thưởng hơn bạn xếp sau mình.

$$1 \leq m, n \leq 70.$$

Hãy tính số cách chia.

Thí dụ, với số phần thưởng $m = 7$, và số học sinh $n = 4$ sẽ có 11 cách chia 7 phần thưởng cho 4 học sinh theo yêu cầu của đầu bài. Đó là:

Phương án	①	②	③	④
1	7	0	0	0
2	6	1	0	0
3	5	2	0	0
4	5	1	1	0
5	4	3	0	0
6	4	2	1	0
7	3	3	1	0
8	3	2	2	0
9	4	1	1	1
10	3	2	1	1
11	2	2	2	1

Bài giải

1. Lập hệ thức

Gọi $\text{Chia}(i, j)$ là số cách chia i phần thưởng cho j học sinh, ta thấy:

- Nếu không có học sinh nào ($j = 0$) thì không có cách chia nào ($\text{Chia} = 0$).
- Nếu không có phần thưởng nào ($i = 0$) thì chỉ có một cách chia ($\text{Chia}(0, j) = 1$ - mỗi học sinh nhận 0 phần thưởng). Ta cũng quy ước $\text{Chia}(0, 0) = 1$.
- Nếu số phần thưởng ít hơn số học sinh ($i < j$) thì trong mọi phương án chia, từ học sinh thứ $i + 1$ trở đi sẽ không được nhận phần thưởng nào:

$$\text{Chia}(i, j) = \text{Chia}(i, i) \text{ nếu } i < j.$$

Ta xét tất cả các phương án chia trong trường hợp $i \geq j$. Ta tách các phương án chia thành hai nhóm không giao nhau dựa trên số phần thưởng mà học sinh đứng cuối bảng thành tích, học sinh thứ j , được nhận:

- Nhóm thứ nhất gồm các phương án trong đó học sinh thứ j không được nhận thưởng, tức là i phần thưởng chỉ chia cho $j - 1$ học sinh và do đó, số cách chia, tức là số phần tử của nhóm này sẽ là: $\text{Chia}(i, j - 1)$.
- Nhóm thứ hai gồm các phương án trong đó học sinh thứ j cũng được nhận thưởng. Khi đó, do học sinh đứng cuối bảng thành tích được nhận thưởng thì mọi học sinh khác cũng sẽ có thưởng. Do ai cũng được thưởng nên ta bớt của mỗi người một phần thưởng (để họ lĩnh sau), số phần thưởng còn lại ($i - j$) sẽ được chia cho j học sinh. Số cách chia khi đó sẽ là $\text{Chia}(i - j, j)$.

Tổng số cách chia cho trường hợp $i \geq j$ sẽ là tổng số phần tử của hai nhóm, ta có:

$$\text{Chia}(i, j) = \text{Chia}(i, j - 1) + \text{Chia}(i - j, j).$$

Tổng hợp lại ta có:

Điều kiện	$\text{Chia}(i, j)$
i : số phần thưởng	
j : số học sinh	
$j = 0$	$\text{Chia}(i, j) = 0$
$i = 0 \text{ and } j \neq 0$	$\text{Chia}(i, j) = 1$

$$\begin{aligned}
 i < j & \quad Chia(i, j) = Chia(i, i) \\
 i \geq j & \quad Chia(i, j) = Chia(i, j-1) + Chia(i-j, j)
 \end{aligned}$$

Các tính chất của hàm Chia(i, j)
Chia i phần thưởng cho j học sinh

2. Tổ chức dữ liệu và chương trình

Ta có phương án đầu tiên của giải thuật Chia như sau:

```

(*-----
PHUONG AN 1: de quy.
So cach Chia i phan thuong cho j hs
-----*)

function Chia(i, j: integer): longint;
begin
    if j = 0 then Chia := 0
    else {j > 0}
        if i = 0 then {i = 0; j > 0}
            Chia := 1
        else {i, j > 0}
            if i < j then {0 < i < j}
                Chia := Chia(i, i)
            else {i >= j > 0}
                Chia := Chia(i, j-1) + Chia(i-
j, j);
end;

```

	①	②	③	④	
①	0	9	1	1	0
②	9	9	2	1	0
③	6	6	1	0	0
④	5	5	2	1	1
⑤	3	3	1	1	0
⑥	2	2	1	0	0
⑦	1	1	0	0	0
⑧	1	1	1	1	1

*Số lần gọi hàm Chia cục bộ
khi tính hàm Chia(⑧, ④)*

Phương án này chạy chậm vì phát sinh ra quá nhiều lần gọi hàm trùng lặp. Bảng dưới đây liệt kê số lần gọi hàm Chia khi giải bài toán chia thưởng với bảy phần thưởng ($m = 7$) và 4 học sinh ($n = 4$). Thí dụ, hàm Chia(1, 1) sẽ được gọi 9 lần, ... Tổng số lần gọi hàm Chia là 79. 79 lần gọi hàm để sinh ra kết quả 11 là quá tốn kém.

Làm tốt lần 1: Phương án 1 khá dễ triển khai nhưng chương trình sẽ chạy rất lâu, bạn hãy thử gọi Chia(66, 32) để trải nghiệm được điều trên. Diễn tả đệ quy thường trong sáng, nhân bản, nhưng khi thực hiện sẽ sinh ra hiện tượng gọi lặp lại những hàm đệ quy. Cải tiến đầu tiên là tránh những lần gọi lặp như vậy. Muốn thế chúng ta tính sẵn các giá trị của hàm theo các trị của đầu vào khác nhau và điền vào một mảng hai chiều cc.

Mảng cc được mô tả như sau:

		j - 1	j	
i - j			[i - j, j]	
...			...	

```

const
  MN = 70; { giới hạn trên
            của m và n }
type
  m11 = array[0..MN] of longint;
  m12 = array[0..mn] of m11;
var cc: m12;

```

i		[i, j-1]	[i, j]	

Ta quy ước $cc[i, j]$ chứa số cách chia i phần thưởng cho j học sinh.

Theo phân tích của phương án 1, ta có:

- ♦ $cc[0, 0] = 1; cc[i, 0] = 0$, với $i := 1..m$.
- ♦ $cc[i, j] = cc[i, i]$, nếu $i < j$
- ♦ $cc[i, j] = cc[i, j-1] + cc[i-j, j]$, nếu $i \geq j$.

Từ đó ta suy ra quy trình điền trị vào bảng cc như sau:

- ♦ Khởi trị
- ♦ $cc[0, 0] := 1;$
- ♦ với $i := 1..m: cc[i, 0] := 0;$
- ♦ Điền bảng: Lần lượt điền theo từng cột $j := 1..n$. Tại mỗi cột j ta đặt:
- ♦ với $i := 0..j-1: cc[i, j] := cc[i, i];$
- ♦ với $i := j..m: cc[i, j] := cc[i, j-1] + cc[i-j, j];$

Nhận kết quả: Sau khi điền bảng, giá trị $cc[m, n]$ chính là kết quả cần tìm.

```

(*-----
PHUONG AN 2: dùng mảng 2 chiều cc
cc[i,j] = Chia(i,j) - số cách chia i
phần thưởng cho j hs
-----*)
function Chia2(m,n: integer):longint;
var i,j: integer;
begin
  cc[0,0] := 1;
  for i := 1 to m do cc[i,0] := 0;
  for j := 1 to n do
    begin
      for i := 0 to j-1 do cc[i,j] := cc[i,i];
      for i := j to m do
        cc[i,j] := cc[i,j-1] + cc[i-j,j];
      end;
    Chia2 := cc[m,n];
  end;
end;

```

Làm tốt lần 2: Dùng mảng hai chiều chúng ta chỉ có thể tính toán được với dữ liệu nhỏ. Bước cải tiến sau đây khá quan trọng: chúng ta dùng mảng một chiều. Quan sát kỹ quy trình gán trị cho mảng hai chiều theo từng cột chúng ta dễ phát hiện ra rằng cột thứ j có thể được tính toán từ cột thứ $j-1$. Nếu gọi c là mảng một chiều sẽ dùng, ta cho số học sinh tăng dần bằng cách lần lượt tính j bước, với $j := 1..n$. Tại bước thứ j , $c[i]$ chính là số cách chia i phần thưởng cho j học sinh. Như vậy, tại bước thứ j ta có:

- $c[i]$ tại bước $j = c[i]$ tại bước $(j-1)$, nếu $i < j$. Từ đây suy ra đoạn $c[0..(j-1)]$ được bảo lưu.
- $c[i]$ tại bước $j = c[i]$ tại bước $(j-1) + c[i-j]$ tại bước j , nếu $i \geq j$.

Biểu thức thứ hai cho biết khi cập nhật mảng c từ bước thứ $j-1$ qua bước thứ j ta phải tính từ trên xuống, nghĩa là tính dần theo chiều tăng của $i := j..m$.

Mảng c được khởi trị ở bước $j=0$ như sau:

- $c[0] = 1; c[i] = 0$, với $i := 1..m$.

Với ý nghĩa là, nếu có 0 học sinh thì chia 0 phần thưởng cho 0 học sinh sẽ được quy định là 1. Nếu số phần thưởng m khác 0 thì chia m phần thưởng cho 0 học sinh sẽ được 0 phương án.

Ta có phương án ba, dùng một mảng một chiều c như sau:

```
(*-----*
      PHUONG AN 3: dung mang 1 chieu c
      Tai buoc j, c[i] = so cach chia i
      phan thuong cho j hoc sinh.
-----*)
function Chial(m,n: integer):longint;
var i,j: integer;
begin
  fillchar(c,sizeof(c),0);
  c[0] := 1;
  for j := 1 to n do
    for i := j to m do c[i] := c[i]+c[i-j];
  Chial := c[m];
end;
```

Để so sánh các phương án bạn hãy đặt một bộ đếm nhịp của máy như sau:

```
nhip: longint absolute $0000:$046c;
{xac dinh nhip thoi gian }
t: longint; {ghi nhan nhip }
```

Sau đó bạn tạo một dữ liệu kiểm thử để so sánh ba phương án đã phân tích ở phần trên như sau:

```
procedure test;
begin
  randomize; {Khoi dong bo sinh so ngau nhien }
  repeat
    m := random(mn)+1; {sinh ngau nhien so phan
    thuong m }
    n := random(mn)+1; {sinh ngau nhien so hs n }
    writeln(m,bl,n); {xem du lieu vao }
    t := Nhup; {dat nhup cho PA 3 }
    {Phuong an 3 }
    writeln('Mang 1 chieu: ',Chial(m,n));
    {bao thoi gian }
    writeln((Nhup-t)/18.2):0:0,' giay');
    t := Nhup; {dat nhup cho PA 2}
    writeln('Mang 2 chieu: ',Chia2(m,n)); {PA 2 }
    {bao thoi gian }
    writeln((Nhup-t)/18.2):0:0,' giay');
    t := Nhup; {dat nhup cho PA 1 }
    writeln('De quy: ',Chia(m,n));
    {bao tgian}
    writeln((Nhup-t)/18.2):0:0,' giay');
  until readkey = #27; {lap den khi bam ESC }
end;
```

Các giá trị m – số phần thưởng và n – số học sinh được sinh ngẫu nhiên nhờ hàm random. Trước đó cần gọi thủ tục randomize để chuẩn bị khởi tạo bộ sinh số ngẫu nhiên.

Trong bộ nhớ của máy tính có 4 byte bắt đầu từ địa chỉ \$0000:\$046c dùng để ghi số nhị phân của máy tính. Mỗi lần đọc giá trị của biến Nhị phân ta có thể lấy được số nhị phân hiện hành của máy. Hiệu số hai lần đọc nhị phân liên tiếp sẽ cho ta tổng số nhị phân tính từ lần đọc thứ nhất đến lần đọc thứ hai. Chia giá trị này cho 18.2 ta có thể quy ra lượng thời gian chạy máy tính bằng giây. Lệnh `write(r:d:p)` hiển thị số thực r với d vị trí và p chữ số sau dấu phẩy. Nếu đặt $d = p = 0$ thì số thực r sẽ được hiển thị đầy đủ.

```
(* Pascal *)
uses crt;
const
  MN = 70; {gioi han tren cua m va n }
  nl = #13#10; {xuong dong }
  bl = #32; {dau cach }
type
  ml1 = array[0..MN] of longint;
  ml2 = array[0..mn] of ml1;
var
  cc: ml2; {cho phuong an 2 - mang 2 chieu }
  m,n: integer;
  c: ml1; {cho phuong an 3 - mang 1 chieu }
  nhip: longint absolute $0000:$046c;
  {xac dinh nhip thoi gian }
  t: longint; {ghi nhan nhip }
(*-----
          PHUONG AN 1: de quy
          So cach Chia i phan thuong cho j hs
          -----*)
function Chia(i,j: integer):longint; tự viết
(*-----
          PHUONG AN 2: dung mang 2 chieu cc
          cc[i,j] = so cach chia i phan thuong
                    cho j hs
          -----*)
function Chia2(m,n: integer):longint; tự viết
(*-----
          PHUONG AN 3: dung mang 1 chieu c
          Tai buoc j, c[i] = so cach chia i
                    phan thuong cho j hoc sinh.
          -----*)
function Chial(m,n: integer):longint; tự viết
procedure test; tự viết
BEGIN
  Test;
END.
```

Quan sát hoạt động của chương trình bạn sẽ rút ra được ý nghĩa của các phương án cải tiến.

Chú thích

Bài toán trên còn có cách phát biểu khác như sau: Hãy tính số cách biểu diễn số tự nhiên m thành tổng của n số tự nhiên sắp theo trật tự không tăng. Thí dụ, với $m = 7, n = 4$ ta có:

$$7 = 7 + 0 + 0 + 0 = 6 + 1 + 0 + 0 = \dots$$

// C#

```

using System;
using System.IO;
namespace SangTao1
{
    /*-----
    *                      Chia thuong
    * -----*/
    class ChiaThuong
    {
        static void Main()
        {
            Console.WriteLine(Chia(7, 4));
            Console.WriteLine("\n Fini");
            Console.ReadLine();
        } // Main
        static long Chia(int m, int n)
        {
            long[] c = new long[m+1];
            Array.Clear(c, 0, c.Length);
            c[0] = 1;
            for (int j = 1; j <= n; ++j)
                for (int i = j; i <= m; ++i)
                    c[i] += c[i - j];
            return c[m];
        }
    } // ChiaThuong
} // SangTao1

```

Bài 7. 2. Palindrome

Olympic Quốc tế, năm 2000, Bắc Kinh, Trung Quốc.

Dãy kí tự s được gọi là đối xứng (palindrome) nếu các phần tử cách đều đầu và cuối giống nhau. Cho dãy s tạo bởi n kí tự gồm các chữ cái hoa và thường phân biệt và các chữ số. Hãy cho biết cần xoá đi từ s ít nhất là bao nhiêu kí tự để thu được một dãy đối xứng. Giả thiết rằng sau khi xoá bớt một số kí tự từ s thì các kí tự còn lại sẽ tự động xích lại sát nhau.

Dữ liệu vào ghi trong tệp văn bản **PALIN.INP** với cấu trúc như sau:

Dòng đầu tiên là giá trị n , $1 \leq n \leq 1000$.

Dòng thứ hai là n kí tự của dãy viết liền nhau.

PALIN.INP	PALIN.OUT
9 bae ad bad b	4

Dữ liệu ra ghi trong tệp văn bản **PALIN.OUT**: số lượng kí tự cần xoá.

Thí dụ, với dãy s gồm 9 kí tự, $s = \text{'bae**ad**bad**b**'}$ thì cần xoá ít nhất 4 kí tự, chẳng hạn, các kí tự thứ 5, 7, 8 và 9 sẽ thu được dãy đối xứng chiều dài 5 là baeab:

bae**ad**bad**b** \rightarrow baeab

Dĩ nhiên là có nhiều cách xoá. Thí dụ, có thể xoá các kí tự thứ 2, 3, 4 và 6 từ dãy s để thu được dãy con đối xứng khác là bdadb với cùng chiều dài 5:

bae**ad**bad**b** \rightarrow bdadb

Tuy nhiên đáp số là số ít nhất các kí tự cần loại bỏ khỏi s thì là duy nhất và bằng **4**.

Bài giải

Bài toán này đã được nhiều bạn đọc công bố lời giải với một mảng hai chiều kích thước n^2 hoặc vài ba mảng một chiều kích thước n , trong đó n là chiều dài của dữ liệu vào.

Với một nhận xét nhỏ ta có thể phát hiện ra rằng chỉ cần dùng một mảng một chiều kích thước n và một vài biến đơn là đủ.

Gọi dãy dữ liệu vào là s . Ta tìm chiều dài của dãy con đối xứng v dài nhất trích từ s . Khi đó số kí tự cần xoá từ s sẽ là $t = \text{length}(s) - \text{length}(v)$. Dãy con ở đây được hiểu là dãy thu được từ s bằng cách xoá đi một số phần tử trong s . Thí dụ với dãy $s = \text{baeadbadb}$ thì dãy con đối xứng dài *nhất* của s sẽ là baeab hoặc bdadb, \dots Các dãy này đều có chiều dài 5.

Lập hệ thức: Gọi $p(i, j)$ là chiều dài của dãy con dài nhất thu được khi giải bài toán với dữ liệu vào là đoạn $s[i..j]$. Khi đó $p(1, n)$ là chiều dài của dãy con đối xứng dài nhất trong dãy n kí tự $s[1..n]$ và do đó số kí tự cần loại bỏ khỏi dãy $s[1..n]$ sẽ là

$$n - p(1, n)$$

Đó chính là đáp số của bài toán.

Ta liệt kê một số tính chất quan trọng của hàm hai biến $p(i, j)$. Ta có:

- Nếu $i > j$, tức là chỉ số đầu trái lớn hơn chỉ số đầu phải, ta quy ước đặt $p(i, j) = 0$.
- Nếu $i = j$ thì $p(i, i) = 1$ vì dãy khảo sát chỉ chứa đúng 1 kí tự nên nó là đối xứng.
- Nếu $i < j$ và $s[i] = s[j]$ thì $p(i, j) = p(i + 1, j - 1) + 2$. Vì hai kí tự đầu và cuối dãy $s[i..j]$ giống nhau nên chỉ cần xác định chiều dài của dãy con đối xứng dài nhất trong đoạn giữa là $s[i + 1, j - 1]$ rồi cộng thêm 2 đơn vị ứng với hai kí tự đầu và cuối dãy là được.
- Nếu $i < j$ và $s[i] \neq s[j]$, tức là hai kí tự đầu và cuối của dãy con $s[i..j]$ là khác nhau thì ta khảo sát hai dãy con là $s[i..(j - 1)]$ và $s[(i + 1)..j]$ để lấy chiều dài của dãy con đối xứng dài nhất trong hai dãy này làm kết quả:

$$p(i, j) = \max(p(i, j - 1), p(i + 1, j))$$

Vấn đề đặt ra là cần tính $p(1, n)$. Mà muốn tính được $p(1, n)$ ta phải tính được các $p(i, j)$ với mọi $i, j = 1..n$.

Phương án đệ quy

Phương án đệ quy dưới đây như mô tả trong hàm nguyên **rec(i, j)** tính trực tiếp giá trị **p(i, j)** theo các tính chất đã liệt kê. Đáp số cho bài toán khi đó sẽ là **n-rec(1, n)**

```
(*-----
                        Phuong an de quy
-----*)
function rec(i, j: integer): integer;
begin
  if i > j then rec := 0
  else if i = j then rec := 1
  else {i < j}
    if s[i] = s[j]
    then rec := rec(i+1, j-1)+2
    else {i < j & s[i] ≠ s[j]}
      rec := max(rec(i, j-1), rec(i+1, j));
```

end;

		j-1	j				b	a	e	a	d	b	a	d	b
							❶	❷	❸	❹	❺	❻	❼	❽	❾
					b	❶	1	1	1	3	3	5	5	5	5
					a	❷	0	1	1	3	3	3	3	3	3
i		[i, j-1]	[i, j]		e	❸	0	0	1	1	1	1	3	3	3
i+1		[i+1, j-1]	[i+1, j]		a	❹	0	0	0	1	1	1	3	3	3
					d	❺	0	0	0	0	1	1	1	3	3
					b	❻	0	0	0	0	0	1	1	1	3
					a	❼	0	0	0	0	0	0	1	1	1
					d	❽	0	0	0	0	0	0	0	1	1
					b	❾	0	0	0	0	0	0	0	0	1

Gia trị của hàm $p(i,j)$ đối với dãy baeadbadb
 $i,j=1..9$

Dùng một mảng hai chiều

Gọi đệ quy sẽ phát sinh các lời gọi hàm trùng lặp như đã phân tích trong bài toán 7.1. Ta khắc phục điều này bằng cách sử dụng một mảng hai chiều để tính trước các giá trị của hàm $p(i, j)$, mỗi giá trị được tính tối đa một lần. Nếu dùng một mảng hai chiều, thí dụ mảng $p[0..n, 0..n]$ thì giá trị của $p[i, j]$ sẽ được điền lần lượt theo từng cột, từ cột thứ 1 đến cột thứ n . Tại mỗi cột ta điền từ dưới lên trên. Ta lưu ý:

- Phần tử tại cột i , dòng j là giá trị $p[i, j]$ chính là chiều dài của dãy con đối xứng dài nhất khi khảo sát dãy con $s[i..j]$.
- Với các trị $i > j$, ta quy định $p[i, j] = 0$. Như vậy nửa tam giác dưới của ma trận p sẽ chứa toàn 0.
- Nếu $i = j$ thì $p[i, j] = 1$. Như vậy, mọi trị trên đường chéo chính của ma trận p sẽ là 1.
- Với các ô còn lại, tọa độ (i, j) sẽ thỏa điều kiện $i < j$, nên $p[i, j]$ sẽ được tính như sau:

```
if s[i] = s[j] then p[i,j] = p[i+1,j-1]+2
else p[i,j] := max(p[i,j-1], p[i+1,j])
```

Bạn hãy thử điền một vài giá trị cho bảng trên để rút ra quy luật.

Hãy bắt đầu với cột 1: $p[1, 1] = 0$;

Sau đó đến cột 2:

$p[2, 2] = 1$; $p[1, 2] = \max(p[1, 1], p[2, 2]) = 1$, vì $s[1] \neq s[2]$.

Rồi đến cột 3:

$p[3, 3] = 1$; $p[2, 3] = \max(p[2, 2], p[3, 3]) = 1$, vì $s[2] \neq s[3]$;

$p[1, 3] = \max(p[1, 2], p[2, 3]) = 1$, vì $s[1] \neq s[3]$,...

Dùng hai mảng một chiều

Ta sẽ không theo đuổi phương án dùng mảng hai chiều mà hãy căn cứ vào quy luật điền mảng hai chiều để vận dụng cho hai mảng một chiều là $v[0..(n+1)]$ và $d[0..(n+1)]$. Theo kinh nghiệm, ta nên khai báo kích thước mảng rộng hơn chừng hai phần tử để sử dụng các phần tử này như những lính canh chứa các giá trị khởi đầu phục vụ cho các trường hợp chỉ số i, j nhận các giá trị 0 hoặc $n+1$.

Giả sử mảng v chứa các giá trị đã điền của cột $j-1$ trong mảng hai chiều p . Ta sẽ điền các giá trị cho cột j của mảng p vào mảng một chiều d . Như vậy, tại bước j , phần tử $v[i]$ sẽ ứng với phần tử $p[j-1, i]$ còn phần tử $d[i]$ sẽ ứng với $p[j, i]$. Thủ tục điền trị cho cột d tại bước j dựa theo kết quả lưu trong cột v của bước $j-1$ khi đó sẽ như sau:

```
for i := j-1 downto 1 do
begin
  if s[i] = s[j] then d[i] := v[i+1]+2
  else d[i] := max(v[i], d[i+1]);
end;
```

Sau mỗi lần lặp với $j := 1..n$ ta chuyển giá trị của d cho v để chuẩn bị cho bước tiếp theo.

```
(*-----
      Quy hoạch động với 2 mảng
      1 chiều d, v
-----*)
procedure QHD2;
var i, j: integer;
begin
  fillchar(v, sizeof(v), 0);
  for j := 1 to n do
    begin
      d[j] := 1;
      for i := j-1 downto 1 do
        begin
          if s[i] = s[j] then d[i] := v[i+1]+2
          else d[i] := max(v[i], d[i+1]);
        end;
      v := d;
    end;
  writeln(n1, n-d[1]); {dap so}
end;
```

Dùng một mảng một chiều

Có thể chỉ sử dụng một mảng một chiều d cho bài toán này với nhận xét sau đây. Tại bước cập nhật thứ j , với mỗi $i = (j-1)..1$ ta có $d[i] = p[i, j]$ và được tính như sau:

- ♦ Nếu $s[i] = s[j]$ thì $d[i]$ tại bước j bằng $d[i+1]$ tại bước $j-1$ cộng với 2.
- ♦ Nếu $s[i] \neq s[j]$ thì
 $d[i]$ tại bước j bằng $\max(d[i]$ tại bước $j-1, d[i+1]$ tại bước $j)$.

Nếu ta tính từ dưới lên, tức là tính $d[i]$ với $i = n..1$ thì $d[i+1]$ cũ sẽ bị ghi đè. Ta dùng hai biến phụ t và tr để bảo lưu giá trị này.

```
(*-----
      Quy hoạch động với mảng 1 chiều d
-----*)
procedure QHD1;
var i, j, t, tr: integer;
```

```

begin
  for j := 1 to n do
    begin
      tr := 0;
      d[j] := 1;
      for i := j-1 downto 1 do
        begin
          t := d[i];
          if s[i]=s[j] then d[i] := tr+2
            else d[i] := max(d[i],d[i+1]);
          tr := t;
        end;
      end;
      writeln(nl,n-d[1]); {dap so}
    end;
  end;

```

Dĩ nhiên phương án dùng một mảng một chiều sẽ được coi trọng vì tiết kiệm được miền nhớ. Tuy nhiên, tính ý một chút, bạn sẽ phát hiện ra rằng thời gian tính toán theo phương án này không ít hơn phương án dùng hai mảng một chiều. Thật vậy, để tính mỗi phần tử ta phải dùng thêm hai phép gán, trong khi dùng hai mảng một chiều ta chỉ phải thêm một phép gán cho mỗi phần tử. Hơn nữa, dùng hai mảng một chiều thường tránh được nhầm lẫn, do đó nhiều người thường chọn phương án này.

Toàn văn chương trình với ba phương án, đệ quy, dùng hai mảng một chiều và dùng một mảng một chiều khi đó sẽ như sau.

```

(* Pascal *)
uses crt;
const mn = 51;
  bl = #32; nl = #13#10;
  fn = 'palin.inp';
  gn = 'palin.out';
type ml1 = array[0..mn] of integer;
  ml2 = array[0..mn] of ml1;
  mc1 = array[0..mn] of char;
var n: integer; { Chiều dài xâu }
  f,g: text;
  s: mc1; { xâu cần xử lý }
  d,v: ml1;
  c: ml2;
procedure Doc; tự viết
function Max(a,b: integer): integer; tự viết
(*-----
          Phương án đệ quy
-----*)
function rec(i,j: integer): integer; tự viết
(*-----
          Quy hoạch động với mảng 2 chiều c
-----*)
function QHD2C: integer; tự viết
(*-----
          Quy hoạch động với 2 mảng
          1 chiều d, v

```

```

-----*)
function QHD2DV: integer; tự viết
(*-----
  Quy hoạch động với mảng 1 chiều d
-----*)
function QHD1: integer; tự viết
procedure Test;
begin
  Doc;
  writeln(nl, 'Phuong an 1: De qui: ', n-rec(1,n));
  writeln(nl, 'Phuong an 2: Mang 2 chieu: ', n-QHD2C);
  writeln(nl, 'Phuong an 3: Hai Mang 1 chieu D, V: ', n-
QHD2DV);
  writeln(nl, 'Phuong an 4: Mang 1 chieu D: ', n-QHD1);
end;
BEGIN
  Test;
  readln;
END.

```

// C#

```

using System;
using System.IO;
namespace SangTaoT1
{
    /*-----
    *                      Palindrome
    * -----*/
    class Palin
    {
        static string fn = "palin.inp";
        static string gn = "palin.out";
        static string s;
        static int n = 0;
        static void Main()
        {
            Doc();
            File.WriteAllText(gn, XuLi().ToString());
            // Doc lai de kiem tra
            Console.WriteLine("\n Input file " + fn);
            Console.WriteLine(File.ReadAllText(fn));
            Console.WriteLine("\n Output file " + gn);
            Console.WriteLine(" So ki tu can xoa: " +
                File.ReadAllText(gn));
            Console.ReadLine();
        }
        static void Doc()
        {
            StreamReader f = File.OpenText(fn);
            n = int.Parse((f.ReadLine()).Trim());
            s = (f.ReadLine()).Trim();
        }
    }
}

```

```

        f.Close();
    }
    static int XuLi()
    {
        int[] d = new int[n + 1];
        int tr = 0;
        int t = 0;
        for (int j = 0; j < n; ++j)
        {
            tr = 0;
            d[j] = 1;
            for (int i = j - 1; i >= 0; --i)
            {
                t = d[i];
                d[i] = (s[i]==s[j])?(tr+2)
                        :Max(d[i],d[i+1]);
                tr = t;
            }
        }
        return n - d[0];
    }
    static int Max(int a, int b)
    {
        return (a > b) ? a : b;
    }
} // Palin
} // SangTao1

```

Bài 7.3. Cắm hoa

Olympic Quốc tế năm 1999.

Cần cắm hết k bó hoa khác nhau vào n lọ xếp thẳng hàng sao cho bó hoa có số hiệu nhỏ được đặt trước bó hoa có số hiệu lớn. Với mỗi bó hoa i ta biết giá trị thẩm mỹ khi cắm bó hoa đó vào lọ j là $v[i, j]$.

Yêu cầu: Xác định một phương án cắm hoa sao cho tổng giá trị thẩm mỹ là lớn nhất.

Dữ liệu vào ghi trong tệp văn bản **HOA.INP**:

- Dòng đầu tiên là hai trị k và n .
- Từ dòng thứ hai trở đi là các giá trị $v[i, j]$ trong khoảng $0..10$, với $i = 1..k$ và $j = 1..n$; $1 \leq k \leq n \leq 100$.

Dữ liệu ra ghi trong tệp văn bản **HOA.OUT**: dòng đầu tiên là tổng giá trị thẩm mỹ của phương án cắm hoa tối ưu. Từ dòng thứ hai là dãy k số hiệu lọ được chọn cho mỗi bó hoa.

Các số liệu vào và ra đều là số tự nhiên và được ghi cách nhau bởi dấu cách trên mỗi dòng.

Thí dụ:

	HOA.INP	HOA.OUT
4	6	24
<u>1</u>	1 6 4 3 10	1 3 4 6
9	1 <u>4</u> 7 2 7	
7	2 6 <u>10</u> 2 3	

6 10 7 1 3 9

Kết quả cho biết tổng giá trị thẩm mỹ sẽ đạt là 24 (điểm) nếu cắm hoa như sau:

- Bó hoa 2 cắm vào lọ 3;
- Bó hoa 3 cắm vào lọ 4;
- Bó hoa 4 cắm vào lọ 6.

Bài giải

Trước hết ta đọc dữ liệu từ tệp **HOA.INP** vào các biến k , n và $v[i, j]$.

```
(*-----*)
                        Doc du lieu
-----*)
procedure doc;
var i,j:byte;
begin
  assign(f,fn);
  reset(f);
  readln (f,k,n);
  for i := 1 to k do
    for j := 1 to n do
      read(f,v[i,j]);
  close(f);
end;
Các hằng và biến được khai báo như sau:
const
  fn = 'hoa.inp'; {File du lieu vao }
  gn = 'hoa.out'; {File du lieu ra }
  mn = 101; {So luong toi da cac lo hoa: 100 }
  bl = #32; {Dau cach }
  nl = #13#10; {Xuong dong }
  kk = (mn+7) div 8; {So bit danh dau cac lo hoa }
type
  mb1 = array[0..mn] of byte; {mang byte 1 chieu }
  mb2 = array[0..mn] of mb1; {mang byte 2 chieu }
  ml1 = array[0..kk] of byte;
  ml2 = array[0..mn] of ml1;
  mil = array[0..mn] of integer;
var
  n,k: byte; {n - so luong lo, k - so luong bo hoa }
  v: mb2;
  {v[i,j] - do tham my khi cam bo hoa i vao lo j }
  L: ml2;
  {cac mang danh dau lo hoa
    bit(i) = 1: lo hoa duoc chon
    bit(i) = 0: lo hoa roi}
  T: mil; {T[i,j]: tong so do tham mi
    khi cam i bo hoa vao day j lo }
  f,g: text; {files input va output }
```

1. **Lập hệ thức:** Gọi $T(i, j)$ là tổng giá trị thẩm mỹ khi giải bài toán với i bó hoa mã số 1.. i và j lọ mã số 1.. j , tức là độ thẩm mỹ thu được khi cắm hết i bó hoa đầu tiên vào j lọ đầu tiên, ta thấy:

- a) Nếu số bó hoa nhiều hơn số lọ, $i > j$ thì không có cách cắm nào vì đầu bài yêu cầu phải cắm hết các bó hoa, mỗi bó vào đúng 1 lọ.

$$T(i, j) = 0$$

- b) Nếu số bó hoa bằng số lọ ($i = j$) thì chỉ có một cách cắm là bó nào vào lọ đó.

- c) Ta xét trường hợp số bó hoa ít hơn hẳn số lọ ($i < j$). Có hai tình huống: lọ cuối cùng, tức lọ thứ j được chọn cho phương án tối ưu và lọ thứ j không được chọn.

- Nếu lọ cuối cùng, lọ thứ j được chọn để cắm bó hoa (cuối cùng) i thì $i - 1$ bó hoa đầu tiên sẽ được phân phối vào $j - 1$ lọ đầu tiên. Tổng giá trị thẩm mỹ s khi đó sẽ là $T(i - 1, j - 1) + v[i, j]$.

- Nếu lọ thứ j không được chọn cho phương án tối ưu thì i bó hoa phải được cắm vào $j - 1$ lọ đầu tiên và do đó tổng giá trị thẩm mỹ sẽ là $T(i, j - 1)$.

Tổng hợp lại ta có giá trị tối ưu khi cắm i bó hoa vào j lọ là:

$$T(i, j) = \max \{ T(i - 1, j - 1) + v[i, j], T(i, j - 1) \}$$

2. **Tổ chức dữ liệu và chương trình:** Nếu dùng mảng hai chiều T thì ta có thể tính như trong bảng dưới đây:

	...	$j - 1$	j	
...		
$i - 1$...	$[i - 1, j - 1]$		
i	...	$[i, j - 1]$	$[i, j] ?$	
...		
$T(i, j) = \max \{ T(i - 1, j - 1) + v[i, j], T(i, j - 1) \}$				

Ngoài ra, ta còn cần đặt trong mỗi ô của bảng trên một mảng dữ liệu bao gồm n phần tử để đánh dấu lọ hoa nào được chọn cho mỗi tình huống. Gọi mảng dữ liệu đó là $L[i, j]$, ta dễ thấy là nên điền bảng lần lượt theo từng cột, tại mỗi cột ta điền bảng từ dưới lên theo luật sau:

- Nếu $T[i - 1, j - 1] + v[i, j] > T[i, j - 1]$ thì ta phải thực hiện hai thao tác:

- o Đặt lại trị $T[i, j] := T[i - 1, j - 1] + v[i, j]$.
- o Ghi nhận việc chọn lọ hoa j trong phương án mới, cụ thể lấy phương án cắm hoa $(i - 1, j - 1)$ rồi bổ sung thêm thông tin chọn lọ hoa j như sau: đặt $L[i, j] := L[i - 1, j - 1]$ và đánh dấu phần tử j trong mảng $L[i, j]$.
- Nếu $T[i - 1, j - 1] + v[i, j] \leq T[i, j - 1]$ thì ta sẽ không chọn lọ j để cắm bó hoa i và do đó chỉ cần copy $L[i, j - 1]$ sang $L[i, j]$, tức là ta bảo lưu phương án $(i, j - 1)$.

3. **Làm tốt.** Phương án dùng mảng hai chiều tốn kém về miền nhớ. Ta có thể dùng một mảng một chiều $T[0..100]$ xem như một cột của bảng T nói trên. Ta duyệt j bước. Tại bước thứ j , giá trị $T[i]$ sẽ là trị tối ưu khi cắm hết i bó hoa vào j lọ. Như vậy, tại bước thứ j ta có:

- Nếu $T[i - 1]$ tại bước $j - 1 + v[i, j] > T[i]$ tại bước $j - 1$ thì ta phải thực hiện hai thao tác:

- o Đặt lại trị $T[i]$ tại bước $j := T[i - 1]$ tại bước $j - 1 + v[i, j]$.
- o Ghi nhận việc chọn lọ hoa j trong phương án mới, cụ thể lấy phương án cắm hoa $(i - 1)$ ở bước $j - 1$ rồi bổ sung thêm thông tin chọn lọ hoa j như sau: đặt $L[i]$ tại bước $j := L[i - 1]$ tại bước $j - 1$ và đánh dấu phần tử j trong mảng $L[i]$.
- Nếu $T[i - 1]$ tại bước $j - 1 + v[i, j] \leq T[i]$ tại bước $j - 1$ thì ta không phải làm gì vì sẽ bảo lưu phương án cũ.

Biểu thức so sánh cho biết khi cập nhật mảng T từ bước thứ $j - 1$ qua bước thứ j ta phải tính từ dưới lên, nghĩa là tính dần theo chiều giảm của $i := j..1$.

Để đánh dấu các lọ hoa ta dùng mảng $L[0..MN]$ mỗi phần tử $L[i]$ lại là một dãy s byte. Nếu dùng một bit cho mỗi lọ hoa thì số byte cần dùng để đánh dấu tối đa MN lọ hoa phải là:

$$kk = (MN+7) \text{ div } 8$$

Với $MN = 101$ ta tính được

$$kk = (101+7) \text{ div } 8 = 13$$

Khi cần đánh dấu lọ hoa thứ j trong dãy $L[i]$ ta bật bit thứ j trong $L[i]$. Khi cần xem lọ thứ j có được chọn hay không ta gọi hàm **GetBit** để lấy trị (0 hoặc 1) của bit j trong dãy bit $L[i]$.

Ta chú ý tới hai biểu thức sau:

- Để xác định byte thứ mấy trong dãy chứa bit j ta tính:

$$b := j \text{ div } 8;$$

- Để xác định vị trí của bit j trong byte thứ b ta tính:

$$p := j \text{ mod } 8;$$

```
(* -----
  Cho gia tri bit thu j trong day byte L[i]
-----*)
function getbit(i,j: byte):byte;
var b,p: byte;
begin
  b := j div 8;
  p := j mod 8;
  getbit := (L[i][b] shr p) and 1;
end;

(* -----
  Gan tri 1 cho bit j trong day byte L[i]
-----*)
procedure batbit(i,j:byte);
var b,p: byte;
begin
  b := j shr 3;
  p := j and 7;
  L[i][b] := L[i][b] or (1 shl p);
end;
```

Với $j = 0$, tức là khi không có lọ nào và không có bó hoa nào ta khởi trị:

```
fillchar(L[0],16,0);
```

```
T[0] := 0;
```

Với mỗi $j = 1..n$, ta lưu ý số bó hoa phải không lớn hơn số lọ, tức là $i \leq j$. Với $i = j$ ta sẽ cắm mỗi bó vào một lọ. Để thực hiện điều này ta lưu ý rằng phần tử $L[j-1]$ tại bước trước đã cho biết $j-1$ lọ đều có hoa do đó ta chỉ cần đánh dấu lọ thứ j cho bước j :

```
L[j] := L[j-1];
```

```
batbit(j,j);
```

Như vậy ta cần chia quá trình duyệt theo các lọ hoa từ $1..n$ thành hai giai đoạn.

Giai đoạn 1: Duyệt từ lọ 1 đến lọ k , trong đó k chính là số bó hoa và theo đầu bài, $k \leq n$.

Giai đoạn 2: Duyệt nốt $n - k$ lọ hoa còn lại.

Phương án quy hoạch động với mảng một chiều khi đó sẽ như sau:

```
(*-----
               Quy hoạch dong
-----*)
procedure xuly;
var i,j: byte;
begin
  {1. Khoi tri }
  fillchar(L,sizeof(L),0);
  {danh dau cac lo hoa duoc chon }
  T[0] := 0; {do tham mi }
  {Vi co k bo hoa nen xet k lo dau tien }
  for j := 1 to k do
    begin
      L[j] := L[j-1];
      batbit(j,j);
      T[j] := T[j-1]+v[j,j];
      for i := j-1 downto 1 do
        if T[i] < T[i-1]+v[i,j] then
          begin
            T[i] := T[i-1]+v[i,j];
            L[i] := L[i-1];
            batbit(i,j);
          end;
        end;
      {xet cac lo con lai }
      for j := k+1 to n do
        for i := k downto 1 do
          if T[i] < T[i-1]+v[i,j] then
            begin
              T[i] := T[i-1]+v[i,j];
              L[i] := L[i-1];
              batbit(i,j);
            end;
          end;
    end;
end;

(* Pascal *)
(*=====
               Hoa.pas: Quy hoạch dong
=====*)
uses crt ;
const
  fn = 'hoa.inp'; {File du lieu vao }
  gn = 'hoa.out'; {File du lieu ra }
  mn = 101; {So luong toi da cac lo hoa: 100 }
  bl = #32; {Dau cach }
  nl = #13#10; {Xuong dong }
  kk = (mn+7) div 8; {So bit danh dau cac lo hoa }
type
  mb1 = array[0..mn] of byte; {mang byte 1 chieu }
```

```

mb2 = array[0..mn] of mb1; {mang byte 2 chieu }
ml1 = array[0..kk] of byte;
ml2 = array[0..mn] of ml1;
mil = array[0..mn] of integer;
var
  n,k: byte; {n - so luong lo, k - so luong bo hoa }
  v: mb2;
  {v[i,j] - do tham my khi cam bo hoa i vao lo j }
  L: ml2;
  {cac mang danh dau lo hoa
    bit(i) = 1: lo hoa duoc chon
    bit(i) = 0: lo hoa roi }

  T: mil;
  {T[i,j] tong do tham my khi cam i bo hoa vao day j
  lo }
  f,g: text; {files input va output }
  (*-----
    Doc du lieu
    -----*)
procedure doc; tự viết
(* -----
Cho gia tri bit thu j trong day byte L[i]
-----*)
function getbit(i,j: byte):byte; tự viết
(* -----
Gan tri 1 cho bit j trong day byte L[i]
-----*)
procedure batbit(i,j:byte); tự viết
(*-----
    Quy hoạch dong
    -----*)
procedure xuly; tự viết
(*-----
Ghi ket qua T[k] -
Tong do tham mi cac lo duoc chon
-----*)
procedure ghi; tự viết
BEGIN
  doc; xuly; ghi;
END.

```

// C#

```

using System;
using System.IO;

namespace SangTaoTl
{
  /*-----
   *           Cam hoa
   * -----*/
  class CamHoa
  {

```

```

const string fn = "hoa.inp";
const string gn = "hoa.out";
static int k = 0; // so hoa
static int n = 0; // so lo
// v[i,j] = do tham mi khi cam
// hoa i vao lo j
static int[,] v;
static void Main()
{
    Run();
    Console.WriteLine("\n Fini");
    Console.ReadLine();
} // Main
static void Run()
{
    Doc(); Show();
    DayLo Lo = new DayLo(n + 2);
    int Vmax = XuLi(Lo);
    Ghi(Vmax, Lo); KiemTra();
}
// Ghi file
static void Ghi(int Vmax, DayLo d)
{
    StreamWriter g = File.CreateText(gn);
    g.WriteLine(Vmax);
    for (int i = 1; i <= n; ++i)
        if (d.Bit(i) > 0) g.Write(i + " ");
    g.Close();
}
// Doc lai file gn de kiem tra
static void KiemTra() tự viết
// Dynamic Programming
// T(i,j) max tham mi voi i bo hoa va j lo
// T(i,j) = max {T(i-1,j-1)+v[i,j],T(i,j-1)}
// T(0,0) = T(i,0)= T(0,j) = 0
// Tinh theo cot, duoi len
static int XuLi(DayLo d)
{
    if (k > n) return 0;
    int[] t = new int[k + 2];
    DayLo[] Lo = new DayLo[k + 2];
    for (int i = 0; i <= k + 1; ++i)
        Lo[i] = new DayLo(n + 2);
    Array.Clear(t, 0, t.Length);
    // xet k hoa va k lo
    for (int j = 1; j <= k; ++j) // lo
    { // so hoa i <= so lo j
        for (int i = j; i > 0; --i) // hoa
            if (t[i - 1] + v[i, j] > t[i])
            {
                t[i] = t[i - 1] + v[i, j];
                Lo[i - 1].CopyTo(Lo[i]);
                Lo[i].BatBit(j);
            }
        }
    }
}

```

```

    }
}
// xet cac lo con lai
for (int j = k + 1; j <= n; ++j)
{
    for (int i = k; i > 0; --i)
        if (t[i - 1] + v[i, j] > t[i])
        {
            t[i] = t[i - 1] + v[i, j];
            Lo[i - 1].CopyTo(Lo[i]);
            Lo[i].BatBit(j);
        }
    }
    Lo[k].CopyTo(d);
    return t[k];
}
static void Doc()
{
    int[] a = Array.ConvertAll((File.
        ReadAllText(fn)).Split(
        new char[] { '\0', '\n', '\t', '\r', ' ' },
        StringSplitOptions.RemoveEmptyEntries),
        new Converter<String, int>(int.Parse));
    k = a[0]; // so hoa
    n = a[1]; // so lo
    v = new int[k + 2, n + 2];
    int i = 2;
    for (int d = 1; d <= k; ++d)
        for (int c = 1; c <= n; ++c)
            v[d, c] = a[i++];
    // dien 0 vao cac vi tri con lai
    for (int j = 0; j <= n; ++j)
        v[0, j] = v[k + 1, j] = 0;
}
// Hien thi du lieu
static void Show() tự viết
} // CamHoa
// The hien 1 day lo
class DayLo
{
    public const int bitnum = 32;
    // so bit cho 1 bien int = 32 = 2^5
    public int Size;
    public uint[] Data;
    public DayLo(int n) // day 0/1 gom n lo hoa
    {
        Size = (n + bitnum - 1) / bitnum;
        Data = new uint[Size];
        for (int i = 0; i < Size; ++i)
            Data[i] = (uint)0;
    }
    // Gan tri 1 cho bit i trong day lo
    // i >> 5 = i / 2^5 = i / 32

```

```

public void BatBit(int i)
{
    Data[i >> 5] |= ((uint)1) <<
                    (i & (bitnum-1));
}
// Gán trị 0 cho bit i trong day lo
public void TatBit(int i)
{
    Data[i>>5] &= ~(((uint)1)<<(i&(bitnum-1))));
}
// Lấy trị của bit i trong day lo
public int Bit(int i)
{
    return (int)((Data[i>>5]>>
                    (i&(bitnum-1)))&((uint)1));
}
// CopyTo DayLo this sang DayLo y
public void CopyTo(DayLo y)
{
    int len = (Size <= y.Size) ? Size :
y.Size;
    for (int i = 0; i < len; ++i) y.Data[i]
        = Data[i];
}
} // DayLo
} // SangTao1

```

Bài toán sau đây là một cách phát biểu khác của bài toán cắm hoa:

Bài toán

Câu lạc bộ - Học sinh giỏi Tin học, Hà Nội, năm 2000

Cần bố trí k nhóm học sinh vào k trong số n phòng học chuyên đề sao cho nhóm có số hiệu nhỏ được xếp vào phòng có số hiệu nhỏ hơn phòng chứa nhóm có số hiệu lớn. Với mỗi phòng có nhận học sinh, các ghế thừa phải được chuyển ra hết, nếu thiếu ghế thì phải lấy từ kho vào cho đủ mỗi học sinh một ghế. Biết số học sinh trong mỗi nhóm và số ghế trong mỗi phòng. Hãy chọn phương án bố trí sao cho tổng số lần chuyển ghế ra và chuyển ghế vào là ít nhất.

Bài 7.4. Tìm các đường ngắn nhất

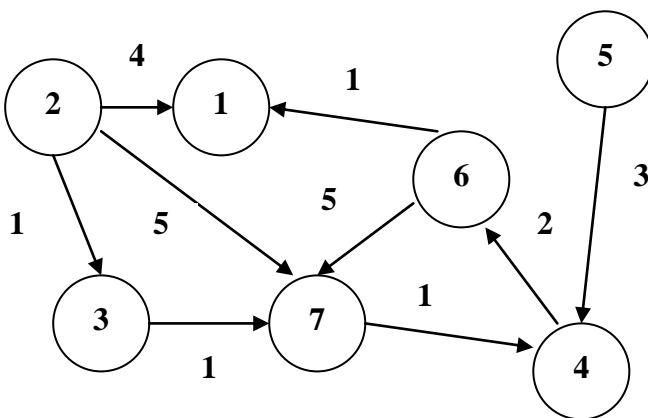
Cho một đồ thị có hướng gồm n đỉnh mã số từ $1..n$ với các cung (u, v) có hướng đi từ đỉnh u đến đỉnh v và có chiều dài thể hiện đường đi nối từ đỉnh u đến đỉnh v . Viết chương trình tìm mọi đường đi ngắn nhất từ một đỉnh s cho trước tới các đỉnh còn lại của đồ thị.

Dữ liệu vào được ghi trong một tệp văn bản tên **DIJ.INP** có cấu trúc như sau:

- Dòng đầu ghi hai số tự nhiên n và s cách nhau bởi dấu cách, trong đó n là số lượng đỉnh của đồ thị, s là số hiệu của đỉnh xuất phát.
- Từ dòng thứ hai ghi lần lượt độ dài đường đi từ đỉnh i đến các đỉnh $1, 2, \dots, n$; $i = 1..n$. Giá trị 0 cho biết không có cung nối hai đỉnh tương ứng. Với mọi đỉnh $i = 1..n$, cung (i, i) được xem là không tồn tại và ghi chiều dài là 0. Các số cùng dòng cách nhau qua dấu cách. Dạng dữ liệu cho như vậy được gọi là ma trận kề của đồ thị.

Thí dụ sau đây cho biết đồ thị có bảy đỉnh, cần tìm các đường đi ngắn nhất từ đỉnh 2 tới các đỉnh còn lại của đồ thị. Cung (2, 1) có chiều dài 4,...

DIJ . INP								
7	2							
0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	5	
0	0	0	0	0	0	0	1	
0	0	0	0	0	0	2	0	
0	0	0	3	0	0	0	0	
1	0	0	0	0	0	0	5	
0	0	0	1	0	0	0	0	



Dữ liệu ra được ghi trong tệp văn bản **DIJ . OUT** gồm n dòng. Thông tin về mỗi đường đi ngắn nhất từ đỉnh s đến các đỉnh còn lại được ghi trên 1 dòng. Số đầu tiên của dòng là chiều dài đường đi. Nếu không tồn tại đường đi thì ghi giá trị 0. Tiếp đến, trong trường hợp có đường đi từ đỉnh s đến đỉnh i thì ghi dãy đỉnh xuất hiện lần lượt trên đường đi, đỉnh đầu tiên, dĩ nhiên là s , đỉnh cuối cùng là i . Đường đi từ đỉnh i tới chính đỉnh đó được coi là không tồn tại, $i = 1..n$. Thí dụ trên cho ta kết quả

- Đường ngắn nhất từ đỉnh 2 đến đỉnh 1 có chiều dài 4, cách đi: $2 \rightarrow 1$.
- Đường ngắn nhất từ đỉnh 2 đến đỉnh 2: không có (thực ra, theo lẽ thường là có đường chiều dài 0).
- Đường ngắn nhất từ đỉnh 2 đến đỉnh 3 có chiều dài 1, cách đi: $2 \rightarrow 3$.
- Đường ngắn nhất từ đỉnh 2 đến đỉnh 4 có chiều dài 3, cách đi: $2 \rightarrow 3 \rightarrow 7 \rightarrow 4$.
- Đường ngắn nhất từ đỉnh 2 đến đỉnh 5: không có.
- Đường ngắn nhất từ đỉnh 2 đến đỉnh 6 có chiều dài 5, cách đi: $2 \rightarrow 3 \rightarrow 7 \rightarrow 4 \rightarrow 6$.
- Đường ngắn nhất từ đỉnh 2 đến đỉnh 7 có chiều dài 2, cách đi: $2 \rightarrow 3 \rightarrow 7$.

DIJ . OUT				
4	2	1		
0				
1	2	3		
3	2	3	7	4
0				
5	2	3	7	4
2	2	3	7	

Bài giải

Thuật giải quy hoạch động được trình bày dưới đây mang tên Dijkstra, một nhà tin học lỗi lạc người Hà Lan. Bản chất của thuật toán là sửa định, chính xác ra là sửa trọng số của mỗi đỉnh.

Theo sơ đồ giải các bài toán quy hoạch động trước hết ta xây dựng hệ thức cho bài toán.

Gọi $p(i)$ là độ dài đường ngắn nhất từ đỉnh s đến đỉnh i , $1 \leq i \leq n$. Ta thấy, hàm $p(i)$ phải thoả các tính chất sau:

- $p(s) = 0$: đường ngắn nhất từ đỉnh xuất phát s đến chính đỉnh đó có chiều dài 0.
- Với $i \neq s$, muốn đến được đỉnh i ta phải đến được một trong các đỉnh sát trước đỉnh i . Nếu j là một đỉnh sát trước đỉnh i , theo điều kiện của đầu bài ta phải có

$$a[j, i] > 0$$

trong đó $a[j, i]$ chính là chiều dài cung ($j \rightarrow i$).

Trong số các đỉnh j sát trước đỉnh i ta cần chọn đỉnh nào?

Kí hiệu $\text{path}(x, y)$ là đường đi ngắn nhất qua các đỉnh, xuất phát từ đỉnh x và kết thúc tại đỉnh $y \neq x$. Khi đó đường từ s đến i sẽ được chia làm hai đoạn, đường từ s đến j và cung $(j \rightarrow i)$:

$$\text{path}(s, i) = \text{path}(s, j) + \text{path}(j, i)$$

trong đó $\text{path}(j, i)$ chỉ gồm một cung:

$$\text{path}(j, i) = (j \rightarrow i)$$

Do $p(i)$ và $p(j)$ phải là ngắn nhất, tức là phải đạt các trị \min , ta suy ra điều kiện để chọn đỉnh j sát trước đỉnh i là tổng chiều dài đường từ s đến j và chiều dài cung $(j \rightarrow i)$ là ngắn nhất. Ta thu được hệ thức sau:

$$p(i) = \min \{p(j) + a[j, i] \mid a[j, i] > 0, j = 1..n\}$$

Để ý rằng điều kiện $a[j, i] > 0$ cho biết j là đỉnh sát trước đỉnh i .

Điều tài tình là Dijkstra đã cung cấp thuật toán tính đồng thời mọi đường đi ngắn nhất từ đỉnh s đến các đỉnh còn lại của đồ thị. Thuật toán đó như sau.

Thuật toán thực hiện n lần lặp, mỗi lần lặp ta chọn và xử lý 1 đỉnh của đồ thị. Tại lần lặp thứ k ta khảo sát phần của đồ thị gồm k đỉnh với các cung liên quan đến k đỉnh được chọn trong phần đồ thị đó. Ta gọi phần này là đồ thị con thu được tại bước xử lý thứ k của đồ thị ban đầu và kí hiệu là $G(k)$. Với đồ thị này ta hoàn tất bài giải tìm mọi đường đi ngắn nhất từ đỉnh xuất phát s đến mọi đỉnh còn lại của $G(k)$. Chiều dài thu được ta gán cho mỗi đỉnh i như một trọng số $p[i]$. Ngoài ra, để chuẩn bị cho bước tiếp theo ta đánh giá lại trọng số cho mọi đỉnh kế sau của các đỉnh trong $G(k)$.

Khởi trị: Gán trọng số $p[i] = \infty$ cho mọi đỉnh, trừ đỉnh xuất phát s , gán trị $p[s] = 0$.

Ý nghĩa của thao tác này là khi mới đứng ở đỉnh xuất phát s của đồ thị con $G(0)$, ta coi như chưa thăm mảnh nào của đồ thị nên ta chưa có thông tin về đường đi từ s đến các đỉnh còn lại của đồ thị ban đầu. Nói cách khác ta coi như chưa có đường đi từ s đến các đỉnh khác s và do đó, độ dài đường đi từ s đến các đỉnh đó là ∞ .

Giá trị ∞ được chọn trong chương trình là:

$$\text{MAXWORD} = 65535.$$

Tại bước lặp thứ k ta thực hiện các thao tác sau:

- Trong số các đỉnh chưa xử lý, tìm đỉnh i có trọng số \min .
- Với mỗi đỉnh j chưa xử lý và kề sau với đỉnh i , ta chỉnh lại trọng số $p[j]$ của đỉnh đó theo tiêu chuẩn sau:

Nếu $p[i] + a[i, j] < p[j]$ thì gán cho $p[j]$ giá trị mới:

$$p[j] = p[i] + a[i, j]$$

Ý nghĩa của thao tác này là: nếu độ dài đường đi $\text{path}(s, j)$ trong đồ thị con $G(k-1)$ không qua đỉnh i mà lớn hơn độ dài đường đi mới $\text{path}(s, j)$ có qua đỉnh i thì cập nhật lại theo đường mới đó.

- Sau khi cập nhật ta cần lưu lại vết cập nhật đó bằng lệnh gán $\text{before}[i] = j$ với ý nghĩa là, đường ngắn nhất từ đỉnh s tới đỉnh j cần đi qua đỉnh i .
- Đánh dấu đỉnh i là đã xử lý.

Như vậy, tại mỗi bước lặp ta chỉ xử lý đúng một đỉnh i có trọng số \min và đánh dấu duy nhất đỉnh đó.

```
(*-----
Thuat toan Dijkstra
-----*)
```

```
procedure Dijkstra;
var i,k,j: byte;
begin
  Init;
```

```

for k := 1 to n do
begin
  i := Min; { tìm đỉnh i có trong số p[i] ->
min }
  d[i] := 1; {đánh dấu đỉnh i là đã xử lý }
  for j := 1 to n do
    if d[j] = 0 then {đỉnh chưa tham }
    if a[i,j] > 0 then {có đường đi i -> j }
      if p[i] + a[i,j] < p[j] then
        begin {sửa đỉnh }
          p[j] := p[i] + a[i,j];
          before[j] := i;
        end;
    end;
  end;
end;

```

Thuật toán chứa hai vòng for lồng nhau do đó có độ phức tạp là n^2 .

Sau khi hoàn thành thuật toán Dijkstra ta cần gọi thủ tục Ket (kết) để ghi lại kết quả theo yêu cầu của đầu bài như sau.

Với mỗi đỉnh $i = 1..n$ ta cần ghi vào tệp output chiều dài đường đi từ s đến i bao gồm giá trị $p[i]$ và các đỉnh nằm trên đường đó.

Chú ý rằng nếu $p[i]$ nhận giá trị khởi đầu tức là MAXWORD = 65535 thì tức là không có đường đi từ s đến i .

```

(*-----
Ket thuc thuat toan:ghi ket qua vao tep g
-----*)
procedure Ket;
var i: byte;
begin
  assign(g,gn); rewrite(g);
  for i := 1 to n do
    if (i=s) or (p[i] = MAXWORD) then
      writeln(g,0)
    else
      begin
        write(g,p[i],bl);
        path(i);
        writeln(g);
      end;
  close(g);
end;

```

Về ý nghĩa, mảng before chứa các con trỏ ngược từ mỗi đỉnh i đến đỉnh sát trước đỉnh i trên đường đi ngắn nhất, do đó ta phải lần ngược bằng thủ tục đệ quy path(i) để ghi vào tệp g vết của đường đi theo trật tự từ s đến i .

```

(*-----
Giai trinh duong ngan nhat tu s den i.
Ghi vao file g
-----*)
procedure path(i: byte);
begin
  if i=0 then exit;

```

```

        path(before[i]);
        write(g,i,bl);
    end;

(* Pascal *)
(*-----
DIJ.PAS Tim cac duong ngan nhat tu mot dinh
toi cac dinh con lai trong do thi co huong
(thuat giai Dijkstra)
-----*)
{$B-}
uses crt;
const
    MN = 100; {gioi han so dinh }
    MAXWORD = 65535; {Gia tri duong vo cung }
    fn = 'DIJ.INP';
    gn = 'DIJ.OUT';
    bl = #32; {dau cach }
    nl = #13#10;{xuong dau dong moi }

type
    mb1 = array[0..MN] of byte;
    mb2 = array[0..MN] of mb1;
    mw1 = array[0..MN] of word;

var
    a: mb2; {ma tran ke}
    before: mb1; {before[i] - dinh sat truoc dinh i}
    p: mw1; {p[i] - trong so dinh i}
    d: mb1; {d[i]=0: dinh i chua xu ly
            d[i]=1: dinh i da xu ly}
    n: byte; {so luong dinh}
    s: byte; {dinh xuat phat}
    f,g: text;

(*-----
    Doc du lieu vao ma tran ke a
    -----*)
procedure Doc; tự viết
(*-----
Hien thi du lieu de kiem tra thu tuc doc
-----*)
procedure Xem; tự viết
(*-----
Khoi tri
- trong so cac dinh: vo cung
- trong so dinh xuat phat s, p[s] = 0
- cac dinh sat truoc: 0
-----*)
procedure init;
var i: byte;
begin
    for i := 0 to n do
        begin
            d[i] := 0;

```

```

        p[i] := MAXWORD;
        before[i] := 0;
    end;
    p[s] := 0;
end;
(*-----
    Giai trình duong ngan nhut tu s den i.
    Ghi vao tep g
-----*)
procedure path(i: byte); tự viết
(*-----
Ket thuc thuat toan:
ghi ket qua vao file g
-----*)
procedure Ket; tự viết
(*-----
Trong so cac dinh chua xu ly,
chon dinh trong so min
-----*)
function Min: byte;
var m, i: byte;
begin
    m := 0;
    for i := 1 to n do
        if d[i]= 0 then {dinh i chua xu li }
            if p[i] <= p[m] then m := i;
        Min := m;
    end;
(*-----
    Thuat toan Dijkstra
-----*)
procedure Dijkstra; tự viết
BEGIN
    Doc; Xem; Dijkstra; ket;
END.

```

Ta minh hoạ tiến trình hoạt động của thuật toán Dijkstra qua thí dụ đã cho.

Sau khi đọc dữ liệu từ tệp `f=DIJ.INP` ta có $n = 7, s = 2$. Đồ thị có 7 đỉnh, đỉnh xuất phát là 2. Ma trận kề `a` thu được như sau:

a							
0	0	0	0	0	0	0	0
4	0	1	0	0	0	5	
0	0	0	0	0	0	1	
0	0	0	0	0	2	0	
0	0	0	3	0	0	0	
1	0	0	0	0	0	5	
0	0	0	1	0	0	0	

Khởi trị

Đỉnh	d	p	before
1	0	65535	0
2	0	0	0
3	0	65535	0
4	0	65535	0
5	0	65535	0

6	0	65535	0
7	0	65535	0

Bước lặp $k = 1$

$i = \min = 2$ với $p[2] = 0$.

Các đỉnh chưa xử lí và kề với đỉnh 2 sẽ được sửa trọng số là 1, 3 và 7 (có dấu ✓).

Vì $p[2] + a[2, 1] = 0 + 4 = 4 < p[1] = 65535$ nên $p[1]$ được sửa thành 4 và $before[1]$ được sửa thành 2.

Vì $p[2] + a[2, 3] = 0 + 1 = 1 < p[3] = 65535$ nên $p[3]$ được sửa thành 1 và $before[3]$ được sửa thành 2.

Vì $p[2] + a[2, 7] = 0 + 4 = 5 < p[7] = 65535$ nên $p[7]$ được sửa thành 5 và $before[7]$ được sửa thành 2.

Đỉnh	d	p	before
1✓	0	65535/4	0/2
2	0/1	0	0
3✓	0	65535/1	0/2
4	0	65535	0
5	0	65535	0
6	0	65535	0
7✓	0	65535/5	0/2

Bước lặp $k = 2$

$i = \min = 3$ với $p[3] = 1$.

Đỉnh chưa xử lí và kề với đỉnh 3 sẽ được sửa trọng số là đỉnh 7.

Vì $p[3] + a[3, 7] = 1 + 1 = 2 < p[7] = 5$ nên $p[7]$ được sửa thành 2 và $before[7]$ được sửa thành 3.

Đỉnh	d	p	before
1	0	65535/4	0/2
2	0/1	0	0
3	0/1	65535/1	0/2
4	0	65535	0
5	0	65535	0
6	0	65535	0
7✓	0	65535/5/2	0/2/3

Bước lặp $k = 3$

$i = \min = 7$ với $p[7] = 1$

Đỉnh chưa xử lí và kề với đỉnh 7 sẽ được sửa trọng số là đỉnh 4.

Vì $p[7] + a[7, 4] = 2 + 1 = 3 < p[4] = 65535$ nên $p[4]$ được sửa thành 3 và $before[4]$ được sửa thành 7.

Đỉnh	d	p	before
1	0	65535/4	0/2
2	0/1	0	0
3	0/1	65535/1	0/2
4✓	0	65535/3	0/7
5	0	65535	0
6	0	65535	0
7	0/1	65535/5/2	0/2/3

Bước lặp $k = 4$

$i = \min = 4$ với $p[4] = 3$.

Đỉnh chưa xử lí và kề với đỉnh 4 sẽ được sửa trọng số là đỉnh 6.

Vì $p[4] + a[4, 6] = 3 + 2 = 5 < p[6] = 65535$ nên $p[6]$ được sửa thành 5 và $\text{before}[6]$ được sửa thành 4.

Đỉnh	d	p	before
1	0	65535/4	0/2
②	0/1	0	0
③	0/1	65535/1	0/2
④	0/1	65535/3	0/7
5	0	65535	0
6✓	0	65535/5	0/4
⑦	0/1	65535/5/2	0/2/3

Bước lặp $k = 5$

$i = \min = 1$ với $p[1] = 4$.

Không có đỉnh chưa xử lí nào kề với đỉnh 1.

Đỉnh	d	p	before
①	0/1	65535/4	0/2
②	0/1	0	0
③	0/1	65535/1	0/2
④	0/1	65535/3	0/7
5	0	65535	0
6	0	65535/5	0/4
⑦	0/1	65535/5/2	0/2/3

Bước lặp $k = 6$

$i = \min = 6$ với $p[6] = 5$.

Không có đỉnh chưa xử lí nào kề với đỉnh 6. Chú ý rằng đỉnh 7 kề với đỉnh 6 nhưng đỉnh 7 này đã xử lí rồi.

Đỉnh	d	p	before
①	0/1	65535/4	0/2
②	0/1	0	0
③	0/1	65535/1	0/2
④	0/1	65535/3	0/7
5	0	65535	0
⑥	0/1	65535/5	0/4
⑦	0/1	65535/5/2	0/2/3

Thuật toán dừng.

Lưu ý rằng đỉnh xuất phát cho bài toán này là $s = 2$. Ta minh hoạ giải trình kết quả cho ba thí dụ sau.

Đường đi ngắn nhất từ đỉnh $s = 2$ đến đỉnh $t = 4$:

Vì $p[4] = 3$ nên độ dài đường đi là 3.

Để giải trình vết của đường đi từ 2 đến 4 ta dựa vào mảng $\text{before}[1..7]$ như sau:

Vì $\text{before}[4] = 7$, tức là trước khi đến đỉnh 4 phải qua đỉnh 7 nên ta có

$$7 \rightarrow 4$$

Vì $\text{before}[7] = 3$, tức là trước khi đến đỉnh 7 phải qua đỉnh 3 nên ta có

$$3 \rightarrow 7 \rightarrow 4$$

Vì $\text{before}[3] = 2$, tức là trước khi đến đỉnh 3 phải qua đỉnh 2 nên ta có

$$2 \rightarrow 3 \rightarrow 7 \rightarrow 4$$

Kết quả này được ghi ở dòng thứ tư của tệp DIJ.OUT như sau:

3 2 3 7 4

trong đó số đầu tiên 3 cho biết chiều dài đường đi, dãy số còn lại giải trình vết của đường đi từ đỉnh 2 đến đỉnh 4.

Đường đi ngắn nhất từ đỉnh $s = 2$ đến đỉnh $t = 5$:

Vì $p[5] = 32767$ ứng với giá trị dương vô cùng ∞ khởi trị lúc đầu nên không có đường đi từ đỉnh 2 đến đỉnh 5.

Ta ghi kết quả 0 tại dòng 5 của tệp DIJ.OUT .

Đường đi ngắn nhất từ đỉnh $s = 2$ đến đỉnh $t = 2$:

Vì $s = t$ nên ta coi như không có đường đi từ đỉnh 2 đến đỉnh 2.

Ta ghi kết quả 0 tại dòng 5 của tệp DIJ.OUT .

Các dạng khác của bài toán Dijkstra

Lưu ý rằng ma trận kề có thể chứa các giá trị thực, tuy nhiên cần giả thiết rằng mọi giá trị trong ma trận kề phải là các số không âm. Với các số âm bài toán sẽ phức tạp hơn.

P1. Nếu đồ thị đã cho là vô hướng ta giải như trên, chỉ lưu ý đến tính đối xứng khi đọc dữ liệu vào ma trận kề a .

P2. Nếu đề bài chỉ yêu cầu tìm một đường đi từ đỉnh s đến đỉnh t ta thực hiện các bước sau:

1. Đọc dữ liệu.
2. Gọi thuật toán Dijkstra.
3. Ghi kết quả $p[t]$ và giải trình một đường theo thuật toán $\text{path}(t)$.

// C#

```
using System;
using System.IO;
using System.Collections;
namespace SangTaoT1
{
    /*-----
    *   Thuật toán Dijkstra
    *   Tìm mọi đường ngắn nhất từ một đỉnh
    *   đến mọi đỉnh còn lại
    *   -----*/
    class Dijkstra
```

```

{
    const string fn = "Dij.inp";
    const string gn = "Dij.out";
    static int n = 0; // so dinh
    static int s = 0; // dinh xuất phát
    // c[i,j] ma tran ke cho biet
    // do dai cung (i,j)
    static int[,] c;
    static int[] d; // danh dau dinh
    static int[] t; // tro truoc
    static int[] p; // trong so dinh
    static void Main()
    {
        Run();
        Console.ReadLine();
    } // Main
    static void Run()
    {
        Doc(); Show(); Dij();
        Ghi(); Test();
        Console.WriteLine("\n Fini");
        Console.ReadLine();
    }
    // Kiem tra lai tep output
    static void Test() tự viết
    static void Ghi()
    {
        StreamWriter g = File.CreateText(gn);
        for (int i = 1; i <= n; ++i)
            if (i == s || p[i] == int.MaxValue)
                g.WriteLine(0);
            else
            {
                g.Write(p[i] + " ");
                int u = InvPath(i);
                for (int j = u; j > 0; --j)
                    g.Write(d[j] + " ");
                g.WriteLine();
            }
        g.Close();
    }
    // Lan nguoc duong di
    // tu dinh v den dinh s
    // ghi tam vao mang d
    static int InvPath(int v)
    {
        int i = 0;
        do
        {
            d[++i] = v;
            v = t[v];
        } while (v != 0);
        return i;
    }
}

```

```

}
static void Dij()
{
    for (int i = 0; i <= n; ++i)
    { //d: danh dau dinh, t: tro truoc
        d[i] = t[i] = 0;
        p[i] = int.MaxValue; // Trong so
    }
    p[s] = 0; // s: dinh xuất phát
    for (int i = 1; i < n; ++i)
    {
        int u = Minp(); // u: dinh trong so min
        d[u] = 1; // danh dau dinh da xet
        // sua lai nhan dinh
        for (int v = 1; v <= n; ++v)
            if (d[v] == 0) // dinh chua xet
                if (c[u, v] > 0) // co cung(u,v)
                    if (p[u] + c[u, v] < p[v])
                    { // sua lai nhan dinh v
                        p[v] = p[u] + c[u, v];
                        // chon cach di tu u -> v
                        t[v] = u;
                    }
    }
}
// Tim trong so cac dinh chua
// xu li mot dinh j co p min
static int Minp()
{
    int jmin = 0;
    for (int i = 1; i <= n; ++i)
        if (d[i] == 0) // dinh i chua xet
            if (p[i] < p[jmin]) jmin = i;
    return jmin;
}
static void Doc()
{
    int[] a = Array.ConvertAll((File.
        ReadAllText(fn)).Split(
        new char[] { '\0', '\n', '\t', '\r', ' ' },
        StringSplitOptions.RemoveEmptyEntries),
        new Converter<String, int>(int.Parse));
    n = a[0]; // so dinh
    s = a[1]; // dinh xuất phát
    c = new int[n + 1, n + 1];
    d = new int[n + 1];
    t = new int[n + 1];
    p = new int[n + 1];
    int k = 2;
    for (int i = 1; i <= n; ++i)
        for (int j = 1; j <= n; ++j)
            c[i, j] = a[k++];
}

```

```
        // Hien thi du lieu
        static void Show() tự viết
        // hien thi mang
        static void Print(int[] a, int n) tự viết
    } // Dijkstra
} // SangTao1
```

CHƯƠNG 8

SUY NGẪM

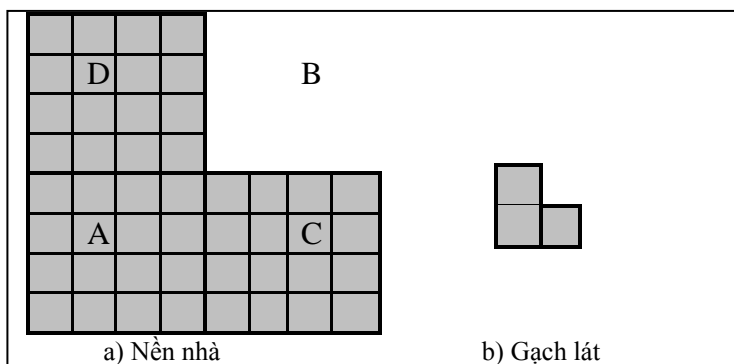
Chương này giới thiệu một số bài toán thuộc các lớp thuật giải khác nhau để bạn đọc tự luyện tập.

Thông thường, nếu chỉ biết một phương pháp giải mà gặp bài toán "trúng tủ", nghĩa là bài toán vận dụng chính phương pháp đã biết thì ta gần như không phải suy nghĩ gì. Tuy nhiên, khi đã có trong tay một số phương pháp thì việc chọn thuật giải cho mỗi bài toán cụ thể sẽ không dễ dàng chút nào.

Luyện tập và suy ngẫm để tìm kiếm đường đi trong các tình huống như trên sẽ cung cấp cho chúng ta nhiều kinh nghiệm quý.

Bài 8.1. Lát nền

Người ta cần lát kín một nền nhà hình vuông cạnh dài $n = 2^k$, (k là một số tự nhiên trong khoảng 1..6) khuyết một phần tư tại góc trên phải bằng những viên gạch màu hình thước thợ (chữ L) tạo bởi 3 ô vuông đơn vị như trong hình 2b. Hai viên gạch kề cạnh nhau dù chỉ 1 đơn vị dài phải có màu khác nhau. Hãy cho biết một cách lát với số màu ít nhất.



Hình 2

Dữ liệu vào: tệp văn bản `NEN.INP` chứa số tự nhiên n .

Dữ liệu ra: tệp văn bản `NEN.OUT`. Dòng đầu tiên là hai số tự nhiên m biểu thị số viên gạch và c là số màu cần dùng cho việc lát nền. Tiếp đến là một phương án lát nền tìm được, trong đó mỗi viên gạch lát được tạo bởi ba chữ số giống nhau thể hiện màu của viên gạch đó. Các số trên mỗi dòng cách nhau qua dấu cách.

Thí dụ, với $n = 8$ ta có một cách lát nền như sau:

Lời giải sau đây của bạn Lê Hoàng Hải, lớp 10 chuyên Tin, Đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội (năm 2002).

Để tính số viên gạch ta lấy ba phần tư diện tích hình vuông phủ nền nhà chia cho 3 là diện tích 1 viên gạch thước thợ:

$$\text{sogach} := ((3 * n * n) \text{ div } 4) \text{ div } 3;$$

NEN . INP	NEN . OUT
8	16 3
	3 3 1 1
	3 2 2 1
	1 2 3 3
	1 1 3 2
	3 3 1 2 2 3 1 1
	3 2 1 1 3 3 2 1
	1 2 2 3 1 2 2 3
	1 1 3 3 1 1 3 3

3	3	1	1						
3	2	2	1						
1	2	3	3						
1	1	3	2						
3	3	1	2	2	3	1	1		
3	2	1	1	3	3	2	1		
1	2	2	3	1	2	2	3		
1	1	3	3	1	1	3	3		

Hình 3. Nền nhà với $n = 8$

Về số màu, với $n = 2$ thì chỉ cần 1 viên gạch màu 1. Với mọi $n > 2$ ta sẽ trình bày một thuật toán cần tối đa ba màu.

Đầu tiên ta gọi thủ tục `Init` để khởi trị với hình vuông cạnh $v = 2$ nằm ở góc dưới trái của nền nhà được biểu diễn dưới dạng một mảng hai chiều `a`: ba ô trong hình vuông 2×2 sẽ được điền giá trị 1, ô nằm ở góc trên phải được điền giá trị 2 (phần tô đậm, hình 6). Gọi hình được khởi trị là `A`. Mỗi bước tiếp theo ta thực hiện ba thao tác biến hình sau đây:

- Tịnh tiến `A` đi lên theo đường chéo để thu được hình `B` (xem thủ tục `DichCheo`).
- Lật `A` sang phải (tức là lấy đối xứng gương qua trục tung) để thu được hình `C` (xem thủ tục `LatPhai`).
- Lật `A` lên trên (tức là lấy đối xứng gương qua trục hoành) để thu được hình `D` (xem thủ tục `LatLen`).

Chú ý rằng khi lật ta cần thực hiện thêm phép hoán đổi trị 1 và 3 cho nhau.

Mỗi lần lặp như vậy ta sẽ thu được hình vuông có cạnh tăng gấp đôi hình trước.

Khi $v = n$ ta gọi thủ tục `Finis` để ghi ba mảnh `D`, `A` và `C` vào tệp kết quả.

1	2								
1	1								

3	3	1	2						
3	2	1	1						
1	2	2	3						
1	1	3	3						

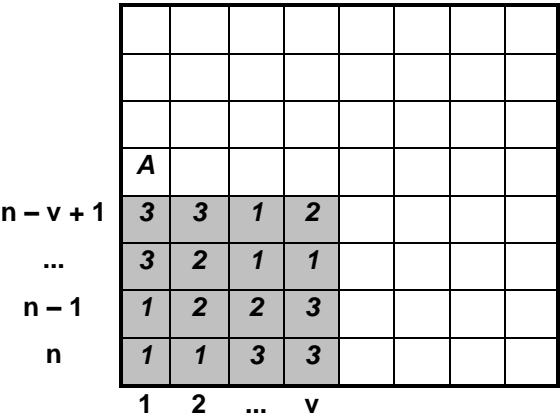
3	3	1	1						
3	2	2	1						
1	2	3	3						
1	1	3	2						
3	3	1	2	2	3	1	1		
3	2	1	1	3	3	2	1		
1	2	2	3	1	2	2	3		
1	1	3	3	1	1	3	3		

Hình 6

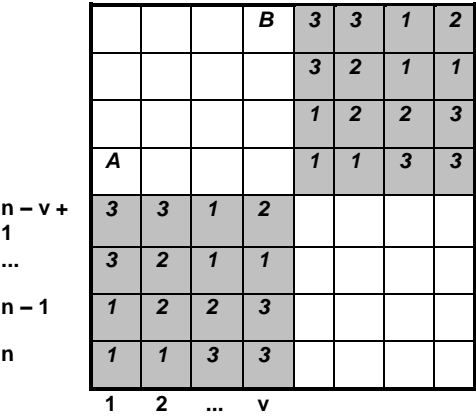
Ta biểu diễn nền nhà hình vuông qua một mảng hai chiều a với các dòng được mã số $1..n$ từ trên xuống và các cột được mã số từ $1..n$ từ trái qua phải. Khi đó viên gạch ở góc dưới trái sẽ có tọa độ $[n, 1]$. Sau khi đọc dữ liệu để nhận giá trị n , ta gán trị ban đầu cho 4 viên gạch ở góc dưới trái như sau:

```
a[n-1,1] := 1; a[n-1,2] := 2;
a[n,1] := 1; a[n,2] := 1;
```

Kí hiệu A là hình vuông cạnh dài v đơn vị nằm tại góc dưới trái của nền nhà, tức là phần mảng $v[n-v+1..n, 1..v]$ trong hình 7. Khi đó các thủ tục di chuyển hình vuông A sẽ được mô tả như sau.



Hình 7. Hình vuông a cạnh v được chọn để biến hình



Hình 8. Dịch chéo A để thu được B

Để dịch chéo hình vuông A ta copy dần từng phần tử trong các dòng, từ dòng $n-v+1$ đến dòng cuối cùng, dòng thứ n đến vị trí mới (h. 8).

```

(*-----
Dich hình vuông A cạnh v,
a[n-v+1..n,1..v]o góc dưới trái đi lên theo
đường chéo chính của nên nhà để nhận được B.
-----*)

procedure DichCheo(v: word);
var i,j:word;
begin
  for i := n-v+1 to n do
    for j := 1 to v do
      a[i-v,j+v] := a[i,j];
end;

```

	A						C
$n - v + 1$	3	3	1	2	2	3	1
...	3	2	1	1	3	3	2
$n - 1$	1	2	2	3	1	2	2
n	1	1	3	3	1	1	3
	1	2	...	v			

Hình 9. Lật A qua phải để thu được C

	3	3	1	1	D		
	3	2	2	1			
	1	2	3	3			
	1	1	3	2			
$n - v + 1$	3	3	1	2	A		
...	3	2	1	1			
$n - 1$	1	2	2	3			
n	1	1	3	3			
	1	2	...	v			

Hình 10. Lật A lên trên để thu được D

Để lật phải hình vuông A ta cũng chuyển dần từng phần tử trong các dòng, từ dòng $n - v + 1$ đến dòng cuối cùng, dòng thứ n đến vị trí mới. Tuy nhiên cần lưu ý thay đổi trị màu từ 1 sang màu 3 và ngược lại. Thao tác này được thực hiện qua phép gán $4 - c$, trong đó c là màu của ô gốc. Nếu $c = 2$ thì $4 - 2 = 2$, tức là màu 2 không thay đổi (h. 9).

```

(*-----
  Lat hình vuông canh v, a[n-v+1..n,1..v]
  o goc duoi trai sang phai, doi mau gạch
  de nhan duoc manh C.
-----*)

procedure LatPhai(v: word);
var i,j,v2:word;
begin
  v2 := 2*v;
  for i := n-v+1 to n do
    for j := 1 to v do
      a[i,v2-j+1] := 4-a[i,j];
    end;
end;

```

Để lật hình vuông A lên trên ta cũng làm tương tự như lật phải (h. 10).

```

(*-----
  Lat hình vuông canh v, a[n-v+1..n,1..v]
  o goc duoi len tren, doi mau gạch
  de nhan duoc manh D.
-----*)

procedure LatLen(v: word);
var i,j,v2:word;
begin
  v2 := n-2*v+1;
  for i := 0 to v-1 do
    for j := 1 to v do
      a[v2+i,j] := 4-a[n-i,j];
    end;
end;

```

Bình luận

Thuật giải sử dụng hai phép biến hình cơ bản trong chương trình phổ thông là phép dời hình (tịnh tiến) và đối xứng qua trục. Việc hoán đổi trị 1 và 3 cho nhau là một ý tưởng thông minh. Mỗi ô trong bảng được điền đúng một lần do đó độ phức tạp tính toán của thuật toán là n^2 , trong khi các bài giải khác đều phải sử dụng các phép dò tìm để xác định màu tô và gọi đệ quy nên thường tốn kém về miền nhớ và thời gian hơn nhiều lần.

(* Pascal *)

```

(*****
LATNEN - lat nen nha
hình vuông khuyết góc bang
cac vien gạch mau hình L
*****)
const
  fn = 'NEN.INP'; {input file}
  gn = 'NEN.OUT'; {output file}
  bl = #32; {dau cach}
  mn = 65; {kich thước tối đa của n}
var {n - chiều dài cạnh nen nha}
  f,g: text;
  a: array[0..mn,0..mn] of byte; {nen nha}

```

```

    {a[i] - mau vien gach lat}
    n,n2: word; {n2 = n/2}
    sogach: word;
    {-----
    Doc du lieu va khoi tri
    -----}
procedure Init;
begin
    {Doc du lieu}
    assign(f,fn); reset(f);
    readln(f,n); close(f);
    n2 := n div 2;
    sogach := ((3*n*n) div 4) div 3;
    {Dat tam 4 o vuong (2 X 2) tai goc duoi trai}
    a[n-1,1] := 1; a[n-1,2] := 2;
    a[n,1] := 1; a[n,2] := 1;
end;
    {-----
    Ket thuc, ghi tep
    -----}
procedure Fini(somau:word);
var i,j: word;
begin
    assign(g,gn); rewrite(g);
    writeln(g,sogach,bl,somau);
    {ghi goc tren trai D}
    for i := 1 to n2 do
        begin
            for j := 1 to n2 do
                write(g,a[i,j],bl);
            writeln(g);
        end;
    {ghi nua duoi A va C}
    for i := n2+1 to n do
        begin
            for j := 1 to n do
                write(g,a[i,j],bl);
            writeln(g);
        end;
    close(g);
end;
    (*-----
Dich hinh vuong A canh v, a[n-v+1..n,1..v]
o goc duoi trai di len theo duong cheo
chinh cua nen nha de nhan duoc manh B.
-----*)
procedure DichCheo(v: word); tự viết
    (*-----
Lat hinh vuong canh v, a[n-v+1..n,1..v]
o goc duoi trai sang phai, doi mau gach
de nhan duoc manh C.
-----*)
procedure LatPhai(v: word); tự viết

```

```

(*-----
Lat hình vuông canh v, a[n-v+1..n,1..v]
o goc duoi len tren, doi mau gach
de nhan duoc manh D.
-----*)

procedure LatLen(v: word); tự viết
procedure Run;
var v:word;
begin
  Init; {khởi tri voi n = 2}
  if n = 2 then
    begin
      Fini(1); exit;
    end;
  v := 1;
  repeat
    v := v*2;
    if v < n2 then DichCheo(v);
    LatPhai(v);
    LatLen(v);
  until v = n2;
  Fini(3);
end;
BEGIN
  Run;
END.

// C#
using System;
using System.IO;
namespace SangTao1
{
  /*-----
  *      Lat nen
  * -----*/
  class LatNen
  {
    const string fn = "Nen.inp";
    const string gn = "Nen.out";
    static int n = 0; // canh nen nha
    static int[,] c; // nen nha
    static void Main()
    {
      Run();
      Console.ReadLine();
    } // Main
    static void Run()
    {
      Doc(); Lat(); Ghi();
      Test();
      Console.WriteLine("\n Fini");
      Console.ReadLine();
    }
  }
}

```

```

// Kiem tra ket qua
static void Test() tự viết
static void Ghi()
{
    StreamWriter g = File.CreateText(gn);
    int n2 = n / 2;
    const string BL = " ";
    if (n == 2)
        g.WriteLine(1 + BL + 1);
    else g.WriteLine(((n*n)/4)+BL+3);
    for (int i = 0; i < n2; ++i)
    {
        for (int j = 0; j < n2; ++j)
            g.Write(c[i, j] + BL);
        g.WriteLine();
    }
    for (int i = n2; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
            g.Write(c[i, j] + BL);
        g.WriteLine();
    }
    g.Close();
}

static void Lat()
{
    c = new int[n, n];
    // Khoi tri
    for (int i = n - 2; i < n; ++i)
        for (int j = 0; j < 2; ++j)
            c[i, j] = 1;
    c[n - 2, 1] = 2;
    int k = 2;
    while (k < n)
    {
        Len(k);
        Phai(k);
        DichCheo(k);
        k *= 2;
    }
}

// Lat ma tran kXk len tren
static void Len(int k)
{
    for (int i = n - k; i < n; ++i)
        for (int j = 0; j < k; ++j)
            c[2*(n-k)-i-1, j] = 4-c[i, j];
}

// Lat ma tran kXk sang phai
static void Phai(int k)
{
    for (int i = n - k; i < n; ++i)
        for (int j = 0; j < k; ++j)

```

```

        c[i,2*k-j-1] = 4-c[i,j];
    }
    // dịch ma tran kXk theo hướng Đông-Bắc
    static void DichCheo(int k)
    {
        for (int i = n - k; i < n; ++i)
            for (int j = 0; j < k; ++j)
                c[i - k, j + k] = c[i, j];
    }
    static void Doc()
    {
        n = int.Parse(File.ReadAllText(fn).Trim());
        Console.WriteLine("\n Da doc n = " + n);
    }
} // LatLen
} // SangTao1

```

Bài 8.2. Chữ số cuối khác 0

Đề thi Tin học Quốc gia Ireland, 1994.

Tìm chữ số cuối cùng khác 0 của $n!$ với n trong khoảng 1..100.

Thí dụ:

- $n = 7$, kết quả = 4.
- $n = 15$, kết quả = 8.

Bài giải

Thuật giải của bạn Việt Hưng (Hà Tây, 2002) cho phép mở rộng giới hạn của n đến hàng chục vạn và nếu bạn muốn, có thể tiếp tục mở rộng đến hàng triệu.

Ý tưởng chính của Việt Hưng nằm ở công thức: $2 \times 5 = 10$ (hai lần năm là mười). Thật vậy, ta biết:

$$n! = 1.2.3...n$$

Các chữ số cuối cùng bằng 0 của n giai thừa được sinh ra khi và chỉ khi trong khai triển ra thừa số nguyên tố của tích trên có chứa các cặp thừa số 2 và 5. Vậy thì trước hết ta đếm số lượng các thừa số 2, kí hiệu là d_2 và số lượng các thừa số 5, kí hiệu là d_5 .

Thí dụ, với $n = 15$ ta có dạng khai triển ra thừa số nguyên tố của n giai thừa như sau:

$$n! = 1.2.3.(2.2).5.(2.3).7.(2.2.2).9.(2.5).11.(2.2.3).13.(2.7).(3.5)$$

Do đó $d_2 = 11$ và $d_5 = 3$. Vậy ta có ba cặp $2.5 = 10$ và số mũ đôi ra của thừa số 2 so với thừa số 5 sẽ là $d_2 - d_5 = 11 - 3 = 8$. Khi đó, kết quả sẽ là:

$$\text{chữ số cuối cùng khác 0 của } 15! = \text{chữ số cuối cùng của } k.2^{d_2-d_5}$$

Trong đó k là tích của các thừa số còn lại.

Dễ thấy với mọi n , ta luôn có $d_2 \geq d_5$ vì cứ hai số liên tiếp thì có một số chẵn (chia hết cho 2), còn năm số liên tiếp mới có một số chia hết cho 5.

Việc còn lại là lấy tích k của các số còn lại. Vì tích này không bao giờ tận cùng bằng 0 cho nên ta chỉ cần giữ lại một chữ số cuối cùng bằng cách lấy mod 10.

Để tính chữ số tận cùng của 2^m với $m = d_2 - d_5 > 0$ ta để ý đến tính tuần hoàn của nó, cụ thể là ta chỉ cần tính chữ số tận cùng của $2^{(m \bmod 4)}$ với các trường hợp:

$$m \bmod 4 = 0, 1, 2 \text{ và } 3.$$

Theo thí dụ trên ta có $m \bmod 4 = 8 \bmod 4 = 0$, do đó chữ số cuối của 2^m là 6 chứ không phải là 1 vì $m > 0$. Ta tính được (những cặp 2 và 5 được gạch dưới)

$$\begin{aligned} 15! &= 1.2.3.4.5.6.7.8.9.10.11.12.13.14.15 = \\ &= 1.2.3.(2.\underline{2}).\underline{5}.(2.3).7.(2.2.2).9.(\underline{2}.\underline{5}).11.(2.2.3).13.(\underline{2}.\underline{7}).(\underline{3}.\underline{5}) \\ &\Rightarrow (3.3.7.9.11.3.13.7.3).2^8 \bmod 10 = \\ &= ((k \bmod 10) \cdot (2^8 \bmod 10)) \bmod 10 = (3.6) \bmod 10 = 8. \end{aligned}$$

Chữ số cuối cùng khác 0 của $15!$ là 8.

Lưu ý rằng $(a.b) \bmod m = ((a \bmod m).(b \bmod m)) \bmod m$ cho nên ta có thể lấy mod ở các bước trung gian với số lần tùy ý.

Để tránh việc tràn số khi sử dụng các biến dung lượng nhỏ như kiểu `word` thay vì dùng `longint` chúng ta có thể tăng thêm một phép toán mod nữa. Chẳng hạn, khi tích lũy kết quả, thay vì viết

k := (k*c) mod 10;

ta nên viết

k := (k*(c mod 10)) mod 10;

trong đó k là số có một chữ số, c là số có thể rất lớn, đủ để làm tích $(k*c)$ vượt quá giới hạn cho phép. Thí dụ, nếu khai báo kiểu dữ liệu là `word` thì khi $k = 8$, $c = 17999$ ta có $k*c = 8*17999 = 143992 > 65535$ (giới hạn của `word`), trong khi 8 và 17999 đều nằm trong giới hạn cho phép.

Bình luận

Để ý rằng:

$14! = 87178291200$, có chữ số cuối cùng khác 0 là 2

$15! = 1307674368000$, có chữ số cuối cùng khác 0 là 8.

Nếu để tính $15!$ mà bạn chỉ lấy một chữ số cuối khác 0 của các phép tính trung gian thì sau khi tính chữ số cuối của $14!$ bạn sẽ nhận được 2 và cuối cùng là:

$$(2*15) \bmod 10 = 30 \bmod 10 = 3.$$

Kết quả này là sai vì chữ số cuối khác 0 của $15!$ là 8.

Chương trình sau đây chứa thủ tục `find` tìm chữ số cuối cùng khác 0 của $n!$ với n trong khoảng 1..65535.

Ta thực hiện một cải tiến nhỏ như sau. Thay vì đếm số lượng các thừa số 2 (d_2) và thừa số 5 (d_5) sau đó làm phép trừ $d_2 - d_5$, ta đếm luôn một lần hiệu số này và ghi vào biến m . Cụ thể là với mỗi giá trị $i = 2..n$ ta đếm số lượng các thừa số 2 trước, sau đó trừ đi số lượng các thừa số 5.

(* Pascal *)

```
(*-----
  Tim chu so khac khong cuoi cung cua n!
  -----*)
uses crt;
function find(n: longint): longint;
var m: longint;
    {m - hieu so cac thua so 2 va thua so 5}
    i,k,c: longint;
begin {k - ket qua trung gian}
    k := 1; m := 0;
    find := k;
    if (n <= 1) then exit;
```

```

d := 0;
for i := 2 to n do
begin
  c := i;
  while c mod 2 = 0 do
  begin
    m := m+1; c := c div 2;
  end;
  while c mod 5 = 0 do
  begin
    m := m-1; c := c div 5;
  end;
  k := (k*(c mod 10)) mod 10;
end;
case (m mod 4) of
  0: c := 6;
  1: c := 2;
  2: c := 4;
  3: c := 8;
end;
find := (k*c) mod 10;
end;
procedure run;
var n: longint;
begin
  writeln('-----');
  repeat
    write(' Nap N (Muon dung chuong trinh, bam 0 ):
    ');
    read(n);
    if n = 0 then halt;
    writeln(' find(' ,n,') = ',find(n));
  until false;
end;
BEGIN
  run;
END.

```

Kĩ thuật gán trước

Bạn có thể thay lệnh Case trong việc tính chữ số tận cùng của 2^m bằng các thao tác sau:

1. Khai báo và gán trước trị cho mảng **LuyThua**

```
const LuyThua: array[0..3] of word = (6,2,4,8);
```

Ý nghĩa của dòng lệnh trên: khai báo một mảng kiểu word với bốn phần tử có chỉ số biến thiên từ 0..3 và gán trước trị cho mảng này là:

```

LuyThua[0] := 6; {=  $2^4 \bmod 10$ }
LuyThua[1] = 2; {=  $2^1 \bmod 10$ }
LuyThua[2] = 4; {=  $2^2 \bmod 10$ }
LuyThua[3] = 8; {=  $2^3 \bmod 10$ }

```

Chú ý rằng, do đòi hỏi phải khai báo trước và gán đồng thời nên Turbo Pascal 7.0 quy định đặt khai báo này sau từ khoá **const**. Tuy nhiên **LuyThua** vẫn là biến mảng chứ không thể là hằng.

2. Khi cần tìm chữ số cuối cùng c của 2^m bạn khỏi phải dùng lệnh **case**, chỉ cần viết:

```
c := LuyThua[m mod 4];
Hàm Find theo phương án mới sẽ như sau:
const LuyThua: array[0..3] of word = (6,2,4,8);
{-----}
Find - Tìm chữ số cuối cùng khác 0 của n!
Phuong an dung mang LuyThua gan truoc
-----}

function find(n: longint): longint;
var m: longint;
{m - hieu so cac thua so 2 va thua so 5}
i,k,c: longint; {k - ket qua trung gian}
begin
  k := 1; m := 0;
  find := k;
  if (n <= 1) then exit;
  d := 0;
  for i := 2 to n do
  begin
    c := i;
    while c mod 2 = 0 do
    begin
      m := m+1; c := c div 2;
    end;
    while c mod 5 = 0 do
    begin
      m := m-1; c := c div 5;
    end;
    k := (k*(c mod 10)) mod 10;
  end;
  find := (k*LuyThua[m mod 4]) mod 10;
end;
```

// C#

```
using System;
namespace Tap1
{
  /*-----
  * Chu so cuoi khac 0 cua n!
  * -----*/
  class ChuSoCuoi
  {
    static void Main()
    {
      Test(200);
    } // Main
    static void Test(int n)
```

```

    {
        for (int i = 1; i < n; ++i)
        {
            Console.WriteLine("\n Chu so cuoi khac 0 cua " +
                               i + "! = " + Find(i) + ". ");
            Console.WriteLine("Bam exit de thoat: ");
            if (Console.ReadLine() == "exit") break;
        }
    }
    static int Find(int n)
    {
        if (n < 2) return 1;
        int m = 0;
        long k = 1;
        int c = 0;
        int[] mu = { 6, 2, 4, 8};
        for (int i = 2; i <= n; ++i)
        {
            c = i;
            while (c % 2 == 0)
            {
                ++m; c /= 2;
            }
            while (c % 5 == 0)
            {
                --m; c /= 5;
            }
            k = (k * (c % 10)) % 10;
        }
        return (int) ((k * mu[m % 4]) % 10);
    }
} // ChuSoCuoi
} // SangTao1

```

Bài tập làm thêm

Bài T1. Cho biết $N!$ tận cùng với bao nhiêu chữ số 0. Thí dụ, $15!$ tận cùng với 3 chữ số 0, vì $15! = 1307674368000$.

Gợi ý Nếu p là một số nguyên tố và p^K là nhân tử trong dạng phân tích $N!$ ra thừa số nguyên tố thì k được tính bằng tổng của các thương nguyên trong phép chia liên tiếp của N cho p .

Thí dụ, với $N = 15$, $p = 2$ và kí hiệu : là phép chia nguyên, ta tính được $15 : 2 = 7$; $7 : 2 = 3$; $3 : 2 = 1$; $1 : 2 = 0$. Do đó $k = 7 + 3 + 1 + 0 = 11$.

Với $N=15$, $p = 5$ ta tính được $15 : 5 = 3$; $3 : 5 = 0$. Do đó $k = 3 + 0 = 3$.

Như vậy $15! = 2^{11}.5^3.C$.

Chứng minh điều này khá dễ, bạn chỉ cần viết dãy $1.2...N$ thành các dòng, mỗi dòng p thừa số

1	...	p	Vậy là trong tích $1...N$ chứa K_1 thừa số p . Trong tích
$p+1$...	$2p$	$1...K_1$ sẽ chứa $K_1 : p$ thừa số $p...$
...	Bài T2. Phân tích $N!$ ra thừa số nguyên tố.
...	...	$k_1 p = N!$	Thí dụ, $15! = 2^{11}.3^6.5^3.7^2.11.13$.

Gợi ý Sử dụng kết quả Bài 1. Trước hết bạn cần viết hàm $Mu(n,p)$ cho ra số mũ cao nhất của số nguyên tố p trong dạng phân tích $n!$ ra thừa số nguyên tố.

Thí dụ, $Mu(15,2) = 11$; $Mu(15,5) = 3$.

Bài 8.3. Hình chữ nhật tối đại trong ma trận 0/1.

Cho một ma trận biểu diễn bằng một mảng hai chiều kích thước $N \times M$ ô và chỉ chứa các kí tự 0 và 1. Tìm hình chữ nhật chứa toàn kí tự 1 và có diện tích lớn nhất (gọi là hình chữ nhật tối đại).

Dữ liệu vào: Tập văn bản **CNMAX.INP**:

- Dòng đầu tiên: số tự nhiên M ,
 $3 \leq M \leq 70$
- Tiếp đến là các dòng dài bằng nhau thể hiện một xâu gồm M kí tự là các chữ cái 0/1 viết liền nhau.
- Số dòng của tệp input có thể lên tới 60 nghìn.

Dữ liệu ra: tệp văn bản **CNMAX.OUT**:

- Dòng đầu tiên: Diện tích của hình chữ nhật tối đại ABCD chứa toàn kí tự 1 tìm được.
- Dòng thứ hai: Tọa độ dòng và cột của đỉnh A.
- Dòng thứ ba: Tọa độ dòng và cột của đỉnh C.

	①	②	③	④	⑤	⑥	⑦	⑧	⑨
①	1	0	0	0	0	0	0	0	0
②	1	1	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	1	0
③	0	0	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	0	0
④	0	0	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	0	0
⑤	0	0	0	0	0	0	0	0	0

Hình 4

Thí dụ, với ma trận 5×9 chứa dữ liệu như hình 4 thì hình chữ nhật tối đại, tạm gọi là ABCD, có diện tích là 15 ô và có tọa độ điểm A là (dòng 2, cột 3) và điểm C là (dòng 4, cột 7) như trong hình 4.

CNMAX.INP	CNMAX.OUT
9	15
100000000	2 3
111111110	4 7
0011111100	
0011111100	
000000000	

Chúng ta sẽ xây dựng một thuật toán giải bài toán tổng quát hơn như sau:

Bài toán 8.3.1. Sân bay vũ trụ

Người ta cần xác định một vùng hình chữ nhật ABCD có diện tích lớn nhất và bằng phẳng trên hành tinh Vega để xây dựng sân bay vũ trụ. Bạn được cung cấp một mảnh bản đồ hành tinh Vega, nơi cần xác định vị trí xây dựng một sân bay. Mảnh bản đồ có dạng hình chữ nhật gồm nhiều dòng điểm mã số từ 1 trở đi, mỗi dòng có

đúng M điểm mã số từ 1 đến M . Mỗi điểm được tô một màu thể hiện độ cao của điểm đó. Yêu cầu: xác định hình chữ nhật ABCD chứa nhiều điểm đồng màu nhất.

Dữ liệu vào: Tập văn bản **CNMAX . INP**.

- Dòng đầu tiên: số tự nhiên M , $3 \leq M \leq 70$.
- Tiếp đến là các dòng dài bằng nhau thể hiện một xâu gồm M kí tự là các chữ cái $a..z$ viết liền nhau. Mỗi kí tự biểu diễn cho một màu thể hiện độ cao của điểm tương ứng trên mảnh bản đồ hành tinh Vega. Hai kí tự khác nhau thể hiện hai độ cao khác nhau. Hai điểm cùng độ cao được biểu diễn với cùng một kí tự.
- Số dòng của tệp input có thể lên tới 60 nghìn.

Thí dụ:

CNMAX . INP	CNMAX . OUT
20	80
bcccddeabcvvvvvvvb	2 6
bbbbccccccccbbbbb	9 15
vvvvcccccccccccb	
vvccccccccccccbbbbb	
pppppccccccccabbbb	
pppppcccccccczzzzz	
sscccccccccccczzzzz	
sssscccccccccccczzz	
hhhhhcccccccczzzzz	
uuuuuuuczzzzzzzzzz	

Dữ liệu ra: tệp văn bản **CNMAX . OUT**:

- Dòng đầu tiên: Diện tích của hình chữ nhật tối đại ABCD tìm được.
- Dòng thứ hai: Tọa độ dòng và cột của đỉnh A (ô Tây-Bắc).
- Dòng thứ ba: Tọa độ dòng và cột của đỉnh C (ô Đông-Nam).

Tính tổng quát của bài toán 8.3.1 thể hiện ở điểm sau:

- Tệp không chỉ chứa các kí tự 0/1.

Bài giải

Do không thể đọc toàn bộ dữ liệu từ tệp **CNMAX . INP** vào một mảng để xử lí nên chúng ta sẽ đọc mỗi lần một dòng vào biến kiểu xâu kí tự (string) y . Vì hình chữ nhật ABCD cần tìm chứa cùng một loại kí tự cho nên các dòng của hình sẽ liên thông nhau. Để phát hiện tính liên thông chúng ta cần dùng thêm một biến kiểu xâu kí tự x để lưu dòng đã đọc và xử lí ở bước trước. Tóm lại là ta cần xử lí đồng thời hai dòng: x là dòng trước và y là dòng đang xét.

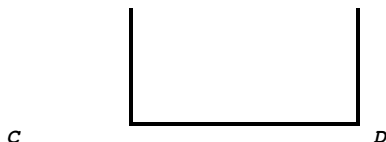
Nếu xét các cột trong hình chữ nhật cần tìm ta thấy chúng phải chứa cùng một loại kí tự. Ta dùng một mảng h với ý nghĩa phần tử $h[i]$ của mảng cho biết tính từ vị trí thứ i của dòng y trở lên có bao nhiêu kí tự giống nhau (và giống với kí tự $y[i]$). Ta gọi $h[i]$ là độ cao của cột i và mảng h khi đó sẽ được gọi là độ cao của dòng đang xét.

Thí dụ, mảng tích lũy độ cao của dòng thứ 5 trong thí dụ đã cho sẽ là:

$$h = (1, 1, 1, 1, 1, 4, 4, 4, 4, 4, 4, 5, 4, 4, 1, 2, 2, 4, 5)$$

A
B





Biết độ cao, với hai dòng x và y chúng ta dễ dàng tính được diện tích của mỗi hình chữ nhật ABCD chứa phần tử $y[i]$ trên cạnh DC. Thật vậy, giả sử ta đang xét kí tự thứ $i = 8$ trên dòng thứ 5 như đã nói ở phần trên. Ta có $h[8] = h[7] = h[6] = 4$; $h[9] = h[10] = h[11] = 4$; $h[12] = 5$; $h[13] = h[14] = h[15] = 4$; $h[16] = 1, \dots$ Vậy thì, khi ta đi từ i về hai phía, trái và phải, nếu gặp các kí tự giống kí tự $y[i]$ còn độ cao thì không nhỏ hơn $h[i]$ ta sẽ thu được hình chữ nhật lớn nhất chứa kí tự $y[i]$.

Với dòng thứ 5 là y đang xét, ta có:

```
x = 'vccccccccccccccbbbbb'; {dòng thu 4 }
```

```
y = 'pppppccccccccccabbbb'; {dòng thu 5 }
```

dòng	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
4 x	v	v	c	c	c	c	c	c	c	c	c	c	c	c	c	b	b	b	b	b
5 y	p	p	p	p	p	c	c	c	c	c	c	c	c	c	a	b	b	b	b	b
h	1	1	1	1	1	4	4	4	4	4	4	5	4	4	4	1	2	2	4	5

trong đó x chứa dữ liệu của dòng thứ 4, y chứa dữ liệu của dòng thứ 5 trong tệp **CNMAX.INP**.

Từ điểm $i = 8$ dịch chuyển về bên trái ta thu được điểm $c1 = 6$; dịch chuyển về bên phải ta thu được điểm $c2 = 15$. Điều kiện để dịch chuyển là:

- ♦ $(y[c1] = y[i]) \text{ and } (h[c1] \geq h[i])$, nếu qua trái và
- ♦ $(y[c2] = y[i]) \text{ and } (h[c2] \geq h[i])$, nếu qua phải.

Hai thao tác trên được đặt trong hàm tính diện tích của hình chữ nhật ABCD lớn nhất chứa điểm $y[i]$. Hàm cho ra ba giá trị:

- ♦ $c1$: điểm trái nhất hay là toạ độ cột của đỉnh D.
- ♦ $c2$: điểm phải nhất hay là toạ độ cột của đỉnh C.
- ♦ Diện tích của hình.

```
(*-----
      Dien tich cua hinh chua diem y[i]
-----*)
function DienTich(i: byte; var c1,c2: byte): longint;
begin
  {Qua trai}
  c1 := i;
  while (y[c1-1] = y[i]) and (h[c1-1] >= h[i])
    do dec(c1);
  {Qua phai}
  c2 := i;
  while (y[c2+1] = y[i]) and (h[c2+1] >= h[i])
    do inc(c2);
  DienTich := h[i]*(c2 + 1 - c1);
end;
```

Phần xử lí chính được đặt trong thủ tục Run.

Mảng $h[1..m]$ lúc đầu được khởi trị toàn 0. Sau mỗi lần đọc dòng y ta chỉnh lại độ cao h theo tiêu chuẩn sau.

Tại điểm i đang xét trên dòng y , nếu $y[i] = x[i]$ độ cao $h[i]$ được tăng thêm 1. Ngược lại, nếu $y[i] \neq x[i]$ ta đặt lại độ cao $h[i] = 1$.

```
{Chỉnh do cao}
for i := 1 to m do
  if y[i] = x[i] then inc(h[i]) else h[i] := 1;
```

Một vài chú giải

1. Chúng ta sử dụng phần tử $h[0] = 0$ và $h[m+1] = 0$ để chặn vòng lặp, cụ thể là để các điều kiện $(h[c1-1] \geq h[i])$ và $(h[c2+1] \geq h[i])$ trở thành **false** ở cuối vòng lặp.
2. Một con đếm dòng **d** kiểu **longint** cho biết ta đang xử lí dòng nào của tệp.
3. Dòng x lúc đầu được khởi trị toàn dấu cách là kí tự không có trong văn bản thể hiện tâm bản đồ.
4. Mỗi khi xử lí xong dòng y ta cần sao chép giá trị của **y** cho **x** để lưu giữ cho bước tiếp theo. Khi đó x sẽ trở thành dòng trước.
5. Mỗi khi tìm được một hình chữ nhật, ta so sánh diện tích của hình với diện tích lớn nhất hiện có (**dtmax**) để luôn luôn đảm bảo rằng **dtmax** chính là diện tích lớn nhất trong vùng đã khảo sát.

Thủ tục **Run** được thiết kế theo sơ đồ sau:

1. Mở tệp dữ liệu **CNMAX.INP**;
2. Đọc giá trị chiều dài mỗi dòng vào biến **m**;
3. Khởi trị:
 - Con đếm dòng **d** := 0;
 - Dòng trước **x** toàn dấu cách;
 - Mảng chiều cao **h** toàn 0;
 - **dtmax** := 0; {diện tích max}
4. Lặp cho đến khi hết tệp **CNMAX.INP**:
 - 4.1. Đọc dòng **y**;
 - 4.2. Tăng con đếm dòng **d**;
 - 4.3. Chỉnh độ cao **h[1..m]**;
 - 4.4. Xử lí mỗi kí tự **y[i]** của dòng **y**; **i** = 1..m.
 - 4.4.1. Tìm diện tích **dt** của hình chữ nhật lớn nhất chứa phần tử $y[i]$; cho giá trị ra là **dt** và hai chỉ số đầu trái **c1** và đầu phải **c2** thể hiện cạnh đáy của hình chữ nhật.
 - 4.4.2. Nếu **dt** > **dtmax**: chỉnh lại các giá trị

Diện tích max: **dtmax** := **dt**;

Toạ độ đỉnh **A** (**Axmax**, **Aymax**):

Axmax := **d** - **h[i]** + 1;

Aymax := **c1**;

Toạ độ đỉnh **C** (**Cxmax**, **Cymax**):

Cxmax := **d**;

Cymax := **c2**;
 - 4.5. Sao chép dòng y sang x : **x** := **y**;

5. Đóng tệp **CNMAX.INP**.

6. Thông báo kết quả.

```

procedure Run;
var i,c1,c2: byte;
dt: longint;
begin
    assign(f,fn); reset(f); readln(f,m);
    d := 0;
    {Khoi tri cho dong dau tien}
    x := BL;
    for i := 1 to m do x := x + BL;
    {Khoi tri cho chieu cao}
    fillchar(h,sizeof(h),0);
    while not eof(f) do
        begin
            readln(f,y);
            inc(d);
            {Chinh do cao}
            for i := 1 to m do
                if y[i] = x[i] then inc(h[i]) else h[i] := 1;
            for i := 1 to m do
                begin
                    dt := DienTich(i,c1,c2);
                    if dt > dtmax then
                        begin
                            dtmax := dt;
                            Axmax := d-h[i]+1; Aymax := c1;
                            Cxmax := d; Cymax := c2;
                        end;
                end;
            x := y;
        end;
    close(f);
    Ket; {ghi ket qua}
end;

```

Độ phức tạp tính toán

Giả sử tệp **CNMAX.INP** chứa n dòng, mỗi dòng chứa m kí tự. Khi xử lí một dòng, tại mỗi kí tự thứ i trên dòng đó ta dịch chuyển qua trái và qua phải, tức là ta phải thực hiện tối đa m phép duyệt. Vậy, với mỗi dòng gồm m kí tự ta phải thực hiện m^2 phép duyệt. Tổng cộng, với n dòng ta thực hiện tối đa $t = n.m^2$ phép duyệt.

(* Pascal *)

```

(*****
    CNMAX - Dien tich hinh chu nhat
    toi dai
    *****)
uses crt;
const
    fn = 'CNMAX.INP'; {Ten tep chua du lieu vao}
    gn = 'CNMAX.OUT'; {Ten tep chua du lieu ra}
    MN = 80; {chieu rong van ban}

```

```

BL = #32; {dau cach}
NL = #13#10; {xuong dong moi}
var f,g: text;
m: byte; {chieu rong tam ban do}
d: longint; {dem dong}
x,y: string; {x - dong tren
               {y - dong duoi}
h: array[0..MN] of longint;
{chieu cao cua cac cot}
dtmax: longint; {Dien tich max}
Axmax, Cxmax: longint; {toa do diem A, C}
Aymax, Cymax: byte;
(*-----
      Ghi file ket qua
-----*)
procedure Ket;
begin
  assign(g,gn); rewrite(g);
  writeln(g,Dtmax);
  writeln(g,Axmax,BL,Aymax);
  writeln(g,Cxmax,BL,Cymax);
  close(g);
end;
(*-----
      Dien tich cua hinh chua diem y[i]
-----*)
function DienTich(i: byte; var c1,c2: byte): longint;
  tự viết
procedure Run; tự viết
BEGIN
  run;
END.

// C#
using System;
using System.IO;
namespace SangTao1
{
  /*-----
   *      Chu nhât toi dai
   * -----*/
  class ChuNhatMax
  {
    static string fn = "cnmax.inp"; // input file
    static string gn = "cnmax.out"; // output file
    static string x;// dong tren
    static string y;// dong duoi
    static int m = 0; // chieu dai moi dong
    static int[] d; // cao do
    static int dong = 0; // dem dong
    static int smax = 0; // dien tich max
    static int ad = 0; // toa do dong cua dinh A
    static int ac = 0; // toa do cot cua dinh A
  }
}

```

```

static int cd = 0; // toa do dong cua dinh C
static int cc = 0; // toa do cot cua dinh C

static void Main()
{
    Run(); Ghi(); Test();
    Console.ReadLine();
} // Main
static void Ghi()
{
    File.WriteAllText(gn, smax + "\n" +
        ad + " " + ac + "\n" + cd + " " + cc);
}
static void Test() tự viết
static void Run()
{
    StreamReader f = File.OpenText(fn);
    do // Bo cac dong trong dau file
    {
        y = f.ReadLine().Trim();
    } while (y.Length == 0);
    m = int.Parse(y);
    Console.WriteLine(m);
    d = new int[m + 2];
    x = new string('#', m + 2);
    Array.Clear(d, 0, d.Length);
    dong = 0; smax = 0;
    while (!f.EndOfStream)
    {
        y = f.ReadLine().Trim();
        if (y.Length < m) break;
        ++dong; XY(); x = y;
    }
    f.Close(); ac++; cc++;
}
// Xu li cap dong x va y
static void XY()
{
    int t = 0; // chi so trai
    int p = 0; // chi so phai
    int dt = 0; // dien tich
    for (int i = 0; i < m; ++i)
        if (y[i] == x[i]) ++d[i];
        else d[i] = 1;
    for (int i = 0; i < m; ++i)
    {
        t = Trai(i); p = Phai(i);
        dt = (p - t + 1) * d[i];
        if (smax < dt)
        {
            smax = dt;
            ad = dong - d[i] + 1; ac = t;
            cd = dong; cc = p;
        }
    }
}

```

```

    }
    }
}
// quet tu k qua phai
static int Phai(int k)
{
    for (int i = k + 1; i < m; ++i)
        if (d[i] < d[k]) return i - 1;
    return m - 1;
}
// quet tu k qua trai
static int Trai(int k)
{
    for (int i = k - 1; i >= 0; --i)
        if (d[i] < d[k]) return i + 1;
    return 1;
}
} // ChuNhatMac
} // SangTao1

```

Ứng dụng

Bài toán tìm hình chữ nhật tối đại thường dùng trong lĩnh vực đồ hoạ và xử lý ảnh. Dưới đây liệt kê vài ứng dụng điển hình.

1. Trong khi vẽ bản đồ ta thường phải tìm một hình chữ nhật tối đại trong một vùng, chẳng hạn lãnh thổ của một quốc gia để có thể viết các kí tự vào đó như tên quốc gia, tên châu lục.
2. Trong hình học phân hình (fractal) ta thường phải tìm một số hình vuông hoặc chữ nhật tối đại thoả mãn một số tiêu chuẩn cho trước để làm mẫu.

Trong bài này, để trình bày vấn đề được đơn giản chúng ta đã thay mỗi điểm bằng một kí tự.

Bài tập làm thêm

Bài T1. Với mỗi kí tự c cho trước hãy tìm trong tệp CNMAX.INP một hình chữ nhật tối đại chứa toàn kí tự c .

Bài T2. Với cặp giá trị (k, c) cho trước hãy tìm hình chữ nhật tối đại chứa cùng một loại kí tự và đồng thời chứa điểm nằm trên dòng k , cột c của tệp CNMAX.INP.

Bài 8.4. Ma phương

Ma phương là những bảng số hình vuông trong đó mỗi dòng, mỗi cột và mỗi đường chéo đều cùng thoả một số tính chất nào đó. Các nhà thiên văn cổ Trung Hoa cho rằng mỗi tinh tú trên bầu trời đều ứng với một ma phương. Nhiều nhà toán học cổ Ai Cập, Ấn Độ và sau này các nhà toán học phương Tây đã phát hiện những tính chất khá lí thú của các loại ma phương. Trong bài này ta giới hạn khái niệm ma phương theo nghĩa sau.

Ma phương bậc n là một bảng số hình vuông, mỗi cạnh n ô chứa các số từ 1 đến n^2 sao cho tổng các số trên mỗi dòng, trên mỗi cột và trên mỗi đường chéo đều bằng nhau. Tổng này được gọi là đặc số của ma phương.

4	9	2
3	5	7
8	1	6

(a)

16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1

(b)

(a) – ma phương bậc 3, đặc số $S_3 = 15$

(b) – ma phương bậc 4, đặc số $S_4 = 34$

Yêu cầu: Với mỗi trị $N = 1..20$ xây dựng ma phương bậc n .

Bài giải

Ta dễ dàng tính được đặc số của ma phương bậc n qua nhận xét: tổng của một cột (dòng) chính là tổng của toàn bộ các số nằm trong bảng chia cho số cột (số dòng):

$$S_n = (1 + 2 + \dots + n^2)/n = n(n^2 + 1)/2.$$

Theo các thí dụ trên ta có:

Đặc số của ma phương bậc 3: $S_3 = 3(9 + 1)/2 = 15$.

Đặc số của ma phương bậc 4: $S_4 = 4(16 + 1)/2 = 34$.

Như vậy, trong ma phương bậc 3, tổng của các số nằm trên cùng hàng, cùng cột hoặc cùng đường chéo đều là 15. Trong ma phương bậc 4, tổng này là 34.

Tính chất trên không thay đổi nếu ta điền lần lượt các số hạng của một cấp số cộng vào ma phương.

Ngoài bậc $n = 2$, với mọi số tự nhiên $n \geq 1$ đều tồn tại ma phương với nhiều cách bố trí khác nhau. Chúng ta sẽ tìm hiểu hai thuật toán để cài đặt.

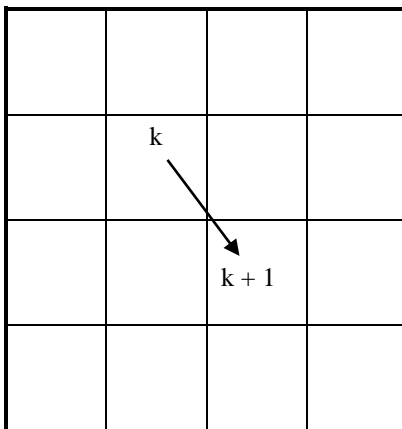
Với mỗi n cho trước ta xét tính chẵn lẻ của nó. Nếu n lẻ ta gọi thủ tục MPL (ma phương bậc lẻ), ngược lại ta gọi thủ tục MPC (ma phương bậc chẵn).

```
(*-----
                        Ma phuong
-----*)
procedure MP(bac: word);
begin
  n := bac;
  if n = 2 then exit;
  if Odd(n) then MPL else MPC;
  Test;
end;
```

Trong đó n là biến tổng thể chứa bậc của ma phương cần xây dựng.

Hàm $\text{Odd}(n)$ cho giá trị đúng (true) khi n là một số lẻ, ngược lại hàm nhận giá trị sai (false).

Thủ tục **MPL(n)** : Ma phương bậc lẻ



Điền ô theo hướng Đông - Nam cho ma phương bậc lẻ.

Ta dùng một mảng hai chiều M để chứa ma phương cần xây dựng theo các bước sau đây. Để cho tiện ta đặt tên cho ma phương cần xây dựng là hình vuông ABCD.

Bước 1. Khởi trị: Điền các số 0 vào bảng $M[1..n, 1..n]$.

Bước 2. Xác định ô xuất phát: Đó là ô giữa của dòng cuối, tức là ô $M[i, j]$ với $i = n, j = (n \text{ div } 2) + 1$.

Bước 3. Điền số: Với mỗi k biến thiên từ 1 đến n^2 ta thực hiện các thao tác sau:

3.1. Điền ô (i, j) : $M[i, j] := k$;

3.2. Xác định vị trí ii, jj mới để điền số tiếp theo $(k + 1)$.

Vị trí ii, jj mới được xác định theo nguyên tắc Đông-Nam với ý nghĩa là sau khi đã điền giá trị k , giá trị $k + 1$ sẽ được viết vào ô nằm ở vị trí Đông-Nam so với ô chứa k . Như vậy, nếu $M[i, j] = k$ thì vị trí ô chứa $k + 1$ sẽ là $ii := i + 1, jj := j + 1$. Có thể sẽ xảy ra các tình huống sau đây:

3.2.1. $(i = n)$ và $(j = n)$: Số k đã viết ở góc Đông-Nam (góc C) của ma phương. Khi đó, nếu đi tiếp theo hướng Đông-Nam thì sẽ rơi vào ô nằm ngoài bảng. Ta gọi tình huống này là tình huống Đông-Nam và xử lý như sau: viết số $k + 1$ vào ô sát trên ô chứa k , tức là chọn

$ii := n - 1; jj := n;$

Ta gọi phương thức xử lý này là đèn trên: Đèn vào ô trên ô vừa viết (xem các số 6 \rightarrow 7 trong ma phương bậc 3).

3.2.2. $(i = n)$ và $(j < n)$: Số k đã viết nằm ở cạnh DC và khác ô C. Ta gọi tình huống này là tình huống Nam và xử lý theo phương thức "nổi bọt" như sau: Viết $k + 1$ vào vị trí Đông-Nam tức là ô

$i + 1 = n + 1, j + 1.$

Dĩ nhiên ô này nằm ở ngoài bảng, dưới cạnh DC. Bây giờ ta cho nó nổi lên tới cạnh AB. Như vậy ta sẽ chọn

$ii := (i \bmod n) + 1;$

$jj := (j \bmod n) + 1;$

(xem các số $1 \rightarrow 2$ và $8 \rightarrow 9$ trong ma phương bậc 3).

3.2.3. ($i < n$) và ($j = n$): Số k đã viết nằm ở cạnh BC và khác ô C. Ta gọi tình huống này là tình huống Đông và xử lý theo phương thức đẩy trái như sau: Viết $k + 1$ vào vị trí Đông-Nam tức là ô

$$i+1, j+1 = n+1$$

Ô này nằm ngoài bảng, bên phải cạnh BC. Bây giờ ta đẩy trái nó sang tới cạnh AD. Giống như trên, ta chọn:

`ii := (i mod n) + 1;`

`jj := (j mod n) + 1;`

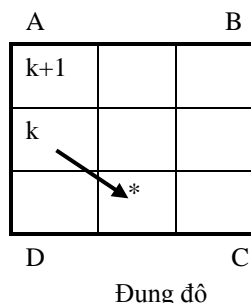
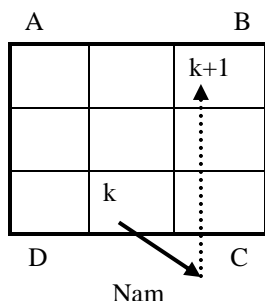
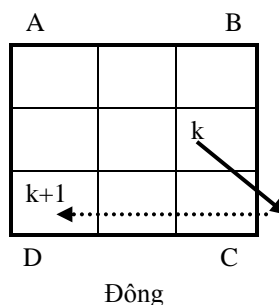
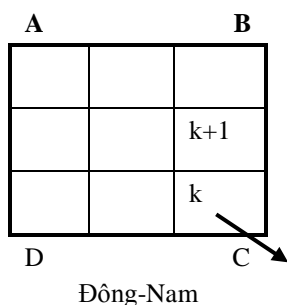
(xem các số $2 \rightarrow 3$ và $7 \rightarrow 8$ trong ma phương bậc 3).

3.2.4. Đụng độ: Cuối cùng có thể ô (ii, jj) rơi vào trong bảng nhưng ở đó đã có số được viết trước. Ta gọi tình huống này là tình huống đụng độ và xử lý theo phương thức đền trên như tình huống Đông-Nam (xem bước đi 3 và 4 trong ma phương bậc 3). Trường hợp này ta chọn:

`ii := i - 1; jj := j;`

Sử dụng phép toán đồng dư ta có thể giải quyết tự động các tình huống Nam và Đông theo công thức

`ii := (i mod n) + 1 ; jj := (j mod n) + 1;`



Các tình huống khi điền ma phương bậc lẻ

Dưới đây là thủ tục ma phương bậc lẻ.

(*-----
 Ma phuong bac le
 -----*)

```
procedure MPL;
var k,i,ii,j,jj: word;
begin
  {Buoc 1: Khoi tri}
```

```

fillchar(M, sizeof(M), 0);
{Buoc 2: Xac dinh o xuat phat}
i := n;
j := n div 2 + 1;
{Buoc 3: Dien so}
for k := 1 to n*n do
begin
  {3.1. Dien o (i,j)}
  M[i,j] := k;
  {3.2 Xac dinh vi tri ii,jj
   cho so tiep theo (k+1)}
  if (i=n) and (j=n) then
  {3.2.1.Tinh huong Dong-Nam:Den tren}
  begin
    ii := n-1; jj := n;
    end
  else
    begin {3.2.2 va 3.2.3.}
      ii := i mod n + 1;
      jj := j mod n + 1;
      end;
    if M[ii,jj]<>0 {o da dien} then
      begin
        {3.2.4. Dung do: Den tren}
        ii := i-1; jj := j;
        end;
      i := ii; j := jj;
    end;
  end;
end;

```

Thủ tục **MPC(n)**: Ma phương bậc chẵn

Bước 1. Khởi trị: Điền các số từ 1 đến n^2 vào bảng theo trật tự từ trên xuống dưới, từ trái sang phải. Thí dụ, với $n = 4$, M được khởi trị như sau:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Khởi trị cho ma phương bậc 4

Bước 2. Tạo xâu mẫu: Ta tạo xâu mẫu s phục vụ cho việc đổi chỗ các số trong M. Xâu mẫu s có chiều dài $k = n \text{ div } 2$ và bao gồm các kí tự 'T', 'D', 'N' và 'B' với các ý nghĩa sau:

- 'T' - thực hiện phép đối xứng tâm: Đổi chỗ các cặp phần tử:

$$M[i, j] \leftrightarrow M[n-i+1, n-j+1]$$

$$M[i, n-j+1] \leftrightarrow M[n-i+1, j]$$

- 'D' - thực hiện phép đối xứng theo trục dọc: Đổi chỗ cặp phần tử:

$$M[i, j] \leftrightarrow M[i, n-j+1]$$

- 'N' - thực hiện phép đối xứng theo trục ngang: Đổi chỗ cặp phần tử:

$$M[i, j] \leftrightarrow M[n-i+1, j]$$

- 'B' - không làm gì.

Xâu mẫu s được tạo như sau:

2.1. Khởi trị xâu rỗng $s[0] := \#0;$

2.2. Tính $k := n \text{ div } 2;$

2.3. Nạp ($k \text{ div } 2$) kí tự 'T' cho s .

```
for i := 1 to (k div 2) do
    s := s+TT;
```

2.4. Nếu k lẻ nạp thêm hai kí tự 'DN' cho s :

```
if Odd(k) then s := s+DD+NN;
```

2.5. Điền thêm các kí tự 'B' cho đủ k kí tự trong s .

```
for i := length(s)+1 to k do s := s+BB;
```

trong đó các hằng TT, DD, NN và BB được khai báo như sau

```
const
    TT = 'T'; {doi xung Tam}
    DD = 'D'; {doi xung Doc}
    NN = 'N'; {doi xung Ngang}
    BB = 'B'; {Bo qua}
```

Các thí dụ tạo xâu mẫu s :

- Với $n = 4$ ta có $k = n \text{ div } 2 = 2$ (chẵn), $k \text{ div } 2 = 1$, do đó $s = \text{'TB'}$

- Với $n = 6$ ta có $k = n \text{ div } 2 = 3$ (lẻ), $k \text{ div } 2 = 1$, do đó $s = \text{'TDN'}$

- Với $n = 18$ ta có $k = n \text{ div } 2 = 9$ (lẻ), $k \text{ div } 2 = 4$, do đó $s = \text{'TTTTDNBBBB'}$

Thủ tục khởi trị cho ma phương bậc chẵn sẽ như sau:

```
(*-----
    Khoi tri cho ma phuong bac chan
-----*)
```

```
procedure InitMPC;
```

```
var i, j: word;
```

```
begin
```

```
    {Buoc 1. Khoi tri}
```

```
    for i := 1 to n do
```

```
        for j := 1 to n do
```

```
            M[i, j] := Num(i, j);
```

```
    {Buoc 2: Tao xau mau}
```

```
    s[0] := #0; {Khoi tri xau rong}
```

```
    k := n div 2;
```

```
    {Nap (k div 2) ki tu T}
```

```
    for i := 1 to (k div 2) do s := s+TT;
```

```
    {Nap them 2 ki tu D va N neu k le}
```

```
    if Odd(k) then s := s+DD+NN;
```

```
    {Bu cac ki tu B cho du k ki tu}
```

```
    for i := length(s)+1 to k do s := s+BB;
```

```
end;
```

Chú ý rằng để khởi trị giá trị rỗng cho xâu s ta có hai cách:

Cách 1. Viết hai dấu nháy đơn sát nhau:

```
s := ' ';
```

Cách 2. Gán cho phần tử $s[0]$ là nơi chứa chiều dài xâu s giá trị #0 ứng với kí tự có mã ASCII là 0:

```
s[0] := #0;
```

Cách thứ nhất khiến bạn đọc dễ nhầm với dấu cách, nên dùng cách thứ hai.

Bước 3. Điền số theo xâu mẫu

```
for i := 1 to k do  
  begin  
    XuLyDong(i);  
    QuayXauMau;  
  end;
```

Để xử lí dòng i ta lần lượt xét các kí tự $s[j]$ trong xâu mẫu và xử lí từng phần tử $M[i, j]$, $j = 1..length(s)$.

- Nếu $s[j] = 'T'$ ta gọi thủ tục **Tam** (i, j, n) để thực hiện đối xứng tâm cho các ô (i, j) và ($i, n - j + 1$).
- Nếu $s[j] = 'D'$ ta gọi thủ tục **Doc** (i, j, n) để thực hiện đối xứng theo trục dọc cho ô (i, j).
- Nếu $s[j] = 'N'$ ta gọi thủ tục **Ngang** (i, j, n) để thực hiện đối xứng theo trục ngang cho ô (i, j).
- Nếu $s[j] = 'B'$ ta bỏ qua.

```
procedure XuLyDong(i: word);  
var j: word;  
begin  
  for j := 1 to k do  
    case s[j] of  
      TT: Tam(i, j);  
      DD: Doc(i, j);  
      NN: Ngang(i, j);  
    end;  
  end;
```

Đề ý rằng tại bước khởi trị số nằm trên ô (i, j) sẽ có giá trị $(i - 1) * n + j$. Ta sẽ sử dụng hàm **Num** (i, j, n) để tính giá trị này.

```
(*-----  
  So nam tren hang i, cot j.  
-----*)  
function Num(i, j: word): word;  
begin  
  Num := (i-1)*n+j;  
end;
```

Các thủ tục lấy đối xứng qua tâm, dọc và ngang là khá đơn giản.

```
(*-----  
  Lay doi xung qua tam (2 cap so)  
-----*)  
procedure Tam(i, j: word);  
begin  
  M[i, j] := Num(n-i+1, n-j+1);
```

```

    M[n-i+1,n-j+1] := Num(i,j);
    M[n-i+1,j] := Num(i,n-j+1);
    M[i,n-j+1] := Num(n-i+1,j);
end;
(*-----
                        Doi xung doc
-----*)
procedure Doc(i,j: word);
begin
    M[i,j] := Num(i,n-j+1);
    M[i,n-j+1] := Num(i,j);
end;
(*-----
                        Doi xung ngang
-----*)
procedure Ngang(i,j: word);
begin
    M[i,j] := Num(n-i+1,j);
    M[n-i+1,j] := Num(i,j);
end;

```

Mỗi lần xử lí xong một dòng ta cần quay xâu mẫu s thêm một vị trí theo chiều kim đồng hồ, kí tự cuối xâu được đưa về đầu xâu, các kí tự khác được dịch qua phải một vị trí.

Thí dụ về quay xâu mẫu s : Với $n = 18$ ta có:

```

s = 'TTTTDNBBB'
sau lần quay thứ nhất ta thu được
s = 'BTTTTDNBB'
sau lần quay thứ hai ta thu được
s = 'BBTTTTDNB'
...

```

Để quay xâu $s[1..k]$ ta lấy kí tự cuối cùng $s[k]$ ghép lên khúc đầu $s[1..k-1]$, tức là đặt

```
s := s[k]+s[1..k-1]
```

Để lấy đoạn $s[1..k-1]$ ta gọi hàm copy:

```
copy(s,1,len-1)
```

trong đó len chính là chiều dài xâu s .

Lệnh

```
copy(s,i,m)
```

sẽ tạo ra một xâu con của xâu s với m kí tự kể từ kí tự thứ i :

```
copy(s,i,m) = s[i..(i+m-1)]
```

```

(*-----
    Quay xau mau s 1 vi tri
-----*)

```

```

procedure QuayXauMau;
var len: byte;
begin
    len := length(s);
    s := s[len] + copy(s,1,len-1);
end;

```

Sau đây là thủ tục tạo ma phương bậc chẵn.

```
(*-----
      Ma phuong bac chan
-----*)
procedure MPC;
var i,j: word;
begin
  if n=2 then exit;
  InitMPC;
  {Dien so theo xau mau}
  for i := 1 to k do
    begin
      XuLyDong(i);
      QuayXauMau;
    end;
end;
```

Sau khi đã tạo xong ma phương ta cần kiểm tra lại xem ma trận M có thoả các tính chất cơ bản của ma phương bậc n cho trước hay không.

Thủ tục `Test` sẽ hiển thị ma phương trên màn hình và đồng thời kiểm tra xem các tổng các phần tử trên mỗi dòng, tổng các phần tử trên mỗi cột, tổng các phần tử trên đường chéo chính $c1$ và tổng các phần tử trên đường chéo phụ $c2$ có bằng đặc số của ma phương hay không.

Ta sử dụng hai mảng `dong[1..n]`, `cot[1..n]` và hai biến đơn $c1$, $c2$ để tính các tổng này bằng một lần duyệt mảng hai chiều M .

Để tính đặc số ta lưu ý kiểu của biến chứa đặc số d là `longint` trong khi đó các trị của mảng M là `word`. Trong Turbo Pascal có nguyên tắc sau:

Nguyên tắc định kiểu cho biểu thức

Kiểu của trị của biểu thức số sẽ là kiểu rộng nhất trong số các kiểu của các đại lượng (hằng và biến) có mặt trong biểu thức.

Theo quy định này, với khai báo n là biến kiểu `word` thì kiểu của trị của biểu thức tính đặc số của ma phương bậc n

$$((n*n+1)*n) \text{ div } 2$$

cũng sẽ là `word`.

Điều này có thể dẫn đến tràn số.

Ta khắc phục bằng thao tác hai bước như sau:

```
d := 0;
d := d + ((n*n+1)*n) div 2;
```

Khi đó, do biểu thức vế phải của phép gán có chứa biến d thuộc kiểu `longint` nên kết quả sẽ có kiểu `longint`.

Bạn cũng có thể dùng toán tử chuyển sang kiểu `longint` một trong các đại lượng trong biểu thức vế phải, chẳng hạn:

```
d := ((n*n+longint(1))*n) div 2
(*-----
      Hien thi va kiem tra ket qua
-----*)
procedure Test;
var i,j: word;
```

```

dong, cot: ML1;
d,c1,c2: longint;
begin {Tinh Dac so}
  d := 0;
  d := d+((n*n+1)*n) div 2;
  writeln(NL,' Ma phuong bac ',n,', Dac so: ',d);
  fillchar(dong,sizeof(dong),0);
  fillchar(cot,sizeof(cot),0);
  c1 := 0;
  c2 := 0;
  for i := 1 to n do
    begin
      writeln;
      c1 := c1 + M[i,i];
      c2 := c2 + M[i,n-i+1];
      for j := 1 to n do
        begin
          write(M[i,j]:4);
          dong[i] := dong[i] + M[i,j];
          cot[j] := cot[j] + M[i,j];
        end;
      end;
      writeln;
    for i := 1 to n do
      begin
        if dong[i] <> d then
          writeln(' Sai dong ',i,BL,dong[i]);
        if cot[i] <> d then
          writeln(' Sai cot ',i,BL, cot[i]);
        end;
        if c1 <> d then writeln(' Sai cheo chinh ',c1);
        if c2 <> d then writeln(' Sai cheo phu ',c2);
      end;
    end;
end;

```

(* Pascal *)

Chương trình MAPHUONG.PAS dưới đây tạo bộ dữ liệu kiểm thử với các ma phương từ bậc 1 đến bậc 20.

```

(* MAPHUONG.PAS *)
uses crt;
const
  MN = 50;
  TT = 'T'; {doi xung Tam}
  DD = 'D'; {doi xung Doc}
  NN = 'N'; {doi xung Ngang}
  BB = 'B'; {Bo qua}
  BL = #32; {dau cach}
  NL = #13#10; {qua dong moi}
type
  MW1 = array[0..MN] of word;
  MW2 = array[0..MN] of MW1;
  ML1 = array[0..MN] of longint;
var M: MW2;

```

```

n,k: word;
s: string;
(*-----
   Hien thi va kiem tra ket qua
   -----*)
procedure Test; tự viết
(*-----
   So nam tren hang i, cot j
   -----*)
function Num(i,j: word):word;
begin
  Num := (i-1)*n+j;
end;
(*-----
   Lay doi xung qua tam (2 so)
   -----*)
procedure Tam(i,j: word); tự viết
(*-----
   Doi xung doc
   -----*)
procedure Doc(i,j: word); tự viết
begin
  M[i,j] := Num(i,n-j+1);
  M[i,n-j+1] := Num(i,j);
end;
(*-----
   Doi xung ngang
   -----*)
procedure Ngang(i,j: word); tự viết
(*-----
   Quay xau mau s 1 vi tri
   -----*)
procedure QuayXauMau; tự viết
(*-----
   Khoi tri cho ma phuong bac chan
   -----*)
procedure InitMPC; tự viết
procedure XuLyDong(i: word); tự viết
(*-----
   Ma phuong bac chan
   -----*)
procedure MPC; tự viết
(*-----
   Ma phuong bac le
   -----*)
procedure MPL; tự viết
(*-----
   Ma phuong
   -----*)
procedure MP(bac: word); tự viết
(*-----
   Test cac ma phuong bac 1..20
   -----*)

```

```

procedure Run;
var i: word;
begin
    clrscr;
    for i := 1 to 20 do MP(i);
end;
BEGIN
    Run;
END.

// C#
using System;
namespace SangTao1
{
    class maphuong
    {
        const int mn = 50;
        static int[,] a = new int[mn, mn]; // Ma phuong
        static char[] s = new char[mn]; // Xau mau
        static void Main(string[] args)
        {
            for (int n = 1; n <= 20; ++n)
            { MaPhuong(n); Console.ReadLine(); }
        }
        static void MaPhuong(int n)
        {
            if (n == 2)
                Console.WriteLine("\n Khong ton tai "+
                                   "Ma phuong bac 2");
            else if (n % 2 == 1) MaPhuongLe(n);
            else MaPhuongChan(n);
        }
        static void MaPhuongLe(int n)
        {
            Array.Clear(a, 0, a.Length);
            int i = n - 1;
            int j = n / 2;
            int nn = n * n;
            a[i, j] = 1;
            for (int k = 2; k <= nn; ++k)
            { Next(ref i, ref j, n); a[i, j] = k; }
            Test(n); Print(n);
        }
        // Ma phuong le: Tim o dien tiep
        static void Next(ref int i, ref int j, int n)
        {
            int ic = i; int jc = j;
            i = (i + 1) % n; j = (j + 1) % n;
            if (i + j == 0)
            { i = n - 2; j = n - 1; }
            if (a[i, j] > 0)
            { i = ic - 1; j = jc; }
        }
    }
}

```

```

static void MaPhuongChan(int n)
{
    // Khoi tri 1, 2, ..., n*n
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            a[i, j] = Num(n, i, j);
    // Tao xau mau
    int k = n / 2;
    int k2 = k / 2;
    for (int i = 0; i < k2; ++i)
        s[i] = 'T';
    for (int i = k2; i < k; ++i)
        s[i] = 'B';
    if (k%2==1) // k/2 le
    { s[k2] = 'D'; s[k2 + 1] = 'N'; }
    for (int i = 0; i < k; ++i)
    {
        for (int j = 0; j < k; ++j)
            switch (s[j])
            {
                case 'T': Tam(n, i, j); break;
                case 'D': Doc(n, i, j); break;
                case 'N': Ngang(n, i, j); break;
            } // switch
        Quay(k);
    }
    Test(n); Print(n);
}
// Quay xau mau s qua phai 1 vi tri
static void Quay(int k)
{
    char x = s[k - 1];
    Array.Copy(s, 0, s, 1, k - 1);
    s[0] = x;
}
// Doi xung qua Tam
static void Tam(int n, int i, int j)
    tự viết
// Doi xung qua truc doc
static void Doc(int n, int i, int j)
    tự viết
// Doi xung qua truc ngang
static void Ngang(int n, int i, int j)
    tự viết
// So nam tai ding i, cot j
static int Num(int n, int i, int j)
{ return (i * n) + j + 1; }
// Kiem tra cac ma phuong
static bool Test(int n)
{
    int c1 = 0;
    int c2 = 0;
    // row[i] = tong dong i

```

```

int[] row = new int[n];
// col[j] = tong cot j
int[] col = new int[n];
int dacso = (n * n + 1) * n / 2;
Console.WriteLine("\n\n Ma phuong bac "
                  + n);

Console.WriteLine(" Dac so: " + dacso);
for (int i = 0; i < n; ++i)
    { row[i] = col[i] = 0;}
// tinh tong cac cot va dong
for (int i = 0; i < n; ++i)
{
    c1 += a[i, i];
    c2 += a[i, n - 1 - i];
    for (int j = 0; j < n; ++j)
        { row[i] += a[i, j];
          col[j] += a[i, j];}
}
if (c1 != dacso)
{
    Console.WriteLine("Loi Duong cheo 1:"
                      +" Dac so " + dacso + ", " + c1);
    return false;
}
if (c2 != dacso)
{
    Console.WriteLine("Loi Duong cheo 2:"
                      +" Dac so " + dacso + ", " + c2);
    return false;
}
for (int i = 0; i < n; ++i)
{
    if (row[i] != dacso)
    {
        Console.WriteLine("Loi dong "
                          +(i+1)+ ": "+" Dac so "+dacso
                          +", "+row[i]);
        return false;
    }
    if (col[i] != dacso)
    {
        Console.WriteLine("Loi cot "
                          +(i+1)+": "+" Dac so "+dacso
                          + ", " + col[i]);
        return false;
    }
}
return true;
}
// Hien thi Ma phuong
static void Print(int n)
{
    for (int i = 0; i < n; ++i)

```

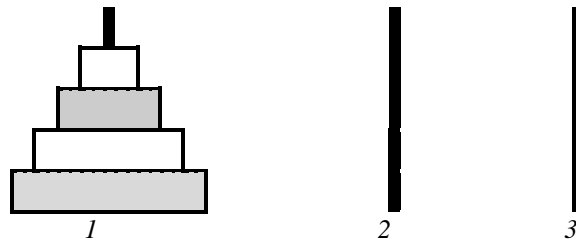
```

    {
        Console.WriteLine();
        for (int j = 0; j < n; ++j)
            Console.Write("{0} ", a[i, j]);
    }
} // MaPhuong
} // SangTao1

```

Các bài toán Tháp Hà Nội

Bài 8.5. Tháp Hà Nội cổ



Hình 5. Bài toán tháp Hà Nội

Có ba cọc cắm tại ba vị trí là 1, 2 và 3 như hình 5. Trên một cọc, gọi là cọc a có một chồng gồm n đĩa bằng gỗ hình tròn to nhỏ khác nhau được xuyên lỗ ở giữa tựa như những đồng xu và đặt chồng lên nhau để tạo ra một toà tháp. Người chơi phải chuyển được toà tháp sang cọc $b \neq a$ theo các quy tắc sau:

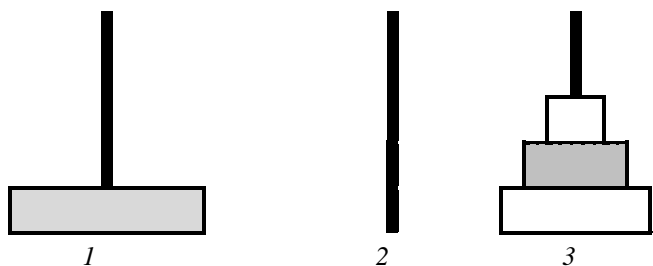
- (1) Người chơi được sử dụng cọc còn lại để đặt tạm các tầng tháp.
- (2) Mỗi lần chỉ được chuyển 1 tầng tháp từ một cọc sang một trong hai cọc còn lại.
- (3) Không bao giờ được đặt tầng tháp lớn lên trên tầng tháp nhỏ.

Hãy tìm cách giải bài toán trên với số lần chuyển ít nhất.

Thuật toán

Chắc chắn là bạn đã biết cách giải bài toán trên. Tuy nhiên để có thể giải dễ dàng các biến thể của bài toán tháp Hà Nội chúng ta thử tìm hiểu một cách lập luận sau.

Giả sử ta quan sát một người chơi giỏi, tức là người chuyển được n tầng tháp từ cọc 1 sang cọc 2 với số lần chuyển tối thiểu. Ta dùng một chiếc máy ảnh chụp từng kết quả trung gian sau mỗi bước chuyển của người chơi này. Tổng số bức ảnh, trừ tấm ảnh ban đầu, chính là số bước chuyển các tầng. Trong số các bức ảnh chắc chắn phải có một bức như hình 10.



Hình 10. Một ảnh phải có

Tại sao vậy? Tại vì chừng nào chưa dỡ được $n - 1$ tầng tháp ở phía trên của vị trí 1 để chuyển sang vị trí 3 thì anh ta không thể chuyển được tầng tháp cuối, tức là tầng lớn nhất sang vị trí 2.

Gọi $Hn(n, a, b)$ là thủ tục chuyển n tầng tháp từ vị trí a sang vị trí $b \neq a$, ta thấy:

- Nếu $n = 0$: không phải làm gì;
 - Nếu $n > 0$ ta phải thực hiện ba bước sau:
- ☐ Thoạt tiên chuyển $n - 1$ tầng tháp từ vị trí a sang vị trí $c = 6 - a - b$:
- $Hn(n-1, a, 6-a-b)$
- ☐ Sau đó chuyển tầng lớn nhất từ vị trí a sang vị trí b :
- $a \rightarrow b$
- ☐ Cuối cùng chuyển $n - 1$ tầng tháp từ c sang b :
- $Hn(n-1, 6-a-b, b)$

Để ý rằng, do ta mã hoá các cọc là 1, 2 và 3 cho nên biết hai trong ba vị trí đó, là x , y chẳng hạn, ta dễ dàng tính được vị trí còn lại z theo hệ thức

$$z = 6 - x - y$$

Thủ tục chuyển n tầng từ cọc a sang cọc b được viết bằng Pascal như sau:

```

procedure Hn(n, a, b: byte) ;
begin
  if n = 0 then exit;
  Hn(n-1, a, 6-a-b) ;
  write(a, ' -> ', b, ' ');
  Hn(n-1, 6-a-b, b) ;
end;

```

Chọn phương thức ghi tệp hoặc màn hình

Có thể chọn một trong hai cách ghi kết quả vào tệp văn bản hoặc hiển thị lên màn hình. Bạn chỉ cần lưu ý rằng màn hình được định nghĩa như là một tệp văn bản. Chính xác hơn là như sau. Trong Turbo Pascal vùng đệm màn hình, tức là nơi chứa dữ liệu để xuất ra màn hình, được định nghĩa dưới dạng một tệp. Nếu ta mở một tệp với tên rỗng như sau:

```

assign(f, '');
rewrite(f) ;

```

trong đó f là biến kiểu tệp văn bản được khai báo như sau

```

var f: text;

```

thì sau đó mọi lệnh

```

write(f, ...) ;

```

sẽ xuất dữ liệu ra màn hình.

Ngược lại, khi tên tệp trong lệnh mở tệp nói trên khác rỗng, thí dụ:

```
assign(f, 'hanoi.out');
rewrite(f);
```

thì sau đó mọi lệnh

```
write(f,...);
```

sẽ ghi dữ liệu vào tệp **hanoi.out** trên đĩa.

Chương trình hoàn chỉnh dưới đây sẽ ghi kết quả vào tệp văn bản **hanoi.out** được mở trên đĩa. Trước khi ghi chính thức, bạn hãy chạy thử chương trình với lệnh mở tệp có tên rỗng để kiểm tra kết quả trên màn hình. Sau khi thấy ưng ý, bạn hãy viết tên tệp cụ thể để lưu kết quả vào đĩa.

Chương trình sử dụng biến đếm *d* nhằm đếm số bước chuyển.

```
(*   Pascal   *)

uses crt;
var d: longint;
f: text;
procedure Hn(n,a,b: byte);
begin
    if n = 0 then exit;
    Hn(n-1,a,6-a-b);
    inc(d);
    writeln(f,d,'. ',a,' -> ',b);
    Hn(n-1,6-a-b,b);
end;
procedure runHn(n: byte);
begin
    d := 0;
    assign(f, 'hanoi.out');
    rewrite(f);
    writeln('-----');
    Hn(n,1,2);
    writeln(f,'Total: ',d,' step(s)');
    close(f);
    readln;
end;
BEGIN
    runHn(3);
END.
```

Khi thực hiện chương trình **Hanoi.pas** với $n = 3$ ta thu được kết quả sau:

```
1. 1 -> 2
2. 1 -> 3
3. 2 -> 3
4. 1 -> 2
5. 3 -> 1
6. 3 -> 2
7. 1 -> 2
Total: 7 step(s)
```

// C#

```
using System;
namespace Tap1
```

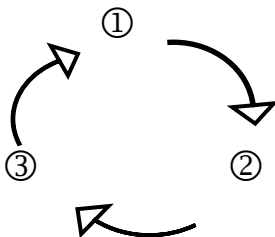
```

{
    /*-----
    *                Thap Ha Noi
    * -----*/
    class ThapHaNoi
    {
        static int d = 0; // đếm số lần chuyển
        static void Main()
        {
            Console.WriteLine("\n Ha Noi ");
            HaNoi(3, 1, 2);
            Console.ReadLine();
        } // Main
        static void HaNoi(int n, int a, int b)
        {
            if (n == 0) return;
            HaNoi(n - 1, a, 6 - a - b);
            Console.WriteLine(++d +
                               ". " + a + " -> " + b);
            HaNoi(n - 1, 6 - a - b, b);
        }
    } // class
} // space

```

Bài 8.6. Tháp Hà Nội xuôi

Nội dung giống như bài toán tháp Hà Nội cổ chỉ sửa lại quy tắc (2) như sau: (2) Mỗi lần chỉ được chuyển 1 tầng tháp từ một cọc sang cọc sát nó theo chiều kim đồng hồ.



Điều kiện này quy định ba phép chuyển 1 tầng tháp giữa các cọc như sau:

① → ②, hoặc ② → ③, hoặc ③ → ①.

Hãy tìm cách giải bài toán với số lần chuyển ít nhất.

Bài giải

Suy nghĩ một chút bạn sẽ thấy cái lợi của nguyên tắc "Bức ảnh buộc phải có". Đặt tên các tầng tháp theo thứ tự từ nhỏ đến lớn là $1..n$. Ta mô tả mỗi tấm ảnh như một bộ ba $(a:[A], b:[B], c:[C])$ trong đó A, B và C là các tầng tháp đặt tại mỗi vị trí tương ứng. Gọi a là vị trí xuất phát, b là vị trí đích, bài toán trên có thể được phát biểu như sau:

Gia thiết: $(a:[1..n], b:[], c:[])$

...

Kết luận: $(a:[], b:[1..n], c:[])$

Với ý nghĩa là cho biết bức ảnh ban đầu và cuối cùng. Hãy liệt kê ít nhất các bức ảnh cần có ở giữa ba dấu chấm (...) sao cho bộ ảnh giải thích được quá trình chuyển tháp theo các điều kiện cho trước.

Mỗi bức ảnh được gọi là một hình trạng. Ngoài hai hình trạng đầu tiên và cuối cùng, một trong những hình trạng buộc phải có sẽ là $(a: [n], b: [], c: [1..n-1])$. Tiếp đó là hình trạng $(a: [], b: [n], c: [1..n-1])$ thu được sau lệnh chuyển $a \rightarrow b$

Gọi $H_{nx}(n, a, b)$ là thủ tục giải bài toán tháp Hà Nội xuôi, chuyển tháp n tầng từ vị trí a sang vị trí b . Ta xét hai trường hợp.

a) Trường hợp vị trí b đứng sát vị trí a theo chiều kim đồng hồ:

Theo trường hợp này ta có các cặp (a, b) sau đây: (1, 2), (2, 3) và (3, 1).

Đặc tả cho điều kiện của trường hợp này là $b = (a \bmod 3) + 1$

Với ý nghĩa là, nếu biết mã số vị trí a thì vị trí b sát sau a xuôi theo chiều kim đồng hồ sẽ được tính theo công thức trên.

Nếu vị trí các cọc được bố trí như sau

	①	②	③
thì giữa a và b có ba tình huống, cụ thể là			
Tình huống	①	②	③
1	a	b	
2		a	b
3	b		a

Tháp Hà Nội xuôi

Đặc tả a và b kề nhau $b = (a \bmod 3) + 1$

Dựa vào các hình trạng buộc phải có ta có thể mô tả việc chuyển tháp trong trường hợp này giống như trong bài toán tháp Hà Nội cổ, cụ thể là:

Hình trạng	Ý nghĩa	Lệnh
$(a: [1..n], b: [], c: [])$	Hình trạng ban đầu với vị trí b sát vị trí a . $b = (a \bmod 3) + 1$ Để chuyển n tầng từ a sang b theo chiều kim đồng hồ ta phải...	
$(a: [n], b: [], c: [1..n-1])$	Chuyển (tạm) $n-1$ tầng từ a qua $c = 6 - a - b$.	$H_{nx}(n-1, a, 6 - a - b)$
$(a: [], b: [n], c: [1..n-1])$	sau đó chuyển tầng còn lại của a qua b .	$a \rightarrow b$
$(a: [], b: [1..n], c: [])$	Cuối cùng chuyển $n-1$ tầng từ $c = 6 - a - b$ qua b là hoàn tất.	$H_{nx}(n-1, 6 - a - b, b)$

Đoạn trình cho trường hợp này sẽ là

```

if  $b = (a \bmod 3) + 1$  then
  begin { $b$  kề  $a$ }
     $H_{nx}(n-1, a, 6-a-b)$  ;
    inc ( $d$ ) ;
    writeln ( $f, d, '. ', a, ' \rightarrow ', b$ ) ;
     $H_{nx}(n-1, 6-a-b, b)$  ;
  end ;

```

end ...

b) Trường hợp vị trí b không đúng sát vị trí a theo chiều kim đồng hồ:

Các cặp (a, b) khi đó sẽ là: (1, 3), (2, 1) và (3, 2). Đặc điểm chung của các cặp này là: nếu đi từ a đến b theo chiều kim đồng hồ chúng ta phải vượt qua c là vị trí nằm giữa a và b.

Tình huống	①	②	③
1	a		b
2	b	a	
3		b	a

Tháp Hà Nội xuôi

Đặc tả a và b không kề nhau $b \neq (a \bmod 3) + 1$

Các hình trạng buộc phải có khi đó sẽ là:

Hình trạng	Ý nghĩa	Lệnh
(a: [1..n], c: [], b: [])	Hình trạng ban đầu với vị trí b cách a qua c theo chiều kim đồng hồ. $b \neq (a \bmod 3) + 1$ Để chuyển n tầng từ a sang b cách qua vị trí c = 6 - a - b ta phải...	
...		
(a: [n], c: [], b: [1..n - 1])	Chuyển (tạm) n - 1 tầng từ a qua b.	Hnx(n - 1, a, b)
(a: [], c: [n], b: [1..n - 1])	sau đó chuyển (tạm) tầng còn lại của a qua c = 6 - a - b.	$a \rightarrow 6 - a - b$
...		
(a: [1..n - 1], c: [n], b: [])	Rồi lại chuyển (tạm) n - 1 tầng từ b qua a nhằm giải phóng vị trí đích b...	Hnx(n - 1, b, a)
(a: [1..n - 1], c: [], b: [n])	để chuyển tầng lớn nhất n từ c qua b.	$6 - a - b \rightarrow b$
(a: [], c: [], b: [1..n])	Cuối cùng chuyển n - 1 tầng từ a qua b là hoàn tất.	Hnx(n - 1, a, b)

(* Pascal *)

```
(*****
      Ha Noi xuoi
*****)
uses crt;
var d: longint;
f: text;
procedure Hnx(n,a,b: byte);
begin
  if n = 0 then exit;
  if b = (a mod 3)+1 then
    begin {b ke a}
      Hnx(n-1,a,6-a-b);
      inc(d);
      writeln(f,d,' ',a,' -> ',b);
      Hnx(n-1,6-a-b,b);
    end;
end;
```

```

        end
    else {b cách a qua vị trí c}
    begin
        Hnx(n-1,a,b);
        inc(d);
        writeln(f,d,'. ',a,' -> ',6-a-b);
        Hnx(n-1,b,a);
        inc(d);
        writeln(f,d,'. ',6-a-b,' -> ',b);
        Hnx(n-1,a,b);
    end;
end;
procedure runHnx(n: byte);
begin
    d := 0;
    assign(f,'hnx.out');
    rewrite(f);
    writeln('-----');
    Hnx(n,1,2);
    writeln(f,'Total: ',d,' step(s)');
    close(f);
    readln;
end;
BEGIN
    runHnx(3);
END.

```

Lời gọi runHnx(3) chuyển n tầng tháp từ cọc 1 sang cọc 2 sẽ cho ta kết quả sau:

1. 1 -> 2	9. 3 -> 1
2. 2 -> 3	10. 1 -> 2
3. 1 -> 2	11. 3 -> 1
4. 3 -> 1	12. 2 -> 3
5. 2 -> 3	13. 1 -> 2
6. 1 -> 2	14. 3 -> 1
7. 2 -> 3	15. 1 -> 2
8. 1 -> 2	Total: 15 step(s)

// C#

```

using System;
namespace SangTao1
{
    /*-----
    *                Tháp Ha Noi Xuoi
    * -----*/
    class ThapHaNoiXuoi
    {
        static int d = 0; //so buoc chuyen tang thap
        static void Main()
        {
            Console.WriteLine("\n Ha Noi Xuoi");
            HaNoiXuoi(3, 1, 2);
            Console.WriteLine("\n Total: ")

```

```

        + d + " steps");
    Console.ReadLine();
} // Main
static void HaNoiXuoi(int n, int a, int b)
{
    if (n == 0) return;
    if (b == (a % 3) + 1)
    {
        HaNoiXuoi(n - 1, a, 6 - a - b);
        Console.WriteLine(++d + ". " + a
            + " -> " + b);
        HaNoiXuoi(n - 1, 6 - a - b, b);
    }
    else // a c b, c = 6-a-b
    {
        HaNoiXuoi(n - 1, a, b);
        Console.WriteLine(++d + ". " + a
            + " -> " + (6 - a - b));
        HaNoiXuoi(n - 1, b, a);
        Console.WriteLine(++d + ". " +
            (6 - a - b) + " -> " + b);
        HaNoiXuoi(n - 1, a, b);
    }
}
} // ThapHaNoiXuoi
} // SangTao1

```

Bài 8.7. Tháp Hà Nội ngược

Nội dung giống như bài toán tháp Hà Nội cổ chỉ sửa lại quy tắc (2) như sau: (2) Mỗi lần chỉ được chuyển 1 tầng tháp từ một cọc sang cọc sát nó về hướng ngược chiều kim đồng hồ.

Điều kiện này quy định ba phép chuyển 1 tầng tháp như sau:

② ← ③, hoặc ① ← ②, hoặc ③ ← ①.

Hãy tìm cách giải bài toán với số lần chuyển ít nhất.

Bài giải

Bài này tương tự như bài Hà Nội xuôi. Ta chỉ cần xác định điều kiện kề giữa hai cọc tháp và lưu ý đến chiều chuyển các tầng tháp ngược chiều quay của kim đồng hồ.

a) Trường hợp vị trí b đứng sát vị trí a ngược chiều kim đồng hồ:

Theo trường hợp này ta có các cặp (a, b) sau đây: (3, 2), (2, 1) và (1, 3).

Hoán đổi vị trí của hai đại lượng a và b trong điều kiện kề của bài toán Hà Nội xuôi ta thu được điều kiện kề trong trường hợp này là

$$a = (b \bmod 3) + 1$$

Tình huống	①	②	③
1	a		b
2	b	a	
3		b	a

Tháp Hà Nội ngược

Đặc tả a và b kề nhau $a = (b \bmod 3) + 1$

Với ý nghĩa là, nếu biết mã số vị trí a thì vị trí b sát sau a ngược chiều kim đồng hồ sẽ được tính theo công thức trên.

Dựa vào các hình trạng buộc phải có ta có thể mô tả việc chuyển tháp trong trường hợp này giống như trong bài toán tháp Hà Nội xuôi, cụ thể là:

Hình trạng	Ý nghĩa	Lệnh
(a: [1..n], b: [], c: [])	Hình trạng ban đầu với vị trí b sát vị trí a . $a = (b \bmod 3) + 1$ Để chuyển n tầng từ a sang b ngược chiều kim đồng hồ ta phải...	
...		
(a: [n], b: [], c: [1..n - 1])	Chuyển (tạm) $n - 1$ tầng từ a qua $c = 6 - a - b$.	Hnn($n - 1$, a , $6 - a - b$)
(a: [], b: [n], c: [1..n - 1])	sau đó chuyển tầng còn lại của a qua b .	$a \rightarrow b$
...		
(a: [], b: [1..n], c: [])	Cuối cùng chuyển $n - 1$ tầng từ $c = 6 - a - b$ qua b là hoàn tất.	Hnn($n - 1$, $6 - a - b$, b)

Đoạn trình cho trường hợp này sẽ là

```

if a = (b mod 3)+1 then
  begin {b ke a}
    hnn(n-1, a, 6-a-b);
    inc(d);
    writeln(f,d, ' ', a, ' -> ', b);
    hnn(n-1, 6-a-b, b);
  end..

```

b) Trường hợp vị trí b không đứng sát vị trí a theo chiều ngược kim đồng hồ:

Các cặp (a, b) khi đó sẽ là: (1, 2), (2, 3) và (3, 1). Đặc điểm chung của các cặp này là: nếu đi từ a đến b ngược chiều kim đồng hồ chúng ta phải vượt qua c là vị trí nằm giữa a và b .

Tình huống	①	②	③
1	a	b	
2		a	b
3	b		a

Tháp Hà Nội ngược
Đặc tả a và b không kề nhau
 $a \neq (b \bmod 3) + 1$

Các hình trạng buộc phải có khi đó sẽ rất giống với tình huống tương tự của bài toán Hà Nội xuôi:

Hình trạng	Ý nghĩa	Lệnh
(a: [1..n], c: [], b: [])	Hình trạng ban đầu với vị trí b cách a qua c ngược chiều kim đồng hồ. $a \neq (b \bmod 3) + 1$ Để chuyển n tầng từ a sang b cách qua vị trí c = $6 - a - b$ ta phải...	
...		
(a: [n], c: [], b: [1..n - 1])	Chuyển (tạm) n - 1 tầng từ a qua b.	Hnn(n - 1, a, b)
(a: [], c: [n], b: [1..n - 1])	sau đó chuyển (tạm) tầng còn lại của a qua c = $6 - a - b$.	$a \rightarrow 6 - a - b$
...		
(a: [1..n-1], c: [n], b: [])	Rồi lại chuyển (tạm) n - 1 tầng từ b qua a nhằm giải phóng vị trí đích b...	Hnn(n - 1, b, a)
(a: [1..n-1], c: [], b: [n])	để chuyển tầng lớn nhất n từ c qua b.	$6 - a - b \rightarrow b$
(a: [], c: [], b: [1..n])	Cuối cùng chuyển n - 1 tầng từ a qua b là hoàn tất.	Hnx(n - 1, a, b)

(* Pascal *)

```

(*****
Hano.pas - Hà Nội Ngược
Chuyển pháp ngược chiều kim đồng hồ.
*****)
uses crt;
var d: longint;
f: text;
procedure hnn(n,a,b: byte);
begin
  if n = 0 then exit;
  if a = (b mod 3)+1 then
    begin {b ke a}
      hnn(n-1,a,6-a-b);
      inc(d);
      writeln(f,d,'. ',a,' -> ',b);
      hnn(n-1,6-a-b,b);
    end
  else {b cách a qua vi tri c}
    begin
      hnn(n-1,a,b);
      inc(d);
      writeln(f,d,'. ',a,' -> ',6-a-b);
      hnn(n-1,b,a);
      inc(d);
      writeln(f,d,'. ',6-a-b,' -> ',b);
      hnn(n-1,a,b);
    end;
end;
procedure runhnn(n: byte);
begin
  d := 0;

```

```

    assign(f, 'hnn.out');
    rewrite(f);
    writeln('-----');
    hnn(n,1,2);
    writeln(f, 'Total: ', d, ' step(s)');
    close(f);
    readln;
end;
BEGIN
    runHnn(3);
END.

```

Kết quả:

1. 1 -> 3	12. 3 -> 2
2. 3 -> 2	13. 2 -> 1
3. 1 -> 3	14. 3 -> 2
4. 2 -> 1	15. 1 -> 3
5. 3 -> 2	16. 3 -> 2
6. 1 -> 3	17. 1 -> 3
7. 3 -> 2	18. 2 -> 1
8. 1 -> 3	19. 3 -> 2
9. 2 -> 1	20. 1 -> 3
10. 1 -> 3	21. 3 -> 2
11. 2 -> 1	Total: 21 step(s)

Nhận xét

Mới xem ta có cảm tưởng rằng lời gọi $Hnn(3, 1, 2)$ và $Hnx(3, 1, 2)$ để chuyển tháp 3 tầng từ cọc 1 sang cọc 2 phải cho cùng một số bước chuyển các tầng là 15. Tuy nhiên, lời gọi $Hnn(3, 1, 2)$ cho ta 21 bước chuyển các tầng, trong khi lời gọi $Hnx(3, 1, 2)$ chỉ cần 15 bước chuyển các tầng.

Suy ngẫm một chút bạn sẽ giải thích được nghịch lý này.

Hãy thử gọi Hà Nội ngược để chuyển tháp 3 tầng từ cọc 3 sang cọc 2:

$Hnn(3, 3, 2)$

Ta sẽ thấy chỉ cần 15 bước!!!

Lại gọi Hà Nội xuôi để chuyển tháp 3 tầng từ cọc 1 sang cọc 3:

$Hnx(3, 1, 3)$

Ta lại thấy 21 bước.

Như vậy, Hnx và Hnn là đối xứng lệch. Nếu hai cọc, nguồn và đích kề nhau thì số lần chuyển tháp 3 tầng sẽ là 15. Ngược lại, khi hai cọc đó không kề nhau thì số lần chuyển tháp 3 tầng sẽ là 21. Hai cọc 1 và 2 là kề nhau đối với tháp Hà Nội xuôi nhưng không kề nhau đối với tháp Hà Nội ngược. Tương tự, hai cọc 3 và 2 là kề nhau đối với tháp Hà Nội ngược nhưng không kề nhau đối với tháp Hà Nội xuôi.

Ta nhận xét rằng: nếu lấy hai số a, b khác nhau bất kì trong ba số 1, 2 và 3 thì giữa a và b chỉ xảy ra một trong hai trường hợp loại trừ nhau sau đây:

$$b = (a \bmod 3) + 1, \text{ hoặc } a = (b \bmod 3) + 1$$

Do đó, quan hệ kề nhau trong hai bài toán Tháp Hà Nội xuôi và ngược là phủ định đối với nhau.

	Hà Nội xuôi	Hà Nội ngược
$b = (a \bmod 3) + 1$	a và b kề nhau	a và b không kề nhau
$a = (b \bmod 3) + 1$	a và b không kề nhau	a và b kề nhau

*Quan hệ kề nhau trong hai bài toán
tháp Hà Nội xuôi và ngược*

// C#

```
using System;
namespace SangTao1
{
    /*-----
    *          Tháp Ha Noi Nguoc
    * -----*/
    class ThapHaNoiNguoc
    {
        static int d = 0;
        static void Main()
        {
            Console.WriteLine("\n Ha Noi Nguoc ");
            HaNoiNguoc(3, 1, 2);
            Console.WriteLine("\n Total: " + d
                               + " steps");
            Console.ReadLine();
        } // Main
        static void HaNoiNguoc(int n, int a, int b)
        {
            if (n == 0) return;
            if (a == (b % 3) + 1)
            {
                HaNoiNguoc(n - 1, a, 6 - a - b);
                Console.WriteLine(++d + ". " +
                                   a + " -> " + b);
                HaNoiNguoc(n - 1, 6 - a - b, b);
            }
            else // b c a, c = 6-a-b
            {
                HaNoiNguoc(n - 1, a, b);
                Console.WriteLine(++d + ". " + a +
                                   " -> " + (6 - a - b));
                HaNoiNguoc(n - 1, b, a);
                Console.WriteLine(++d + ". " +
                                   (6 - a - b) + " -> " + b);
                HaNoiNguoc(n - 1, a, b);
            }
        }
    } // ThapHaNoiNguoc
} // SangTao1
```

Bài 8.8. Tháp Hà Nội thẳng

Nội dung giống như bài toán tháp Hà Nội cổ chỉ sửa lại quy tắc (2) như sau: (2) Mỗi lần chỉ được chuyển 1 tầng tháp từ một cọc sang cọc kề nó, không được vòng từ 3 sang 1 hay 1 sang 3.

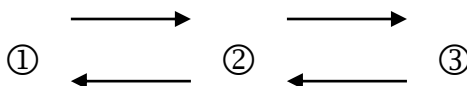
Điều kiện này quy định bốn phép chuyển 1 tầng tháp như sau:

$$\textcircled{1} \rightarrow \textcircled{2}, \textcircled{2} \rightarrow \textcircled{1}, \textcircled{2} \rightarrow \textcircled{3}, \textcircled{3} \rightarrow \textcircled{2}$$

hoặc, theo cách biểu diễn khác:

$$\textcircled{1} \leftrightarrow \textcircled{2} \leftrightarrow \textcircled{3}$$

tức là chỉ được chuyển qua lại giữa hai cọc kề nhau. Giả thiết là các cọc được sắp thành hàng như sau:



Hãy tìm cách giải bài toán với số lần chuyển ít nhất.

Bài giải

Giống như đã phân tích trong các bài toán Hà Nội trước, ta có:

- Hình trạng xuất phát: $(a:[1..n], b:[], c:[])$
- ...
- Hình trạng kết thúc: $(a:[], b:[1..n], c:[])$
- Hình trạng buộc phải có: $(a:[n], b:[], c:[1..n-1])$

Ta phân biệt hai trường hợp:

- Hai cọc a và b đứng kề nhau trên đường thẳng.
- Hai cọc a và b cách nhau qua c .

Trường hợp thứ nhất Nếu vị trí các cọc được bố trí như sau

$$\textcircled{1} \qquad \qquad \textcircled{2} \qquad \qquad \textcircled{3}$$

thì giữa a và b có bốn tình huống, cụ thể là:

Tình huống	①	②	③
1	a	b	
2	b	a	
3		a	b
4		b	a

Tháp Hà Nội thẳng

Đặc tả a và b kề nhau $\text{abs}(a - b) = 1$

Trường hợp này được đặc tả là

$$\text{abs}(a - b) = 1$$

Hình trạng

$(a:[1..n], b:[], c:[])$

Ý nghĩa

Hình trạng ban đầu với vị trí b kề vị trí a trên đường thẳng.

$$\text{abs}(a - b) = 1$$

Lệnh

Để chuyển n tầng từ a sang b
theo đường thẳng ta phải...

...	
(a: [n], b: [], c: [1..n - 1])	Chuyển (tạm) $n - 1$ tầng từ a qua $c = 6 - a - b$; $\text{Hnt}(n - 1, a, 6 - a - b)$;
(a: [], b: [n], c: [1..n - 1])	sau đó chuyển tầng còn lại của $a \rightarrow b$ qua b .
...	
(a: [], b: [1..n], c: [])	Cuối cùng chuyển $n - 1$ tầng từ $c = 6 - a - b$ qua b là hoàn tất. $\text{Hnt}(n - 1, 6 - a - b, b)$;

Trường hợp thứ hai a và b cách nhau qua c trên đường thẳng. Ta có $c = 2$ và chỉ có hai tình huống cho a và b như sau:

Tình huống	①	②	③
1	a	c	b
2	b	c	a

Hình trạng	Ý nghĩa	Lệnh
(a: [1..n], c: [], b: [])	Hình trạng ban đầu với vị trí b không kề với vị trí a trên đường thẳng. $\text{abs}(a - b) \neq 1$ Ta có $c = 2$. Để chuyển n tầng từ a sang b cách qua vị trí $c = 2$ ta phải...	
(a: [n], c: [], b: [1..n - 1])	Chuyển (tạm) $n - 1$ tầng từ a qua b .	$\text{Hnt}(n - 1, a, b)$
(a: [], c: [n], b: [1..n - 1])	sau đó chuyển (tạm) tầng cuối của a qua $c = 2$.	$a \rightarrow 2$
(a: [1..n-1], c: [n], b: [])	Rồi lại chuyển (tạm) $n - 1$ tầng từ b qua a nhằm giải phóng vị trí đích b .	$\text{Hnt}(n - 1, b, a)$
(a: [1..n-1], c: [], b: [n])	để chuyển tầng lớn nhất n từ $c = 2$ qua b .	$2 \rightarrow b$
(a: [], c: [], b: [1..n])	Cuối cùng chuyển $n - 1$ tầng từ a qua b là hoàn tất.	$\text{Hnt}(n - 1, a, b)$

(* Pascal *)

```

(*****
HNT.PAS - Ha Noi thang
Chuyen n tang thap tu coc a
sang coc b theo duong thang
1->2, 2->1 hoac 2->3, 3->2
  1 2 3
*****)

```

```

uses crt;
var d: longint;
f: text;
procedure Hnt(n,a,b: byte);
begin
  if n = 0 then exit;
  if abs(a-b) = 1 then
    begin
      Hnt(n-1,a,6-a-b);
      inc(d);
      writeln(f,d,'. ',a,' -> ',b);
      Hnt(n-1,6-a-b,b);
    end
  else
    {-----}
    abs(a-b)=2 tuc la a = 1, b = 3
    hoac a = 3, b = 1, do do c=2
    {-----}
    begin
      Hnt(n-1,a,b);
      inc(d);
      writeln(f,d,'. ',a,' -> 2');
      Hnt(n-1,b,a);
      inc(d);
      writeln(f,d,'. 2 -> ',b);
      Hnt(n-1,a,b);
    end;
end;
procedure runHnt(n: byte);
begin
  d := 0;
  assign(f,'hnt.out');
  rewrite(f);
  writeln('-----');
  Hnt(n,1,2);
  writeln(f,'Total: ',d,' step(s)');
  close(f);
  readln;
end;
BEGIN
  runHnt(3);
END.

```

Kết quả

1. 1 -> 2
2. 2 -> 3
3. 1 -> 2
4. 3 -> 2
5. 2 -> 1
6. 2 -> 3
7. 1 -> 2
8. 2 -> 3
9. 1 -> 2
10. 3 -> 2

```

11. 2 -> 1
12. 3 -> 2
13. 1 -> 2

```

Total: 13 step(s)

// C#

```

using System;
namespace SangTao1
{
    /*-----
    *          Thap Ha Noi Thang
    * -----*/
    class ThapHaNoiThang
    {
        static int d = 0;
        static void Main()
        {
            Console.WriteLine("\n Ha Noi Thang");
            HaNoiThang(3, 1, 2);
            Console.WriteLine("\n Total: " +
                              d + " steps");
            Console.ReadLine();
        } // Main
    } /*-----
        Ha Noi Thang
        * -----*/
    static void HaNoiThang(int n, int a, int b)
    {
        if (n == 0) return;
        if (Math.Abs(a - b) == 1)
        {
            HaNoiThang(n - 1, a, 6 - a - b);
            Console.WriteLine(++d +
                              ". "+a+" -> "+b);
            HaNoiThang(n - 1, 6 - a - b, b);
        }
        else // a c b, b c a, c = 6-a-b
        {
            HaNoiThang(n - 1, a, b);
            Console.WriteLine(++d +
                              ". "+a+" -> "+(6-a-b));
            HaNoiThang(n - 1, b, a);
            Console.WriteLine(++d +
                              ". "+(6-a-b)+" -> "+b);
            HaNoiThang(n - 1, a, b);
        }
    }
} // ThapHaNoiThang
} // SangTao1

```

Bài 8.9. Tháp Hà Nội sắc màu (Hà Nội Cầu vồng)

Người ta sơn mỗi tầng tháp một màu và quy định luật chuyển cho mỗi loại tầng theo màu như mô tả trong bảng sau.

Kí hiệu	Màu	Ý nghĩa chuyển tầng	Quy tắc
x	<u>x</u> anh	<u>x</u> uôi chiều kim đồng hồ	① → ② → ③ → ①
n	<u>n</u> âu	<u>n</u> gược chiều kim đồng hồ	① ← ② ← ③ ← ①
t	<u>t</u> rắng	<u>t</u> hăng	① ↔ ② ↔ ③
h	<u>h</u> ồng	tự do (<u>h</u> à nội kinh điển)	① ↔ ② ↔ ③ ↔ ①

Ngoài ra, dĩ nhiên vẫn phải theo quy định là tầng to không được đặt lên trên tầng nhỏ.

Hãy tìm cách giải bài toán với số lần chuyển ít nhất.

Thí dụ, với các tầng tháp được tô màu từ trên (tầng nhỏ nhất) xuống dưới (tầng lớn nhất) là:



Hà Nội sắc màu 5 tầng $xnxt$

và cần chuyển tháp từ cọc 1 sang cọc 2 thì phải thực hiện tối thiểu 31 lần chuyển các tầng như sau:

- | | |
|------------|-------------------|
| 1. 1 -> 2 | 17. 3 -> 1 |
| 2. 1 -> 3 | 18. 3 -> 2 |
| 3. 2 -> 3 | 19. 1 -> 2 |
| 4. 1 -> 2 | 20. 3 -> 1 |
| 5. 3 -> 1 | 21. 2 -> 3 |
| 6. 3 -> 2 | 22. 2 -> 1 |
| 7. 1 -> 2 | 23. 3 -> 1 |
| 8. 1 -> 3 | 24. 3 -> 2 |
| 9. 2 -> 3 | 25. 1 -> 2 |
| 10. 2 -> 1 | 26. 1 -> 3 |
| 11. 3 -> 1 | 27. 2 -> 3 |
| 12. 2 -> 3 | 28. 1 -> 2 |
| 13. 1 -> 2 | 29. 3 -> 1 |
| 14. 1 -> 3 | 30. 3 -> 2 |
| 15. 2 -> 3 | 31. 1 -> 2 |
| 16. 1 -> 2 | Total: 31 step(s) |

Bài giải

Điều lí thú là thủ tục Hà Nội sắc màu là khá tổng quát vì ta có thể dùng nó để gọi cho các bài toán về tháp Hà Nội đã xét. Bảng dưới đây cho biết cách sử dụng thủ tục Hnm cho các trường hợp riêng.

Muốn gọi	Thì gọi	Chú thích
Hn (n, a, b)	s := 'hh...h' ; Hnm (length (s) , a , b) ;	s chứa n kí tự 'h'
Hnx (n, a, b)	s := 'xx...x' ; Hnm (length (s) , a , b) ;	s chứa n kí tự 'x'

```

Hnn(n,a,b)  s := 'nn...n';           s chứa n kí tự 'n'
             Hnm(length(s),a,b);

Hnt(n,a,b)  s := 'tt...t';           s chứa n kí tự 't'
             Hnm(length(s),a,b);

```

Ta quy ước dữ liệu ban đầu được mô tả trong biến tổng thể kiểu xâu kí tự s với khai báo như sau:

```
var s: string
```

Trong thí dụ trên, s sẽ được gán trị là

```
s := 'xnxht';
```

Khi đó có thể khai báo thủ tục tháp Hà Nội sắc màu như sau:

```
procedure Hnm(n: byte; a,b: byte);
```

trong đó n là số tầng tháp, a và b là hai cọc khác nhau cho trước và nhận các giá trị 1, 2 hoặc 3.

Ta viết thêm thủ tục **runHnm(Thap: string)** để tổ chức lời gọi thuận tiện cho bài toán Hà Nội sắc màu như sau.

Tham biến **Thap** kiểu **string** sẽ chứa mô tả cụ thể cho các tầng tháp. Thí dụ, lời gọi

```
runHnm('xnxht');
```

sẽ xử lí tháp 5 tầng, tính từ trên xuống, tức là từ tầng nhỏ nhất có mã số 1 đến tầng đáy, tầng lớn nhất mang mã số 5 như sau:

Tầng (i)	Màu (Thap[i])
1	'x'
2	'n'
3	'x'
4	'h'
5	't'

Gọi thủ tục cho bài

Hà Nội Sắc màu

```
runHnm('xnxht');
```

Cách mã số này ngược với quy tắc gọi các tầng trong đời thường. Người ta thường mã số các tầng của toà nhà từ dưới lên là 1, 2,...

Với lời gọi

```
runHnm(Thap:string);
```

thì giá trị của tham trị **Thap** sẽ được truyền cho một biến tổng thể s:

```
s := Thap;
```

Sau đó là lời gọi cụ thể **Hnm** theo số tầng tháp **length(s)**, cọc nguồn (nơi đặt tầng tháp ban đầu) **a = 1** và cọc đích, nơi cần chuyển tháp đến, **b = 2**.

```
Hnm(length(s),1,2);
```

Sở dĩ phải dùng một biến tổng thể s để lưu lại cấu hình tháp vì ta muốn hạn chế tổn kém phát sinh trong lời gọi đệ quy của thủ tục **Hnm**.

Nếu khai báo Hnm với bốn tham biến như sau:

```
procedure Hnm(Thap: string; n,a,b: byte);
```

thì rõ ràng là sẽ tốn kém hơn.

Biến tổng thể d được khởi trị 0 sẽ được dùng để đếm số bước chuyển các tầng tháp.

```
procedure runHnm(Thap:string);
begin
  s := Thap;
  d := 0;
  assign(f, 'hnm.out');
  rewrite(f);
  writeln('-----');
  Hnm(length(s), 1, 2);
  writeln(f, 'Total: ', d, ' step(s)');

  close(f);
  readln;
end;
BEGIN

  runHnm('txhxn');

END.
```

Mỗi khi xử lý một tầng tháp i , ta xem màu $s[i]$ của tầng và lựa chọn quyết định như trong bảng sau

Ta nhận xét rằng, trong mọi tình huống, tức là với mọi màu khác nhau của tầng tháp, khi hai cọc a và b kề nhau thì ta cần thực hiện các thao tác như nhau, cụ thể là:

Nếu $s[i]$ có màu	Thì xử lý như thủ tục
'x'	hnx
'n'	hnn
't'	hnt
'h'	hn

Phương thức xử lý cho bài toán

Hà Nội sắc màu

Vậy ta chỉ cần đặc tả tính chất kết giữa hai cọc a và b là xong.

Ta thấy dựa theo luật chuyển với tháp Hà Nội cổ, hai cọc a và b khác nhau tùy ý được xem là kề nhau. Với tháp Hà Nội xuôi, cọc b phải đứng sát cọc a theo chiều kim đồng hồ:

$a \rightarrow b$

Chuyển tầng cuối cùng từ a sang

$b \quad b = (a \bmod 3) + 1$

$Hnm(n-1, 6-a-b, b)$

Chuyển $n-1$ tầng tháp từ $c = 6-a-b$ qua b

Hà Nội sắc màu

Chuyển tháp trong trường hợp a và b kề nhau

phải đứng sát b theo chiều kim đồng hồ:

$a = (b \bmod 3) + 1$

Với tháp Hà Nội thẳng, như ta đã biết, giá trị của a và b phải lệch nhau đúng 1 đơn vị:

Với tháp Hà Nội ngược, b phải đứng sát a theo chiều quay ngược của kim đồng hồ. Nói cách khác, a

$$\text{abs}(a-b) = 1$$

Bảng dưới đây mô tả các trường hợp trên.

Bài toán	Kí hiệu	Điều kiện a kề b ($a \neq b$)	Minh hoạ
Hà Nội Cổ	h	Luôn đúng (True)	①↔②↔③↔① ①②, ②①, ①③, ③① ②③, ③②
Hà Nội Xuôi	x	$b = (a \bmod 3) + 1$	① → ② → ③ → ① ①②, ②③, ③①
Hà Nội Ngược	n	$a = (b \bmod 3) + 1$	① ← ② ← ③ ← ① ①③, ③②, ②①
Hà Nội Thẳng	t	$\text{abs}(a - b) = 1$	① ↔ ② ↔ ③ ①②, ②①, ②③, ③②

Hà Nội sắc màu

Các điều kiện kề giữa hai cọc a và b

a, b = 1, 2, 3; $a \neq b$

Hàm Ke (kề) khi đó sẽ được mô tả như sau.

```
function Ke(n,a,b: byte): Boolean;
begin
  case s[n] of
    'x': ke := (b = (a mod 3)+1);
    'n': ke := (a = (b mod 3)+1);
    't': ke := (abs(a-b) = 1);
    'h': ke := True;
  end {case};
end;
```

Tương tự, khi hai cọc a và b không kề nhau ta cũng thực hiện các thao tác như nhau.

Biết hàm Ke, ta dựa vào các thuật toán riêng cho mỗi trường hợp để viết phương án cho bài toán Hà Nội sắc màu như sau:

```
(*-----
Ha Noi sac mau
x: xanh - xuai chieu kim dong ho
n: nau - nguc chieu kim dong ho
t: trang - chuyen thang
h: hong - chuyen tu do
-----*)

procedure Hnm(n: byte; a,b: byte);
begin
  if n = 0 then exit;
  if Ke(n,a,b) then
    begin
      Hnm(n-1,a,6-a-b);
      inc(d);
      writeln(f,d,'. ',a,' -> ',b);
      Hnm(n-1,6-a-b,b);
    end else
    begin
```

```

        Hnm(n-1,a,b);
        inc(d);
        writeln(f,d,'. ',a,' -> ',6-a-b);
        Hnm(n-1,b,a);
        inc(d);
        writeln(f,d,'. ',6-a-b,' -> ',b);
        Hnm(n-1,a,b);
    end;
end;

(* Pascal *)

(*****
HNM.PAS - Thỏp Hà Nội màu
x - xanh: Xuôi chiều kim đồng hồ
n - nâu: Ngược chiều kim đồng hồ
t - Trắng: thắng theo hàng ngang
h - Hà Nội cổ: kinh điển
*****)
uses crt;
var d: longint; {dem so buoc chuyen}
f: text; {output file}
s: string; {cau hinh thap}
{-----}
Kiem tra tinh chat ke giua 2 coc a va b
{-----}
function Ke(n,a,b: byte): Boolean;
begin
    case s[n] of
        'x': ke := (b = (a mod 3)+1);
        'n': ke := (a = (b mod 3)+1);
        't': Ke := (abs(a-b) = 1);
        'h': Ke := True;
    end {case};
end;
(*-----*)
Ha Noi sac mau
x: xanh - xuai chieu kim dong ho
n: nau - nguoc chieu kim dong ho
t: trang - chuyen thang
h: hong - chuyen tu do
-----*)
procedure Hnm(n: byte; a,b: byte); tự viết
{-----}
To chuc goi thu tuc Hnm
{-----}
procedure runHnm(Thap:string);
begin
    s := Thap;
    d := 0;
    assign(f,'hnm.out');
    rewrite(f);
    writeln('-----');

```

```

Hnm(length(s),1,2);
writeln(f,'Total: ',d,' step(s)');
close(f);
readln;
end;
BEGIN
    runHnm('txhxn');
END.

```

// C#

```

using System;
namespace SangTao1
{
    /*-----
    *                Thap Ha Noi
    * -----*/
    class ThapHaNoi
    {
        static int d = 0;
        static char[] s = new char[64];
        static void Main()
        {
            Console.WriteLine("\n Ha Noi Sac Mau");
            RunHaNoiSacMau("xnxht", 1, 2);
            Console.WriteLine("\n Total: " +
                               d + " steps");
            Console.ReadLine();
        } // Main
        static bool Ke(char KieuThap, int a, int b)
        {
            switch (KieuThap)
            {
                case 'x': return (b == (a % 3) + 1);
                case 'n': return (a == (b % 3) + 1);
                case 't': return (Math.Abs(a - b) == 1);
            }
            return true;
        }
    }
    /*-----
    Ha Noi sac mau
    x: xanh - xuoi chieu kim dong ho
    n: nau - nguoc chieu kim dong ho
    t: trang - chuyen thang
    h: hong - chuyen tu do
    -----*/
    void HaNoiSacMau(int n, int a, int b)
    {
        if (n == 0) return;
        if (Ke(s[n], a, b))
        {
            HaNoiSacMau(n - 1, a, 6 - a - b);

```

```

        Console.WriteLine(++d) +
        ". (" + s[n] + ") " + a + " -> " + b);
        HaNoiSacMau(n - 1, 6 - a - b, b);
    }
    else
    {
        HaNoiSacMau(n - 1, a, b);
        Console.WriteLine(++d) +
            ". (" + s[n] + ") " +
            a + " -> " + (6 - a - b));
        HaNoiSacMau(n - 1, b, a);
        Console.WriteLine(++d) +
            ". (" + s[n] + ") " +
            (6 - a - b) + " -> " + b);
        HaNoiSacMau(n - 1, a, b);
    }
}
static void RunHaNoiSacMau(string w,
                           int a, int b)
{
    d = 0;
    w.CopyTo(0, s, 1, w.Length);
    HaNoiSacMau(w.Length, a, b);
}
} // ThapHaNoi
} // SangTao1

```

Hướng dẫn kiểm thử

Ta sẽ dùng kỹ thuật đối sánh để kiểm thử các trường hợp.

Ta chọn và cố định các giá trị n , a và b . Chẳng hạn, $n = 4$, $a = 1$, $b = 2$.

Khi đó ta có:

RunHn(n) và **RunHnm**('hhhh') cho cùng kết quả.

RunHnx(n) và **RunHnm**('xxxx') cho cùng kết quả.

RunHnn(n) và **RunHnm**('nnnn') cho cùng kết quả.

RunHnt(n) và **RunHnm**('tttt') cho cùng kết quả.

Nếu ghi các kết quả này vào các tệp tương ứng, sau đó gọi thủ tục đối sánh từng cặp hai tệp tương ứng thì có thể kiểm định được một số trường hợp.

Để xây dựng các tình huống kiểm thử khác nhau còn lại bạn để ý đến các tính chất thuận nghịch của các thủ tục khác nhau. Thí dụ, như đã phân tích ở phần trên, các lời gọi **Hnx**(3, 1, 2) và **Hnn**(3, 3, 1) phát sinh cùng một số lần chuyển.

Bạn hãy thử phát hiện thêm sự tương đồng giữa các lời gọi Hnm với các tham trị khác nhau.

Đọc thêm

LƯỢC SỬ

Một số bài toán về tháp Hà Nội đã được đưa vào các kì thi Olympic Tin học tại một số quốc gia. Chúng ta thử tìm hiểu cội nguồn của các bài toán thuộc loại này.

Năm 1883, trên một tờ báo ở Paris có đăng bài mô tả một trò chơi toán học của giáo sư Claus với tên là Tháp Hà Nội. Nội dung trò chơi được mọi người say mê làm thử chính là bài toán Tháp Hà Nội cổ.

Thời đó ở thủ đô Paris dân chúng đổ xô nhau mua đồ chơi này và suốt ngày ngồi chuyển tháp. Trong lịch sử về các trò chơi thông minh đã từng có những cơn sốt như vậy. Tính trung bình mỗi thế kỉ có một vài cơn sốt trò chơi. Thế kỉ thứ XX có cơn sốt Rubic, thế kỉ XIX là các trò chơi 15 và tháp Hà Nội. Bài toán này nổi tiếng đến mức trở thành kinh điển trong các giáo trình về thuật giải đệ quy và được trình bày trong các thông báo chính thức của các phiên bản chuẩn của các ngữ trình như ALGOL-60, ALGOL-68, Pascal, Delphy, C, C++, Ada,... khi muốn nhấn mạnh về khả năng đệ quy của các ngôn ngữ đó.

Theo nhà nghiên cứu Henri De Parville công bố vào năm 1884 thì tác giả của trò chơi tháp Hà Nội có tên thật là nhà toán học Eduard Lucas, người có nhiều đóng góp trong lĩnh vực số luận. Mỗi khi viết về đề tài giải trí thì ông đổi tên là Claus. Bạn có để ý rằng Claus là một hoán vị các chữ cái của từ Lucas.

De Parville còn kể rằng bài toán tháp Hà Nội bắt nguồn từ một tích truyền kì ở Ấn Độ. Một nhóm cao tăng Ấn Độ giáo được giao trọng trách chuyển dần 64 đĩa vàng giữa ba cọc kim cương theo các điều kiện đã nói ở bài toán Tháp Hà Nội cổ. Khi nào hoàn tất công việc, tức là khi chuyển xong toà tháp vàng 64 tầng từ vị trí ban đầu sang vị trí kết thúc thì cũng là thời điểm tận thế. Sự việc này có xảy ra hay không ta sẽ xét ở bài tập 8.10.

Lời giải được công bố đầu tiên cho bài toán tháp Hà Nội là của Allardice và Frase, năm 1884.

Năm 1994 David G. Poole ở Đại học Trent, Canada đã viết một bài khảo cứu cho tờ Mathematics Magazine số tháng 12 nhan đề "Về các tháp và các tam giác của giáo sư Claus" cùng với một phụ đề "Pascal biết Hà Nội". Poole đã liệt kê 65 công trình khảo cứu bài toán tháp Hà Nội đăng trên các tạp chí toán-tin trong khoảng mười năm. Tác giả cũng chỉ ra sự liên quan giữa các công thức tính số lần chuyển các tầng tháp và một phương pháp quen biết dùng để tính các hệ số của dạng khai triển nhị thức Newton $(a + b)^n$. Phương pháp này được gọi là Tam giác Pascal, mang tên nhà toán học kiêm vật lí học Pháp Blaise Pascal (1623-1662), người đã chế tạo chiếc máy tính quay tay đầu tiên trên thế giới.

Một số nhà nghiên cứu trong và ngoài nước có bàn luận về địa danh Hà Nội. Theo tôi vấn đề này vẫn còn ngò. Hầu hết các bài viết xoay quanh đề tài chuyển tháp nói trên đều dùng thuật ngữ bài toán tháp Hà Nội. Khi giới thiệu về bài toán Hà Nội nhiều tháp Dudeney đặt tên là bài toán đồ của Reve (The Reve's Puzzle). Tuy nhiên, nhiều nhà nghiên cứu cho rằng tốt hơn cả là nên đặt tên và phân loại theo tên nguyên thủy của bài toán, nghĩa là Tháp Hà Nội.

Ngoài các dạng Tháp Hà Nội đã liệt kê ở phần trên một số tác giả còn đề xuất những dạng khá kì lạ, chẳng hạn như bài toán sau đây.

Hà Nội nhiều tháp

Trong trò chơi này người ta làm thêm những cọc, chẳng hạn thay vì ba ta dùng bốn cọc và cũng có thể bố trí tháp tại nhiều cọc. Ý kiến này do H.E. Dudeney, một tác giả hàng đầu về toán học giải trí người Anh đưa ra vào năm 1908. Đã có nhiều bài đăng lời giải cho bài toán này, có những bài mới xuất hiện gần đây vào những năm 1988 và 1989. Dù vậy chưa ai chứng minh được rõ ràng số lần chuyển của bài giải là tối thiểu như đã làm với các dạng tháp Hà Nội khác.

Bài tập

Bạn hãy thử lập công thức tính số lần chuyển các tầng tối thiểu cho các bài toán sau:

Tháp Hà Nội,
Tháp Hà Nội Xuôi,
Tháp Hà Nội Ngược và
Tháp Hà Nội Thăng.

Lời cảm ơn Các tư liệu trên và một số tư liệu khác trong bài được trích dẫn từ các bài viết của giáo sư viện sĩ Nguyễn Xuân Vinh, Khoa Kỹ thuật không gian, Đại học Michigan, cộng tác viên NASA, Hoa Kỳ. Tác giả xin chân thành cảm ơn giáo sư đã cho phép trích dẫn và chỉ giáo về các phương pháp truyền thụ tri thức khoa học cho giới trẻ.

NXH

8/4/2008

Sửa ngày 4/4/09

Nguyễn Xuân Huy
 s,ng t¹o trong
 thuËt to,n
 vụ lÛp tr×nh
 vớì C#, Pascal

tuyển c,c bùi to,n tin nòng cao
 cho hãc sinh vụ sinh vi^an giãì

TÛp mét

Lời giớì thiÖu

*S, ch tr×nh bùy cã hõ thèng c,c ph--ng ph.p thiÖt
 kÕ thuËt to,n minh hãa qua c,c bùi to,n thi hãc
 sinh giãì vụ Olympic hãc sinh vụ sinh vi^an trong
 n-íc, khu vùc vụ quèc tÕ. C,c bùi to,n ®Òu ®-íc
 ph©n tÝch ®Çy ®ñ kìm theo thuËt to,n vụ toùn v"n
 ch--ng tr×nh viÖt tr^an ng«n ng÷ C# vụ Pascal.
 S, ch lụ tụi liÖu bæ Ých cho hãc sinh trung hãc,
 gi,o vi^an c,c tr-êng phæ th«ng vụ cao ®¹ng vụ
 sinh vi^an c,c tr-êng ®¹i hãc muèn hoiùn thiÖn kiÖn
 thøc ®Ó tham dù c,c kú thi Olympic Tin hãc quèc
 gia vụ quèc tÕ.
 C,c ch--ng tr×nh song ng÷ Pascal vụ C# gióp cho
 b¹n ®ãc chuyón ®æi nhanh chãng sang c,c m«i
 tr-êng lÛp tr×nh ti^an tiÖn.*