

# 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

# MỤC LỤC

	<b>Lời nói đầu</b>	i
<b>Chương I</b>	<b>GIẢI MỘT BÀI TOÁN TIN</b>	<b>1</b>
<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
<b>Chương II</b>	<b>SINH DỮ LIỆU VÀO VÀ RA</b>	<b>27</b>
<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 tệp tăng</i>	40
<i>Bài 2.7.</i>	<i>Sinh ngẫu nhiên tệp 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>Tệp các hoán vị</i>	49
<i>Bài 2.11.</i>	<i>Đọc dữ liệu từ tệp vào mảng biết hai kích thước</i>	53
<i>Bài 2.12.</i>	<i>Đọc dữ liệu từ tệp vào mảng biết một kích thước</i>	56
<i>Bài 2.13.</i>	<i>Đọc dữ liệu từ tệp 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
<b>Chương III</b>	<b>BÀN PHÍM VÀ MÀN HÌNH</b>	<b>79</b>
<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
<b>Chương IV</b>	<b>TỔ CHỨC DỮ LIỆU</b>	<b>107</b>
<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
<b>Chương V</b>	<b>PHƯƠNG PHÁP THAM LAM</b>	<b>153</b>
<i>Bài 5.1.</i>	<i>Bảng 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
<b>Chương VI</b>	<b>PHƯƠNG PHÁP QUAY LUI</b>	<b>193</b>
<i>Bài 6.1.</i>	<i>Tâm Hậu</i>	195
<i>Bài 6.2.</i>	<i>Tù chuẩn</i>	207
<i>Bài 6.3.</i>	<i>Tìm đường trong mê cung</i>	216
<b>Chương VII</b>	<b>QUY HOẠCH ĐỘNG</b>	<b>227</b>
<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
<b>Chương VIII</b>	<b>SUY NGÂM</b>	<b>267</b>
<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 đa trong ma trận 0/1</i>	281
<i>Bài 8.4.</i>	<i>Ma phượng</i>	291
<i>Bài 8.5.</i>	<i>Tháp Hà Nội cô</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 vồng)</i>	325

## Lời nói đầu

*Thể 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

té, một phương pháp vạn năng như vậy là không hữu hiệu. Tuỳ 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, Bản 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 Thành - 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 bài toán rồi thử giải các thí dụ đó để hình thành dần những *hướng dẫn* 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 kĩ 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 thuận thiện

*Tìm tất cả các số tự nhiên hai chữ số mà 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:

$$(1) \quad x = ab.$$

$$(2) \quad a, b = 0..9 (a và b biến thiên trong khoảng 0..9).$$

$$(3) \quad a > 0 \text{ vì } x \text{ là số có hai chữ số.}$$

$$(4) \quad (ab, ba) = 1.$$

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

$$(5) \quad x = 10..99 (x biến thiên từ 10 đến 99, vì  $x$  là số có hai chữ số).$$

$$(6) \quad (x, x') = 1.$$

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 thoả các tính chất sau:  $x = 10..99$  ( $x$  nằm trong khoảng từ 10 đến 99).*

$$(7) \quad x' = \text{Đơn}(x)*10 + \text{Chục}(x).$$

$$(8) \quad (x, x') = 1 \text{ (ước chung lớn nhất của } x \text{ và } x' \text{ bằng } 1).$$

Đặc tả trên được thể hiện qua ngôn ngữ编程 trình tự 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 minh đặc tả (10):

**ucln(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;

```

$Don(x) = (x \bmod 10)$ : số dư của phép chia nguyên  $x$  cho 10, thí dụ:

**Đơn(19) = 19 mod 10 = 9.**

$Chuc(x) = (x \bmod 10)$ : 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. Để 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 thỏa điều kiện đầu bài,  $n$  là số lượng các số tìm được.

*Bước 2.* Hiển 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 thi **{\$B-}** trong chương trình NTCN (nguyên tố cùng nhau) dưới đây đặt chế độ kiểm tra biểu thức lôgic 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 lôgic 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 thi **{\$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
    { ghep 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 thoai 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  $\text{Ucln}(x, x) = x \geq 10 \neq 1$ .

Nếu  $b = 0$  ta có  $x = 10a$  và  $x' = a$ . Ta thấy  $\text{Ucln}(10a, a) = a = 1$  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  $b$  thoả 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:  $x = 100a + 10b + c$ .
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 dây  $a, b, c$  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

- Trong **C#** một hàm có thể cho ra giá trị là một mảng như hàm **Find** trong chương trình trên.
- 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  $10*10*10 = 1000$  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 cặp số nhân với công bội là một số tự nhiên khác 0.

## Đặc tả

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

1.  $x$  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  $d \leq \text{trunc}(\sqrt{9 \text{ div } a})$ , trong đó  $\sqrt{}$  là hàm tính căn bậc hai,  $\text{trunc}$  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  $\text{trunc(sqrt(9 div a))}$ . 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 nhầm tính trước cận trên của  $d$  thì sẽ đỡ phải gọi các hàm `trunc` và `sqrт` 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 1
(* Pascal *)
```

---

(\*-----  
Cac so tu nhien 3 chu so  
lap thanh cap han

```

-----*)
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 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

- Trước khi gọi hàm **random** ta cần gọi thủ tục **randomize** để máy tính khởi động cơ chế phát sinh số ngẫu nhiên.
- 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.  
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  $d+c-d = c$ .

```
(* 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 luỹ 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 đê 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 mang nguyen a thanh 2 doan
    co tong bang nhau
----- *)
program ChiaTiep1;
{$B-}
uses crt;
const MN = 500; Esc = #27;
var a: array [1..MN] of integer;
    n: integer;
(*-----
    Sinh ngau nhien n so nguyen khong am
    cho mang a
----- *)
procedure Gen(m: integer);
    var i,d,t: integer;
begin
    randomize; n := m;
    if random(2)=0 then
        begin {khoi tri tuy y}
            for i := 1 to n do a[i]:=random(m);
            exit;
        end;
    { Khoi tri mang co tong d phan tu dau
      bang tong cac phan tu con lai }
    d := random(n div 2)+ 1; { diem 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; { vo nghiem }
    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

- Muốn dừng chương trình hãy nhấn phím Esc có mã ASCII là #27.
- 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 SangTaol
{
    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 giao 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
} // SangTaol

```

### 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  $\text{sum}(a[i+1..n]) = tk$ .

Để 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 luỹ 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 luỹ **tr** để hi vọng đạt được  $tr = tk$ . Nếu sau khi tích luỹ **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; { vo nghiem }
    { Xu li truong hop co nghiem }
    t1 := t div (k+1); { doan tong nho }
    tk := t - t1; { tk = k * t1 }
    tr := 0; { tong rieng 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 nghiem i }
                    Chia:= i; exit;
                end;
        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 mang nguyen a thanh 2 doan
  co tong ti le 1:k
----- *)
{$B-}
uses Crt;
const MN = 500;
      Esc = #27; { dau thoat }
      bl = #32; { dau cach }
      nl = #13#10; { xuong dong }
var   a: array [0..MN] of integer;
      n: integer;
(*-----
  Sinh ngau nhien n so nguyen khong am cho mang a
----- *)
Procedure Gen(m,k: integer);
  var i,d: integer; t: longint;
begin
  n := m; t := 0;
  if random(2) = 0 then { vo nghiem }
    begin
      for i := 1 to n do a[i]:= random(n);
      exit;
    end;
  { co nghiem }
  d := random(n div 2)+1; { diem chia }
  for i := 1 to d do
    begin
      a[i] := random(n); t := t+a[i];
    end;
  if (random(2) = 0) then
  { doan a[1..d] gap k lan doan cuoi }
    a[d] := a[d]+(k-1)*t
  else { doan cuoi gap k lan doan 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 SangTaol
{
/*
 *      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 tuc, ");
            Console.Write("\n Bam T de thoat: ");
        } 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 1 phan 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;
}

```

## 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 phác 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 ngau nhien n so nguyen trong khoang  
    d den c va ghi vao mang 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 ngau nhien cho mang nguyen a
   n phan tu trong khoang -M..M; M > 0.
-----*)

program RGen;
uses crt;
const MN = 1000;
var a: array[1..MN] of integer;
(*-----
   sinh ngau nhien n so nguyen trong khoang
   d den c va ghi vao mang 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 ngau nhien 20 so trong khoang -8..8 }
  Gen(n,-8,8);
  Xem(n);
  readln;
end;
BEGIN
  Test;
END.

// C#
using System;
using System.Collections.Generic;
using System.Text;
namespace SangTaol
{
  /*-----
   *      Sinh ngau nhien n so
   *      trong khoang 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] := 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:  
 $(i = 2..n): a[i] := a[i - 1] + random(n)$ , 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 ngau nhien n so
     *      tao thanh day khong giam
     * -----
    */
    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 ngau nhien cho mang nguyen a
     mot hoan vi cua 1..n
-----*)
program GenPer;
const MN = 1000; { so luong toi da }
               Esc = #27; { dau thoat }
               BL = #32; { dau cach }
var a: array[1..MN] of integer;
(*-----
     Sinh du lieu
-----*)
procedure Gen(n: integer);
var i,j,x: integer;
begin
{ Khoi tao hoan vi don vi }

```

```

        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 SangTaol
{
    /*
     * 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

- Chọn số lượng các phần tử trong mỗi đoạn là `random(n 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.

- 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:

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

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

$$(i = d..c) : a[i] := random(tr)$$

- Đồng thời chỉnh giá trị còn lại của tr:

$$tr := tr - a[i]$$

Ta có:

$$\begin{aligned} a[d] &< t \\ a[d+1] &< t - a[d] \\ a[d+2] &< t - a[d+1] - a[d] \\ &\dots \\ a[c-1] &< t - a[d] - a[d+1] - \dots - a[c-2] \end{aligned}$$

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

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

- 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 - so toi da phan tu trong moi doan}
  i := 0; {chi dan lien tuc cho cac phan tu moi sinh}
  for j := 1 to k do {sinh doan thu j}
  begin
    tr := t;
    for p := 1 to random(s) do
      {random(s)+1 = so phan tu trong 1 doan }
    begin
      inc(i);
      a[i] := random(tr);
      tr := tr - a[i]; {gia tri con lai cua tong}
    end;
    inc(i); {i phan tu cuoi cung cua doan j}
    a[i] := tr;
  end;
  {bu 0 cho cac phan tu con lai}
  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 SangTao1
{
  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 để tiếp 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;
    }
    // diện 0 cho số 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 tách lẻ

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 truoc 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 SangTaol
{
    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() == "");
        }
        // Kiểm tra kết quả
        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 Đoán thứ nhất: " +
                "sum(a[0.." + (n1 - 1) + "]) = " + t1);
            int t2 = 0;
            for (int i = n1; i < a.Length; ++i)
                t2 += a[i];
            Console.WriteLine("\n Đoán thứ hai: " +
                "sum(a[" + n1 + ".." + (a.Length - 1) + "]) = " + t2);
            if ((t1 == k * t2) || (t2 == k * t1))
                Console.WriteLine("\n ĐÚNG");
            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 ngau nhien n so tu nhien sap tang
  va ghi vao tep van ban co ten cho truoc
-----*)
program FincGen;
uses crt;
const BL = #32; { dau cach }
(*-----
  Sinh du lieu
-----*)
procedure Gen(fn: string; n: integer);
var f: text; i: integer; x: longint;
begin
  assign(f,fn); {fn: file name (ten tep)}
  rewrite(f); randomize;
  x := 0;
  for i:= 1 to n do
    begin
      x := x+random(10); write(f,x,BL);
      { moi dong trong file chua 20 so }
      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
} // SangTao1

```

### 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  $a[i]$ ,  $i = 2..n$   
 $\text{for } i:=2 \text{ to } n \text{ do } a[i]:= a[i-1] + d;$
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; {công 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 SangTaoI
{
    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
} // SangTaoI

```

### 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

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

#### 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;
  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 SangTaol
{
  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 đồ 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 sotoi 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.WriteLine(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  $10*10*10 = 1000$  lần duyệt trong khi với  $h = 10$  chỉ có 63 số thoả mãn điều kiện của bài toán. Dùng phương pháp sinh ta nhận được đúng 63 số cần tìm, không phải duyệt thừa số nào.

### Bài 2.10. Tệp 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 thoả 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 thoả 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>	5	<u>6</u>	8

*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 SangTao1
{
    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; // đếm số hoán vị 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
} // SangTao1

```

### 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  $n, m$  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 chữ 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 ≠ 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 IORESULT <> 0 then exit; {không mở được file}
  read(f,n,m);{doc kich thuoc n va m cua mang }
  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('Khong mo duoc tep ');
  readln;

```

```

END.

// C#
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
namespace sabgTaol
{
    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, toc, dong tep
            string[] ss = (File.ReadAllText(fn)).
                Split(cc,
                    StringSplitOptions.RemoveEmptyEntries);
            // Chuyen sang mang 1 chieu 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 đổi 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  $3 \times 3$  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; { kich thuoc mang }
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),
                new 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.

Ví dụ, hình bên có 5 tàu.

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

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 lênh 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	B	
0	1	1	1	1	1		
0	1	1	1	1	1		
D	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 SangTaol
{
    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
} // SangTao1

```

### Bài 2.15. Sắp đoạn

Trong một tệp văn bản chứa những đoạn cắt ra từ một trục số. Mỗi đoạn có dạng  $<d, c>$  trong đó " $<$ " có thể là một trong hai kí tự ( hoặc  $[$ ,  $>$  có thể là một trong hai kí tự ) hoặc  $c$ ,  $d$  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  $<d, c>$  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  $toan[i] = BL$  (dấu cách).

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

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 toán hạng}
  MangToan = array[1..2] of char; {2 toán tử +,*}
  KieuDoan = record
    mo: char;
    dong: char;
    so: MangSo;
    Toan: MangToan;
    len: integer;
  end;
```

Các đoạn đọc được sẽ được ghi dàn 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 ký tự trong tệp văn bản.

Mỗi lần bạn sờ một kí tự c nào đó rồi dưa 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 lý tệp input ta có thể gặp năm trạng thái như sau:

**$q = 0$ : Trạng thái mở 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:

**$so[1] Toan[1] so[2]$**

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 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 phep toan}
    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;
    end;
  end;
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
    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;
    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])
  end;
end;

```

```

        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 SangTaol
{
    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; // trang thai
int i = 0; // bien tro trong s
int dau, cuoi;
for (; s[i] != Ket; ++i)
{
    switch(q)
    {
        case 0: // Tim dau doan
            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].So1 = 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 doan
            d[n].Dong = s[--i];
            dau = d[n].So1;
            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,i);
    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;** **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 phần mở rộng phụ thuộc vào từng phiên bản cài đặt của hệ điều hành.

```

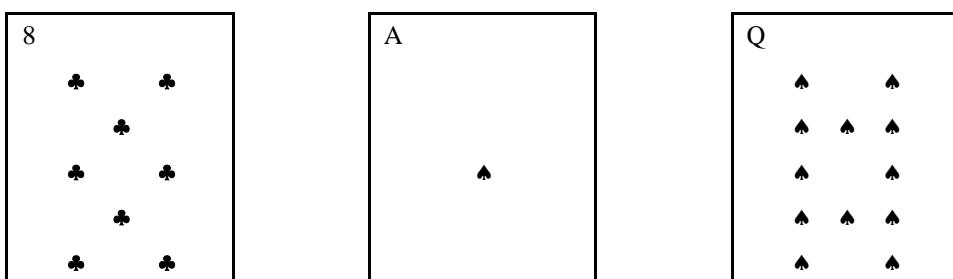
using System;
using System.IO;
namespace SangTao1
{
    class ASCII
    {
        static void Main()
        {
            string fn = "ASCII.TXT";
            StreamWriter g = File.CreateText(fn);
            for (int i = 0; i < 128; ++i)
                g.WriteLine("{0}: {1}", i, (char)i);
       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





*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 toạ 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 gắp 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 gắp 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

<b>DX</b>	<b>9</b>	<b>9</b>
<b>DY</b>	<b>12</b>	<b>6</b>
<b>TX</b>	<b>2</b>	<b>1</b>

*Các tham số kích thước quân bài DX × 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 thuoc quan bai}
    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, nhep);
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 SangTaoI
{
    /*
     *          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; // kich thuoc quan bai
        // 1 khoang cach giua 2 dong
        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"};
        // Dat mau nen va text cho man hinh
private void SetColors(ConsoleColor bg, ConsoleColor fg)
{
    Console.BackgroundColor = bg;
    Console.ForegroundColor = fg;
}
        // Viet s tai cot x, dong y
private void WriteAt(string s, int x, int y)
{
    Console.SetCursorPosition(x, y);
    Console.Write(s);
} // WriteAt
// Ve bo bai tai vi tri 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(); // tra lai nen cu
} // Draw
/*
----- Ve 5 dong trong quan bai -----
----- */
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, nhep);
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-trai
}
```

```

    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 thầm 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:

<b>LEN:</b>	<b>200</b>	Mũi tên trỏ lên ↑
<b>XUONG:</b>	<b>208</b>	Mũi tên trỏ xuống ↓
<b>PHAI:</b>	<b>205</b>	Mũi tên trỏ qua phải →
<b>TRAI:</b>	<b>203</b>	Mũi tên trỏ qua trái ←
<b>ESC (27)</b>	và <b>ENTER/RETURN (13)</b>	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(' Nhấn Phím (Bấm ESC để thoát): ');
    k:= GetKey;
    if k > 128 then
      writeln(' Phím mo rong (0, ',k-128,',') ==> ',k)
    else
      writeln(' Phím thường ',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  $4 \times 4$  ô theo hình trạng ban đầu như rong hinh. 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:

- a) Đảo ngẫu nhiên các quân cờ để chuyển từ hình trạng ban đầu về một hình trạng H nào đó.
- b) 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 lôgic `Dung`).

Hai thủ tục `DaoNgauNhien` 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 xoá 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; {xoá cho đến hết dòng}
  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 tấm 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 tấm 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  $a[i \ mod \ b]$  thêm  $k$  đơn vị và tìm đến cột  $x[i \ mod \ b]$ , dòng  $y[i \ mod \ b]$  để cập nhật lại giá trị này. Cuối cùng ta chuyển giá trị  $z$  vào  $a[i \ mod \ b]$ . 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 chuyên 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;
end;
gotoxy(xx,yy); write(' KET ');
readln;
end;
BEGIN
    BangNhay(3,6,10);
    (* Loi giao: -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 họa ham Console.ReadKey()
using System;
using System.Text;
class Sample
{
    public static void Main()
    {
        Test();
    }
    public static void Test()
    {
        ConsoleKeyInfo k;
        do {
            Console.Write("\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;
                }
            }
        } while (c != ' ');
    }
}

```

```
        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.WriteLine(" Mui ten xuong");
        break;
    case ConsoleKey.UpArrow:
        Console.WriteLine(" Mui ten len");
        break;
    case ConsoleKey.End: Console.WriteLine(" End");
        break;
    case ConsoleKey.Escape: Console.WriteLine(" 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 dd1 = 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 = dd1; // toa do o trong
static public int dong = dd1;
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 cotty = Console.CursorTop;
    int dongx = Console.CursorLeft;
    Console.SetCursorPosition(dongx, cotty);
    Console.Write(d);
    do
    {
        VeO(dong, cot);
        if (Sanhab())
        {
            Console.SetCursorPosition(dongx+2,
                                      cotty);
            Console.Write(" Chuc mung " +
                          "Thanh Cong !!!");
            Console.ReadLine();
            return;
        }
        k = GetKey(); ++d;
        Console.SetCursorPosition(dongx, cotty);
        Console.Write("           ");
        Console.SetCursorPosition(dongx, cotty);
        Console.Write(d);
        switch (k)
        {
            case LEN: // Day quan duoi o trong LEN
                if (dong < dd1)
                {
                    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 < dd1)
    {
        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 =

```



```

        }
        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[dd1,dd1] = a[dd1, dd1] = 0;
    dong = dd1; cot = dd1;
    DaoNgauNhien(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: Tệp văn bản **CUM.INP** chứa một dòng kiểu xâu kí tự (string) là biểu thức cần xử lí.

Dữ liệu ra: Tệp 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 gấp lõi 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:

- 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**).
- 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  
    KhoiTrieSt; {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;  
end;
```

Sau khi duyệt xong xâu **s** ta ghi kết quả vào tệp **output.g**. Hàm **copy(s,i,d)** cho xâu gồm **d** kí tự được cắt từ xâu **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 mbl = array[1..255] of byte;
```

```

var s: string; {chua bieu thuc can xu li}
    st: mb1; {stack}
    {stack luu vi tri xuat hien dau ( trong xau s)
     p: integer; {con tro stack}
     dau,cuoi: mb1; {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 gõp

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: Tệp 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: <b>BAIGOP.INP</b>	Kết quả hiển thị trên màn hình
10 6	
2 5	
3 3	
4 7	
1 5	
2 8	
9 3	

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 kỹ và cố gắng cài đặt cho nhuần nhuyễn. Như sau này sẽ thấy, nhiều thuật toán xử lý 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 quân 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: 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 nhất trong các phần tử có giá trị lớn hơn nó theo phuong thuc: mỗi phần tử trong một tập đều trả trực tiếp đến một phần tử nhỏ hơn nó và có trong tập đó. Phần tử nhỏ nhất trong tập trả lời chính nó.*

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

$A = \{\underline{3} \rightarrow \underline{3}, 4 \rightarrow \underline{3}, 7 \rightarrow \underline{3}\}$ . 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:

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

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

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

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

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

tức là ta thực hiện đúng một thao tác sửa  $3 \rightarrow 3$  thành  $3 \rightarrow \underline{2}$ : để hợp hai tập ta chỉ việc đổi 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 trả 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 trả đến 2}

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

Sau khi hợp nhất A với B ta được:

$$a[3] = 2; \{chỗ sửa duy nhất\}$$

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

(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 **Find(x)=Find(y)** thì không xảy ra việc gộp bài vì **x** và **y** cùng nằm trong một tập. Ngược lại, nếu **Find(x)>>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 **Find(x) >> Find(y)** ta giảm **c** 1 đơn vị: **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 SangTaol
/*
 *             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
} // SangTaol

```

### 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 cảng 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.

<b>CHUOI.DAT</b>	Trong thí dụ này, các thông báo trên màn hình sẽ là:
------------------	--

4	Số màu trong chuỗi: 5
4	Cắt giữa hạt thứ 7 và thứ 8, tổng số lớn nhất là 7.
1	
5	
5	
5	
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 họa 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  $2*(n - 1)$ .

CHUOI.DAT	
4	4 a[1]
4	b[2] 4 7 a[2]
1	b[3] 1 4 a[3]
5	b[4] 5 8 a[4]
5	b[5] 5 8 a[5]
5	b[6] 5 8 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;
        {nói 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;
end;

```

Theo thí dụ trên, sau khi nối **b[2..n]** vào sau **a[1..(n - 1)]** ta thu được  
 $a[1..12] = (4, 7, 4, 8, 8, 8, 5, 5, 5, 1, 4)$

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 chuoi
-----*)
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 lí một trong hai điểm: điểm đầu hoặc điểm cuối của đoạn đó. Ta chọn điểm đầu. Kỹ thuật này được gọi là quản lí theo đoạn.

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

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

mời xem tưởng 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 phân 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;
```

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 cảng ra. Từ đó ta phác thảo được sơ đồ cho thủ tục `xuly` để tìm điểm cắt **ĐiemCat** 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 > Lmax thì
    Đặt lại Lmax := 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 lí riêng hai đoạn này ở bước khởi trị như sau:

```

{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;

```

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

```

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;

(* 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],bl);
end;
(*-----
      Dem so mau trong chuoi
-----*)
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;
procedure run;
var i: integer;
begin
  Doc; Xem; writeln;
  writeln('So mau trong chuoi: ',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(' Chiieu 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 SangTaoI
{
  class ChuoiHat
  {

```

```

const string fn = "chuoi.dat";
static int[] a; // chuoi 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 chuoi
 * -----
 */
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 giua 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 giua hat thu " +
                (DiemCat + 1));
            Console.WriteLine(" va hat thu " +
                ((DiemCat + 1) % n + 1));
            Console.WriteLine(" Chieu dai max = " + Lmax);
        }
    } // class
} // SangTao1

```

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

abc.inp	abc.out
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  $s = 'dabdec'$  thì xâu thứ cấp thứ 2 sẽ là  $[2] = 'abdecd'$ . 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]	abdecd	[3]	3
③	[3]	bdecdab	[6]	6
④	[4]	decdab	[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  $N \log_2 N$  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
    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 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 *Find* ( $k$ ) cho ra vị trí gốc của xâu thứ cấp sẽ đúng thứ  $k$  trong dãy đã sắp. Trong thí dụ trên *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;
(*  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 SangTaol
{
    /*
     *   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 PhanTuGuia() // 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);
}
/*-----
 * Kiem tra lai bang cach dung
 * thuat toan QSort theo chi dan
 * -----*/
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
} // SangTaol

```

### 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
cabxabcdab	cabxabcdab
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* = 'cabxabcdab' thì *w[5..8]* = 'abcd'. Gọi

$s$  là xâu mẫu,  $x$  là xâu 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 kĩ thuật tổ chức hậu tố như sau.

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

```

s[1..10] = 'cabxabcdab'
s[2..10] = 'abxabcdab'
s[3..10] = 'bxabcdab'
s[4..10] = 'xabcdab'
s[5..10] = 'abcdab'
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 kí tự đầu tiên.

Trước hết ta sắp tăng các hậu tố của xâu mẫu  $s$  theo trật tự từ điển. Sử dụng một mảng chỉ dẫn  $id$ , trong đó  $id[i]$  trả đến vị trí đầu tiên của hậu tố trong xâu 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 xâu mẫu  $s[1..10] = 'cabxabcdab'$  ta thu được:

i	id[i]	Hậu tố	Xâu
1	9	s[9..10]	ab
2	5	s[5..10]	abcdab
3	2	s[2..10]	abxabcdab
4	10	s[10..10]	b
5	6	s[6..10]	bcdab
6	3	s[3..10]	bxabcdab
7	1	s[1..10]	cabxabcdab
8	7	s[7..10]	cdab
9	8	s[8..10]	dab
10	4	s[4..10]	xabcdab

**Sắp tăng theo chỉ dẫn các hậu tố của xâu**  
 $s[1..10] = 'cabxabcdab'$

Việc còn lại là so sánh xâu  $x$  với các hậu tố  $s[i..j]$  để tìm khúc đầu chung dài nhất giữa chúng. Thí dụ, với  $x[1..4] = 'abcd'$  thì khúc đầu chung dài nhất tìm được với hậu tố  $s[5..10]$  do  $id[2]$  trả 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 va d}
writeln(g,v,BL,d);
end;
end;

```

Để 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 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 := -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.

```

(*-----
      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;

(*  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
mb1 = array[0..MN] of integer;
var
f,g: text;
s,x: string; {s: xau mau}
id: mb1; {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 SangTaol
{
    /*
     *          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; // vi tri xuat hien khuc dau x trg s
        static int d = 0; // chieu dai x trong s
        static int n = 0; // chieu dai xau mau
        static void Main()
        {
            Run();
            Test();
            Console.ReadLine();
        } // Main

        // Doc lai file gn de kiem tra ket qua
        static void Test()
        {
            Console.WriteLine("\n Kiem tra lai ket qua \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);
            // Bo qua cac dong trong dau tien
            while ((s = (f.ReadLine()).Trim()) == "") ;
            n = s.Length; // Len xau mau
            id = new int[n];
            // Khoi tri 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 gọi ý chúng ta tìm một trật tự hợp lí 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 cài biến 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:

	<b>BANGNHAC.INP</b>	<b>BANGNHAC.OUT</b>
- Bài 1 phát trong thời gian 7 phút.	3	2 2
- Bài 2 phát trong thời gian 2 phút.	7 2 3	3 5
- Bài 3 phát trong thời gian 3 phút.		1 12

Dữ liệu ra được ghi trong tệp văn bản

19

**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. Để dàng sửa chương trình để vận dụng cho trường hợp dung lượng tăng 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 giao}
    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 SangTaoI
{
    /*
     *          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 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 **viec.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:**

**viec.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 **viec.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 **viec.out** sẽ như sau:

**viec.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

- thưởng 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 trực thời gian  $b$ . Nếu giờ  $h$  trên trực 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 trực thời gian  $b$ ,  $b[m] := k$ . Sau khi xếp việc xong, có thể trên trực thời gian còn những thời điểm rỗng, 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: trực thời gian với 5 thời điểm ứng với  $T_{\max} = 5$  là thời điểm muôn 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 trực 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 trực 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 mang 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 dây 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 = 'vietc.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 viet}
  f,g: text;
  M: integer; {so viet 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 gieu}
  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;
end;

```

```

(*-----
Xep viec theo giao 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]; {vieu 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;
(*-----
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;
(*-----
          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 SangTaol
{
    /*-----
     *      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. Giải thích là có thể lấy một tỉ lệ tùy ý ở mỗi vật.

Dữ liệu vào: Tệp 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.

<b>BALO.INP</b>		<b>BALO.OUT</b>
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 tr/kg,	3
5 4	- Vật thứ hai có trọng lượng 5, đơn giá 4,	0
4 2		3

<b>3 8</b> <b>16 6</b>	<ul style="list-style-type: none"> <li>- Vật thứ ba có trọng lượng 4, đơn giá 2,</li> <li>- Vật thứ tư có trọng lượng 3, đơn giá 8,</li> <li>- Vật thứ năm có trọng lượng 16, đơn giá 6.</li> </ul> <p>(tr. - triệu đồng)</p>	<b>16</b> <b>172</b>
---------------------------	---	-------------------------

Dữ liệu ra: Tập văn bản tên **balo.out**:

- $N$  dòng, dòng thứ  $i$  cho biết trọng lượng cần lấy ở vật thứ  $i$ .
- Dòng cuối cùng ghi tổng giá trị thu được.

## Hướng dẫn

Có nhiều bài toán thuộc họ xếp ba lô, thuật toán cho bài này thuộc lớp tham lam.

Dễ thấy tiêu chuẩn chọn là giá đơn vị cao. Ta duyệt các vật theo giá đơn vị từ cao trở xuống. Vật được chọn sẽ được lấy tối đa. Như vậy, nếu tổng trọng lượng các vật bằng hoặc lớn hơn sức mang của ba lô thì bao giờ ba lô cũng được xếp đủ.

(\* Pascal \*)

```
(*-----*
     BALO.PAS
-----*)
program balo;
uses crt;
const
  MN = 200;
  fn = 'BaLo.inp';      gn = 'BaLo.out';
var
  a,id: array[1..MN] of integer; {a[i] tr lg vat i}
  gdv: array[1..MN] of integer; {gdv[i] don gia vat i}
  f, g: text;
  n,m: integer; {n: so vat; m: trong luong balo}
  t,tt: integer;
  {t: tong trong luong con duoc xep vao balo}
  {tt: tong gia tri da lay}
(*-----
     Doc du lieu
-----*)
procedure Doc;
var i,k: integer;
begin
  assign(f,fn); reset(f); readln(f,n,m);
  for i := 1 to n do read(f,a[i],gdv[i]);
  close(f);
end;
(*-----
     Khoi tri cho chi dan
-----*)
procedure InitID;
var i: integer;
begin
  for i := 1 to n do id[i] := i;
end;
(*-----
     Sap giam theo gia don vi
-----*)
```

```

-----*)
procedure IDQuickSort(d,c: integer);
var i, j, k, x: integer;
begin
  i := d; j := c;
  x := gdv[id[(i+j) div 2]]; {phantu gieu}
  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;
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. Tính liên thông của đồ thị cho biết với hai đỉnh cho trước tuy ý ta luôn tìm được các cạnh gối đầu nhau để đi từ đỉnh này đến đỉnh kia. Hãy chỉ ra một phần  $P$  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 đất liền. 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.

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

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 rùng. 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 (rùng) 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 nên gốc đó.

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

```
(* Pascal *)
(*-----
  DOTHI.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 gieu}
    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 nhat 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 nhat *)
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;
        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 nhat
     * -----
    */
    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 nhât 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 kĩ thuật săn đuôi như sau: dùng biến lôgic **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 lí (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; {so phan 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 SangTaol
{
  /*
   *       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 trả về giá trị true và số  $i$ , ngược lại, nếu không gặp số nguyên, hàm trả về 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.

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

### Thí dụ:

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

## Thuật toán

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

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

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

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

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

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

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

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

Để ý rằng trộn tệp  $f$  với tệp  $g$  hay trộn tệp  $g$  với tệp  $f$  thì số thao tác ghi tệp như nhau và cùng bằng  $|f| + |g|$ . Giả sử ta có ba tệp 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 tệp  $\textcircled{1}$  với tệp  $\textcircled{2}$  ghi tạm vào tệp  $\textcircled{4}$ . Số thao tác ghi sẽ là  $(5 + 1) = 6$  và tệp  $\textcircled{4}$  có 6 phần tử.

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

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

$$6 + 8 = 14.$$

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

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

ta dễ dàng tính được tổng số thao tác ghi tệp 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	$(\textcircled{1} \oplus \textcircled{2}) \oplus \textcircled{3}$	$2(5 + 1) + 2 = 2.6 + 2 = 14$
2	$(\textcircled{1} \oplus \textcircled{3}) \oplus \textcircled{2}$	$2(5 + 2) + 1 = 2.7 + 1 = 15$
3	$(\textcircled{2} \oplus \textcircled{3}) \oplus \textcircled{1}$	$2(1 + 2) + 5 = 2.3 + 5 = 11$ (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 lý.
- Kết nạp  $h$  vào danh sách các tệp cần xử lý

**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 Mihailo (Mike) Hoffman là người đầu tiên đề xuất.

Ta minh họa 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 lý	Hai tệp có số phần tử min	Trộn	Số thao tác ghi tệp
1	$(\textcircled{1}:10, \textcircled{2}:5, \textcircled{3}:4, \textcircled{4}:4, \textcircled{5}:3)$	$\textcircled{5}:3, \textcircled{3}:4$	$\textcircled{5} \oplus \textcircled{3} \rightarrow \textcircled{6}$	7
2	$(\textcircled{1}:10, \textcircled{2}:5, \textcircled{4}:4, \textcircled{6}:7)$	$\textcircled{2}:5, \textcircled{4}:4$	$\textcircled{2} \oplus \textcircled{4} \rightarrow \textcircled{7}$	9
3	$(\textcircled{1}:10, \textcircled{6}:7, \textcircled{7}:9)$	$\textcircled{6}:7, \textcircled{7}:9$	$\textcircled{6} \oplus \textcircled{7} \rightarrow \textcircled{8}$	16
4	$(\textcircled{1}:10, \textcircled{8}:16)$	$\textcircled{1}:10, \textcircled{8}:16$	$\textcircled{1} \oplus \textcircled{8} \rightarrow \textcircled{9}$	26
Kết quả	$(\textcircled{9}:26)$			58

### Minh họa thuật toán Huffman với dữ liệu vào

$$(\textcircled{1}:10, \textcircled{2}:5, \textcircled{3}:4, \textcircled{4}:4, \textcircled{5}:3)$$

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 (tệp) 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 trên 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  $s(1), \dots, s(n)$ .

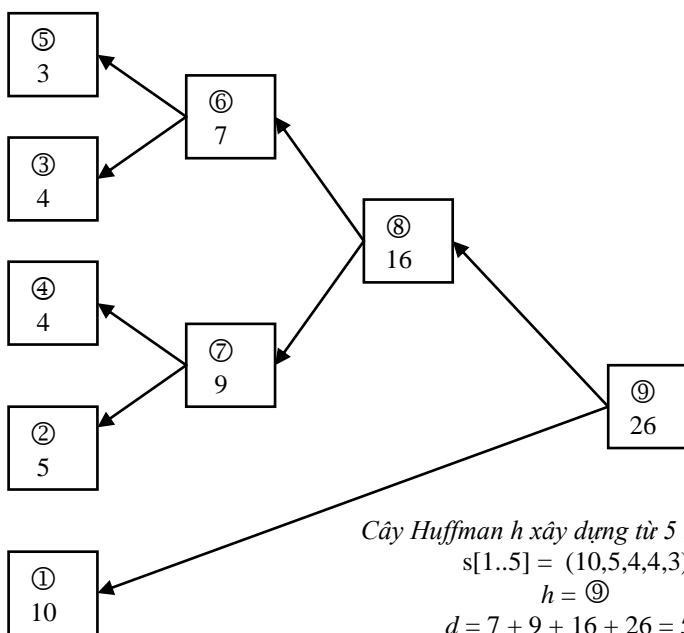
$h := n$ ;

#### 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  $h$  trả đến  $u$  và  $v$  và  $s(h) = s(u) + s(v)$ .

xong 1 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  $h$ ,  $t[i]$  trả đến con trái của đỉnh  $h$ ,  $p[i]$  trả đến con phải của đỉnh  $h$ . Hai con trái  $t[h]$  và phải  $p[h]$  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 lí 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 lí và đánh dấu luôn đỉnh tìm được (là đã xử lí).

```

{-----
  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 SangTaol
{
  /*
   *      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 lí 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

$$v = (v[1], \dots, v[i], v[i + 1])$$

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: lui 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 đẽ 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 nghiêm;
  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 nghiêm;
    else
      Ghi nhận: d nghiêm;
    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 nghiêm;
    else
      Ghi nhận: d nghiêm;
    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

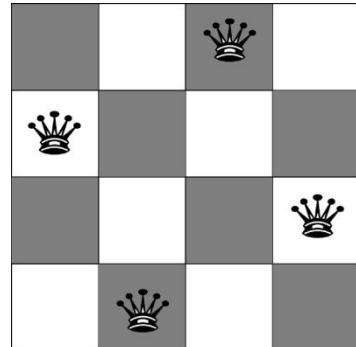
Trước hết ta đặt các quân Hậu ở mép ngoài bàn cờ. Hậu thứ  $i$  sẽ đứng ở đầu cột thứ  $i$ . Sau đó ta dịch dần các Hậu vào trong các dòng của bàn cờ và ghi nhận vị trí của chúng vào một mảng  $v$ . Phần tử  $v[i]$  của mảng  $v$  cho biết phải đặt Hậu thứ  $i$ , tức là Hậu chiếm cột  $i$  tại dòng  $v[i]$ .

Thí dụ, với bàn cờ  $4 \times 4$  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  $i - j = \text{abs}(v[i] - v[j])$  (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 nghiem: 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 kich thuoc ban co}
  n := M;
  {Khởi trị: Đặt cỗc 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 {co nghiem v[1..n]}
      begin
        KetQual(n);
        exit;
      end;
    if i < 1 then {vo nghiem}
      begin
        KetQual(0);
        exit;
      end;
    i := i + 1;
  until KetQual(n) = 0;
```



```

        end;
if Tim(i) {co cach di }
    then inc(i) {Tien}
else
begin {Lui}
    v[i] := 0;
    dec(i);
end;
until false;
end;

```

Thủ tục có hai tính huống, KetQual1(n): hiển thị mảng  $v[1..n]$ , trong đó  $v[i]$  là dòng đặt Hậu  $i$ , KetQual1(0) : thông báo vô nghiệm.

Hàm Tim(i) thực hiện chức năng sau đây: xuất phát từ dòng Hậu  $i$  đang đứng là  $v[i]$  đầy 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.

```

(*-----
    Xuat phat tu dong v[i]+1, tim dong moi
    co the dat duoc Hau 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.

```

(*-----
    Kiem tra xem co dat duoc Hau i
    tai o (v[i],i) cua ban co khong ?
-----*)
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]))
            {Hau j an duoc Hau i}
            then exit;
    DatDuoc := true;
end;

```

Thao tác Tiên đơn giản là chuyển qua xét Hậu kế tiếp, Hậu  $i+1$ .

Tien: Chuyển qua Hậu tiếp theo  
 $\text{inc}(i);$

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  $8 \times 8$  ta thu được 92 nghiệm. Với bàn cờ  $10 \times 10$  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 ');\br/>            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, kich thuoc ban co}
  g: text; {tep ket qua}
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 XepHau1(M: byte); tự viết
(*-----
   Ghi nghiem thu d vao tep 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
  XepHau1(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 đúng độ với tất cả  $(i-1)$  hậu đặt trước đó không. Thời gian chi phí tập trung ở chính điểm này.

Để cài 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  $c2[i+j] = 1$  có nghĩa là đã có hậu kiểm soát đường chéo này. Ngược lại, khi  $c2[i+j] = 0$  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:

$(dd[j] = 0) \text{ and } (c1[i-j] = 0) \text{ and } (c2[i+j] = 0)$ , hay  
 $(dd[j] + c1[i-j] + c2[i+j] = 0)$

(\* 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;
(*-----
    Xuat phat tu dong v[i]+1,
    tim dong j co the dat duoc Hau 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;
(*-----
    Hien thi nghiem tren man hinh
    Cho bai toan tim 1 nghiem
    k = 0: vo nghiem
    k = n: co nghiem v[1..n]
-----*)
procedure Ket1(k: integer);
var i: integer;
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: 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; { Dang xet Hau 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 tim kiem, v[i] - dong dat Hau i
    static int[] dd = new int[mn + 1];
    // dd[i] = 1: dong i bi cam
    static int[] c1 = new int[mn2 + 1];
    // c1[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; // kich thuoc ban co
static void Main()
{
    Console.WriteLine("\n Test 1: Tim 1 nghiem " +
                      " voi n = 1..10 ");
    Test1();
    Console.ReadLine();
    Console.WriteLine("\n Test 2: Tim moi nghiem " +
                      " voi n = 8");
    Test2();
    Console.WriteLine("\n Fini ");
    Console.ReadLine();
} // Main
// Test 1: tim 1 nghiem voi n = 1..10
static void Test1()
{
    for (int i = 1; i <= 10; ++i)
    {
        Console.Write(" \n n = " + i + ": ");
        if (XepHau(i)) Print(v, n);
        else Console.WriteLine(" Vo Nghiem");
    }
}
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; // Hau dang xet
    int dong = 0;// dong dat Hau k
    do
    {
        if (k > n) return true;
        if (k < 1) return false;
        NhacHau(k);
        if ((dong = TimNuocDi(k)) > 0)
        {
            DatHau(k, dong);
            v[k++] = dong;// tien
        }
        else v[k--] = 0; // lui
    } while (true);
}
// Nhac Hau k khoi vi tri dang dat
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);
}

```

```

    }
    // Dich hau k tu vi tri hiên 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.WriteLine(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.

- Với giá trị  $N$  cho trước, hiển thị trên màn hình một từ chuẩn loại 3 có chiều dài  $N$ .
- 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 chả liên tiếp hai từ con giống nhau là 213.

Tương tự, 12332 không phải là từ chuẩn vì chả 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: Dừng 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 nghiem v[1..n]}
        begin
            KetQua1(n); {in nghiem v[1..n]}

```

```

    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 {Lại}
      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.

$\text{Tim} = \text{true}$  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  $\text{Tim} = \text{false}$ .

```

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); tu 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);
    end;
  until i > n;
end;

```

```

        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; // kich thuoc ban co
                static int k = 0;
                static void Main()
                {
                    int sl = 10;
                    Console.WriteLine("Test 1: Tim 1 nghiem voi n = "+sl);
                    Test1(sl);
                    Console.WriteLine("Test 2: Tim moi nghiem voi n "+sl);
                    Test2(sl);
                    Console.WriteLine("\n Doc lai Ket qua:\n");
                    Console.WriteLine(File.ReadAllText(fn));
                    Console.WriteLine("\n Fini");
                    Console.ReadLine();
                }
                // Test 2: tim moi nghiem
                static void Test2(int sl)
                {
                    Console.WriteLine(" Tong cong " +
                        TimMoiTu(sl) + " "
                        nghiem");
                }
                // Tim moi nghiem. Phuong phap gia 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.WriteLine(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 gối đầu 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:* Tệp 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 tệp 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.

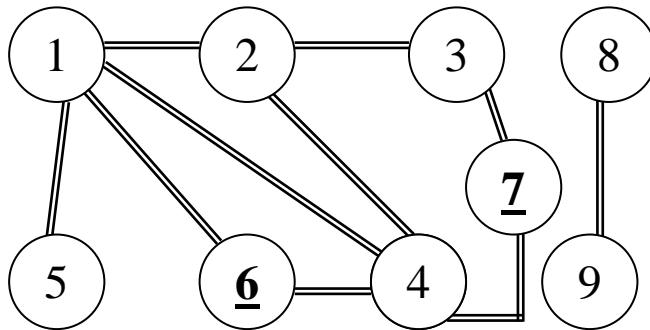
Với thí dụ đã cho kết quả có thể là:

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 → 4 → 2 → 3 → 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 luỹ 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 *sợi chỉ Arian* được phỏng theo một truyền thuyết cổ Hy Lạp sau đây. Anh hùng Te-dây phải tìm diệt con quái vật nhân nguru (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 nó rồi cuốn chỉ quay ra cửa hang, nơi em tróng 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. Đến đâu chàng nhớ nhả chỉ theo 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 - Tim duong trong me cung
(Thuat toan Arian)
s: dinh xuat phat
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 xuat phat }
  d[s] := 1; {da tham dinh s }
  repeat
    if v[k] = t then {den dich }
      begin
        ket(k); {ghi ket qua: giao 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 kè của đồ thị biểu diễn mê cung. Mảng a sẽ đổi xứng vì mê cung là đồ thị vô hướng. Đây cũng chính là lí do giải thích dữ liệu vào chỉ cho dưới dạng nửa trên của ma trận kè.

```
(*-----
  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 hàng kiểu **string bl = #32** là mã ASCII của dấu cách, hàng **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;
end;

```

Thủ tục **Kết(k)** - ghi đường đi  $v[1..k]$  từ  $s$  đến  $t$  tìm được vào tệp **output**.

**Kết(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;
  close(g);
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 kè 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. Dương 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;

```

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 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 Te-dây 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 duyet }
  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(v, 0, v.Length);
    k = 1; // Buoc duyet
    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;
}
} // McCung
} // 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 lý 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

- 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.
- 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.
- 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	$\text{Chia}(i, j)$
$i: \text{số phần thưởng}$	
$j: \text{số học sinh}$	
$j = 0$	$\text{Chia}(i, j) = 0$
$i = 0 \text{ and } j \neq 0$	$\text{Chia}(i, j) = 1$

$i < j$  $Chia(i, j) = Chia(i, i)$  $i \geq j$  $Chia(i, j) = Chia(i, j - 1) + Chia(i - j, j)$ *Các tính chất của hàm Chia(i, j)**Chia i phần thương cho j học sinh*

## 2. Tô chức dữ liệu và chương trình

Ta có phương án đầu tiên của giải thuật Chia như sau:

(\*-----

**PHƯƠNG ÁN 1: đệ quy.**

So cách Chia i phần thương 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	⑤
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(⑦, ①)

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 thương với bảy phần thương ( $m = 7$ ) 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, nhàn tản, 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: dung mang 2 chieu cc
cc[i,j] = Chia(i,j) - so cach chia i
phan thuong 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;

```

**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 kĩ 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 phuong 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 Chia1(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];  
    Chia1 := 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,b1,n); {xem du lieu vao }  
    t := Nhip; {dat nhip cho PA 3 }  
    {Phuong an 3 }  
    writeln('Mang 1 chieu: ',Chia1(m,n));  
    {bao thoi gian }  
    writeln((Nhip-t)/18.2):0:0,' giay');  
    t := Nhip; {dat nhip cho PA 2 }  
    writeln('Mang 2 chieu: ',Chia2(m,n)); {PA 2 }  
    {bao thoi gian }  
    writeln((Nhip-t)/18.2):0:0,' giay');  
    t := Nhip; {dat nhip cho PA 1 }  
    writeln('De quy: ',Chia(m,n));  
    {bao tgian}  
    writeln((Nhip-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 nhịp của máy tính. Mỗi lần đọc giá trị của biến Nhip ta có thể lấy được số nhịp hiện hành của máy. Hiệu số hai lần đọc nhịp liên tiếp sẽ cho ta tổng số nhịp 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
  m11 = array[0..MN] of longint;
  m12 = array[0..mn] of m11;
var
  cc: m12; {cho phuong an 2 - mang 2 chieu }
  m,n: integer;
  c: m11; {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; tuy viet
(*-----
  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; tuy viet
(*-----
  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; tuy viet
procedure test; tuy viet
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:

<b>PALIN.INP</b>	<b>PALIN.OUT</b>
9 <b>baeadbadb</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,  $s = \text{'baeadbadb'}$  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  $\text{baeab}$ :

$$\underline{\text{baeab}}\underline{\text{adb}} \rightarrow \text{baeab}$$

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  $\text{bdadb}$  với cùng chiều dài 5:

$$\underline{\text{bae}}\underline{\text{adb}}\underline{\text{adb}} \rightarrow \text{bdadb}$$

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  $s = baeadbab$  thì dãy con đối xứng dài nhất của  $s$  sẽ là  $baeab$  hoặc  $bdbab$ ,... 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:

$$p(i,j) = \max(p(i,j-1), p(i+1,j))$$

Ván 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;
```

**end;**

	j-1	j	b	a	e	a	d	b	a	d	b
			①	②	③	④	⑤	⑥	⑦	⑧	⑨
			b ①	1 1	1 1	3 3	3 5	5 5	5 5	5 5	
			a ②	0 1	1 1	3 3	3 3	3 3	3 3	3 3	
i	[i, j-1]	[i, j]	e ③	0 0	1 1	1 1	1 1	3 3	3 3	3 3	
i+1	[i+1, j-1]	[i+1, j]	a ④	0 0	0 1	1 1	1 1	3 3	3 3	3 3	
			d ⑤	0 0	0 0	0 1	1 1	1 3	3 3		
			b ⑥	0 0	0 0	0 0	1 1	1 1	1 3		
			a ⑦	0 0	0 0	0 0	0 0	1 1	1 1		
			d ⑧	0 0	0 0	0 0	0 0	0 0	1 1		
			b ⑨	0 0	0 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, toạ độ  $(i, j)$  sẽ thoả 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 dong voi 2 mang  
1 chieu 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(nl,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 dong voi mang 1 chieu 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;

```

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 mil = array[0..mn] of integer;
  mi2 = array[0..mn] of mil;
  mc1 = array[0..mn] of char;
var  n: integer; { Chieu dai xau }
  f,g: text;
  s: mc1; { xau can xu li }
  d,v: mil;
  c: mi2;
procedure Doc; tự viết
function Max(a,b: integer): integer; tự viết
(*-----
----- Phuong an de quy
-----*)
function rec(i,j: integer): integer; tự viết
(*-----
----- Quy hoach dong voi mang 2 chieu c
-----*)
function QHD2C: integer; tự viết
(*-----
----- Quy hoach dong voi 2 mang
----- 1 chieu d, v
-----*)

```

```

-----*)
function QHD2DV: integer; tự viết
(*-----
   Quy hoạch dong voi mang 1 chieu 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
} // SangTaol

```

### 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 mĩ 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 mĩ 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 mĩ 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:**

<b>HOA.INP</b>	<b>HOA.OUT</b>
----------------	----------------

4	6	24
1	1 6 4 3 10 1 3 4 6	
9	1 4 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 mĩ sẽ  
đạt là 24 (điểm) nếu cắm hoa như sau:

- Bó hoa 1 cắm vào lọ 1;

- 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 mĩ khi giải bài toán với  $i$  bó hoa mã số  $1..i$  và  $j$  lọ mã số  $1..j$ , tức là độ thẩm mĩ 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 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 mĩ 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 mĩ 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:

- Đặt lại trị  $T[i, j]:= T[i-1, j-1] + v[i, j]$ .
- 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:

- Đặt lại trị  $T[i]$  tại bước  $j:= T[i-1]$  tại bước  $j - 1 + v[i, j]$ .
- 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 lợ hoa ta dùng mảng  $L[0..MN]$  mỗi phần tử  $L[i]$  lại là một dãy  $s$  byte. Nếu dùng một bit cho mỗi lợ hoa thì số byte cần dùng để đánh dấu tối đa  $MN$  lợ hoa phải là:

**kk = (MN+7) div 8**

Với  $MN = 101$  ta tính được

**kk = (101+7) div 8 = 13**

Khi cần đánh dấu lợ hoa thứ  $j$  trong dãy  $L[i]$  ta bật bit thứ  $j$  trong  $L[i]$ . Khi cần xem lợ 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 div 8;**

- Để xác định vị trí của bit  $j$  trong byte thứ  $b$  ta tính:

**p := j mod 8;**

```
(* -----
    Cho giá trị bit thứ 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;
(* -----
    Gán trị 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 lợ nào và không có bô hoa nào ta khởi trị:

```
fillchar(L[0],16,0);
T[0] := 0;
```

Với mỗi  $j = 1..n$ , ta lưu ý số bô hoa phải không lớn hơn số lợ, tức là  $i \leq j$ . Với  $i = j$  ta sẽ cắm mỗi bô vào một lợ. Để thực hiện điều này ta lưu ý rằng phần tử  $L[j-1]$  tại bước trước đã cho biết  $j-1$  lợ đều có hoa do đó ta cần đánh dấu lợ thứ  $j$  cho bước  $j$ :

```
L[j] := L[j-1];
batbit(j,j);
```

Như vậy ta cần chia quá trình duyệt theo các lợ hoa từ  $1..n$  thành hai giai đoạn.

Giai đoạn 1: Duyệt từ lợ 1 đến lợ  $k$ , trong đó  $k$  chính là số bô hoa và theo đầu bài,  $k \leq n$ .

Giai đoạn 2: Duyệt nốt  $n - k$  lợ hoa còn lại.

Phương án quy hoạch động với mảng một chiều khi đó sẽ như sau:

```
(*-----
      Quy hoach 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
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;
(* Pascal *)
(*=====

      Hoa.pas: Quy hoach 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;
ml1 = 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: ml1;
  {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 SangTaoT1
{
  /*-----*
   *           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));
}
// Gan tri 0 cho bit i trong day lo
public void TatBit(int i)
{
    Data[i>>5] &= ~(((uint)1)<<(i&(bitnum-1)));
}
// Lay tri cua 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.*

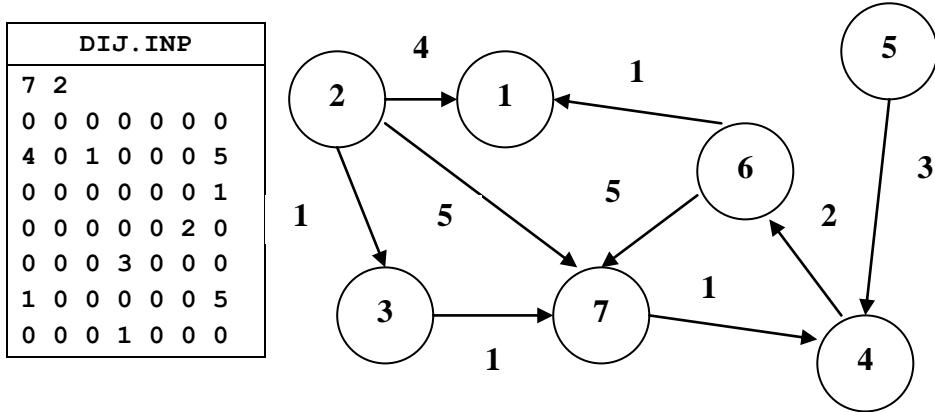
### Bài 7.4. Tìm các đường ngắn nhất

*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,..., 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 kè 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,...



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 | 6 |  |  |
| 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 đỉnh, 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:

a)  $p(s) = 0$ : đường ngắn nhất từ đỉnh xuất phát  $s$  đến chính đỉnh đó có chiều dài 0.

b) 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 từ  $x$  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 ( $j \rightarrow i$ ):

**path(s, i) = path(s, j) + path(j, i)**

trong đó  $\text{path}(j, i)$  chỉ gồm một cung:

**path(j, i) = (j → 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:

$$p(i) = \min \{p(j) + a[j, i] / a[j, i] > 0, j = 1..n\}$$

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 lí 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 kè 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  $\infty$ .

Giá trị  $\infty$  được chọn trong chương trình là:

**MAXWORD = 65535 .**

Tại bước lặp thứ  $k$  ta thực hiện các thao tác sau:

- Trong số các đỉnh chưa xử lí, tìm đỉnh  $i$  có trọng số min.
- Với mỗi đỉnh  $j$  chưa xử lí và kè 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 lí.

Như vậy, tại mỗi bước lặp ta chỉ xử lí 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; { tim dinh i co trong so p[i] ->
min }
  d[i] := 1; {danh dau dinh i la da xu li }
  for j := 1 to n do
    if d[j] = 0 then {dinh chua tham }
    if a[i,j] > 0 then {co duong di i -> j }
      if p[i] + a[i,j] < p[j] then
        begin {sua dinh }
        p[j] := p[i] + a[i,j];
        before[j] := i;
      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 lùi 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 giao 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; tuy viet
(*-----
Hien thi du lieu de kiem tra thu tuc doc
-----*)
procedure Xem; tuy viet
(*-----
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 trinh duong ngan nhat tu s den i.
   Ghi vao tep g
-----*)
procedure path(i: byte); tự viết
(*-----
   Ket thuc thuật toán:
   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;
(*-----
   Thuật toán Dijkstra
-----*)
procedure Dijkstra; tự viết
BEGIN
    Doc; Xem; Dijkstra; ket;
END.

```

Ta minh họa tiền 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:

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

a
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

6	0	65535	0
7	0	65535	0

Bước lắp  $k = 1$

$$i = \min = ② \text{ với } p[②] = 0.$$

Các đỉnh chưa xử lí và kè 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  $\text{before}[1]$  được sửa thành ②.

Vì  $p[②] + a[②, 3] = 0 + 1 = 1 < p[3] = 65535$  nên  $p[3]$  được sửa thành 1 và  $\text{before}[3]$  được sửa thành ②.

Vì  $p[②] + a[②, 7] = 0 + 4 = 5 < p[7] = 65535$  nên  $p[7]$  được sửa thành 5 và  $\text{before}[7]$  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 lắp  $k = 2$

$$i = \min = ③ \text{ với } p[③] = 1.$$

Đỉnh chưa xử lí và kè 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  $\text{before}[7]$  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 lắp  $k = 3$

$$i = \min = 7 \text{ với } p[7] = 1$$

Đỉnh chưa xử lí và kè với đỉnh 7 sẽ được sửa trọng số là đỉnh 4.

Vì  $p[7] + a[7, 4] = 1 + 1 = 2 < p[4] = 65535$  nên  $p[4]$  được sửa thành 3 và  $\text{before}[4]$  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 lắp  $k = 4$

$$i = \min = 4 \text{ với } p[4] = 3.$$

Đỉnh chưa xử lí và kè 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 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  $k = 5$

$$i = \min = ① \text{ với } p[1] = 4.$$

Không có đỉnh chưa xử lí nào kè 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  $k = 6$

$$i = \min = ⑥ \text{ với } p[6] = 5.$$

Không có đỉnh chưa xử lí nào kè với đỉnh ⑥. Chú ý rằng đỉnh 7 kè 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 họa 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 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

$$3 \rightarrow 7 \rightarrow 4$$

Đỉnh	d	p	before
1	1	4	2
2	1	0	0
3	1	1	2
4	1	3	7
5	0	65535	0
6	1	5	4
7	1	2	3

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  $\infty$  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 kè có thể chứa các giá trị thực, tuy nhiên cần giả thiết rằng mọi giá trị trong ma trận kè 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 kè  $a$ .

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 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 con lâi
     * -----*/
    class Dijkstra
```

```

{
    const string fn = "Dij.inp";
    const string gn = "Dij.out";
    static int n = 0; // so dinh
    static int s = 0; // dinh xuat phat
    // 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 xuat phat
    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 xuat phat
        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 2

## Các hàm Next

Trong hầu hết các bài của Chương, khi trình bày tham biến kiểu mảng trong các hàm và thủ tục ta giả thiết là các kiểu này đã được khai báo trước. Thí dụ, kiểu mảng nguyên một chiều được khai báo như sau:

```
(* Pascal *) type mil = array[0..MN] of integer;  
trong đó MN là hằng số đủ lớn cho kích thước mỗi bài toán, thí dụ  
const MN = 2000;  
Trong C# mảng được khai báo trực tiếp hoặc thông qua class, thí dụ,  
int [] a = new int [2000];  
Class Array { ... };
```

Tùy theo bài toán và ngôn ngữ lập trình đã chọn, ta có thể hoặc không sử dụng phần tử đầu tiên và cuối cùng của mảng. Như vậy, mảng x gồm n phần tử sẽ được kí hiệu là  $x[1..n]$  trong Pascal hoặc  $x[0..n-1]$  trong C#. Trong Pascal khai báo tham biến kiểu var (truyền theo biến hay địa chỉ) cho mảng thì thủ tục sẽ được gọi nhanh hơn, trong C# các mảng được ngầm định là truyền theo biến / địa chỉ.

### Bài 2.1 Số sát sau cùng độ cao

Chiều dài của một số tự nhiên là số chữ số của số đó. Độ cao của một số tự nhiên là tổng các chữ số của số đó. Cho số tự nhiên  $x$  ghi trong hệ đếm  $b$ , có chiều dài  $N$ . Tìm số tự nhiên  $y$  sát sau  $x$  có cùng chiều dài, cùng độ cao và cùng hệ đếm với  $x$ .

DOCAO.INP	DOCAO.OUT
10 5 2 3 9 9 0	1 2 4 0 8 9

Dữ liệu vào: tệp văn bản DOCAO.INP

Dòng đầu tiên: hai số tự nhiên  $b$  và  $N$  cách nhau qua dấu cách,  $2 \leq b \leq 100$ ,  $2 \leq N \leq 1000$ .

Dòng thứ hai: số  $x$  với các chữ số ghi cách nhau qua dấu cách.

Dữ liệu ra: tệp văn bản DOCAO.OUT

Dòng đầu tiên: ghi 1 nếu có nghiệm, 0: nếu vô nghiệm.

Dòng thứ hai: số  $y$  với các chữ số ghi cách nhau qua dấu cách.

### Thuật toán

Độ cao của số  $x$  sẽ không đổi nếu ta đồng thời tăng và giảm hai chữ số của  $x$  cùng một đơn vị. Ta duyệt lần lượt các chữ số của  $x$  từ phải qua trái, trước hết tìm chữ số  $x_j > 0$  đầu tiên để có thể giảm 1 đơn vị. Tiếp đến ta duyệt tiếp từ  $j-1$  qua trái tìm một chữ số  $x_i < (b-1)$  đầu tiên sau  $j$  để có thể tăng thêm 1 đơn

vị. Nếu không tìm được  $x_j$  hoặc  $x_i$  thì  $x$  không có số sát sau. Nếu tìm được đồng thời hai chữ số  $x_j$  và  $x_i$  như trên thì ta sửa  $x$  như sau:

- Giảm  $x_j$  1 đơn vị,
- Tăng thêm  $x_i$  1 đơn vị,
- Lật lại đoạn  $x[i+1..n]$ .

Với thí dụ  $x[1..5] = (2,\underline{3},9,\underline{9},0)$  trong hệ đếm thập phân ( $b = 10$ ) ta tìm được  $j = 4$ ,  $x[j] = 9$ ,  $i = 2$ ,  $x[i] = 3$ . Sau khi giảm  $x[4]$  và tăng  $x[2]$  1 đơn vị ta thu được  $x[1..5] = (2,\underline{4},9,\underline{8},0)$ . Số này còn lớn, nếu lật lại đoạn  $x[3..5]$  sẽ thu được  $x[1..5] = (2,\underline{4},0,\underline{8},9)$ . Đây là số cần tìm.

Vì sao lại làm như vậy? Giải thích điều này khá dễ nếu để ý rằng  $x[j+1..n]$  chứa toàn 0 (chữ số nhỏ nhất trong hệ đếm b) và  $x[i+1..j-1]$  chứa toàn  $(b-1)$  (chữ số lớn nhất trong hệ đếm b). Từ đó suy ra rằng đoạn  $x[i+1..n]$  được sắp tăng. Lật lại đoạn đó ta sẽ thu được dãy các chữ số giảm dần. Vì  $x[i]$  đã được thêm 1 đơn vị nên nó lớn hơn số ban đầu. Khi lật lại ta sẽ thu được số sát sau số ban đầu.

Hàm Next dưới đây biến đổi trực tiếp  $x[1..n]$  để thu được số sát sau. Ta sử dụng phần tử  $x[0] = b$  làm giới hạn cho quá trình duyệt ngược. Phần tử  $x[0]$  này được gọi là *lính canh*. Nó có nhiệm vụ làm cho vòng lặp dừng một cách tự nhiên mà không cần phải kiểm tra giới hạn chỉ số của mảng (rang check).

Độ phức tạp:  $c\sqrt{N}$ , do mỗi chữ số của  $x$  được thăm và xử lí không quá 2 lần.

#### (\* Pascal \*)

```
function Next(var x: mil; n,b: integer): Boolean;
  var i,j,t,b1: integer;
begin
  Next := FALSE;
  x[0] := b; j := n;
  while (x[j] = 0) do j := j - 1;
  if (j = 0) then exit; { ko co so sat sau }
  i := j - 1; b1 := b - 1;
  while (x[i] = b1) do i := i - 1;
  if (i = 0) then exit; { Ko co so sat sau }
  x[j] := x[j] - 1; x[i] := x[i] + 1;
  i := i + 1; j := n;
  { Lat doan x[i..n] }
  while (i < j) do
    begin
      t := x[i]; x[i] := x[j]; x[j] := t;
      i := i + 1; j := j - 1;
    end;
  Next := TRUE;
end;
```

#### // C#

```
static bool Next(int[] x, int n, int b) {
  int i, j, b1 = b - 1;
  for (j = n - 1; j >= 0; --j)
    if (x[j] > 0) break;
  if (j < 0) return false;
  for (i = j - 1; i >= 0; --i)
    if (x[i] < b1) break;
  if (i < 0) return false;
  --x[j]; ++x[i];
  ++i; j = n - 1;
  int t;
  while (i < j) {
    t = x[i]; x[i] = x[j]; x[j] = t;
```

```

        ++i; --j;
    }
    return true;
}

```

## Bài 2.2 Số sát sau cùng chữ số

Cho số tự nhiên  $x$  chiều dài  $N$ . Hãy đổi chỗ các chữ số của  $x$  để thu được số  $y$  sát sau số  $x$ .

NXT.INP	NXT.OUT
6 239521	1 251239

Dữ liệu vào: tệp văn bản NXT.INP

Dòng đầu tiên: số tự nhiên  $N$ ,  $2 \leq N \leq 1000$ .

Dòng thứ hai: số  $x$

Dữ liệu ra: tệp văn bản NXT.OUT

Dòng đầu tiên: ghi 1 nếu có nghiệm, 0: nếu vô nghiệm. Dòng thứ hai: số  $y$ .

### Thuật toán

Trước hết để ý rằng muốn thu được số sát sau của  $x$  thì ta phải sửa các chữ số ở hàng thấp nhất có thể của  $x$ , do đó thuật toán sẽ duyệt các chữ số của  $x$  từ phải qua trái. Ta sẽ tìm hai chữ số  $x_j$  và  $x_i$  đầu tiên của  $x$  tính từ phải qua trái thỏa các điều kiện sau:

Thuận thế phải nhất:  $x_i < x_j$ ,  $1 \leq i < j \leq N$ :  $x_i$  đứng trước  $x_j$  và nhỏ hơn  $x_j$ .

Nếu không tìm được hai chữ số như vậy tức là  $x[1..n]$  là dãy được sắp giảm dần thì mọi hoán vị các chữ số của  $x$  không thể cho ra số lớn hơn  $x$ : bài toán vô nghiệm.

Nếu tìm được một thuận thế phải nhất ( $x_i, x_j$ ) như trên thì ta sửa  $x$  như sau:

- Đổi chỗ  $x_i$  và  $x_j$ ,
- Lật lại đoạn  $x[i+1..n]$ .

Với thí dụ  $x[1..6] = (2, \underline{3}, 9, \underline{5}, 2, 1)$  ta tìm được:  $i = 2$ ,  $x[2] = \underline{3}$ ,  $j = 4$ ,  $x[4] = \underline{5}$ .

Sau khi hoán vị  $x[i]$  và  $x[j]$  ta thu được,  $x = (2, \underline{5}, 9, \underline{3}, 2, 1)$

Số này còn lớn, nếu lật lại đoạn  $x[3..6]$  sẽ thu được,  $x = (2, \underline{5}, 1, 2, \underline{3}, 9)$ . Đây là số cần tìm.

Dưới đây là thuật toán vận dụng thuận thế phải nhất để tạo ra số sát sau theo điều kiện của đầu bài.

1. Tìm điểm gãy: Duyệt ngược  $x[1..n]$  để tìm  $i$  đầu tiên thỏa  $x[i] < x[i+1]$ . Nếu tìm được  $i$  thì thực hiện bước 2, ngược lại: dừng thuật toán với kết quả vô nghiệm.

2. Tìm điểm vượt: Duyệt ngược  $x[i..n]$  để tìm  $j$  đầu tiên thỏa  $x[i] < x[j]$ . Để ý rằng, nếu đã tìm được  $i$  thì  $j$  luôn tồn tại (?).

3. Hoán vị  $x[i]$  và  $x[j]$ ,
4. Lật đoạn  $x[i+1..N]$ .

**Độ phức tạp:** Cỡ  $N$  vì mỗi chữ số được thăm và xử lí không quá 2 lần.

Hàm Next dưới đây sửa trực tiếp  $x[1..n]$  để thu được số sát sau. Vì số  $x$  có thể có đến 1000 chữ số nên ta biểu diễn  $x$  theo kiểu mảng kí tự với khai báo `type mc1 = array[0..1000] of char`. Ta cũng sử dụng phần tử  $x[0]$  làm lính canh và khởi trị  $x[0] := pred('0')$  là kí tự sát trước chữ số 0.

(\* Pascal \*)

```

function Next(var x: mc1; n: integer): Boolean;
var i,j: integer;
t: char;
begin
Next := false; x[0] := pred('0');
{ Tim diem gay } i := n - 1;
while (x[i] >= x[i + 1]) do i := i - 1;

```

```

{ x[i] < x[i+1] }
if (i = 0) then exit; { Ko co diem gay: vo nghiem }
{ Tim diem vuot } j := n;
while (x[j] <= x[i]) do j := j - 1;
{ Doi cho } t := x[i]; x[i] := x[j]; x[j] := t;
{ Lat doan x[i+1..n] } i := i + 1; j := n;
while (i < j) do
begin
    t := x[i]; x[i] := x[j]; x[j] := t;
    i := i + 1; j := j - 1;
end;
Next := true;
end;

// C#
static bool Next(char[] x, int n) {
    int i, j ;
    // Tim diem gay i
    for (i = n - 2; i >= 0; --i)
        if (x[i] < x[i+1]) break;
    if (i < 0) return false; // vo nghiem
    // Tim diem vuot
    for (j = n-1; j > i; --j)
        if (x[j] > x[i]) break;
    char t = x[i]; x[i] = x[j]; x[j] = t; // Doi cho
    // Lat doan x[i+1..n-1]
    ++i; j = n-1;
    while (i < j){
        t = x[i]; x[i] = x[j]; x[j] = t;
        ++i; --j;
    }
    return true;
}

```

### Bài 2.3 Các hoán vị

Olimpic Moscva

*Liệt kê tăng dần theo thứ tự từ điển các hoán vị của các số 1..N.*

*Dữ liệu vào: tệp văn bản HV.INP chứa duy nhất số N,  $1 \leq N \leq 9$ .*

*Dữ liệu ra: tệp văn bản HV.OUT*

*Mỗi dòng một hoán vị.*

HV.INP	HV.OUT
3	1 2 3
	1 3 2
	2 1 3
	2 3 1
	3 1 2
	3 2 1

### Thuật toán

Sử dụng hàm Next trong bài trước. Khởi trị cho x là hoán vị đơn vị  $x = (1, 2, \dots, N)$ .

Độ phức tạp cho hàm Next:  $2N$ , cho cả bài:  $2N(N!)$ .

Trong các chương trình dưới đây ta xây dựng các hàm Next không có tham biến nhằm mục đích đẩy nhanh quá trình tính toán. Như vậy, dữ liệu được cho dưới dạng các biến tổng thể, bao gồm n - chiều dài của các hoán vị,  $x[0..n-1]$  - mảng chứa hoán vị.

```
(* Pascal *)
(* **** Liet ke cac hoan vi cua 1..N
   theo thu tu tang dan
*)
program CacHoanVi;
uses crt;
const
  bl = #32; mn = 10; fn = 'HV.INP'; gn = 'HV.OUT';
type
  mb1 = array[0..mn] of byte;
var
  x: mb1; { chua hoan vi }
  n: byte; { Len(x) }
  f,g: text; { input, output files }
procedure Doc;
begin
  assign(f,fn); reset(f); readln(f,n); close(f);
end;
function Next: Boolean;
  var i,j,t : byte;
begin
  Next := false;
  { Tim diem gay }
  i := n - 1;
  while (x[i] >= x[i + 1]) do i := i - 1;
  { x[i] < x[i+1] }
  if (i = 0) then exit;
  j := n;
  while (x[j] <= x[i]) do j := j - 1;
  t := x[i]; x[i] := x[j]; x[j] := t;
  i := i + 1; j := n;
  while (i < j) do
    begin
      t := x[i]; x[i] := x[j]; x[j] := t;
      i := i + 1; j := j - 1;
    end;
  Next := true;
end;
procedure Run;
  var i: byte;
begin
  Doc; x[0] := 0; // Dat linh canh
  assign(g,gn); rewrite(g);
  for i := 1 to n do x[i] := i;// Hoan vi don vi
  repeat
    for i := 1 to n do write(g,x[i],bl);
    writeln(g);
  until not Next;
  close(g);
end;
BEGIN
```

```

Run;
END.

// C#
using System;
using System.IO;
namespace SangTao2 {
    /*-----
     *          Cac Hoan Vi
     *  Liet ke cac hoan vi (1,2,...,n)
     *  theo trat tu tu dien tang dan
     * -----
    */
    class CacHoanVi {
        const string fn = "hv.inp";
        const string gn = "hv.out";
        static char[] x; // chua cac hoan vi
        static int n; // so phan tu
        static void Main(){
            Run();
            Console.ReadLine();
        } // Main
        static void Run() {
            n = int.Parse((File.ReadAllText(fn)).Trim());
            x = new char[n + 1];
            for (int i = 0; i < n; ++i)
                x[i] = (char) ('1' + i);
            StreamWriter g = File.CreateText(gn);
            do {
                for (int i = 0; i < n; ++i) g.WriteLine(x[i]);
                g.WriteLine();
            } while (Next());
            g.Close();
            XemKetQua();
        }
        // Hien thi du lieu de kiem tra
        static void XemKetQua() {
            Console.WriteLine(File.ReadAllText(fn));
            Console.WriteLine(File.ReadAllText(gn));
        }
        static bool Next(){
            int i, j;
            // Tim diem gay i
            for (i = n - 2; i >= 0; --i)
                if (x[i] < x[i + 1]) break;
            if (i < 0) return false; // vo nghiem
            // Tim diem vuot
            for (j = n - 1; j > i; --j)
                if (x[j] > x[i]) break;
            char t = x[i]; x[i] = x[j]; x[j] = t; // Doi cho
            // Lat doan x[i+1..n-1]
            ++i; j = n - 1;
            while (i < j){
                t = x[i]; x[i] = x[j]; x[j] = t;
                ++i; --j;
            }
            return true;
        }
    }
}

```

```

        }
    } // CacHoanVi
} // SangTao2

```

## Bài 2.4 Tô hợp

Liệt kê các tổ hợp chập K của N phần tử 1..N theo thứ tự từ điển tăng dần.

TOHOP.INP	TOHOP.OUT
5 3	1 2 3 1 2 4 1 2 5 1 3 4 1 3 5 1 4 5 2 3 4 2 3 5 2 4 5 3 4 5

Dữ liệu vào: tệp văn bản TOHOP.INP  
Dòng đầu tiên: hai số N và K cách nhau qua dấu cách,  
 $1 \leq N \leq 9, K \leq N$ .

Dữ liệu ra: tệp văn bản TOHOP.OUT

Mỗi dòng một tổ hợp, các số trên cùng dòng cách nhau qua dấu cách.

### Thuật toán

**Phương án 1.** Ta khởi trị cho mảng x[1..K] là tổ hợp nhỏ nhất (1,2,...,K). Sau đó ta dùng hàm Next để sinh ra tổ hợp sát sau của x. Hàm Next hoạt động theo 2 pha như sau:

Pha 1. Dõi. Duyệt ngược từ K qua trái bỏ qua những phần tử mang giá trị ...N-2, N-1, N đứng cuối mảng. Nếu sau khi dõi x không còn phần tử nào thì kết thúc với Next = false với ý nghĩa là sát sau tổ hợp x không còn tổ hợp nào. Thí dụ, nếu N = 7, K = 5,  $x[1..5] = (2,3,5,6,7)$  thì sau khi dõi ba phần tử cuối của x ta thu

được  $i = 2, x[1..2] = (2,3)$ . Điều này cho biết sẽ còn tổ hợp sát sau.

Pha 2. Xếp.

2.1. Tăng phần tử  $x[i]$  thêm 1 đơn vị. Tiếp tục với thí dụ trên ta thu được  $x[1..2] = (2,4)$

2.2. Xếp tiếp vào x cho đủ K phần tử theo trật tự tăng dần liên tục. Tiếp tục với thí dụ trên ta thu được  $x[1..5] = (2,4,5,6,7)$ .

Ta sử dụng phần tử  $x[0] = N$  làm lính canh.

(\* Pascal, Phương án 1 \*)

```

function Next: Boolean;
  var i, j, b: integer;
begin
  Next := false; x[0] := N;
  { Pha 1. Do }
  i := k; b := n - k;
  while (x[i] = b + i) do i := i - 1;
  if (i = 0) then exit;
  { Pha 2. Xep }
  x[i] := x[i] + 1;
  for j := i + 1 to k do x[j] := x[j-1] + 1;
  Next := true;
end;

```

Độ phức tạp: cho hàm Next:  $2N$ , cho cả bài:  $2N \cdot C_N^K = (2N \cdot N!) / (K! (N-K)!)$ .

**Phương án 2.** Ta cải tiến hàm Next như sau. Giả sử sau pha 1 ta thu được vị trí  $i$  thỏa  $x[i] \neq n-k+i$ . Ta gọi vị trí này là vị trí cập nhật và sẽ điều khiển nó thông qua một biến v. Ta khởi trị cho x và v như sau

```

for i := 1 to k do x[i] := i;
if (x[k] = n) then v := 0 else v := k;

```

Sau đó mỗi lần gọi hàm Next ta kiểm tra

Nếu  $v = 0$  thì dừng hàm Next.

Nếu  $v \neq 0$  ta thực hiện pha 2 sau đó chỉnh lại giá trị của  $v$  như sau:

Nếu  $x[k] = n$  thì tức là  $x[v..k] = (n-k-v, ..., n-1, n)$  thi lần gọi Next tiếp theo sẽ cập nhật tại vị trí  $v-1$ , ngược lại, nếu  $x[k] \neq n$  thì lần gọi Next tiếp theo sẽ cập nhật tại vị trí  $k$ .

Độ phức tạp: cho hàm Next: N. Cho cả bài:  $N.C_N^K = (N. N!) / (K! (N-K)!)$ .

(\* Pascal, Phương án 2 \*)

```
*****
To hop chap k cua n phan tu
PHUONG AN 2
*****)
program ToHopKN;
uses crt;
const
  bl = #32; mn = 10; fn = 'TOHOP.INP'; gn = 'TOHOP.OUT';
type
  mb1 = array[0..mn] of byte;
var
  x: mb1;
  n, k, v: byte;
  f,g: text;
procedure Doc;
begin
  assign(f,fn); reset(f); readln(f,n,k); close(f);
end;
function Next: Boolean;
  var i: byte;
begin
  Next := false;
  if (v = 0) then exit;
  { Pha 2. Xep }
  x[v] := x[v] + 1;
  for i := v + 1 to k do x[i] := x[i-1] + 1;
  if (x[k] = n) then v := v - 1 else v := k;
  Next := true;
end;
procedure Run;
  var i: byte;
begin
  Doc;
  assign(g,gn); rewrite(g);
  for i := 1 to k do x[i] := i;
  if (x[k] = n) then v := 0 else v := k;
  repeat
    for i := 1 to k do write(g,x[i],bl);
    writeln(g);
  until not Next;
  close(g);
end;
BEGIN
  Run;
END.

// C#
using System;
```

```

using System.IO;
namespace SangTao2 {
    /*-----
        To hop (Phuong an 2)
        Liet ke cac to hop chap k
        cua n phan tu 1, 2, ..., n
    -----*/
    class ToHop2 {
        const string fn = "ToHop.inp";
        const string gn = "ToHop.out";
        static int[] x;
        static int n = 0; // so phan tu nen
        static int k = 0; // so phan tu trong 1 to hop
        static int v = 0; // vi tri cap nhat trong x
        static void Main() {
            GhiToHop(); XemKetQua();
            Console.WriteLine("fini"); Console.ReadLine();
        } // Main
        // Doc lai cac tep inp va out de kiem tra
        static void XemKetQua() {
            Console.WriteLine(File.ReadAllText(fn));
            Console.WriteLine(File.ReadAllText(gn));
        }
        static bool Next(){
            if (v == 0) return false;
            ++x[v];
            for (int i = v + 1; i <= k; ++i) x[i] = x[i - 1] + 1;
            v = (x[k] == n) ? v - 1 : k;
            return true;
        }
        static void Doc(){
            char[] cc = new char[] { '\n', ' ', '\t', '\r' };
            string[] ss = (File.ReadAllText(fn)).Split(cc,
                StringSplitOptions.RemoveEmptyEntries);
            n = int.Parse(ss[0]); k = int.Parse(ss[1]);
        }
        static void GhiToHop(){
            Doc();
            // Tao tep ket qua ToHop.out
            StreamWriter g = File.CreateText(gn);
            // Khoi tri;
            x = new int[k + 1];
            for (int i = 1; i <= k; ++i) x[i] = i;
            v = (x[k] == n) ? 0 : k;
            do {
                for (int i = 1; i <= k; ++i) g.Write(x[i] + " ");
                g.WriteLine();
            } while (Next());
            g.Close();
        }
    } // ToHop2
} // SangTao2

```

**Chú ý** Bạn đọc lưu ý rằng thuật toán trên cho ra dây sáp tăng các tổ hợp và trong mỗi tổ hợp các thành phần cũng được sáp tăng.

## Bài 2.5 Số Kapreka

Số Kapreka mang tên nhà toán học Ấn Độ và được mô tả như sau. Đó là số tự nhiên  $x$  viết trong hệ đếm  $b$  có đúng  $K$  chữ số khác nhau đôi một và  $x = x'' - x'$ , trong đó  $x''$  và  $x'$  lần lượt là các số thu được bằng cách sắp lại các chữ số của số  $x$  theo trật tự giảm và tăng dần. Với mỗi cặp giá trị  $B$  và  $K$  hãy tìm một số Kapreka.

KAPREKA.INP	KAPREKA.OUT
10 4	6174

Dữ liệu vào: tệp văn bản KAPREKA.INP  
 Dòng đầu tiên: hai số  $B$  và  $K$  cách nhau qua dấu cách,  
 $2 \leq B \leq 10, K < B$ .  
 Dữ liệu ra: tệp văn bản KAPREKA.OUT  
 Số  $x$  viết trong hệ đếm  $B$ .  
 Bộ dữ liệu trên cho biết: Trong hệ đếm thập phân ( $B = 10$ ),  $x = 6174$  là số Kapreka có 4 chữ số (khác nhau đôi một),  
 $x'' - x' = 7641 - 1467 = 6174 = x$ .

## Thuật toán

Ta dựa vào thuật toán tổ hợp Next của bài trước, sinh lần lượt các số  $K$  chữ số trong hệ  $b$ . Lưu ý rằng hệ đếm  $b$  sử dụng b chữ số  $1..(b-1)$ . Với mỗi số  $x$  được sinh ra theo thuật toán Next ta tính hiệu  $y = x'' - x'$ , trong đó  $x''$  là số thu được bằng cách sắp lại các chữ số của  $x$  theo trật tự giảm dần và  $x' -$  tăng dần. Nếu  $y$  chỉ chứa các chữ số của  $x$  thì  $y$  chính là một số Kapreka. Do các tổ hợp  $x$  được sinh ra đã chứa các chữ số đôi một khác nhau và được sắp tăng, nên ta luôn có  $x'' = x$ .

Để tìm hiệu của hai số trong hệ  $b$  ta nên *biểu diễn ngược* các số dưới dạng mảng  $K$  phần tử nhận các giá trị trong khoảng  $0..b-1$ . Thí dụ số  $x = 1234$  trong hệ 10 sẽ được biểu diễn là  $x[1..4] = (4,3,2,1)$ .

Giả sử  $x = (x_1, x_2, \dots, x_K)$  và  $y = (y_1, y_2, \dots, y_K)$ . Ta tính hiệu  $z = x - y = (z_1, z_2, \dots, z_K)$  theo qui tắc sau:

Tính  $z = x + y^* + 1$ , trong đó  $y^*$  là dạng bù ( $b-1$ ) của  $y$ .

Sau đó ta bỏ đi số nhớ cuối cùng.

Dạng bù ( $b-1$ )  $y^* = (y_1^*, y_2^*, \dots, y_K^*)$  của số  $y$  được tính như sau:  $y_i^* = (b-1) - y_i, i = 1..K$ .

Thí dụ, tính  $9217 - 468$  trong hệ 10. Ta có  $x[1..4] = (7,1,2,9)$ ,  $y[1..4] = (8,6,4,0)$ , do đó  $y^*[1..4] = (1,3,5,9)$ . Vậy  $x - y = x + y^* + 1 = (7,1,2,9) + (1,3,5,9) + (1,0,0,0) = (9,4,7,8)$ . Kết quả là,  $9217 - 468 = 8749$ .

Qui tắc trên được giải thích như sau. Xét các số trong hệ đếm  $b$ . Kí hiệu  $z = b-1$ , khi đó số  $(z, z, \dots, z)$  gồm  $K$  chữ số  $z$  chính là  $b^K - 1$  và  $y^* = (b^K - 1) - y$ . Khi đó,  $x - y = x - y + (b^K - 1) + 1 - b^K = x + ((b^K - 1) - y) + 1 - b^K = x + y^* + 1 - b^K$ . Việc bỏ số nhớ cuối cùng tương đương với phép trừ  $b^K$  vào kết quả.

Dưới đây là thủ tục tính hiệu  $z = x - y$  cho các số viết ngược có tối đa  $K$  chữ số trong hệ  $b$ .

```

procedure Hieu;
  var i,c,t: integer;
begin
  c := 1; { so nho }
  for i := 1 to K do
    begin
      t := x[i] + ((b-1)-y[i]) + c;
      z[i] := t mod b;
      c := t div b;
    end;
end;

```

**Kaprekar D. R. (1905-1986)** là nhà toán học Ấn Độ say mê lý thuyết số từ nhỏ. Sau khi tốt nghiệp Đại học Tổng hợp Bombay năm 1929 ông làm giáo viên phổ thông tại Devlali, Ấn Độ. Ông viết nhiều bài khảo cứu nổi tiếng về lý thuyết số, ma phương và các tính chất kỳ lạ của thế giới số.

Để ý rằng phép cộng hai số một chữ số trong hệ đếm  $b > 1$  bắt kì cho số nhớ tối đa là 1. Ngoài ra do các phép toán **div** và **mod** thực hiện lâu hơn các phép cộng và trừ nên ta có thể viết lại thủ tục trên như sau.

```

procedure Hieu;
  var i,c,t: integer;
begin
  c := 1;
  for i := 1 to K do
    begin
      t := x[i] + (b-1-y[i]) + c;
      if (t >= b) then
        begin z[i] := t - b; c := 1; end
      else begin z[i] := t; c := 0; end;
    end;
end;

```

Với số  $x$  có  $K$  chữ số sắp tăng túc là dạng viết ngược của  $x''$  ta có thể thực hiện phép trừ  $y = x'' - x'$  bằng các thao tác trên chính  $x$  theo hai chiều duyệt xuôi và ngược. Khi thực hiện phép lấy hiệu ta cũng đồng thời kiểm tra xem mỗi chữ số của  $y$  có xuất hiện đúng một lần trong  $x$  hay không. Nếu đúng, ta cho kết quả là **true**, ngược lại, ta cho kết quả **false**. Để thực hiện việc này ta dùng mảng  $d[1..K]$  đánh dấu sự xuất hiện của các chữ số trong  $x$  và  $y$ .

```

(*-----
  y = x'' - x' (he dem B)
-----*)
function Hieu: Boolean;
  var i,c,t: integer;
begin
  fillchar(d,sizeof(d),0); { mang danh dau }
  Hieu := false;
  { Ghi nhan cac xuat hien cua x[i] }
  for i := 1 to k do d[x[i]] := 1;
  c := 1; { c: so nho }
  for i := 1 to k do
    begin
      t := x[i] + (b - 1 - x[k-i+1]) + c;
      if (t >= b) then
        begin y[i] := t - b; c := 1; end
      else begin y[i] := t; c := 0; end;
      if (d[y[i]] = 0) then exit;
      if (d[y[i]] = 1) then d[y[i]] := 0;
    end;
  Hieu := true;
end;

```

Dưới đây cung cấp 15 thí dụ để bạn đọc test chương trình. Kết quả 0 cho biết không tồn tại số Kapreka cho trường hợp đó.

NN	B	K	Đáp số	N	B	K	Đáp số	NN	B	K	Đáp số
1	4	3	132	6	8	2	25	1 1	9	7	0
2	5	4	0	7	8	3	374	1 2	9	8	0
3	6	3	253	8	8	7	6417532	1 3	10	3	495

4	6	4	0		9	9	5	62853	1	10	4	6174
5	6	5	41532	0	1	9	6	0	1	10	9	864197532

15 thí dụ về các số Kapreka

(\* Pascal \*)

```

(****** So Kapreka *****)
program SoKapreka;
uses crt;
const mn = 11; fn = 'KAPREKA.INP'; gn = 'KAPREKA.OUT';
type mb1 = array[0..mn] of byte;
var x,y,d: mb1;
    b,k,b1,v: integer;
{-----
  b - he dem
  k - so chu so
  b1 - chu so lon nhat trong he b, b1 = b-1
  v - bien kiem soat cho ham Next
-----}
f,g: text;
procedure Doc;
begin assign(f,fn); reset(f); readln(f,b,k); close(f);
  b1 := b-1; { Chu so cao nhat trong he dem b }
end;
function Next: Boolean;
  var i: integer;
begin
  Next := false;
  if (v = 0) then exit;
  x[v] := x[v] + 1;
  for i := v + 1 to k do x[i] := x[i-1] + 1;
  if (x[k] = b1) then v := v - 1 else v := k;
  Next := true;
end;
{-----
  y = x' - x'
-----*)
function Hieu: Boolean;
  var i,c,t: integer;
begin
  fillchar(d,sizeof(d),0);
  Hieu := false;
  { Ghi nhau cac xuat hien cua x[i] }
  for i := 1 to k do d[x[i]] := 1;
  c := 1; { c: so nho }
  for i := 1 to k do
  begin
    t := x[i] + (b1 - x[k-i+1]) + c;
    if (t > b1) then
      begin t := t - b; c := 1; end
    else c := 0;
  end;
  if (d[0] = 1) then Hieu := true;
end;

```

```

        if (d[t] = 0) then exit; { t ko xuat hien trong x }
        y[i] := t; d[t] := 0;
    end;
    Hieu := true;
end;
function Kapreka: Boolean;
    var i: integer;
        t: Boolean;
begin
    Kapreka := true;
    { Khoi tri x la to hop tang nho nhat }
    { x[1..k] = (0,1,...,k-1) }
    for i := 1 to k do x[i] := i-1;
    if (x[k] = b1) then v := 0 else v := k;
    repeat
        if (Hieu) then exit;
        until not next;
    Kapreka := false;
end;
procedure Run;
    var i: byte;
begin
    Doc;
    assign(g,gn); rewrite(g);
    if (Kapreka) then
        for i := k downto 1 do write(g,y[i])
    else write(g,0);
    writeln(g); close(g);
end;
BEGIN
    Run;
END.

```

```

// C#
using System;
using System.IO;
namespace SangTao2 {
    /*
     * So Kapreka
     * x'' - x' = x
     * x'' - so giam
     * x' - so tang
     */
    class Kapreka {
        const string fn = "Kapreka.inp";
        const string gn = "Kapreka.out";
        static int[] x; // so x
        static int[] y; // y = x'' - x'
        static int[] d;
        static int b; // he dem
        static int k; // so chu so
        static int b1; // b-1: chu so cao nhat trong he b
        static int v; // bien cam canh
        static void Main() {
            Doc(); Ghi(Kap()); XemKetQua();
            Console.WriteLine("\n fini");
    }
}

```

```

        Console.ReadLine();
    } // Main
    static void Ghi(int ket){
        StreamWriter g = File.CreateText(gn);
        if (ket == 0) g.WriteLine(0);
        else for (int i = k; i > 0; --i) g.WriteLine(y[i]);
        g.Close();
    }
    // Doc lai cac tep inp va out de kiem tra
    static void XemKetQua(): tự viết
    static bool Next() {
        if (v == 0) return false;
        ++x[v];
        int j = x[v] + 1;
        for (int i = v + 1; i <= k; ++i, ++j) x[i] = j;
        v = (x[k] == b1) ? v - 1 : k;
        return true;
    }
    static void Doc() {
        char[] cc = new char[] { '\n', ' ', '\t', '\r' };
        string[] ss = (File.ReadAllText(fn)).Split(cc,
            StringSplitOptions.RemoveEmptyEntries);
        b = int.Parse(ss[0]); // he dem
        k = int.Parse(ss[1]); // so chu so
        b1 = b - 1; // chu so cao nhat cua he dem b
    }
    // y = x'' - x'
    static bool Hieu() {
        int c = 1, t = 0;
        Array.Clear(d, 0, d.Length);
        for (int i = 1; i <= k; ++i) d[x[i]] = 1;
        for (int i = 1; i <= k; ++i){
            t = x[i] + (b1 - x[k - i + 1]) + c;
            if (t > b1) { c = 1; t = t - b; }
            else c = 0;
            if (d[t] == 0) return false;
            y[i] = t; d[t] = 0;
        }
        return true;
    }
    static int Kap() { // Khoi tri;
        x = new int[k + 1];
        y = new int[k + 1];
        d = new int[b];
        for (int i = 1; i <= k; ++i) x[i] = i - 1;
        v = (x[k] == b1) ? 0 : k;
        do {
            if (Hieu()) return 1;
        } while (Next());
        return 0;
    }
} // class Kapreka
} // SangTao2

```

### Chú thích

Bạn có thể sử dụng thuật toán sau đây:

Khởi trị: Tạo số x hệ b gồm k chữ số khác nhau;  
 Lặp từ 1 đến  $b^k - 1$   
 Tính  $y = x' - x'$ ;  
 Nếu  $x = y$  thì cho kết quả x là số Kapreka; stop;  
 Nếu không gán  $x := y$ ;  
 Xong lặp.

## Bài 2.6 Khóa vòng

Một ổ khóa gồm M vòng chữ và N vòng số. Mỗi vòng chữ hoặc số chứa các giá trị biến thiên từ giới hạn nhỏ nhất a đến giới hạn lớn nhất b. Hãy liệt kê tăng dần theo trật tự từ điển các giá trị có thể có của khóa.

KHOA.INP	KHOA.OUT	
1 2	12	Dữ liệu vào: tệp văn bản KHOA.INP
B C	B20	Dòng đầu tiên: hai số tự nhiên M và N, $1 \leq M, N \leq 5$ .
2 3	B21	Dòng thứ i trong số M+N dòng tiếp theo: giới hạn $a_i$ và $b_i$ cho các vòng khóa.
0 2	B22	Dữ liệu ra: tệp văn bản KHOA.OUT
	B30	Dòng đầu tiên: Tổng số khả năng.
	B31	Từ dòng thứ hai trở đi: mỗi dòng một giá trị khóa liệt kê tăng dần theo trật tự từ điển. Các kí tự chữ và số trong mỗi khóa được viết liền nhau, không có dấu cách ở giữa. Các giá trị chữ được lấy từ bảng chữ HOA tiếng Anh.
	B32	
	C20	
	C21	
	C22	
	C30	
	C31	
	C32	

### Thuật toán

Phương pháp: duyệt toàn bộ các tổ hợp.

Nếu toàn bộ N vòng khóa đều chỉ chứa các chữ số với giới hạn biết trước từ cận dưới  $a[i]$  đến cận trên  $b[i]$ ,  $i = 1..N$  thì ta dùng hàm Next sinh ra lần lượt các tổ hợp N phần tử  $c[1..N]$  như sau.

Khởi trị:  $c[1..N]$  là tổ hợp nhỏ nhất chứa toàn cận dưới:

$c[i] := a[i]$ ,  $i = 1..N$ .

Xử lí:

repeat

Ghi tổ hợp  $c[1..N]$ ;

until not Next;

Mỗi lần gọi hàm Next ta thực hiện giống như phép đếm: Duyệt ngược  $c[1..N]$  với mỗi  $c[i] = b[i]$  ta đặt lại  $c[i] := a[i]$ . Gặp  $c[i]$  đầu tiên thỏa điều kiện  $c[i] < b[i]$  thì tăng vòng khóa i thêm 1 nấc. Nếu không gặp phần tử i như vậy thì chứng tỏ đã xử lí xong tổ hợp cao nhất.

Việc còn lại là chuyển các vòng chữ sang vòng số tương ứng.

Độ phức tạp:  $(b_1-a_1+1)(b_2-a_2+1)\dots(b_v-a_v+1)$ ,  $v = M+N$ .

(\* Pascal \*)

```

*****
Khoa Vong
*****)
program KhoaVong;
uses crt;
const mn = 20;

```

```

bl = #32; nl = #13#10; fn = 'KHOA.INP'; gn = 'KHOA.OUT';
ChuCai = ['A'..'Z'];
type mil = array[0..mn] of integer;
var m,n: integer;
    a,b,c: mil;
    f,g: text;
    m: longint;
procedure Doc;
    var i: integer;
    c: char;
begin
    assign(f,fn); reset(f); readln(f,m,n);
    n := m + n;
    for i := 1 to m do
        begin
            repeat
                read(f,c);
            until c in ChuCai;
            a[i] := ord(c) - ord('A');
            repeat
                read(f,c);
            until c in ChuCai;
            b[i] := ord(c) - ord('A');
        end;
    for i := m + 1 to n do read(f,a[i],b[i]);
    close(f);
    m := 1;
    for i := 1 to n do m := m * (b[i] - a[i] + 1);
end;
function Min(a,b: integer): integer;
begin
    if (a < b) then Min := a else Min := b;
end;
function Next: Boolean;
    var i: integer;
begin
    Next := false;
    i := n;
    while (c[i] = b[i]) do
        begin
            c[i] := a[i];
            i := i - 1;
        end;
    if (i = 0) then exit;
    c[i] := c[i] + 1;
    Next := true;
end;
procedure Duyet;
    var i: integer;
begin
    for i := 1 to n do c[i] := a[i];
    c[0] := -1;
    assign(g,gn); rewrite(g); writeln(g,m);
    repeat
        for i := 1 to m do write(g,chr(ord('A')+c[i]));
        for i := m + 1 to n do write(g,c[i]);
        writeln(g);

```

```

        until not Next;
        close(g);
    end;
BEGIN
    Doc; Duyet;
END.

// C#
using System;
using System.IO;
namespace SangTao2 {
/*
 *                      Khoa Vong
 * -----*/
class KhoaVong {
    const string fn = "Khoa.inp";
    const string gn = "Khoa.out";
    static int [] x; // to hop
    static int[] vmin; // can duoi
    static int[] vmax; // can tren
    static int m; // so luong vong chu
    static int n; // so luong vong so
    static int mn; // m+n
    static void Main() {
        Doc(); Ghi(); XemKetQua(); Console.ReadLine();
    } // Main
    // Doc lai cac tep inp va out de kiem tra
    static void XemKetQua(): tự viết
    static bool Next() {
        int i;
        for (i = mn - 1; i >= 0; --i)
            if (x[i] == vmax[i]) x[i] = vmin[i];
            else break;
        if (i < 0) return false;
        ++x[i];
        return true;
    }
    static void Doc() {
        char [] cc = new char [] {'\n',' ','\t','\r'};
        string [] ss = (File.ReadAllText(fn)).Split(cc,
            StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        m = int.Parse(ss[k++]); // m vong chu
        n = int.Parse(ss[k++]); // n vong so
        mn = m + n;
        vmin = new int [mn];
        vmax = new int [mn];
        for (int i = 0; i < m; ++i) {
            vmin[i] = (int)ss[k++][0] - (int)'A';
            vmax[i] = (int)ss[k++][0] - (int)'A';
        }
        for (int i = m; i < mn; ++i) {
            vmin[i] = int.Parse(ss[k++]);
            vmax[i] = int.Parse(ss[k++]);
        }
    }
}

```

```

        static void Ghi() {
            StreamWriter g = File.CreateText(gn);
            // khai tri x
            x = new int[mn];
            for (int i = 0; i < mn; ++i) x[i] = vmin[i];
            do {
                for (int i = 0; i < m; ++i)
                    g.Write((char)(x[i] + (int)'A'));
                for (int i = m; i < mn; ++i)
                    g.Write(x[i]);
                g.WriteLine();
            } while (Next());
            g.Close();
        }
    } // KhoaVong
} // SangTao2

```

## Bài 2.7 Trả tiền

Có  $N$  loại tiền mệnh giá  $m_i$  và số lượng  $s_i$ ,  $i = 1..N$ . Xác định số lượng mỗi loại để có thể trả lại  $V$  đồng.

TRATIEN.INP	TRATIEN.OUT
6 156 1 2 5 10 20 50 4 7 2 3 6 2	0 3 0 0 5 1

Dữ liệu vào: tệp văn bản **TRATIEN.INP**

Dòng đầu tiên: hai số tự nhiên  $N$  và  $V$ ,  $2 \leq N \leq 15$ .

Dòng thứ hai:  $N$  số tự nhiên  $m_1, m_2, \dots, m_N$ .

Dòng thứ ba:  $N$  số tự nhiên  $s_1, s_2, \dots, s_N$ .

Dữ liệu ra: tệp văn bản **TRATIEN.OUT**

$N$  số tự nhiên  $c_1, c_2, \dots, c_N$  thể hiện số lượng tờ tiền mỗi loại cần trả,  $c_1m_1 + c_2m_2 + \dots + c_Nm_N = V$ . Nếu vô nghiệm: ghi số 0.

Trong các tệp \*.INP và \*.OUT các số trên cùng dòng cách nhau qua dấu cách.

## Thuật toán

Đây là loại toán Balo với dữ liệu nhỏ vì trong thực tế số mệnh giá không nhiều, thí dụ, tiền Việt chỉ có các loại sau đây là thông dụng 100, 200, 500, 1.000, 2.000, 5.000, 10.000, 20.000, 50.000, 100.000, 200.000, 500.000. Nếu tính theo đơn vị 100 đồng thì ta có thể viết lại dây trên cho gọn hơn như sau:

1, 2, 5, 10, 20, 50, 100, 200, 500, 1.000, 2.000, 5.000.

Ta duyệt các tổ hợp số tờ tiền phải trả cho mỗi loại mệnh giá, cận dưới là 0 cận trên là  $\min(s_i, v \text{ div } m_i)$  vì để trả lại  $v$  đồng bằng loại mệnh giá  $m_i$  ta dùng tối đa ( $v \text{ div } m_i$ ) tờ.

Độ phức tạp:  $(b_1-a_1+1)(b_2-a_2+1)\dots(b_v-a_v+1)$ ,  $v = M+N$ .

Chú ý: Sau này ta sẽ xây dựng thuật toán tốt hơn cho bài toán trả tiền. Thuật toán này dựa trên một số kiến thức số học.

```

(* Pascal *)
(*****)
      Tra tien
(*****)
program TraTien;
uses crt;
const mn = 20; bl = #32; nl = #13#10;
fn = 'TRATIEN.INP'; gn = 'TRATIEN.OUT';
type mil = array[0..mn] of integer;

```

```

(*-----
  n - so luong cac loai tien
  v - so tien can tra lai
  vt - gia tri tam thoi
  m[1..n] - cac menh gia
  s[1..n] - so luong to tien
  c[1..n] - so luong can chon
-----*)

var n,v,vt: integer;
    m,s,c: mil;
    f,g: text;
procedure Doc;
    var i: integer;
begin
    assign(f,fn); reset(f); readln(f,n,v);
    for i := 1 to n do read(f,m[i]);
    for i := 1 to n do read(f,s[i]);
    close(f);
end;
function Min(a,b: integer): tuy viet
function Next: Boolean;
    var i: integer;
begin
    Next := false;
    i := n;
    while (c[i] = s[i]) do
        begin
            vt := vt - c[i] * m[i];
            c[i] := 0;
            i := i - 1;
        end;
    if (i = 0) then exit;
    c[i] := c[i] + 1;
    vt := vt + m[i];
    Next := true;
end;
function Duyet: Boolean;
    var i: integer;
begin
    { Khoi tri }
    for i := 1 to n do
        begin
            s[i] := min(s[i],v div m[i]);
            c[i] := 0;
        end;
    c[0] := -1; vt := 0; { tong gia tri cua 1 phuong an }
    Duyet := true;
repeat
    if (vt = v) then exit;
until not Next;
Duyet := false;
end;
procedure Run;
    var i: integer;
begin
    Doc; assign(g,gn); rewrite(g);
    if (Duyet) then

```

```

        for i := 1 to n do write(g,c[i],bl);
        else writeln(g,0);
    close(g);
end;
BEGIN
    Run;
END.

// C#
using System;
using System.IO;
namespace SangTao2 {
/*
 *                  Tra Tien
 * -----
class TraTien {
    const string fn = "TraTien.inp";
    const string gn = "TraTien.out";
    static int[] c; // phuong an dang duyet
    static int[] s; // so luong to tien
    static int[] m; // menh gia
    static int n; // so luong menh gia
    static int v; // tong so tien can tra
    static int t; // tong so tien cua 1 phuong an
    static void Main() {
        Doc();
        int kq = XuLi();
        Ghi(kq);
    } // Main
    static bool Next() {
        int i = n;
        while (c[i] == s[i]) { t -= c[i] * m[i]; c[i] = 0; --i; }
        if (i == 0) return false;
        ++c[i]; t += m[i]; return true;
    }
    static void Doc() {
        char[] cc = new char[] { '\n', ' ', '\t', '\r' };
        string[] ss = (File.ReadAllText(fn)).Split(cc,
            StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        n = int.Parse(ss[k++]); // n so luong tien
        v = int.Parse(ss[k++]); // v so tien can tra lai
        m = new int[n + 1]; // cac menh gia
        s = new int[n + 1]; // so luong to moi loai
        for (int i = 1; i <= n; ++i)
            m[i] = int.Parse(ss[k++]);
        for (int i = 1; i <= n; ++i)
            s[i] = Min(v/m[i], int.Parse(ss[k++]));
    }
    static int Min(int a, int b) { return (a < b) ? a : b; }
    static int XuLi() {
        c = new int[n + 1];
        for (int i = 1; i <= n; ++i) c[i] = 0;
        t = 0;
        do { if (t == v) return 1; } while (Next());
        return 0;
    }
}

```

```

        }
        static void Ghi(int kq) {
            Streamwriter g = File.CreateText(gn);
            if (kq == 0) g.WriteLine(kq);
            else for (int i = 1; i <= n; ++i) g.WriteLine(c[i] + " ");
            g.Close();
        }
    } // TraTien
} // SangTao2

```

## Bài 2.8 Dãy Farey

Cho số tự nhiên  $N > 0$ . hãy liệt kê theo trật tự tăng dần các phân số  $t/m$  thỏa đồng thời các tính chất sau:

- $t/m$  là phân số tối giản biến thiên trong khoảng  $0..1$ ,
- $m$  biến thiên trong khoảng  $1..N$ .

FAREY.INP	FAREY.OUT
5	11 0 1 1 5 1 4 1 3 2 5 1 2 3 5 2 3 3 4 4 5 1 1

Dữ liệu vào: tệp văn bản FAREY.INP chứa số  $N$ .

Dữ liệu ra: tệp văn bản FAREY.OUT

Dòng thứ nhất:  $D$  – số lượng các phân số trong dãy.

Tù dòng thứ hai: mỗi dòng hai số tự nhiên  $t/m$  ghi cách nhau qua dấu cách, thể hiện một phân số trong dãy sắp tăng.

### Thuật toán

Nếu sinh lần lượt các phân số (PS) rồi sắp xếp thì khả tổn bộ nhớ vì tối đa phải dành đủ bộ nhớ để lưu trữ  $n^2$  PS.

**Phương án 1.** Nếu  $t/m$  và  $a/b$  là hai PS số liên tiếp trong dãy Farey thì

Farey là nhà địa chất học người Anh. Ông mô tả dãy phân số trên vào năm 1816.

$$a/b = \min \{ x/y \mid x/y > t/m, y = 1..n, x \leq y, (x,y) = 1 \}$$

trong đó  $(x,y)$  là ước chung lớn nhất của  $x$  và  $y$ .

Các PS  $x/y$  trong tập trên được gọi là các ứng viên. Ta sẽ đề cử càng ít ứng viên càng tốt.

Với  $y = 1$ , do  $x \leq y$  nên ta có ngay PS  $1/1$  là phần tử lớn nhất trong dãy.

Với mỗi  $y = 2..n$  ta xét PS  $x/y$  là PS đầu tiên lớn hơn  $t/m$ .

Ta có từ  $t/m < x/y$  ta suy ra  $mx > ty$  nên  $x > (ty \text{ div } m)$ . Nếu biết  $m$  ta chọn  $x = (ty \text{ div } m) + 1$  sẽ thu được PS  $x/y$  thỏa đồng thời các tính chất sau:

- $1 \leq m \leq n$
- $x/y$  là PS đầu tiên lớn hơn  $t/m$ .

Đặc tả trên được thu gọn lại với  $n-1$  ứng viên như sau,

$$a/b = \min \{ x/y \mid y = 2..n, x = (ty \text{ div } m) + 1 \}$$

Như vậy, nếu đã sinh được PS  $t/m$  cho dãy Farey thì PS tiếp theo  $a/b$  sẽ được chọn là PS nhỏ nhất trong tập  $n-1$  PS nói trên. Đề ý rằng  $0/1$  là PS đầu tiên và  $1/1$  là PS cuối cùng của dãy Farey. Thủ tục Next( $n,t,m$ ) trong phương án 1 sẽ xác định PS  $a/b$  sát sau PS  $t/m$  trong dãy Farey. Giá trị tìm được sẽ đặt ngay trong  $t/m$ .

Độ phức tạp. Xuất phát từ PS đầu tiên 0/1, mỗi lần ta phải sinh ra  $n-1$  ứng viên để từ đó chọn ra 1 PS trong dãy. Nếu dãy có s PS thì ta phải thực hiện  $s(n-1)$  phép toán trên các PS. Giá trị max của s là  $n^2$ . Vậy độ phức tạp tính toán vào cỡ  $n^3$ .

## Bình luận

Nếu từ PS t/m trong dãy Farey và giá trị mẫu số y trong khoảng  $2..n$  cho trước ta sinh ra PS x/y thông qua hệ thức  $x = (ty \text{ div } m) + 1$  thì PS x/y có thể chưa tối giản. Thí dụ, với  $n = 15$ ,  $t/m = 3/4$ ,  $y = 12$  ta có  $x = (ty \text{ div } m) + 1 = 10$  thì PS  $10/12$  không tối giản do đó ta cần gọi thủ tục RutGon để giản ước PS x/y.

Vì không tính trước được số lượng các PS trong dãy nên ta cần đếm dần và ghi tạm dãy PS vào tệp **FAREY.TMP**. Sau đó mở tệp **FAREY.OUT** ghi số lượng s và chuyển dữ liệu từ tệp **FAREY.TMP** sang tệp **FAREY.OUT**, cuối cùng xóa tệp **FAREY.TMP**.

**Phương án 2.** Ta có thể sinh dần các phân tử cho dãy Farey như sau. Cho hai PS  $a/b$  và  $c/d$ , PS  $(a+c)/(b+d)$  được gọi là PS trung bình của hai PS này.

**Nhận xét.** Nếu  $t_1 / m_1, t_2 / m_2, t_3 / m_3$  là ba PS liên tiếp trong dãy Farey thì PS giữa là PS trung bình của hai PS kia.

Ta có thuật toán sau:

Xuất phát với mẫu số  $m = 1$  ta có dãy 2 PS:  $0/1, 1/1$ .

Với mỗi mẫu số  $m = 2..n$  ta sinh các PS trung bình có mẫu số  $m$  của hai PS kè nhau trong dãy trước và xen PS này vào giữa hai PS sinh ra nó dần vào trong dãy kết quả.

$m = 2$ : thêm các PS trung bình với mẫu bằng 2:  $0/1, \underline{1/2}, 1/1$ .

$m = 3$ : thêm các PS trung bình với mẫu bằng 3:  $0/1, \underline{1/3}, 1/2, \underline{2/3}, 1/1$ .

...

Các phân số mới sinh trong mỗi lần duyệt được gạch dưới.

Ta dùng hai mảng: a lưu các PS của dãy trước, b lưu các PS của dãy sau. Sau mỗi bước lặp ta chuyển b qua a. Dữ liệu được mô tả như sau:

```
const mn = 1000;
type
  PS = record tu, mau: byte end;
  mps = array[0..mn] of PS; { mảng các PS }
var a,b: mps;
```

Độ phức tạp. Thời gian:  $n^3$ , miền nhớ: 2 mảng kích thước  $n^2$ .

**Phương án 3.** Ta sử dụng một số tính chất của dãy Farey để tiếp tục cải tiến thuật toán.

Nếu  $t_1 / m_1, t_2 / m_2, t_3 / m_3$  là ba PS liên tiếp trong dãy Farey thì

1.  $t_2m_1 - t_1m_2 = 1$ ,
2.  $m_1 + m_2 > n$ ,
3.  $t_2 / m_2 = (t_1 + t_3) / (m_1 + m_3)$ ,
4.  $t_3 = vt_2 - t_1, m_3 = vm_2 - m_1$  với  $v = (m_1 + n) \text{ div } m_2$ .

Từ tính chất 4 ta suy ra ngay cách xác định PS  $t_3/m_3$  thông qua hai PS sát trước.

Các trình dưới đây minh họa 3 phương án với các kết quả hiển thị trên màn hình để bạn đọc có thể theo dõi.

```
(* Pascal *)
(*-----*
  Ba phuong an cho bai Day Farey
-----*)
uses crt;
const b1 = #32; nl = #13#10;
var n: integer;
{ Uoc chung lon nhat cua hai so tu nhien a, b }
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;
{ Rut gon PS a/b thanh PS t/m }
procedure RutGon(a,b:integer; var t,m:integer);
var d:integer;
begin d :=Ucln(a,b); t := a div d; m := b div d; end;
{ Tim PS sat sau PS t/m, ket qua dat trong t/m }
function Next(n: integer; var t,m: integer): Boolean;
var a,b,x,y: integer;
begin
  if (t+m=2) then begin Next := false; exit end;
  a := 1; b := 1;
  for y := 2 to n do
  begin
    x := t*y div m + 1;
    if a*y > b*x then begin a := x; b:=y end;
  end;
  RutGon(a,b,t,m); Next := true;
end;
procedure Farey1(n: integer);
  var t,m,d:integer;
begin
  writeln(nl,'Farey1'); d := 0;
  t := 0; m := 1;
  repeat
    write(t,'/',m,bl); inc(d);
  until not Next(n,t,m);
  writeln(nl,'Total: ',d,' PS');
  readln;
end;
procedure Farey2(n: byte);
const mn = 1000;
type PS = record tu,mau: byte end;
  mps1 = array[0..mn] of PS;
var a,b: mps1; { 2 day PS a , b }
  d,k,i,m:integer;
begin
  writeln(nl,'Farey2'); d := 2;
  a[1].tu := 0; a[1].mau := 1; { PS dau day }
  a[2].tu := 1; a[2].mau := 1; { PS thu hai }
  for m:=2 to n do
  begin
    begin
      k := 0; inc(k); b[k] := a[k];
      for i := 2 to d do
      begin
        if a[i].mauta[i-1].mau = m then
        begin
          inc(k); b[k].tu := a[i-1].tu + a[i].tu;
          b[k].mau := a[i-1].mau + a[i].mau;
        end;
        inc(k); b[k] := a[i];
      end;
      a := b; d := k;
    end;
  end;
end;

```

```

for i := 1 to d do write(a[i].tu,'/',a[i].mau,bl);
writeln(nl,'Total ',d,' PS');
readln;
end;
procedure Farey3(n: integer);
  var t1,m1,t2,m2,t3,m3,v,d: integer;
begin
  writeln(nl,'Farey3'); d := 2;
  t1 := 0; m1 := 1; { PS dau day }
  t2 := 1; m2 := n; { PS thu hai }
  write(t1,'/',m1,bl,t2,'/',m2,bl);
  while (t2 + m2 <> 2) do
  begin
    v := (m1+n) div m2;
    t3 := v*t2 - t1; m3 := v*m2 - m1;
    write(t3,'/',m3,bl); inc(d);
    t1 := t2; t2 := t3;
    m1 := m2; m2 := m3;
  end;
  writeln(nl,'Total ',d,' PS'); readln;
end;
BEGIN
  n := 5;
  Farey1(n); Farey2(n); Farey3(n);
END.

```

```

// C#
using System;
using System.IO;
namespace SangTao2 {
/*-----*
 * Ba phuong an cho bai Day Farey
 * -----*/
class Farey {

    static void Main() {
        int n = 10;
        Farey1 x = new Farey1(); x.Run(n); Console.ReadLine();
        (new Farey2()).Run(n); Console.ReadLine();
        (new Farey3()).Run(n); Console.ReadLine();
        Console.WriteLine("\n fini");
        Console.ReadLine();
    } // Main
} // Farey
class Farey1 {
    public Farey1() { }
    public void Run(int n) {
        int d = 0;
        int t = 0, m = 1; // PS dau day
        do {
            Console.Write(t + "/" + m + " ");
            ++d;
        } while (Next(n, ref t, ref m));
        Console.WriteLine(" * Farey1: Total " + d + " PS");
    }
    public bool Next(int n, ref int t, ref int m) {

```

```

        if (t + m == 2) return false;
        int a = 1, b = 1, x, y;
        for (y = 2; y <= n; ++y) {
            x = t * y / m + 1;
            if (a * y > b * x) { a = x; b = y; }
        }
        RutGon(a, b, ref t, ref m);
        return true;
    }
    Console.WriteLine(" * Farey1: Total " + d + " PS");
}
public void Next(int n, ref int t, ref int m) {
    int a = 1, b = 1, x, y;
    for (y = 2; y <= n; ++y) {
        x = t * y / m + 1;
        if (a * y > b * x) { a = x; b = y; }
    }
    RutGon(a, b, ref t, ref m);
}
public int Ucln(int a, int b) {
    int r;
    while (b != 0) { r = a % b; a = b; b = r; }
    return a;
}
public void RutGon(int a, int b, ref int t, ref int m)
{ int d = Ucln(a, b); t = a / d; m = b / d; }
} // Farey1
class Farey2 {
    public Farey2() { }
    public void Run(int n) {
        int mn = 10000;
        PS[] a = new PS[mn];
        PS[] b = new PS[mn];
        int d = 0;
        a[d++] = new PS(0, 1); //PS dau day
        a[d++] = new PS(1, 1); // PS cuoi day
        for (int m = 2; m <= n; ++m) {
            int k = 0; b[k++] = a[0];
            for (int i = 1; i < d; ++i) {
                if (a[i].mau + a[i - 1].mau == m)
                    b[k++] = new PS(a[i], a[i - 1]);
                b[k++] = a[i];
            }
            d = k; Array.Copy(b, 0, a, 0, d);
        }
        for (int i = 0; i < d; ++i) a[i].Print();
        Console.WriteLine(" * Farey2: Total " + d + " PS");
    }
    public struct PS {
        public int tu, mau;
        public PS(int t, int m) { tu = t; mau = m; }
        public PS(PS x, PS y)
        { tu = x.tu + y.tu; mau = x.mau + y.mau; }
        public void Print()
        { Console.Write(tu + "/" + mau + " "); }
    } // PS
} // Farey2

```

```

class Farey3 {
    public Farey3() { }
    public void Run(int n) {
        int d = 2;
        int t1 = 0, m1 = 1, t2 = 1, m2 = n; // hai PS dau day
        int t3, m3, v;
        Console.WriteLine(t1 + "/" + m1 + " " + t2 + "/" + m2 + " ");
        while (t2 + m2 != 2) {
            ++d; v = (m1 + n) / m2;
            t3 = v * t2 - t1; m3 = v * m2 - m1;
            Console.WriteLine(t3 + "/" + m3 + " ");
            t1 = t2; m1 = m2; t2 = t3; m2 = m3;
        }
        Console.WriteLine(" * Farey3: Total " + d + " PS");
    }
} // Farey3
} // SangTao2

```

## Bài 2.9 Quý Mùi

Minh muốn làm một thiệp chúc Tết Quý Mùi có nền được tạo bởi  $2^n$  dòng, mỗi dòng là một dãy kí tự gồm  $n$  chữ cái ‘Q’ và ‘M’ sao cho hai dòng kề nhau khác nhau tại đúng một vị trí, dòng cuối cùng cũng được coi là kề với dòng đầu tiên. Giả sử bạn có thể giúp Minh làm điều đó. Với mỗi giá trị  $n$  và  $k$  cho trước bạn hãy hiển thị dòng thứ  $k$  trong tấm thiệp trên. Các dòng được mã số từ 1 trở đi,  $1 \leq n \leq 30$ .

### Thuật toán

Kí hiệu  $T(n)$  là tấm thiệp được thiết kế với giá trị  $n$  cho trước.  $T(n)$  ở dạng đầy đủ sẽ chứa  $2^n$  dòng.  $T(1)$  chứa hai dòng là ‘Q’ và ‘M’. Giả sử ta đã thiết kế xong  $T(n-1)$ , khi đó  $T(n)$  sẽ được thiết kế theo 3 bước sau.

Bước 1. Lật: Lật  $T(n-1)$  xuống phía dưới, tức là lấy đối xứng qua đường kẻ ngang cuối tấm thiệp. Ta kí hiệu phần đối xứng của  $T(n-1)$  là  $T^*(n-1)$ .

Bước 2. Thêm Q: Viết thêm kí tự ‘Q’ vào mọi dãy của  $T(n-1)$ .

Bước 3. Thêm M: Viết thêm kí tự ‘M’ vào mọi dãy của  $T^*(n-1)$ .

Ta có thể viết thêm kí tự vào đầu hoặc cuối dãy. Trong bài này ta chọn đầu dãy.

Dễ dàng chứng minh rằng thuật toán trên sinh ra các tấm thiệp thỏa các yêu cầu của đầu bài. Thực vậy, ta gọi P là tính chất “Hai dòng kề nhau khác nhau tại đúng một vị trí”. Khi đó  $T(1)$  thỏa P là hiển nhiên vì nó chỉ chứa 2 dòng ‘Q’ và ‘M’. Giả sử  $T(n-1)$  thỏa P. Khi đó đương nhiên  $T^*(n-1)$  cũng thỏa P. Do phép đối xứng, dòng cuối cùng của  $T(n-1)$  và dòng đầu tiên của  $T^*(n-1)$  giống nhau nên khi thêm ‘Q’ cho dòng trên và ‘M’ cho dòng dưới chúng sẽ khác nhau tại vị trí thêm đó. Tương tự, dòng đầu tiên của  $T(n-1)$  và dòng cuối cùng của  $T^*(n-1)$  giống nhau nên khi thêm ‘Q’ cho dòng đầu và ‘M’ cho dòng cuối chúng sẽ khác nhau tại vị trí thêm.

Dựa theo thuật toán trên ta viết hàm Line( $n, k$ ) sinh ra dòng thứ  $k$  trên tấm thiệp  $T(n)$ . Thí dụ,  $\text{Line}(3, 7) = \text{'MQM'}$ . Hàm sẽ lặp  $n$  lần, mỗi lần sinh 1 kí tự theo chiều ngược lại với kiến thiết trên. Ta thấy, nếu  $k > 2^{n-1}$  thì chúng tờ dòng  $k$  nằm trong  $T^*(n-1)$ , do đó kí tự đầu dòng của nó sẽ phải là ‘M’ và dòng từ  $T(n-1)$  lật xuống dòng  $k$  sẽ có chỉ số  $2^n - k + 1$ , ngược lại, nếu  $k \leq 2^{n-1}$  thì dòng  $k$  nằm trong  $T(n-1)$ , do đó kí tự đầu dòng của nó là ‘Q’.

(\* Pascal \*)

$n = 1$	$n = 2$	$n = 3$
Q	Q Q	QQ QQ
M	M QM	QM QQM
	M MM	MM QMM
	Q MQ	MQ QMQ
		MQ MMQ
		MM MMM
		QM MQM
		QQ MQQ

Thiết kế các thiệp  $T(1)$ ,  $T(2)$  và  $T(3)$

```

function Line(n: integer; k: longint): string;
var s: string; m: longint; i: integer;
begin
  m := 1; m := m shl n; { m = 2^n }
  for i := n downto 1 do
    begin
      m := m shr 1; { m div 2 }
      if (k <= m) then s := s + 'Q'
      else
        begin
          s := s + 'M'; k := 2*m - k + 1;
        end;
    end;
  Line := s;
end;

// C#
static public string Line(int n, int k) {
  string s = "";
  int m = 1 << n; // m = 2^n
  for (int i = n-1; i >= 0; --i) {
    m >>= 1;
    if (k <= m) s += 'Q';
    else { s += 'M'; k = 2*m - k + 1; }
  }
  return s;
}

```

Độ phức tạp. n.

**Chú thích.** Kiểu int trong C# tương đương với kiểu longint trong Pascal.

## Mã Gray

Mã Gray của một số tự nhiên k là số (k shr 1) xor k = (k div 2) xor k, trong đó xor là phép toán cộng loại trừ theo bit: a xor b = 0 khi và chỉ khi a = b.

Các tính chất của mã Gray

1. Mã Gray của hai số tự nhiên khác nhau thì khác nhau:  $k_1 \neq k_2 \Rightarrow \text{Gray}(k_1) \neq \text{Gray}(k_2)$ .
2. Mã Gray của hai số tự nhiên liên tiếp khác nhau tại đúng 1 bit.
3. Gray(0) = 0; Gray(1) = 1;
4. Nếu số x có n bit thì  $\text{Gray}(2^n - 1) = 2^{n-1}$ .

k	Gray(k)	GLine(4,k)	k	Gray(k)	GLine(4,k)
0: 0000	0: 0000	QQQQ	8: 1000	12: 1100	MMQQ
1: 0001	1: 0001	QQQM	9: 1001	13: 1101	MMQM
2: 0010	3: 0011	QQMM	10: 1010	15: 1111	MMMM
3: 0011	2: 0010	QOMQ	11: 1011	14: 1110	MMMQ
4: 0100	6: 0110	QMMQ	12: 1100	10: 1010	MQMQ
5: 0101	7: 0111	QM MM	13: 1101	11: 1011	MQMM
6: 0110	5: 0101	QM QM	14: 1110	9: 1001	MQQM
7: 0111	4: 0100	QM QQ	15: 1111	8: 1000	MQQQ

Mã Gray và giá trị của hàm Gline của 16

---

số tự nhiên đầu tiên  $k = 0..15$ .

Nhờ mã Gray ta có thể viết hàm GLine(n,k) cho ra dòng k trong thiệp T(n) một cách đơn giản như sau:

Bước 1. Tính  $x = \text{Gray}(k) = (k \text{ shr } 1) \text{ xor } k = (k \text{ div } 2) \text{ xor } k$ .

Bước 2. Xét n bit thấp của x, nếu là 1 thì viết 'M' nếu là 0 thì viết 'Q'.

Frank Gray là nhà vật lý học Mỹ làm nghiên cứu viên tại hãng Bell Labs với hàng loạt phát minh có giá trị được ứng dụng trong truyền hình, cơ học, điện tử và toán học. Năm 1947 Gray đăng ký bằng phát minh về *mã nhị phân phản hồi* sau này được gọi là *mã Gray*.

(\* Pascal \*)

```
function Gray(k: longint): longint;
begin Gray := (k shr 1) xor k end;
function Gline(n: integer; k: longint): string;
  var s: string; i: integer;
  const cc: array[0..1] of char = ('Q', 'M');
begin
  k := Gray(k); s = '';
  for i := n-1 downto 0 do
    s = s + cc[(k shr i) and 1];
  GLine := s;
end;
```

// C#

```
static public int Gray(int k){ return (k >> 1) ^ k; }
static public string GLine(int n, int k) {
  string cc = "QM";
  string s = "";
  k = Gray(k);
  for (int i = n-1; i >= 0; --i)
    s += cc[(k >> i) & 1];
  return s;
}
```

## Bài 2.10 Tổng đoạn

Một dãy con gồm các phần tử liên tiếp nhau trong một dãy cho trước được gọi là *đoạn*. Cho dãy gồm N số tự nhiên. Tìm đoạn ngắn nhất có tổng các phần tử bằng giá trị K cho trước.

TDOAN.INP	TDOAN.OUT
21 17 0 2 3 2 10 1 5 5 6 12 20 30 14 8 0 11 0 6 0 0 5	16 3

Dữ liệu vào: tệp văn bản **TDOAN.INP**

Dòng thứ nhất: hai số tự nhiên N và K,  
 $1 \leq N \leq 2000$ .

Từ dòng thứ hai trở đi: các phần tử của dãy.

Dữ liệu ra: tệp văn bản **TDOAN.OUT**

Chứa một dòng duy nhất gồm hai số tự nhiên d – chỉ số đầu đoạn và L – số phần tử trong đoạn (chiều dài đoạn). Nếu vô nghiệm thì ghi 0 0.

Trong các tệp, dữ liệu trên cùng dòng cách nhau qua dấu cách.

## Thuật toán

Ta giải bằng kĩ thuật *cửa sổ trượt* như sau. Xét đoạn  $a[i..j]$  với tổng  $S = a[i] + a[i+1] + \dots + a[j]$ ,  $i \leq j$ . Đoạn này được gọi là cửa sổ. Ta cho cửa sổ này trượt dần qua phải và xét ba tình huống sau đây.

1) ( $S = K$ ): ta ghi nhận điểm đầu  $i$  và độ dài đoạn là  $j-i+1$ . Nếu độ dài này nhỏ hơn độ dài LMin thì ta cập nhật lại các giá trị iMin và Lmin (thủ tục Update). Rồi tiếp tục xét cửa sổ mới là  $a[i+1..j]$ .

2) ( $S < K$ ): Ta dịch đầu phái của cửa sổ từ  $j$  sang  $j+1$ , giữ nguyên đầu trái (thủ tục Right).

3) ( $S > K$ ): Ta co đầu trái của cửa sổ từ  $i$  thành  $i+1$  (thủ tục Left).

Ta đặt phần tử  $a[n+1] = 0$  làm lính canh.

```
*****
TONG DOAN - Doan ngan nhat
co tong K
*****)
program TDoan;
uses crt;
const
  mn = 2001;  bl = #32;
  fn = 'TDOAN.INP'; gn = 'TDOAN.OUT';
type mw1 = array[0..mn] of word;
var f,g: text;
  n,k: word;
  a: mw1;
  iMin, LMin: word;
  iLeft,iRight: word;
  sum: word;
procedure Doc;
  var i: word;
begin
  assign(f,fn); reset(f); readln(f,n, k);
  for i := 1 to n do read(f,a[i]);
  close(f);
end;
procedure Left;
begin
  sum := sum - a[iLeft]; iLeft := iLeft + 1;
  if (iLeft > iRight) then
    begin iRight := iLeft; sum := a[iLeft]; end;
end;
procedure Right;
begin
  iRight := iRight + 1;  sum := sum + a[iRight];
end;
procedure Update;
begin
  if (LMin > iRight - iLeft + 1) then
    begin iMin := iLeft; LMin := iRight - iLeft + 1; end;
  Left;
end;
procedure XuLi;
begin
  iLeft := 1; iRight := iLeft;
  LMin := n + 1; sum := a[1];
  repeat
    if (sum = k) then Update
    else if (sum < k) then Right
      else { sum > k } Left;
  until (iRight > n);
  if (LMin = n+1) then LMin := 0;
end;
```

```

procedure Ghi;
begin
  assign(g,gn); rewrite(g); writeln(g,iMin,bl,LMin); close(g);
end;
BEGIN
  Doc; XuLi; ghi;
END.

// C#
using System;
using System.IO;
using System.Collections;
namespace SangTao2 {
  class TongDoan {
    const string fn = "TDoan.inp";
    const string gn = "TDoan.out";
    static public int n; // n - so phan tu
    static public int k; // Tong can chon
    static public int sum; // tong hien co
    static public int ileft, iright; // hai dau cua so
    static public int imin, lmin;
    static public int [] a;
    static void Main(string[] args) {
      Doc(); XuLi(); Ghi(); XemKetQua();
      Console.WriteLine("\n Fini ");
      Console.ReadLine();
    }
    static public void XemKetQua() {
      Console.WriteLine("\n Input: "+fn);
      Console.WriteLine(File.ReadAllText(fn));
      Console.WriteLine("\n Output: "+gn);
      Console.WriteLine(File.ReadAllText(gn));
    }
    static public void XuLi(){
      ileft = 0; iright = ileft;
      sum = a[ileft]; lmin = n + 1;
      while (iright < n)
        if (sum == k) Update();
        else if (sum < k) Right();
        else /* s > k */ Left();
      ++imin;
    }
    static public void Update() {
      if (lmin > iright - ileft + 1)
      { imin = ileft; lmin = iright - ileft + 1; }
      Left();
    }
    static public void Left(){
      sum -= a[ileft++];
      if (ileft > iright)
        { iright = ileft; sum = a[ileft]; }
    }
    static public void Right() { sum += a[++iright]; }
    static public void Doc(){
      int[] v =
        Array.ConvertAll(((File.ReadAllText(fn))).Split(

```

```

        new char[] { ' ', '\t', '\r', '\n' },
        StringSplitOptions.RemoveEmptyEntries),
        new Converter<string, int>(int.Parse));
    int j = 0; n = v[j++]; k = v[j++];
    a = new int[n + 1]; a[n] = 0;
    for (int i = 0; i < n; ++i) a[i] = v[j++];
}
static public void Ghi() {
    if (lmin == n + 1) File.WriteAllText(gn, "0");
    else
        File.WriteAllText(gn, imin.ToString() + " " + lmin.ToString());
}
} // TongDoan
} // SangTao2

```

## Bài 2.11 Đoạn không giảm dài nhất

Dijkstra E.

Cho dãy gồm  $N$  số nguyên. Tìm đoạn không giảm có chiều dài lớn nhất.

MDOAN.INP	MDOAN.OUT
12 1 5 5 1 3 3 3 5 7 9 1 2	4 7

Dữ liệu vào: tệp văn bản MDOAN.INP

Dòng thứ nhất: số tự nhiên  $N$ ,  $1 \leq N \leq 20000$ .

Từ dòng thứ hai trở đi: các phần tử của dãy.

Dữ liệu ra: tệp văn bản MDOAN.OUT

Chứa một dòng duy nhất gồm hai số tự nhiên  $d$  – chỉ số đầu đoạn và  $L$  – số phần tử trong đoạn (chiều dài đoạn).

Trong các tệp, dữ liệu trên cùng dòng cách nhau qua dấu cách.

Thí dụ trên cho ta đoạn không giảm dài nhất bao gồm 7 phần tử bắt đầu từ phần tử thứ tư trong dãy (các phần tử được gạch dưới):

1 5 5 1 3 3 3 5 7 9 1 2

### Thuật toán

Đây là bài dễ, ta đọc dãy các phần tử từ input file và so sánh hai phần tử liên tiếp nhau là x (phần tử đọc trước tại bước i) và y (phần tử đọc sau tại bước i+1). Nếu  $y < x$  thì coi như kết thúc một đoạn không giảm, ta cập nhật để ghi nhận lại đoạn không giảm dài nhất. Các biến tổng thể trong chương trình được dùng như sau:

MaxLen – chiều dài của đoạn không giảm dài nhất hiện tìm được,

imax - chỉ số đầu tiên của đoạn không giảm dài nhất hiện tìm được,

ileft – chỉ số đầu tiên của đoạn không giảm đang xét.

Độ phức tạp:  $c\sqrt{N}$ .

(\* Pascal \*)

```

*****
MDOAN - Doan tang dai nhat
*****)
program MDolan;
uses crt;
const
  bl = #32; fn = 'MDOAN.INP'; gn = 'MDOAN.OUT';
var f,g: text;

```

```

n: integer;
a: mw1;
iLeft, imax: integer;
MaxLen: integer;
procedure Update(i: integer);
begin
  if (MaxLen < i - iLeft) then
    begin
      MaxLen := i - iLeft;
      imax := iLeft; ileft := i;
    end;
  iLeft := i;
end;
procedure XuLi;
var i, x, y: integer;
begin
  assign(f,fn); reset(f); readln(f,n);
  read(f,x);
  iLeft := 1; MaxLen := 0;
  for i := 2 to n do
    begin
      read(f,y);
      if (y < x) then Update(i);
      x := y;
    end;
  Update(n+1);
  close(f);
end;
procedure Ghi;
begin
  assign(g,gn); rewrite(g);
  writeln(g,imax,bl,MaxLen);
  close(g);
end;
BEGIN
  XuLi; ghi;
END.

```

Trong phương án C# dưới đây ta đọc toàn bộ dữ liệu vào một mảng a rồi xử lý trên mảng này.

```

// C#
using System;
using System.IO;
using System.Collections;
namespace SangTao2 {
  class DoanKhongGiam {
    const string fn = "MDoan.inp";
    const string gn = "MDoan.out";
    static public int n; // n - so phan tu
    static public int imax; // chi so dau cua doan max
    static public int ileft; // chi so dau cua doan dang xet
    static public int maxlen; // chieu dai max
    static public int [] a;
    static void Main(string[] args) {
      Doc(); XuLi(); Ghi(); XemKetQua();
      Console.WriteLine("\n Fini ");
      Console.ReadLine();
    }
  }
}

```

```

    }
    static public void XemKetQua(): tự viết
    static public void XuLi() {
        ileft = 0; maxlen = 0;
        for (int i = 1; i < n; ++i)
            if (a[i] < a[i-1]) Update(i);
        Update(n);
    }
    static public void Update(int i) {
        if (maxlen < i - ileft)
            { maxlen = i - ileft; imax = ileft; ileft = i; }
    }
    static public void Doc(): tự viết
    static public void Ghi() {
        File.WriteAllText(gn, imax.ToString() + " " +
                           maxlen.ToString()); }
    } // DoanKhongGiam
} // SangTao2

```

## Bài 2.12 Đoạn đơn điệu dài nhất

Dijkstra E.

Cho dãy gồm  $N$  số nguyên. Tìm đoạn đơn điệu (không giảm hoặc không tăng) có chiều dài lớn nhất.

DONDIEU.INP	DONDIEU.OUT
12 1 5 5 1 3 3 3 5 7 9 1 2	4 7

dấu cách.

### Thuật toán



**Edsger Wybe Dijkstra (1930-2002)**

Sinh năm 1930 tại Rotterdam, Holland. 1948-1956 học Toán và Vật lý lý thuyết tại Đại học Leyden. 1952-1962 nghiên cứu tại Trung tâm Toán học Amsterdam. 1962-1973 Giáo sư Toán tại Đại học Bách khoa Eindhoven, Holland và Đại học Texas Austin. Dijkstra là một trong những người đi tiên phong trong lĩnh vực lập trình, người khởi xướng và đặt nền móng cho nguyên lý lập trình cấu trúc.

```

<oooooooooooo><oooooooooooo>
.. Edsger Wybe Dijkstra
.. (photo ©2002 Hamilton
.. Richards)
<oooooooooooo><oooooooooooo>

```

Nhận xét:

Đoạn có 1 phần tử là đoạn đơn điệu (tăng, giảm),

Đoạn gồm một dãy liên tiếp các phần tử bằng nhau là đoạn đơn điệu (tăng, giảm).

Ta dùng hai biến đếm các phần tử tăng hoặc bằng nhau liên tiếp, dt và đếm các phần tử giảm hoặc bằng nhau liên tiếp, dg. Nếu  $a_i = a_{i-1}$  ta tăng đồng thời dt và dg 1 đơn vị. Nếu  $a_i > a_{i-1}$  ta tăng dt thêm 1 đơn vị và đặt lại dg = 1. Nếu  $a_i < a_{i-1}$  ta tăng dg thêm 1 đơn vị và chỉnh lại dt = 1. Sau mỗi bước ta cập nhật đoạn đơn điệu dài nhất tìm được. Chương trình Pascal đọc và xử lí trực tiếp file input, chương trình C# đọc toàn bộ dữ liệu vào mảng rồi xử lí trên mảng.

Độ phức tạp:  $c\tilde{N}$ .

Các biến tổng thể:

n: số lượng phần tử,

dt: đếm số phần tử trong dãy tăng,

dg: đếm số phần tử trong dãy giảm.

iMax: chỉ số đầu của đoạn đơn điệu dài nhất,

MaxLen: chiều dài (số phần tử) của đoạn đơn điệu dài nhất.

(\* Pascal \*)

```

program DonDieu;
uses crt;
const
  bl = #32;  fn = 'DONDIEU.INP';  gn = 'DONDIEU.OUT';
var f,g: text;
  n: integer;
  dt,dg: integer;
  iMax, MaxLen: integer;
function Max(a,b,c: integer): integer;
begin
  if (a < b) then a := b; { a = Max(a,b) }
  if (a > c) then Max := a
  else Max := c;
end;
procedure XuLi;
  var i,m,x,y: integer;
begin
  assign(f,fn); reset(f);
  readln(f,n); read(f,x);
  dt := 1; dg := 1;
  MaxLen := 1; iMax := 1;
  for i := 2 to n do
    begin
      read(f,y);
      if (y = x) then
        begin dt := dt + 1; dg := dg + 1; end
      else if (y > x) then
        begin dt := dt + 1; dg := 1; end
      else { y < x }
        begin dg := dg + 1; dt := 1; end;
      if (MaxLen < dt) then
        begin MaxLen := dt; iMax := i;
        end;
    end;
  writeln(gn,MaxLen,iMax);
end;

```

```

m := Max(MaxLen, dt, dg);
if (m > MaxLen) then
begin  MaxLen := m; iMax := i - MaxLen + 1; end;
x := y;
end;
close(f);
end;
procedure Ghi;
begin
assign(g,gn); rewrite(g);
writeln(g, iMax, bl, MaxLen); close(g);
end;
BEGIN
XuLi; Ghi;
END.

// C#
using System;
using System.IO;
using System.Collections;
namespace SangTao2 {
class DonDieu {
const string fn = "DonDieu.inp";
const string gn = "DonDieu.out";
static public int n; // n - so phan tu
static public int imax; // chi so dau tien cua doan max
static public int maxlen; // chieu dai max
static public int [] a;
static void Main(string[] args) {
Doc(); XuLi(); Ghi(); XemKetQua();
Console.WriteLine("\n Fini ");
Console.ReadLine();
}
static public void XemKetQua(): tự viết
static public void XuLi(){
imax = 0; maxlen = 1;
int dt = 1, dg = 1;
int m;
for (int i = 1; i < n; ++i){
if (a[i] == a[i - 1]) { ++dt; ++dg; }
else if (a[i] < a[i - 1]) { dt = 1; ++dg; }
else /* a[i] > a[i-1] */ { ++dt; dg = 1; }
m = Max(maxlen, dt, dg);
if (maxlen < m)
{ maxlen = m; imax = i - maxlen + 1; }
}
}
static public int Max(int a, int b, int c){
if (a < b) a = b; // a = Max(a,b)
return (a > c) ? a : c;
}
static public void Doc(): tự viết
static public void Ghi(): tự viết
} // DonDieu
} // SangTao2

```

## Bài 2.13 Lũy thừa 2, 3 và 5

Dijkstra E.

Với mỗi giá trị  $N$  cho trước hãy sinh  $N$  số đầu tiên theo trật tự tăng dần là tích các lũy thừa của 2, 3 và 5.

LUYTHUA.INP	LUYTHUA.OUT
12	1 2 3 4 5 6 8 9 10 12 15 16

Dữ liệu vào: tệp văn bản LUYTHUA.INP

Chứa số tự nhiên  $N$ ,  $1 \leq N \leq 1000$ .

Dữ liệu ra: tệp văn bản LUYTHUA.OUT

$N$  số tìm được, mỗi dòng một số.

### Thuật toán

Gọi  $S$  là tập các số cần tìm. Ta có

(i)  $1 \in S$

(ii) Nếu  $x \in S$  thì  $2x, 3x, 5x \in S$ .

Giả sử các phần tử trong  $S$  được sắp tăng và ta đã tìm được phần tử thứ  $i$ . Ta kí hiệu  $S(i) = \{a_1, a_2, \dots, a_i\}$ . Để tìm phần tử thứ  $i+1$  ta nhận xét

$$a_{i+1} = \text{Min} \{ 2x, 3y, 5z \mid x, y, z \in S(i), 2x > a_i, 3y > a_i, 5z > a_i \}$$

Ta sử dụng 3 biến  $i2, i3, i5$  để ghi nhận các chỉ số trong  $S$  sao cho  $a_{i2} = x, a_{i3} = y$  và  $a_{i5} = z$ . Các biến  $a[1], i2, i3$  và  $i5$  được khởi trị 1.

Khi đó hàm Next( $i$ ) sinh phần tử sát sau phần tử  $A[i]$  sẽ như sau:

```

function Next(i: integer): integer;
begin
    while (a[i2] * 2 <= a[i]) do i2 := i2 + 1;
    while (a[i3] * 3 <= a[i]) do i3 := i3 + 1;
    while (a[i5] * 5 <= a[i]) do i5 := i5 + 1;
    Next := Min(a[i2]*2, a[i3]*3, a[i5]*5);
end;

(* Pascal *)
(***** LUY THUA cua 2, 3, 5 *****)
program LuyThua;
uses crt;
const bl = #32; mn = 1001; fn = 'LUYTHUA.INP'; gn = 'LUYTHUA.OUT';
type m11 = array[0..mn] of longint;
var f,g: text;
n: integer;
a: m11;
procedure Doc: tự viết;
function Min(a,b,c: longint): tự viết;
function Next(i: integer): tự viết;
procedure Sinh;
var i: longint;
begin
    assign(g,gn); rewrite(g);

```

```

a[1] := 1; writeln(g,1);
i2 := 1; i3 := 1; i5 := 1;
for i := 2 to n do
begin
  a[i] := Next(i-1);
  writeln(g,a[i]);
end;
close(g);
end;
BEGIN
  Doc; Sinh;
END.

// C#
using System;
using System.IO;
namespace SangTao2 {
    /*-----*
     * -----*/
    class LuyThua235 {
        const string fn = "LuyThua.inp";
        const string gn = "LuyThua.out";
        static public int n; // so luong phan tu
        static public int[] a;
        static void Main(){
            Doc(); Sinh(); Ghi(); XemKetQua();
            Console.WriteLine("\n fini");
            Console.ReadLine();
        } // Main
        static public void Doc()
        { n = int.Parse(File.ReadAllText(fn).Trim()); }
        static public void Sinh(){
            a = new int[n];
            int i2 = 0, i3 = 0, i5 = 0; a[0] = 1;
            int n1 = n-1;
            for (int i = 0; i < n1; ++i){ // Next
                while (a[i2] * 2 <= a[i]) ++i2;
                while (a[i3] * 3 <= a[i]) ++i3;
                while (a[i5] * 5 <= a[i]) ++i5;
                a[i + 1] = Min(a[i2] * 2, a[i3] * 3, a[i5] * 5);
            }
        }
        static public int Min(int x, int y, int z) : tự viết
        static public void Ghi(){
            StreamWriter g = new StreamWriter(gn);
            for (int i = 0; i < n; ++i) g.WriteLine(a[i]);
            g.Close();
        }
        static void XemKetQua(): tự viết
    } // LuyThua235
} // space

```

# Chương 3

## Trò chơi

Các bài toán trò chơi khá đa dạng và thường là khó.

Chúng ta xét loại trò chơi thứ nhất với các giả thiết sau đây:

1. Trò chơi gồm hai đầu thủ là A và B, luân phiên nhau, mỗi người đi một nước. Ta luôn giả thiết đầu thủ đi trước là A.

2. Hai đầu thủ đều chơi rất giỏi, nghĩa là có khả năng tính trước mọi nước đi.

3. Đầu thủ nào đến lượt mình không thể đi được nữa thì chịu thua và ván chơi kết thúc.

4. Không có thể hòa, sau hữu hạn nước đi sẽ xác định được ai thắng, ai thua.

Giả thiết chơi giỏi nhằm tránh các trường hợp “*ăn may*”, tức là các trường hợp do đối phương hớ hênh mà đi lạc nước. Điều này tương đương với giả thiết cả hai đầu thủ đều có thể tính trước mọi nước đi (với loại trò chơi hữu hạn) hoặc cả hai đầu thủ đều biết cách đi tốt nhất. Để tiện trình bày chúng ta gọi các trò chơi loại này là *choi cờ*, mỗi thế của bàn cờ là một *tình huống* với dữ liệu cụ thể, ta thường gọi là *một cấu hình*.

Các bài toán tin liên quan đến loại trò chơi này thường là:

- Lập trình để xác định với một thế cờ cho trước thì người đi trước (đầu thủ A) sẽ thắng hay thua.
- Lập trình để máy tính chơi với người. Dĩ nhiên chương trình bạn lập ra là dành cho máy tính.
- Lập trình để hai máy tính chơi với nhau.

Với loại trò chơi này có một *heuristic* mang tính chỉ đạo sau đây:

*Trước hết cần xác định được một tính chất  $T$  thỏa các điều kiện sau đây:*

- a) Thế thua cuối cùng thỏa  $T$ ,
- b) Mọi nước đi luôn luôn biến  $T$  thành  $V = \text{not } T$ ,
- c) Tồn tại một nước đi để biến  $V$  thành  $T$ .

*Tính chất  $T$  được gọi là bất biến thua của trò chơi.*

Việc chuyển thế  $X$  thành  $\text{not } X$  thường được gọi là *lật thế*  $X$ . Các qui tắc a - c có thể phát biểu lại như sau:

<p><i>T</i> được gọi là bát biến thua nếu</p> <ul style="list-style-type: none"> <li>a') Thé thua cuối cùng thỏa <i>T</i>,</li> <li>b') Mọi nước đi từ <i>T</i> đều lật <i>T</i> thành <i>V</i>,</li> <li>c') Tồn tại một nước đi để lật <i>V</i> thành <i>T</i>.</li> </ul>	<p>Đáu thủ nào có cách đẩy đáu thủ khác vào thế thua <i>T</i> thì đáu thủ đó sẽ thắng.</p> <p>Đáu thủ nào không thể đẩy đáu thủ khác vào thế thua <i>T</i> thì đáu thủ đó sẽ thua.</p>
--	--

Nước đi ở đây được hiểu là nước đi hợp lệ tức là nước đi tuân thủ các qui định của trò chơi, thí dụ "xe liền, pháo cách" trong cờ tướng qui định rằng quân xe có thể "ăn" trực tiếp các quân của đối phương nằm trên đường đi của nó, còn quân pháo thì phải "ăn" qua một quân đệm.

Điểm khó nhất của loại toán này là xác định bát biến thua.

### Bài 3.1. Bóc sỏi A

Trên bàn có một đồng sỏi *N* viên, hai đáu thủ *A* và *B* lần lượt đi, *A* đi nước đầu tiên. Mỗi nước đi đáu thủ buộc phải bốc từ 1 đến *M* viên sỏi khỏi bàn. Đáu thủ nào đến lượt mình không đi nổi thì thua. Cả hai đáu thủ đều chơi rất giỏi. Với hai số *N* và *M* cho trước hãy cho biết *A* thắng (ghi 1) hay thua (ghi 0).

Ta thử chơi với *M* = 3 và vài dữ liệu ban đầu *N* = 1, 2, ... Để tính nước đi cho đáu thủ *A* bạn hãy kẻ một bảng gồm 2 dòng. Dòng thứ nhất là các giá trị của *N*. Dòng thứ hai được ghi 0 ứng với tình huống *A* thua và 1 cho tình cho trường hợp *A* thắng, nếu *A* là đáu thủ đi nước đầu tiên.

<b><i>M</i> = 3</b>	<b><i>N</i> =</b>	0   1   2   3   4   5   6   7   8   9   10   11   12   13   ...
<i>A</i> thắng (1) hay thua (0)?		0   1   1   1   0   1   1   1   0   1   1   1   0   1   ...
Cách đi: số viên cần bốc để chắc thắng	#	1   2   3   #   1   2   3   #   1   2   3   #   1   ...

Một vài tình huống cho bài Bóc sỏi *A*, *M* = 3; # - đáu hàng/bốc tạm 1 viên

Thí dụ, với *M* = 3 cho trước và cố định, *A* là đáu thủ đi trước, ta có  
*N* = 0 là một thế thua, vì *A* không có cách đi.

*N* = 1 là một thế thắng, vì *A* sẽ bốc 1 viên, *B* hết cách đi.

*N* = 2 là một thế thắng, vì *A* sẽ bốc 2 viên, *B* hết cách đi.

*N* = 3 là một thế thắng vì *A* sẽ bốc 3 viên, *B* hết cách đi.

*N* = 4 là một thế thua, vì dù *A* bốc 1, 2, hoặc 3 viên đều dẫn đến các thế thắng là 3, 2, 1...

Làm thế nào để xác định được bát biến của trò chơi? Phương pháp đơn giản là tư duy Nhân - Quả hay là lập luận lùi. Cụ thể là, nếu biết kết quả là *Q* ta hãy gắng tìm nguyên nhân *N* sinh ra *Q*. Ta để ý rằng,

<i>Qui tắc xác định thế thắng / thua</i>
<p>Từ một thế <i>X</i> đang xét,</p> <ul style="list-style-type: none"> <li>• nếu tìm được một nước đi dẫn đến thế thua <i>T</i> thì <i>X</i> sẽ là thế thắng <i>V</i>, và</li> <li>• nếu mọi nước đi từ <i>X</i> đều dẫn đến thế thắng <i>V</i> thì <i>X</i> sẽ là thế thua <i>T</i>.</li> </ul>

Trước hết ta sẽ tìm một thế thua nhỏ nhất của cuộc chơi hay còn gọi là thế thua kết hoặc thế thua cuối cùng, vì đáu thủ nào gặp thế này đều phải đầu hàng và ván chơi kết thúc.

Dễ thấy thế thua kết sẽ là *N* = 0: Hết sỏi, không thể thực hiện được nước đi nào.

Vậy trước đó, những nước đi nào có thể dẫn đến thế thua  $T(N = 0)$ ?

Do mỗi đầu thủ chỉ được phép bốc 1, 2 hoặc 3 viên nên các thế thắng V trước đó chỉ có thể là  $N = 1, 2, \text{ hoặc } 3$ . Ta viết

$$T(N = 0) \leftarrow V(N = 1 | 2 | 3) \leftarrow T(N = ?)$$

trong đó T là kí hiệu cho thế thua, V là kí hiệu cho thế thắng.

Ta thử xác định thế thua  $T(N = ?)$ . Để thấy với  $N = 4$  thì mọi cách bốc 1, 2 hoặc 3 viên sỏi đều dẫn đến thế thắng  $V(N = 3 | 2 | 1)$ . Ta có,

$$T(N = 0) \leftarrow V(N = 1 | 2 | 3) \leftarrow T(N = 4) \dots$$

Đến đây ta có thể dự đoán bắt biển thua sẽ là  $N = 4k$ , cho trường hợp  $M = 3$ , hoặc tổng quát hơn,  $N = k(M+1)$ ,  $k \geq 0$ .

Vậy bắt biển thua là:

T: Số viên sỏi trong đồng là bội của  $M+1$ :  $N = k(M+1)$ ,  $k \geq 0$ .

Ta sẽ chứng minh rằng nếu  $N = k(M+1)$ ,  $k = 0, 1, 2, \dots$  thì người đi trước (là A) luôn luôn thua.

Trước hết để ý rằng nếu đầu thủ A gặp số sỏi là bội của  $M+1$  thì với mọi cách đi của A số sỏi còn lại sẽ không phải là bội của  $M+1$ . Thật vậy, muốn bảo toàn tính chất là bội của  $M+1$  thì A buộc phải bốc một bội nào đó của  $M+1$ , đây là điều không được phép vì vi phạm luật chơi.

Giả sử  $N = k(M+1)$ ,  $k \geq 1$ . Gọi số sỏi A bốc là  $s$ . Ta có, do  $1 \leq s \leq M$  nên B sẽ bốc  $u = (M+1)-s$  viên sỏi và do đó số sỏi còn lại sẽ là  $N = k(M+1) - s - ((M+1) - s) k(M+1) - (M+1) = (k-1)(M+1)$ . Đây là một bội của  $(M+1)$ .

Nếu số sỏi là bội của  $M+1$  thì với mọi cách đi hợp lệ, số sỏi còn lại sẽ không còn là bội của  $M+1$ . Nếu số sỏi không phải là bội của  $M+1$  thì luôn luôn tồn tại một cách đi để chỉnh số sỏi trở thành bội của  $M+1$ .

Kết luận Bài Bốc sỏi A
Nếu số sỏi $N = k(M+1)$ , $k \geq 0$ thì đầu thủ nào đi trước sẽ thua.

Với giả thiết A là đầu thủ đi trước, ta viết hàm **Ket(N, M)** cho ra giá trị 1 nếu A thắng, ngược lại hàm cho giá trị 0 nếu A thua. Hàm có hai tham biến: N là số viên sỏi trong đồng, M là giới hạn số viên sỏi được phép bốc. Hàm đơn thuần chỉ kiểm tra xem trị N có là bội của  $M+1$  hay không.

(\* Pascal \*)

```
function Ket(N,M: integer): integer;
begin
  if (N mod (M+1) = 0) then ket := 0 else Ket := 1;
end;
```

Hàm **CachDi(N, M)** dưới đây đảm nhận chức năng hướng dẫn người chơi chọn một cách đi. Trước hết cần kiểm tra xem thế đang xét là thắng hay thua. Nếu đó là thế thua và còn sỏi thì bốc 1 viên nhầm kéo dài thời gian thua. Nếu đó là thế thắng thì bốc số sỏi dư để số sỏi còn lại sẽ là bội của  $(M+1)$ .

(\* Pascal \*)

```
function CachDi(N,M: integer);
var r: integer;
begin
  r := N mod (M+1);
  if r = 0 then { thua }
  begin
    if N = 0 then CachDi := 0 else CachDi := 1;
    exit;
  end;
```

```

CachDi := r;
end;

// C#
static int Ket(int n, int m) { return (n%(m+1) == 0) ? 0 : 1; }
static int CachDi(int n, int m) {
    int r = n % (m+1);
    if (r == 0) // thua
        return (n == 0) ? 0 : 1;
    return r;
}

```

## Bài 3.2. Bóc sỏi B

Cho đồng sỏi  $N$  viên, hai đầu thủ  $A$  và  $B$  lần lượt đi,  $A$  đi nước đầu tiên. Mỗi nước đi đầu thủ được phép bóc từ 1 đến  $M$  viên sỏi. Đầu thủ nào thực hiện nước đi cuối cùng thì thua. Cả hai đầu thủ đều chơi rất giỏi. Với hai số  $N$  và  $M$  cho trước hãy cho biết  $A$  thắng (ghi 1) hay thua (ghi 0).

Ta nhận thấy bài này chỉ khác bài Bóc sỏi A ở điều kiện thua: *ai bóc quân cuối cùng sẽ thua.*

Chắc chắn là bạn sẽ có thể xác định ngay được bất biến thua của trò chơi này là  $N = k(M+1) + 1$ ,  $k \geq 0$ . Tuy nhiên, để hình thành kỹ năng phát hiện luật chơi cho các bài toán khó hơn sau này, bạn hãy gắng thực hiện các bước tìm kiếm theo các sơ đồ sau đây:

Sơ đồ 1: Thử với vài dữ liệu ban đầu:  $M = 3$ ;  $N = 1, 2, \dots$

<b>M = 3</b>	<b>N =</b>	1	2	3	4	5	6	7	8	9	10	11	12	13	...
<i>A thắng (1) hay thua (0)?</i>	0	1	1	1	0	1	1	1	0	1	1	1	0	...	
<i>Cách đi: số viên cần bóc để chắc thắng.</i>	#	1	2	3	#	1	2	3	#	1	2	3	#	...	

Một vài tình huống cho bài Bóc sỏi B,  $M = 3$ ; # - đầu hàng/bóc tạm 1 viên

Sơ đồ 2: Tính hai hai thế thua liên tiếp theo lập luận lùi (Nhân - quả).

<b>Sơ đồ tính hai thế thua liên tiếp</b>	
<i>Bước 1</i>	Xác định thế thua nhỏ nhất: $T(N = 1)$
<i>Bước 2</i>	Xác định các thế thắng $V$ dẫn đến $T$ : Từ $V$ có một nước đi dẫn đến $T$ . $T(N=1) \leftarrow V(N = 2   3   4)$
<i>Bước 3</i>	Xác định thế thua $T$ dẫn đến $V$ : Mọi cách đi từ $T$ đều rơi vào $V$ . $T(N=1) \leftarrow V(N = 2   3   4) \leftarrow T(N=5)$
<i>Bước 4</i>	Tổng quát hóa, xây dựng và chứng minh công thức xác định bất biến thua: $N = k(M+1)+1, k \geq 0$

Sơ đồ 3: Tổng quát hóa (Chi tiết hóa Bước 4 trong Sơ đồ 2). Trong sơ đồ 3 dưới đây ta ký hiệu  $X(k)$  là số viên sỏi tại thế  $X$  xét trong bước  $k = 0, 1, 2, \dots$   $X$  có thể là thế thắng  $V$  hoặc thế thua  $T$ , chú ý rằng bước  $k$  được xét theo quá trình lập luận lùi chứ không xét theo diễn tiến của trò chơi.

Bước 4. Tống quát hóa	
Bước 4.1	Thé thua nhỏ nhất: $T(0) = 1$ .
Bước 4.2	Giả thiết thé thua tại bước k là $T(k)$ (số viên sỏi để người đi trước thua).
Bước 4.3	Xác định các thé thắng $V(k)$ dẫn đến $T(k)$ : Có một cách đi để trên bàn còn $T(k)$ viên sỏi. $T(k) \leftarrow V(k) = T(k) + d; 1 \leq d \leq M.$
Bước 4.4	Xác định thé thua $T(k+1)$ dẫn đến $V(k)$ : Mọi cách đi từ $T(k+1)$ đều rơi vào $V(k) = T(k) + d$ ; $1 \leq d \leq M: \\ T(k) \leftarrow V(k) = T(k) + d; 1 \leq d \leq M \leftarrow T(k+1) = \text{Max} \{V(k)\} + 1 = T(k) + (M+1)$
Bước 4.5	Chứng minh công thức $T(k)$ bằng qui nạp: $T(k) = k(M+1)+1$

Dự đoán công thức: Ta có, theo công thức thu được ở Bước 4.4,  $T(k+1) = T(k)+(M+1)$ ,

$$T(0) = 1;$$

$$T(1) = T(0)+(M+1) = 1+(M+1);$$

$$T(2) = T(1)+(M+1) = 1+(M+1)+(M+1) = 2(M+1)+1;$$

$$T(3) = T(2)+(M+1) = 2(M+1)+1+(M+1) = 3(M+1)+1;$$

...

$$\text{Dự đoán: } T(k) = k(M+1)+1.$$

Chứng minh bát biến thua: Nếu số sỏi trong đồng là  $T(k) = k(M+1)+1$ ,  $k \geq 0$  thì ai đi trước sẽ thua.

Cơ sở qui nạp: với  $k = 0$  ta có  $T(0) = 0.(M+1)+1 = 1$ . Đây là thé thua nhỏ nhất.

Giả sử với  $k \geq 1$  ta có thé thua là  $T(k) = k(M+1)+1$ . Ta chứng minh rằng  $T(k+1) = (k+1)(M+1)+1$  sẽ là thé thua tiếp theo và giữa hai thé thua này là các thé thắng. Thật vậy, vì  $T(k)$  là thé thua nên các thé có dạng  $V(k) = T(k)+d$ ,  $1 \leq d \leq M$  sẽ đều là thé thắng. Từ đây suy ra thé thua tiếp sau đó phải là  $T(k+1) = T(k)+M+1 = k(M+1)+1+(M+1) = (k+1)(M+1)+1$ .

#### Kết luận Bài Bố c sỏi B

Nếu số sỏi  $N = k(M+1)+1$ ,  $k \geq 0$   
thì đấu thủ nào đi trước sẽ thua.

Ta cũng có thể sử dụng hàm  $f(N)$  xác định xem với đồng sỏi có  $N$  viên thì người đi trước sẽ thắng ( $f(N) = 1$ ) hay thua ( $f(N) = 0$ ). Ta có,  $f(1) = 0$  vì với 1 viên sỏi thì ai bốc viên đó sẽ thua. Giả sử  $f(N) = 0$ . Để thấy, khi đó  $f(N+d) = 1$  với  $1 \leq d \leq M$ , vì chỉ cần bốc  $d$  viên sỏi là dẫn đến thé thua. Tiếp đến  $f(N+(M+1)) = 0$  vì với mọi cách bốc  $s$  viên sỏi,  $1 \leq s \leq M$  đối phương sẽ bốc tiếp  $u = (M+1)-s$  để số sỏi còn lại là  $N$  viên ứng với thé thua. Từ đó suy ra  $f(N) = 0$  với  $N = k(M+1)+1$ ; còn lại là  $f(N) = 1$ .

Hai hàm **Ket(N, M)** và **CachDi(N, M)** với  $N > 0$  khi đó sẽ như sau.

```
(* Pascal *)
function Ket(N,M: integer): integer; {0: thua; 1: thang}
begin
    if (N mod (M+1) = 1) then ket := 0 else Ket := 1;
end;
function CachDi(N,M: integer);
var r: integer;
begin
    r := N mod (M+1);
    if (r = 1) then { thua: boc tam 1 vien }
```

```

CachDi := 1 else
    if (r = 0) then CachDi := M
        else CachDi := r-1;
end;

// C#
static int Ket(int n, int m)// 0: thua; 1: thang
{ return (n % (m+1) == 1) ? 0 : 1; }
static int CachDi(int n, int m) {
    int r = n % (m+1);
    if (r == 1) // thua, boc tam 1 vien
        return 1;
    else return (r == 0) ? m : r-1;
}

```

### Bài 3.3. Bóc sỏi C

Cho đồng sỏi  $N$  viên, hai đấu thủ  $A$  và  $B$  lần lượt đi,  $A$  đi trước,  $B$  đi sau. Tại mỗi nước đi, đấu thủ buộc phải bóc tối thiểu 1 quân, tối đa nửa số quân trong đồng. Đầu thủ nào đến lượt mình không đi nổi thì thua. Cả hai đấu thủ đều chơi rất giỏi. Cho biết  $A$  thắng hay thua.

Chú ý:

- Nếu số quân lẻ thì bóc nửa non,
- Đồng nào còn 1 quân thì không có cách bóc ở đồng đó, vì  $1 \text{ div } 2 = 0$  trong khi yêu cầu của luật chơi là phải bóc tối thiểu 1 quân.

Sơ đồ 1: Thử với vài dữ liệu ban đầu:  $N = 1, 2, 3, \dots$

<b>N</b>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	...
<i>A thắng (1) hay thua (0)?</i>	0	1	0	1	1	1	0	1	1	1	1	1	1	1	0	1	...
<i>Cách đi: số sỏi cần bóc để chắc thắng</i>	#	1	#	1	2	3	#	1	2	3	4	5	6	7	#	1	...

Một vài tình huống cho bài Bóc sỏi C; # - đầu hàng/bóc tạm 1 viên.

Sơ đồ 2: Khảo sát hai thế thua liên tiếp theo lập luận lùi (Nhân - quả).

<b>Sơ đồ tính hai thế thua liên tiếp</b>	
Bước 1	Xác định thế thua nhỏ nhất: $T(N = 1)$
Bước 2	Xác định các thế thắng $V$ dẫn đến $T$ : Từ $V$ có một nước đi dẫn đến $T$ . $T(N = 1) \leftarrow V(N = 2)$
Bước 3	Xác định thế thua $T$ dẫn đến $V$ : Mọi cách đi từ $T$ đều rơi vào $V$ . $T(N = 1) \leftarrow V(N = 2) \leftarrow T(N = 3)$
Bước 4	Tổng quát hóa, xây dựng và chứng minh công thức xác định bất biến thua: $N = 2^k - 1, k \geq 1$

Sơ đồ 3: Tổng quát hóa (Chi tiết hóa Bước 4 trong Sơ đồ 2). Trong sơ đồ 3 dưới đây ta kí hiệu X(k) là số viên sỏi tại thế X xét trong bước  $k = 0, 1, 2, \dots$  theo lập luận lùi từ thế thua nhỏ nhất trở đi. X có thể là thế thắng V hoặc thế thua T.

<b>Bước 4. Tổng quát hóa</b>	
<i>Bước 4.1</i>	Thế thua nhỏ nhất: $T(0) = 1$ .
<i>Bước 4.2</i>	Giả thiết thế thua $T(k)$ có $N$ viên sỏi: $T(k) = N$ .
<i>Bước 4.3</i>	Xác định các thế thắng $V(k)$ dẫn đến $T(k)$ : Có một cách đi để trên bàn còn $N$ viên sỏi. $T(k) \leftarrow V(k) = N + d; 1 \leq d \leq N.$
<i>Bước 4.4</i>	Xác định thế thua $T(k+1)$ dẫn đến $V(k)$ : Mọi cách đi từ $T(k+1)$ đều rơi vào $V(k) = N + d$ ; $1 \leq d \leq N$ . $T(k) \leftarrow V(k) = N + d; 1 \leq d \leq N \leftarrow T(k+1) = \text{Max} \{ V(k) \} + 1 = N + N + 1$
<i>Bước 4.5</i>	Chứng minh công thức $T(k)$ bằng qui nạp: $T(k) = 2^k - 1$ .

Gọi  $T(k)$  là thế thua khảo sát tại bước thứ  $k$ . Ta có,

Thế thua nhỏ nhất  $T(0) = 1$ : nếu có 1 viên sỏi thì không đi nỗi: chịu thua.

Giả sử thế thua tại bước thứ  $k$  là  $T(k) = N$

Khi đó các thế thắng dẫn đến thế  $T(k)$ , theo luật chơi sẽ là  $V(k) = T(k)+d$ ,  $1 \leq d \leq T(k)$  và thế thua tiếp sau đó phải là  $T(k+1) = T(k)+T(k)+1 = 2T(k)+1$ .

Tổng hợp lại ta có

$$T(0) = 1,$$

$$T(1) = 2T(0)+1 = 2+1,$$

$$T(2) = 2T(1)+1 = 2(2+1) + 1 = 2^2+2+1,$$

$$T(3) = 2T(2)+1 = 2(2^2+2+1)+1 = 2^3+2^2+2+1.$$

...

Áp dụng công thức  $a^{k+1} - 1 = (a^k + a^{k-1} + \dots + a + 1)(a - 1)$  ta dự đoán:

$$T(k) = 2^k + 2^{k-1} + \dots + 2 + 1 = (2^{k+1} - 1)/(2 - 1) = 2^{k+1} - 1.$$

Ta dùng qui nạp toán học để chứng minh rằng  $T(k) = 2^{k+1} - 1$ .

Với  $k = 0$ , ta có  $T(0) = 2^1 - 1 = 2 - 1 = 1$ . Vậy  $T(k)$  đúng với  $k = 0$ .

Giả sử  $T(k) = 2^{k+1} - 1$ . Ta chứng minh  $T(k+1) = 2^{k+2} - 1$ .

Ta có,  $T(k+1) = 2T(k)+1 = 2(2^{k+1}-1)+1 = 2^{k+2}-2+1 = 2^{k+2}-1$ , đpcm.

### Kết luận Bài Bố c sỏi C

Nếu số sỏi  $N$  có dạng  $2^k - 1$ ,  $k \geq 1$  thì đâu thủ nào đi trước sẽ thua.

Các số dạng  $2^k - 1$  được gọi là số Mersenne mang tên nhà toán học Pháp thế kỉ thứ XVII, người đầu tiên nghiên cứu chúng.

Với giả thiết A là đấu thủ đi trước, ta viết hàm **Ket(N)** cho ra giá trị 1 nếu A thắng, ngược lại hàm cho giá trị 0 nếu A thua. Hàm chỉ đơn thuần kiểm tra xem N có phải là số Mersenne hay không. Nếu đúng như vậy thì đấu thủ A sẽ thua, ngược lại là A thắng.



Marin Mersenne (1588-1648) là con một gia đình nông dân Pháp. Lúc đầu ông học thần học và triết học, sau chuyển sang nghiên cứu toán học và âm nhạc. Ông để lại những kết quả lý thú về cơ sở toán học của âm nhạc, hình học và lý thuyết số.

Ta dễ ý rằng  $N = 2^k - 1$  tương đương với  $N+1 = 2^k$ . Vì các số nguyên trong máy tính đều được biểu diễn dưới dạng nhị phân, nên để tính giá trị  $2^k$  ta chỉ việc dịch số 1 qua trái k bit.

(\* Pascal \*)

```
function Ket(N : integer) : integer;
var m,n1: integer;
begin
  n1 := N + 1; m := 1;
  while (m < n1) do m := m shl 1;
  { m =  $2^k \geq n1 = N+1 \Rightarrow N \leq 2^k - 1$  }
  if m = n1 then Ket := 0 else Ket := 1;
end;
```

Hàm **CachDi** dưới đây sẽ xác định số sỏi cần bóc cho mỗi tình huống. Trước hết hàm kiểm tra xem số sỏi trong đồng có phải là số Mersenne hay không qua hệ thức  $N+1 = 2^k$ ? Nếu  $N = 2^k - 1$  thì người nào đi sẽ thua, do đó ta chọn cách đi chậm nhất bằng việc bóc 1 viên sỏi. Dĩ nhiên nếu  $N = 1$  thì ta phải chịu thua bằng cách gán **CachDi** = 0. Ta xét trường hợp N không phải là số Mersenne. Khi đó tồn tại một số nguyên k thỏa  $2^k - 1 < N < 2^{k+1} - 1$ . Ta cần bóc bớt số sỏi chênh lệch là  $s = N - (2^k - 1)$  để số sỏi còn lại có dạng  $2^k - 1$ . Ta chứng minh  $1 \leq s \leq N/2$ , tức là cách đi này là hợp lệ theo quy định của bài. Thật vậy, do  $2^k - 1 < N$  nên  $s = N - (2^k - 1) \geq 1$ . Mặt khác, nếu  $s > N/2$  tức là  $N - (2^k - 1) > N/2$  thì  $N/2 > 2^k - 1$  hay  $N > 2^{k+1} - 2$ . Từ đây suy ra  $N \geq 2^{k+1} - 1$  mâu thuẫn với điều kiện của k. Vậy ta phải có  $s \leq N/2$ .

(\* Pascal \*)

```
function CachDi(N : integer) : integer;
var m, n1: integer;
begin
  n1 := N + 1; m := 1;
  while (m < n1) do m := m shl 1;
  { m =  $2^k \geq n1 = N+1 \Rightarrow N \leq 2^k - 1$  }
```

```

if m = n1 then { N = 2k - 1: Thua }
begin
    if N = 1 then CachDi := 0
    else CachDi := 1;
    exit;
end;
{ m = 2k > N+1 }
m := m shr 1;
{ m = 2k-1 < N+1 < 2k = 2m ==> m-1 < N < 2m-1 }
CachDi := N-m+1;
end;

// C#
static int Ket(int n) {
    int m = 1, n1 = n + 1;
    while (m < n1) m <= 1;
    // m = 2k ≥ n1 = n+1 ==> n ≤ 2k-1
    Ket = (m == n1) ? 0 : 1;
}
static int CachDi(int n) {
    int m = 1, n1 = n + 1;
    while (m < n1) m <= 1;
    // m = 2k ≥ n1 = n+1 ==> n ≤ 2k-1
    if (m == n1) // Thua
        return (n == 1) ? 0 : 1;
    // m = 2k > n+1
    m >= 1;
    // m = 2k-1 < n+1 < 2k = 2m ==> m-1 < n < 2m-1
    return n-m+1;
}

```

### Bài 3.4. Chia đoạn

Dạng phát biểu khác của Bài Bóc sỏi C

Cho một đoạn thẳng trên trục số dài  $N$  đơn vị với các điểm chia nguyên. Hai bạn lần lượt thực hiện thao tác sau đây: Cắt đoạn thẳng tại một điểm nguyên nằm trong đoạn để thu được 2 đoạn con sau đó vát đi đoạn ngắn, trao đoạn dài cho người kia. Nếu hai đoạn bằng nhau thì vát đi một đoạn tùy ý. Bạn nào đến lượt mình không thể thực hiện được thao tác trên thì thua. Hãy cho biết bạn đi trước thắng hay thua. Giả thiết rằng hai bạn đều chơi rất giỏi.

### Bài 3.5. Bóc sỏi D

Cho 2 đồng sỏi với số viên sỏi là  $N$  và  $M$  viên. Hai người chơi A và B, A luôn đi trước. Lượt chơi: Chọn đồng tùy ý, bóc tối thiểu 1 viên và tối đa cả đồng. Đầu thủ nào đến lượt mình mà không đi nổi thì thua. Hãy cho biết A thắng hay thua. Giả thiết rằng hai đầu thủ đều chơi rất giỏi.

Thuật toán

Bài này khá dễ giải.

<i>Bất biến thua cho Bài Bóc sỏi D</i>
<i>Số sỏi của hai đồng bằng nhau.</i>

Nếu số sỏi của hai đồng khác nhau thì A là đấu thủ đi trước sẽ cân bằng hai đồng bằng cách chọn đồng lớn rồi bốc bỏ số sỏi chênh lệch để số sỏi của hai đồng trở thành bằng nhau. Khi B đi thì sẽ biến hai đồng thành khác nhau, đến lượt A lại cân bằng hai đồng...

Ta cũng dễ dàng viết được hàm kết như sau:

```
(* Pascal *)
function Ket(N,M : integer) : integer;
begin
  if N = M then Ket := 0 else Ket := 1;
end;

Thủ tục CachDi dưới đây sẽ xác định đồng và số sỏi cần bốc cho mỗi tình huống. Hàm nhận vào là N - số lượng sỏi của đồng thứ nhất và M - số lượng sỏi của đồng thứ hai và cho ra hai giá trị: D - đồng sỏi cần chọn và S - số viên sỏi cần bốc tại đồng đó. Nếu N = M, tức là gấp thê thua thì đánh bốc 1 viên tại đồng tùy chọn, một cách ngẫu nhiên. Ta qui ước D = 0 là tinh huống chịu thua, tức là khi cả hai đồng đã hết sỏi.

(* Pascal *)
procedure CachDi(N, M : integer; var D,S : integer);
{ Dong 1: N viên, Dong 2: M viên sỏi }

begin
  if N = M then { Se Thua }
  begin
    if N = 0 then D := 0 { Het soi: dau hang }
    else
      begin { Keo dai cuoc choi }
        S := 1; { boc 1 viên }
        D := random(2)+1;{tai 1 dong tuy chon}
      end;
    exit;
  end;
  { Tinh huong thang }
  if N > M then D := 1 else D := 2; { Chon dong nhanh sỏi }
  S := abs(N-M); { Boc so sỏi chenh lech }
end;

// C#
static int Ket(int n, int m) {
// Đồng 1: n viên; Đồng 2: m viên
  return (n == m) ? 0 : 1;
}
static void CachDi(int n, int m, out int s, out int d) {
  Random r = new Random();
  if (n == m) { // thua
    if (n == 0) d = 0;
    else {
      s = 1;
      d = r.Next(2) + 1;
    }
  }
  // n != m: thang
  d = (n > m) ? 1 : 2;
  s = (d == 1) ? (n - m) : (m - n);
}
```

#### Bạn thử nghĩ

Tình hình sẽ ra sao nếu ta xét lại bài này với điều kiện thu như sau: đấu thủ bốc những quân cuối cùng còn trên bàn sẽ thua?

## Bài 3.6. Bốc sỏi E

Cho 2 đống sỏi với số viên sỏi lần lượt là  $N$  và  $M$  viên. Hai người chơi  $A$  và  $B$ ,  $A$  luôn đi trước. Lượt chơi: Chọn đống tùy ý, bốc tối thiểu 1 viên và tối đa cả đống. Đầu thủ nào bốc những quân cuối cùng còn trên bàn sẽ thua. Hãy cho biết  $A$  thắng hay thua. Giá thiết rằng hai đầu thủ đều chơi rất giỏi.

Thuật toán

Dễ thấy khi một trong hai đống chỉ còn 1 viên sỏi, đống thứ hai có sỏi thì ai đi trước sẽ thắng, vì người đó chỉ việc bốc hết đống sỏi còn lại. Ta xét trường hợp  $N, M > 1$ . Với trường hợp này ta sử dụng bất biến thua  $T(N=M)$  và tìm cách cân bằng hai đống sỏi.

		M						
		0	1	2	3	4	5	...
N	①	1	0	1	1	1	1	...
	①	0	1	1	1	1	1	...
	②	1	1	0	1	1	1	...
	③	1	1	1	0	1	1	...
	④	1	1	1	1	0	1	...
	⑤	1	1	1	1	1	0	...
...								

Gọi  $A$  là đầu thủ đi trước, ta kí hiệu  $f(N,M)$  là hàm hai biến cho giá trị 1 nếu  $A$  thắng, và giá trị 0 nếu  $A$  thua,  $N$  và  $M$  là số sỏi trong hai đống. Dễ thấy  $f$  là hàm đối xứng, tức là  $f(N,M) = f(M,N)$  vì trật tự của hai đống sỏi không quan trọng. Để tính trị của  $f$  ta sử dụng ma trận hai chiều  $f$ , các dòng ứng với giá trị  $N$ , các cột ứng với giá trị  $M$ . Ma trận này đối xứng qua đường chéo chính, do đó ta luôn giả thiết là  $N \leq M$  và sẽ lân lượt điền trị theo các dòng, tại mỗi dòng  $N$  ta bắt đầu điền trị từ các cột  $M \geq N$  trở đi. Ta có nhận xét thú vị sau đây:

$$* f(1,0) = f(0,1) = f(N,N) = 0, N > 1,$$

\* Các giá trị còn lại trong bảng đều bằng 1.

Hàm **Ket** và thủ tục **CachDi** sẽ như sau.

Bất biến thua cho bài Bốc sỏi E
I. $N = M > 1$ , hoặc
2. $(0, 1), (1, 0)$

```
function Ket(N,M : integer) : integer;
begin
  if (N + M = 1) or ((N = M) and (N > 1))
    then Ket := 0 else Ket := 1;
end;
```

Với thủ tục **CachDi** cho tình huống thắng ta phải xét khá nhiều trường hợp.

Trường hợp 1. Chỉ còn một đống: ta bốc đống kia, bớt lại một viên.

Trường hợp 2. Có một đống chứa duy nhất 1 viên sỏi: ta bốc hết đống kia.

Hai trường hợp 1 và 2 có thể gộp làm 1 như sau:

Trường hợp 1&2. Nếu một đống còn không quá 1 viên sỏi thì bốc ở đống kia số sỏi  $N+M-1$ .

Trường hợp 3. Cân bằng số sỏi hai đống bằng cách bốc số sỏi chênh lệch.

Để ý rằng trong cả 3 trường hợp thắng và cả trường hợp thua ta đều chọn đồng có nhiều sỏi hơn để bốc.

```

procedure CachDi(N,M : integer; var D,S : integer);
begin
    { Chon dong nhanh sỏi }
    if N > M then D := 1 else D := 2;
    if (N + M = 1) or ((N = M) and (N > 1)) then
        begin { Se Thua }
            S := 1; { boc 1 vien }
            exit;
        end;
    { Cac tinh huong thang }
    if (N < 2) or (M < 2) then S := N+M-1
    else S := abs(N-M);
end;

// C#
static int Ket(int n, int m) {
    return (n + m == 1 || (n == m && n > 1)) ? 0 : 1;
}
static void CachDi(int n, int m, out int d, out int s) {
    d = (n > m) ? 1 : 2;
    if (n + m == 1 || (n == m && n > 1)) { // se thua
        s = 1; // boc 1 vien
        return;
    }
    s = (n < 2 || m < 2) ? (n+m-1) : Math.Abs(n - m);
}

```

### Bài 3.7. Bốc sỏi F

Cho 2 đồng sỏi với số viên sỏi lần lượt là  $N$  và  $M$  viên,  $N, M > 1$ . Hai người chơi A và B, A luôn đi trước. Lượt chơi: Chọn đồng tùy ý, bốc tối thiểu 1 viên và tối đa nửa số viên của đồng. Đầu thủ nào đến lượt mình mà không đi nổi thì thua. Hãy cho biết A thắng hay thua. Giả thiết rằng hai đầu thủ đều chơi rất giỏi.

Thuật toán

Mời xem ta thấy rằng bất biến thua cho bài này cũng là  $N = M$ . Dự đoán của bạn gần đúng vì  $N = M$  chỉ là một trường hợp đặc biệt của bất biến thua. Để thấy, nếu  $N = M = 1$  thì hết cách đi. Nếu  $N = M > 1$  và A đi trước thì B chỉ việc cân bằng lại số sỏi của hai đồng là chắc thắng. Vậy  $N = M$  là một điều kiện thua (cho người đi trước).

Để lập bảng tính trị của hàm  $f(N,M)$  ta cũng nhận xét rằng hàm này đối xứng, tức là  $f(N,M) = f(M,N)$  vì trật tự của các đồng sỏi là không quan trọng. Cũng chính vì  $f(N,M)$  là hàm đối xứng nên ta chỉ cần tính trị của  $f(N,M)$  với  $N \leq M$  rồi lấy đối xứng qua đường chéo chính của bảng trị. Với mỗi  $N$  cho trước ta lần lượt điền từng dòng  $N$  của bảng với  $M = N, N+1, N+2, \dots$ . Theo nhận xét trên ta có ngay  $f(N,N) = 0$  với mọi  $N$ . Từ thế thua này ta thấy ngay  $f(N,N+d) = 1$  với mọi  $d = 1, 2, \dots, N$  vì từ các thế này ta có thể bốc  $d$  viên từ đồng thứ hai (đồng có  $M=N+d$  viên) để dẫn đến thế thua  $f(N,N)$ . Từ đây suy ra thế thua tiếp theo sẽ là  $f(N,N+N+1) = f(N,2N+1)$ . Tương tự, thế thua tiếp sau thế này phải là  $f(N,2(2N+1)+1) = f(N,2^2N+2+1)$ . Tổng quát hóa ta thu được kết quả sau:

Nếu đồng sỏi thứ nhất có  $N$  viên thì các thế thua sẽ có dạng  $f(N,M)$  với  $M = 2^kN+2^{k-1}+\dots+2+1 = 2^kN+(2^{k-1}+\dots+2+1)$ . Áp dụng công thức

$$2^k-1 = (2-1)(2^{k-1}+2^{k-2}+\dots+2+1) = 2^{k-1}+2^{k-2}+\dots+2+1$$

ta thu được  $M = 2^kN+2^k-1$  hay  $M+1 = 2^k(N+1)$ .

Vậy

**Bát biến thua cho Bài Bóc sói F**

Số sói trong hai đồng thỏa hệ thức  
 $(N+1) = 2^k(M+1), k \geq 0$  (\*)

Từ hệ thức (\*) ta suy ra  $N = M$  là trường hợp riêng khi  $k = 0$ .

Hàm Ket và thủ tục CachDi khi đó sẽ như sau.

Để ý rằng các số nguyên trong máy tính được biểu diễn dưới dạng nhị phân nên giá trị  $2^k$  tương đương với toán tử dịch trái 1 k bit, 1 shl k.

```
function Ket(N,M : integer) : integer;
  var N1,M1,t: integer;
begin
  N1 := N+1; M1 := M+1;
  if N1 < M1 then
    begin
      t := N1; N1 := M1; M1 := t;
    end; { N1 ≥ M1 }
  while M1 < N1 do M1 := M1 shl 1; { 2*M1 }
  if (M1 = N1) then Ket := 0 else Ket := 1;
end;
```

Với thủ tục CachDi cho tinh huống thua thì A đành bóc 1 viên ở đồng nhiều sói. Trong trường hợp thắng ta cũng chọn đồng nhiều sói để bóc bớt S viên sao cho số sói giữa hai đồng thỏa hệ thức (\*). Ta tính S như sau. Giả sử  $N > M$  và k là số nguyên đầu tiên thỏa hệ thức  $N+1 < 2^k(M+1)$ . Khi đó, do điều kiện thắng nên ta phải có  $2^{k-1}(M+1) < N+1 < 2^k(M+1)$ . Vậy số sói chênh lệch cần bóc bớt ở đồng nhiều sói sẽ là  $S = N+1-2^{k-1}(M+1)$ . Ta chứng minh rằng  $1 \leq S \leq N/2$ . Thật vậy, vì  $2^{k-1}(M+1) < N+1$  nên  $S = N+1-2^{k-1}(M+1) \geq 1$ . Mặt khác, nếu  $S > N/2$  thì  $2S > N$  hay  $2N+2-2^k(M+1) > N$ . Từ đây rút ra  $N+1 > 2^k(M+1)-1$ , hay  $N+1 \geq 2^k(M+1)$ , mâu thuẫn với giả thiết về k.

```
procedure CachDi(N,M : integer; var D,S : integer);
  var N1,M1,t: integer;
begin
  { N = M = 1: dau hang }
  if (N = M) and (N = 1) then
    begin
      D := 0; S := 0;
      exit;
    end;
  { Chon dong nhieu soi }
  if N > M then D := 1 else D := 2;
  N1 := N+1; M1 := M+1;
  if N1 < M1 then
    begin
      t := N1; N1 := M1; M1 := t;
    end; { N1 ≥ M1 }
  while M1 < N1 do M1 := M1 shl 1; { 2*M1 }
  if (M1 = N1) then { Thua }
    begin { Se Thua }
      S := 1; { boc 1 vien }
      exit;
    end;
  { Cac tinh huong thang }
  M1 := M1 shr 1;
  S := N1-M1;
end;
```

```

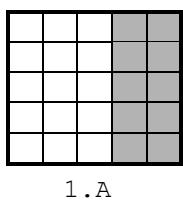
// C#
    static int Ket(int n, int m) {
        if (n < m) {
            int t = n; n = m; m = t;
        } // n >= m
        int n1 = n + 1, m1 = m + 1;
        while (m1 < n1) m1 <<= 1;
        return (m1 == n1) ? 0 : 1;
    }
    static void CachDi(int n, int m, out int d, out int s){
        // n = m = 1: dau hang
        if (n == m && n == 1){
            s = d = 0;
            return;
        }
        // Chon dong nhieu soi
        d = (n >= m) ? 1 : 2;
        int n1 = n + 1, m1 = m + 1;
        if (n1 < m1) {
            int t = n1; n1 = m1; m1 = t;
        } // n1 >= m1
        while (m1 < n1) m1 <<= 1;
        if (m1 == n1) // thua {
            s = 1; // boc 1 vien tai dong d
            return;
        }
        // Cac tinh huong thang
        m1 >>= 1; s = n1 - m1;
    }
}

```

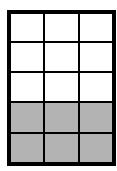
### Bài 3.8. Chia Hình chữ nhật

*Olimpic Quốc tế*

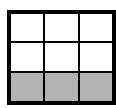
Cho một lưới chữ nhật kích thước  $N \times M$  đơn vị nguyên. Hai bạn lần lượt thực hiện thao tác sau đây: Cắt hình theo một đường kẻ trong lưới đi qua một điểm nguyên trên một cạnh và không trùng với đỉnh để thu được 2 hình chữ nhật sau đó vát đi hình có diện tích nhỏ hơn, trao hình có diện tích lớn hơn cho người kia. Nếu hai hình có diện tích bằng nhau thì vát đi một hình tùy ý. Bạn nào đến lượt mình không thể thực hiện được thao tác trên thì thua. Hãy cho biết bạn đi trước thắng hay thua. Giả thiết rằng hai bạn đều chơi rất giỏi.



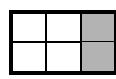
1.A



2.B



3.A



4.B



5.A



6.

R

Với hình chữ nhật  $5 \times 5$  đấu thủ A sẽ thua sau 6 nước đi.

Sau mỗi lần cắt, mảnh trắng có diện tích lớn hơn

sẽ được giao cho đấu thủ tiếp theo,

mảnh xám sẽ được bỏ đi.

Gợi ý. Bài này hoàn toàn tương đương như Bài Bốc sỏi F.

## Bài 3.9. Bốc sỏi G

(Dạng tổng quát).

Cho  $N$  đồng sỏi với số viên sỏi lần lượt là  $S_i$ ,  $i = 1, 2, \dots, N$ . Hai người chơi A và B, A luôn đi trước. Lượt chơi: Chọn đồng tùy ý, bốc tối thiểu 1 viên và tối đa nửa số viên của đồng. Đầu thủ nào đến lượt mình mà không đi nổi thì thua. Hãy cho biết A thắng hay thua. Giả thiết rằng hai đầu thủ đều chơi rất giỏi.

## Bài 3.10. Chia Hình hộp

(Cho trường hợp 3 đồng sỏi)

Cho một lưới hình hộp chữ nhật kích thước  $N \times M \times H$  đơn vị nguyên. Hai bạn lần lượt thực hiện thao tác sau đây: Cắt hình theo một thiết diện đi qua một điểm nguyên trên một cạnh, không trùng với đỉnh và vuông góc với cạnh đó để thu được 2 hình hộp chữ nhật sau đó vát đi hình có thể tích nhỏ hơn, trao hình có thể tích lớn hơn cho người kia. Nếu hai hình có cùng thể tích thì vát đi một hình tùy ý. Bạn nào đến lượt mình không thể thực hiện được thao tác trên thì thua. Hãy cho biết bạn đi trước thắng hay thua. Giả thiết rằng hai bạn đều chơi rất giỏi.

Để giải bài 3 đồng sỏi chúng ta cần một chút trợ giúp của toán học. Bạn xem ba mệnh đề dưới đây và giải thử một số thí dụ nhỏ, sau đó thử bắt tay chứng minh các mệnh đề đó. Bạn cũng có thể xem lại các chứng minh đã trình bày trong các bài giải nói trên.

Cơ sở toán học

Định nghĩa 1. Các số tự nhiên dạng  $2^k - 1$ ,  $k = 0, 1, 2, \dots$  được gọi là số Mersenne.

Thí dụ, các số 0, 1, 3, 7, 15, 31, 63, 127, 255, 511, 1023 là những số Mersenne ứng với các giá trị  $k = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$  và 10.

Các số nằm giữa các số trên thí dụ, 2, 4, 5, 6, 8, 9, 10, 11, ... không phải là số Mersenne.

**Mệnh đề 1.** Cho số tự nhiên  $n$ . Nếu  $n$  không phải là số Mersenne thì ta luôn luôn tìm được số tự nhiên  $S$ ,  $1 \leq S \leq n/2$  để  $n-S$  là một số Mersenne.

Thí dụ,

1.  $n = 2$ ,  $S = ?$
2.  $n = 10$ ,  $S = ?$
3.  $n = 1534$ ,  $S = ?$

Gợi ý. Xác định  $k$  để  $2^k < n+1 < 2^{k+1}$ . Sau đó tính  $S = n+1 - 2^k$ .

Đáp án: 1.  $S = 1$ ; 2.  $S = 3$ ; 3.  $S = 511$ .

Cho 2 số tự nhiên  $n$  và  $m$ . Xét hệ thức

$$n + 1 = 2^k(m + 1), k = 0, 1, 2, \dots \quad (*)$$

**Mệnh đề 2**

- a) Hai số Mersenne bất kì đều thỏa hệ thức (\*).
- b) Có những cặp số tự nhiên thỏa hệ thức (\*) nhưng không phải là số Mersenne.
- c) Nếu cặp số tự nhiên  $n$  và  $m$  thỏa hệ thức (\*) và một trong hai số đó là số Mersenne thì số kia cũng phải là số Mersenne.

Gợi ý

- a)  $n = 2^a - 1$ ,  $m = 2^b - 1$ ,  $a \geq b \Rightarrow n+1 = (m+1) \cdot 2^{a-b}$ .
- b) Thí dụ, 5 và 11:  $(11+1) = (5+1) \cdot 2^1$ . Với mọi  $a, b$  nguyên:  $5 \neq 2^a - 1$ ,  $11 \neq 2^b - 1$ .
- c)  $n+1 = (m+1) \cdot 2^k$ ,  $n = 2^a - 1 \Rightarrow 2^a = (m+1) \cdot 2^k$  hay  $m = 2^{a-k} - 1$ .

Mệnh đề 3. Cho 2 số tự nhiên  $n$  và  $m$ ,  $n > m$ . Nếu  $n$  và  $m$  không thỏa hệ thức (\*) thì ta luôn luôn tìm được số  $S$ ,  $1 \leq S \leq n/2$  để  $n-S$  và  $m$  thỏa hệ thức (\*).

Thí dụ,

1.  $n = 12$ ,  $m = 3$ ,  $S = ?$

$$2. n = 50, m = 5, S = ?$$

$$3. n = 54, m = 6, S = ?$$

Đáp án: 1.  $S = 5$ ; 2.  $S = 3$ ; 3.  $S = 27$ .

Gọi ý. Xác định k max để  $2^k(m+1) < n+1$ . Sau đó tính  $S = n+1 - 2^k(m+1)$ .

Tiếp theo sẽ là bài toán bóc nhiều đồng sỏi với luật bóc số quân không hạn chế trong một đồng duy nhất đã chọn.

### Bài 3.11. Trò chơi NIM

*Trò chơi NIM có xuất xứ từ Trung Hoa, dành cho hai đấu thủ A và B với các nước đi lần lượt đan nhau trên một đấu trường với N đồng sỏi. Người nào đến lượt đi thì được chọn tùy ý một đồng sỏi và bóc tối thiểu là 1 viên, tối đa là cả đồng đã chọn. Ai đến lượt mình không thể thực hiện được nước đi sẽ thua. Ta giả thiết là A luôn đi trước và hai đấu thủ đều chơi rất giỏi. Cho biết A thắng hay thua?*

Thuật toán

Gọi số viên sỏi trong các đồng là  $S_1, S_2, \dots, S_N$ .

Kí hiệu  $\oplus$  là tổng loại trừ (xor). Đặt  $x = S_1 \oplus S_2 \oplus \dots \oplus S_N$ . Ta chứng minh rằng *bất biến thua của trò chơi NIM là  $x = 0$ , tức là nếu  $x = 0$  thì đến lượt ai đi người đó sẽ thua.*

Trước hết nhắc lại một số tính chất của phép toán  $\oplus$  theo bit.

1)  $a \oplus b = 1$  khi và chỉ khi  $a \neq b$ .

2)  $a \oplus 0 = a$

3)  $a \oplus 1 = \text{not } a$

4) Tính giao hoán:  $a \oplus b = b \oplus a$

5) Tính kết hợp:  $(a \oplus b) \oplus c = a \oplus (b \oplus c)$

6) Tính lũy linh:  $a \oplus a = 0$

7)  $a \oplus b \oplus a = b$

8) Tính chất 7 có thể mở rộng như sau: Trong một biểu thức chỉ chứa phép xor ta có thể xóa đi chẵn lần các phần tử giống nhau, kết quả sẽ không thay đổi.

Để dễ nhớ ta gọi phép toán này là *so khác – so* xem hai đối tượng có khác nhau hay không. Nếu khác nhau là đúng (1) ngược lại là sai (0).

Bất biến  $x = 0$  có ý nghĩa như sau: Nếu viết các giá trị  $S_i, i = 1..N$  dưới dạng nhị phân vào một bảng thì số lượng số 1 trong mọi cột đều là số chẵn.

	Dạng nhị phân			
<b>S1 = 13</b>	1	1	0	1
<b>S2 = 14</b>	1	1	1	0
<b>S3 = 6</b>	0	1	1	0
<b>S4 = 7</b>	0	1	1	1
<b>S5 = 2</b>	0	0	1	0
<b><math>\oplus x = 0</math></b>	0	0	0	0

Bảng bên cho ta  $S_1 \oplus S_2 \oplus S_3 \oplus S_4 \oplus S_5 = 13 \oplus 14 \oplus 6 \oplus 7 \oplus 2 = 0$ .

Nếu  $x$  là tổng xor của các  $S_i, i = 1..N$ , với mỗi  $i = 1..N$  ta kí hiệu  $K(i)$  là *tổng xor khuyết i* của các  $S_i$  với cách tính như sau:  $K(i) = S_1 \oplus S_2 \oplus \dots \oplus S_{i-1} \oplus S_{i+1} \oplus \dots \oplus S_N$ . Như vậy  $K(i)$  là tổng xor của các  $S_j$  sau khi đã loại trừ phần tử  $S_i$  và  $x$  chính là tổng xor đủ của các  $S_i, i = 1..N$ . Do  $S_i \oplus S_i = 0$  và  $0 \oplus y = y$  với mọi  $y$  nên  $K(i) = x \oplus S_i$ . Để cho tiện, ta cũng kí hiệu  $K(0)$  chính là tổng xor đủ của các  $S_i, i = 1..N$ . Với thí dụ đã cho ta tính được các tổng khuyết như sau:

$$K(0) = S_1 \oplus S_2 \oplus S_3 \oplus S_4 \oplus S_5 = 13 \oplus 14 \oplus 6 \oplus 7 \oplus 2 = 0.$$

$$K(1) = S_2 \oplus S_3 \oplus S_4 \oplus S_5 = 14 \oplus 6 \oplus 7 \oplus 2 = 13,$$

$$K(2) = S_1 \oplus S_3 \oplus S_4 \oplus S_5 = 13 \oplus 6 \oplus 7 \oplus 2 = 14,$$

$$K(3) = S_1 \oplus S_2 \oplus S_4 \oplus S_5 = 13 \oplus 14 \oplus 7 \oplus 2 = 6,$$

$$K(4) = S_1 \oplus S_2 \oplus S_3 \oplus S_5 = 13 \oplus 14 \oplus 6 \oplus 2 = 7,$$

$$K(5) = S_1 \oplus S_2 \oplus S_3 \oplus S_4 = 13 \oplus 14 \oplus 6 \oplus 7 = 2.$$

Ta phát hiện được qui luật lí thú sau đây:

**Mệnh đề 1.** Cho  $x$  là tổng xor của  $N$  số tự nhiên,  $S_i$ ,  $x = S_1 \oplus S_2 \oplus \dots \oplus S_N$ . Khi đó  $K(i) = x \oplus S_i$ ,  $i = 1, 2, \dots, N$ . Tức là muốn bỏ một số hạng trong tổng  $\oplus$  ta chỉ việc  $\oplus$  thêm tổng với chính số hạng đó. Nói riêng, khi  $x = 0$  ta có  $K(i) = S_i$ ,  $i = 1, 2, \dots, N$ .

*Chứng minh*

Gọi  $x$  là tổng xor đủ của các số đã cho,  $x = S_1 \oplus S_2 \oplus \dots \oplus S_N$ . Vận dụng ính giao hoán và tính lũy đồng ta có thể viết  $x \oplus S_i = (S_1 \oplus S_2 \oplus \dots \oplus S_{i-1} \oplus S_{i+1} \oplus S_N) \oplus (S_i \oplus S_i) = K(i) \oplus 0 = K(i)$ ,  $i = 1, 2, \dots, N$ , đpcm.

Ta chứng minh tiếp các mệnh đề sau:

**Mệnh đề 2.** Nếu  $x \neq 0$  thì có cách đi hợp lệ để biến đổi  $x = 0$ .

*Chứng minh*

Do  $x \neq 0$  nên ta xét chữ số 1 trái nhất trong dạng biểu diễn nhị phân của  $x = (x_m, x_{m-1}, \dots, x_0)$ ,  $x_j = 1$ ,  $x_i = 0$ ,  $i > j$ . Do  $x$  là tổng xor của các  $S_i$ ,  $i = 1..N$ , nên tồn tại một  $S_i = (a_m, a_{m-1}, \dots, a_0)$  để chữ số  $a_j = 1$ . Ta chọn đồng  $S_i$  này (dòng có dấu \*). Khi đó, ta tính được  $K(i) = x \oplus S_i = (x_m \oplus a_m, x_{m-1} \oplus a_{m-1}, \dots, x_0 \oplus a_0) = (b_m, b_{m-1}, \dots, b_0)$  với  $b_i = x_i \oplus a_i$ ,  $0 \leq i \leq m$ . Ta có nhận xét sau đây:

\* Tại các cột  $i > j$ :  $b_i = a_i$ , vì  $b_i = x_i \oplus a_i = 0 \oplus a_i = a_i$ ,

\* Tại cột  $j$  ta có:  $b_j = 0$ , vì  $b_j = x_j \oplus a_j = 1 \oplus 1 = 0$ .

Do  $a_j = 1$ ,  $b_j = 0$  và mọi vị trí  $i > j$  đều có  $b_i = a_i$  nên  $S_i > K(i)$ . Nếu ta thay dòng  $S_i$  bằng dòng  $K(i)$  thì tổng xor y khi đó sẽ là

$$y = (x \oplus S_i) \oplus K(i) = K(i) \oplus K(i) = 0.$$

Vậy, nếu ta bốc tại đồng  $i$  số viên sỏi  $v = S_i - K(i)$  thì số sỏi còn lại trong đồng này sẽ là  $K(i)$  và khi đó tổng xor sẽ bằng 0, đpcm.

**Mệnh đề 3.** Nếu  $x = 0$  và còn đồng sỏi khác 0 thì mọi cách đi hợp lệ đều dẫn đến  $x \neq 0$ .

*Chứng minh*

Cách đi hợp lệ là cách đi làm giảm thực sự số sỏi của một đồng  $S_i$  duy nhất nào đó,  $1 \leq i \leq N$ . Giả sử đồng được chọn là  $S_i = (a_m, a_{m-1}, \dots, a_0)$ . Do  $S_i$  bị sửa nên chắc chắn có một bit nào đó bị đảo (từ 0 thành 1 hoặc từ 1 thành 0). Ta gọi bit bị sửa đó là  $a_j$ . Khi đó tổng số bit 1 trên cột  $j$  sẽ bị tăng hoặc giảm 1 đơn vị và do đó sẽ không còn là số chẵn. Từ đó suy ra rằng bit  $j$  trong  $x$  sẽ là 1, tức là  $x \neq 0$  đpcm.

Phản lập luận chủ yếu trong mệnh đề 2 nhằm mục đích chỉ ra sự tồn tại của một tập  $S_i$  thỏa tính chất  $S_i > x \oplus S_i$ . Nếu tìm được tập  $S_i$  như vậy ta sẽ bốc  $S_i - (x \oplus S_i)$  viên tại đồng sỏi  $i$ .

Giả thiết rằng mảng  $S[1..N]$  kiểu nguyên chứa số lượng sỏi của mỗi đồng đã khởi tạo như một đối tượng dùng chung, ta viết hàm **Ket** và thủ tục **CachDi** như sau.

Hàm **Ket** sẽ cho ra giá trị là tổng xor  $x$  của các đồng sỏi. Như vậy, khi  $x = 0$  thì người nào đi sẽ thua, ngược lại, khi  $x \neq 0$  thì người nào đi sẽ thắng.

```
function Ket(N: integer): integer;
  var x, i: integer;
begin
  x := 0;
  for i := 1 to N do x := x xor S[i];
  Ket := x;
end;
```

Thủ tục **CachDi** hoạt động như sau:

Gọi hàm **x = Ket**. Nếu  $x = 0$  tức là sẽ thua thì chọn một đồng còn sỏi, thí dụ đồng còn nhiều sỏi nhất, để bốc tạm 1 viên nhằm kéo dài cuộc chơi. Nếu  $x = 0$  và các đồng đều hết sỏi thì đương nhiên là phải chịu thua. Trường hợp  $x \neq 0$  thì ta tìm cách đi chắc thắng như sau:

Bước 1. Tìm đồng sỏi  $i$  thỏa điều kiện  $x \oplus S_i < S_i$ .

Bước 2. Bốc tại đồng  $i$  đó  $S_i - (x \oplus S_i)$  viên.

	Dạng nhị phân			
	<b>x3</b>	<b>x2</b>	<b>x1</b>	<b>x0</b>
<b>S1 = 12</b>	1	1	0	0
<b>S2 = 14</b>	1	1	1	0
<b>* S3 = 6</b>	0	1	1	0
<b>S4 = 3</b>	0	0	1	1
<b>S5 = 2</b>	0	0	1	0
<b>⊕ x = 5</b>	0	1	0	1

```

procedure CachDi(N: integer; var D,V: integer);
  var x,i: integer;
begin
  x := Ket(N);
  if x = 0 then { Thua }
  begin
    D := 1;
    for i := 2 to N do
      if (S[i] > S[D]) then D := i;
    if (S[D] = 0) {Het soi: Dau hang}
      then D := 0
    else S := 1;
    exit;
  end;
  { Chac thang }
  for D:=1 to N do
    if (s[D] > (x xor S[D])) then
      begin
        V := S[D]-(x xor S[D]); {boc V vien tai dong D }
        exit;
      end;
  end;

```

Trong các hàm C# dưới đây mảng s được khai báo n phần tử, các phần tử được mã số từ 0 đến n-1, trong khi các đồng sói được mã số từ 1 đến n do đó chúng ta phải lưu ý chuyên đổi chỉ số cho thích hợp

```

// C#
static int Ket() {
  int x = 0;
  for (int i = 0; i < s.Length; ++i) x ^= s[i];
  return x;
}
static int CachDi(ref int d, ref int v) {
  int x = Ket();
  if (x == 0) { // Thua
    d = 0;
    for (int i = 1; i < s.Length; ++i)
      if (s[i] > s[d]) d = i;
    // s[d] = Max(s[i] | i = 0..n-1)
    if (s[d] > 0){ v = 1; ++d; }
    return x;
  }
  // Thang
  for (d = 0; d < s.Length; ++d)
    if ((x ^ s[d]) < s[d]){
      v = s[d] - (x ^ s[d]);
      ++d;
      return x;
    }
  return x;
}

```

### Bài 3.12. Cờ bảng

Bàn cờ là một tấm bảng chữ nhật  $N$  dòng mã số từ trên xuống lần lượt là  $1, 2, \dots, N$  và  $M$  cột mã số từ trái sang lần lượt là  $1, 2, \dots, M$ ;  $2 \leq N \leq 500$ ,  $2 \leq M \leq 50$ . Một quân cờ @ được đặt tại dòng  $x$ , cột  $y$ . Hai đầu thủ  $A$  và  $B$  luân phiên mỗi người đi một nước như sau: buộc phải chuyển quân cờ @ từ cột hiện đứng là  $y$

sang cột k tùy chọn nhưng phải khác cột y. Việc chuyển này phải được thực hiện nghiêm ngặt như sau: trước hết đẩy ngược quân cờ @ lên k dòng. Nếu quân cờ vẫn còn trong bảng thì rẽ phải hoặc trái để đặt quân cờ vào cột k, ngược lại nếu quân cờ rời ra ngoài bảng thì coi như thua.

Như vậy, nếu quân cờ đang đặt tại vị trí (x,y) muốn chuyển quân cờ sang cột  $k \neq y$  thì trước hết phải đẩy quân cờ đến dòng  $x-k$ . Nếu  $x-k \geq 1$  thì được phép đặt quân cờ vào vị trí mới là  $(x-k, k)$ .

Giả thiết A luôn luôn là đấu thủ đi nước đầu tiên và hai đấu thủ đều chơi rất giỏi. Biết các giá trị  $N, M, x$  và  $y$ . Hãy cho biết A thắng (ghi 1) hay thua (ghi 0)?

	❶	❷	❸	❹
❶		A <sub>3</sub>		
❷				
❸	B <sub>2</sub>			
❹			A <sub>1</sub>	
❺				
❻				
❼	@			
❽				

Với  $N = 8$ ,  $M = 4$ , quân cờ @ đặt tại vị trí xuất phát (7,2) và A đi trước ta thấy A sẽ thắng sau 3 nước đi đan xen tính cho cả hai đấu thủ như sau:

1. A chuyển @ từ vị trí @ (7,2) sang vị trí A<sub>1</sub>(4,3)
2. B chuyển @ từ vị trí A<sub>1</sub>(4,3) sang vị trí B<sub>2</sub>(3,1)
3. A chuyển @ từ vị trí B<sub>2</sub>(3,1) sang vị trí A<sub>3</sub>(1,2)

B chịu thua vì hết cách đi!

Thuật toán

Ta thử vận dụng kĩ thuật Nhân - Quả để điền trị 1/0 vào mỗi ô  $(i, j)$  của bảng a với ý nghĩa như sau: nếu gặp thẻ cờ có quân cờ @ đặt tại vị trí  $(i, j)$  thì ai đi trước sẽ thắng (1) hay thua (0). Ta sẽ duyệt theo từng dòng từ 1 đến N, trên mỗi dòng ta duyệt từ cột 1 đến cột M.

Ta có nhận xét quan trọng sau đây. Nếu ô  $(i, j) = 0$  tức là gặp thẻ thua thì các ô đi đến được ô này sẽ là những thẻ thắng. Đó chính là các ô  $(i+j, k)$  với  $1 \leq k \leq M$  và  $k \neq j$ .

Từ nhận xét này ta viết ngay được hàm **Ket** - kiểm tra xem người đi trước với quân cờ @ tại ô  $(x, y)$  trên bàn cờ  $N \times M$  sẽ thắng (1) hay thua (0).

Trước hết ta lấp đầy trị 0 cho bảng - mang hai chiều a[1..N, 1..M] kiểu byte, sau đó lần lượt duyệt các phần tử của bảng và điền trị 1 theo nhận xét trên.

```
function Ket(x,y: integer): integer;
var i,j,k: integer;
begin
  fillchar(a,sizeof(a),0);
  for i := 1 to x-1 do
    for j := 1 to Min(x-i,M) do
      if (a[i,j] = 0) then
        {Điền 0 cho các ô (i+j, k); k=1..M, k ≠ j }
        for k := 1 to M do
          if (k <> j) then a[i+j,k] := 1;
  Ket := a[x,y];
end;
```

Thuật toán trên đòi hỏi độ phức tạp  $O(NM^2)$ .

#### Nhận xét

Nếu  $[i, j] = 0$  thì mọi ô trên dòng  $i+j$ , trừ ô  $(i+j, j)$  đều nhận trị 1.

$[i, j] = 0 \Rightarrow [i+j, k] = 1, k = 1..M, k \neq j$ .

	❶	❷	❸	❹
❶	0	0	0	0
❷	0	0	0	0
❸	0	0	0	0
❹	0	0	0	0
❺	0	0	0	0
❼	0	0	0	0
❽	0	0	0	0

	❶	❷	❸	❹
❶	0	0	0	0
❷	0	1	1	1
❸	1	1	1	1
❹	1	1	0	1
❺	1	1	1	0
❼	0	0	0	0
❽	0	0	0	0

∅	0	0	0	0
∅	0	0	0	0

1	1	1	1
0	0	0	0

Lắp đầy 0 (bảng trái) rồi điền trị (bảng phải)

cho cờ bảng  $N = 8, M = 4$ .

Kết quả với  $x = 7, y = 2: a[7,2] = 1$  (A thắng).

Để tham gia cuộc chơi với các giá trị  $N, M$  và  $(x,y)$  cho trước dĩ nhiên bạn cần tính trước bảng  $a$  theo thủ tục **Ket** nói trên. Sau đó, mỗi lần cần đi bạn chọn nước đi theo hàm **CachDi** như mô tả dưới đây. Hàm nhận vào các giá trị  $N, M$  là kích thước dòng và cột của bảng; dòng  $sx$ , cột  $sy$  là vị trí đang xét của quân cờ @ và cho ra một trong ba giá trị loại trừ nhau như sau:

**CachDi** = 1 nếu tìm được một vị trí chắc chắn  $(nx,ny)$  để đặt quân cờ;

**CachDi** = 0 nếu không thể tìm được vị trí chắc chắn nào nhưng còn nước đi do đó buộc phải đi đến vị trí  $(nx,ny)$ ;

**CachDi** = -1 nếu hết cách đi, tức là chấp nhận thua để kết thúc ván cờ.

Ta thấy sau khi gọi thủ tục **Ket** thì dòng đầu tiên của bảng  $a$  chứa toàn 0 và phần tử  $a[2,1]$  cũng nhận trị 0. Đó là những thế buộc phải đầu hàng vì đã hết cách đi. Vậy tình huống đầu hàng (hay hết cách đi) sẽ là

$sx = 1$ , hoặc

$sx = 2$  và  $sy = 1$ .

Ngoài ra, do thủ tục **Ket** đã được gọi, tức là bảng  $a$  đã được điền thể hiện mọi cách đi của cuộc chơi nên  $a[sx,sy]$  cho ta ngay giá trị chắc chắn hoặc chắc thua của tình huống xuất phát từ ô  $(sx,sy)$ .

Nếu  $a[sx,sy] = 0$  ta đành chọn nước đi có thể thua chậm theo Heuristic sau đây: *Tìm cách đi gần quân cờ @ từ vị trí  $(sx,sy)$  lên càng ít ô càng tốt.*

Nếu  $a[sx,sy] = 1$  ta chọn nước đi có thể thua nhanh theo Heuristic sau đây: *Tìm cách đi gần quân cờ @ từ vị trí  $(sx,sy)$  lên cao nhất có thể được*, tức là lên vị trí  $(nx, ny)$  thỏa đồng thời các điều kiện

$a[nx, ny] = 0$ ,

$nx$  càng nhỏ càng tốt.

Sau này ta sẽ thấy các Heuristics trên chỉ là một *cách đi tốt* chứ chưa phải là *cách đi tối ưu*.

```
function CachDi(M, sx, sy: integer; var nx,ny: integer): integer;
begin
  if (sx = 1) or ((sx = 2) and sy = 1)) then
    begin { Het cach di: Dau hang }
      CachDi := -1;
      exit;
    end;
  CachDi := a[sx,sy];
  if CachDi = 0 then { Con nuoc di nhung se thua }
    for ny := 1 to Min(sx-1,M) do
      if (ny <> sy) then
        begin
          nx := sx - ny;
          exit;
        end;
    { Chac thang }
    for ny := Min(sx-1,M) downto 1 do
      if (ny <> sy) then
        if (a[sx-ny,ny] = 0) then
          begin { chac thang }
            nx := sx - ny; exit;
          end;
    end;
```

```
end;
```

Hàm **Min(a,b)** cho ra giá trị min giữa hai số a và b cần được mô tả trước như sau:

```
function Min(a,b: integer): integer;
begin
  if (a < b) then Min := a else Min := b;
end;
```

Chương trình C# dưới đây mô tả một ván Cờ Bảng  $50 \times 7$  giữa hai đấu thủ A (đi trước) và B. Quân cờ xuất phát tại vị trí (49,5). Ván này A sẽ thắng.

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1 {
  class Program {
    static int maxn = 501, maxm = 51;
    static int[,] a = new int [maxn,maxm];
    static void Main(string[] args) {
      Game(50, 7, 49, 5);
      Console.WriteLine("\n \n Fini");
      Console.ReadLine();
    }
    static void Game(int n, int m, int x, int y) {
      Console.WriteLine("\n The co " + n +
        " X "+m+" @ = ("+x+","+y+ ")");
      Ket(m, x, y);
      Show(a, x, m);
      while (true) {
        Console.Write("\n A: ( " + x + " , " + y + " ) ");
        if (CachDi(m, x, y, ref x, ref y) == -1) {
          Console.Write("Dau hang !!!");
          Console.ReadLine();
          return;
        }
        else Console.Write(" ==> ( " + x + " , " + y + " ) ");
        if (Console.Read() == '.') return;
        Console.WriteLine("\n B: ( " + x + " , " + y + " ) ");
        if (CachDi(m, x, y, ref x, ref y) == -1) {
          Console.Write("Dau hang !!!");
          Console.ReadLine();
          return;
        }
        else Console.WriteLine(" ==> ( " + x + " , " + y + " ) ");
        if (Console.Read() == '.') return;
      } // while
    }
    static int CachDi(int m, int sx, int sy, ref int nx, ref int ny)
    {
      if (sx == 1 || (sx == 2 && sy == 1)) return -1;
      if (a[sx, sy] == 0) { // Di duoc, nhung thua
        for (ny = 1; ny <= Min(sx-1,m); ++ny)
          if (ny != sy) { nx = sx - ny; return 0; }
      }
      for (ny = Min(sx-1,m); ny > 0; --ny)
        if (ny != sy)
          if (a[sx - ny, ny] == 0) // Chac thang
            { nx = sx - ny; return 1; }
```

```

        return 0;
    }
    static int Min(int a, int b) { return (a < b) ? a : b; }
    static int Ket(int m, int x, int y) {
        Array.Clear(a, 0, a.Length);
        for (int i = 1; i < x; ++i) { // voi moi dong i
            int minj = Min(x - i, m);
            for (int j = 1; j <= minj; ++j) // xet cot j
                if (a[i, j] == 0)
                    for (int k = 1; k <= m; ++k)
                        if (k != j) a[i + j, k] = 1;
        }
        return a[x, y];
    }
    static void Show(int[,] s, int n, int m) {
        Console.WriteLine();
        for (int i = 1; i <= n; ++i) {
            Console.Write("\n"+i+"." );
            for (int j = 1; j <= m; ++j)
                Console.Write(a[i, j] + " ");
        }
    }
}
// Program
}

```

Nếu N có kích thước lớn, thí dụ, cỡ triệu dòng, còn số cột M vẫn đủ nhỏ, thí dụ  $M \leq 50$  và để ra chỉ yêu cầu cho biết người đi trước thắng hay thua chứ không cần lý giải từng nước đi thì ta vẫn có thể sử dụng một mảng nhỏ cỡ  $51 \times 51$  phần tử để giải bài toán trên. Ta khai báo kiểu mảng như sau:

```

const mn = 51;
type
    MB1 = array[0..mn] of byte;
    MB2 = array[0..mn] of MB1;
var a: MB2;
    N: longint;
    M: integer;
    ...

```

Ta sử dụng mảng index dưới đây để chuyển đổi các số hiệu dòng tuyệt đối thành số hiệu riêng trong mảng nhỏ a. Bạn chỉ cần lưu ý nguyên tắc sau đây khi xử lý các phép thu gọn không gian: *Không ghi vào vùng còn phải đọc dữ liệu*. Thực chất đây là một hàm *băm* các giá trị i trong khoảng  $1..N$  vào miền  $0..M$  bằng phép chia dư:  $i \rightarrow (i-1) \text{ mod } (M+1)$  như mô tả trong hàm **index**.

```

function index(i,M: integer): integer;
begin
    index := i mod (M+1);
end;
function Ket(M: integer;
             x: longint; y: integer): integer;
var i: longint; j,k: integer;
begin
    fillchar(a,sizeof(a),0);
    for i:= 1 to x-1 do
        begin
            k := index(i+M,M);
            fillchar(a[k],sizeof(a[k]),0);
            for j:=1 to Min(x - i, M)do
                if (a[index(i,M),j] = 0) then
                    for k := 1 to M do

```

```

        if (k <> j) then
            a[index(i+j,M),k] := 1;
    end;
    Ket := a[index(x,M),y];
end;

//C#
static int Index(int i, int m) { return i % (m + 1); }
static int Ket(int m, int x, int y){
    int id, Minj, i, j, k, v ;
    Array.Clear(a, 0, b.Length);
    for (i = 1; i < x; ++i) {
        id = Index(i + m, m);
        for (v = 1; v <= m; ++v) a[id, v] = 0;
        minj = Min(x - i, m);
        for (j = 1; j <= minj; ++j) // xet cot j
            if (a[Index(i, m), j] == 0)
                for (k = 1; k <= m; ++k)
                    if (k != j) a[Index(i + j, m), k] = 1;
    }
    return a[Index(x,m), y];
}

```

Đến đây ta thử mở rộng điều kiện của bài toán như sau: Hãy cho biết, với các giá trị cho trước là kích thước bảng  $N \times M$ , vị trí xuất phát của quân cờ @ (x,y) và đầu thủ A đi trước thì A thắng hoặc thua sau bao nhiêu nước đi ?

### Nguyên tắc của các trò chơi đối kháng

*Nếu biết là thắng thì tìm cách thắng nhanh nhất,  
Nếu biết là sẽ thua thì cố kéo dài cuộc chơi để  
có thể thua chậm nhất.*

Ta vẫn sử dụng bảng A để diễn trị với các qui ước mới sau đây:

Nếu từ ô (i,j) người đi trước có thể thắng sau b nước đi thì ta đặt  $a[i,j] = +b$ ; ngược lại nếu từ ô này chỉ có thể dẫn đến thua sau tối đa b nước đi thì ta đặt  $a[i,j] = -b$ . Một nước đi là một lần di chuyển quân cờ của một trong 2 người chơi. Ta cũng qui ước  $a[i,j] = 0$  có nghĩa là đầu thủ xuất phát từ ô (i,j) sẽ hết cách đi do đó chấp nhận thua ngay. Kí hiệu  $(i,j) \rightarrow (u,v)$  nếu có nước đi hợp lệ từ ô (i,j) sang ô (u,v). Từ nguyên tắc của các trò chơi đối kháng ta suy ra

(1) Nếu từ ô (i,j) có những nước đi hợp lệ dẫn đến thua (làm cho đối phương) thua thì ta chọn cách thắng nhanh nhất bằng cách đặt

$$a[i,j] = \min \{ -a[u,v] \mid (i,j) \rightarrow (u,v), a[u,v] \leq 0 \} + 1$$

(2) Nếu từ ô (i,j) mọi nước đi hợp lệ đều dẫn đến thua (tạo cho đối phương thắng) thì ta phải chọn cách thua chậm nhất bằng cách đặt

$$a[i,j] = -(\max \{ a[u,v] \mid (i,j) \rightarrow (u,v), a[u,v] > 0 \} + 1)$$

	1	2	3	4
1	0	0	0	0

Sau khi lấp đầy 0 cho bảng a ta lần lượt duyệt các dòng i từ 1 đến x. Với mỗi dòng ta duyệt các cột j từ 1 đến M. Nếu gặp trị  $a[i,j] = 0$  ta hiểu là vị trí này sẽ dẫn đến thua. Ta cần tính số bước thua chậm nhất rồi gán cho  $a[i,j]$ . Tiếp đến, do vị trí  $(i,j)$  là thua nên ta phải cập nhật lại các giá trị  $a[u,v]$  ứng với các vị trí  $(u,v) \rightarrow (i,j)$ .

Bạn thử điền trị cho bảng a với  $N = 12, M = 4$ . Trong thí dụ này,  $a[8,2] = -4$  có nghĩa là nếu quân cờ @ đặt ở vị trí  $(8,2)$  thì người đi trước sẽ thua sau 4 nước đi. Thật vậy, gọi A là người đi trước, ta thấy nếu A di chuyển @ đến  $(7,1)$  thì B sẽ đi tiếp để thắng sau 3 nước đi; A không thể đến dòng 6 (?). Nếu A đến  $(5,3)$  thì B sẽ đi tiếp để thắng sau 1 nước. Nếu A đến  $(4,4)$  thì B sẽ đi tiếp 1 nước nữa để đến  $(1,3)$  là thắng.

Tại sao trong hàm **Ket** ta phải tính thế thua trước khi tính thế thắng. Vì khi gặp  $a[i,j] = 0$  thì ta cần cập nhật giá trị này để biết được là sẽ thua sau bao nhiêu nước đi. Sau đó, do cơ chế lập luận lùi, chỉ khi nào ta biết số nước thua tại  $a[i,j]$  thì mới tính tiếp được các thế thắng dẫn đến hé thua này.

```

function Ket(M,x,y: integer): integer;
  var i, j: integer;
begin
  fillchar(a,sizeof(a), 0);
  for i := 1 to x do
    for j := 1 to M do
      if (a[i,j] = 0) then
        begin
          TinhNuocThua(M,i,j);
          TinhNuocThang(M,x,i,j);
        end;
      Ket := a[x,y];
    end;
  Procedure TinhNuocThua(M,i,j: integer);
    var k, vmax: integer;
  begin
    vmax := -1;
    for k := 1 to Min(i-1,M) do
      if (k <> j) then
        vmax := Max(vmax, a[i-k,k]);
    a[i,j] := -(vmax + 1);
  end;
  Procedure TinhNuocThang(M,x,i,j: integer);
    var k, d, v: integer;
  begin { Xet dong i+j }
    d := i+j;
    if (d <= x) then { quan co con trong vung can xu ly }
      begin
        v := -a[i,j] + 1;
        for k := 1 to M do
          if (k <> j) then
            if (a[d,k] > 0) then a[d,k] := Min(a[d,k], v)
            else a[d,k] := v;
      end;
  end;
// C#
  static int Ket(int n, int m, int x, int y) {
    Array.Clear(a, 0, a.Length);
    for (int i = 1; i <= x; ++i) { // voi moi dong i
      for (int j = 1; j <= m; ++j) // xet cot j

```

2	0	1	1	1
3	1	1	1	1
4	1	1	-2	1
5	1	1	1	-2
6	-2	-2	-2	-2
7	3	3	3	3
8	3	-4	3	3
9	3	3	3	3
10	3	3	3	5
11	-4	-4	-4	-4
12	-4	5	5	5

Tab Game với  
 $N = 12, M = 4$ .

```

        if (a[i, j] == 0) {
            TinhNuocThua(m, i, j);
            TinhNuocThang(m, x, i, j);
        }
    }
    return a[x,y];
}
static void TinhNuocThua(int m, int i, int j) {
    int vmax = -1, km = Min(i-1,m);
    for (int k = 1; k <= km; ++k)
        if (j != k) vmax = Max(vmax, a[i - k, k]);
    a[i,j] = -(vmax + 1);
}
static void TinhNuocThang(int m, int x, int i, int j){
    int d = i + j;
    if (d > x) return;
    int vmin = -a[i,j]+1;
    for (int k = 1; k <= m; ++k)
        if (k != j)
            a[d,k] = (a[d, k] > 0) ? Min(a[d, k], vmin) : vmin;
}

```

Với  $N$  cờ triệu và  $M$  đú nhỏ bạn cũng có thể vận dụng các kĩ thuật tương tự như đã trình bày để có thể tính số nước thắng/thua, tuy nhiên trong trường hợp này bạn phải khai báo mảng  $a$  rộng gấp đôi, tức là  $a$  phải chứa  $2M+2$  dòng gồm  $M$  dòng trước dòng đang xét và  $M$  dòng sau dòng đang xét. Các dòng trước và sau này dùng để cập nhật số nước đi thắng/thua.

Đến đây ta có thể sử dụng thuật toán Cờ bảng để giải bài Cờ Đáy sau đây.

### Bài 3.13. Cờ đú

Bàn cờ là một giải băng chia ô mă số từ 1 đến  $N$ . Hai đấu thủ  $A$  và  $B$  đan xen nhau, mỗi người đi một nước,  $A$  luôn đi trước. Một quân cờ @ đặt cạnh ô  $x$ . Tại nước đi thứ  $i$  phải đặt quân cờ lên (về phía chỉ số nhỏ)  $d_i$  ô,  $1 \leq d_i \leq M$  và không được lặp lại cách vừa đi của đấu thủ trước, tức là  $d_i \neq d_{i-1}$ . Đấu thủ nào đến lượt mình không đi nổi thì thua. Giả thiết là hai đấu thủ đều chơi rất giỏi. Biết  $N, M, x$  và  $A$  là đấu thủ đi trước. Hãy cho biết

a)  $A$  thắng (ghi 1) hay thua (ghi 0).

b)  $A$  thắng hay thua sau bao nhiêu nước đi?

Giả sử hai đấu thủ đi v nước đi với mức đú lần lượt là  $d_1, d_2, \dots, d_v$  thì giả thiết của đề bài cho biết hai mức đú kề nhau là  $d_{i-1}$  và  $d_i$  phải khác nhau. Giả sử bàn cờ có  $N = 12$  ô, quân cờ đặt cạnh ô  $x = 7$ , mỗi nước đi phải di chuyển quân cờ lên 1, 2.., hoặc  $M = 4$  ô và không được lặp lại nước vừa đi của người trước. Nếu  $A$  đi trước thì sẽ có thể thắng sau tối đa 4 nước đi. Chẳng hạn,  $A$  đi lên 1 bước  $d_1 = 1$  đú đến ô 6.  $B$  có thể chọn một trong 3 cách đi ứng với  $d_2 = 2, 3$ , hoặc 4. đú đến ô 4, ô 3 hoặc ô 2. Nếu  $B$  chọn  $d_2 = 2$  đú đến ô 4 thì  $A$  di chuyển lên theo  $d_3 = 3$  đú đến ô 1 khiến cho  $B$  thua. Nếu  $B$  chọn  $d_2 = 3$  đú đến ô 3 thì  $A$  di chuyển lên theo  $d_3 = 2$  đú đến ô 1 khiến cho  $B$  thua.

Cờ đú  $N = 12, M = 4, x = 7$   
và Tab Game tương ứng.

khiến cho  $B$  thua. Cuối cùng, nếu  $B$  chọn  $d = 4$  đú đến ô 2 thì  $A$  di chuyển lên theo  $d_3 = 1$  đú đến ô 1 khiến cho  $B$  thua. Các giá trị  $d_i$  theo từng phương án khi đó là như sau:

Phương án 1: (1, 2, 3).

Phương án 2: (1, 3, 2).

Phương án 3: (1, 4, 1).

Thuật toán

	0	1	2	3	4
1	0	0	0	0	0
2	0	1	1	1	1
3	1	1	1	1	1
4	1	1	-2	1	
5	1	1	1	-2	
6	-2	-2	-2	-2	
7	@	3	3	3	3
8	3	-4	3	3	
9	3	3	3	3	
10	3	3	3	5	
11	-4	-4	-4	-4	
12	-4	5	5	5	

Chúng ta hãy thiết lập sự tương ứng giữa cờ đầy và cờ bảng và chỉ ra rằng A thắng cờ đầy khi và chỉ khi A thắng cờ bảng tương ứng. Cờ đầy N ô, giới hạn bước đi M và ô xuất phát x tương ứng với cờ bảng N ô dài, M ô ngang và vị trí xuất phát  $(x,y)$  trong đó  $y = d_1$  là nước đi hợp lệ đầu tiên đến 1 ô thua,  $x := x - d_1$ .

Có một cách tư duy để có thể dễ dàng tính được  $(x,y)$  như sau. Ta thêm cho cờ bảng một ô giả mang mã số 0 và qui định rằng quân cờ bảng xuất phát tại ô  $(x,0)$  - dòng v, cột 0. Sau đó, trong cuộc chơi với cờ bảng này không ai được phép di chuyển đến cột 0. Nếu A đi đầu tiên thì chắc chắn sẽ di chuyển từ ô  $(x,0)$  đến một ô mà B chắc thua. Trong thí dụ này, nước đi đầu tiên của A sẽ là  $(7,0) \rightarrow (6,1)$ , tức là chuyển quân cờ từ cột 0 sang cột 1.

Bài cờ đầy mới xem tưởng như đơn giản nhưng để giải được ta lại phải xét bài khó hơn nhưng có lời giải trong sáng, dễ hiểu

### Bài 3.14. Bóc sỏi H

*Dạng phát biểu khác của bài Cờ đầy*

*Cho đồng sỏi N viên, hai đấu thủ A và B lần lượt đi, A đi nước đi đầu tiên. Mỗi nước đi đầu thủ buộc phải bóc tối thiểu 1 viên, tối đa M viên trong đồng và không được lặp lại nước vừa đi của người trước. Thí dụ, nếu đấu thủ A vừa bóc v viên sỏi thì đến lượt mình, đấu thủ B không được bóc v viên nữa. Đấu thủ nào đến lượt mình không đi nổi thì thua. Cả hai đấu thủ đều chơi rất giỏi. Cho biết a) A thắng hay thua. b) A thắng hay thua sau bao nhiêu nước đi?*

# Chương 4

## Các thuật toán sắp đặt

### 4.1 Cờ tam tài

Olympic quốc tế

Một số quốc gia như Ba Lan, Bỉ, Pháp... có quốc kỳ tạo từ ba giải màu thường được gọi là cờ tam tài. Ba bạn trẻ A, B và C chơi trò ghép hình để tạo thành một lá cờ tam tài với ba giải màu dọc lần lượt tính từ trái qua phải là xanh (X), trắng (T) và đỏ (D). Một bàn để ghép cờ có kích thước  $2N \times 3N$  ô vuông đơn vị được kẻ sẵn thành lưới ô vuông với mã số các hàng tính từ trên xuống dưới là 1, 2, ...,  $2N$  và mã số các cột tính từ trái qua phải là 1, 2, ...,  $3N$ . Đầu tiên bạn A chọn một ô trên cột 1 có tọa độ là  $(Ax, Ay = 1)$ , bạn B chọn một ô trên dòng cuối cùng có tọa độ là  $(Bx=2N, By)$ , bạn C chọn một ô trên cột cuối cùng có tọa độ là  $(Cx, Cy = 3N)$ . Sau đó lần lượt theo thứ tự quay vòng A, B, C ba bạn chọn các mảnh ghép đơn vị  $1 \times 1$  với màu phù hợp để đặt vào các ô trong bàn cờ. Lần đầu tiên mỗi bạn đặt một mảnh ghép vào ô đã chọn. Những lần tiếp theo, đến lượt mình, mỗi bạn đặt một số mảnh ghép kề với mảnh ghép do chính bạn ấy đã đặt tại lần trước. Dĩ nhiên, mỗi ô trên bàn chỉ được đặt đúng 1 mảnh ghép. Bạn nào không thể ghép được thì bạn đó ngừng chơi, những người còn lại sẽ tiếp tục chơi đến khi hoàn thành lá cờ. Biết các giá trị  $N$ ,  $Ax$ ,  $By$  và  $Cx$ . Hãy cho biết mỗi bạn đã ghép được bao nhiêu mảnh mỗi màu.

<b>A</b>					
					<b>C</b>
	<b>B</b>				

Cờ tam tài  $4 \times 6$

$N = 2$

Với thí dụ như trong hình,  $N = 2$ ,  $Ax = 2$ ,  $By = 2$ ,  $Cx = 3$  ta tính được kết quả như trong bảng. Ý nghĩa của các ô trên bàn ghép cờ cho biết bạn nào trong lần đi thứ mấy của mình, ghép mảnh màu gì. Thí dụ, A5:T cho biết bạn A, trong lần đi thứ 5 ghép mảnh màu trắng. Ô xuất phát của mỗi bạn kí hiệu là 0.

	<b>①</b>	<b>②</b>	<b>③</b>	<b>④</b>	<b>⑤</b>	<b>⑥</b>
①	A1:X	A2:X	A3:T	A4:T	C3:D	C2:D
②	A0:X	A1:X	A2:T	A3:T	C2:D	C1:D

	Xanh	Trắng	Đỏ
<b>A</b>	5	4	0

③	A1:X	B1:X	B2:T	C2:T	C1:D	<u>C0:D</u>
④	B1:X	B0:X	B1:T	B2:T	C2:D	C1:D

Cờ tam tài,  $N = 2$ ,  $A(2,1)$ ,  $B(4,2)$ ,  $C(3,6)$

X: Xanh, T: Trắng, D: Đỏ.

B	3	3	0
C	0	1	8

Kết quả

## Thuật toán

Bài này khá dễ giải. Nếu bạn khéo tổ chức dữ liệu thì chương trình sẽ rất gọn. Trước hết ta cần xác định rằng mỗi ô  $(i,j)$  trên bàn cờ sẽ do bạn nào ghép: A, B hay C? Ta định nghĩa khoảng cách giữa hai ô  $(i,j)$  và  $(x,y)$  trên bàn cờ là số ô ít nhất nằm trên đường đi từ ô này đến ô kia qua các ô kè cạnh nhau. Khoảng cách này chính là tổng chiều dài hai cạnh kè nhau của hình chữ nhật nhận hai ô đã cho làm hai đỉnh đối diện, do đó được tính theo công thức

$$d = \text{abs}(i-x) + \text{abs}(j-y) + 1$$

Giá trị d có ý nghĩa gì? Nếu ta qui định đánh số các lần đi cho mỗi đầu thu là 0, 1, 2, ... thì  $d-1$  cho biết lần đi thứ mấy của mỗi bạn. Vì trật tự tính lần đi của các bạn là A  $\rightarrow$  B  $\rightarrow$  C nên ta cần xác định giá trị min trong ba khoảng cách  $d_A$ ,  $d_B$  và  $d_C$ . Tuy nhiên chúng ta sẽ không ngoan một chút, cụ thể là ta sẽ tính d theo công thức hụt 1

$$d = \text{abs}(i-x) + \text{abs}(j-y)$$

và viết hàm min3 nhận vào là ba giá trị  $d_A$ ,  $d_B$  và  $d_C$  và cho ra là tên của người được ghép mảnh tại ô đang xét.

```
function Min3(a,b,c: integer): char;
  var k: char;
begin
  k := 'A';
  if a > b then begin k := 'B'; a := b end;
  if a > c then k := 'C';
  Min3 := k;
end;
```

Sau khi xác định được chủ của mảnh ghép tại ô  $(i,j)$  ta dễ dàng tính được màu của mảnh ghép tại ô đó. Vì lá cờ có ba màu và ta tạm qui ước các giải màu tính từ trái qua phải là 0, 1 và 2 nên màu cần chọn để đặt tại ô  $(i,j)$  khi đó sẽ là  $(j-1) \bmod N$ .

Ta khai báo mảng kq dùng để tích lũy kết quả như sau:

```
kq: array['A'..'C',0..2] of integer;
```

Khi đó  $c[v,i]$  sẽ cho biết bạn v đã ghép bao nhiêu quân màu i,  $v = 'A', 'B', 'C'$ ;  $i = 0$  (màu Xanh), 1 (màu Trắng), 2 (màu Đỏ).

Các biến chung của chương trình sẽ là:

```
var
n: integer; { Ban co co kich thuoc 2nx3n }
Ax,Ay,Bx,By,Cx,Cy: integer; { Toa do xuat phat cua A, B, C }
kq: array['A'..'C',0..2] of integer; { Chua ket qua }
```

Thủ tục XuLi sẽ duyệt lần lượt mỗi ô  $(i, j)$  trên bàn cờ, xác định chủ nhân của ô này và số màu của mảnh cần ghép để tích lũy cho chủ nhân đó.

```
procedure XuLi;
  var i,j: integer;
begin
  fillchar(kq,sizeof(kq),0);
  for I := 1 to 2*N do
    for j := 1 to 3*N do
      inc(c[Min3(abs(i-Ax)+abs(j-Ay),abs(i-Bx)+abs(j-By),abs(i-Cx)+abs(j-Cy))],
```

```

    abs(i-Cx)+abs(j-Cy)),(j-1) div N];
end;

```

## Chương trình C#

Chương trình C# dưới đây thực hiện với dữ liệu cho trước  $N = 2$ , A(2,1), B(4,2), C(3,6).

// C#

```

using System;
using System.Collections.Generic;
using System.Text;
namespace SangTao2 {
    class CoTamTai {
        static int n = 2; // Ban co kich thuoc 2Nx3N
        static int[,] kq = new int[3,3];
        static int Ax = 2, Ay = 1, Bx = 2*n, By = 2,
                  Cx = 3, Cy = 3*n; // Toa do xuat phat

        static void Main(string[] args) {
            XuLi();
            for (int i = 0; i < 3; ++i) {
                for (int j = 0; j < 3; ++j)
                    Console.Write(kq[i, j] + " ");
                Console.WriteLine();
            }
            Console.ReadLine();
        }
        static int Min3(int a, int b, int c) {
            int min = 0;
            if (a > b) { min = 1; a = b; }
            if (a > c) min = 2;
            return min;
        }
        static void XuLi() {
            Array.Clear(kq, 0, kq.Length);
            int n2 = 2 * n;
            int n3 = 3 * n;
            for (int i = 1; i <= n2; ++i)
                for (int j = 1; j <= n3; ++j)
                    ++kq[Min3(Math.Abs(i-Ax)+Math.Abs(j-Ay),
                               Math.Abs(i-Bx)+Math.Abs(j-By),
                               Math.Abs(i-Cx)+Math.Abs(j-Cy)),(j-1)/n];
        }
    } // Co Tam Tai
} // SangTao2

```

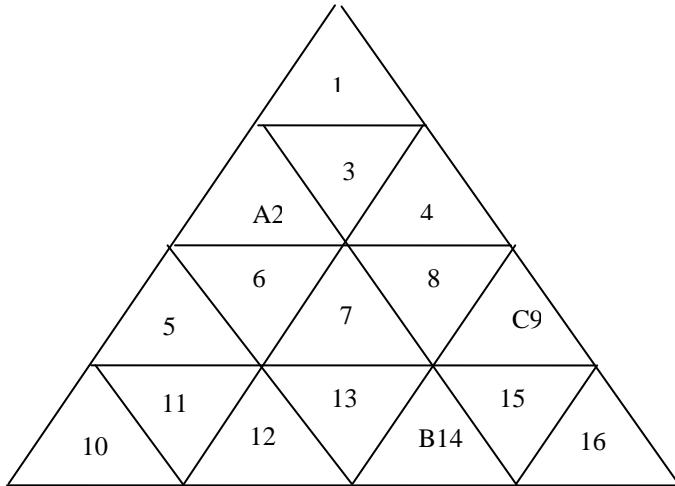
## Độ phức tạp

Ta phải duyệt mọi ô trên bàn cờ vậy độ phức tạp tính toán cỡ  $N^2$ .

Bài sau đây tương tự như bài trên nhưng khó hơn về các thủ tục mã hóa.

## 4.2 Lưới tam giác đều

*Cho tam giác đều ABC, đỉnh A, cạnh dài N đơn vị. Tại các điểm chia nguyên trên các cạnh ta kẻ các đường thẳng song song chia tam giác thành  $N^2$  tam giác đơn vị (TGDV). Mã số cho các TGDV theo trật tự từ trên xuống và từ trái qua phải là 1, 2, ...,  $N^2$ . Ba bạn A, B và C được cấp mỗi bạn một TGDV khác nhau làm nơi xuất phát trên các cạnh AB cho bạn A, BC cho bạn B và AC cho bạn C. Lần lượt theo thứ tự quay*



Lưới Tam giác  $N = 4$ ,  $NA = 2$ ,  $NB = 14$ ,  
 $NC = 9$ .

Kết quả,  $A: 9$ ,  $B: 5$ ,  $C: 2$ .

vòng  $A$ ,  $B$ ,  $C$  viết chữ cái tên mình vào các TGĐV kề cạnh với các tam giác mà mình đã viết ở lần trước. Biết các giá trị  $N$ , và các điểm xuất phát  $NA$ ,  $NB$  và  $NC$ , tính số chữ cái mỗi loại mỗi bạn đã viết.

## Tổ chức dữ liệu

Các biến dùng chung:

```
var n: longint; { Do dai canh tam giac }
f,g: text; { input, output file }
AN, BN, CN: longint; { O xuat phat }
Ad, Av, Bd, Bv, Cd, Cv: longint; { Toa do xuat phat }
Ak,Bk,Ck: longint;
A,B,C: longint; { con dem }
Kq: array ['A'..'C'] of longint;
```

trong đó  $n$  là chiều dài một cạnh của tam giác đều;  $AN$ ,  $BN$  và  $CN$  là số hiệu của các ô xuất phát tương ứng cho  $A$ ,  $B$  và  $C$ .

## Thuật toán

Xét các tam giác đơn vị từ đỉnh xuống đến cạnh đáy của bàn cờ. Ta thấy, trên dòng 1 có 1 TGĐV, dòng 2 có 3, dòng 3 có 5 TGĐV... Tổng quát, trên dòng  $i$  tính từ đỉnh xuống đến đáy sẽ có  $2^i - 1$  TGĐV. Trên mỗi dòng  $i$  ta gán số hiệu cho các TGĐV là  $1, 2, \dots, 2i-1$ . Ta định nghĩa tọa độ của một tam giác đơn vị có số hiệu (tuyệt đối theo đầu bài) cell là cặp số  $(d, v)$  trong đó  $d$  là số hiệu dòng chứa TGĐV đó và  $v$  là số hiệu của tam giác đó trên dòng  $d$ . Thủ tục **ToaDo** dưới đây tính tọa độ cho một TGĐV theo cell - số hiệu (tuyệt đối) của TGĐV như cách mã số của đề bài. Thủ tục cho ra hai giá trị, dòng - dòng chứa TGĐV cell và viTri - số hiệu của TGĐV trên dòng đó mà ta gọi là số hiệu tương đối. Thí dụ, **ToaDo(15, d, v)** cho ta  $d = 4$ ,  $v = 6$ .

```

procedure ToaDo(cell: longint;var dong, viTri:longint);
begin
  dong := 0;
  while cell > 0 do
    begin
      dong := dong + 1;
      cell := cell - (2*dong-1);
    end;
  viTri := cell + (2*dong-1);
end;

```

Hàm **KhoangCach** dưới đây tính khoảng cách giữa hai TGĐV theo tọa độ (d1,v1) và (d2,v2), trong đó d1, d2 là số hiệu dòng, v1 và v2 là số hiệu tương đối của chúng (trên dòng). Giống như bài trước, khoảng cách trong bài này chính là số TGĐV ít nhất, kề cạnh nhau trên đường đi từ TGĐV (d1,v1) đến TGĐV (d2,v2). Trước hết ta đổi chỗ hai tọa độ, nếu cần, sao cho tam giác thứ nhất luôn nằm ở dòng trên so với tam giác thứ hai, tức là  $d1 \leq d2$ . Sau đó ta nhận xét như sau:

*Nếu một TGĐV có đỉnh quay lên trên thì*

*\* Số hiệu tương đối của nó là số lẻ, và*

*\* Nó sẽ là đỉnh của một tam giác đều chứa nó và có các cạnh song song với các cạnh của bàn cờ.*

*Nếu một TGĐV có đỉnh quay xuống dưới thì*

*\* Số hiệu tương đối của nó là số chẵn, và*

*\* TGĐV kề cạnh với nó trên cùng dòng sẽ có đỉnh quay lên trên.*

Ta gọi các TGĐV có đỉnh quay lên trên là *tam giác lẻ* để phân biệt với các TGĐV *chẵn* - có đỉnh quay xuống dưới.

Nếu TGĐV thứ nhất (d1,v1) là tam giác lẻ ta xét tam giác lớn hơn tạo bởi các TGĐV nhận tam giác lẻ này làm đỉnh và có cạnh đáy trên dòng d2. Ta tính hai đỉnh trên đáy của tam giác này trên dòng d2 là C1 và C2 theo công thức

```

d := 2*(d2 - d1);
c1 := v1;
c2 := v1 + d;

```

Tiếp đến ta xét vị trí v2 trên cạnh đáy có thể nằm giữa C1 và C2 hoặc nằm ngoài đoạn [C1, C2] đồng thời xét v2 là tam giác chẵn hay lẻ.

```

function KCLe(d1,v1,d2,v2: longint):longint;
var c1,c2,d: longint;
begin
  { v1 <= v2 }
  d := 2*(d2 - d1);
  c1 := v1;
  c2 := v1 + d;
  if (c1 <= v2) and (v2 <= c2) then
    begin
      if odd(v2) then KCLe := d
      else KCLe := d - 1;
      exit;
    end;
  KCLe := d + Min(abs(v2-c1),abs(v2-c2));
end;

```

Nếu TGĐV thứ nhất (d1,v1) là tam giác chẵn thì ta lùi lại một dòng để xét TGĐV lẻ có chung đáy với TGĐV thứ nhất rồi tính toán như trên và giảm kết quả 1 đơn vị.

```

function KhoangCach(d1,v1,d2,v2: longint):longint;
var t: longint;
begin
  if d1 > d2 then
    begin

```

```

t := v1; v1 := v2; v2 := t;
t := d1; d1 := d2; d2 := t;
end;
{ v1 <= v2 }
if odd(v1) then KhoangCach := KCLe(d1,v1,d2,v2)
else KhoangCach := KCLe(d1-1,v1-1,d2,v2) - 1;
end;
procedure XuLi;
var d,v,j: longint;
    Ad, Av, Bd, Bv, Cd, Cv: longint;
begin
    fillchar(kq,sizeof(kq),0);
    ToaDo(NA, Ad, Av);
    ToaDo(NB, Bd, Bv);
    ToaDo(NC, Cd, Cv);
    for d := 1 to N do
        for v := 1 to 2*d - 1 do
            inc(kq[Min3(KhoangCach(Ad,Av,d,v),
                          KhoangCach(Bd,Bv,d,v),
                          KhoangCach(Cd,Cv,d,v))]);
end;

```

## Chương trình C#

Chương trình C# dưới đây giải bài toán với dữ liệu cho trước N = 4, A, B và C lần lượt xuất phát tại các TGĐV 2, 14 và 9 như thí dụ đã cho.

```

// C#
using System;
using System.Collections.Generic;
using System.Text;
namespace SangTao2 {
    class TamGiacDeu {
        static int n = 4, NA = 2, NB = 14, NC = 9;
        static int[] KQ = new int[3];
        static void Main(string[] args){
            XuLi();
            for (int i = 0; i < 3; ++i)
                Console.WriteLine(KQ[i] + " ");
            Console.ReadLine();
        }
        // Tinh dong va vi tri tren dong
        // theo so hieu cua TGĐV
        static void ToaDo(int cell, out int dong, out int viTri){
            dong = 0;
            while (cell > 0){
                ++dong; cell -= (2*dong - 1);
            }
            viTri = cell + (2*dong - 1);
        }
        static int KhoangCach(int d1, int v1, int d2, int v2){
            if (d1 > d2){
                int t;
                t = d1; d1 = d2; d2 = t;
                t = v1; v1 = v2; v2 = t;
            }
            return (v1%2==1)?KCLe(d1,v1,d2,v2):KCLe(d1-1,v1-1,d2,v2)-1;
        }
    }
}
```

```

        }
        static int KCLe(int d1, int v1, int d2, int v2){
            int c1=v1, d=2*(d2-d1), c2=v1+d;
            // Xet tam giac voi 3 dinh v1 c1 c2
            if (c1 <= v2 && v2 <= c2)
                return (v2 % 2 == 1) ? d : d-1;
            return d + Math.Min(Math.Abs(v2-c1),Math.Abs(v2-c2));
        }
        static int Min3(int a, int b, int c){
            int min = 0;
            if (a > b) { min = 1; a = b;}
            if (a > c) min = 2;
            return min;
        }
        static void XuLi(){
            int Ad, Av, Bd, Bv, Cd, Cv;
            ToaDo(NA,out Ad, out Av);
            ToaDo(NB,out Bd, out Bv);
            ToaDo(NC,out Cd, out Cv);
            Array.Clear(Kq, 0, Kq.Length);
            for (int d = 1; d <= n; ++d){
                int vv = 2*d-1;
                for (int v = 1; v <= vv; ++v)
                    ++KQ[Min3(KhoangCach(Ad,Av,d,v),
                               KhoangCach(Bd,Bv,d,v),
                               KhoangCach(Cd,Cv,d,v))];
            }
        }
    } // Tam Giac Deu
} // SangTao2

```

## Độ phức tạp

Ta phải duyệt mọi TGĐV trên bàn cờ, vậy độ phức tạp tính toán cỡ  $N^2$ .

### 4.3 Dạng biểu diễn của thừa

Cho số tự nhiên  $n \leq 480.000$ . Hãy phân tích  $n!$  ra tích của các thừa số nguyên tố theo trật tự tăng dần. Thí dụ,  $13! = 2^{10}.3^5.5^2.7.11.13$ . Kết quả hiển thị dưới dạng các dòng, mỗi dòng một số nguyên tố tiếp đến là số mũ tương ứng. Các số trên cùng dòng cách nhau qua dấu cách. Thí dụ trên cho ta kết quả hiển thị như sau

```

2 10
3 5
5 2
7 1
11 1
13 1

```

## Thuật toán

**Nhận xét** Cho số tự nhiên N và một số nguyên tố p. Khi đó,

Nếu viết dãy thừa số 1, 2, ..., N vào một bảng có p cột thì ta thấy có  $n_1 = N \text{ div } p$  dòng chia p,  $2p, \dots, n_1.p$  (ở cột cuối cùng). Nhóm các phần tử này lại ta được,

$1p.2p.\dots.n_1p = (1.2\dots.n_1).p^{n_1}$ . Thực hiện tương tự với tích  $1.2\dots.n_1$  ta thu được  $n_2 = n_1 \text{ div } p$  dòng chia p,  $2p, \dots, n_2.p$ ... Từ đây ta suy ra lũy thừa k của p,  $p^k$  trong dạng phân tích của  $N!$  sẽ là  $k = n_1+n_2+\dots+n_v$ , trong

đó  $n_i = n_{i-1} \text{ div } p$ ,  $n_1 = N \text{ div } p$ ,  $n_v = 0$ ,  $i = 2..v$ . Hàm tính lũy thừa của  $p$  trong dạng phân tích của  $N!$  bằng các phép chia liên tiếp khi đó sẽ như sau,

```
function Power(n,p: longint): byte;
var k: byte;
begin
  k := 0;
  while (n <> 0) do
    begin
      n := n div p;
      k := k + n;
    end;
  Power := k;
end;
```

Ta dùng hàm **NextPrime** để sinh lần lượt các số nguyên tố  $p$  trong khoảng  $2..N$  và tính **Power(N,p)**. Nếu giá trị này lớn hơn 0 thì ta hiển thị kết quả.

```
procedure Fac(n: longint);
const bl = #32; { Dau cach }
var p: longint; k: byte;
begin
  writeln;
  p := 2;
  while p <= n do
    begin
      k := Power(n,p);
      if (k > 0) then writeln(p,bl,k);
      p := NextPrime(p);
    end;
end;
```

Hai hàm phụ trợ.

Hàm **IsPrime(p)** kiểm tra  $p$  có phải là số nguyên tố hay không bằng cách xét xem trong khoảng từ 2 đến  $\sqrt{p}$  có ước nào không.

```
function IsPrime(p: longint): Boolean;
var i: longint;
begin
  IsPrime := false;
  if p < 2 then exit;
  for i := 2 to round(sqrt(p)) do
    if p mod i = 0 then exit;
  IsPrime := True;
end;
```

Hàm **NextPrime(p)** sinh số nguyên tố sát sau  $p$  bằng cách duyệt tuần tự các số lẻ sau  $p$  là  $p+2k$  nếu  $p$  lẻ và  $(p-1) + 2k$ , nếu  $p$  chẵn.

```
function NextPrime(p: longint): longint;
begin
  if p < 2 then
    begin
      NextPrime := 2;
      exit;
    end;
  if not odd(p) then p := p-1;
  repeat
    p := p+2;
  until IsPrime(p);
```

```

NextPrime := p;
end;

```

Ta có thể cải tiến khá mạnh tốc độ tính toán bằng các kỹ thuật sau.

Sinh sẵn các số nguyên tố trong khoảng từ 1..N bằng giải thuật Sàng mang tên nhà toán học Hi Lạp Eratosthenes. Từ vài nghìn năm trước, Eratosthenes đã dạy như sau:

### Bài giảng của Eratosthenes

Nếu trò muốn liệt kê toàn bộ các số nguyên tố nằm trong khoảng từ 1 đến N hãy làm như sau

1. Viết dãy số từ 1 đến N.
2. Xóa đi số 1 vì nó không phải là số nguyên tố, cũng không phải là hợp số. Nó là một số đặc biệt.
3. Lần lượt duyệt từ 2 đến  $\sqrt{N}$  như sau. Nếu gặp số chưa bị xóa thì đó chính là một số nguyên tố. Trò hãy xóa mọi bội của số này kể từ bình phương của nó trở đi.

Khi kết thúc, những số nào không bị xóa trên tấm bảng sẽ là các số nguyên tố. Đó là kho các số nguyên tố trong khoảng 1..N.



**Eratosthenes (276-194 tr. CN)** Nhà toán học lỗi lạc Hy Lạp Cổ đại. Ông sinh tại Cyrene, theo học trường phái Plato tại Athens. Hoàng đế Ptolemy II mời ông đến Alexandria để dạy cho hoàng tử

Sau ông được giao phụ trách thư viện Alexandria, một trung tâm lưu trữ và bảo tồn các tác phẩm văn hóa và khoa học nổi tiếng đương thời. Ngoài các công trình toán học, Eratosthenes còn có những đóng góp rất giá trị về đo lường. Ông đã tiến hành đo kích thước Trái Đất.

Thời đó chưa có giấy viết nên thay trò phải viết trên những tấm bảng bằng đất sét vào lúc đất còn dẻo, các số bị xóa được đục thủng. Sau khi phơi khô ta thu được những tấm bảng thủng lỗ chổ như một cái sàng gạo.

Với mảng **a[0..MN] of byte** đủ lớn, thí dụ, MN = 60.000 ta có thể ghi nhận các số nguyên tố nằm trong khoảng 1..MN. Ta qui ước a[i] = 0 thì i là số nguyên tố, a[i] = 1 ứng với số i bị dùi thủng nên i không phải là số nguyên tố.

```

procedure Eratosthenes(n: longint);
var i,j: longint;
begin
  fillchar(a,sizeof(a),0);
  for i := 2 to round(sqrt(n)) do
    if a[i]=0 then
      for j := i to (n div i) do a[i*j] := 1;
end;

```

Thủ tục phân tích N! ra thừa số nguyên tố dạng cài tiến sẽ như sau,

```

procedure NewFac(n: longint);
const bl = #32; { Dau cach }
var i,p: longint;
begin
  Eratosthenes(n);
  writeln;
  for i := 2 to n do
    if a[i] = 0 then
      begin
        p := Power(n,i);
        if p > 0 then writeln(i,bl,p);
      end;
end;

```

```
end;
```

Dùng kỹ thuật đánh dấu bit có thể tạo kho số nguyên tố cỡ 8.MN vì một byte có 8 bit, mỗi bit sẽ quản lí 1 số.

Mảng a vẫn được khai báo như trước: **a[0..MN] of byte** (quan trọng là chỉ số phải tính từ 0 trở đi) nhưng lúc này mỗi phần tử a[i] sẽ quản lí 8 số chứ không phải một số như trước. Tiếp đến bạn cần viết thêm ba thủ tục sau đây:

Thủ tục **BitOn(i)** - đặt trị 1 cho bit thứ i trong dãy bit a (bật bit). Các bit trong dãy a sẽ được mã số từ 0 đến 8MN-1= 480.000-1. Bản thân số 480.000 là hợp số nên ta có thể bỏ qua.

<pre><b>procedure BitOn(i: longint);</b>   <b>var b,p: longint;</b> <b>begin</b>   <b>b := i shr 3; { i div 8 }</b>   <b>p := i and 7; { i mod 8 }</b>   <b>a[b] := a[b] or (1 shl p);</b> <b>end;</b></pre>	<p><b>Đặt trị 1 cho bit i trong dãy bit a</b></p> <ol style="list-style-type: none"> <li>1. Xác định xem bit i nằm trong byte nào <b>b := i div 8</b></li> <li>2. Xác định xem bit i là bit thứ mấy trong byte b (tính theo trật tự 7,6,5,4,3,2,1,0) <b>p := i mod 8</b></li> <li>3. Lấy số nhị phân 8 bit 00000001 dịch trái p vị trí rồi cộng logic theo bit với a[b]. <b>a[b] := a[b] or (1 shl p);</b></li> </ol>
--	---

Bạn ghi nhớ sự tương đương của các phép toán sau đây

Phép toán	Phép toán tương đương
<b>x div 2<sup>k</sup></b>	<b>x shr k</b>
<b>x mod 2<sup>k</sup></b>	<b>x and 2<sup>k</sup>-1</b>
	Tính theo dạng này sẽ nhanh hơn

Thủ tục **BitOff(i)** đặt trị 0 cho bit thứ i trong dãy bit a (tắt bit).

<pre><b>procedure BitOff(i: longint);</b>   <b>var b,p: longint;</b> <b>begin</b>   <b>b := i shr 3; { i div 8 }</b>   <b>p := i and 7; { i mod 8 }</b>   <b>a[b]:=a[b] and (not(1 shl p));</b> <b>end;</b></pre>	<p><b>Đặt trị 0 cho bit i trong dãy bit a</b></p> <ol style="list-style-type: none"> <li>1. Xác định xem bit i nằm trong byte nào <b>b := i div 8;</b></li> <li>2. Xác định xem bit i là bit thứ mấy trong byte b (tính theo trật tự 7,6,5,4,3,2,1,0) <b>p := i mod 8;</b></li> <li>3. Lấy số nhị phân 6 bit 00000001 dịch trái p vị trí, lật rồi nhân logic theo bit với a[b]. <b>a[b]:=a[b] and (not(1 shl p));</b></li> </ol>
---	--

Hàm **GetBit(i)** cho ra trị (1/0) của bit i trong dãy bit a.

<pre><b>function GetBit(i: longint): byte;</b>   <b>var b,p: longint;</b></pre>	<p><b>Đặt trị 0 cho bit i trong dãy bit a</b></p> <ol style="list-style-type: none"> <li>1. Xác định xem bit i nằm trong byte nào</li> </ol>
---	--

<pre> begin   b := i shr 3;   p := i and 7; { i mod 8 }   GetBit := (a[b] shr p) and 1; end; </pre>	<p><b>b := i div 8;</b></p> <p>2. Xác định xem bit <math>i</math> là bit thứ mấy trong byte <math>b</math> (tính theo trật tự 7,6,5,4,3,2,1,0)</p> <p><b>p := i mod 8;</b></p> <p>3. Dịch <math>a[b]</math> qua phái <math>p</math> vị trí, rồi nhân logic theo bit với 00000001 để lấy bit phái nhất (bit 0).</p> <p><b>GetBit := (a[b] shr p) and 1;</b></p>
---	--

Các thủ tục cơ bản theo kỹ thuật xử lí bit khi đó sẽ như sau.

```

procedure Eratosthenes_B(n: longint);
var i,j: longint;
begin
  fillchar(a,sizeof(a),0);
  for i:=2 to round(sqrt(n)) do
    for j:=i to (n div i) do
      BitOn(i*j);
end;
procedure BFac(n: longint);
const bl = #32; { Dau cach }
var i,p: longint;
begin
  Eratosthenes_B(n);
  writeln;
  for i:=2 to n do
    if GetBit(i)=0 then
      begin
        p := Power(n,i);
        if P > 0 then writeln(i,bl,p);
      end;
end;

```

## Chương trình C#

```

//  C#
using System;
using System.Collections.Generic;
using System.Text;
namespace SangTao2 {
  class GiaiThua {
    static byte[] a = new byte[40000];
    static void Main(string[] args){
      BFac(13);
      Console.ReadLine();
    }
    static int Power(int n, int p){
      int k = 0;
      while (n != 0){ n /= p; k += n; }
      return k;
    }
    static void Fac(int n) {
      Console.WriteLine();
      int p = 2, k;
      while (p <= n){
        k = Power(n,p);

```

```

        if (k > 0) Console.WriteLine(p+" "+k);
        p = NextPrime(p);
    }
}

static bool IsPrime(int p){
    if (p<2) return false;
    if (p==2) return true;
    if (p % 2 == 0) return false;
    int cp = (int)(Math.Sqrt(p));
    for (int i=3; i <= cp; i+=2)
        if (p % i == 0) return false;
    return true;
}

static int NextPrime(int p){
    if (p < 2) return 2;
    if (p % 2 == 0) --p;
    do { p += 2; } while (!IsPrime(p));
    return p;
}

// Sang Eratosthene dung byte
static void Eratosthenes(int n){
    Array.Clear(a,0,a.Length);
    int sn = (int)Math.Sqrt(n);
    for (int i = 2; i <= sn; ++i)
        if (a[i]==0){
            int ni = n/i;
            for (int j = i; j <= ni; ++j) a[i*j] = 1;
        }
    }

// Gan 1 cho bit i
static void BitOn(int i){
    int b = i >> 3;
    int p = i & 7;
    a[b] |= (byte)(1 << p);
}

// Gan 0 cho bit i
static void BitOff(int i){
    int b = i >> 3;
    int p = i & 7;
    a[b] &= (byte)~(1 << p);
}

// Lay tri cua bit i
static byte GetBit(int i) {
    int b = i >> 3;
    int p = (i & 7);
    return (byte)((a[b] >> p)&1);
}

// Sang Eratosthene dung bit
static void Eratosthenes_B(int n){
    Array.Clear(a, 0, a.Length);
    int sn = (int)Math.Sqrt(n);
    for (int i = 2; i <= sn; ++i)
        if (GetBit(i) == 0) {
            int ni = n / i;
            for (int j = i; j <= ni; ++j) BitOn(i * j);
        }
}

```

```

        static void BFac(int n){
            int p;
            Eratosthenes_B(n);
            for (int i = 2; i <= n; ++i)
                if (GetBit(i)==0)
                {
                    p = Power(n,i);
                    if (p > 0) Console.WriteLine(i+" "+p);
                }
        }
    } // GiaiThua
} // SangTao2

```

## Độ phức tạp

Để liệt kê các số nguyên tố từ 1..N ta duyệt từ 1 đến  $\sqrt{N}$ , với mỗi số nguyên tố ta phải gạch tối đa  $\sqrt{N}$  các bội của chúng. Vậy độ phức tạp tính toán cỡ  $N \cdot \sqrt{N}$ .

## 4.4 Xếp sỏi

Cho một bảng chia lưới ô vuông  $N$  dòng mã số  $1..N$  tính từ trên xuống và  $M$  cột mã số  $1..M$  tính từ trái sang. Mỗi ô được phép đặt không quá 1 viên sỏi. Người ta cho trước giới hạn tổng số sỏi được phép đặt trên dòng  $i$  là  $d_i$ ,  $i = 1..N$  và trên mỗi cột  $j$  là  $C_j$ ,  $j = 1..M$ . Hãy tìm một phương án xếp được nhiều sỏi nhất trong bảng, biết rằng các dữ liệu đều hợp lệ và bài toán luôn có nghiệm.

### Thuật toán

Tổ chức dữ liệu:

```

const MN = 101;
d: array[0..MN] of integer;
c: array[0..MN] of integer;
a: array[1..MN,1..MN] of byte;

```

trong đó d là mảng chứa giới hạn sỏi trên dòng, c - trên cột, a là mảng hai chiều biểu diễn bảng chia lưới ô vuông,  $a[i,j] = 1$  - có viên sỏi đặt tại dòng  $i$ , cột  $j$ ;  $a[i,j] = 0$  - không có sỏi tại ô này. Ta thực hiện kỹ thuật hai pha như sau.

```

procedure XepSoi;
var j: integer;
begin
    fillchar(a,sizeof(a),0);
    d[0] := M+1; { dat linh canh }
    { Pha 1 } XepDong;
    { Pha 2 } for j := 1 to M do ChinhCot(j);
end;

```

Pha thứ nhất: Xếp tối đa sỏi vào mỗi dòng. Mỗi dòng  $i$  ta xếp liền nhau  $d[i]$  viên sỏi. Đồng thời ta sử dụng lại các biến mảng d và c với ý nghĩa sau đây:  $d[i]$  cho biết vị trí cột của viên sỏi cuối cùng trên dòng  $i$ .  $c[j]$  cho biết số sỏi còn có thể xếp thêm trên cột  $j$ . Dĩ nhiên, ta phải chỉnh lại các giá trị  $c[j]$  mỗi khi xếp thêm 1 viên sỏi vào cột này. Nếu  $c[j] < 0$  tức là ta cần bớt sỏi ở cột  $j$ . Thủ tục xếp dòng khi đó sẽ như sau.

```

procedure XepDong;
var i,j: integer;
begin
    for i := 1 to N do
        for j := 1 to d[i] do
            begin
                a[i,j] := 1; dec(c[j]);
            end;
    end;

```

Pha thứ hai: Sau khi xếp xong N dòng ta tiến hành chỉnh từng cột j có giá trị  $c[j] < 0$  đến khi nào  $c[j] = 0$ . Để chỉnh cột j theo phương pháp tham lam ta duyệt để chọn một dòng imin có chứa sỏi tại cột j và đầu phải  $d[imin]$  đạt giá trị nhỏ nhất. Sau đó ta chuyển viên sỏi trên dòng imin từ cột j sang cột  $d[imin]+1$  và chỉnh lại các giá trị  $c[j]$  và  $d[imin]$ . Để tìm dòng imin ta cần dùng phần tử  $d[0]$  với giá trị lớn nhất làm phần tử khởi đầu. Ta có thể cho giá trị này là M+1, vì mỗi dòng không thể có quá M viên sỏi. Bạn cần lưu ý rằng khi  $d[imin] = M$  tức là mọi viên sỏi cùng trên mỗi dòng đều chiếm vị trí tại cột M tức là hết chỗ để đặt sỏi.

```

procedure ChinhCot(j: integer);
begin
  while c[j] < 0 do GiamCot(j);
end;
procedure GiamCot(j: integer);
var i: integer;
begin
  i := DongMin(j);
  a[i,j] := 0; { Bo vien soi } inc(c[j]);
  if d[i] = M then exit;
  inc(d[i]); a[i,d[i]] := 1; { Dat 1 vien vao day }
  dec(c[d[i]]);
end;
function DongMin(j: integer): integer;
var i,imin: integer;
begin
  imin := 0;
  for i:=1 to N do
    if a[i,j]=1 then
      if d[i] < d[imin] then imin := i;
  DongMin := imin;
end;

```

Thí dụ dưới đây minh họa thuật toán với  $N = M = 4$ ;  $d = (3,2,1,2)$ ,  $c = (2,2,2,2)$ .

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

3  
2  
1  
2

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

3  
2  
1  
2

Cấu hình ban đầu

Sau Pha 1

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

3  
2  
1  
2

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

3  
2  
1  
2

-2 -1 1 2

-1 -2 1 2

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

3  
3  
2  
2

0 -2 0 2

Chỉnh cột 1

	3
	3
	3
	3
0 -2 0 2	2
0 -1 -1 2	2
0 0 -2 2	2

Chỉnh cột 2

	3
	4
	4
	4
0 0 -2 2	3
0 0 -1 1	3
0 0 0 0	3

Chỉnh cột 3

## Độ phức tạp

Ta cần chỉnh M cột. Mỗi cột ta cần lặp tối đa N lần, mỗi lần giảm được 1 viên sỏi trong cột. Để giảm 1 viên sỏi này ta phải duyệt N dòng để tìm imin. Tổng cộng ta cần cỡ  $MN^2$  thao tác.

## Chương trình C#

```
// C#
using System;
using System.Collections.Generic;
using System.Text;
namespace SangTao2 {
    class XepSoi {
        const int n = 4, m = 4;
        static byte[,] a = new byte[n+1,m+1];
        static int[] d = new int[n+1] {0,3,2,1,2};
        static int[] c = new int[m+1] {0,2,1,2,3};
        static void Main(string[] args) {
            Xep(); Show();
            Console.ReadLine();
        }
        static void Show(){
            for (int i = 1; i <= n; ++i){
                for (int j = 1; j <= m; ++j)
                    Console.Write(a[i, j]);
                Console.WriteLine();
            }
        }
        static void Xep(){
            Array.Clear(a,0,a.Length);
            d[0] = m+1;
        }
    }
}
```

```

        XepDong();
        for (int j = 1; j <= m; ++j) ChinhCot(j);
    }
    static void XepDong(){
        for (int i = 1; i <= n; ++i)
            for (int j = 1; j <= d[i];++j){
                a[i,j] = 1; --c[j];
            }
    }
    static void ChinhCot(int j) {
        while (c[j] < 0) GiamCot(j);
    }
    static void GiamCot(int j){
        int i = DongMin(j);
        a[i,j] = 0; // Bot 1 vien tai o (i,j)
        ++c[j];
        if (d[i]==m) return; // het cho dat tren dong i
        ++d[i]; a[i,d[i]] = 1; // Dat 1 vien vao o (i,d[i])
        --c[d[i]];
    }
    static int DongMin(int j){
        int imin = 0;
        for (int i = 1; i <= n; ++i)
            if (a[i,j]==1)
                if (d[i] < d[imin]) imin = i;
        return imin;
    }
} // XepSoi
} // SangTao2

```

## 4.5 Dãy các hoán vị

Dãy các hoán vị của  $N$  chữ cái HOA đầu tiên trong bảng chữ tiếng Anh được sắp theo trật tự từ điển tăng dần và viết liền nhau thành một dãy kí tự duy nhất. Hãy cho biết kí tự thứ  $M$  trong dãy tính từ  $1$  trở đi,  $2 \leq N \leq 10$ ,  $1 \leq M \leq N.N!$ . Thi dụ, với  $N=3$ , ta có dãy 6 hoán vị xếp theo trật tự từ điển là  $ABC$ ,  $ACB$ ,  $BAC$ ,  $BCA$ ,  $CAB$ ,  $CBA$ . Sau khi ghép chúng ta thu được dãy duy nhất gồm 18 kí tự  $ABCACBBACBCACABCBA$ . Kí tự thứ  $M=15$  trong dãy là:  $B$ .

### Thuật toán

Nếu ta viết mỗi hoán vị trên 1 dòng thì kí tự thứ  $M$  sẽ nằm trên dòng  $d = (M-1) \text{ div } N$  (tính từ dòng 0) và sẽ chiếm vị trí  $v = ((M-1) \text{ mod } N)+1$  (tính từ 1) trên dòng  $d$  đó. Như vậy ta cần xác định hoán vị trên dòng  $d$  rồi lấy kí tự nằm ở vị trí  $v$  làm kết quả.

Để xác định hoán vị  $(c_1, c_2, \dots, c_N)$  tại dòng  $d$  ta lần lượt tính các kí tự  $c_i$ ,  $i = 1..N$ . Ta phân hoạch các hoán vị theo nhóm. Nếu bỏ kí tự đầu tiên thì ta còn lại  $(N-1)!$  hoán vị, khi đó hoán vị tại dòng  $d$  sẽ rơi vào nhóm  $d \text{ div } (N-1)!$  và sẽ chiếm dòng  $d \text{ mod } (N-1)!$  trong nhóm đó. Tương tự, ta tính cho các kí tự thứ 2, 3, ...,  $N-1$ . Kí tự còn lại sẽ chiếm vị trí thứ  $N$ . Nếu biết nhóm  $d$  của kí tự thứ  $i$  trong hoán vị thì ta tính được chính kí tự đó như sau.

$d = 1$  ứng với kí tự thứ nhất trong số các kí tự chưa dùng,

$d = 2$  ứng với kí tự thứ hai trong số các kí tự chưa dùng,

...

Tổng quát,  $d$  ứng với kí tự thứ  $d$  trong số các kí tự chưa dùng.

Mỗi lần xác định được kí tự nào thì ta đánh dấu kí tự đó bằng thủ tục Mark.

Để tránh việc tính  $n!$  ta viết thủ tục ThuongDu( $z$ ,  $n$ ,  $q$ ,  $r$ ) cho ra thương  $q$  và dư  $r$  của phép chia số tự nhiên  $z$  cho  $n!$ , cụ thể là  $q = z \text{ div } n!$  và  $r = z \text{ mod } n!$ . Thủ tục này khá đơn giản. Ta có

$q_1 = z \text{ div } n; r_1 = z \text{ mod } n \Rightarrow z = q_1.n + r_1;$   
 $q_2 = q_1 \text{ div } (n-1); r_2 = q_1 \text{ mod } (n-1) \Rightarrow q_1 = q_2.(n-1) + r_2;$   
 ...  
 $q_{n-1} = q_{n-2} \text{ div } 2; r_{n-1} = q_{n-2} \text{ mod } 2 \Rightarrow q_{n-2} = q_{n-1}.2 + r_{n-1}.$   
 $q_n = q_{n-1} \text{ div } 1 = q_{n-1}; r_n = q_{n-1} \text{ mod } 1 = 0.$

Thay lần lượt các đại lượng của dòng dưới vào dòng trên ta thu được  $q = q_{n-1}$  và  $r = r_1 + n.r_2 + (n-1).r_3 + \dots + 3.r_{n-1} + 2.r_n$ . Nhận xét này cho phép ta xây dựng thủ tục theo kỹ thuật chia liên tiếp như sau.

```

procedure ThuongDu(z,n: longint;var q,r: longint);
  var c: longint;
begin
  r := 0; q := z; c := 1;
  while n > 1 do
    begin
      r := r + (q mod n)*c;
      q := q div n;
      c := n; n := n - 1;
    end;
end;

```

Thủ tục Test trong chương trình dưới đây tính mọi xuất hiện của các kí tự ( $M = 1..24^4$ ) trong dãy các hoán vị với  $N = 4$ .

## Chương trình Pascal

```

(* Pascal *)
uses crt;
const MN = 20; bl = #32;
var b: array[0..MN] of byte;
{ d = z div n! r = z mod n! }
procedure ThuongDu(z,n: longint;var q,r: longint);
Tự viết
{ Danh dau ki tu v thu k
  trong so cac ki tu chua dung }
procedure Mark(N,k,v: integer);
var i,d: integer;
begin
  d := 0;
  for i := 1 to N do
    if b[i] = 0 then
      begin
        d := d+1;
        if d = k then
          begin
            b[i] := v;
            exit;
          end;
      end;
  end;
{ Xac dinh ki tu thu M trong day cac hoan vi }
function Value(N: integer;M: longint): char;
var i,j,v: integer;
  th,du,d: longint;
begin
  fillchar(b,sizeof(b),0);
  d := (M-1) div N; { Dong chua ki tu M }

```

```

v := (M-1) mod N + 1; { vi tri cua M tren dong d }
{ xac dinh hoan vi tai dong d }
j := N-1;
for i := 1 to N-1 do
begin
    ThuongDu(d,j,th,du);
    Mark(N, th+1,i);
    j := j-1;
    d := du;
end;
Mark(N,1,N);
for i:=1 to N do
if b[i] = v then
begin
    Value := chr(ord('A') + i-1);
    exit;
end;
end;
procedure Test;
var N: integer;
    M: longint;
begin
    N := 4; writeln;
    for M := 1 to 24*N do
begin
    write(Value(N,M));
    if M mod N = 0 then
begin
        if readkey = #27 then halt else writeln;
    end;
end;
end;
BEGIN
    Test;
END.

```

## Chương trình C#

```

// C#
using System;
using System.Collections.Generic;
using System.Text;

namespace SangTao2 {
    class DayHoanVi {
        const int MN = 20;
        static int [] b = new int [MN+1];
        static void Main(string[] args){
            Test();
        }
        // q = z / n!; r = z % n!
        static void ThuongDu(long z, int n,
                            out long q, out long r ){
            q = z; r = 0;
            int c = 1;

```

```

        while (n > 1) {
            r += (q % n) * c;
            q /= n;
            c = n; --n;
        }
    }
    static void Mark(int n, long k, int v){
        int d = 0;
        for (int i = 1; i <= n; ++i)
            if (b[i]==0){
                ++d;
                if (d==k){ b[i] = v; return; }
            }
    }
    static char Value(int n, long m){
        Array.Clear(b, 0, b.Length);
        long d = (int) (m - 1) / n;
        // d - Dong chua ki tu can tim
        int v = (int)(m - 1) % n + 1;
        // v - vi tri cua ki tu tren dong d
        int j = n - 1;
        long th, du;
        for (int i = 1; i < n; ++i){
            ThuongDu(d, j, out th, out du);
            Mark(n, th + 1, i);
            d = du; --j;
        }
        Mark(n, 1, n);
        for (int i = 1; i <= n; ++i)
            if (b[i]==v) return (char)('A'+i-1);
        return (char)0;
    }
    // test voi n=4, m=1..n.n!
    static void Test(){
        int n = 4;
        int m4 = 24 * n;
        string s;
        for (long m = 1; m <= m4; ++m) {
            Console.WriteLine(Value(n, m));
            if (m % n == 0){
                s = Console.ReadLine();
                if (s == "stop") break;
            }
        }
    }
}
} // DayHoanVi
} // SangTao2

```

N	N!	N	N!
1	1	1	39916800
2	2	1	479001600
3	6	1	6227020800
4	24	2	87178291200

5	120	1	1307674368000
6	720	3	20922789888000
7	5040	1	355687428096000
8	40320	4	6402373705728000
9	362880	1	121645100408832000
1	362880	5	243290200817664000
0	0	1	
		6	
		1	
		7	
		1	
		8	
		1	
		9	
		2	
		0	

*Giai thừa của 20 số nguyên dương đầu tiên*

Với C# bạn có thể dùng kiểu int64 hoặc long với 64 bit (8 byte) biểu diễn số nguyên trong khoảng [-9.223.372.036.854.775.808, 9.223.372.036.854.775.807].

### Độ phức tạp

Thuật toán chỉ đòi hỏi  $N = 20$  phép chia các số nguyên có tối đa 20 chữ số và gọi thủ tục `Mark N` lần, mỗi lần gọi phải thực hiện phép duyệt trên dãy  $N$  phân tử. Tổng cộng là  $N^2$  phép toán, tức là cỡ 400 phép toán thay vì 2432902008176640000 phép toán nếu ta sinh lần lượt  $N!$  hoán vị bằng phương pháp vét cạn với  $N = 20$ .

### 4.6 Bộ bài

Trên bàn đặt một bộ bài gồm  $n-1$  quân bài mã số  $1, 2, \dots, n-1$ ,  $3 \leq n \leq 10000$ . Trọng tài chỉ định bạn lấy  $k$  quân bài. Sau đó trọng tài đưa ra một số tự nhiên  $s$ . Bạn cần cố gắng thực hiện ít nhất  $m$  thao tác thuộc một trong hai loại sau đây:

- Lấy thêm một quân bài từ trên bàn,

- Bỏ bớt một quân bài trên tay,

để cuối cùng đạt được hệ thức

$$t \bmod n = s \bmod n \quad (*)$$

trong đó  $t$  là tổng số hiệu các quân bài có trên tay bạn sau khi bạn đã hoàn tất  $m$  thao tác như trên.

Dữ liệu vào: file văn bản **BAI.INP**

Dòng đầu tiên: 3 số tự nhiên  $n$ ,  $k$  và  $s$ .

Từ dòng thứ hai trở đi:  $k$  số tự nhiên thể hiện mã số của các quân bài cần lấy lúc đầu.

Dữ liệu ra: Hiển thị trên màn hình

Dòng đầu tiên: số tự nhiên  $m$  cho biết số thao tác ít nhất cần thực hiện

Tiếp đến là  $m$  dòng, mỗi dòng là một thao tác lấy thêm hoặc bỏ bớt một quân bài  $v$ .  $v > 0$  cho biết cần lấy thêm (từ trên bàn) quân bài  $v$ ;  $v < 0$  cho biết cần bớt (từ trên tay) quân bài  $v$  để đạt được hệ thức (\*).

Thí dụ, với  $n = 8$ , trọng tài cho số  $s = 22$  và chỉ định bạn lấy  $k = 3$  quân bài là 2, 3 và 6.

Nếu bạn bỏ quân bài 2 và lấy quân bài 5 thì tổng  $t = 3 + 6 + 5 = 14$ . Khi đó  $t \bmod n = 14 \bmod 8 = 6 = s \bmod n = 22 \bmod 8$ .

Vậy một lời giải cho bộ dữ liệu này là

Thực hiện 2 thao tác: -2 và +5

BAI . INP	MÀN HÌNH
8 3 22	2
2 3 6	-2
	5

### Ý NGHĨA

Cho bộ bài gồm 8 quân. Lúc đầu trọng tài chỉ định bạn lấy  $k = 3$  quân bài mã số 2, 3 và 6. Ngoài ra trọng tài đưa ra số  $s = 22$ .

Sau đó bạn thực hiện 2 thao tác

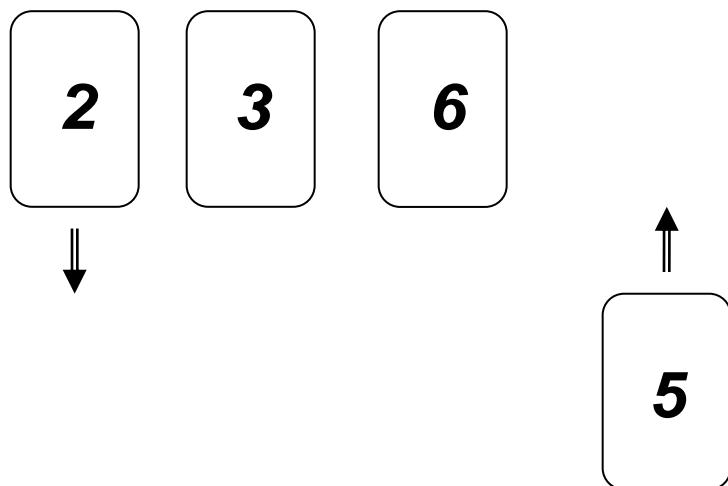
- bỏ quân bài 2

- lấy thêm quân bài 5.

Khi đó tổng số hiệu các quân bài có trên tay bạn sẽ là:

$$T = 3 + 6 + 5 = 14$$

$$T \bmod N = 14 \bmod 8 = 6 = s \bmod 8 = 22 \bmod 8.$$



$$n = 8; s = 22; \text{ Trên tay giữ } k = 3 \text{ quân bài } 2, 3, 6.$$

Lời giải: Bỏ quân bài 2, lấy thêm quân bài 5.

$$t = 3+6+5 = 14,$$

$$t \bmod 8 = 14 \bmod 8 = 6 = s \bmod 8 = 22 \bmod 8.$$

## Thuật toán

Ta sẽ chứng minh rằng với không quá 2 thao tác (+) lấy thêm / (-) bỏ bớt một quân bài ta có thể đạt được hệ thức (\*).

Trước hết ta nhắc lại các phép toán đồng dư. Với số nguyên dương  $n$  cho trước ta xét tập các số dư trong phép chia một số tự nhiên  $x$  cho  $n$ ,  $x \bmod n$ ,  $Z_n = \{0, 1, 2, \dots, n-1\}$ . Trên  $Z_n$  các phép toán cộng và nhân được thực hiện như bình thường sau đó lấy kết quả chia dư cho  $n$ . Phép toán lấy số đối của số  $x$  cho ta  $n-x$ . Phép trừ  $x-y$  được đổi thành phép cộng  $x$  với số đối của  $y$ . Ta có

Cộng:  $(x + y) \bmod n$

Nhân:  $x^*y \text{ mod } n$

Lấy số đối của  $x$ :  $n - x$

Trừ:  $(x + (n-y)) \text{ mod } n$ .

Hãy tưởng tượng các số của  $Z_n$  là  $0, 1, \dots, n-1$  được bố trí trên một vòng tròn như trên mặt đồng hồ. Để tính tổng  $x+y$  ta xuất phát từ  $x$  và di chuyển  $y$  bước theo chiều kim đồng hồ (còn gọi là *di chuyển xuôi*), mỗi bước ta chuyển qua một số. Kết quả sẽ là điểm dừng cuối cùng. Để tính hiệu  $x - y$  ta cũng xuất phát từ  $x$  và di chuyển  $y$  bước theo chiều ngược lại (*di chuyển ngược*). Để ý rằng, trên vòng tròn gồm  $n$  số, di chuyển xuôi  $y$  bước sẽ cho cùng kết quả như di chuyển ngược  $(n-y)$  bước, và ngược lại, di chuyển ngược  $y$  bước sẽ tương đương như di chuyển xuôi  $(n-y)$  bước. Điều này có nghĩa là muốn thêm  $b$  đơn vị cho đại lượng  $t$  ta có thể bớt  $(n-b)$  đơn vị và ngược lại, muốn bớt  $b$  đơn vị từ đại lượng  $t$  ta có thể thêm cho  $t$   $(n-b)$  đơn vị. Ta cũng để ý rằng số hiệu của mọi quân bài đều nhỏ thua  $n$  và mỗi quân bài hoặc là có trên tay người chơi, hoặc là nằm trên bàn. Vì lẽ trên, đôi khi người ta nói tính toán theo *đồng dư* (modulo) chính là tính toán trên *vòng tròn*.

Bạn cũng cần ghi nhớ tính chất sau đây:

Với mọi số tự nhiên  $x, y$  và  $n, n > 0$  và với mọi phép toán số học  $\theta \in \{+, -, *\}$  ta luôn có

$$(x \theta y) \text{ mod } n = ((x \text{ mod } n) \theta (y \text{ mod } n)) \text{ mod } n$$

Công thức trên cho ta quy tắc dễ hiểu sau đây: Khi tính trị của các biểu thức số học chỉ chứa các phép toán cộng, trừ và nhân trong  $Z_n$  ta có thể thực hiện phép lấy số dư *mod n* trên các hạng tử và các kết quả trung gian.

Vì lũy thừa nguyên dương tương đương với phép nhân liên tiếp, ta suy ra hệ quả sau:

$$a^k \text{ mod } n = (a \text{ mod } n)^k \text{ mod } n$$

Sau khi đã biết các giá trị input là  $n, k, s$  và số hiệu các quân bài cần lấy lên tay, ta gán trị cho mảng  $a[1..n-1]$  như sau:  $a[i] = 1$  cho biết quân bài  $i$  có trên tay, ngược lại,  $a[i] = 0$  cho biết quân bài  $i$  còn nằm trên bàn. Ví dụ  $a[1..6] = [1, 1, 0, 0, 1, 0]$  có số hiệu 2, 3 và 6 nên  $a = (0,1,1,0,0,1,0)$  ứng với  $a[2] = a[3] = a[6] = 1$ , các giá trị  $a[i]$  còn lại đều bằng 0.

Trước hết ta tính tổng số hiệu của các quân bài có trong tay lúc đầu và đặt trong biến  $t$ . Sau đó ta tính  $t := t \text{ mod } n$  và  $s := s \text{ mod } n$ . Với thí dụ đã cho, trọng tài yêu cầu ta lấy 3 quân bài có số hiệu 2, 3 và 6 nên  $a = (0,1,1,0,0,1,0)$  ứng với  $a[2] = a[3] = a[6] = 1$ , các giá trị  $a[i]$  còn lại đều bằng 0.

$$t = 2+3+6 = 11, \text{ do đó } t \text{ mod } n = t \text{ mod } 8 = 3$$

$$\text{và } s \text{ mod } 8 = 22 \text{ mod } 8 = 6$$

Tức là  $t = 3$  và  $s = 6$ .

Giả sử  $t \geq s$ , ta đặt  $b = t - s$  và xét các trường hợp loại trừ nhau sau đây:

1.  $b = 0$ : Hệ thức (\*) đã thỏa, ta không phải làm gì. Ta thông báo  $m = 0$ , trong đó  $m$  là số thao tác  $+/-$  cần thực hiện.

2. Quân bài  $b$  có trên tay, tức là  $a[b] = 1$ : Ta chỉ việc bỏ quân bài này xuống, khi đó tổng  $t$  sẽ giảm  $b$  đơn vị theo mod  $n$ .

3. Quân bài  $(n-b)$  có trên bàn, tức là  $a[n-b] = 0$ : Ta chỉ việc lấy thêm quân bài này. Khi đó tổng  $t$  sẽ được thêm  $(n-b)$  đơn vị theo mod  $n$ , điều này tương đương với việc giảm tổng  $t$  đi  $b$  đơn vị theo mod  $n$ .

4. Nếu không xảy ra các trường hợp 1, 2 và 3 như trên, tức là  $b \neq 0, a[b] = 0, a[n-b] = 1$ , ta tiến hành như sau:

Tìm hai quân bài  $u$  và  $v$  thỏa các điều kiện sau

Quân bài  $u$  có trên tay,  $a[u] = 1$ ,

Quân bài  $v$  có trên bàn,  $a[v] = 0$ ,

$u = (k*b) \text{ mod } n; v = ((k-1)*b) \text{ mod } n$ ,  $k$  là một số tự nhiên. Điều này có nghĩa là  $u$  lớn hơn  $v$  b đơn vị theo mod  $n$ .

Nếu tìm được hai quân bài  $u$  và  $v$  như trên ta sẽ thực hiện hai thao tác: bỏ quân bài  $u$  ( $-u$ ) và lấy thêm quân bài  $v$  ( $+v$ ). Khi đó tổng  $t$  sẽ được giảm một lượng  $b$  theo mod  $n$ . Thật vậy,

$$(u - v) \text{ mod } n = (k*b - (k-1)*b) \text{ mod } n = b.$$

Trường hợp  $t < s$  ta phải thêm  $b = s - t$  đơn vị cho cho  $t$ . Việc này tương đương với giảm  $t$  bớt  $(n-b)$  đơn vị. Đặt  $b = n-b$  rồi lặp lại thủ tục trên sẽ cho ta kết quả tương ứng.

Ta chứng minh rằng nếu gặp tình huống 4 thì bao giờ cũng có thể tìm được hai số nguyên  $a$  và  $n$  như đã mô tả. Trên hai ngàn năm trước nhà toán học Cố Hy Lạp Diophantus đã phát biểu và chứng minh định lý sau:

**Định lý** Cho phương trình  $ax \equiv b \pmod{n}$ , với các hệ số  $a, b, n$  là các số tự nhiên,  $n > 0$ . Gọi  $d$  là ước chung lớn nhất của  $a$  và  $n$ ,  $d = (a, n)$ . Khi đó

a) Nếu  $d$  không là ước của  $b$  thì phương trình vô nghiệm.

b) Nếu  $b = kd$  thì phương trình có đúng  $d$  nghiệm trong tập  $\mathbb{Z}_n$ . Các nghiệm này có dạng  $(x + i(n/d)) \pmod{n}$ , trong đó  $x$  là một nghiệm tùy ý,  $i = 0, 1, 2, \dots, (d-1)$ .

Phương trình  $ax \equiv b \pmod{n}$  được người đời sau gọi là phương trình Diophantus.

Chứng minh

Nếu  $x$  là nghiệm của phương trình  $ax \equiv b \pmod{n}$  thì  $ax \equiv b$  có cùng số dư theo  $\pmod{n}$  nên hiệu của chúng sẽ chia hết cho  $n$ ,  $ax - b = kn$ , hay  $ax - kn = b$ . Mặt khác, do  $d = (a, n)$  nên  $a$  và  $n$  đều chia hết cho  $d$  và do đó hiệu  $ax - kn$  cũng chia hết cho  $d$ , thế tức là  $b$  phải chia hết cho  $d$ . Giả sử  $b = md$  tức là phương trình có nghiệm. Gọi  $x$  là nghiệm nguyên không âm nhỏ nhất của phương trình trên, ta dễ dàng kiểm tra được rằng  $x + i(n/d)$ ,  $i = 0, 1, \dots, (d-1)$  cũng là nghiệm của phương trình đó. Thật vậy, ta để ý rằng nếu  $d$  là ước chung lớn nhất của  $a$  và  $n$  thì  $an/d$  chính là bội chung nhỏ nhất của chúng, nghĩa là  $an/d$  chia hết cho  $a$  và  $n$ . Ta có

$$\begin{aligned} a(x+i(n/d)) \pmod{n} &= ((ax \pmod{n}) + (i(an)/d) \pmod{n}) \pmod{n} \\ &= (b \pmod{n} + 0) \pmod{n} = b \pmod{n}. \end{aligned}$$

Ta chứng minh xong.

Thí dụ 1. Giải phương trình sau

$$6x \equiv 21 \pmod{9}$$

Phương trình trên tương đương với phương trình sau:

$$6x \equiv 3 \pmod{9}$$

Ta có  $d = (6, 9) = 3$ . Vì 3 là ước của  $3$  nên phương trình đã cho có 3 nghiệm. Để thấy  $x = 2$  là một nghiệm của phương trình. Vậy các nghiệm của phương trình dưới dạng tổng quát là

$$x + i(9/3) = 2 + 3i, i = 0, 1, 2$$

Cụ thể là  $x_1 = 2$ ,  $x_2 = 5$  và  $x_3 = 8$  là 3 nghiệm trong tập  $\mathbb{Z}_9 = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ .

Thí dụ 2. Giải phương trình

$$4x \equiv 5 \pmod{12}$$

Ta có,  $d = (4, 12) = 4$  không phải là ước của 5. Phương trình vô nghiệm.

Trở lại bài toán trên, khi gặp tình huống 4 ta có  $a[b] = 0$  và  $a[n-b] = 1$ . Xét phương trình  $bx \equiv n-b \pmod{n}$ . Vì  $1 \leq b < n$  nên  $1 \leq n-b < n$  và do đó  $(n-b) \pmod{n} = n-b$ , phương trình đã cho có thể viết lại là  $bx \equiv n-b \pmod{n}$ .

Theo tính chất: ước chung lớn nhất của hai số tự nhiên  $(a, b)$  sẽ không đổi nếu ta thay số lớn nhất trong hai số đó bằng hiệu của nó với số thứ hai, đặt  $d = (b, n)$ , ta có  $d = (b, n-b)$ , tức là  $n-b$  chia hết cho  $d$ , do đó phương trình  $bx \equiv n-b \pmod{n}$  luôn có nghiệm. Từ nhận xét này suy ra rằng vòng lặp **repeat** trong đoạn trình dưới đây luôn kết thúc.

```

u := b;
repeat
    v := u;
    u := (u+b) mod n;
until a[u] = 1;

```

Thật vậy, sau  $k$  lần lặp ta thu được  $u = kb$  do phương trình  $bx \equiv n-b \pmod{n}$  có nghiệm nên sẽ tồn tại một giá trị  $k$  để  $u = kb \equiv n-b \pmod{n}$ . Do  $a[n-b] = 1$  nên tối đa sau  $k$  lần lặp thì vòng lặp phải kết thúc và ta sẽ thu được  $u = kb \equiv n-b \pmod{n}$ . Vì  $v$  mang giá trị sát trước của  $u$  nên  $v = (k-1)b \pmod{n}$ .

Ta có thuật toán sau đây

1. Đọc dữ liệu vào các biến n, k và s
2. Khởi trị cho mảng a[1..n-1] với a[i] = 1 nếu quân bài i có trên tay, a[i] = 0 nếu quân bài i còn trên bàn.
3. Tính t = tổng số hiệu các quân bài có trên tay.
4. Tính t := t mod n; s := s mod n.
5. Nếu t ≥ s: đặt b := t - s; ngược lại đặt b := n - (s - t).

Ý nghĩa: cần giảm b đơn vị từ tổng t để đạt hệ thức

$$t \bmod n = s \bmod n \quad (*)$$

6. Xét các trường hợp loại trừ nhau sau đây

6.1 b = 0: Đặt m = 0; Thông báo: “Không làm gì”; Stop.

6.2 a[b] = 1 (Quân bài b có trên tay):

Thông báo: “Thực hiện m = 1 thao tác – b: Bỏ quân bài b”; Stop.

6.3 a[b] = 0 và a[n-b] = 0 (Quân bài b không có trên tay, quân bài (n-b) có trên bàn):

Thông báo: “Thực hiện m = 1 thao tác +(n-b): Lấy quân bài (n-b)”; Stop.

6.4 a[b] = 0 và a[n-b] = 1: (Quân bài b không có trên tay, quân bài (n-b) không có trên bàn)

6.4.1 Tính u và v

```
u := b;
repeat
    v := u;
    u := (u+b) mod n;
until a[u] = 1;
```

6.4.2 Thông báo: “Thực hiện m = 2 thao tác

-u: Bỏ quân bài u

+v: Lấy quân bài v.”

6.4.3 Stop

Từ chứng minh trên ta rút ra độ phức tạp của thuật toán là O(n) vì trong trường hợp xấu nhất ta duyệt 1 lần mảng a chứa n-1 phần tử.

Tổ chức dữ liệu:

```
const mn = 10000; { Max n }
bl = #32; { Dấu cách }
nl = #13#10; { New line: xuống dòng }
ESC = #27;
fn = 'bai.inp';
type mil = array[0..mn] of integer;
var a: mil; { Dánh dấu các quân bài }
n, k : integer; { n-1: số lượng quân bài }
{ k: số lượng các quân bài trên tay }
t , s: longint; { t: tổng số hiệu các quân bài trên tay }
{ s: số đối chứng của trọng tài }
f: text; { input file }
```

Thủ tục đọc dữ liệu: Mở input file, đọc các giá trị n, k và s, đọc số hiệu và đánh dấu k quân bài được chọn, tính tổng t của chúng }

```
procedure Doc;
var i,j: integer;
begin
  assign(f,fn); reset(f);
  read(f,n,k,s);
  t := 0; fillchar(a,sizeof(a),0);
  for i := 1 to k do
  begin
    read(f,j); a[j] := 1; t := t+j;
  end;
  close(f);
```

```

end;
Thủ tục xử lí.

procedure XuLi;
var b,u,v: integer;
begin
  t := t mod n; s := s mod n;
  if t >= s then b := t-s else b := n-(s-t);
  if (b = 0) then
    begin
      Ket(0,0,0);
      exit
    end;
  if (a[b] = 1) then
    begin { Quan bai b co tren tay }
      Ket(1,-b,0); { bo xuong }
      exit
    end;
  if (a[n-b] = 0) then
    begin { Quan bai n-b co tren ban }
      Ket(1,n-b,0); { Lay len }
      exit
    end;
  { Quan bai b khong co tren tay
    Quan bai n-b khong co tren ban }
  u := b;
  repeat
    v := u;
    u := (u+b) mod n;
  until (a[u] = 1);
  Ket(2,-u,v); { bo u, lay v }
end;

```

Thủ tục **Ket(m,u,v)** thông báo kết quả ứng với các trường hợp:

m = 1: Bỏ bớt hoặc lấy thêm 1 quân bài u;

m = 2: Bỏ quân bài u, lấy quân bài v.

```

procedure Ket(m,u,v: integer);
begin
  case m of
    0: write(nl,'Khong lam gi',nl);
    1: begin
        write(nl,' Thuc hien 1 thao tac: ');
        if (u > 0) then write('+',u,nl)
        else write(u,nl);
      end;
    2: begin
        write(nl,' Thuc hien 2 thao tac: ');
        if (u > 0) then write('+',u,bl)
        else write(u,bl);
        if (v > 0) then write('+',v,nl)
        else write(v,nl);
      end;
  end;
end;

```

Độ phức tạp tính toán: N.

Chương trình C# dưới đây hiển thị kết quả trên màn hình.

## Chương trình C#

```
// C#
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
namespace SangTao2 {
    class BoBai {
        const string fn = "bai.inp";
        const int MN = 20;
        static int[] a;
        static int n; // so luong quan bai
        static int k; // so luong quan bai tren tay
        static int s; // so cho truoc trong khoang [1,n];
        static int t;
        static void Main(string[] args){
            Doc(); XuLi();
            Console.ReadLine();
        }
        static void Doc(){
            int[] c =
                Array.ConvertAll((File.ReadAllText(fn)).Split(
                    new char[] { '\0', '\n', '\t', '\r', ' ' },
                    StringSplitOptions.RemoveEmptyEntries),
                    new Converter<String, int>(int.Parse));
            n = c[0]; // so luong quan bai
            k = c[1]; // so luong quan bai tren tay
            s = c[2]; // so cho truoc
            a = new int[n + 1];
            Array.Clear(a, 0, a.Length);
            t = 0;
            for (int i = 3; i < c.Length; ++i){
                a[c[i]] = 1; t += c[i];
            };
        }
        static void XuLi(){
            t %= n; s %= n;
            int b = (t >= s) ? t - s : n - (s - t);
            if (b == 0) { Ket(0, 0, 0); return; }
            // Sua t: giam b don vi hoac tang n-b don vi
            if (a[b] == 1) { // quan b co tren tay
                Ket(1, -b, 0); // bo quan b
                return;
            }
            if (a[n - b] == 0) { // quan n-b co tren ban
                Ket(1, n - b, 0); // lay quan n-b
                return;
            }
            // Quan b tren ban, quan n-b tren tay
            int u = b, v = 0;
            do { v = u; u = (u + b) % n; } while(a[u] == 0);
            Ket(2, -u, v); // bo quan u, lay quan v
        }
        static void Ket(int c, int u, int v) {
            switch (c){

```

```

        case 0: Console.WriteLine(c); break;
        case 1: Console.WriteLine(c + ":" + u); break;
        case 2: Console.WriteLine(c + ":" + u + " ", "+v");
                break;
    }
}
} // Bo Bai
} // SangTao2

```

## 4.7 Thuận thế

Dijkstra E.

Cho hoán vị  $a = (a_1, a_2, \dots, a_N)$  của  $N$  số nguyên dương đầu tiên  $1, 2, \dots, N$ . Một thuận thế của  $a$  là dãy  $b = (b_1, b_2, \dots, b_N)$  trong đó  $b_i$  là số lượng các phần tử nhỏ hơn  $a_i$  và đứng trước  $a_i$ ,  $b_i = |\{a_j | a_j < a_i, j < i\}|$ . Biết trước  $N$ ,  $2 \leq N \leq 1000$ .

a) Cho một hoán vị  $a$ , tính thuận thế  $b$  của  $a$ .

b) Cho thuận thế  $b$ , tìm hoán vị  $a$ .

c) Mọi thuận thế đều có phần tử đầu tiên (trái nhất) là 0 nên ta có thể bỏ phần tử này. Ngoài ra, nếu trong thuận thế còn có phần tử 0 nữa ta bỏ thêm 1 phần tử 0 để thu được một dãy có  $M = N-1$  hoặc  $M = N-2$  phần tử và gọi dãy này là thuận thế thu gọn  $c$ . Cho một thuận thế thu gọn. Hãy tìm hoán vị nhỏ nhất theo trật tự từ điển sinh ra thuận thế thu gọn này.

Thí dụ, với  $N = 5$ ,

a) Cho  $a = (2, 5, 1, 4, 3)$  ta tính được  $b = (0, 1, 0, 2, 2)$ ,

b) Cho  $b = (0, 1, 0, 2, 2)$  ta tìm được  $a = (2, 5, 1, 4, 3)$ ,

c) Cho thuận thế thu gọn  $c = (1, 2, 2)$ ,  $N = 5$ , ta tìm được  $a = (2, 3, 5, 4, 1)$ .

Để ý rằng hai hoán vị  $(2, 5, 1, 4, 3)$  và  $(2, 3, 5, 4, 1)$  cùng sinh ra thuận thế thu gọn  $(1, 2, 2)$ , nhưng hoán vị  $(2, 3, 5, 4, 1)$  nhỏ hơn.

Dữ liệu vào: text file **THUANTHE.INP**

Dòng đầu tiên:  $N$

Từ dòng thứ hai trở đi:  $N$  phần tử của hoán vị  $a$ .

Dòng tiếp theo:  $M$

Trên các dòng tiếp theo:  $M$  phần tử của thuận thế thu gọn.

Dữ liệu trên cùng một dòng cách nhau qua dấu cách.

Dữ liệu ra: Hiển thị trên màn hình theo trật tự sau:

Câu a: Cho hoán vị  $a$ , tìm thuận thế  $b$ .

Câu b: Cho thuận thế  $b$ , tìm hoán vị  $a$ .

Câu c: Cho thuận thế thu gọn  $c$  tìm hoán vị nhỏ nhất  $a$ .

Thuật toán

Việc xác định thuận thế  $b$  từ hoán vị  $a$  là dễ dàng. Hai câu b và c là hơi khó. Chúng ta sẽ sử dụng kỹ thuật đối xứng để trình bày một thuật toán do Dijkstra đề xuất. Theo thuật toán này thì thủ tục cho câu a và b là đối xứng nhau. Thuật toán tiến hành xử lý tại chỗ, nghĩa là không sử dụng mảng phụ mà trực tiếp biến đổi hoán vị  $a$  thành thuận thế lưu luôn trong  $a$  và ngược lại.

Trước hết ta nhận xét rằng với hoán vị đơn vị  $e = (1, 2, \dots, N)$  thì có đúng  $e_i$  phần tử không lớn hơn  $e_i$  và không đứng sau phần tử  $e_i$ ,  $i = 1..N$ . Vậy, nếu trong một hoán vị  $a$  mà ta thấy một phần tử  $a_j \geq a_i$  và  $j \leq i$  thì ta khẳng định rằng chỉ còn đúng  $a_j - 1$  phần tử không lớn hơn  $a_j$  và không đứng sau phần tử  $a_j$ .

Ta khai báo các biến  $x, y, a$  là các mảng để chứa các hoán vị và thuận thế:

```

const MN = 1000;
type mil = array[0..MN] of integer;
var x,y,a: mil;

```

Thủ tục biến đổi hoán vị  $a$  sang thuận thế  $a$  khi đó sẽ như sau:

```

procedure HoanViThuanThe(var a: mil;n: integer);
var i,j: integer;
begin
  for i := n downto 1 do
    for j:=1 to i do
      if (a[j] >= a[i]) then dec(a[j]);
end;

```

Để thu được hoán vị a từ thuận thế a ta chỉ cần viết thủ tục xử lý theo chiều ngược lại. Hai thủ tục như vậy gọi là *đối xứng nhau*.

```

procedure ThuanTheHoanVi(var a: mil;n: integer);
var i,j: integer;
begin
  for i := 1 to n do
    for j := i downto 1 do
      if (a[j] >= a[i]) then inc(a[j]);
end;

```

Hai thủ tục này đều có độ phức tạp  $N^2$ .

Câu c được giải như sau. trước hết thêm một số 0 vào đầu trái của dữ liệu vào a. Sau đó xét hiệu N-M. Nếu N-M=1 thì chúng tôi thuận thế thu gọn a chỉ khuyết một số 0. Ta chỉ việc gọi thủ tục **ThuanTheHoanVi(a,N)** là thu được kết quả. Trường hợp N-M=2 thì ta phải bù thêm một số 0 nữa vào một vị trí nào đó trong a. Ta lần lượt đặt số 0 này vào đầu phải (vị trí N) rồi chuyển dần nó về đầu trái, mỗi lần một vị trí và gọi thủ tục **ThuanTheHoanVi** để sinh ra một dãy a[1..N] sau đó kiểm tra xem dãy này có phải là hoán vị của 1..N hay không. Nếu đúng, ta dừng thuật toán và cho ra kết quả. Để kiểm tra một dãy a[1..N] có phải là một hoán vị của 1..N ta sử dụng một mảng d[1..N] đánh dấu xem mỗi phần tử a[i] có xuất hiện đúng 1 lần hay không. Tuy nhiên trước đó ta phải kiểm tra điều kiện  $1 \leq a[i] \leq N$  để đảm bảo rằng a[i] nằm trong giới hạn của chỉ số mảng d.

```

procedure ThuanTheThuGon(var a: mil; n,m: integer);
var b: mil;
  i: integer;
begin
  move(a[1],a[2],m*sizeof(integer));
  a[1] := 0; inc(m);
  if (n = m) then
    begin
      ThuanTheHoanVi(a,n);
      exit;
    end;
  b := a;
  for i := n downto 2 do
    begin
      { Them 0 tai vi tri i }
      a := b;
      move(a[i],a[i+1],(n-i)*sizeof(integer));
      a[i] := 0;
      ThuanTheHoanVi(a,n);
      if LaHoanVi(a,n) then exit;
    end;
  end;
function LaHoanVi(var a: mil; n: integer): Boolean;
var d: mil;
  i: integer;
begin
  LaHoanVi := false;
  fillchar(d,sizeof(d),0);
  for i := 1 to n do

```

```

begin
  if (a[i] < 1) or (a[i] > n) then exit;
  if (d[a[i]] = 1) then exit;
  d[a[i]] := 1;
end;
LaHoanVi := true;
end;

```

## Chương trình C#

```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System;
namespace SangTao2 {
    class ThuanThe {
        const string fn = "thuanthe.inp";
        const int MN = 1001;
        static int[] a = new int[MN];
        static int[] c = new int[MN];
        static int n; // 1..n
        static int m; // so luong phan tu trong thuan the thu gon
        static void Main(string[] args){
            Doc();
            Console.WriteLine("\n n = " + n + " m = " + m);
            Console.WriteLine("\n Cho Hoan vi: ");
            Show(a, n);
            HoanViThuanThe(a);
            Console.WriteLine("\n Tim Thuan the: ");
            Show(a, n);
            Console.WriteLine("\n Cho Thuan the: ");
            Show(a, n);
            ThuanTheHoanVi(a);
            Console.WriteLine("\n Tim Hoan vi: ");
            Show(a, n);
            Console.WriteLine("\n Cho Thuan the Thu gon: ");
            Show(c,m);
            ThuanTheThuGon(c);
            Console.WriteLine("\n Tim Hoan vi: ");
            Show(c, n);
            Console.ReadLine();
        }
        static void Doc(){
            int[] v =
                Array.ConvertAll((File.ReadAllText(fn)).Split(
                    new char[] { '\0', '\n', '\t', '\r', ' ' },
                    StringSplitOptions.RemoveEmptyEntries),
                    new Converter<String, int>(int.Parse));
            int i = 0;
            n = v[i++];
            for (int j = 1; j <= n; ++j) a[j] = v[i++];
            m = v[i++];
            for (int j = 1; j <= m; ++j) c[j] = v[i++];
        }
        static void HoanViThuanThe(int[] a){
            for (int i = n; i > 0; --i)

```

```

        for (int j = 1; j <= i; ++j)
            if (a[j] >= a[i]) --a[j];
    }
    static void ThuanTheHoanVi(int[] a){
        for (int i = 1; i <= n; ++i)
            for (int j = i; j > 0; --j)
                if (a[j] >= a[i]) ++a[j];
    }
    static void ThuanTheThuGon(int[] c){
        Array.Copy(c, 1, c, 2, m);
        c[1] = 0; ++m;
        if (m == n) { ThuanTheHoanVi(c); return; }
        int [] b = new int [n+1];
        Array.Copy(c,1,b,1,m);
        for (int i = n; i >= 2 ; --i){
            Array.Copy(b,1,c,1,i-1);
            Array.Copy(b, i, c, i + 1, m - i + 1);
            c[i] = 0;
            ThuanTheHoanVi(c);
            if (LaHoanVi(c)) return;
        }
    }
    static bool LaHoanVi(int[] c){
        int[] d = new int[n + 1];
        Array.Clear(d,0,d.Length);
        for (int i = 1; i <= n; ++i)
        {
            if (c[i] < 1 || c[i] > n) return false;
            if (d[c[i]] > 0) return false;
            d[c[i]] = 1;
        }
        return true;
    }
    static void Show(int[] a, int n){
        for (int i = 1; i <= n; ++i)
            Console.Write(a[i] + " ");
    }
} // ThuanThe
} // SangTao2

```

## Độ phức tạp

Thủ tục move(a,b,M) copy m byte từ mảng a sang mảng b. Thủ tục ThuanTheThuGon có độ phức tạp  $N^3$  vì nó gọi thủ tục ThuanTheHoanVi N lần. Hàm kiểm tra một dãy a[1..N] có phải là hoán vị đồi hồi N thao tác và sử dụng một mảng phụ d để đánh dấu các phần tử đã xuất hiện.

## 4.8 Các nhà khoa học

*Olimpic Quốc tế*

Trong một hội nghị khoa học có n nhà khoa học (KH) tổ chức thư giãn dưới hình thức sau. Họ đặt một máy tính trong căn phòng hẹp, ngoài máy ra chỉ có thể chứa thêm 1 người, màn hình máy tính hiện số 0. Sau đó mỗi nhà khoa học buộc phải thực hiện n thao tác loại 1 và n thao tác loại 2 đan xen nhau, trong đó thao tác đầu tiên phải là loại 1.

Thao tác loại 1: Đọc	Thao tác loại 2: Ghi
----------------------	----------------------

- |  |   |
|--|---|
| <ul style="list-style-type: none"> <li>♦ Vào phòng;</li> <li>♦ Đọc và ghi nhớ số trên màn hình;</li> <li>♦ Ra khỏi phòng.</li> </ul> | <ul style="list-style-type: none"> <li>♦ Vào phòng;</li> <li>♦ Lấy số nhớ trong đầu cộng thêm 1 và hiển thị kết quả trên màn hình;</li> <li>♦ Ra khỏi phòng.</li> </ul> |
|--|---|

Khi hiển thị số mới trên màn hình thì số cũ trên màn hình tự động bị xóa và người thực hiện thao tác loại 2 cũng quên luôn số đã nhớ.

Cho trước các giá trị  $n$  và  $m$ . Hãy bố trí một lịch thực hiện để các nhà khoa học hoàn thành trọn vẹn cuộc chơi theo đúng yêu cầu và số cuối cùng hiển thị trên màn hình là  $m$ .

Dữ liệu vào: Tệp văn bản **KH.INP** chứa 2 số  $n$  và  $m$  trên một dòng cách nhau qua dấu cách.

Dữ liệu ra: Tệp văn bản **KH.OUT** chứa một lịch thực hiện gồm một dãy tuần tự các dòng lệnh thuộc một trong hai dạng sau:

Dạng thứ nhất gồm hai số tự nhiên ghi cách nhau qua dấu cách,  $i$  cho biết nhà khoa học  $i$  thực hiện thao tác  $t$ ;  $i \in \{1, 2, \dots, n\}$ ;  $t \in \{1, 2\}$ .

Dạng thứ hai gồm 4 số tự nhiên ghi cách nhau qua dấu cách,  $i t_1 t_2 k$  cho biết nhà khoa học  $i$  thực hiện  $k$  lần liên tiếp các thao tác  $t_1$  và  $t_2$  đan xen nhau;  $i \in \{1, 2, \dots, n\}$ ;  $t \in \{1, 2\}$ ;  $k > 0$ .

Nếu không có cách nào bố trí lịch thì ghi duy nhất một số 0.

Thí dụ,

KH.INP	KH.OUT	Ý nghĩa
<b>3 4</b>	<b>1 1</b> <b>3 1 2 3</b> <b>1 2</b> <b>2 1</b> <b>1 1 2 2</b> <b>2 2</b> <b>2 1 2 2</b>	<p>Có 3 nhà khoa học tham gia trò chơi thư giãn với nhiệm vụ sinh ra kết quả trên màn hình (MH) là số 4. Lịch thực hiện sẽ như sau:</p> <p>Người số 1: Đọc. MH = 0. Đầu(1) = 0.</p> <p>Người số 3: (Đọc; Ghi) 3 lần. MH = 3.</p> <p>Người số 1: Ghi. MH = Đầu(1)+1 = 0+1 = 1.</p> <p>Người số 2: Đọc. Đầu(2) = 1, MH = 1.</p> <p>Người số 1: (Đọc; Ghi) 2 lần. MH = 3.</p> <p>Người số 2 Ghi. MH = Đầu(2)+1 = 1+1 = 2.</p> <p>Người số 2 (Đọc; Ghi) 2 lần. MH = 2+2 = 4.</p> <p>Chú thích: Đầu(<math>i</math>) - số nhớ trong đầu nhà khoa học thứ <math>i</math>.</p>

## Thuật toán

Ta sẽ chỉ ra rằng với mọi  $n \geq 2$  và mọi  $m$  trong khoảng  $2..n^2$  luôn luôn có một lịch thỏa yêu cầu của đầu bài (ta gọi là *lịch hợp lệ*) để màn hình (MH) đạt giá trị  $m$ .

Sau khi mở tệp KH.INP và đọc hai giá trị  $n$ ,  $m$  ta tiến hành xếp lịch và ghi dàn kết quả vào tệp KH.OUT mở sẵn. Trước hết ta nhận xét rằng nếu một người thực hiện liên tiếp  $k$  lần một cặp thao tác (Đọc - Ghi, viết tắt là **ĐG**), tức là sau  $2k$  dòng lệnh

```

i 1
i 2
...
i 1
i 2

```

thì giá trị của MH được tăng thêm  $k$  đơn vị. Dãy  $2k$  lệnh trên có thể viết gộp lại thành một lệnh 4 thành phần là **i 1 2 k**.

Ta tính  $k = m \text{ div } n$  và  $r = m \text{ mod } n$ , ta có  $m = k.n+r$ ,  $0 \leq r < n$ , với ý nghĩa là để đạt được giá trị  $m$  trên MH thi phải có  $k$  người thực hiện đầy đủ và liên tiếp  $n$  cặp thao tác ĐG, ngoài ra phải có ít nhất một người thực hiện thêm  $r$  cặp thao tác ĐG. Do yêu cầu mỗi người buộc phải thực hiện  $n$  thao tác Đ và  $n$  thao tác G, một lẽ tự nhiên, ta phải sử dụng 2 người ghi nhận giá trị hiện có trên MH để khi cần sẽ trả lại giá trị đó (dĩ nhiên là phải cộng thêm 1) nhằm đảm bảo cho các thao tác cần thiết được thực hiện liên tục. Xét 3 trường hợp sau đây.

**Trường hợp 1:**  $r = 0$ , tức là  $m = k.n$ . Ta cần  $k$  người thực hiện liên tiếp  $n$  cặp thao tác ĐG. Tuy nhiên, do những người khác cũng phải thực hiện đầy đủ  $n$  cặp thao tác ĐG cho mỗi người, nên ta chia các thao tác thành hai loại là các *thao tác vô ích* là những thao tác đến một thời điểm nào đó sẽ có một thao tác khác đặt lại giá trị cho MH. Các thao tác còn lại được gọi là *thao tác có ích*. Như vậy trường hợp này cần có  $k$  người thực hiện tổng cộng  $m = k.n$  cặp thao tác có ích và mọi thao tác còn lại là *vô ích*. Lịch khi đó sẽ như sau.

- 1 1 - Người số 1 Đọc và nhớ số 0 trên màn hình. Đầu(1) = 0.
- i 1 2 n ; i = k+1..n - Những người còn lại (mang mã số k+1..n) ĐG n lần vô ích.
- 1 2 - Người số 1 ghi số 1 lên màn hình (có ích), MH = Đầu(1)+1 = 0 + 1 = 1.
- 1 1 2 n-1 - Người số 1 ĐG nốt n-1 lần có ích, MH = n.
- i 1 2 n ; i = 2..k - Những người số 2..k ĐG n lần có ích, MH = n+(k-1).n = k.n = m.

**Trường hợp 2:**  $r \neq 0$  và  $k > 0$ . Ta có  $m = k.n+r$ ,  $0 < r < n$ ,  $k > 0$ . Trường hợp này cần  $n-1$  người thực hiện các thao tác vô ích, 1 người thực hiện  $r$  cặp thao tác ĐG có ích và cũng chính người đó phải thực hiện  $(n-r)$  cặp thao tác vô ích. Ta sử dụng 2 người, số 1 và số 2 như sau.

- 1 1 - Người số 1 Đọc và nhớ số 0. Đầu(1) = 0.
- i 1 2 n ; i = k+2..n - Những người mang mã số từ k+2 đến n ĐG n lần vô ích.
- 1 2 - Người số 1 Ghi 1 lên MH. MH = Đầu(1)+1 = 0 + 1 = 1.
- 2 1 - Người số 2 Đọc và nhớ số 1. Đầu(2) = 1.
- 1 1 2 n-r - Người số 1 ĐG n-r lần vô ích.

Tiếp đến là những thao tác có ích:

- 2 2 - Người số 2 Ghi số 2 lên MH, MH = Đầu(2)+1 = 1 + 1 = 2.
- 1 1 2 r-1 - Người số 1 ĐG nốt r-1 lần có ích, MH = 2+(r-1).
- 2 1 2 n-1 - Người số 2 ĐG nốt n-1 lần có ích, MH = 2+(r-1)+(n-1) = n+r.
- i 1 2 n ; i = 3..k+1 - Những người số 3..k+1 ĐG n lần có ích, MH = n+r+(k-1).n = k.n+r.

**Trường hợp 3:**  $r \neq 0$  và  $k = 0$ , do đó  $m = r \geq 2$ . Trường hợp này cần  $n-1$  người thực hiện các thao tác vô ích, 1 người thực hiện  $r$  cặp thao tác ĐG có ích và cũng chính người đó phải thực hiện  $(n-r)$  cặp thao tác vô ích. Ta sử dụng 2 người, số 1 và số 2 như sau.

- 1 1 - Người số 1 Đọc và nhớ số 0 trên MH. Đầu(1) = 0.
- i 1 2 n ; i = 3..n - Những người từ số 3 đến n ĐG n lần vô ích.
- 2 1 2 n-1 - Người số 2 ĐG n-1 lần vô ích.
- 1 2 - Người số 1 Ghi số 1 lên MH. MH = Đầu(1)+1 = 0+1 = 1.
- 2 1 - Người số 2 Đọc số 1 trên MH. Đầu(2) = 1.
- 1 1 2 n-r+1 - Người số 1 ĐG n-r+1 lần vô ích.
- 2 2 - Người số 2 Ghi số 2 lên MH. MH = Đầu(2)+1 = 1+1 = 2.
- 1 1 2 r-2 - Người số 1 ĐG r-2 lần có ích, MH = 2 + (r-2) = r.

Phần dưới đây trình bày cấu trúc dữ liệu và các thủ tục đọc và xếp lịch. Hai thủ tục phụ trợ Lenh2 và Lenh4 dùng để ghi một lệnh 2 tham biến dạng  $i$  t và lệnh 4 tham biến dạng  $i$   $t_1$   $t_2$  k vào output file g KH.OUT. Trong các chú thích dưới đây  $d[i]$  là số trong đầu nhà KH thứ  $i$ ,  $t[i]$  là số thao tác ĐG nhà KH  $i$  đã thực hiện, MH – màn hình, kí hiệu MH = x cho biết ta không quan tâm đến giá trị của MH vì sớm muộn giá trị này sẽ bị xóa.

```
uses crt;
const
```

```

fn = 'kh.inp'; gn = 'kh.out';
bl = #32; nl = #13#10; mn = 100;
{ bl - dấu cách; nl - xuống dòng }
type
  mil = array[0..mn+1] of integer;
var
f,g: text;
n, m, mh: integer;
d,t: mil;
{ mh - Màn hình }
  d[i] - số nho trong dau,
  t[i] - con dem lenh cua nguoi i }
procedure Doc;
begin
  assign(f,fn); reset(f);
  read(f,n,m); close(f);
end;
procedure Lenth2(i,tt: integer);
begin writeln(g,i,bl,tt); end;
procedure Lenth4(i,tt1,tt2,k: integer);
begin if k > 0 then writeln(g,i,bl,tt1,bl,tt2,bl,k); end;
procedure XepLich;
var k,r,i: integer;
begin
  assign(g,gn); rewrite(g);
  if (n < 2) or (m < 2) or (m > n*n) then
  begin writeln(g,0); close(g); exit; end;
  k := m div n; { k nguoi co ich }
  r := m mod n; { va r thao tac co ich }
  if (r = 0) then
  begin
    Lenth2(1,1); {MH=0,d[1]=0,t[1]=1}
    for i := k+1 to n do Lenth4(i,1,2,n);
    {MH=x,t[i]=2n,i=k+1..n}
    Lenth2(1,2); {MH=1}
    Lenth4(1,1,2,n-1); {MH=n,t[1]=2n}
    for i := 2 to k do Lenth4(i,1,2,n);
    { MH = n+(k-1)n=kn=m,t[i]=2n,i=2..k}
    close(g); exit;
  end;
  { r > 0 }
  if k > 0 then
  begin { r,k > 0 }
    Lenth2(1,1); {MH=0,d[1]=0,t[1]=1}
    { Bo nhung nguoi vo ich }
    for i:=k+2 to n do Lenth4(i,1,2,n);
    {MH=x,t[i]=2n,i=k+2..n}
    Lenth2(1,2); {1 Ghi;MH=1,t[1]=2}
    Lenth2(2,1); {2 Doc;MH=1;d[2]=1,t[2]=1}
    { Cac thao tac vo ich cua 1 }
    Lenth4(1,1,2,n-r); {MH=x,t[1]=2+2(n-r)=2(n-r+1)}
    { Tu day la cac thao tac co ich }
    Lenth2(2,2); {MH=2,t[2]=2}
    Lenth4(1,1,2,r-1);
    {MH=2+r-1,t[1]=2(n-r+1)+2(r-1)=2n}
    Lenth4(2,1,2,n-1);{MH=2+r-1+n-1=n+r,t[2]=2n}
    for i := 3 to k+1 do Lenth4(i,1,2,n);
  end;
end;

```

```

{MH=n+r+(k-1) n=kn+r=m, t[i]=2n, i=3..k+1}
close(g); exit;
end;
{ k = 0, r > 0 }
Lenh2(1,1); {1 Doc,d[1]=0,t[1]=1}
{ Bo nhung nguoi vo ich }
for i:=3 to n do Lanh4(i,1,2,n);{MH=x,t[i]=2n,i=3..n}
{ n-1 thao tac vo ich cua 2 }
Lenh4(2,1,2,n-1);{MH=x,t[2]=2(n-1)}
Lenh2(1,2); {1 Ghi,MH=1,t[1]=2}
Lenh2(2,1); {2 Doc,MH=1,d[2]=1,t[2]=2n-2+1=2n-1}
{ Cac thao tac vo ich cua 1 }
Lenh4(1,1,2,n-r+1);{MH=x,t[1]=2+2(n-r+1)=2(n-r+2)}
Lenh2(2,2); {MH=2,t[2]=2n}
Lenh4(1,1,2,r-2);{MH=2+r-2=r=m,t[1]=2(n-r+2)+2(r-2)=2n}
close(g);
end;

```

Bạn có thể viết thêm thủ tục test để kiểm tra xem lịch đã xếp và ghi trong tệp KH.OUT có thỏa các yêu cầu của đầu bài hay không. Thủ tục sử dụng các mảng sau đây. Mảng d[1..n], d[i] là số nhớ trong đầu người số i. Mảng t[1..n], t[i] là số lần người thứ i thực hiện các thao tác Đọc (1), Ghi (2). Do thao tác Đọc phải thực hiện trước và hai thao tác Đọc - Ghi phải đan xen nên thời điểm sát trước thao tác Đọc của người thứ i ta phải có t[i] là *số chẵn* và thời điểm sát trước thao tác Ghi phải có t[i] là *số lẻ*. Mỗi lần đọc 1 dòng lệnh thủ tục phải xét xem dòng lệnh đó chứa 2 hoặc 4 số. Thủ tục phải thực hiện các kiểm tra sau đây.

Kiểm tra lệnh dạng i v:  $1 \leq i \leq n$ ,  $v = 1$  hoặc  $2$ . Nếu  $v = 1$  thì  $t[i]$  phải là số chẵn, nếu  $v = 2$  thì  $t[i]$  phải lẻ.

Kiểm tra lệnh dạng i v1 v2 k: tương tự như trên.

Thực hiện lệnh i v: Nếu  $v = 1$  (thao tác đọc) thì gán  $d[i] := MH$ ; ngược lại, nếu  $v = 2$  (ghi) thì gán  $MH := d[i] + 1$ . Trong cả hai trường hợp đều tăng con đếm lệnh  $t[i]$  thêm 1 đơn vị.

Sau khi đọc xong tệp KH.OUT phải duyệt lại các con đếm để đảm bảo rằng  $d[i] = 2.n$  với mọi  $i = 1..n$ . Cuối cùng kiểm tra xem  $MH = m$ ?

```

procedure DocLenh(var i,t1,t2,k,v: integer);
begin
  read(g,i,t1); v := 2;
  if seekeoln(g) then exit;
  readln(g,t2,k); v := 4;
end;
procedure XemLenh(i,t1,t2,k,KieuLenh: integer);
begin
  if KieuLenh = 2 then writeln(i,bl,t1)
  else writeln(i,bl,t1,bl,t2,bl,k);
end;
function Lanh(i,c: integer): Boolean;
begin
  Lenh := false;
  if (i < 1) or (i > n) then exit;
  case c of
    1: begin
      if odd(t[i]) then exit;
      inc(t[i]); d[i] := mh;
    end;
    2: begin
      if not(odd(t[i])) then exit;
      inc(t[i]); mh := d[i]+1;
    end;
  end;
end;

```

```

    else exit;
  end;
  Lenth := true;
end;
function KiemTraLenh(i,t1,t2,k,v: integer): Boolean;
  var j: integer;
begin
  if v = 2 then KiemTraLenh := Lenth(i,t1)
  else
  begin
    KiemTraLenh := false;
    for j := 1 to k do
      begin
        if not Lenth(i,t1) then exit;
        if not Lenth(i,t2) then exit;
      end;
    KiemTraLenh := true;
  end;
end;
procedure Test;
  var i,t1,t2,k,v,n2: integer;
begin
  mh := 0;
  fillchar(d,sizeof(d),0);
  fillchar(t,sizeof(t),0);
  assign(g,gn); reset(g);
  while not Seekeof(g) do
  begin
    DocLenh(i,t1,t2,k,v);
    XemLenh(i,t1,t2,k,v);
    if not KiemTraLenh(i,t1,t2,k,v) then
    begin
      writeln('Sai ');
      exit;
    end;
  end;
  n2 := n+n;
  for i:=1 to n do
    if (t[i] <> n2) then
    begin
      writeln('Sai ');
      exit;
    end;
  if (mh <> m) then
  begin
    writeln('Sai ');
    exit;
  end;
  writeln('Dung');
  close(g);
end;

```

## Chương trình C#

```

// C#
using System;

```

```

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

namespace SangTao2 {
    class KhoaHoc {
        const string fn = "KH.INP";
        const string gn = "KH.OUT";
        const int MN = 1002;
        static int[] d = new int[MN]; // So nho trong dau
        static int[] t = new int[MN]; // con dem thao tac
        static int n; // nha khoa hoc
        static int m; // man hinh cuoi
        static int mh;
        static StreamWriter g;
        static StreamReader f;

        static void Main(string[] args) {
            Run();
        }
        static void Run() { // Kiem tra tren 1 file
            Doc();
            Console.WriteLine("n = " + n + "    m = " + m);
            XepLich();
            Console.WriteLine("Output file " + gn);
            Console.WriteLine(File.ReadAllText(gn));
            Console.WriteLine("Now Testing... ");
            Test();
            Console.ReadLine();
        }
        // Kiem tra file output KH.OUT
        static void Test(){
            f = File.OpenText(gn);
            Array.Clear(d, 0, d.Length);
            Array.Clear(t, 0, t.Length);
            mh = 0;
            int i, t1, t2, k, v;
            while (DocLenh(out i, out t1, out t2, out k, out v)){
                XemLenh(i, t1, t2, k, v);
                if (!KiemTraLenh(i, t1, t2, k, v)){
                    Console.WriteLine("SAI LENH");
                    return;
                }
            }
            f.Close();
            for (int j = 1, nn = n + n; j <= n; ++j)
                if (t[j] != nn){
                    Console.WriteLine("SAI SO THAO TAC" + t[j]);
                    return;
                }
            if (mh != m) Console.WriteLine("SAI KET QUA MH");
            else Console.WriteLine(" LAP LICH DUNG ");
        }
        static bool KTLenh2(int i, int tt){
            switch (tt){
                case 1: if (t[i] % 2 != 0) return false;
                ++t[i]; d[i] = mh; return true;
                case 2: if (t[i] % 2 == 0) return false;
            }
        }
    }
}

```

```

        ++t[i]; mh = d[i] + 1; return true;
    default: return false;
}
}
static bool KiemTraLenh(int i, int t1, int t2,
                        int k, int v){
    if (i < 1 || i > n) return false;
    if (v == 2) return KTLenh2(i, t1);
    for (int j = 1; j <= k; ++j){
        if (!KTLenh2(i, t1)) return false;
        if (!KTLenh2(i, t2)) return false;
    }
    return true;
}
static void XemLenh(int i, int t1, int t2, int k, int v)
{
    if (v == 2) Console.WriteLine(i + " " + t1);
    else Console.WriteLine(i+" "+t1+" "+t2+" "+k);
}
static bool DocLenh(out int i, out int t1,
                    out int t2, out int k, out int v){
    i = t1 = t2 = k = v = 0;
    if (f.EndOfStream) return false;
    int[] c = Array.ConvertAll(f.ReadLine()).Split(
        new char[] { '\t', ' ' },
        StringSplitOptions.RemoveEmptyEntries),
        new Converter<String, int>(int.Parse));
    foreach (int x in c) ++v;
    Console.Write(" v = " + v + ": ");
    if (v != 2 && v != 4) return false;
    i = c[0]; t1 = c[1];
    if (v == 4) { t2 = c[2]; k = c[3]; }
    return true;
}
static void Doc(){
    int[] v =
    Array.ConvertAll(File.ReadAllText(fn)).Split(
        new char[] { '\0', '\n', '\t', '\r', ' ' },
        StringSplitOptions.RemoveEmptyEntries),
        new Converter<String, int>(int.Parse));
    n = v[0]; // nha khoa hoc
    m = v[1]; // Gia tri man hinh
}
static void XepLich(){
    g = File.CreateText(gn);
    if (n < 2 || m < 2 || m > n * n){
        g.WriteLine("0");
        g.Close();
        return;
    }
    int k = m / n;
    int r = m % n;
    Console.WriteLine("k = " + k + "    r = " + r);
    if (r == 0){
        Lenh2(1, 1);
        for (int i = k + 1; i <= n; ++i)
            Lenh4(i, 1, 2, n);
    }
}

```

```

        Lenth2(1, 2);
        Lenth4(1, 1, 2, n - 1);
        for (int i = 2; i <= k; ++i) Lenth4(i, 1, 2, n);
        g.Close();
        return;
    }
    if (k > 0){
        Lenth2(1, 1); // 1 Doc
        // Bo nhung nguoi vo ich
        for (int i = k + 2; i <= n; ++i)
            Lenth4(i, 1, 2, n);
        Lenth2(1, 2); // 1 Ghi
        Lenth2(2, 1); // 2 Doc
        Lenth4(1,1,2,n-r); //cac thao tac vo ich cua 1
        // Tu day la cac thao tac co ich
        Lenth2(2, 2); // 2 Ghi
        Lenth4(1, 1, 2, r - 1); // 1 DG r-1 lan
        Lenth4(2, 1, 2, n - 1); // 2 DG n-1 lan
        for (int i = 3, k1 = k + 1; i <= k1; ++i)
            Lenth4(i, 1, 2, n);
        g.Close();
        return;
    }
    // k = 0
    Lenth2(1, 1); // 1 Doc. Tu 3..n vo ich
    for (int i = 3; i <= n; ++i) Lenth4(i, 1, 2, n);
    Lenth4(2, 1, 2, n - 1);
    Lenth2(1, 2); // 1 Ghi
    Lenth2(2, 1); // 2 Doc
    Lenth4(1,1,2,n-r+1); // Cac thao tac vo ich cua 1
    Lenth2(2, 2); // 2 Ghi co ich
    Lenth4(1, 1, 2, r - 2); // 1 Ghi not co ich
    g.Close();
}
static void Lenth2(int i, int t)
{ g.WriteLine(i + " " + t); }
static void Lenth4(int i, int t1, int t2, int k) {
    if (k > 0)
        g.WriteLine(i + " " + t1 + " " + t2 + " " + k);
}
} // KhoaHoc
} // SangTao2

```

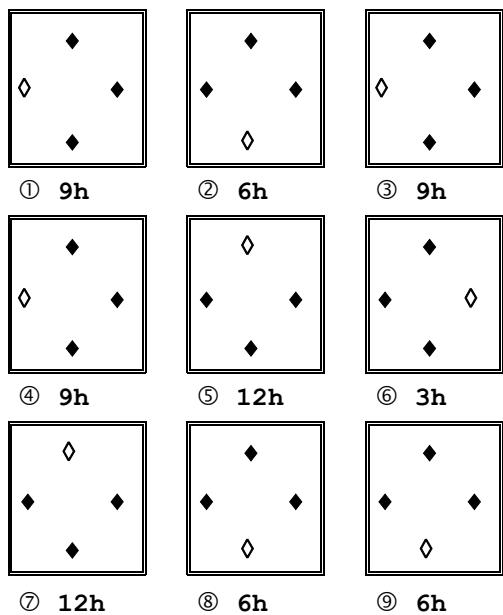
## Độ phức tạp

Thuật toán phát sinh và ghi vào file kết quả tối đa  $2.n^2$  dòng lệnh.

## 4.9 Chín chiếc đồng hồ

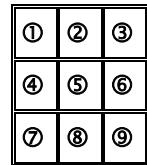
*Olimpic Quốc tế*

Có 9 chiếc đồng hồ điện tử treo trên một bảng theo sơ đồ  $3 \times 3$ . Các đồng hồ được mã số từ 1 đến 9 theo trật tự từ trái qua phải, từ hàng trên xuống hàng dưới. Mỗi đồng hồ có 4 điểm chỉ giờ ứng với các giờ 12, 3, 6 và 9. Giờ hiện tro của mỗi đồng hồ ứng với điểm sáng Ø. Để điều khiển các đồng hồ người ta sử dụng 9 phím với các chức năng được mô tả như trong hình vẽ. Khi nhấn vào một phím thì có 4 đồng hồ đồng loạt nhảy điểm sáng đi  $90^\circ$  theo chiều kim đồng hồ để cộng thêm 3 giờ tính từ giờ hiện tro. Thí dụ, khi nhấn phím 1 thì 4 đồng hồ 1, 2, 4 và 5 sẽ được chỉnh giờ theo luật trên. Biết giờ hiện tại của các đồng hồ. Hãy xác định một dãy ngắn nhất các phím cần nhấn để các đồng hồ đồng loạt tròn 12 giờ.



9 đồng hồ, ◊ giờ đang trỏ

Sơ đồ bố trí  
9 đồng hồ



9 phím  
diều khiển



Dữ liệu vào: tệp văn bản DONGHO.INP gồm 12 dòng  
3 dòng đầu tiên gồm 9 số trỏ giờ tại thời điểm đầu bố trí theo ma trận  $3 \times 3$ .  
Đòng thứ  $i$  trong số 9 dòng tiếp theo,  $i = 1, 2, 3, 4$  ghi 4 số tự nhiên thể hiện mã số của 4 đồng hồ sẽ nhảy điểm sáng  $90^\circ$  khi ta nhấn phím  $i$ .

Các số trên cùng dòng cách nhau qua dấu cách.

Dữ liệu ra: Tệp văn bản DONGHO.OUT gồm hai thông tin:

- Bài toán có nghiệm (ghi số 1) hoặc vô nghiệm (ghi số 0).
- Dãy phím ngắn nhất cần nhấn cho trường hợp có nghiệm.

DONGHO.INP	DONGHO.OUT
9 6 9	1
9 12 3	1 2 4 4
12 6 6	
1 2 4 5	
2 3 5 6	
4 5 7 8	
5 6 8 9	
1 2 3 5	
1 4 5 7	
3 5 6 9	
5 7 8 9	

2	5	6	8	
---	---	---	---	--

Ý nghĩa: Nhấn lần lượt các phím 1, 2, 4,  
4  
cả 9 đồng hồ đều đồng loạt trôi 12 giờ.

## Thuật toán

Ta nhận thấy rằng do các đồng hồ hoạt động độc lập với nhau nên dãy phím cần nhấn là giao hoán, nghĩa là có thể liệt kê dãy phím đó theo trật tự tùy ý. Thí dụ, nhấn các phím 1, 2, 3 sẽ mang lại kết quả như khi nhấn dãy phím 2, 1, 3 hoặc theo bất kì hoán vị nào của 3 phím đó. Ngoài ra, nếu một phím được nhấn đến một bội lần của 4 thì điểm sáng trôi giờ sẽ quay lại đúng vị trí ban đầu, do đó ta không dại gì mà nhấn một phím quá 3 lần. Từ hai nhận xét trên ta thấy rằng có thể dùng kỹ thuật vét cạn, cụ thể là xét các tổ hợp dãy phím  $p[1..9]$ ,  $p[i] = 0..3$  biểu thị số lần bấm phím i. Với mỗi tổ hợp p ta tính xem các đồng hồ được cập nhật ra sao. Nếu cả 9 đồng hồ đều nhảy về thời điểm 12 giờ thì ta chọn phương án có số lần bấm phím ít nhất trong số các tổ hợp ứng viên.

Dưới đây là một vài chi tiết sử dụng trong chương trình. Ta khởi tạo sẵn mảng hai chiều mô tả chức năng của các phím điều khiển, **phim[i, j]** cho biết khi nhấn phím i thì đồng hồ j sẽ được chỉnh.

```
var
  phim: array[1..9, 1..4] of
    integer;
```

Sau khi đọc dữ liệu ứng với thí dụ cụ thể thì mảng **phim** sẽ được gán trị như sau:

```
phim = ((1,2,4,5),
         (2,3,5,6),
         (4,5,7,8),
         (5,6,8,9),
         (1,2,3,5),
         (1,4,5,7),
         (3,5,6,9),
         (5,7,8,9),
         (2,5,6,8));
```

Bạn cũng nên mô tả sẵn một kiểu mảng 9 phần tử để biểu thị các phím và các đồng hồ.

```
type mil = array[1..9] of
  integer;
var
  dongHo: mil;
  f: text;
  imin, imax: longint;
```

Bạn có thể sử dụng 9 vòng **for** lồng nhau ứng với 9 phím, mỗi vòng biến thiên từ 0 đến 3 ứng với số lần nhấn phím.

Biến ts dùng để tính tổng số lần nhấn phím. Để thấy, mỗi phím được nhấn tối đa 3 lần, vậy 9 phím sẽ được nhấn tối đa  $9 \cdot 3 = 27$  lần. Ta lấy 28 làm giá trị khởi đầu cho việc tính  $tsmin$  - tổng số lần nhấn phím ít nhất. Ta cũng nên chuyển các số trên mặt đồng hồ là (12,3,6,9) sang các số hệ 4 là (0,1,2,3) để cho tiện tính toán. Hàm **Sum(p)** tính tổng 9 phần tử của mảng p - đó chính là tổng số lần nhấn phím của phương án p. Hàm **KiemTra(q)** thực hiện việc kiểm tra xem 9 đồng hồ có cùng trôi 12h hay không,  $q[i]$  là số lần đồng hồ i được cập nhật khi thực hiện phương án p.

1	2	3
4	5	6
7	8	9



1	2	3
4	5	6
7	8	9



1	2	3
4	5	6
7	8	9



Chức năng của các phím điều khiển

Bạn cũng có thể sử dụng hệ 4 để xử lí như sau: Các phương án nhán 9 phím biến thiên từ  $(0,0,0,0,0,0,0,0,0)$  đến  $(3,3,3,3,3,3,3,3,3)$  ứng với  $imin = 0$  và  $imax = 4^9 - 1 = 2^{18} - 1 = (1 \text{ shl } 18) - 1 = 262143$ . Ta cho i biến thiên từ imin đến imax. Với mỗi i ta xây dựng phương án nhán phím bằng cách gọi thủ tục Split(i,p) chuyển số i sang dạng biểu diễn hệ 4 để ghi vào mảng p, trong đó  $p[j]$  sẽ là số lần nhán phím j. Biết p ta dễ dàng cập nhật các đồng hồ.

Chương trình dưới đây cài đặt 2 phương án vét cạn, Run1 – chín vòng for lồng nhau và Run2 - tính toán theo hệ 4.

```
(* Pascal *)
(* Dong ho *)
const bl = #32; fn = 'DONGHO.INP'; gn = 'DONGHO.OUT';
type mil = array[1..9] of integer;
var
  dongHo,kq: mil;
  f,g: text;
  imin, imax: longint;
  s, tsmin: integer;
var
phim: array[1..9,1..4] of integer;
procedure Split(x: longint; var a: mil);
var i: integer;
begin
  for i := 1 to 9 do
  begin
    a[i] := x mod 4;
    x := x div 4;
  end;
end;
procedure Doc;
var i,j: integer;
begin
  assign(f,fn); reset(f);
  for i:=1 to 9 do read(f,dongHo[i]);
  for i:=1 to 9 do
    for j:=1 to 4 do read(f,phim[i,j]);
  close(f);
end;
procedure Ghi;
var i,j: integer;
begin
  assign(g,gn); rewrite(g);
  if tsmin = 28 then writeln(g,0) { vo nghiem }
  else
  begin { co nghiem }
    writeln(g,1);
    for i := 1 to 9 do
      for j := 1 to c[i] do write(g,i,bl);
    writeln(g);
  end;
  close(g);
end;
procedure Init;
var i: integer;
begin
  for i:=1 to 9 do dongHo[i] := (dongHo[i] div 3) mod 4;
  imin := 0;
```

```

imax := 1;
imax := (imax shl 18 - 1);
end;
function KiemTra(var q: mil): Boolean;
var i: integer;
begin
  KiemTra := false;
  for i:=1 to 9 do
    if (dongHo[i]+q[i]) mod 4 <> 0 then exit;
  KiemTra := true;
end;
function Sum(var q: mil): integer;
var i,s: integer;
begin
  s := 0;
  for i:=1 to 9 do s := s+q[i];
  Sum := s;
end;
procedure XuLiFor;
var j,k,ts: integer;
  q,p: mil; { p[i] - so lan nhan phim i }
  i,ikq: longint;
begin
  tsmin := 28; { = 3*9+1 }
  for p[1] := 0 to 3 do
    for p[2] := 0 to 3 do
      for p[3] := 0 to 3 do
        for p[4] := 0 to 3 do
          for p[5] := 0 to 3 do
            for p[6] := 0 to 3 do
              for p[7] := 0 to 3 do
                for p[8] := 0 to 3 do
                  for p[9] := 0 to 3 do
                    begin
                      fillchar(q,sizeof(q),0);
                      for j := 1 to 9 do
                        begin
                          for k := 1 to 4 do
                            inc(q[phim[j,k]],p[j]);
                        end;
                      if KiemTra(q) then
                        begin
                          ts := Sum(p);
                          if ts < tsmin then
                            begin
                              tsmin := ts;
                              kq := p;
                            end;
                        end;
                    end;
                end;
  end;
procedure XuLi;
var j,k,ts: integer;
  q,p: mil;
  i,ikq: longint;
begin
  tsmin := 28; { = 3*9+1 }

```

```

for i:=imin to imax do
begin
  Split(i,p); { bam phim j p[j] lan }
  fillchar(q,sizeof(q),0);
  for j := 1 to 9 do
  begin
    for k := 1 to 4 do
      inc(q[phim[j,k]],p[j]);
  end;
  if KiemTra(q) then
  begin
    ts := Sum(p);
    if ts < tsmin then
    begin
      tsmin := ts;
      ikq := i;
    end;
  end;
  Split(ikq,kq);
end;
procedure Run1;
begin
  Doc; Init;
  XuLiFor; Ghi;
end;
procedure Run2;
begin
  Doc; Init;
  XuLi; Ghi;
end;
BEGIN
  Run1;
END.

```

## Chương trình C#

```

// C#
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
namespace SangTao2 {
  class DongHo {
    const string fn = "DONGHO.INP";
    const string gn = "DONGHO.OUT";
    const int cc = 9;
    static int[,] phim = new int [9,4];
    static int[] dongHo = new int[cc];
    static int [] kq = new int [cc];
    static int smin = 0; // Tong so lan nhan phim
    static void Main(string[] args) {
      Run2();
      Console.ReadLine();
    }
    static void Run1(){

```

```

        Doc(); Init(); XuLi(); Ghi(); XemKq();
    }
    static void Run2(){
        Doc(); Init();
        XuLiFor(); Ghi(); XemKq();
    }
    static void XuLiFor(){
        int [] p = new int [cc];// 9 phim
        int[] dh = new int[cc];//so lan nhay kim cua 9 DH
        int s = 0;
        smin = 28;
        for (p[0]=0; p[0] < 4; ++p[0])
            for (p[1]=0; p[1] < 4; ++p[1])
                for (p[2]=0; p[2] < 4; ++p[2])
                    for (p[3]=0; p[3] < 4; ++p[3])
                        for (p[4]=0; p[4] < 4; ++p[4])
                            for (p[5]=0; p[5] < 4; ++p[5])
                                for (p[6]=0; p[6] < 4; ++p[6])
                                    for (p[7]=0; p[7] < 4; ++p[7])
                                        for (p[8]=0; p[8] < 4; ++p[8]){
                                            Array.Clear(dh,0,dh.Length);
                                            for (int j = 0; j < cc; ++j)
                                                //phim j nhan p[j] lan
                                                for (int k = 0; k < 4; ++k)
                                                    // 4 DH chuyen kim
                                                    dh[phim[j, k]] += p[j];
                                            if (KiemTra(dh)){
                                                s = Sum(p);
                                                if (s < smin){
                                                    smin = s;
                                                    p.CopyTo(kq,0);
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        static int Sum(int []p){ // Tong so lan nhan phim
            int s = 0;
            for (int i = 0; i < cc; ++i) s += p[i];
            return s;
        }
        static void Doc() { // Doc du lieu tu input file
            int[] v =
                Array.ConvertAll((File.ReadAllText(fn)).Split(
                    new char[] { '\0', '\n', '\t', '\r', ' ' },
                    StringSplitOptions.RemoveEmptyEntries),
                    new Converter<String, int>(int.Parse));
            int k = 0;
            for (int j = 0; j < cc; ++j) dongHo[j] = v[k++];
            for (int i = 0; i < cc; ++i)
                for (int j = 0; j < 4; ++j)
                    phim[i, j] = v[k++];
        }
        static void Init() { // Khoi tri
            for (int j = 0; j < cc; ++j)
                dongHo[j] = (dongHo[j] / 3) % 4;
            for (int i = 0; i < cc; ++i)
                for (int k = 0; k < 4; ++k)

```

```

        --phim[i, k];
    }
    static void XuLi(){
        int imax = ((int)1 << 18) - 1;
        int[] p = new int[cc]; smin = 28;
        int[] q = new int[cc];
        for (int i = 0; i <= imax; ++i){
            int s = Split(i, p);
            Array.Clear(q, 0, q.Length);
            for (int j = 0; j < cc; ++j)
                for (int k = 0; k < 4; ++k)
                    q[phim[j, k]] += p[j];
            if (KiemTra(q))
                if (s < smin){ smin = s; p.CopyTo(kq, 0); }
        }
    }
    static void Ghi(){
        StreamWriter g = File.CreateText(gn);
        if (smin < 28) { // co nghiem
            g.WriteLine(1);
            for (int i = 0; i < cc; ++i)
                for (int j = 1; j <= c[i]; ++j)
                    g.Write((i + 1) + " ");
        }
        else g.WriteLine(0); // vo nghiem
        g.Close();
    }
    static bool KiemTra(int[] q)// Kiem tra ca 9 DH tro ve 0?
    {
        for (int i = 0; i < cc; ++i)
            if ((dongHo[i] + q[i]) % 4 > 0) return false;
        return true;
    }
    // Tach x thanh cac chu so he 4 va tinh tong
    static int Split(int x, int[] p){
        int s = 0;
        for (int i = 0; i < cc; ++i)
        { p[i] = x % 4; s += p[i]; x /= 4; }
        return s;
    }
    static void XemKq(){
        Console.WriteLine("\n Input file " + fn);
        Console.WriteLine(File.ReadAllText(fn));
        Console.WriteLine("\n Output file " + gn);
        Console.WriteLine(File.ReadAllText(gn));
    }
} // DongHo
} // SangTao2

```

## 4.10 Số duy nhất

*Olimpic Baltics*

Tệp văn bản *UNIQUE.INP* chứa dãy số, mỗi số chiếm một dòng dài không quá 20 chữ số. Trong dãy này có duy nhất một số xuất hiện đúng một lần, các số còn lại đều xuất hiện đúng k lần. Hãy tìm số duy nhất đó. Số k không cho trước, nhưng biết rằng đó là một số chẵn khác 0. Kết quả hiển thị trên màn hình.

## Thuật toán

Ta dựa vào một kiến thức có từ hàng ngàn năm trước, đó là biểu diễn số theo vị trí. Ta lần lượt đọc từng dòng vào một biến string sau đó ghi vào một mảng a số lần xuất hiện của từng chữ số tại từng vị trí, a[c,i] cho biết số lần xuất hiện của chữ số c tại cột i tính từ trái qua phải.

Với thí dụ đã cho, trên cột 1 ta tính được a['1',1] = 5, a['2',1] = 4, a['3',1] = 0,... , a['8',1] = 4...  
Như vậy mảng a có kích thước 10 hàng đủ chứa 10 chữ số '0'..'9' và 20 cột đủ chứa các chữ số dài nhất. Sau khi đọc xong dữ liệu, ta thấy mọi phần tử của mảng a hoặc là chia hết cho k do đó là số chẵn, hoặc là số lẻ có dạng m.k + 1, m = 0, 1, 2,... Nếu a[c,i] là số lẻ thì c sẽ là chữ số xuất hiện tại cột i của số duy nhất cần tìm.

## Chương trình Pascal

```
(* Pascal *)
procedure XuLi;
const mn = 20;
  ChuSo = ['0'..'9'];
  fn = 'unique.inp';
var a: array['0'..'9',1..mn] of integer;
  f: text;
  i: integer;
  s: string;
  cs: char;
begin
  fillchar(a,sizeof(a),0);
  assign(f,fn); reset(f);
  while not seekeof(f) do
  begin
    readln(f,s);
    for i:=1 to length(s) do
      if s[i] in ChuSo then inc(a[s[i],i]);
  end;
  close(f);
  s := ''; { Kết quả }
  for i := 1 to mn do
    for cs := '0' to '9' do
      if Odd(a[cs,i]) then s := s+cs;
  writeln(s);
end;
BEGIN
  XuLi;
  readln;
END.
```

UNIQUE.IN P	MÀN HÌNH
1357	10203040
2008	
80	
1357	
2008	
80	
2008	
1357	
10203040	
80	
2008	
80	
1357	

Giải thích: Số duy nhất cần tìm là 10203040. Các số còn lại đều xuất hiện 4 lần.

## Chương trình C#

```
// C#
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
```

```

namespace SangTao2 {
    class Unique {
        const string fn = "UNIQUE.INP";
        static void Main(string[] args) {
            Console.WriteLine(XuLi());
            Console.ReadLine();
        }
        static string XuLi(){
            string s;
            StreamReader f = File.OpenText(fn);
            int mn = 20;
            int[,] a = new int[10,mn];
            Array.Clear(a, 0, a.Length);
            while (true){
                s = f.ReadLine().Trim();
                if (s.Length == 0) break;
                for (int i = 0; i < s.Length; ++i)
                    if (s[i] >= '0' && s[i] <= '9')
                        ++a[s[i] - '0', i];
            }
            f.Close();
            string kq = "";
            for (int i = 0; i < mn; ++i)
                for (int cs = 0; cs <= 9; ++cs)
                    if (a[cs, i] % 2 == 1)
                        kq += cs;
            return kq;
        }
    } // Unique
} // SangTao2

```

## Độ phức tạp

Nếu có n số dài tối đa m chữ số thì ta cần xét mỗi chữ số 1 lần, nghĩa là tổng cộng cần  $n \cdot m$  thao tác. Duyệt mảng a cần  $10 \cdot m$  thao tác là số rất nhỏ so với  $n \cdot m$ .

## Các biến thể của bài UNIQUE

1. Nếu đầu bài cho biết số  $k$  thì không cần đòi hỏi  $k$  là số chẵn.
  2. Biết duy nhất một số xuất hiện đúng  $m$  lần, các số còn lại đều xuất hiện  $k$  lần như nhau,  $k \neq m$  và  $k$  và  $m$  nguyên tố cùng nhau. Bạn thử suy nghĩ xem có cần biết cụ thể các giá trị của  $m$  và  $k$  không? Bạn thử tìm một số điều kiện của  $k$  và  $m$ ?
  3. Thay các số bằng các dãy ký tự A..Z dài tối đa  $m$ .
- 

21.01.2010

NXH

# TỦ SÁCH TRI THỨC DUY TÂN

---

NGUYỄN XUÂN HUY

## SÁNG TẠO TRONG THUẬT TOÁN VÀ LẬP TRÌNH

với ngôn ngữ Pascal và C++

Tập 3

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

**MỤC LỤC**

<i>Lời nói đầu .....</i>	4
<i>Chương 1 Các thuật toán trên String.....</i>	5
1.1 Xâu kí tự .....	5
1.2 Về tổ chức dữ liệu vào/ra.....	6
1.3 Data .....	6
1.4 Xâu con chung.....	8
1.5 Đoạn chung.....	9
1.6 Đoạn lặp.....	11
1.7 Từ điển.....	14
1.8 TEFI.....	17
1.9 E xiếc .....	20
<i>Chương 2 Xử lí dãy lệnh và biểu thức .....</i>	23
2.1 Val .....	23
2.2 Xâu thu gọn .....	26
2.3 Robot.....	29
2.4 Hàm nhiều biến .....	33
2.5 Files .....	38
2.6 Gen .....	44
2.7 Tối ưu hóa chương trình .....	44
2.8 Mức của biểu thức .....	45
2.9 Tháp .....	46
2.10 Mi trang .....	46
2.11 Xếp thẻ .....	49
2.12 Xếp xe.....	50
<i>Chương 3 Cặp ghép .....</i>	51
3.1 Chị Hằng.....	51
3.2 Domino.....	55
3.3 Thám hiểm.....	59
3.4 Show .....	64
3.5 Cặp ghép cực đại: Chị Hằng 2 .....	70
<i>Chương 4 Các phép lật và chuyển vị.....</i>	75
4.1 Lật xâu .....	75
4.2 Lật số nguyên .....	76
4.3 Sân bay vũ trụ .....	77
4.4 Cân .....	81
4.5 Biprime .....	87
4.6 Chuyển bi.....	90

4.7 Lát nền 2 .....	94
4.8 Test .....	103
4.9 Giải mã.....	105
<i>Chương 5 Luyện tập từ các đề thi .....</i>	<i>110</i>
5.1 Số nguyên tố cùng độ cao .....	110
5.2 Số nguyên tố cùng số bít 1 .....	112
5.3 Cắt hình .....	112
5.4 Tổng nhỏ nhất .....	115
5.5 Lò cò .....	119
5.6 Chuyển tin .....	127
5.7 Mã BW .....	130
5.8 Tam giác Pascal.....	134
5.9 Sơn mô hình.....	138
5.10 Nhúng mô hình.....	141
5.11 Số sát sau nhị phân .....	144
5.12 Hàm $f(n)$ .....	150
5.13 Hàm $h(n)$ .....	151
5.14 Rhythm .....	151
5.15 Cóc.....	152
5.16 Trả tiền .....	154
5.17 Game .....	156
5.18 Robots .....	160

## **Lời nói đầu**

---

Theo yêu cầu của bạn đọc, trong tập 3 này chúng tôi minh họa bằng hai ngôn ngữ lập trình là Pascal và Dev-C++. Pascal là ngôn ngữ lập trình mang tính sư phạm cao và được dùng để giảng dạy trong nhà trường phổ thông theo chương trình hiện hành. Dev-C++ là môi trường mã nguồn mở được các bạn sinh viên yêu thích và thường được chọn làm môi trường lập trình trong các cuộc đua tài quốc gia và quốc tế.

Cả hai môi trường Free Pascal và Dev-C++ đều được cung cấp miễn phí trên Internet.

Chúng tôi hy vọng rằng sẽ tiếp tục nhận được những ý kiến đóng góp quý báu của bạn đọc gần xa về nội dung và hình thức trình bày của bộ sách.

*Hà Nội, Mùa Xuân năm Dần 2010*

*Nguyễn Xuân Huy*

# Chương 1 Các thuật toán trên String

## 1.1 Xâu kí tự

Xâu kí tự là một dãy các kí tự viết liền nhau. Các kí tự được lấy từ một bảng chữ cái cho trước, thông thường là bảng mã ASCII. Trong các bài toán tin, kí tự thường được hiểu là *chữ cái viết HOA* hoặc viết *thường* theo trật tự bố trí trong bảng chữ cái tiếng Anh và các chữ số. Có thể hiểu xâu kí tự là một *mảng* *một chiều* chứa các kí tự. Đôi lúc ta gọi vẫn tắt là *xâu*. Hiểu theo nghĩa này ta có thể khai báo xâu kí tự như sau:

```
// Dev-C++  
char x[1000];  
char *y = new char[1000];
```

Cá hai khai báo trên là tương đương nhau và x, y đều có *dung lượng* hay sức chứa tối 1000 kí tự với các chỉ số từ 0 đến 999. Các xâu kí tự trong C++ được kết thúc bằng kí tự (dấu) *kết xâu* '\0'. Bạn cần chắc chắn rằng dấu kết xâu luôn luôn có mặt trong các xâu do bạn quản lý. Một số hàm hệ thống của C++ tự động đặt dấu kết xâu vào cuối xâu kí tự. Nếu bạn tự viết các hàm xử lí xâu thì bạn cần có thao tác tường minh đặt dấu kết xâu vào cuối xâu. Nếu bạn khai báo xâu kí tự x gồm 1000 kí tự như trên thì bạn chỉ được phép ghi vào xâu đó tối đa 999 kí tự (gọi là *các kí tự có nghĩa*). Vị trí cuối cùng x[999] phải dành để ghi dấu kết xâu '\0'.

Trong Pascal với những xâu ngắn, có chiều dài không quá 255 kí, tự bạn nên sử dụng kiểu string, thí dụ

```
(* Pas *)  
var x: string[100];
```

Khai báo trên cho phép bạn sử dụng xâu x như một mảng gồm 101 phần tử,

```
x: array[0..100] of char;
```

Tuy nhiên, bạn cần nhớ rằng phần tử x[0] được hệ thống dành riêng để ghi *chiều dài hiện hành* của xâu. Thí dụ,

```
(* Pascal *)  
var x: string[100];  
x := 'abc';
```

sẽ gán x[1] = 'a'; x[2] = 'b'; x[3] = 'c'; Riêng x[0] được gán kí tự có mã ASCII là 3: x[0] = #3.

Như vậy bạn được sử dụng đúng 100 kí tự có nghĩa.

Chiều dài hiện hành khác với *sức chứa*. Xâu x nói trên có sức chứa 100 bytes dành cho bạn, không tính byte đầu tiên x[0], còn chiều dài hiện hành là 3. Chiều dài hiện hành được tính trong C++ bằng hàm **strlen**, trong Pascal bằng hàm **length**.

Với những xâu dài trên 255 kí tự bạn nên khai báo như một mảng, thí dụ

```
(* Pascal *)  
var x: array[1..1000] of char;  
và xử lí x như một mảng.
```

Trong C++ cũng có kiểu dữ liệu string dành riêng cho việc quản lý các xâu. Với kiểu này bạn có thể thực hiện một số hàm tiện ích như cộng hai xâu x+y, gán trị x = y, ... Thí dụ,

```
// Dev-C++  
int main(){  
    string x = "abc", y = x;  
    cout << endl << x << " + " << y << " = " << (x+y);  
// abc + abc = abcabc  
    cin.get();  
    return 0;  
}
```

Các xâu trong đề bài đều được hiểu thông nhất với chỉ số tính từ 1 đến N. Khi lập trình bằng C++ bạn lưu ý chuyển đổi kết quả cuối cùng từ chỉ số i sang i+1. Bạn cũng có thể ghi dữ liệu từ chỉ số 1 trở đi, bỏ qua phần tử 0.

Hằng xâu kí tự trong C++ được ghi giữa hai dấu nháy kép, thí dụ "**string in CPP**", trong Pascal được ghi giữa hai dấu nháy đơn, thí dụ, '**string in Pascal**'. Nếu giữa hai dấu nháy đơn hoặc kép ta không ghi kí tự nào thì ta thu được một *xâu rỗng* là xâu có chiều dài 0.

Cho xâu  $s[1..n]$ . Một *đoạn* của  $s$  là dãy liên tiếp các kí tự trong  $s$ . Ta kí hiệu  $s[d..c]$  là đoạn của  $s$  tính từ chỉ số  $d$  đến chỉ số  $c$ . Thí dụ, nếu  $s = 'abcdefghijklmnopqrstuvwxyz'$  thì  $s[2..5] = 'bcde'$  là một đoạn. Đoạn  $s[i..n]$  được gọi là *tiền tố i của s* và được kí hiệu là  $i:s$ . Đoạn  $s[i..n]$  được gọi là *hậu tố i của s* và được kí hiệu là  $s:i$ . Xâu dài n kí tự có đúng n tiền tố và n hậu tố.

Nếu xóa khỏi  $s$  một số kí tự và (tất nhiên) dồn các kí tự còn lại cho kề nhau, ta sẽ thu được một *xâu con* của  $s$ .

## 1.2 Về tổ chức dữ liệu vào/ra

Trong hầu hết các bài ta giả thiết dữ liệu vào và ra được ghi trong các text file \*.INP và \*.OUT. Tên và cách thức ghi dữ liệu trong các file được cho trong từng thí dụ cụ thể của mỗi bài. Theo giả thiết này trong các bài giải sẽ chỉ tập trung giới thiệu những thuật toán cơ bản, các bạn sẽ tự viết phần tổ chức vào/ra để thu được chương trình hoàn chỉnh.

Turbo Pascal và Borland C++ bị hạn chế về miền nhớ. Các bạn nên sử dụng Free Pascal và DevC++ để có thể cấp phát những mảng dữ liệu đủ lớn với hàng tỷ bytes. Các mảng trong C++ được gán chỉ số 0, còn trong Pascal chỉ số mảng do người lập trình tự đặt. Trong DevC++, nếu  $f$  là input file dạng text thì dòng lệnh  $f >> x$  đọc dữ liệu vào đối tượng  $x$  đến khi *gặp dấu cách*. Muốn đọc đầy đủ một dòng dữ liệu chứa cả dấu cách từ input file  $f$  vào một biến mảng kí tự  $s$  ta có thể dùng phương thức `getline` như thí dụ sau đây

```
char s[1001];
```

```
f.getline(s,1000,'\\n');
```

Phương thức này đọc một dòng tối đa 1000 kí tự vào biến  $s$ , và thay dấu kết dòng ' $\n$ ' trong input file bằng dấu kết xâu ' $/0$ ' trong C.

Lệnh `memset(a,0,sizeof(a))` gán toàn 0 cho mọi byte của mảng  $a$ .

Lệnh `memmove(a,b,n)` copy  $n$  byte từ mảng  $b$  sang mảng  $a$ .

Lệnh `strcpy(x,"abcd")`; khởi trị " $abcd$ " cho xâu  $x$

Để làm quen với các thao tác đọc/ghi dữ liệu bạn hãy thử giải bài toán dưới đây.

## 1.3 Data

Trong file văn bản `data.inp` chứa dòng dữ liệu đầu tiên có nội dung  
"Tinh tong cua n so sau day:",  
trong đó  $n$  là một số nguyên dương  
cho trước.

Tiếp đến là  $n$  số nguyên ghi cách  
nhau qua dấu cách.

Yêu cầu: xác định giá trị của  $n$  và tổng của  $n$  số trong file `data.inp` rồi ghi kết quả vào output file `data.out` theo định dạng cho trong bảng.

### Thuật toán

Ta viết thủ tục Tong theo các bước:

1. Mở input file  $f$  tên "data.inp".
2. Cấp phát biến string  $s$ , đọc dòng đầu tiên vào  $s$ .
3. Duyệt  $s$  để tìm kí tự số đầu tiên, đọc tiếp số đó và ghi vào biến  $n$ .
4. Mở output file  $g$  tên "data.out".
5. Ghi dòng đầu tiên "Tong cua n so:" với  $n$  là giá trị cụ thể đọc được tại bước 3.
6. Đọc từng số trong  $n$  số từ file  $f$ , ghi vào file  $g$  kèm dấu  $+/ -$  và cộng dồn vào biến tổng  $t$ .
7. Ghi giá trị tổng  $t$  vào file  $g$ .
8. Đóng các files  $f$  và  $g$ .
9. Thu hồi miền nhớ đã cấp cho  $s$ .

**Độ phức tạp:** Cỡ  $n$ .

```
(* Pascal: data.pas *)
uses crt;
const fn = 'data.inp'; gn = 'data.out';
      bl = #32; { Dấu cách }
      nl = #13#10; { Xuống đầu dòng mới }
var n: integer;
function LaChuSo(c: char): Boolean;
begin
  LaChuSo := (c >= '0') and (c <= '9');
```

Các tiền tố và hậu tố của xâu  $s = 'abcd'$

Tiền tố	Hậu tố
1:s = s[1..1] = 'a'	s:1 = s[1..4] = 'abcd'
2:s = s[1..2] = 'ab'	s:2 = s[2..4] = 'bcd'
3:s = s[1..3] = 'abc'	s:3 = s[3..4] = 'cd'
4:s = s[1..4] = 'abcd'	s:4 = s[4..4] = 'd'

```

end;
procedure Tong;
var i,t,x : integer;
    s: string;
    f,g: text;
begin
    { Mo input file f ten fn = "data.inp" doc dong dau tien vao s }
    assign(f,fn); reset(f); readln(f,s);
    i := 1; { Duyet s tim chu so dau tien }
    while Not LaChuSo(s[i]) do inc(i);
    n := 0; { Doc so trong s ghi vao n }
    while LaChuSo(s[i]) do
begin
    begin
        n := n*10 + (ord(s[i]) - ord('0'));
        inc(i);
    end;
    assign(g,gn); rewrite(g); { Mo output file g ten gn="data.out" }
    writeln(g,'Tong cua ',n,' so:'); { Ghi dong thu nhat vao g }
    t := 0; { Khoi tri bien tich luy t }
    for i := 1 to n do { Doc lan luot tung so x trong n so }
begin
    read(f,x);
    if x > 0 then write(g,'+',x) else write(g,' ',x);
    t := t + x;
end;
writeln(g,' = ',t); { Ghi ket qua }
close(f); close(g); { Dong cac files }
end;
BEGIN
    Tong;
    writeln(nl,' Fini');
    readln;
END.

```

```

// DevC++ Data
#include <string.h>
#include <fstream>
#include <iostream>
#include <stdio.h>
using namespace std;
// DATA AND VARIABLE
const char * fn = "data.inp";
const char * gn = "data.out";
int n;
// PROTOTYPES
void Tong();
bool LaChuSo(char c);
// IMPLEMENTATION
int main(){
    Tong();
    cout << endl << endl << " Fini" << endl;
    cin.get();
    return 0;
}
bool LaChuSo(char c) { return (c >= '0' && c <= '9'); }
void Tong() {
    const int mn = 100;
    int i, t, x;
    ifstream f(fn); // Mo input file f ten fn = "data.inp"
    char *s = new char [mn]; // cap phat s
    f.getline(s,mn,'\n'); // doc toan bo dong thu nhat
    for (i = 0; i < strlen(s); ++i) // duyet xau s tim chu so
        if (LaChuSo(s[i])) break;
}

```

```

n = 0; // khai tri so n
while (LaChuSo(s[i])) { // doc so n
    n = n*10 + int(s[i] - '0');
    ++i;
}
t = 0; // khai tri bien tong t
ofstream g(gn); // Mo output file g ten gn = "data.out"
g << "Tong cua " << n << " so:" << endl;
for (i = 0; i < n; ++i) {
    f >> x; // doc tung so x
    if (x > 0) g << " +" << x; else g << " " << x;
    t += x; // lay tong
}
g << " = " << t;
f.close(); // dong input file
g.close();
delete s; // thu hồi biến s, nếu cần
}

```

#### 1.4 Xâu con chung

Hãy tìm chiều dài lớn nhất k trong số các xâu con chung của hai xâu x và y.

Thí dụ,  $x = "xaxxbxcxd"$ ,  $y = "ayybycdy"$ , chiều dài của xâu con chung dài nhất là 4 ứng với xâu "abcd".

##### Thuật toán

Xét hàm 2 biến  $s(i,j)$  là đáp số khi giải bài toán với 2 tiền tố  $i:x$  và  $j:y$ . Ta có,

- $s(0,0) = s(i,0) = s(0,j) = 0$ : một trong hai xâu là rỗng thì xâu con chung là rỗng nên chiều dài là 0;
- Nếu  $x[i] = y[j]$  thì  $s(i,j) = s(i-1,j-1) + 1$ ;
- Nếu  $x[i] \neq y[j]$  thì  $s(i,j) = \text{Max} \{ s(i-1,j), s(i,j-1) \}$ .

Để cài đặt, trước hết ta mường tượng là có thể sử dụng mảng hai chiều  $v$  với qui ước  $v[i][j] = s(i,j)$ . Sau đó ta cải tiến bằng cách sử dụng 2 mảng một chiều  $a$  và  $b$ , trong đó  $a$  là mảng đã tính ở bước thứ  $i-1$ ,  $b$  là mảng tính ở bước thứ  $i$ , tức là ta qui ước  $a = v[i-1]$  (dòng  $i-1$  của ma trận  $v$ ),  $b = v[i]$  (dòng  $i$  của ma trận  $v$ ). Ta có, tại bước  $i$ , ta xét kí tự  $x[i]$ , với mỗi  $j = 0..len(y)-1$ ,

- Nếu  $x[i] = y[j]$  thì  $b[j] = a[j-1] + 1$ ;
- Nếu  $x[i] \neq y[j]$  thì  $b[j] = \text{Max} \{ a[j], b[j-1] \}$ .

Sau khi đọc dữ liệu vào hai xâu x và y ta gọi hàm XauChung để xác định chiều dài tối đa của xâu con chung của x và y.  $a, b$  là các mảng nguyên 1 chiều.

**Độ phức tạp:** Cỡ  $m.n$ ,  $m = \text{len}(x)$ ,  $n = \text{len}(y)$ .

```

(* XauChung.pas *)
function Max(a,b: integer): integer;
begin if a > b then Max := a else Max := b; end;
function XauChung(var x,y: string): integer;
var m,n,i,j: integer;
    a,b: array[0..255] of integer;
begin
    m := length(x); n := length(y);
    fillchar(a,sizeof(a),0);
    for i := 1 to m do
    begin
        for j := 1 to n do
            if x[i] = y[j] then b[j] := a[j-1]+1
            else b[j] := Max(a[j],b[j-1]);
        a := b;
    end;
    XauChung := a[n];
end;
BEGIN
    writeln;
    writeln(XauChung('xabcxxxxd','aybcydyy')); { 4 }
    readln;
END.

// Dev-C++: XauChung.cpp

```

```

int Max(int a, int b) { return (a > b) ? a : b; }
int XauChung(char *x, char *y) {
    int i,j;
    int m = strlen(x), n = strlen(y);
    int a[n], b[n];
    for (j = 0; j < n; ++j)
        a[j] = (x[0] == y[j]) ? 1 : 0;
    for (i = 1; i < m; ++i) {
        b[0] = (x[i] == y[0]) ? 1 : 0;
        for (j = 1; j < n; ++j)
            if (x[i] == y[j]) b[j] = a[j-1] + 1;
            else b[j] = Max(a[j],b[j-1]);
        memmove(a,b,n*sizeof(int));
    }
    return a[n-1];
}

int main() {
    cout << endl << XauChung("xaxxxbcxd","aybcyydy"); // 4
    cin.get();
    return 0;
}

```

### Cách làm test

Bạn hãy viết ra một xâu s nào đó làm đáp số, tức là xâu con chung, sau đó thêm vào s một số kí tự để nhận được xâu x, rồi lại thêm cho s một số kí tự khác để nhận được xâu y.

### Các bài tương tự

- Xâu chung 2.** Cho hai xâu x gồm m và y gồm n kí tự. Cần xóa đi từ xâu x dx kí tự và từ xâu y dy kí tự để thu được hai xâu giống nhau. Hãy xác định giá trị nhỏ nhất của tổng dx+dy.
- Dãy con chung.** Cho hai dãy số nguyên a gồm m và b gồm n phần tử. Cần xóa đi ít nhất là bao nhiêu phần tử từ mỗi dãy trên để thu được hai dãy giống nhau.

Thuật toán cho bài Xâu chung 2

```

k = XauChung(x,y);
dx = len(x) - k;
dy = len(y) - k;

```

### 1.5 Đoạn chung

Hãy tìm chiều dài lớn nhất k trong số các đoạn chung của hai xâu x và y.

Thí dụ, x = "abcxxabcdx", y = "aybcyabcdy" có chiều dài của đoạn chung dài nhất là 4 ứng với đoạn "abcd".

### Thuật toán

Xét hàm 2 biến s(i,j) là chiều dài lớn nhất của hai đoạn giống nhau x[i-k+1..i] và y[j-k+1..j], k → max. Ta có,

- Nếu x[i] = y[j] thì s(i,j) = s(i-1,j-1) + 1;
- Nếu x[i] ≠ y[j] thì s(i,j) = 0.

Đáp số sẽ là Max { s(i,j) | 1 ≤ i ≤ len(x), 1 ≤ j ≤ len(y) }.

Để cài đặt ta có thể sử dụng hai mảng một chiều như bài trước. Ta cũng có thể sử dụng một mảng một chiều a và hai biến phụ v và t. Biến t lưu tạm giá trị trước khi tính của a[j]. Biến v lấy lại giá trị t để tính cho bước sau.

**Độ phức tạp:** Cỡ m.n, m = len(x), n = len(y).

```

(* DChung.pas *)
function Max(a,b: integer): tự viết;
function DoanChung(x,y: string): integer;
var m,n,i,j,v,t,kmax: integer;
    a: array[1..255] of integer;
begin
    m := length(x); n := length(y); kmax := 0;
    fillchar(a,sizeof(a),0);
    for i := 1 to m do

```

```

begin
    v := 0;
    for j := 1 to n do
    begin
        t := a[j];
        if x[i] = y[j] then a[j] := v+1
        else a[j] := 0;
        kmax := Max(kmax,a[j]);
        v := t;
    end;
end;
DoanChung := kmax;
end;
BEGIN
    writeln(DoanChung('xabcxxabcdxd','aybcyabcdydy')) ; {4}
    writeln(' Fini');
    readln;
END.

// DevC++: DoanChung.cpp
int Max(int a, int b); // tự viết
int DoanChung(char *x, char *y) {
    int i, j, kmax = 0, v, t ;
    int m = strlen(x), n = strlen(y);
    int a[n];
    memset(a,0,sizeof(a));
    for (i = 0; i < m; ++i) {
        v = 0;
        for (j = 0; j < n; ++j) {
            t = a[j];
            if (x[i] == y[j]) a[j] = v + 1;
            else a[j] = 0;
            kmax = Max(kmax,a[j]);
            v = t;
        }
    }
    return kmax;
}
int main() {
    cout << endl << DoanChung("xabcxxabcdxd","aybcyabcdydy");//4
    cin.get();
    return 0;
}

```

### Cách làm test

Test 1. Trước hết viết một xâu s sau đó xây dựng 2 xâu x = y = s. Đáp số len(s). Thí dụ, x = y = s = 'abcaaabb'. Đáp số: 8

Test 2. Sửa lại Test 1 bằng cách thêm vào x và y một số kí tự khác nhau. Đáp số: len(s). Thí dụ, x = 'xy'+s+'uvz'; y = 'uv'+s+'xy'. Đáp số: 8.

Test 3. Sửa lại Test 2 bằng cách chèn thêm một đoạn nhỏ của s vào x và y. Thí dụ, x = 'xy'+s+'uv'+s'; y = 'u' + s' + 'v'+ s +'xy' + s' với s' = 'abcaaab' (hụt 1 kí tự so với s. Đáp số: 8.

### Các bài tương tự

**1. Đoạn chung 2.** Cho hai xâu x gồm m và y gồm n kí tự. Tìm đoạn chung dài nhất của hai xâu này. Kết quả cho ra 4 giá trị dx, cx, dy, cy, trong đó x[dx..cx] = y[dy..cy] là hai đoạn tìm được.

**2. Đoạn chung 3.** Cho hai dãy số nguyên a gồm m và b gồm n phần tử. Xác định chiều dài lớn nhất k để hai dãy cùng chứa k phần tử liên tiếp như nhau: a[i] = b[j], a[i+1] = b[j+1],...,a[i+k-1] = b[j+k-1].

### Thuật toán cho bài Đoạn chung 2

Khi phát hiện a[j] > kmax ta ghi nhận imax = i; jmax = j; kmax = k. Cuối thủ tục ta tính cx = imax; dx = cx-kmax+1; cy = jmax; dy = cy-kmax+1.

## 1.6 Đoạn lặp

Những viên ngọc lập trình (Bentley)

Cho xâu s chứa n kí tự. Hãy xác định ba số nguyên i, j và k thỏa điều kiện  $1 \leq i < j \leq n$ , k là giá trị max thỏa điều kiện  $s[i] = s[j]$ ,  $s[i+1] = s[j+1]$ , ...,  $s[i+k-1] = s[j+k-1]$ . Hai đoạn bằng nhau gồm k kí tự trong s là  $s[i..i+k-1]$  và  $s[j..j+k-1]$ ,  $i < j$ , k max được gọi là hai đoạn lặp trong s.

Thí dụ, s = 'xababababayy' cho ta  $i = 2, j = 4, k = 5$  ứng với đoạn lặp s[2..6] = 'ababa'.

### Thuật toán 1

Bài này khá giống bài đoạn chung. Xét hàm 2 biến s(i,j) là chiều dài lớn nhất của hai đoạn giống nhau  $x[i-k+1..i]$  và  $y[j-k+1..j]$ ,  $i < j$ ,  $k \rightarrow \max$ . Ta có,

- Nếu  $x[i] = x[j]$  thì  $s(i,j) = s(i-1,j-1) + 1$ ;
- Nếu  $x[i] \neq x[j]$  thì  $s(i,j) = 0$ .

Đáp số sẽ là  $\max \{ s(i,j) \mid 1 \leq i \leq \text{len}(x), 1 \leq j \leq \text{len}(y), i < j \}$ .

Để cài đặt ta có thể sử dụng hai mảng một chiều như bài trước. Ta cũng có thể sử dụng một mảng một chiều a và hai biến phụ v và t. Biến t lưu tạm giá trị trước khi tính của a[j]. Biến v lấy lại giá trị t để tính cho bước sau.

**Độ phức tạp:** Cỡ  $n^2$ ,  $n = \text{len}(s)$ .

```
(* Repeat.pas *)
uses crt;
var i,j,k: integer;
procedure DoanLap(s: string; var imax, jmax, kmax: integer);
var n,i,j,v,t: integer;
    a: array[1..255] of integer;
begin
    n := length(s); kmax := 0;
    fillchar(a,sizeof(a),0);
    for i := 1 to n do
    begin
        v := 0;
        for j := i+1 to n do
        begin
            t := a[j];
            if s[i] = s[j] then a[j] := v+1
            else a[j] := 0;
            if kmax < a[j] then
            begin
                kmax := a[j]; imax := i-kmax+1; jmax := j-kmax+1;
            end;
            v := t;
        end;
    end;
BEGIN
    DoanLap('xababababayy',i, j, k);
    writeln(i,' ', j, ' ',k); { i = 2, j = 4, k = 5 }
    readln;
END.

// DevC++: Repeat.cpp
void DoanLap(char *s, int &imax, int &jmax, int &kmax) {
    int i, j , v, t ;
    int n = strlen(s);
    int a[n];
    kmax = 0;
    memset(a,0,sizeof(a));
    for (i = 0; i < n; ++i) {
        v = 0;
        for (j = i+1; j < n; ++j) {
            t = a[j];
            if (s[i] == s[j]) a[j] = v + 1;
            else a[j] = 0;
        }
    }
}
```

```

        if (kmax < a[j]) {
            kmax = a[j]; imax = i-kmax+2; jmax = j-kmax+2;
        }
        v = t;
    }
}
int main() {
    int i, j, k;
    DoanLap("xababababayy", i, j, k);
    cout << endl << i << " " << j << " " << k;//i = 2, j = 4, k = 5
    cin.get();
    return 0;
}

```

### Thuật toán 2 (Bentley, Những viên ngọc lập trình)

1. Sắp tăng theo chỉ dẫn các hậu tố của s theo trật tự từ điển. Gọi dãy được sắp theo chỉ dẫn này là id[1..n], n = len(s).
2. Duyệt dãy được sắp trong id, với mỗi cặp hậu tố đứng kề nhau s:id[i] và s:id[i-1], i = 2..n ta gọi hàm ComLen(id[i], id[i-1]) để tính chiều dài của khúc đầu chung (hay tiền tố) dài nhất của chúng. Đáp số khi đó sẽ là

$$\text{Max} \{ \text{ComLen}(id[i], id[i-1]) \mid i = 2..len(s) \}$$

Thủ tục sắp theo chỉ dẫn id xét các hậu tố s:id[i] được thực hiện theo giải thuật quicksort như sau:

```

void IdSort(char * s, int * id, int d, int c) {
    int i = d, j = c, m = id[(i+j)/2], t;
    while (i < j) {
        while (Sanh(s,id[i],m) < 0) ++i;
        while (Sanh(s,id[j],m) > 0) --j;
        if (i <= j) {
            t = id[i]; id[i] = id[j]; id[j] = t;
            ++i; --j;
        }
    }
    if (d < j) IdSort(s,id,d,j);
    if (i < c) IdSort(s,id,i,c);
}

```

Hàm **Sanh**(s, i, j) so sánh hai hậu tố s:i và s:j theo trật tự từ điển hoạt động theo nguyên tắc sau: Lần lượt so sánh các cặp kí tự s[i] và s[j] cho đến cuối xâu, nếu gặp cặp kí tự khác nhau đầu tiên thì xét: kí tự nào nhỏ hơn thì xâu chứa nó sẽ nhỏ hơn xâu kia.

```

int Sanh(char *s, int i, int j) { //so sanh 2 hau to s:i, s:j
    int k = Min(strlen(s)-i, strlen(s)-j), v;
    for (v = 0; v < k; ++v, ++i, ++j)
        if (s[i] != s[j]) return (s[i] < s[j]) ? -1 : 1;
    return (i < j) ? 1 : ((i > j) ? -1 : 0);
}

```

Hàm ComLen(s, i, j) cho ra chiều dài lớn nhất của hai khúc đầu giống nhau của hai hậu tố s:i và s:j.

```

int ComLen(char *s, int i, int j) {
    int k = Min(strlen(s)-i, strlen(s)-j);
    for (int v = 0; v < k; ++v, ++i, ++j)
        if (s[i] != s[j]) return v;
    return k;
}

```

Thuật toán Bentley khi đó sẽ được triển khai qua hàm sau,

```

void DoanLap(char *s, int & imax, int & jmax, int & kmax) {
    int i, k;
    int n = strlen(s);
    int id[n];
    for (i = 0; i < n; ++i) id[i] = i;
    IdSort(s,id,0,n-1);
    for (i = 1; i < n; ++i) {

```

```

        if ((k = ComLen(s, id[i], id[i-1])) > kmax) {
            kmax = k; imax = id[i]+1; jmax = id[i-1]+1;
        }
    }
    if (imax > jmax) { i = imax; imax = jmax; jmax = i; }
}

(* DoanLap2.pas *)
uses crt;
var i,j,k: integer;
type mil = array[1..256] of integer;
function Min(a,b: integer): integer;
begin if a < b then Min := a else Min := b; end;
function Sanh(var s: string; i,j: integer): integer;
    var k, v: integer;
begin
    k := Min(length(s)-i,length(s)-j);
    for v := 0 to k do
        if s[i+v] <> s[j+v] then
        begin
            if s[i+v] < s[j+v] then Sanh := -1 else Sanh := 1;
            exit;
        end;
        if i < j then Sanh := 1
        else if i > j then Sanh := -1 else Sanh := 0;
end;
procedure IdSort(var s: string; var id: mil; d,c: integer);
    var i, j, m, t: integer;
begin
    i := d; j := c; m := id[(i+j) div 2];
    while (i <= j) do
    begin
        while Sanh(s,id[i],m) < 0 do inc(i);
        while Sanh(s,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 IdSort(s,id,d,j);
    if i < c then IdSort(s,id,i,c);
end;
function ComLen(var s: string; i, j: integer): integer;
    var v,k: integer;
begin
    k := Min(length(s)-i, length(s)-j);
    for v := 0 to k do
        if s[i+v] <> s[j+v] then
        begin ComLen := v; exit end;
    ComLen := k+1;
end;
procedure DoanLap(s: string; var imax, jmax, kmax: integer);
var n,i,j,k: integer;
    id: mil;
begin
    n := length(s); kmax := 0;
    for i := 1 to n do id[i] := i;
    IdSort(s,id,1,n);
    for i := 2 to n do
    begin
        k := ComLen(s,id[i],id[i-1]);
        if k > kmax then

```

```

begin
    kmax := k; imax := id[i]; jmax := id[i-1];
end;
end;
if imax > jmax then
begin i := imax; imax := jmax; jmax := i end;
end;
BEGIN
DoanLap('xabababayy',i, j, k);
writeln; writeln(i,' ', j, ' ',k);
readln;
END.

```

### Cách làm Test

Xây dựng 4 xâu X, Y, A và B không có các kí tự chung. Ghép XABAB...ABY một số lần. Đáp số: i = len(X) + 1, j = len(X)+Len(A)+Len(B)+1, k = (v-1).(len(A)+len(b)) với v là số lần lặp các cặp AB.

Thí dụ, với X = 'xy'; Y = 'zt'; A = 'abcde'; B = 'fghhik' ta có thể xây dựng các Test sau đây.

Test 1. s = XABABY = 'xyabcdefghhikabcdeffghhikzt'. Đáp số i = 3, j = 2 + 5 + 6 + 1 = 14, k = 5+6 = 11.

Test 2. s = XABABABY = 'xyabcdefghhikabcdeffghhikabcdeffghhikzt'. Đáp số i = 3, j = 14, k = 2\*11 = 22.

### 1.7 Từ điển

Olimpic Moscow

Các từ trong bài được hiểu là một dãy liên tiếp các chữ cái a, b, ..., z. Một file văn bản chứa một từ điển T gồm tối đa  $n = 100$  từ khác nhau đối với. Mỗi từ dài không quá 50 kí tự và được viết trên một dòng. Cho một từ s dài không quá 200 kí tự. Hãy cho biết cần xóa đi khỏi s tối thiểu bao nhiêu chữ cái để phần còn lại tạo thành dãy liên tiếp các từ trong từ điển T, mỗi từ có thể xuất hiện nhiều lần.

Thí dụ,

dic.inp	dic.out	Giải thích
6 abba not is astra saint panama <b>saintpanamtranaisnotsa</b> intabba	5	Sau khi xóa 5 chữ cái (gạch dưới) <b>saintpanamtranaisnotsa</b> intabba ta thu được dãy ghép của các từ 5, 6, 3, 2, 5, 1 <b>saintpanamaisnotsa</b> intabba

### Thuật toán

Gia sử T là tập n từ trong từ điển, s là từ cần xử lí. Gọi d(i) là hàm cho đáp số khi giải bài toán với tiền tố i:s = s[1..i]. d(i) là số kí tự tối thiểu cần xóa khỏi s[1..i] để phần còn lại của s[1..i] tạo thành dãy liên tiếp các từ trong từ điển T. Với mỗi từ w dài m kí tự trong từ điển T ta xét hàm Nhung(w,i) có chức năng nhúng từ w vào tiền tố i:s như sau. Hàm cho ra chỉ số v thỏa hai điều kiện sau:

1. w là xâu con của s[v..i], nghĩa là nếu xóa đi một số kí tự khỏi s[v..i] ta sẽ thu được từ w,

2. s[v] = w[1], s[i] = w[m].

Nếu w được nhúng trong s[v..i] thì số kí tự cần xóa khỏi s[v..i] để thu được từ w sẽ là  $i-v+1-\text{len}(w)$ . Nếu từ w được chọn thì tổng số kí tự cần xóa khỏi tiền tố i:s sẽ là  $d(v-1) + i-v+1-\text{len}(w)$ . Ta cần chọn w sao cho giá trị này đạt min. Vậy,

$$d(i) = \min \{ d(v-1) + i-v+1-\text{len}(w) \mid w \in T, v = \text{Nhung}(w,i) \}$$

Khi w không thể nhúng được trong s[1..i] ta đặt v = Nhung(w,i) = 0 (pascal) hoặc -1 (C).

```

(* TuDien.pas *)
uses crt;
const fn = 'dic.inp'; gn = 'dic.out'; nl = #13#10; bl = #32;
type str = string[52];
var f,g: text;
s: string[202];
w: array [1..102] of str;
n: integer;
d: array[0..202] of integer;

```

```

kq: integer;
procedure Doc;
  var i: integer;
begin
  assign(f,fn); reset(f); readln(f,n);
  for i := 1 to n do readln(f,w[i]);
  readln(f,s); close(f);
end;
procedure Ghi(v: integer);
begin
  assign(g,gn); rewrite(g);
  writeln(g,v);
  close(g);
end;
function Nhung(var w: str; i: integer): integer;
var j: integer;
begin
  Nhung := 0; j := length(w);
  if j > i then exit;
  if w[j] <> s[i] then exit;
  for i := i downto 1 do
    if (s[i] = w[j]) then
      begin
        dec(j);
        if j = 0 then begin Nhung := i; exit; end;
      end;
  end;
  function Min(a,b: integer): integer;
begin if (a < b) then Min := a else Min := b; end;
procedure Tinhd(i: integer);
  var j,v: integer;
begin
  d[i] := d[i-1]+1;
  for j := 1 to n do
  begin
    v := Nhung(w[j],i);
    if v > 0 then d[i] := Min(d[i], d[v-1]+i-v+1-length(w[j]));
  end;
end;
function XuLi: integer;
  var m, i: integer;
begin
  d[0] := 0; m := length(s);
  for i := 1 to m do Tinhd(i);
  XuLi := d[m];
end;
BEGIN
  Doc;
  kq := XuLi; Ghi(kq);
  writeln(nl,nl,kq,nl,' Fini '); readln;
END.

```

```

// DevC++: TuDien.cpp
#include <string.h>
#include <fstream>
#include <iostream>
#include <stdio.h>
using namespace std;
// DATA AND VARIABLE
const char * fn = "dic.inp";
const char * gn = "dic.out";
char w[102][52];

```

```

char s[202];
int d[202];
int n, lens;
// P R O T O T Y P E S
void Doc();
void Xem();
int Nhung(char *, int i);
int XuLi();
void Tinhd(int);
int Min(int, int);
void Ghi(int);
// I M P L E M E N T A T I O N
int main(){
    Doc(); Xem();
    int kq = XuLi();
    Ghi(kq);
    cout << endl << kq;
    cout << endl << endl << " Fini" << endl;
    cin.get();
    return 0;
}
void Ghi(int v) {
    ofstream g(gn);
    g >> v;
    g.close();
}
int Min(int a, int b) { return (a < b) ? a : b; }
void Doc() {
    ifstream f(fn);
    f >> n;
    for (int i = 0; i < n; ++i) f >> w[i];
    f >> s; lens = strlen(s);
    f.close();
}
void Xem() {
    cout << endl << n << " " << s;
    for (int i = 0; i < n; ++i) cout << endl << w[i];
}
// Nhun tu w vao tien to s:i
int Nhung(char *w, int i) {
    int j = strlen(w)-1;
    if (i < j) return -1;
    if (w[j] != s[i]) return -1;
    for (; i >= 0; --i)
        if (w[j] == s[i]) {
            --j;
            if (j < 0) return i;
        }
    return -1;
}
int XuLi() {
    for (int i = 0; i < lens; ++i) Tinhd(i);
    return d[lens-1];
}
void Tinhd(int i) {
    int j, k, v;
    d[i] = (i == 0) ? 1 : (d[i-1]+1);
    for (j = 0; j < n; ++j)
        if ((v = Nhung(w[j],i)) >= 0) {
            k = (v == 0) ? 0 : d[v-1];
            d[i] = Min(d[i], k+i-v+1-strlen(w[j]));
        }
}

```

**Độ phức tạp.** Cõ p.n.m, trong đó p = len(s), n là số từ trong từ điển T, m là chiều dài của từ dài nhất trong T.

**Chú thích** Bạn có thể cài tiến chương trình như sau. Khi đọc dữ liệu và tổ chức từ điển T bạn có thể loại trước khỏi T những từ w nào mà kí tự đầu tiên w[1] hoặc kí tự cuối cùng w[m] không xuất hiện trong s vì khi đó chắc chắn là hàm Nhung sẽ cho giá trị -1. Tốt hơn cả là bạn cho w trượt trong s để xác định xem w đặt lọt tại các chỉ số nào trong s.

## 1.8 TEFI

TEFI.INP	TEFI.OUT
15 27 .....***..... ***.....*....***.. *.....*.....*.. ***.....***.. *....*****.....*.. *.....*.....*....*....*.... ***..*....*....***....*.... ....*....*....*....*.... ....*....*....*....*....*.... ....*....*....*....*.... ....*....*....*....*.... ....*....*....*....*.... ....*....*....*....*.... ....*....*....*.... ....*....*.... ....*....*....	3 2 2 3

Trong text file tên TEFI.INP gồm n dòng và m cột người ta dùng dấu chấm '.' tạo thành một bảng nền. Trên bảng nền đó người ta dùng dấu sao '\*' để viết các chữ IN HOA T, E, F và I theo các qui tắc sau:

- chân phương, không có chân
  - đơn nét
  - các nét song song không dính nhau
  - các nét của hai chữ không dính nhau mà cách nhau ít nhất một dấu chấm
- Hãy đếm số chữ cái mỗi loại.  
Thí dụ bên cho biết có 3 chữ T, 2 chữ E, 2 chữ F và 3 chữ I.

## Thuật toán

Ta xét hai dòng x và y liên tiếp nhau trong bảng nền và thử phát hiện đặc trưng của các chữ cái dựa trên 2 dòng này. Ta thấy, chữ T có đặc trưng duy nhất (không lẫn với các chữ cái khác) là đầu trên gồm 1 vạch ngang (-) và 1 số đứng (|) dính nhau. Chữ I có đặc trưng duy nhất là một số đứng (|). Trong chữ E có 2 nét ngang trên và 2 nét ngang dưới, chữ F có 2 nét ngang trên và 1 nét ngang dưới.

i			i	i	i						
x	*	*	.	*	*	.	*	*	.	*	*
y	*		.	*	.	*		.	*	*	.
Dấu trên			Dấu trên			Nét ngang			Nét ngang		
chữ T			chữ I			trên và			dưới của		
						giữa của			chữ E và F		

$dnt = \text{số nét ngang trên}.$

$dnd = \text{số nét ngang dưới}.$

$dx = \text{đếm số chữ X};$

$X = \{T,E,F,I\}.$

Mỗi chữ E có 2 nét ngang trên và 2 nét ngang dưới. Mỗi chữ F có 2 nét ngang trên và 1 nét ngang dưới.

$de + df = dnt / 2; de = dnd - dnt / 2.$

$df = dnt / 2 - de.$

Để tiện xử lý ta thêm vào đầu và cuối mỗi xâu một dấu chấm ''. Các trường hợp cần xét được mô tả chi tiết dưới dạng bảng quyết định.

Bảng quyết định cho bài toán TEFI

Bảng quyết định gồm 2 phần: phần điều kiện và phần quyết định. Các điều kiện được liệt kê độc lập nhau.

Giá trị 1 ứng với điều kiện đúng (true), 0 ứng với điều kiện sai (false), dấu - cho biết điều kiện này không cần xét.

Bảng được đọc theo cột: nếu các điều kiện trong cột thuộc phần điều kiện được thỏa thì quyết định tại cột đó được chọn.

$x - \text{dòng trên}; y - \text{dòng dưới}.$

<b>Điều kiện</b>	<b>x[i] = '*'</b>	1	0	1	1
	<b>x[i-1] = '**'</b>	1	-	0	0
	<b>x[i+1] = '**'</b>	-	-	1	-
	<b>y[i] = '*'</b>	1	1	1	1
	<b>y[i-1] = '**'</b>	-	0	0	0
	<b>y[i+1] = '**'</b>	-	0	-	1
<b>Quyết định</b>	<b>T (chữ T)</b>	1			
	<b><math>\Gamma</math> (nét ngang trên)</b>			1	
	<b>L (nét ngang dưới)</b>				1
	<b>I (chữ I)</b>		1		

Dựa vào bảng quyết định ta duyệt đồng thời dòng trên x và dòng dưới y để đếm các giá trị sau:

dt là số lượng chữ T, di là số lượng chữ i, dnt là số lượng nét ngang trên  $\Gamma$  và dnd là số lượng nét ngang dưới L. Từ các giá trị dnt và dnd ta dễ dàng tính được số lượng chữ E và số lượng chữ F. Vì mỗi chữ E và mỗi chữ F đều có cùng 2 nét ngang trên nên  $de + df = dnt / 2$  (1). Mỗi chữ E có 2 nét ngang dưới, mỗi chữ F có 1 nét ngang dưới nên  $2.de + df = dnd$  (2). Trừ từng vế của (2) cho (1) ta thu được  $de = dnd - dnt / 2$ . Từ (1) ta suy ra  $df = dnt / 2 - de$ .

**Độ phức tạp.** cỡ m.n = dung lượng input file.

```
(* TEFI.PAS *)
uses crt;
const fn = 'tefi.inp'; gn = 'tefi.out';
bl = #32; nl = #13#10; ss = '*'; cc = '.';
var x, y: string;
dt, de, df, di: integer;
n, m: integer;
dnt, dnd: integer;
f,g: text;
procedure EF(i: integer);
begin
  if (x[i+1] = ss) then inc(dnt)
  else if (y[i+1] = ss) then inc(dnd)
end;
procedure TEF(i: integer);
begin
  if (y[i] = cc) then exit;
  if (x[i-1] = cc) then EF(i)
  else inc(dt);
end;
procedure II(i: integer); { x[i] = cc }
```

```

begin
  if (y[i] = ss) and (y[i-1] = cc) and (y[i+1] = cc) then inc(di);
end;
procedure TEFI;
var i,j: integer;
begin
  dt := 0; di := 0; dnt := 0; dnd := 0;
  fillchar(x,sizeof(x),cc);
  assign(f,fn); reset(f); readln(f,n,m);
  for j := 1 to n do
    begin
      readln(f,y); y := cc + y + cc;
      for i := 2 to m do
        begin
          if (x[i] = ss) then TEF(i) else II(i);
        end;
      x := y;
    end;
    close(f);
    i := dnt div 2; { de + df } de := dnd - i; df := i - de;
end;
BEGIN
  TEFI;
  writeln('T = ',dt,' E = ',de,' F = ',df,' I = ', di);
  readln;
END.

```

```

// DevC++: TEFI.CPP
#include <string.h>
#include <fstream>
#include <iostream>
#include <stdio.h>
using namespace std;
const char * fn = "tefi.inp";
const char * gn = "tefi.out";
string x, y;
char ss = '*', cc = '.';
int n, m;
int dt, de, df, di, dnt, dnd;
// P R O T O T Y P E S
void TEFI();
void TEF(int);
void EF(int);
void II(int);
// I M P L E M E N T A T I O N
int main(){
  TEFI();
  cout << endl << " T: " << dt << " E: " << de;
  cout << " F: " << df << " I: " << di;
  cout << endl << endl << " Fini ";// 3
  cin.get();
  return 0;
}
void TEF(int i) {
  if (y[i] == cc) return;
  if (x[i-1] == cc) EF(i);
  else ++dt;
}
void EF(int i){
  if (x[i+1] == ss) ++dnt;
  else if (y[i+1] == ss) ++dnd;
}

```

```

void II(int i) {
    if (y[i] == ss && y[i-1] == cc && y[i+1] == cc) ++di;
}
void TEFI() {
    int i, j;
    dt = di = dnt = dnd = 0;
    ifstream f(fn);
    f >> n >> m; f.get();
    x = cc;
    for (i = 0; i < 8; ++i) x = x + x;
    for (j = 0; j < n; ++j) {
        f >> y; y = cc + y + cc;
        for (i = 1; i <= m; ++i)
            if (x[i] == ss) TEF(i);
            else II(i);
        x = y;
    }
    f.close();
    i = dnt / 2; // i = de + df
    de = dnd - i; df = i - de;
}

```

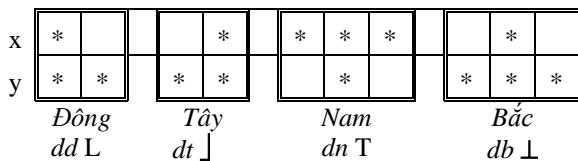
### 1.9 E xiéc

EXIEC.INP	EXIEC.OUT
15 27 ..... ***.....****. *.....*.....*.. ***.....***.. *.....*****.. *.....*....*....*..... ***..*.*.*.*.*..... .....*.*.*.*.*..... .....*....*....*.. .....*****....*****.. ..****.....*.. .....*.....****.. ..****..*....*.. .....*....*....*.. ..****..*****....*.. 	2 2 1 2

Trong text file tên EXIEC.INP gồm  $n$  dòng và  $m$  cột người ta dùng dấu chấm '.' tạo thành một bảng nền. Trên bảng nền đó người ta dùng dấu sao '\*' để viết các chữ IN HOA E với nét ngang giữa hướng về phía Đông, Tây, Nam và Bắc. Qui tắc viết giống như trong bài TEFI. Hãy đếm số chữ cái mỗi loại.

Thí dụ bên cho biết có 2 chữ E (nét ngang hướng về phía Đông), 2 chữ Ǝ (nét ngang hướng về phía Tây). 1 chữ E úp (nét ngang hướng về phía Nam), và 2 chữ E ngửa (nét ngang hướng về phía Bắc).

### Thuật toán



*Đặc trưng của các chữ E xiéc. Các biến dd, dt, dn, db dùng để đếm số lần xuất hiện của các đặc trưng.*

Hai E Nam và E Bắc được đặc trưng duy nhất qua hướng của nét ngang giữa giống chữ T (E Nam) và L (E Bắc). Mỗi E Đông có 2 nét dạng L và mỗi E Tây có 2 nét dạng J. Ngoài ra, mỗi chữ E Bắc còn có 1 nét L và 1 nét J. Ta có

$$\text{Số chữ E Đông} = (dd - db)/2,$$

$$\text{Số chữ E Tây} = (dt - db)/2,$$

$$\text{Số chữ E Nam} = dn,$$

$$\text{Số chữ E Bắc} = db.$$

*Độ phức tạp.* cõ m.n = dung lượng input file.

```

(* EXIEC.PAS *)
uses crt;

```

```

const fn = 'Exiec.inp'; gn = 'Exiec.out';
bl = #32; nl = #13#10; ss = '*'; cc = '.';
var x, y: string;
dd, dt, dn , db: integer;
n, m: integer;
f,g: text;
procedure TayNam(i: integer);
begin
  if (y[i-1] = ss) then inc(dft)
    else if (x[i-1] = ss) and (x[i+1] = ss) then inc(dn)
end;
procedure DongBac(i: integer);
begin
  if (y[i-1] = ss) then inc(db)
    else inc(dd);
end;
procedure DongTayNamBac(i: integer);
begin
  if (y[i+1] = ss) then DongBac(i) else TayNam(i);
end;
procedure EXIEC;
var i,j: integer;
begin
  dd := 0; dt := 0; dn := 0; db := 0;
  assign(f,fn); reset(f); readln(f,n,m);
  fillchar(x,sizeof(x),cc);
  for j := 1 to n do
  begin
    readln(f,y); y := cc + y + cc;
    for i := 2 to m do
      if (x[i] = ss) and (y[i] = ss) then DongTayNamBac(i);
    x := y;
  end;
  close(f);
  dd := (dd-db) div 2; dt := (dt-db) div 2;
end;
BEGIN
  EXIEC;
  writeln(dd,' ',dt, ' ', dn, ' ', db); readln;
END.

```

```

// DevC++: EXIEC.CPP
#include <string.h>
#include <fstream>
#include <iostream>
#include <stdio.h>
using namespace std;
const char * fn = "exiec.inp";
const char * gn = "exiec.out";
string x, y;
char ss = '*', cc = '.';
int n, m;
int dd, dt, dn, db; // huong tro cua vach giua: Dong Tay Nam Bac
// P R O T O T Y P E S
void EXIEC();
void DongTayNamBac(int);
void DongBac(int);
void TayNam(int);
// I M P L E M E N T A T I O N
int main(){
  EXIEC();
  cout << endl << dd << " " << dt;
  cout << " " << dn << " " << db;

```

## Chương 2 Xử lí dãy lệnh và biểu thức

### 2.1 Val

Cho các biến được gán trị  $a = 0, b = 1, c = 2, \dots, z = 25$ . Tính trị của biểu thức số học được viết đúng cú pháp, chứa các tên biến, các phép toán  $+, -, *, /$  (chia nguyên) và các cặp ngoặc  $()$ .

Thí dụ, biểu thức,  $(b+c)*(e-b) + (y-x)$  sẽ có giá trị  $(1+2)*(4-1) + (24-23) = 3*3+1 = 10$ .

#### Thuật toán

Do phải ưu tiên thực hiện các phép toán nhân (\*) và chia (/) trước các phép toán cộng (+) và trừ (-), ta qui ước các phép toán nhân và chia có bậc cao hơn bậc của các phép toán cộng và trừ. Gọi s là string chứa biểu thức, ta duyệt lần lượt từng kí tự s[i] của s và sử dụng hai ngăn xếp v và c để xử lí các tình huống sau:

1. Nếu  $s[i]$  là biến 'a', 'b', ... thì ta nạp trị tương ứng của biến đó vào ngăn xếp (stack) v.
2. Nếu  $s[i]$  là dấu mở ngoặc '(' thì ta nạp dấu đó vào ngăn xếp c.
3. Nếu  $s[i]$  là các phép toán '+', '−', '\*', '/' thì ta so sánh bậc của các phép toán này với bậc của phép toán p trên ngọn ngăn xếp c.
  - 3.1. Nếu  $Bac(s[i]) \leq Bac(p)$  thì ta lấy phép toán p ra khỏi ngăn xếp c và thực hiện phép toán đó với 2 phần tử trên cùng của ngăn xếp v. Bước này được lặp đi lặp lại khi  $Bac(s[i]) > Bac(p)$ . Sau đó làm tiếp bước 3.2.
  - 3.2. Nạp phép toán s[i] vào ngăn xếp c.
4. Nếu  $s[i]$  là dấu đóng ngoặc ')' thì ta dỡ dần và thực hiện các phép toán trên ngọn ngăn xếp c cho đến khi gặp dấu '(' đã nạp trước đó.

Thuật toán được xây dựng trên giả thiết biểu thức s được viết đúng cú pháp. Về bản chất, thuật toán xử lý và tính toán đồng thời trị của biểu thức s theo nguyên tắc *phép toán sau* hay là *kí pháp Ba Lan* do nhà toán học Ba Lan Lucasiewics đề xuất. Theo kí pháp này, biểu thức  $(b+c)*(e-b) + (y-x)$  sẽ được viết thành  $bc+eb-*yx-+$  và được thực hiện trên ngăn xếp v như sau. Gọi iv là con trỏ ngọn của ngăn xếp v, iv được khởi trị 0:

1. Nạp trị của biến b vào ngăn xếp v: iv := iv + 1; v[iv] := (b); trong đó (b) là trị của biến b.
2. Nạp trị của biến c vào ngăn xếp v: iv := iv + 1; v[iv] := (c);
3. Thực hiện phép cộng hai phần tử trên ngọn ngăn xếp v, ghi kết quả vào ngăn dưới, bỏ ngăn trên:  $v[iv-1] := v[iv-1] + v[iv]; iv := iv - 1;$
4. Nạp trị của e vào ngăn xếp v: iv := iv + 1; v[iv] := (e);
5. Nạp trị của b vào ngăn xếp v: iv := iv + 1; v[iv] := (b);
6. Thực hiện phép trừ hai phần tử trên ngọn ngăn xếp v, ghi kết quả vào ngăn dưới, bỏ ngăn trên:  $v[iv-1] := v[iv-1] - v[iv]; iv := iv - 1;$
7. Thực hiện phép nhân hai phần tử trên ngọn ngăn xếp v, ghi kết quả vào ngăn dưới, bỏ ngăn trên:  $v[iv-1] := v[iv-1] * v[iv]; iv := iv - 1;$
8. Nạp trị của y vào ngăn xếp v: iv := iv + 1; v[iv] := (y);
9. Nạp trị của x vào ngăn xếp v: iv := iv + 1; v[iv] := (x);
10. Thực hiện phép trừ hai phần tử trên ngọn ngăn xếp v, ghi kết quả vào ngăn dưới, bỏ ngăn trên:  $v[iv-1] := v[iv-1] - v[iv]; iv := iv - 1;$
11. Thực hiện phép cộng hai phần tử trên ngọn ngăn xếp v, ghi kết quả vào ngăn dưới, bỏ ngăn trên:  $v[iv-1] := v[iv-1] + v[iv]; iv := iv - 1;$

Kết quả cuối cùng có trong v[iv].

Bạn nhớ khởi trị ngăn xếp c bằng kí tự nào đó không có trong biểu thức, thí dụ '#'. Phép toán này sẽ có bậc 0 và dùng làm phần tử đệm để xử lý tình huống 3.

Bạn cần đặt kí hiệu # vào đáy của ngăn xếp c để làm lính canh. Vì khi quyết định có nạp phép toán p nào đó vào ngăn xếp c ta cần so sánh bậc của p với bậc của phép toán trên ngọn của ngăn xếp c. Như vậy # sẽ có bậc 0. Bạn có thể thêm một phép kiểm tra để phát hiện lỗi "chia cho 0" khi thực hiện phép chia. Bạn cũng có thể phát triển thêm chương trình để có thể xử lí các biểu thức có chứa các phép toán một ngôi !, ++, --, ... và các lời gọi hàm.

**Độ phức tạp.** cõi n, trong đó n là số kí hiệu trong biểu thức.

```

uses crt;
const fn = 'val.inp'; gn = 'val.out';
nl = #13#10; bl = #32; mn = 500;
var
  c: array[0..mn] of char; {Ngăn xếp c chứa các phép toán}
  ic: integer;
  v: array[0..mn] of integer; {Ngăn xếp v chứa trị của các biến}
  iv: integer;
  function LaBien(c: char): Boolean;
begin LaBien := (c in ['a'..'z']); end;
  function LaPhapToan(c: char): Boolean;
begin LaPhapToan := (c in ['+', '-', '*', '/']) end;
  function Val(c: char): integer; { trị của biến c }
begin Val := Ord(c)-ord('a'); end;
  function Bac(p: char): integer; { Bậc của phép toán p }
begin
  if (p in ['+', '-']) then Bac := 1
  else if (p in ['*', '/']) then Bac := 2
  else Bac := 0;
end;
(* Thực hiện phép toán 2 ngôi trên ngọn ngăn xếp v *)
procedure Tinh(p: char);
begin
  case p of
    '+': begin v[iv-1] := v[iv-1] + v[iv]; dec(iv) end;
    '-': begin v[iv-1] := v[iv-1] - v[iv]; dec(iv) end;
    '*': begin v[iv-1] := v[iv-1] * v[iv]; dec(iv) end;
    '/': begin v[iv-1] := v[iv-1] div v[iv]; dec(iv) end;
  end
end;
procedure XuLiToan(p: char);
begin
  while (Bac(c[ic]) >= Bac(p)) do
    begin Tinh(c[ic]); dec(ic) end;
    inc(ic); c[ic] := p; { nạp phép toán p }
end;
procedure XuLiNgoac;
begin
  while (c[ic] <> '(') do begin Tinh(c[ic]); dec(ic) end;
  dec(ic); { Bỏ ngoặc }
end;
function XuLi(s: string): integer;
  var i: integer;
begin
  ic := 0; c[ic] := '#'; iv := -1;
  for i := 1 to length(s) do
    if LaBien(s[i]) then begin inc(iv); v[iv] := Val(s[i]) end
    else if s[i] = '(' then begin inc(ic); c[ic] := '(' end
    else if LaPhapToan(s[i]) then XuLiToan(s[i])
    else if s[i] = ')' then XuLiNgoac;
    while (ic > 0) do begin Tinh(c[ic]); dec(ic) end;
    XuLi := v[iv];
end;
BEGIN
  writeln(nl, XuLi('(b+c)*(f-a+b-c+d)/(c*d+b)'), { 3 })
  readln;
END.

```

```

// DevC++: Val
#include <string.h>
#include <fstream>
#include <iostream>

```

```

#include <stdio.h>
using namespace std;
// Mo ta Du lieu va bien
const int mn = 500;
char s[mn]; // bieu thuc
char c[mn]; //ngan xep phep toan va dau (
int ic; // con tro ngan xep c
int v[mn]; //ngan xep tinh toan
int iv; // con tro ngan xep v
int kq; // ket qua
int n; // len - so ki tu trong bieu thuc
// Khai bao cac ham
int XuLi();
bool LaBien(char c); // kiem tra c la bien ?
bool LaPhepToan(char c); // kiem tra c la phep toan +, -, *, / ?
void XuLiPhepToan(char pt);
void XuLiNgoac();
int Bac(char pt); // Bac cua phep toan +, - (1), *, / (2)
int Val(char c); // Tinh tri cua bien c
void Tinh(char pt); // thuc hien phep toan pt

// Cai dat

int main() {
    strcpy(s,"(b+c)*(e-b) + (y-x)");
    cout << endl << " input: " << s;
    kq = XuLi();
    cout << endl << endl << " Dap so: " << kq << endl ;
    cout << endl << endl << " Fini" << endl;
    cin.get();
    return 0;
}
int XuLi() {
    ic = 0; c[ic] = '#'; n = strlen(s); iv = -1;
    int i;
    for (i = 0; i < n; ++i)
        if (LaBien(s[i])) v[++iv] = Val(s[i]);
        else if (s[i]=='(') c[++ic] = '(';
        else if (s[i]==')') XuLiNgoac();
        else if (LaPhepToan(s[i])) XuLiPhepToan(s[i]);
    while (LaPhepToan(c[ic])) { Tinh(c[ic]); --ic; }
    return v[iv];
}
// Val('A') = 0; Val('B') = 1; . . .
int Val(char c) { return (int)(c-'a'); }
int Bac(char pt) {
    if (pt == '+' || pt == '-') return 1;
    else if (pt == '*' || pt == '/') return 2;
    else return 0;
}
bool LaBien(char c) { return (c >= 'a' && c <= 'z'); }
bool LaPhepToan(char c) {return(c=='+'||c=='-'||c=='*'||c=='/');}
void XuLiPhepToan(char pt) {
    while (Bac(c[ic]) >= Bac(pt)) { Tinh(c[ic]); --ic; }
    c[++ic] = pt;
}
void XuLiNgoac(){
    while (c[ic] != '(') { Tinh(c[ic]); --ic; }
    --ic; // bo dau '('
}
void Tinh(char pt) { // Thuc hien phep toan pt
    switch(pt) {
        case '+': v[iv-1] = v[iv-1]+v[iv]; --iv; break;

```

```

        case '-': v(iv-1) = v(iv-1)-v(iv); --iv; break;
        case '*': v(iv-1) = v(iv-1)*v(iv); --iv; break;
        case '/': v(iv-1) = v(iv-1)/v(iv); --iv; break;
    }
}

```

## 2.2 Xâu thu gọn

Một xâu chỉ gồm các chữ cái A, B, C,...,Z có thể được viết gọn theo các quy tắc sau:

1.  $Xm$  – gồm  $m$  chữ cái  $X$ ;
2.  $(S)m$  – gồm  $m$  lần viết xâu thu gọn  $S$ .

Nếu  $m = 0$  thì đoạn cần viết sẽ được bỏ qua, nếu  $m = 1$  thì có thể không viết  $m$ . Thí dụ,  $(AB3(C2D)2(C5D)0)2A3$  là xâu thu gọn của xâu  $ABBBCCDCCDABBBCCDCCDAA$ .

Cho xâu thu gọn  $s$ . Hãy viết dạng đầy đủ (còn gọi là dạng khai triển) của xâu nguồn sinh ra xâu thu gọn  $s$ . Trong xâu thu gọn có thể chứa các dấu cách nhưng các dấu cách này được coi là vô nghĩa và do đó không xuất hiện trong xâu nguồn.

### Thuật toán

Ta triển khai theo kỹ thuật *hai pha*. Pha thứ nhất: Duyệt xâu  $s$  và tạo ra một chương trình  $P$  phục vụ cho việc viết dạng khai triển ở pha thứ hai. Pha thứ hai: Thực hiện chương trình  $P$  để tạo ra xâu nguồn.

**Pha thứ nhất:** Duyệt từng ký tự  $s[v]$  và quyết định theo các tình huống sau:

- Nếu  $s[v]$  là chữ cái  $C$  thì đọc tiếp số  $m$  sau  $C$  và tạo ra một dòng lệnh mới dạng  $(n, C, m)$ , trong đó  $n$  là số hiệu riêng của dòng lệnh,  $C$  là chữ cái cần viết,  $m$  là số lần viết chữ cái  $C$ ;
- Nếu  $s[v]$  là dấu mở ngoặc '(' thì ghi nhận vị trí dòng lệnh  $n+1$  vào stack  $st$ ;
- Nếu  $s[v]$  là dấu đóng ngoặc ')' thì đọc tiếp số  $m$  sau ngoặc, lấy giá trị  $t$  từ ngọn ngắn xếp và tạo ra một dòng lệnh mới dạng  $(n, #, m, t)$ . Dòng lệnh này có ý nghĩa như sau: Cần thực hiện lặp  $m$  lần đoạn trình từ dòng lệnh  $t$  đến dòng lệnh  $n$ . Nếu số  $m = 0$  thì ta xóa các dòng lệnh từ dòng  $t$  đến dòng hiện hành  $n$ . Nếu  $n = 1$  thì ta không tạo dòng lệnh mới.

Với thí dụ đã cho, sau pha 1 ta sẽ thu được chương trình  $P$  gồm các dòng lệnh như sau:

n	c	m	t	Ý nghĩa của dòng lệnh
1	A	1		Viết A 1 lần
2	B	3		Viết B 3 lần
3	C	2		Viết C 2 lần
4	D	1		Viết D 1 lần
5	#	2	3	Lặp 2 lần từ dòng lệnh 3 đến dòng lệnh 5
6	#	2	1	Lặp 2 lần từ dòng lệnh 1 đến dòng lệnh 6
7	A	3		Viết A 3 lần

Chương trình  $P$  gồm 7 dòng lệnh thu được sau khi thực hiện Pha 1 với xâu  $(AB3(C2D)2(C5D)0)2A3$ .

**Pha thứ hai:** Thực hiện chương trình  $P$ .

Ta thực hiện từng dòng lệnh của chương trình từ dòng 1 đến dòng  $n$ . Với thí dụ đã cho, sau khi thực hiện 4 dòng lệnh đầu tiên ta thu được kết quả  $ABBBCCD$ . Tiếp đến dòng lệnh 5 cho ta biết cần thực hiện việc lặp 2 lần các lệnh từ dòng 3 đến dòng 5. Sau mỗi lần lặp ta giảm giá trị của cột  $m$  tương ứng. Khi  $m$  giảm đến 0 ta cần khôi phục lại giá trị cũ của  $m$ . Muốn vậy ta cần thêm một cột nữa cho bảng. Cột này chứa giá trị ban đầu của  $m$  để khi cần ta có thể khôi phục lại. Ta sẽ gọi cột này là  $R$ . Theo giải trình trên, sau khi thực hiện dòng 5 ta thu được xâu  $ABBBCCDABBBCCD$ . Với dòng lệnh 6, lập luận tương tự ta thu được xâu  $ABBBCCDABBBCCDABBBCCD$

Cuối cùng, sau khi thực hiện dòng lệnh 7 ta thu được kết quả

$ABBBCCDABBBCCDAAA$

**Độ phức tạp** Cõi  $n$ , trong đó  $n$  là số ký tự trong xâu input.

```

(* XauGon.pas *)
uses crt;
const mn = 500; BL = #32; NL = #13#10;
ChuSo = ['0'..'9']; ChuCai = ['A'..'Z'];
type mil = array[0..mn] of integer;
      mc1 = array[0..mn] of char;
var M, T, R, st: mil; { M: so lan lap; T: tu; R: luu, st: stack }
c: mc1; { Lenh }
p: integer; { nghan stack }
s: string;
v: integer; { chi dan cua s }

```

```

n: integer; { so dong lenh }
procedure Cach; begin while (s[v] = BL) do inc(v); end;
function DocSo: integer;
  var so: integer;
begin so := 0; Cach;
  if Not (s[v] in ChuSo) then begin DocSo := 1; exit; end;
  while (s[v] in ChuSo) do
  begin
    so := so*10 + (Ord(s[v]) - ord('0'));
    inc(v);
  end;
  DocSo := so;
end;
procedure LenhDon(ch: char);
  var so: integer;
begin
  inc(v); so := DocSo;
  if so = 0 then exit;
  inc(n); C[n] := ch; M[n] := so;
end;
procedure NapNgoac;
begin inc(v); inc(p); st[p] := n+1 end;
procedure LenhLap;
  var tu, so: integer;
begin
  inc(v); tu := st[p]; dec(p);
  so := DocSo;
  if (so = 0) then n := tu-1;
  if (so < 2) then exit;
  inc(n); C[n] := '#'; M[n] := so; T[n] := tu; R[n] := so;
end;
procedure Pha2;
  var i,j: integer;
begin
  for i := 1 to n do
  begin
    write(NL,i,'.',C[i],BL,M[i],BL);
    if C[i] = '#' then write(T[i],BL,R[i]);
  end;
  i := 1;
  while (i <= n) do
  begin
    if (C[i] = '#') then
    begin
      dec(R[i]);
      if (R[i] = 0) then begin R[i] := M[i]; inc(i) end
      else i := T[i];
    end
    else
    begin
      for j := 1 to M[i] do write(C[i]);
      inc(i);
    end;
  end;
end;
procedure KhaiTrien(var s: string);
var i: integer;
begin
  s := s + '#'; v := 1; p := 0;
  while (s[v] <> '#') do
  begin
    if (s[v] in ChuCai) then LenhDon(s[v])
    else if (s[v] = '(') then NapNgoac

```

```

    else if (s[v] = ')') then LenhLap
        else inc(v);
    end;
    write(NL,s , ' = ');
    Pha2;
end;
BEGIN
    s := ' (AB3(C2D)2(C5D)0)2A3';
    KhaiTrien(s);
    readln;
END.

// DevC++: XauGon.cpp
#include <string.h>
#include <fstream>
#include <iostream>
using namespace std;
// DATA AND VARIABLE
const int mn = 500;
const char BL = 32;
char C[mn]; // noi dung lenh
int M[mn]; // so lan lap
int T[mn]; // lap tu dong lenh nao
int R[mn]; // luu gia tri M
int n; // con dem dong lenh
char s[mn]; // xau thu gon
int v; // chi so duyet s
int st[mn]; // stack
int p; // chi so ngon stack st
// PROTOTYPES
void KhaiTrien(char *);
void LanhDon(char);
void LenhLap();
int DocSo();
void Cach();
bool LaChuSo(char);
bool LaChuCai(char);
void Pha2();
// IMPLEMENTATION
int main() {
    strcpy(s, "(AB3(C2D)2(C5D)0)2A3");
    KhaiTrien(s);
    cout << endl; system("PAUSE");
    return EXIT_SUCCESS;
}
bool LaChuCai(char c) { return (c >= 'A') && (c <= 'Z'); }
bool LaChuSo(char c) { return (c >= '0') && (c <= '9'); }
void Cach() { while (s[v] == BL) ++v; }
int DocSo() {
    int so = 0;
    Cach();
    if (!LaChuSo(s[v])) return 1;
    while (LaChuSo(s[v])) {
        so = so*10 + int(s[v]-'0'); ++v;
    }
    return so;
}
void LanhDon(char ch) {
    int so;
    ++v; so = DocSo();
    if (so == 0) return;
    ++n; C[n] = ch; M[n] = so;
}

```

```

    }
    void LenhLap() {
        int so;
        ++v; // bo qua dau )
        so = DocSo();
        int tu = st[p--];
        if (so == 0) { n = tu-1; return; }
        if (so == 1) return;
        ++n; C[n] = '#'; M[n] = R[n] = so; T[n] = tu;
    }
    void KhaiTrien(char *s ) {
        // Pha1
        p = 0; n = 0; // init
        for (v = 0; s[v];) {
            if (LaChuCai(s[v])) LenhDon(s[v]);
            else if (s[v] == '(') { st[++p] = n+1; ++v; }
            else if (s[v] == ')') LenhLap();
            else ++v;
        }
        Pha2();
    }
    void Pha2() {
        int i, j;
        cout << endl << s << " = ";
        i = 1;
        while (i <= n) {
            if (C[i] == '#') {
                --R[i];
                if (R[i] == 0) { R[i] = M[i]; ++i; }
                else i = T[i];
            }
            else {
                for (j = 1; j <= M[i]; ++j) cout << C[i];
                ++i;
            }
        }
    }
}

```

### 2.3 Robot

Một Robot được lập trình để di chuyển trên mặt phẳng tọa độ xoy chia lưới đơn vị. Chương trình điều khiển Robot được viết dưới dạng xâu gọn như trong bài Xâu gọn và gồm dãy lệnh với ý nghĩa như sau:

$Gn$  – đi thẳng  $n$  bước qua các điểm nguyên,

$Rn$  – quay phải  $n$  lần  $45^\circ$ ,

$Ln$  – quay trái  $n$  lần  $45^\circ$ .

Robot được đặt tại vị trí xuất phát là góc tọa độ, mặt hướng theo trục oy.

Yêu cầu: Xác định tọa độ  $(x,y)$  nơi Robot dừng chân sau khi thực hiện chương trình ghi trong string s.

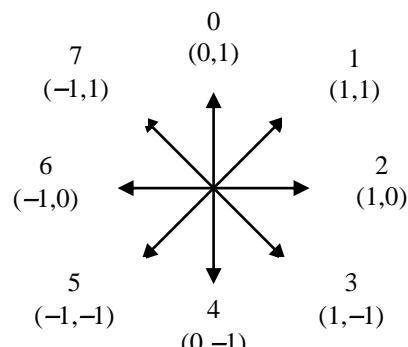
Thí dụ, Sau khi thực hiện chương trình  $s = "(GR3(G2L)2(L5G)0)2G3"$  Robot sẽ dừng chân tại vị trí  $(10,-4)$  trên mặt phẳng tọa độ.

#### Thuật toán

Pha 1 hoàn toàn giống bài Xâu thu gọn. Riêng với pha 2 ta cần thay lệnh hiển thị bằng việc tính vị trí của Robot sau khi thực hiện mỗi dòng lệnh.

Ta mã số 8 hướng di chuyển trên mặt phẳng tọa độ từ 0 đến 7. Với mỗi hướng ta xác định các giá trị dx và dy khi cho Robot đi 1 bước theo hướng đó. Thí dụ, theo hướng  $h = 0$  thì Robot sẽ di chuyển từ tọa độ  $(x,y)$  sang tọa độ  $(x, y + 1)$ , như vậy  $dx = 0$ ,  $dy = 1$ .

Các giá trị này được khởi tạo sẵn trong một mảng hai chiều huong trong đó  $dx = huong[h][0]$ ,  $dy = huong[h][1]$ .



Vì có 8 hướng nên nếu Robot đang hướng mặt về hướng h thì sau khi quay phải k lần hướng mặt của Robot sẽ là  $h = (h+k) \bmod 8$ , còn khi quay trái k lần ta sẽ có  $h = (h + n - (k \bmod 8)) \bmod 8$ . Bạn để ý rằng phép trừ k đơn vị trên vòng tròn n điểm sẽ được đổi thành phép cộng với  $n-k$ .

**Độ phức tạp**  $C \propto n$ , trong đó  $n$  là số kí tự trong xâu input.

```
(* Robot.pas *)
uses crt;
const mn = 500; BL = #32; NL = #13#10; xx = 0; yy = 1;
huong: array[0..7,0..1] of integer
      = ( (0,1), (1,1), (1,0), (1,-1),
          (0,-1), (-1,-1), (-1,0), (-1,1) );
ChuSo = ['0'..'9']; ChuCai = ['A'..'Z'];
type mil = array[0..mn] of integer;
         mc1 = array[0..mn] of char;
var M, T, R, st: mil; { M: so lan lap; T: tu; R: luu, st: stack }
c: mc1; { Lenh }
p: integer; { ngon stack }
s: string;
v: integer; { chi dan cua s }
n: integer; { so dong lenh }
x,y: integer; { Toa do Robot }
h: integer; { huong di chuyen cua Robot }
procedure Cach; begin while (s[v] = BL) do inc(v); end;
function DocSo: integer;
begin so := 0; Cach;
  if Not (s[v] in ChuSo) then begin DocSo := 1; exit; end;
  while (s[v] in ChuSo) do
  begin
    so := so*10 + (Ord(s[v]) - ord('0'));
    inc(v);
  end;
  DocSo := so;
end;
procedure LanhDon(ch: char);
begin so := 0;
begin
  inc(v); so := DocSo;
  if so = 0 then exit;
  inc(n); C[n] := ch; M[n] := so;
end;
procedure NapNgoac;
begin inc(v); inc(p); st[p] := n+1 end;
procedure LanhLap;
begin tu, so: integer;
begin
  inc(v); tu := st[p]; dec(p);
  so := DocSo;
  if (so = 0) then n := tu-1;
  if (so < 2) then exit;
  inc(n); C[n] := '#'; M[n] := so; T[n] := tu; R[n] := so;
end;
procedure ThucHien(i: integer);
begin
  case C[i] of
    'G': begin x:=x+M[i]*huong[h,xx];y:=y+M[i]*huong[h,yy] end;
    'R': h := (h + M[i]) mod 8;
    'L': h := (h + 8 - (M[i] mod 8)) mod 8;
  end;
end;
```

```

    end;
end;
procedure Pha2;
var i: integer;
begin
  x := 0; y := 0; h := 0;
  for i := 1 to n do
  begin
    write(NL,i,'.',C[i],BL,M[i],BL);
    if C[i] = '#' then write(T[i],BL,R[i]);
  end;
  i := 1;
  while (i <= n) do
  begin
    if (C[i] = '#') then
    begin
      dec(R[i]);
      if (R[i] = 0) then begin R[i] := M[i]; inc(i) end
      else i := T[i];
    end
    else
    begin
      ThucHien(i); { thuc hien dong lenh i }
      inc(i);
    end;
  end;
end;
procedure Go(var s: string);
begin
  s := s + '#'; v := 1; p := 0;
  while (s[v] <> '#') do
  begin
    if (s[v] in ChuCai) then LenhDon(s[v])
    else if (s[v] = '(') then NapNgoac
    else if (s[v] = ')') then LenhLap
    else inc(v);
  end;
  write(NL,s , ' = ');
  Pha2;
end;
BEGIN
  s := '(GR3(G2L)2(L5G)0)2G3';
  Go(s);
  writeln(NL,NL,'Ket qua (x,y) = ',x,BL,y); { (x,y) = (10,-4) }
  readln;
END.

// DevC++: Robot.cpp
#include <string.h>
#include <fstream>
#include <iostream>
using namespace std;
// DATA AND VARIABLE
const int mn = 500;
const char BL = 32;
char C[mn]; // noi dung lenh
int M[mn]; // so lan lap
int T[mn]; // lap tu dong lenh nao
int R[mn]; // luu gia tri M
int n; // con dem dong lenh
char s[mn]; // xau thu gon
int v; // chi so duyet s
int st[mn]; // stack

```

```

int p; // chi so ngon stack st
int x, y; // Toa do (x,y) cua Robot
const int xx = 0, yy = 1;
int step[8][2] = {{0,1},{1,1},{1,0},{1,-1},
                  {0,-1},{-1,-1},{-1,0},{-1,1}};
int h; // huong di chuyen cua Robot
// P R O T O T Y P E S
void Go(char *);
void LenhDon(char);
void LenhLap();
int DocSo();
void Cach();
bool LaChuSo(char);
bool LaChuCai(char);
void Pha2();
void ThucHien(int);
// I M P L E M E N T A T I O N
int main(){
    strcpy(s,"(GR3(G2L)2(L5G)0)2G3"); // (x,y) = (10,-4)
    Go(s);
    cout << endl << x << " " << y;
    cout << endl; system("PAUSE");
    return EXIT_SUCCESS;
}
bool LaChuCai(char c) { return (c >= 'A') && (c <= 'Z'); }
bool LaChuSo(char c) { return (c >= '0') && (c <= '9'); }
void Cach() { while (s[v] == BL) ++v; }
int DocSo() {
    int so = 0;
    Cach();
    if (!LaChuSo(s[v])) return 1;
    while (LaChuSo(s[v])) {
        so = so*10 + int(s[v]-'0'); ++v;
    }
    return so;
}
void LenhDon(char ch) {
    int so;
    ++v; so = DocSo();
    if (so == 0) return;
    ++n; C[n] = ch; M[n] = so;
}
void LenhLap() {
    int so;
    ++v; // bo qua dau
    so = DocSo();
    int tu = st[p--];
    if (so == 0) { n = tu-1; return; }
    if (so == 1) return;
    ++n; C[n] = '#'; M[n] = R[n] = so; T[n] = tu;
}
void Go(char *s ) {
    cout << endl << "input: " << s;
    // init
    p = 0; n = 0;
    for (v = 0; s[v];) {
        if (LaChuCai(s[v])) LenhDon(s[v]);
        else if (s[v] == '(') { st[++p] = n+1; ++v; }
        else if (s[v] == ')') LenhLap();
        else ++v;
    }
    Pha2();
}

```

```

void ThucHien(int i) {
    switch(C[i]) {
        case 'G': x += M[i]*step[h][xx]; y += M[i]*step[h][yy]; break;
        case 'R': h = (h+M[i]) % 8; break;
        case 'L': h = (h+8-(M[i]%8)) % 8; break;
    }
}
void Pha2() {
    int i;
    cout << endl << s << " = ";
    x = y = 0; h = 0; i = 1;
    while (i <= n) {
        if (C[i] == '#') {
            --R[i];
            if (R[i] == 0) { R[i] = M[i]; ++i; }
            else i = T[i];
        }
        else {
            ThucHien(i); // thuc hien dong lenh i
            ++i;
        }
    }
}

```

## 2.4 Hàm nhiều biến

Một số hàm có số tham biến không hạn chế,

Thí dụ 1: Hàm ucln – tính ước chung lớn nhất của các số nguyên được định nghĩa như sau:

$$ucln() = 0, \text{ không có tham biến, qui ước} = 0$$

$$ucln(x) = |x|, ucln \text{ của số } x \text{ là giá trị tuyệt đối của chính số đó}$$

$$ucln(a,b) = ucln(b,a),$$

$$ucln(a,0) = a,$$

$$ucln(a,b) = ucln(a \bmod b, b)$$

$$ucln(x_1, x_2, \dots, x_n) = ucln(ucln(x_1, x_2, \dots, x_{n-1}), |x_n|), n \geq 2.$$

Thí dụ 2: Hàm sum – tính tổng của các số nguyên:

$$sum() = 0,$$

$$sum(x) = x,$$

$$sum(x_1, x_2, \dots, x_n) = sum(sum(x_1, x_2, \dots, x_{n-1}), x_n), n \geq 2.$$

Ngoài ra còn các hàm lấy min, max của dãy phần tử...

Cho một biểu thức được viết đúng cú pháp, chĩa các hằng nguyên, các biến  $a, b, \dots$  được gán sẵn các trị  $a = 0, b = 1, \dots$ , các phép toán số học  $+, -, *, /$  (chia nguyên),  $\%$  (chia dư), các cặp ngoặc và các lời gọi hàm nhiều biến  $@$ . Hãy tính giá trị của biểu thức nếu  $@$  là hàm  $ucln$ .

Thí dụ,  $16$  sẽ là giá trị của biểu thức  $(10 + @((12, 30 + @((6, 8)) + 17 * @() + 2) * @((1, 3)))$ . Thật vậy, ta có  $@() = 0; @((6, 8)) = 2; @((12, 30 + @((6, 8))) = @((12, 30 + 2)) = @((12, 32)) = 4; @((1, 3)) = 1; (10 + @((12, 30 + @((6, 8)) + 17 * @() + 2) * @((1, 3))) = (10 + 4 + 17 * 0 + 2) * 1 = 16 * 1 = 16$ .

## Thuật toán

Ta mở rộng thuật toán của bài Val để có thể xử lý thêm các trường hợp sau. Thứ nhất, chương trình phải nhận biết được phép toán đảo dấu. Đây là phép toán 1 ngôi khác với phép trừ là phép toán 2 ngôi. Thí dụ, biểu thức  $-a + b$  có phép toán đảo dấu. Phép này cũng khá dễ nhận biết. Nếu gặp dấu  $-$  và trong ngọn của ngăn xếp  $c$  không chứa phép toán nào thì phép  $-$  này sẽ là phép toán đổi dấu. Ta nạp vào ngăn xếp  $c$  kí hiệu  $!$  cho phép đổi dấu nhằm phân biệt tường minh với phép toán trừ. Kỹ thuật này có thể gây nhập nhằng, thí dụ, khi xử lý biểu thức  $a - b$  thì dấu  $-$  gấp đầu tiên nên trong ngăn xếp  $c$  không chứa phép toán nào. Hệ thống sẽ coi là phép toán đổi dấu. Ta khắc phục tình huống này bằng cách sau. Sau khi thực hiện hết các phép toán trong ngăn xếp  $c$ , nếu trong ngăn xếp tính toán  $t$  còn hơn 1 phần tử thì ta cộng dồn kết quả vào  $t[1]$ . Như vậy ta đã giả thiết  $a - b = a + (-b)$  trong đó  $-$  là phép đổi dấu. Thứ hai, chương trình phải xử lý được các tình huống gọi hàm  $@$  với các tham biến khác nhau. Khi gặp kí hiệu  $@$  ta xác định xem giữa cặp ngoặc  $()$  có đối tượng nào không. Nếu không có, ta ghi nhận một lời gọi hàm rỗng trong ngăn xếp  $c$ . Trong danh sách tham biến của lời gọi hàm có thể chứa dấu phẩy dùng để ngăn cách các tham biến. Ta cũng nạp dàn các dấu ngón này vào ngăn xếp  $c$ . Thủ tục Cach bỏ qua các dấu cách trong xâu input  $s$ , tìm đến kí tự có nghĩa  $s[v]$  tiếp theo.

Với hai ngăn xếp c dùng để ghi nhận các dấu phép toán và t dùng để chứa các giá trị cần tính toán ta tổ chức đọc duyệt xâu input s và xử lí như sau.

1. Khởi trị các ngăn xếp,
2. Với mỗi kí tự s[v] ta xét

2.1 s[v] = '@': Nạp @ vào c; đọc tiếp s để xác định xem giữa cặp ngoặc () có kí hiệu nào không. Nếu không có: nạp thêm \$ vào c để ghi nhận lời gọi hàm rỗng;

2.2 s[v] = '(': Nạp ( vào c;

2.3 s[v] là chữ số '0'..'9': Đọc số này và nạp vào ngăn xếp t;

2.4 s[v] là tên biến (chữ cái 'a'..'z'): Nạp trị của các biến này vào ngăn xếp t. Trị của biến x được tính theo công thức x - 'a';

2.5 s[v] là dấu phẩy: Thực hiện các phép toán (nếu có) trên ngọn ngăn xếp c để tính trị của biểu thức ứng với tham số này. Thí dụ, s = "@(a+1,..." thì ta phải tính trị của a + 1 trước khi gọi hàm;

2.5 s[v] = ')': Ta cần xác định xem dấu ')' đóng một biểu thức con hay đóng danh sách tham biến của một lời gọi hàm. Trước hết ta thực hiện các phép toán (nếu có) trên ngọn ngăn xếp c để tính trị của biểu thức kết thúc bằng dấu ')'. Kế đến ta duyệt ngược ngăn xếp c để xác định vị trí của dấu ')'. Sát trước vị trí này có thể có dấu @ hoặc không. Nếu có ta tính hàm. Nếu sát sau ')' là kí hiệu '\$' thì ta hiểu là hàm rỗng, ta sinh trị 0 cho ngăn xếp t.

2.6 s[v] là dấu các phép toán +, -, \*, /, %: Ta thực hiện các phép toán bậc cao trên ngọn ngăn xếp c rồi nạp phép toán s[v] này vào c. Riêng với phép – ta cần xác định xem có phải là phép đảo dấu hay không.

```
(* Func.pas *)
uses crt;
const bl = #13#10; nl = #32; mn = 500;
var
  c: array[0..mn] of char; { ngan xep phep toan }
  ic: integer; { ngon ngan xep c }
  t: array[0..mn] of integer; { ngan xep tinh toan }
  it: integer; { ngon ngan xep t }
  s: string; { input }
  v: integer; { duyet s }
function Ucln(a,b: integer): integer;
var r: integer;
begin a := abs(a); b := abs(b);
  while b > 0 do
    begin r := a mod b; a := b; b := r; end;
    Ucln := a;
end;
procedure Cach;
begin while s[v] = bl do inc(v);
function LaBien(c: char): Boolean;
begin LaBien := (c in ['a'..'z']); end;
function LaChuSo(c: char): Boolean;
begin LaChuSo := (c in ['0'..'9']); end;
function LaPhepToan(c: char): Boolean;
begin LaPhepToan := (c in ['+', '-', '*', '/', '%', '!']) end;
function Val(c: char): integer;
begin Val := Ord(c)-ord('a'); end;
function Bac(p: char): integer;
begin
  if (p in ['+', '-']) then Bac := 1
  else if (p in ['*', '/', '%']) then Bac := 2
  else if (p = '!') then Bac := 3
  else Bac := 0;
end;
function DocSo: integer;
  var so: integer;
begin
  so := 0;
  while LaChuSo(s[v]) do
```

```

begin
  so := so*10 + Ord(s[v])-ord('0');
  inc(v);
end;
DocSo := so;
end;
procedure Tinh(p: char);
begin
  case p of
    '+': begin t[it-1] := t[it-1] + t[it]; dec(it) end;
    '-': begin t[it-1] := t[it-1] - t[it]; dec(it) end;
    '*': begin t[it-1] := t[it-1] * t[it]; dec(it) end;
    '/': begin t[it-1] := t[it-1] div t[it]; dec(it) end;
    '%': begin t[it-1] := t[it-1] mod t[it]; dec(it) end;
    '!!': begin t[it] := -t[it] end; { phép đảo dấu }
  end
end;
procedure Napc(ch: char); begin inc(ic); c[ic] := ch end;
procedure NapBien(x: char);
begin
  inc(v); inc(it); t[it] := Val(x);
end;
procedure NapSo;
var so: integer;
begin
  inc(it); t[it] := DocSo;
end;
procedure NapPhepToan(p: char);
begin
  inc(v);
  if p = '-' then
    if Not LaPhepToan(c[ic]) then begin Napc('!!'); exit end;
    while (Bac(c[ic]) >= Bac(p)) do begin Tinh(c[ic]); dec(ic) end;
    Napc(p);
  end;
procedure NapPhay;
begin
  inc(v);
  while LaPhepToan(c[ic]) do begin Tinh(c[ic]); dec(ic) end;
  Napc(',');
end;
procedure NapNgoac;
begin
  inc(v); Napc('()');
end;
procedure NapHam;
begin
  inc(v); Napc('@');
  Cach; NapNgoac; { bo qua ( )
  Cach; if s[v] = ')' then { Ham rong } Napc('$');
end;
procedure XuLiHam(i: integer);
  var j,kq: integer;
begin
  if c[i+1] = '$' then
  begin
    inc(it); t[it] := 0;
    ic := i - 2;
    exit
  end;
  kq := 0;
  for j := it-ic+i to it do kq := Ucln(kq,t[j]);
  it := it-ic+i; t[it] := kq;

```

```

    ic := i - 2;
end;
procedure XuLiNgoac; { gap }
  var i: integer;
begin
  inc(v);
  while LaPhepToan(c[ic]) do begin Tinh(c[ic]); dec(ic) end;
  i := ic;
  while (c[i] <> '(') do dec(i); { Tim ngoac ( }
  if c[i-1] = '@' then XuLiHam(i) else dec(ic); { Bo ( }
end;
function BieuThuc(var s: string): integer;
  var i: integer;
begin
  s := s + '#'; ic := 1; c[ic] := '#'; it := 0; v := 1;
  while (s[v] <> '#') do
    if LaBien(s[v]) then NapBien(s[v])
    else if LaChuSo(s[v]) then NapSo
    else if LaPhepToan(s[v]) then NapPhepToan(s[v])
    else if s[v] = ',' then NapPhay
    else if s[v] = '(' then NapNgoac
    else if s[v] = '@' then NapHam
    else if s[v] = ')' then XuLiNgoac
    else inc(v);
  while (LaPhepToan(c[ic])) do begin Tinh(c[ic]); dec(ic) end;
  for i := 2 to it do t[1] := t[1]+t[i];
  BieuThuc := t[1];
end;
BEGIN
  s := '@(-70,12+10-b,@(y+5,10)*c+12)';
  writeln(nl,s, ' = ',BieuThuc(s)); { 7 }
  readln;
END.
```

```

// Dev-C++: Func
#include <string.h>
#include <fstream>
#include <iostream>
using namespace std;
// DATA AND VARIABLE
const int mn = 500;
const char BL = ' ';//32;
char c[mn]; // stack lenh
int ic; // ngon stack c
int t[mn]; // stack tinh toan
int it; // ngon stac v
int v; // duyet s
char s[mn]; // xau input
// PROTOTYPES

int BieuThuc(char *);
int DocSo();
void Cach();
bool LaPhepToan(char );
bool LaChuSo(char );
bool LaBien(char );
int Val(char );
int Bac(char );
void NapHam();
void NapNgoac();
void NapSo();
void NapPhay();
void XuLiNgoac();
```

```

void XuLiHam(int);
void Tinh(char);
void XuLiToan(char);
int Ucln(int , int);
// I M P L E M E N T A T I O N
int main() {
    strcpy(s,"(10+@12,30+@6,8)+17*@()^2)*@1,3");
    cout << endl << "Ket qua: " << BieuThuc(s); // 16
    cout << endl; system("PAUSE");
    return EXIT_SUCCESS;
}
int Ucln(int a, int b) {
    int r;
    a = abs(a); b = abs(b);
    while (b) { r = a % b; a = b; b = r; }
    return a;
}
int Bac(char p) {
    if (p == '+' || p == '-') return 1;
    else if (p == '*' || p == '/' || p == '%') return 2;
    else if (p == '!') return 3;
    else return 0;
}
bool LaChuSo(char c) { return (c >= '0') && (c <= '9'); }
bool LaPhepToan(char c) {
    return (c=='!' || c=='+' || c=='-' || c=='*' || c=='/' || c=='%');
} // ! phep dau dau
bool LaBien(char c) { return (c >= 'a') && (c <= 'z'); }
int Val(char x) { return int(x-'0'); }
void Cach() { while (s[v] == BL) ++v; }
int DocSo() {
    int so = 0;
    Cach();
    if (!LaChuSo(s[v])) return 1;
    while (LaChuSo(s[v])) {
        so = so*10 + int(s[v]-'0'); ++v;
    }
    return so;
}
void NapNgoac() {
    c[++ic] = '('; ++v;
}
void NapHam() {
    c[++ic] = '@'; ++v; // bo qua dau @ trong s
    Cach(); // tim dau (
    c[++ic] = s[v]; // Nap (
    ++v; // bo qua dau ( tring s
    Cach(); if (s[v]==')') c[++ic] = '$';// ham rong
}
void NapSo() { t[++it] = DocSo(); }
void NapVal(char x) { t[++it] = Val(x); ++v; }
void Tinh(char p) {
    switch(p) {
    case '+': t[it-1] += t[it]; --it; break;
    case '-': t[it-1] -= t[it]; --it; break;
    case '*': t[it-1] *= t[it]; --it; break;
    case '/': t[it-1] /= t[it]; --it; break;
    case '%': t[it-1] %= t[it]; --it; break;
    case '!': t[it] = -t[it]; break;
    }
}
void XuLiHam(int i) { // c[i-1..i] = @(
    int kq = 0 ; // ket qua

```

```

        if (c[i+1]=='$') { // ham rong
            ic = i-2; t[++it] = 0;
            return;
        }
        int k = ic - i + 1; // so tham bien
        int jj = it;
        it = it - k + 1;
        for(int j = it; j <= jj; ++j)    kq = Ucln(kq, t[j]);
        t[it] = kq;
        ic = i-2;
    }
    void XuLiNgoac() { // Gap dau : s[v] = ')'
        int i;
        ++v; // bo qua ) trong s
        while (LaPhepToan(c[ic])) { // Thuc hien cac phep toan tren ngon
            Tinh(c[ic]); --ic;
        }
        i = ic;
        while (c[i] != '(') --i;// tim dau ( trong c
        // c[i] = '('
        if (c[i-1] == '@') XuLiHam(i); // gap ham @
        else --ic; // ko gap ham
    }
    void NapPhay() {
        ++v; // bo qua dau , trong s
        while (LaPhepToan(c[ic])) {
            Tinh(c[ic]); --ic;
        }
        c[++ic] = ',';
    }
    void XuLiToan(char p) {
        ++v; // bo qua ki tu trong s
        if (p=='-')
            if (!LaPhepToan(c[ic])) { c[++ic] = '!' ; return; }
        while (Bac(c[ic]) >= Bac(p)) {
            Tinh(c[ic]); --ic;
        }
        c[++ic] = p;
    }
    int BieuThuc(char *s ) {
        cout << endl << "input: " << s << endl;
        // init
        ic = it = 0; c[++ic] = '#';
        for (v = 0; s[v];) {
            if (s[v] == '@') NapHam();
            else if (s[v] == '(') NapNgoac();
            else if (LaChuSo(s[v])) NapSo();
            else if (LaBien(s[v])) NapVal(s[v]);
            else if (s[v] == ',') NapPhay();
            else if (s[v] == ')') XuLiNgoac();
            else if (LaPhepToan(s[v])) XuLiToan(s[v]);
            else ++v;
        }
        while (LaPhepToan(c[ic])) {
            Tinh(c[ic]); --ic;
        }
        for (;it > 1; --it) t[1] += t[it];
        return t[1];
    }
}

```

## 2.5 Files

<b>files.inp</b>	<b>files.out</b>	Mô hình số liệu và các CL trình bày ở các file 1, 2, 10 bài
------------------	------------------	---

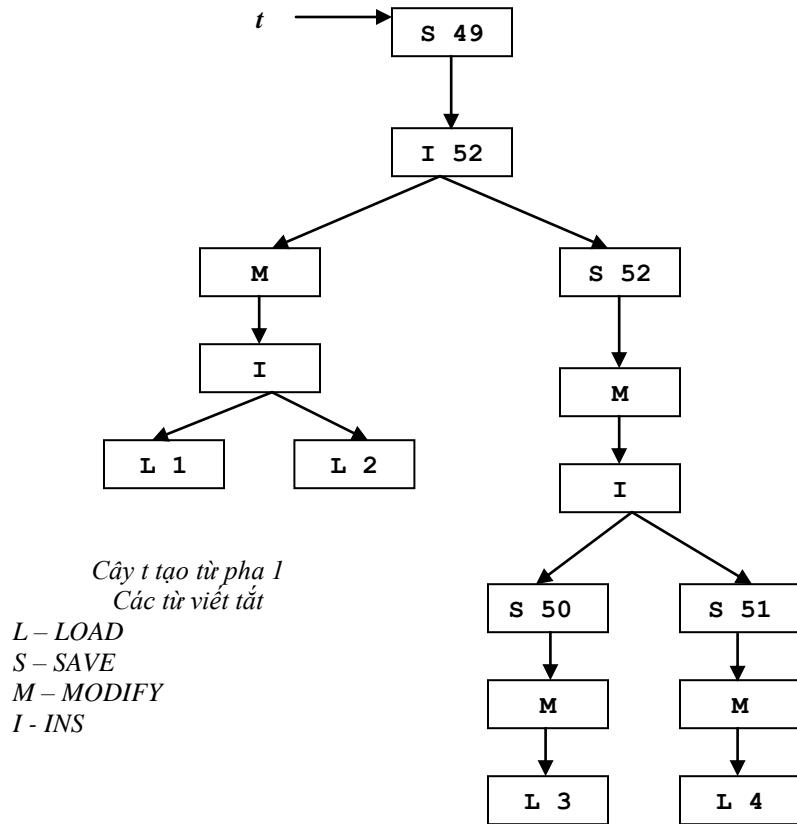
<b>LOAD 3</b>	<b>LOAD 4</b>	lưu trên đĩa cùng để tạo ra một file kết quả có tên 49. Qui trình xử lý được lập trình sẵn. Chương trình được viết bằng ngôn ngữ quản lý file gồm các lệnh
<b>MODI</b>	<b>MODI</b>	<b>LOAD i</b> : đọc file i vào miền nhớ RAM
<b>SAVE 50</b>	<b>SAVE 51</b>	<b>MODI</b> : sửa nội dung hiện có trên RAM
<b>LOAD 4</b>	<b>LOAD 3</b>	<b>INS j</b> : xen file j vào file trên RAM
<b>MODI</b>	<b>MODI</b>	<b>SAVE j</b> : ghi nội dung trên RAM vào file tên j.
<b>SAVE 51</b>	<b>INS 51</b>	<b>END</b> : kết thúc chương trình.
<b>LOAD 50</b>	<b>MODI</b>	Trừ lệnh <b>SAVE 49</b> cuối cùng, các lệnh <b>SAVE</b> đều được sử dụng để lưu tạm các kết quả trung gian với các tên file từ 50 trở đi. Với các file kích thước lớn, người ta muốn hạn chế số lần ghi đĩa bằng lệnh <b>SAVE</b> nhằm tiết kiệm thời gian xử lý. Hãy thay chương trình ghi trong <b>files.inp</b> bằng một chương trình tương đương ghi trong <b>files.out</b> với ít lệnh <b>SAVE</b> . Hai chương trình được xem là tương đương nếu file 49 chứa kết quả cuối cùng có nội dung giống nhau.
<b>INS 51</b>	<b>SAVE 50</b>	
<b>MODI</b>	<b>LOAD 1</b>	
<b>SAVE 52</b>	<b>INS 2</b>	
<b>LOAD 1</b>	<b>MODI</b>	
<b>INS 2</b>	<b>INS 50</b>	
<b>MODI</b>	<b>SAVE 49</b>	
<b>INS 52</b>	<b>END</b>	
<b>SAVE 49</b>		
<b>END</b>		

Trước hết ta xét một thí dụ minh họa. Giả sử lúc đầu mỗi file có chỉ số i chứa một số nguyên dương  $i+10$ ,  $i = 1, 2, \dots, 48$ . Thao tác MODI lật các chữ số trong RAM, tức là viết dãy số theo thứ tự ngược lại. Thao tác INS i đọc số trong file i rồi xen vào sau chữ số đầu tiên của file hiện lưu trên RAM.

CHƯƠNG TRÌNH TRONG FILES.INP	NỘI DUNG TRONG FILE	RAM	CHƯƠNG TRÌNH TRONG FILES.OUT	NỘI DUNG TRONG FILE	RAM
<b>LOAD 3</b>	<b>13</b>	<b>13</b>	<b>LOAD 4</b>	<b>14</b>	<b>14</b>
<b>MODI</b>		<b>31</b>	<b>MODI</b>		<b>41</b>
<b>SAVE 50</b>	<b>31</b>	<b>31</b>	<b>SAVE 51</b>	<b>41</b>	<b>41</b>
<b>LOAD 4</b>	<b>14</b>	<b>14</b>	<b>LOAD 3</b>	<b>13</b>	<b>13</b>
<b>MODI</b>		<b>41</b>	<b>MODI</b>		<b>31</b>
<b>SAVE 51</b>	<b>41</b>	<b>41</b>	<b>INS 51</b>	<b>41</b>	<b>3411</b>
<b>LOAD 50</b>	<b>31</b>	<b>31</b>	<b>MODI</b>		<b>1143</b>
<b>INS 51</b>	<b>41</b>	<b>3411</b>	<b>SAVE 50</b>	<b>1143</b>	<b>1143</b>
<b>MODI</b>		<b>1143</b>	<b>LOAD 1</b>	<b>11</b>	<b>11</b>
<b>SAVE 52</b>	<b>1143</b>	<b>1143</b>	<b>INS 2</b>	<b>12</b>	<b>1121</b>
<b>LOAD 1</b>	<b>11</b>	<b>11</b>	<b>MODI</b>		<b>1211</b>
<b>INS 2</b>	<b>12</b>	<b>1121</b>	<b>INS 50</b>	<b>1143</b>	<b>11143211</b>
<b>MODI</b>		<b>1211</b>	<b>SAVE 49</b>	<b>11143211</b>	<b>11143211</b>
<b>INS 52</b>	<b>1143</b>	<b>11143211</b>	<b>END</b>		
<b>SAVE 49</b>	<b>11143211</b>	<b>11143211</b>			
<b>END</b>					

Khí đó trình tự thực hiện hai chương trình ghi trong input file FILES.INP và output file FILES.OUT được giải trình chi tiết trong bảng. Nội dung ghi trong 4 file mang mã số 1, 2, 3 và 4 đầu tiên lần lượt là 11, 12, 13 và 14. Kết quả cuối cùng của cả hai chương trình đều giống nhau và bằng **11143211**. Chương trình ghi trên file inp chứa 4 lệnh SAVE trong khi chương trình ghi trên file out chứa 3 lệnh SAVE. Bạn cũng cần lưu ý rằng phép xen file INS nói chung không thỏa tính giao hoán và kết hợp. Để minh họa ta tạm qui ước nội dung ghi trong file được đặt giữa cặp ngoặc **[]**. Khi đó, [12] INS [13] = [1132], trong khi [13] INS [12] = [1123], ngoài ra ([12] INS [13]) INS [14] = [1132] INS [14] = [114132], trong khi [12] INS ([13] INS [14]) = [12] INS [1143] = [111432].

## Thuật toán



Thuật toán được triển khai theo 2 pha: *Pha Đọc* và *Pha Ghi*. Pha thứ nhất, Pha Đọc sẽ đọc từng dòng lệnh của chương trình nguồn P từ input file và tạo thành một cây t. Pha thứ hai, Pha Ghi chỉ đơn thuần ghi lại cây t vào output file theo nguyên tắc *ưu tiên lệnh SAVE cho cây con phải*. Ta trình bày chi tiết các kỹ thuật tại hai pha.

Mỗi nút của cây t có 4 thành phần ( $C, f, L, R$ ) trong đó  $C$  là *chữ cái đầu* của lệnh LOAD, SAVE, MODI, END,  $f$  là *số hiệu file* (tên) trong dòng lệnh,  $L$  và  $R$  lần lượt là *con trỏ trái* và *phải* tới nút tiếp theo trên cây. Ta tạm kí hiệu 0 là con trỏ NULL trong C++ và NIL trong Pascal.

**1. Pha Đọc** sẽ đọc lần lượt từng dòng lệnh từ chương trình P vào biến string s và nhận biết lệnh qua chữ cái đầu tiên LOAD, SAVE, MODI, INS, END rồi xử lí theo từng trường hợp như sau:

**LOAD f :** Nếu file f chưa xuất hiện thì tạo một nút mới  $v = (C, f, 0, 0)$  và đánh dấu f là file đã *xuất hiện*. Việc đánh dấu được thực hiện thông qua phép gán  $p[f] = v$  trong đó p là mảng các con trỏ tới các nút, f là số hiệu (tên) file, v là giá trị của con trỏ được cấp phát cho nút được khởi tạo cho file f,  $p[f] = 0$  (NULL/nil) cho biết file f chưa xuất hiện trong chương trình P. Nếu f đã xuất hiện thì ta gán v là con trỏ tới nút đã khởi tạo cho file f,  $v = p[f]$ ;

**SAVE f :** Không tạo nút mới, chỉ ghi nhận f vào biến lastsave để biết phép SAVE cuối cùng của chương trình P sẽ ghi kết quả vào file nào. Đồng thời ta cũng cần đánh dấu  $p[f] = v$  để biết rằng file hiện có trên RAM là v đã được ghi vào file f;

**MODI:** Tạo nút mới  $v = \text{NewElem}'(M', -1, v, 0)$  đặt trên nút v hiện hành. Giá trị của trường fnum được đặt là  $-1$  cho biết lệnh này không quan tâm đến tên file vì nó chỉ MODIFY nội dung của file hiện có trên RAM;

**INS f :** Nếu file f chưa xuất hiện thì tạo nút mới  $v = \text{NewElem}'(I', f, v, 0)$ , ngược lại, nếu file f đã xuất hiện thì tạo nút mới  $v = \text{NewElem}'(I', -1, v, p[f])$ .

**2. Pha Ghi** Giả sử t là cây tạo được sau pha 1. Ta viết thủ tục *ToFile(t)* duyệt cây t theo nguyên tắc *ưu tiên cây con phải* để ghi vào output file chương trình tối ưu số lệnh SAVE như sau. Cụ thể là ta duyệt cây con

phải trước, sau đến cây con trái và cuối cùng là nút giữa. Thủ tục này có tên RNL. Ta xét các tình huống sau đây.

2.1 Cây t chỉ có cây con trái L, t = (L,0): Ta chỉ việc gọi thủ tục ToFile(L).

2.2 Cây t có cả hai cây con trái L và phải R, t = (L,R) và hai cây con này được gắn với nhau bằng lệnh INS: Trước hết phải gọi thủ tục ToFile(R) và ghi kết quả trung gian vào một file tạm m, sau đó gọi thủ tục ToFile(L) rồi thêm vào cuối dòng lệnh INS m.

**Độ phức tạp** Cỡ n – số dòng lệnh trong input file.

**Bình luận** Thuật toán trên cũng bỏ qua trường hợp dãy lệnh SAVE f giống nhau liên tiếp cũng như trường hợp hai lệnh liên tiếp là LOAD f và SAVE f với cùng một tên file. Bạn có thể cải tiến thêm bằng cách đọc input file một lần vào miền nhớ và loại bỏ các lệnh này trước khi tổ chức cây t.

```
(* Files.pas *)
uses crt;
const fn = 'files.inp'; gn = 'files.out';
      mn = 400; nl = #13#10; bl = #32;
      chuso = ['0'..'9'];
type pelem = ^elem;
   elem = record
      com: char; { lenh }
      fnum: integer; { so hieu file }
      L,R: pelem; { tro trai, phai }
   end;
var f,g: text; { f: input file; g: output file }
p: array[1..mn] of pelem; { danh dau cac file }
s: string; { dong lenh }
t: pelem; { cay nhi phan chuong trinh }
lastSave: integer;
tmp: integer; { file trung gian }
function DocSo: integer; { doc so tu dong lenh s }
  var i, so: integer;
begin
  so := 0;
  i := 1;
  while not (s[i] in chuso) do inc(i);
  while (s[i] in chuso) do
  begin
    so := so*10 + ord(s[i]) - ord('0');
    inc(i);
  end;
  DocSo := so;
end;
function NewElem(c: char; fno: integer; Lp,Rp: pelem): pelem;
  var e: pelem;
begin
  new(e);
  e^.com := c; e^.fnum := fno;
  e^.L := Lp; e^.R := Rp;
  NewElem := e;
end;
function Load(fno: integer): pelem;
begin
  if p[fno] = nil then
    p[fno] := NewElem('L',fno,nil,nil);
  Load := p[fno];
end;
procedure Save(v: pelem; fno: integer);
begin lastSave := fno; p[fno] := v; end;
function Ins(v: pelem; fno: integer): pelem;
begin
  if p[fno] = nil then
    Ins := NewElem('I',fno,v,nil)
```

```

    else Ins := NewElem('I', -1, v, p[fno]);
end;
function Doc: pelem;
var i: integer;
    v: pelem; { RAM }
begin
    for i := 1 to mn do p[i] := nil;
    assign(f,fn); reset(f);
    while not seekeof(f) do
begin
    readln(f,s); s := s + '#';
    case s[1] of
    'L': v := Load(DocSo);
    'S': Save(v,DocSo);
    'M': v := NewElem('M', -1, v, nil);
    'I': v := Ins(v,DocSo);
    'E': begin close(f); Doc := v; exit end;
    end;
end;
end;
procedure ToFile(t: pelem);
var sav: integer;
begin
    sav := 0;
    if (t = nil) then exit;
    if (t^.R <> nil) then
begin
    inc(tmp); sav := tmp;
    ToFile(t^.R);
    writeln(g,'SAVE',bl,sav);
end;
    ToFile(t^.L);
    case(t^.com) of
    'I': if (t^.R = nil) then writeln(g,'INS',bl,t^.fnum)
        else if (sav > 0) then writeln(g,'INS',bl,sav);
    'L': writeln(g,'LOAD',bl,t^.fnum);
    'M': writeln(g,'MODI');
    end;
end;
procedure Ghi(t: pelem);
begin
    assign(g,gn); rewrite(g);
    tmp := 49;
    ToFile(t);
    writeln(g,'SAVE',bl,lastsave,nl,'END');
    close(g);
end;
BEGIN
    t := Doc; Ghi(t);
    writeln(nl,' Fini');readln;
END.
```

```

// Dev-C++: Files
#include <string.h>
#include <fstream>
#include <iostream>
using namespace std;
// DATA AND VARIABLE
const char * fn = "files.inp";
const char * gn = "files.out";
const char * LOAD = "LOAD";
const char * MODI = "MODI";
const char * INS = "INS";
```

```

const char * SAVE = "SAVE";
const char * END = "END";
const char star = '*'; // Node
const int mn = 400;
struct elem {
    char com; int fnum; // com: lenh, fnum: file number
    elem * L; elem * R; // tro trai, tro phai
};
elem* p[mn];
int tmp;
int lastsave;
// P R O T O T Y P E S
elem * Doc();
elem * Load(int);
elem * NewNode(char, int, elem*, elem*);
elem * Ins(elem *, int);
void Ghi(elem *t);
void ToFile(elem* );
ofstream g(gn);
// I M P L E M E N T A T I O N
int main() {
    elem * t = Doc();
    Ghi(t);
    cout << endl; system("PAUSE");
    return EXIT_SUCCESS;
}
void Ghi(elem *t) {
    tmp = 49;
    ToFile(t);
    g << "SAVE " << lastsave << endl;
    g << "END" << endl;
    g.close();
}
void ToFile(elem *t) {
    int sav = 0;
    if (t == NULL) return;
    if (t->R != NULL) {
        ++tmp; sav = tmp;
        ToFile(t->R);
        g << "SAVE " << sav << endl;
    }
    ToFile(t->L);
    switch(t->com) {
        case 'I': if (t->R == NULL) g << "INS " << t->fnum << endl;
                    else if (sav > 0) g << "INS " << sav << endl;
                    break;
        case 'L': g << "LOAD " << t->fnum << endl;
                    break;
        case 'M': g << "MODI" << endl;
                    break;
    }
}
elem * Ins(elem *v, int fno) { // Tao node INS
    elem * e = NewNode('I', -1, v, NULL);
    if (p[fno] != NULL) e->R = p[fno];
    else e->fnum = fno;
    return e;
}
elem * Load(int fno) { // Tao node LOAD
    if (p[fno] == NULL)
        p[fno] = NewNode('L', fno, NULL, NULL);
    return p[fno];
}

```

```

elem * NewNode(char cm, int i, elem * lp, elem * rp) {
    elem * pt = new elem; // Tao node moi
    pt->com = cm; pt->fnum = i;
    pt->L = lp; pt->R = rp;
    return pt;
}
elem * Doc() { // Doc vhuong trinh, tao cay
    char s[10];
    ifstream f(fn);
    elem * v = NULL; // RAM
    int i, fno;
    for (i = 0; i < mn; ++i) p[i] = NULL;
    while (1) {
        f >> s; fno = 0;
        switch(s[0]) {
            case 'E': f.close(); return v; // END
            case 'L': f >> fno; v = Load(fno); break; // LOAD
            case 'S': f >> fno; lastsave = fno;
                        p[fno] = v; break; // SAVE
            case 'I': f >> fno; v = Ins(v, fno); break; // INS
            case 'M': v = NewNode('M', -1, v, NULL); break; // MODIFY
        }
    }
}

```

## 2.6 Gen

(Bài tương tự)

Trong phòng thí nghiệm sinh học phân tử người ta bảo quản các đoạn mã di truyền, tạm gọi là các đoạn gen trong các tủ chứa đặc biệt được mã số 1, 2,...,98. Các gen được chỉnh sửa và cây ghép với nhau trên bàn thí nghiệm T để tạo ra một gen cuối cùng đặt trong tủ chứa số 99 theo một chương trình máy tính tự động bao gồm các thao tác sau:

**TAKE i**: lấy một gen từ tủ chứa i đặt lên bàn thí nghiệm T,

**MODI**: làm sạch, sửa chữa các thông tin di truyền trong gen hiện có trên bàn thí nghiệm T,

**INS j**: cây ghép gen hiện có trên T với gen trong tủ chứa j, kết quả thu được một gen trên T,

**STORE k**: cất gen trên T vào tủ chứa k,

**END**: kết thúc chương trình.

Việc lấy và cất giữ gen đòi hỏi các thủ tục rát phức tạp nên người ta cần hạn chế đến mức tối đa các thao tác này. Ngoài ra, lưu ý rằng việc cây ghép gen i vào gen j cho kết quả khác với việc cây ghép gen j vào gen i. Hãy thay chương trình cho trước bằng một chương trình tương đương với ít lệnh STORE nhất theo nghĩa cho cùng kết quả thu được trong tủ chứa 99 như chương trình ban đầu. Trong quá trình thao tác được phép sử dụng các tủ chứa tạm từ 100 trở đi .

## 2.7 Tối ưu hóa chương trình

(Bài tương tự)

prog.inp	prog.out
LOAD x	LOAD x
ADD y	SUB y
SAVE 100	SAVE 100
LOAD x	LOAD x
SUB y	ADD y
SAVE 101	MULT 100
LOAD 100	SAVE z
MULT 101	END
SAVE z	
END	

Trong các bộ xử lý một địa chỉ các thao tác xử lý được thực hiện trên thanh ghi A, các hằng và biến được ghi trong miền nhớ RAM với các địa chỉ 0, 1, 2,... Chương trình được viết dưới dạng một dãy tuần tự các lệnh mã máy bao gồm các lệnh sau

**LOAD i**: đọc dữ liệu từ địa chỉ i vào thanh ghi A,

**ADD j**: cộng số hiện có trên thanh ghi A với số có trong địa chỉ j, kết quả để trên A,  $A := A + j$ , trong đó  $j$  là hiệu nội dung có trong địa chỉ  $j$ ,

**SUB j**:  $A := A - j$ ,

**MULT j**:  $A := A * j$ ,

**DIV j**:  $A := A / j$ ,

**SAVE j**: ghi nội dung trên thanh ghi A vào địa chỉ  $j$ ,

**END**: kết thúc chương trình.

Giả thiết rằng các kết quả tính toán trung gian được lưu tạm vào vùng nhớ tự do có địa chỉ qui ước từ 100 trở đi. Các phép toán không thỏa các tính chất giao hoán và kết hợp. Hãy thay chương trình ghi trên text file tên prog.inp bằng một chương trình tương đương với ít lệnh SAVE nhất và ghi kết quả vào text file tên prog.out.

Thí dụ, file prog.inp là chương trình tính  $z := (x+y)*(x-y)$  với 3 lệnh SAVE, file prog.out là chương trình tương đương với 2 lệnh SAVE.

## 2.8 Mức của biểu thức

Trong các biểu thức tính toán người ta thường dùng các cặp ngoặc (...) để nhóm thành các biểu thức con. Mức của biểu thức được hiểu là số lượng tối đa các cặp ngoặc lồng nhau trong biểu thức, thí dụ biểu thức  $(a+(b-c)*d)-(a-b)$  có mức 2. Cho trước k cặp ngoặc và mức h. Hãy cho biết có thể xây dựng được bao nhiêu biểu thức mức h và sử dụng đúng k cặp ngoặc. Thí dụ, ta có 3 biểu thức mức h = 2 sử dụng đúng k = 3 cặp ngoặc như sau:

```
( () () )
( () ) ()
() ( () )
```

Dạng hàm: Level(k,h)

Test 1. Level(3,2) = 3;

Test 2. Level(19,18) = 35.

### Thuật toán

Gọi  $s(k,h)$  là hàm 2 biến cho ra số lượng các biểu thức khác nhau có mức h và chứa đúng k cặp ngoặc. Xét cặp ngoặc thứ k. Ta thấy,

- Nếu gọi A là biểu thức mức h-1 chứa đúng k-1 cặp ngoặc thì (A) sẽ là biểu thức độ sâu h và chứa đúng k cặp ngoặc.
- Nếu gọi B là biểu thức mức h chứa đúng k-1 cặp ngoặc thì ( ) B và B ( ) sẽ là hai biểu thức mức h và chứa đúng k cặp ngoặc. Tuy nhiên trong trường hợp này ta phải loại đi tình huống ( ) B = B ( ). Tình huống này chỉ xảy ra duy nhất khi B có dạng dãy các cặp ngoặc mức 1: B = ( )...(.). Khi đó ( ) B = ( ) ( ) ... ( ) = B ( ).

Tóm lại, ta có

$$s(k,h) = s(k-1,h-1) + 2s(k-1,h) \text{ với } h > 1, \text{ và}$$

$s(k,1) = 1$ ,  $k = 1, 2, \dots$ , với  $k \geq 1$  cặp ngoặc chỉ có thể viết được 1 biểu thức mức 1 gồm dãy liên tiếp k cặp ngoặc ( ) ( ) ... ( ).

Ngoài ra ta có

$$s(0,h) = 0, h > 0, \text{ với } 0 \text{ cặp ngoặc không thể xây dựng được biểu thức mức } h > 0;$$

$$s(0,0) = 1, \text{ với } 0 \text{ cặp ngoặc có duy nhất 1 biểu thức mức 0 (qui ước).}$$

Cài đặt: Ta có thể cài đặt hàm  $s(k,h)$  với k lần lặp và 2 mảng 1 chiều a và b, trong đó  $a[j]$  là giá trị của hàm  $s(k-1,j)$ ,  $b[j]$  là giá trị của hàm  $s(k,j)$ ,  $j = 1..h$ .

Trước hết ta khởi trị ứng với  $k = 1$ :  $a[1] = b[1] = 1$ ;  $a[i] = 0$ ,  $i = 2..h$  với ý nghĩa sau: có 1 cặp ngoặc thì viết được 1 biểu thức mức 1, không có các biểu thức mức trên 1.

Giả sử tại bước lặp thứ  $k-1$  ta đã tính được các giá trị của hàm  $s(k-1,j)$  và lưu trong mảng a như sau:  $a[j] = s(k-1,j)$ ,  $j = 1..h$ . Khi đó các giá trị của hàm  $s(k,j)$  sẽ được tính và lưu trong mảng b như sau:

$$b[1] = s(k,1) = 1$$

$$b[j] = s(k,j) = s(k-1,j-1) + 2s(k-1,j) = a[j-1] + 2a[j], j = 2..h$$

**Độ phức tạp:**  $k.h$ .

```
(* Level.pas *)
uses crt;
const bl = #32; nl = #13#10; mn = 1000;
function Level(k,h: integer): longint;
var a,b: array[0..mn] of longint;
    i,j: integer;
begin
  fillchar(a, sizeof(a), 0); a[1] := 1; b[1] := 1;
  for i := 2 to k do { i cap ngoac }
  begin
    for j := 2 to h do { i cap ngoac, muc j }
      b[j] := a[j-1] + 2*a[j];
    a := b;
  end;
  Level := a[h];
end;
```

```

BEGIN
    writeln(nl, level(3,2), nl, level(19,18));
    readln;
END.

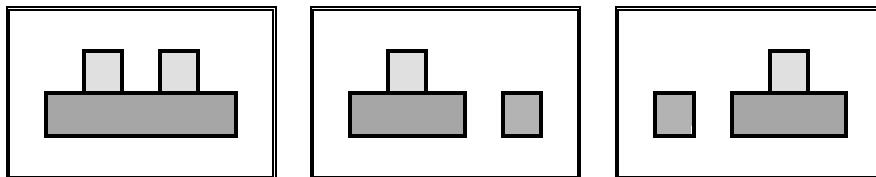
// Dev-C++: Level
#include <string.h>
#include <iostream>
#include <stdio.h>
using namespace std;
// P R O T O T Y P E S
int Level(int, int);
// I M P L E M E N T A T I O N
int main() {
    cout << endl << endl << Level(19,18); // 35
    cout << endl << endl << Level(3,2); // 3
    cout << endl << endl << " Fini" << endl;
    cin.get();
    return 0;
}
int Level(int k, int h){
    int h1 = h+1;
    int *a = new int[h1];
    int *b = new int[h1];
    memset(a,0,d1*sizeof(int)); a[1] = b[1] = 1;
    for (int i = 2; i <= k; ++i) {
        a[i] = 1;
        for (int j = 2; j <= h; ++j) b[j] = a[j-1] + 2*a[j];
        memmove(a,b,h1*sizeof(int));
    }
    delete a; delete b;
    return a[h];
}

```

## 2.9 Tháp

(Bài tương tự)

Các em nhỏ dùng các khối gỗ hình chữ nhật to nhỏ khác nhau và có bề dày 1 đơn vị đặt chồng lên nhau để tạo thành một công trình kiến trúc gồm nhiều tòa tháp. Khối đặt trên phải nhỏ hơn khối dưới, số lượng khối các loại là không hạn chế. Độ cao của công trình được tính theo chiều cao của tháp cao nhất trong công trình. Với  $k$  khối gỗ có thể xây được bao nhiêu kiểu công trình độ cao  $h$ .



3 công trình độ cao 2 được xây bằng 3 khối gỗ

## 2.10 Mì trang

Trong một file văn bản có  $n$  từ. Với mỗi từ thứ  $i$  ta biết chiều dài  $v_i$ , tính bằng số kí tự có trong từ đó. Người ta cần căn lề trái cho file với độ rộng  $m$  kí tự trên mỗi dòng. Mỗi từ cần được xếp trọn trên một dòng, mỗi dòng có thể chứa nhiều từ và trật tự các từ cần được tôn trọng. Hãy tìm một phương án căn lề sao cho phần hụt lớn nhất ở bên phải các dòng là nhỏ nhất. Giả thiết rằng mỗi từ đều có 1 dấu cách ở cuối, dĩ nhiên dấu cách này được tính vào chiều dài từ.

Dữ liệu vào: file văn bản PAGES.INP. Dòng đầu tiên chứa hai số  $n$  và  $m$ .

Tiếp đến là  $n$  chiều dài từ với các giá trị nằm trong khoảng từ 2 đến  $m$ .

Dữ liệu ra: file văn bản PAGES.OUT. Dòng đầu tiên chứa hai số  $h$  và  $k$ , trong đó  $h$  là phần thừa lớn nhất (tính theo số kí tự) của phương án tìm được,  $k$  là số dòng của văn bản đã được căn lề. Tiếp đến là  $k$  số cho biết trên dòng đó phải xếp bao nhiêu từ.

Các số trên cùng dòng cách nhau qua dấu cách.

PAGES . INP	PAGES . OUT
6 10	3 3
2 2 3 3 6 9	3 2 1

Ý nghĩa: Cân xếp 6 từ chiều dài lần lượt là 2, 2, 3, 3, 6 và 9 trên các dòng dài tối đa 10 kí tự. Nếu xếp thành 3 dòng là (2,2,3,3) (6) (9) thì phần hụt tối đa là 4 (trên dòng 2). Nếu xếp thành 3 dòng là (2,2,3) (3,6) (9) thì phần hụt tối đa là 3 (trên dòng 1). Như vậy ta chọn cách xếp thứ hai với 3 dòng: Dòng 1: 3 từ; dòng 2: 2 từ; dòng 3: 1 từ.

## Thuật toán

Gọi  $d(i)$  là đáp số của bài toán với  $i$  từ đầu tiên. Ta xét cách xếp từ  $w[i]$ . Đầu tiên ta xếp  $w[i]$  riêng 1 dòng, độ hụt khi đó sẽ là  $h = m - v[i]$ . Nếu chở hụt còn đủ ta lại kéo từ  $i-1$  từ dòng trên xuống, độ hụt khi đó sẽ là  $h = h - v[i-1], \dots$ . Tiếp tục làm như vậy đến khi độ hụt  $h$  không đủ chứa thêm từ kéo từ dòng trên xuống. Mỗi lần kéo thêm một từ  $j$  từ dòng trên vào dòng mới này ta có một phương án. Độ hụt của phương án này sẽ là  $\max(h - v[j], d(j-1))$ . Ta sẽ chọn phương án nào đạt trị min trong số đó.

Để cài đặt ta sử dụng mảng một chiều  $d$  chứa các giá trị của hàm  $d(i)$ . Ta khởi trị cho  $v[0] = m+1$  làm lính canh,  $d[1] = m - v[1]$ , vì khi chỉ có 1 từ thì ta xếp trên 1 dòng duy nhất và độ hụt là  $m - v[1]$ .

Để xác định sự phân bố số lượng từ trên mỗi dòng ta dùng mảng trả ngược  $t[1..n]$  trong đó  $t[i] = j$  cho ta biết các từ  $j, j+1, \dots, i$  cùng nằm trên một dòng theo phương án tối ưu. Sau đó ta gọi đệ qui muộn mảng  $t$  để tính ra số lượng các từ trên mỗi dòng, ghi vào mảng  $sl$  theo trật tự ngược.

**Độ phức tạp:** Cỡ  $n^2$  vì với mỗi từ  $i$  ta phải duyệt ngược  $i$  lần. Tổng cộng có  $n$  từ nên số lần duyệt sẽ là  $n.n$ .

```
(* Pages.pas *)
uses crt;
const mn = 200; bl = #32; nl = #13#10;
      fn = 'pages.inp'; gn = 'pages.out';
type mil = array[0..mn] of integer;
var n,m,k,h: integer;
    f,g: text;
    v, d, t, sl: mil;
    (* -----
       n - number of words; m - width of page;
       k - number of lines; h - maximum of
            white character of all lines;
       v[i] - length of i-th word;
       d[i] - solution result of input data v[1..i];
       t[i] = j: all words j, j+1,...,i are in one line;
       sl[i] - number of words in i-th line
       -----*)
procedure PP(var x: mil; d,c: integer);
var i: integer;
begin for i := d to c do write(bl,x[i]); end;
function Min(a,b: integer): integer;
begin if a < b then Min := a else Min := b end;
function Max(a,b: integer): integer;
begin if a > b then Max := a else Max := b end;
procedure Doc;
var i: integer;
begin assign(f,fn); reset(f); readln(f,n,m);
  writeln(n,bl,m);
  for i := 1 to n do read(f,v[i]);
end;
```

```

    close(f);
end;
procedure Tinhd(i: integer);
var j,h, hmax: integer;
begin
  h := m; j := i; d[i] := m+1;
  while h >= v[j] do
  begin
    h := h - v[j]; hmax := Max(d[j-1],h);
    if d[i] > hmax then
    begin
      d[i] := hmax; t[i] := j;
    end;
    dec(j);
  end;
end;
function XuLi: integer;
var i: integer;
begin
  t[0] := 0; v[0] := m+1; d[0] := 0;
  for i := 1 to n do Tinhd(i);
  XuLi := d[n];
end;
procedure Xep(i: integer);
begin
  if (i = 0) then exit;
  inc(k); sl[k] := i-t[i]+1;
  Xep(t[i]-1);
end;
procedure Ghi;
var i: integer;
begin
  assign(g,gn); rewrite(g);
  writeln(g,h,bl,k);
  for i := k downto 1 do write(g,sl[i],bl);
  close(g);
end;
procedure Run;
var i: integer;
begin
  Doc; PP(v,1,n); h := XuLi;
  k := 0; Xep(n);
  writeln(nl, h, bl, k, nl);
  for i := k downto 1 do write(bl, sl[i]);
  Ghi;
end;
BEGIN
  Run;
  write(nl, ' Fini ');readln;
END.
```

```

// DevC++: Pages.cpp
#include <fstream>
#include <iostream>
#include <math.h>
using namespace std;
// Data and variables
const char * fn = "pages.inp";
const char * gn = "pages.out";
const int mn = 200;
int n; // so luong tu
int m; // len dong
int k; // so dong can viet
```

```

int h; // do hut toi uu
int v[mn]; // len cac tu
int d[mn]; //
int t[mn]; // con tro nguoc
int sl[mn];
// Interface
void Doc();
void PP(int [], int , int);
int XuLi();
void Tinhd(int);
void Xep(int);
void Ghi();
// Implementation
main () {
    Doc(); h = XuLi();
    k = 0; Xep(n); Ghi();
    cout << endl << h << " " << k << endl;
    for (int i = k; i > 0; --i) cout << " " << sl[i];
    cout << endl << " Fini"; cin.get();
    return 0;
}
void Ghi() {
    ofstream g(gn);
    g << h << " " << k << endl;
    for (int i = k; i > 0; --i) g << sl[i] << " ";
    g.close();
}
void Tinhd(int i) {
    int h = m-v[i]; // cho hut
    d[i] = max(h,d[i-1]); t[i] = i;
    int hmax;
    for (int j = i-1; h >= v[j]; --j) {
        h = h - v[j]; hmax = max(h,d[j-1]);
        if (d[i] > hmax) { d[i] = hmax; t[i] = j; }
    }
}
void Xep(int i) { // xep cac tu tren moi dong
    if (t[i] == 0) return;
    sl[++k] = i-t[i]+1;
    Xep(t[i]-1);
}
int XuLi() {
    v[0] = m+1; d[0] = 0; d[1] = m-v[1]; t[1] = 1;
    for (int i = 2; i <= n; ++i) Tinhd(i);
    return d[n];
}
void Doc() {
    ifstream f(fn); f >> n >> m;
    cout << n << " " << m;
    for (int i = 1; i <= n; ++i) f >> v[i];
    f.close();
    PP(v,1,n);
}
void PP(int a[], int d, int c) {
    cout << endl;
    for (int i = d; i <= c; ++i) cout << a[i] << " ";
}

```

## 2.11 Xếp thẻ

## Chương 3 Cặp ghép

Lớp các bài toán xác định một tương ứng giữa hai tập phần tử A và B cho trước, thí dụ như tập A gồm các em thiếu nhi và tập B gồm các món quà như trong bài toán Chị Hằng dưới đây được gọi là các bài toán cặp ghép và thường được kí hiệu là  $f: A \rightarrow B$  với ý nghĩa là cần xác định một ánh xạ, tức là một phép đặt tương ứng mỗi phần tử  $i$  của tập A với duy nhất một phần tử  $j$  của tập B,  $f(i) = j$ . Một trong các thuật toán giải các bài toán này có tên là thuật toán Ghép cặp. Thuật toán đòi hỏi thời gian tính toán là  $n.m$  phép so sánh trong đó  $n$  là số phần tử (lực lượng) của tập A,  $m$  là số phần tử của tập B,  $n = \|A\|$ ,  $m = \|B\|$ .

Chương này trình bày thuật toán ghép cặp và các biến thể của nó.

### 3.1 Chị Hằng

Nhân dịp Tết Trung Thu Chị Hằng rời Cung Trăng mang m món quà khác nhau mã số 1..m đến vui Trung Thu với n em nhỏ mã số 1..n tại một làng quê. Trước khi Chị Hằng phát quà, mỗi em nhỏ đã viết ra giấy những món quà mà em đó mơ ước. Yêu cầu: giúp Chị Hằng chia cho mỗi em đúng 1 món quà mà em đó yêu thích.

Dữ liệu vào: file văn bản autum.inp

Dòng đầu tiên: hai số  $n$   $m$

Dòng thứ i trong số  $n$  dòng tiếp theo:  $k$   $b_1$   $b_2$  ...  $b_k$  -  $k$  là số lượng quà em i yêu thích;  $b_1$   $b_2$  ...  $b_k$  là mã số các món quà em i yêu thích.

Dữ liệu ra: file văn bản autum.out

Dòng đầu tiên:  $v$  – số em nhỏ đã được nhận quà.

v dòng tiếp theo: mỗi dòng 2 số  $i$   $b$  cho biết em i được nhận món quà  $b$ .

Thí dụ,

autum.inp	autum.out
5 5	5
2 1 5	1 1
2 2 4	2 4
2 1 2	3 2
3 1 4 5	4 5
2 1 3	5 3

#### Ý nghĩa

Có 5 em và 5 món quà. Em 1 thích 2 món quà: 1 và 5; em 2 thích 2 món quà: 2 và 4; em 3 thích 2 món quà: 1 và 2; em 4 thích 3 món quà: 1, 4 và 5; em 5 thích 2 món quà: 1 và 3.

Một phương án xếp em nhỏ → quà như sau: 1→1; 2→4; 3→2; 4→5; 5→3.

### Thuật toán

Giả sử các phần tử của tập nguồn A (các em nhỏ) được mã số từ 1 đến  $n$  và các phần tử của tập đích B (các gói quà) được mã số từ 1 đến  $m$ . Sau khi đọc dữ liệu và thiết lập được ma trận 0/1 hai chiều c với các phần tử  $c[i,j] = 1$  cho biết em  $i$  thích món quà  $j$  và  $c[i,j] = 0$  cho biết em  $i$  không thích quà  $j$ . Nhiệm vụ đặt ra là thiết lập một ánh xạ  $1 \rightarrow 1$   $f$  từ tập nguồn vào tập đích,  $f: A \rightarrow B$ . Ta sử dụng phương pháp *chỉnh dần các cặp đã ghép* để tăng thêm số cặp ghép như sau.

Ta cũng sử dụng hai mảng một chiều A và B để ghi nhận tiến trình chia và nhận quà với ý nghĩa như sau:  $A[i] = j$  cho biết em  $i$  đã được nhận quà  $j$ ;  $B[j] = i$  cho biết quà  $j$  đã được chia cho em  $i$ ;  $A[i] = 0$  cho biết em  $i$  chưa được chia quà và  $B[j] = 0$  cho biết quà  $j$  trong túi quà B còn rồi (chưa chia cho em nào).

Giả sử ta đã chọn được quà cho các em 1, 2, ...,  $i-1$ . Ta cần xác định  $f(i) = j$ , tức chọn món quà  $j$  cho em  $i$ .

Nếu ta tìm ngay được món quà  $j \in B$  thỏa đồng thời các điều kiện sau:

- $B[j] = 0$ :  $j$  là món quà còn trong túi quà B, tức là  $quà j chưa được chia$ ,
- $c[i,j] = 1$ , tức là em  $i$  thích quà  $j$

thì ta đặt  $f(i) = j$  và việc chia quà cho em  $i$  đã xong.

Trường hợp ngược lại, nếu với mọi quà  $j$  thỏa  $c[i,j] = 1$  (em  $i$  thích quà  $j$ ) đều đã được chia cho một em  $t$  nào đó ( $B[j] = t \neq 0$ ) thì ta phải tiến hành *thủ tục thương lượng* với toàn bộ các em đang giữ quà mà bạn  $i$  thích như sau:

- Tạm đề nghị các em đang giữ quà mà bạn  $i$  thích, đặt quà đó vào một túi riêng bên ngoài túi có đè chửi với ý nghĩa "sẽ trao 1 món quà trong túi này cho bạn  $i$ ";
- Đưa những em vừa trả lại quà vào một danh sách st gồm các em cần được ưu tiên tìm quà ngay.

Như vậy, em  $i$  sẽ có quà nếu như ta tiếp tục tìm được quà cho một trong số các em trong danh sách st nói trên. Với mỗi em trong danh sách st ta lại thực hiện các thủ tục như đã làm với em  $i$  nói trên.

Ta cần đánh dấu các em trong danh sách để đảm bảo rằng không em nào xuất hiện quá hai lần và như vậy sẽ tránh được vòng lặp vô hạn.

Sau một số bước lặp ta sẽ thu được dãy

$$t_1 \leftarrow t_2 \leftarrow \dots t_{k-1} \leftarrow t_k$$

với ý nghĩa là

em  $t_1$  sẽ nhận quà từ em  $t_2$ ,

em  $t_2$  sẽ nhận quà từ em  $t_3$ ,

...

em  $t_{k-1}$  sẽ nhận quà từ em  $t_k$ .

và sẽ gặp một trong hai tình huống loại trừ nhau sau đây:

**Tình huống 1:** Ta tìm được một món quà cho em  $t_k$ , nghĩa là với em  $t_k$  ta tìm được một món quà  $j$  còn rỗi ( $B[j] = 0$ ) và  $t_k$  yêu thích ( $c[t_k, j] = 1$ ). Ta gọi thủ tục  $\text{Update}(t_k, j)$  thực hiện dãy các thao tác chuyển quà liên hoàn từ em này cho em kia như sau:

em  $t_k$  trao quà  $q_k$  của mình cho bạn  $t_{k-1}$  để nhận quà mới  $j$

em  $t_{k-1}$  trao quà  $q_{k-1}$  của mình cho bạn  $t_{k-2}$  để nhận quà mới  $q_k$  từ tay bạn  $t_k$ ;

...

em  $t_2$  trao quà  $q_2$  của mình cho bạn  $t_1$  để nhận quà mới  $q_3$  từ tay bạn  $t_3$ ;

em  $t_1$  nhận quà  $j$ . Đây chính là em i mà ta cần chia quà.

Ta thu được:  $f(i) = f(t_1) = q_2$ ;  $f(t_2) = q_3$ ; ...;  $f(t_k) = j$  và hoàn thành việc chia quà cho em i trên cơ sở thương lượng các em trong dãy trên nhường quà cho nhau.

**Tình huống 2:** Không gặp tình huống 1, nghĩa là, với mọi em trong dãy  $t_1, t_2, \dots, t_k$  mọi món quà các em yêu thích đều đã được chia cho em nào đó. Nói cách khác, chiến lược nhường quà cho nhau (để nhận quà khác) không mang lại kết quả. Ta kết luận: "không thể chia quà cho em i".

Do không thể chia được quà cho em i ta tiếp tục chia quà cho các em khác.

### Tổ chức dữ liệu

Mảng nguyên  $t[1..n]$ ,  $t[j] = i$ : em j sẽ nhường quà của mình cho bạn i;

Mảng nguyên  $a[1..n]$ ,  $a[i] = j$ : em i đã được chia quà j;

Mảng nguyên  $b[1..m]$ ,  $b[j] = i$ : quà j đang có trong tay em i. Để ý rằng  $a[b[j]] = j$  và  $b[a[i]] = i$ ;

Mảng nguyên  $st[1..n]$ : danh sách các em tạm trả lại quà và đang cần được ưu tiên nhận quà mới.

Biến nguyên p: trả tối ngần trên cùng của stack st.

Mảng nguyên 2 chiều nhị phân  $c[1..n, 1..m]$ ,  $c[i][j] = 1$  khi và chỉ khi em i thích quà j;

Hàm Xep(i): chia quà cho bạn i; Xep(i) = 1 nếu tìm được một cách chia, ngược lại, khi không tìm được Xep = 0.

Hàm Par thực hiện ghép cặp cho các em theo thứ tự từ 1 đến n.

```
(* Autum.pas *)
uses crt;
const fn = 'autum.inp'; gn = 'autum.out';
bl = #32; nl = #13#10; mn = 201;
type mil = array[0..mn] of integer;
      mb2 = array[0..mn, 0..mn] of byte;
var n, m: integer; { n - so nguoi; m - so qua }
      a, b: mil;
      t: mil;
      st: mil; p: integer;
      c: mb2;
      f,g: text; {f: input file; g: output file }
procedure Doc;
var i,j,k,q: integer;
begin
  assign(f,fn); reset(f); read(f,n,m);
  fillchar(c,sizeof(c),0);
  for i := 1 to n do
  begin
    read(f,k);
    for j := 1 to k do begin read(f,q); c[i][q] := 1 end;
  end;
  close(f);
end;
procedure Print;
var i,j: integer;
```

```

begin
  writeln(nl,n,bl,m);
  for i := 1 to n do
    begin
      for j := 1 to m do write(c[i,j]);
      writeln;
    end;
  end;
procedure Update(i,j: integer);
var q: integer;
begin
  repeat
    q := a[i]; { i bỏ qua q }
    a[i] := j; b[j] := i; { để nhận qua j }
    j := q;
    i := t[i]; { chuyển qua người trước }
  until q = 0;
end;
function Xep(i: integer): integer;
var j: integer;
begin
  Xep := 1;
  p := 0; inc(p); st[p] := i; { nạp st }
  fillchar(t, sizeof(t), 0); t[i] := i;
  while p > 0 do
  begin
    i := st[p]; dec(p); { Lấy ngọn st }
    for j := 1 to m do
      if c[i,j] = 1 then { i thích qua j }
      begin
        if b[j] = 0 then { qua j chưa chia }
          begin Update(i,j); exit end
        else if t[b[j]] = 0 { b[j] chưa có trong st }
          then begin inc(p); st[p] := b[j]; t[b[j]] := i end;
        end;
      end;
    Xep := 0;
  end;
end;
procedure Ghi(v: integer);
var i: integer;
begin
  assign(g,gn); rewrite(g);
  writeln(g,v);
  for i := 1 to n do
    if a[i] > 0 then writeln(g,i,bl,a[i]);
  close(g);
end;
procedure Par; { Ghep cap }
var i,v: integer;
begin
  Doc; Print; v := 0;
  fillchar(a, sizeof(a), 0); b := a;
  for i := 1 to n do v := v + Xep(i);
  Ghi(v);
end;
BEGIN
  Par;
  write(nl,' Fini '); readln;
END.

```

```

// DevC++: Autum.cpp
#include <fstream>

```

```

#include <iostream>
using namespace std;
// D A T A   A N D   V A R I A B L E S
const char * fn = "autum.inp";
const char * gn = "autum.out";
const int mn = 201; // so nguoi va so qua toi da
int a[mn]; // a[i] = j: em i nhan qua j
int b[mn]; // b[j] = i: qua j trong tay em i
int t[mn]; // t[j] = i : em j nhuong qua cho ban i
int c[mn][mn]; // c[i][j] = 1: i thich qua j
int n; // so em nho
int m; // so qua
int st[mn]; // stack
int p; // con tro stack
// P R O T O T Y P E S
int main();
void Doc();
void Print();
int Par();
int Xep(int);
void Update(int, int);
void PrintArray(int [], int , int );
void Ghi(int);
// I M P L E M E N T A T I O N
int main() {
    Doc();
    Print();
    int v = Par();
    Ghi(v);
    cout << endl << "Xep duoc " << v << " em.";
    PrintArray(a,1,n);
    cout << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
void Ghi(int v) {
    ofstream g(gn);
    g << v << endl;
    for (int i = 1; i <= n; ++i)
        if (a[i] > 0) g << i << " " << a[i] << endl;
    g.close();
}
void PrintArray(int a[], int d, int c) {
    int i;
    cout << endl;
    for (i = d; i <= c; ++i) cout << a[i] << " ";
}
void Update(int i, int j){
    int c;
    do {
        c = a[i]; // i bo qua c xuong
        a[i] = j; b[j] = i; // i nhan qua moi j
        i = t[i]; j = c; // chuyen qua nguoi khac
    } while (c > 0);
}
int Xep(int i) { // tim qua cho em i
    memset(t, 0, sizeof(t));
    int j;
    p = 0; st[++p] = i; t[i] = i;
    while(p > 0) {
        i = st[p--]; // lay ngon stack
        for (j = 1; j <= m; ++j) { // duyet cac mon qua
            if (c[i][j] > 0) { // i thich qua j

```

```

        if (b[j] == 0) { // qua j chua chia
            Update(i,j); return 1;
        }
        if (t[b[j]] == 0) { // b[j] chua co trong danh sach
            st[pp] = b[j]; t[b[j]] = i; // nap
        }
    }
}
return 0; // Khong chon duoc qua cho em i
}
int Par(){ // Cap ghep
int v = 0, i;
memset(a,0,sizeof(a)); memset(b,0,sizeof(b));
for (i = 1; i <= n; ++i) v += Xep(i);
return v;
}
void Print() { // Hien thi ma tran c
    cout << endl << n << " " << m << endl;
    int i,j;
    for (i = 1; i <= n; ++i){
        for (j = 1; j <= m; ++j)
            cout << c[i][j] << " ";
        cout << endl;
    }
}
void Doc(){
    memset(c,sizeof(c),0);
    ifstream f(fn);
    f >> n >> m;
    int i,j,k,r;
    for (i = 1; i <= n; ++i) {
        f >> k;
        for (r = 1; r <= k; ++r) {
            f >> j ; c[i][j] = 1;
        }
    }
    f.close();
}

```

**Nhận xét** Ta có thể dùng ngăn xếp hay hàng đợi trong bài này kết quả không phụ thuộc vào trật tự duyệt.

### 3.2 Domino

Cho lưới đơn vị gồm  $n$  dòng và  $m$  cột. Các ô trong lưới được gán mã số  $1, 2, \dots, n \times m$  theo trật tự từ dòng trên xuống dòng dưới, trên mỗi dòng tính từ trái qua phải. Người ta đặt vách ngăn giữa hai ô kề cạnh nhau. Hãy tìm cách đặt nhiều nhất  $k$  quân domino, mỗi quân gồm hai ô kề cạnh nhau trên lưới và không có vách ngăn ở giữa.

<b>domino.inp</b>	<b>domino.out</b>
4 5 8	10
2 3	1 2
4 5	3 4
6 7	5 10
9 10	6 11
10 15	7 8
7 12	9 14
19 20	12 13
13 18	15 20
	16 17
	18 19

### Dữ liệu

Input file: Text file domino.inp.

Dòng đầu tiên:  $3$  số  $n -$  số dòng,  $m -$  số cột,  $v -$  số vách ngăn.

Tiếp đến là  $v$  dòng, mỗi dòng  $2$  số  $a b$  cho biết vách ngăn đặt giữa hai ô này.

Output file: Text file domino.out.

Dòng đầu tiên  $k -$  số quân domino tối đa đặt được trên lưới.

Tiếp đến là  $k$  dòng, mỗi dòng  $2$  số  $x y$  cho biết số hiệu của 2 ô kề cạnh nhau tạo thành một quân domino.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

### Thuật toán

Bài này có hai điểm khác với dạng bài cặp ghép truyền thống. Thứ nhất là ghép cặp được thực hiện từ một tập A với chính nó:  $f: A \rightarrow A$ . Thứ hai, mỗi số trong tập A chỉ có thể ghép tối đa với 4 số trong các ô kề cạnh nó mà ta tạm gọi là *số kè*. Thí dụ, 8 có 4 số kè theo thứ tự trái, phải và trên, dưới là 7, 9 và 3, 13. Các số ngoài rìa và các số có vách ngăn còn có ít bạn hơn. Thí dụ, 20 có đúng 1 bạn.

Khi đọc dữ liệu ta xác định ngay cho mỗi số  $i$  trong bảng bốn số kè và ghi vào mảng ke với ý nghĩa như sau  $ke[i][j] = 1$  khi và chỉ khi  $i$  có số kè tại vị trí  $j$ ;  $j = 1, 2, 3, 4$ . Nếu  $i$  có vách ngăn phải thì  $i$  sẽ mất bạn kè phải. Tương tự, nếu  $i$  có vách ngăn dưới thì  $i$  sẽ mất bạn kè dưới. Ngoài ra, dễ hiểu rằng các số trên dòng 1 thì không có số kè trên, các số trên dòng  $n$  thì không có số kè dưới. Tương tự, các số trên cột 1 thì không có số kè trái, các số trên cột  $m$  thì không có số kè phải.

Nhắc lại rằng ta chỉ cần sử dụng một mảng  $a$  với ý nghĩa  $a[i] = j$ , đồng thời  $a[j] = i$  cho biết số  $i$  chọn số  $kè j$  để ghép thành 1 quân domino. Nói cách khác nếu ta xếp được  $f(i) = j$  thì ta phải gán  $a[i] = j$  và  $a[j] = i$ .

Tiếp đến ta thực hiện thủ tục Xep( $i$ ) cho mọi số chưa được ghép cặp ( $a[i] = 0$ ) và chưa là bạn của bất kì số nào ( $b[i] = 0$ ).

Khi ghi kết quả ra file ta lưu ý là hai cặp ( $i, a[i] = j$ ) và ( $j, a[j] = i$ ) thực ra chỉ tạo thành một quân domino duy nhất, do đó ta chỉ cần ghi một trong hai cặp đó. Ta chọn cặp  $i < a[i]$  để tránh trường hợp không xếp được cho số  $i$ , vì khi đó  $a[i] = 0$  tức là  $i > a[i]$ .

```
(* Domino.pas *)
uses crt;
const maxmn = 201; fn = 'domino.inp'; gn = 'domino.out';
      bl = #32; nl = #13#10; phai = 1; trai = 2; tren = 3; duoi = 4;
type mil = array[0..maxmn] of integer;
var n: integer; { so dong }
     m: integer; { so cot }
     nm: integer; { nm = n.m }
     f,g: text;
     ke: array[0..maxmn,phai..duoi] of Boolean;
     st: mil; { stack } p: integer; { ngon st }
     a, t: mil;
procedure Doc;
var i, j, k, v, t: integer;
begin
  fillchar(ke,sizeof(ke),true);
  assign(f,fn); reset(f);
  readln(f,n,m,v); { v - so vach ngan }
  nm := n*m;
  { dong 1 va dong n } k := (n-1)*m;
  for j := 1 to m do
  begin
    ke[j, tren] := false; ke[j+k, duoi] := false;
  end;
```

```

j := 1; { cot 1 va cot m }
for i := 1 to n do
begin
    ke[j, trai] := false; j := j + m; ke[j-1, phai] := false;
end;
for k := 1 to v do
begin
    readln(f,i,j);
    if i > j then begin t := i; i := j; j := t end; { i < j }
    if i = j-1 then ke[i,phai] := false else ke[i,duoi] := false;
end;
close(f);
end;
procedure Update(i,j: integer);
var q: integer;
begin
repeat
    q := a[i]; { i bo so q }
    a[i] := j; a[j] := i; { de nhan so j }
    j := q;
    i := t[i]; { chuyen qua so truoc }
until q = 0;
end;
function Xep(i: integer): integer;
var j, k: integer;
begin
    Xep := 1;
    p := 0; inc(p); st[p] := i; { nap st }
    fillchar(t, sizeof(t), 0); t[i] := i;
    while p > 0 do
begin
    i := st[p]; dec(p); { Lay ngon st }
    for k := 1 to 4 do
        if ke[i,k] then { i co o ke }
        begin
            case k of
                tren: j := i - m;
                duoi: j := i + m;
                phai: j := i + 1;
                trai: j := i - 1;
            end;
            if a[j] = 0 then { j chua bi chiem }
                begin Update(i,j); exit end
            else if t[a[j]] = 0 { a[j] chua co trong st }
                then begin inc(p); st[p] := a[j]; t[a[j]] := i end;
            end;
        end;
    Xep := 0;
end;
function Par: integer;
var i, v: integer;
begin
    fillchar(a,sizeof(a), 0); v := 0;
    for i := 1 to nm do
        if a[i] = 0 then v := v + Xep(i);
    par := v;
end;
procedure Ghi(k: integer);
var i: integer;
begin
    assign(g,gn); rewrite(g);
    writeln(g, k);
    for i := 1 to nm do

```

```

        if i < a[i] then
            writeln(g,i,bl,a[i]);
        close(g);
    end;
procedure Run;
var k: integer;
begin
    Doc; k := Par; Ghi(k);
end;
BEGIN
    Run;
    writeln(nl,' Fini'); readln;
END.

// DevC++: Domino.cpp
#include <iostream>
#include <fstream>
using namespace std;
// DATA AND VARIABLES
const char * fn = "domino.inp";
const char * gn = "domino.out";
const int Maxmn = 201; // tich max n.m
const int phai = 0;
const int duoi = 1;
const int tren = 2;
const int trai = 3;
int a[Maxmn]; // a[i] = j, a[j] = i: i chon j
int t[Maxmn]; // t[j] = i : j nhuong cho i
bool ke[Maxmn][4]; // moi so co toi da 4 so ke
int n; // so dong
int m; // so cot
int nm;
int st[Maxmn]; // stack
int p; // con tro stack
// PROTOTYPES
int main();
void Doc();
int Par();
int Xep(int);
void Update(int, int);
void Ghi(int);
// IMPLEMENTATION
int main() {
    Doc();
    int k = Par();
    Ghi(k);
    cout << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
void Ghi(int k) {
    ofstream g(gn);
    g << k << endl;
    for (int i = 1; i <= nm; ++i)
        if (i < a[i])
            g << i << " " << a[i] << endl;
    g.close();
}
void Update(int i, int j){
    int c;
    do {
        c = a[i]; // i bo c xuong
        a[i] = j; a[j] = i; // i nhan so moi j

```

```

        i = t[i]; j = c; // chuyen qua so khac
    } while (c > 0);
}
int Xep(int i) { // tim so ghep voi i
    memset(t, 0, sizeof(t));
    int j, k;
    p = 0; st[++p] = i; t[i] = i;
    while(p > 0) {
        i = st[p--]; // lay ngan stack
        for (j = 0; j < 4; ++j) { // duyet cac o ke
            if (ke[i][j] > 0) { // co o ke
                switch(j) {
                    case tren: k = i-m; break;
                    case duoi: k = i+m; break;
                    case phai: k = i+1; break;
                    case trai: k = i-1; break;
                }
                if (a[k] == 0) { // o k chua bi chiem
                    Update(i,k); return 1;
                }
                if (t[a[k]] == 0) { // a[k] chua co trong danh sach
                    st[++p] = a[k]; t[a[k]] = i; // nap
                }
            }
        }
    }
    return 0; // Khong chon duoc cap ghep cho i
}
int Par(){ // Cap ghep
    int v = 0, i;
    memset(a,0,sizeof(a));
    for (i = 1; i <= nm; ++i)
        if (a[i]==0) v += Xep(i);
    return v;
}
void Doc(){
    int i, j, k, r, v; // so vach ngan
    int n1, m1, ij;
    ifstream f(fn);
    f >> n >> m >> v; nm = n*m;
    n1 = n-1; m1 = m-1;
    memset(ke,1, sizeof(ke));
    // duyet cac o trong
    for (j = (n-1)*m, i = 1; i <= m; ++i)
        ke[i][tren] = ke[i+j][duoi] = 0;
    for (j = m-1, i = m; i <= nm; i += m)
        ke[i-j][trai] = ke[i][phai] = 0;
    for (i = 0; i < v; ++i) {
        f >> k >> r;
        if (k > r) { j = k; k = r; r = j; } // k < r
        if (r == k+1) ke[k][phai] = ke[r][trai] = 0;
        else ke[k][duoi] = ke[r][tren] = 0;
    }
    f.close();
}

```

### 3.3 Thám hiểm

Trường A và trường B đều cử n học sinh (HS) tham gia trò chơi Thám hiểm Cung Trăng với n xe tự hành, mỗi xe chờ được 2 người có nhiệm vụ khảo sát một đè tài khoa học và hoạt động tại một vùng riêng trên Cung Trăng với một chủ đề cụ thể ghi trên xe đó. Sau khi xem bản hướng dẫn và trình diễn thử của các xe tự hành, mỗi HS chọn một số xe. Ban tổ chức (BTC) cần chia các em thành các nhóm, mỗi nhóm 2 bạn sẽ

*làm việc trên một xe, một bạn của trường A, một bạn của trường B sao cho 2 bạn trên cùng một xe thì cùng quan tâm đến chủ đề chung. Hãy giúp BTC sắp xếp sao cho nhiều chủ đề nhất được thực hiện.*

*Dữ liệu vào: Text file Discover.inp.*

*Dòng đầu tiên: số tự nhiên n*

*Dòng thứ i trong số n dòng tiếp theo có dạng: k c<sub>1</sub> c<sub>2</sub> ... c<sub>k</sub> trong đó k là số lượng xe HS i của trường A chọn, các số tiếp theo là mã số của các xe đồng thời là mã số của đề tài mà bạn đó chọn.*

*Dòng thứ i trong số n dòng tiếp theo là thông tin của trường B có cấu trúc tương tự như của trường A.*

*Dữ liệu ra: Text file Discover.out gồm n dòng, mỗi dòng 2 học sinh: i của trường A, j của trường B cho biết i và j tạo thành một nhóm do cùng chọn một xe. Dữ liệu trên cùng dòng cách nhau qua dấu cách. Bài toán luôn luôn có nghiệm.*

<b>Discover.inp</b>	<b>Discover.out</b>
5	1 1
3 3 1 4	2 5
2 2 3	4 2
3 4 2 1	3 4
2 1 3	5 3
3 5 4 2	
3 2 3 1	
3 1 4 3	
2 5 2	
3 4 2 1	
3 3 1 2	

### Ý nghĩa

Trường A: 5 HS, trường B: 5 HS, 5 xe = 5 chủ đề.

Kết quả (HS trường A, HS Trường B): (1,1), (2,5), (4,2), (3,4), (5,3)

	<b>Học sinh trường A</b>					<b>Học sinh trường B</b>				
	1	2	3	4	5	1	2	3	4	5
xe	✓1		✓	✓		✓1	✓		✓	✓
1										
xe		✓2	✓		✓	✓		✓	✓	✓2
2										
xe	✓	✓		✓3		✓	✓3			✓
3										
xe	✓		✓4		✓		✓		✓4	
4										
xe				✓5				✓5		
5										

### Thuật toán

Ta thực hiện hai lần ghép cặp Xe-Trường A và Xe-Trường B. Sau lần ghép thứ nhất ta thu được kết quả ghi trong mảng a[1..n] trong đó a[i] = j cho biết xe i sẽ chở bạn j của trường A. Sau lần ghép thứ hai ta lại thu được kết quả ghi trong mảng b[1..n] trong đó b[i] = k cho biết xe i sẽ chở bạn k của trường B. Tổng hợp lại ta có mỗi xe i sẽ chở 2 bạn, bạn a[i] của trường A và bạn b[i] của trường B, i = 1..n.

Khung chương trình khi đó sẽ như sau:

1. Mở file input f "Discover.inp";
2. Đọc số n;

3. Đọc dữ liệu của trường A vào mảng 2 chiều c, c[i][j] = 1 khi và chỉ khi xe i được HS j của trường A chọn;
4. Gọi thủ tục Par(a) để ghép cặp Xe – A cho HS trường A. Kết quả a[i] = j cho biết xe i sẽ chở HS i của trường A, i = 1..n;
5. Đọc tiếp dữ liệu của trường B vào mảng 2 chiều c, c[i][k] = 1 khi và chỉ khi xe i được HS k của trường B chọn;
6. Gọi thủ tục Par(b) để ghép cặp Xe – B cho HS trường B. Kết quả b[i] = k cho biết xe i sẽ chở HS k của trường B, i = 1..n;
7. Ghi các cặp a[i], b[i], i = 1..n vào file output;
8. Đóng các files input và output.

A	B
10110	11011
01101	10111
11010	11001
10101	01010
00001	00100

Bạn lưu ý, với thí dụ đã cho, sau khi đọc n dòng dữ liệu của trường A bạn phải thu được kết quả trong mảng c như mảng A trong bảng. Tương tự, sau khi đọc tiếp n dòng dữ liệu của trường B bạn phải thu được kết quả trong mảng c như mảng B trong bảng.

```
(* Pas: Discover *)
uses crt;
const
fn = 'Discover.inp'; gn = 'Discover.out';
mn = 201; bl = #32; nl = #13#10;
type mil = array[0..mn] of integer;
      mb2 = array[0..mn,0..mn] of byte;
var
f,g: text;
n: integer; { so HS = so xe = so de tai }
c: mb2; { c[i,j] = 1 khi va chi khi hs j chon xe i }
a, b: mil; { xep xe truong a, b }
t, x: mil; { t - tro truoc }
st: mil; { stack }
p: integer; { ngon st }

procedure XemC;
```

```

var i,j: integer;
begin
  write(nl,n);
  for i := 1 to n do
  begin
    writeln;
    for j := 1 to n do write(c[i,j]);
  end;
end;
{ xep lien hoan, xe i, hs j }
procedure Update(var a: mil; i, j: integer);
var c: integer;
begin
  repeat
    c := a[i]; { hs c roi xe i }
    a[i] := j; x[j] := i; { hs j len xe i }
    j := c; i := t[i]; { xet hs tiep }
  until c = 0;
end;
function XepXe(var a: mil; i: integer): integer; { xep xe i }
var j: integer;
begin
  XepXe := 1; p := 1; st[p] := i;
  fillchar(t, sizeof(t), 0); t[i] := i;
  while (p > 0) do
  begin
    i := st[p]; dec(p); { xe i }
    for j := 1 to n do { hs j }
      if (c[i,j] = 1) then { hs j chon xe i }
      begin
        if x[j] = 0 { hs j chua len xe } then
          begin Update(a,i,j); exit end; { xep duoc xe i }
        { Nap x[j] vao st }
        if t[x[j]] = 0 { x[j] chua duoc nap }
          then begin inc(p); st[p] := x[j]; t[x[j]] := i end;
      end;
    end;
  XepXe := 0; { kho xep duoc xe i }
end;
function Par(var a: mil): integer;
var i, k: integer;
begin
  k := 0; { so xe da xep duoc }
  fillchar(a, sizeof(a), 0); x := a;
  for i := 1 to n do k := k + XepXe(a,i);
  Par := k;
end;
procedure Doc;
var soxe, hs, xe, j: integer;
begin
  fillchar(c, sizeof(c), 0);
  for hs := 1 to n do
  begin
    read(f,soxe);
    for j := 1 to soxe do { Hs chon xe }
      begin read(f,xe); c[xe,hs] := 1 end;
  end;
end;
procedure Ghi;
var i: integer;
begin
  assign(g,gn); rewrite(g);

```

```

        for i := 1 to n do writeln(g,a[i],bl,b[i]);
        close(g); close(f);
end;
procedure Run;
var i: integer;
begin
    assign(f,fn); reset(f); readln(f,n);
    Doc; Par(a);
    Doc; Par(b);
    Ghi;
end;
BEGIN
    Run;
    write(nl,nl,' Fini'); readln;
END.

// DevC++: Discover.cpp
#include <fstream>
#include <iostream>
using namespace std;
// DATA AND VARIABLE
const char * fn = "Discover.inp";
const char * gn = "Discover.out";
const int mn = 501;
int a[mn]; // a[i] - xe i cho HS truong A
int b[mn]; // b[i] - xe i cho HS truong B
int x[mn];
int t[mn];
int c[mn][mn]; // c[i][j] = 1 - xe i duoc HS j chon
int n; // so hoc sinh cua moi truong A, B; so xe
int st[mn]; // stack
int p; // ngon stack

ifstream f(fn); // mo san input file

// PROTOTYPES
int main();
void Doc();
void Print();
int Par(int []);
int Xep(int [], int);
void Update(int [], int, int);
void PrintArray(int [], int , int );
void Ghi();

// IMPLEMENTATION
int main() {
    f >> n; Doc(); Print(); Par(a);
    cout << endl << " Xep xong truong A: "; PrintArray(a,1,n);
    Doc(); f.close(); Par(b);
    cout << endl << " Xep xong truong B: "; PrintArray(b,1,n);
    Ghi();
    cout << endl << " Fini "; cin.get();
    return 0;
}
void Ghi() {
    ofstream g(gn);
    for (int xe = 1; xe <= n; ++xe)
        g << a[xe] << " " << b[xe] << endl;
    g.close();
}
void PrintArray(int a[], int d, int c) { // hien thi mang a[d..c]

```

```

        int i;
        cout << endl;
        for (i = d; i <= c; ++i)  cout << a[i] << " ";
    }
    void Update(int a[], int i, int j){
        int xe;
        do {
            xe = a[i]; // HS i roi xe
            a[i] = j; x[j] = i; // vao xe j
            i = t[i]; j = xe;
        } while (xe > 0);
    }
    int Xep(int a[], int i) { // xep HS vao xe i
        memset(t, 0, sizeof(t));
        int j;
        p = 0; st[++p] = i; t[i] = i;
        while(p > 0){
            i = st[p--]; // lay ngon st, xe i
            for (j = 1; j <= n; ++j) { // duyet cac HS j
                if ((c[i][j] == 1)) { // xe i co the xep HS j
                    if (x[j] == 0) { // HS j chua duoc xep
                        Update(a,i,j); return 1;
                    }
                    if (t[x[j]] == 0) { // xe x[j] chua nap
                        st[++p] = x[j]; t[x[j]] = i;
                    }
                } // if
            } // for
        } // while
        return 0; // Khong xep duoc cho HS i
    }
    int Par(int a[]) {
        int d = 0, i;
        memset(a,0,sizeof(a)); memset(x,0,sizeof(b));
        for (i = 1; i <= n; ++i) d += Xep(a,i);
        return d;
    }
    void Print(){
        cout << endl << n << endl;
        int i,j;
        for (i = 1; i <= n; ++i)
        {
            for (j = 1; j <= n; ++j)
                cout << c[i][j] << " ";
            cout << endl;
        }
    }
    void Doc() {
        memset(c,0,sizeof(c));
        int hs, xe, soxe, r;
        for (hs = 1; hs <= n; ++hs) { // truong A
            f >> soxe;
            for (r = 1; r <= soxe; ++r)
            { f >> xe ; c[xe][hs] = 1; }
        }
    }
}

```

### 3.4 Show

Nhóm n học sinh (HS) tham gia trình diễn phần mềm. Mỗi em được trình bày riêng sản phẩm của mình trước Ban giám khảo (BGK) trong thời hạn 30 phút. BGK làm việc trong m ngày, ngày thứ i có thể nhận v<sub>i</sub> HS. Trong bản đăng ký mỗi HS ghi khả năng có thể trình diễn phần mềm của mình vào những ngày nào. Hãy sắp lịch để các HS được trình diễn theo đúng nguyện vọng.

Dữ liệu vào: Text file show.inp

Dòng đầu tiên: hai số  $n$   $m$ .

Dòng thứ hai: các giá trị  $v_1$   $v_2$  ...  $v_m$

Dòng thứ  $i$  trong số  $n$  dòng tiếp theo:  $k$   $u_1$   $u_2$  ...  $u_k$

trong đó  $k$  là số ngày HS  $i$  chọn, các  $u_j$  là những ngày cụ thể HS  $i$  chọn.

Dữ liệu ra: Text file show.out gồm  $m$  dòng,  
dòng thứ  $i$  cho biết ngày thứ  $i$  BGK duyệt trình diễn  
của những HS nào.

Dữ liệu trên cùng dòng cách nhau qua dấu cách

show.inp	show.out
5 3	4 5
2 2 1	1 3
1 2	2
2 1 3	
2 2 3	
1 1	
1 1	

### Ý nghĩa

5 học sinh, 3 ngày trình diễn

	Ngày 1	Ngày 2	Ngày 3
HS 1		✓	
HS 2	✓		✓
HS 3		✓	✓
HS 4	✓		
HS 5	✓		

### Thuật toán

Bài này khá dễ giải nếu ta biết cách biến đổi *ngày thành phiên* theo ý tưởng như sau. Ta hiểu mỗi lượt trình diễn của một HS là một phiên.

	Ngày 1	Ngày 2	Ngày 3		
Phiên	P1	P2	P3	P4	P5
HS 1			✓	✓	
HS 2	✓	✓			✓
HS 3			✓	✓	✓
HS 4	✓	✓			
HS 5	✓	✓			

### Đổi ngày thành phiên:

Nếu trong một ngày nào đó BGK chỉ chấm được cho k HS thì ngày đó có k phiên. Đánh số tuần tự các phiên 1, 2, ...

Ngày 1 có 2 phiên 1 và 2; ngày 2 có 2 phiên 3 và 4; ngày 3 có 1 phiên 5.

Ngày đăng kí của HS cũng được đổi theo. HS 1 đăng kí ngày 1 sẽ đổi thành đăng kí 2 phiên 1 và 2; HS 2 đăng kí 2 ngày 1 và 3 sẽ đổi thành đăng kí 3 phiên 1, 2 và 5,...

Khi đọc dữ liệu ta ghi vào mảng s, s[i] cho biết số hiệu phiên của cuối mỗi ngày. Với thí dụ trên, sa khi đọc dữ liệu bạn phải tính được số phiên sp = 5 và s[0..3] = (0,2,4,5). Ngày thứ nhất kết thúc tại phiên 2; ngày thứ hai kết thúc tại phiên 4 và ngày cuối cùng, ngày thứ m = 3 kết thúc tại phiên 5.

Sau đó bạn tổ chức mảng 2 chiều c, c[i,j] cho biết HS i thích trình diễn tại phiên j. Chú ý rằng HS đăng kí 1 ngày có thể sinh ra nhiều phiên tùy thuộc vào ngày hôm đó có mấy phiên. Phiên chính là số HS được BGK cho phép trình diễn trong ngày đó.

Sau khi thực hiện thủ tục cặp ghép như các bài trước, bạn cần duyệt lại kết quả để đổi phiên thành ngày.

```
(* Show.pas *)
uses crt;
(* DATA AND VARIOUS *)
const fn = 'show.inp'; gn = 'show.out';
mn = 101; bl = #32; nl = #13#10;
type mil = array[0..mn] of integer;
mb2 = array[0..mn,0..mn] of byte;
var
n, m: integer; { n - so hs; m - so ngay }
s: mil; { s[i] - phien cuoi cua ngay i }
a,b: mil;
c: mb2; { c[i,j] = 1 khi va chi khi hs i chon phien j }
t, st: mil; { stack st }
p: integer; { ngon st }
sp: integer; { so phien lam viec cua BGK }
f,g: text;
(* IMPLEMENTATION *)
function ToNgay(p: integer): integer; { Doi Phien p -> ngay }
var ng: integer;
begin
ng := 1;
while (p > s[ng]) do inc(ng);
ToNgay := ng
end;
procedure Ghi;
var i,j,ng: integer;
begin
fillchar(c,sizeof(c),0);
for i := 1 to n do
if (a[i] > 0) then begin ng := ToNgay(a[i]); c[ng,i] := 1;
end;
{ Xep ngay cho hs i }
assign(g,gn); rewrite(g);
for i := 1 to m do { Duyet c theo ngay i }
begin
for j := 1 to n do { Hs j }
if (c[i,j] > 0) then write(g,j,bl);
writeln(g);
end;
close(g);
```

```

end;
procedure Update(i,j: integer);
var c: integer;
begin
repeat
  c := a[i];
  a[i] := j; b[j] := i;
  i := t[i]; j := c;
until c = 0;
end;
function XepHs(i: integer): integer;
var j: integer;
begin
  XepHs := 1; fillchar(t,sizeof(t),0);
  p := 1 ; st[p] := i; t[i] := i;
  while(p > 0) do
begin
  i := st[p]; dec(p); { Lay ngon st, xep phien cho hs i }
  for j := 1 to sp do { duyet cac phien }
    if c[i,j] = 1 then { hs i chon phien j }
    begin
      if b[j] = 0 then { Phien j con roi }
        begin Update(i,j); exit end;
      if (t[b[j]] = 0) then { hs b[j] chua nap }
        begin inc(p); st[p] := b[j]; t[b[j]] := i; end;
      end;
    end;
  XepHs := 0; { Ko xep duoc phien cho hs i }
end;
function Par: integer;
var i, d: integer;
begin
  fillchar(a, sizeof(a),0); b := a;
  d := 0;
  for i := 1 to n do d := d + XepHs(i);
  Par := d;
end;
procedure Doc;
var i,j,k,r,q: integer;
begin
  fillchar(c,sizeof(c),0);
  assign(f,fn); reset(f); readln(f, n, m);
  s[0] := 0;
  for i := 1 to m do { m ngay lam viec }
begin
  read(f,r); { so phien trong ngay i }
  s[i] := s[i-1] + r; { s[i] - phien cuoi cua ngay i }
end;
  sp := s[m]; { Tong so phien }
  for i := 1 to n do { xet nguyen vong cua hs i }
begin
  read(f,k); { so ngay hs i chon }
  for r := 1 to k do
  begin
    read(f,j); { cac phien trong ngay j }
    for q := s[j-1]+1 to s[j] do c[i][q] := 1; { hs i chon
phien q }
  end
end;
  close(f);
end;
procedure PA(var a: mil; d,c: integer); { Print array a[d..c] }
var i: integer;

```

```

begin for i := d to c do write(a[i],bl); end;
procedure Xem;
var i,j: integer;
begin
  write(nl,n,bl,m,nl);
  for i := 1 to n do
    begin
      writeln;
      for j := 1 to sp do write(c[i,j]);
    end;
end;
BEGIN
  Doc; Xem; writeln(nl,' Xep duoc: ',Par);
  Ghi;
  write(nl,' Fini '); readln;
END.

```

```

// DevC++: Show.cpp
#include <iostream>
#include <fstream>
using namespace std;
// D A T A   A N D   V A R I A B L E
const char * fn = "show.inp";
const char * gn = "show.out";
const int mn = 101;
int a[mn];
int b[mn];
int t[mn];
int c[mn][mn];// c[i][j] = 1 khi va chi khi HS i dang ki phien j
int n; // so hoc sinh
int m; // so ngay lam viéc cua BGK
int sp; // so phien lam viec cua BGK
int s[mn]; // s[i] - ma so phien cuoi cua ngay i
int st[mn]; // stack
int p; // ngon stack
// P R O T O T Y P E S
int main();
void Doc();
int Par();
int Xep(int);
void Update(int, int);
void Ghi();
int ToNgay(int);
// I M P L E M E N T A T I O N
int main() {
  Doc(); Par(); Ghi();
  cout << endl << " fini "; cin.get();
  return 0;
}
int ToNgay(int p) { // Phien -> ngay
  int ng = 1;
  while (p > s[ng]) ng++;
  return ng;
}
void Ghi() {
  int i,j;
  memset(c,0,sizeof(c));
  for (i = 1; i <= n; ++i) // hs i
    if (a[i] > 0) c[ToNgay(a[i])][i] = 1;
  // c[j][i] = 1 neu hs i duoc xep cho ngay j
  ofstream g(gn);
  for (int i = 1; i <= m; ++i){ // ngay i
    for (j = 1; j <= n; ++j) // hs j
      if (c[j][i] > 0) g << i << " ";
  }
}

```

```

        if (c[i][j] > 0) g << j << " ";
        g << endl;
    }
    g.close();
}
void Update(int i, int j){
    int c;
    do {
        c = a[i];
        a[i] = j; b[j] = i;
        i = t[i]; j = c;
    } while (c > 0);
}
int Xep(int i) { // xep cho HS i
    memset(t, 0, sizeof(t));
    int j;
    p = 0; st[++p] = i; t[i] = i;
    while(p > 0){
        i = st[p--];
        for (j = 1; j <= sp; ++j) { // duyet cac buoi
            if (c[i][j] > 0) { // i chon j
                if (b[j] == 0) { // j con roi
                    Update(i,j); return 1;
                }
                if (t[b[j]] == 0) { // b[j] chua nap
                    st[++p] = b[j]; t[b[j]] = i;
                }
            }
        }
    }
    return 0; // Khong xep duoc cho HS i
}
int Par(){
    int d = 0, i;
    memset(a,0,sizeof(a)); memset(b,0,sizeof(b));
    for (i = 1; i <= n; ++i) d += Xep(i);
    return d;
}
void Doc(){
    memset(c,0,sizeof(c));
    ifstream f(fn);
    f >> n >> m; // n - so hs; m - so ngay
    int i,j,k,r,q;
    s[0] = 0;
    for (i = 1; i <= m; ++i) {
        f >> r; // so phien trong ngay i
        s[i] = s[i-1] + r; // s[i] - phien cuoi cua ngay i
    }
    sp = s[m];
    for (i = 1; i <= n; ++i){ // nguyen vong cua hs i
        f >> k; // so ngay hs i chon
        for (r = 1; r <= k; ++r){
            f >> j ; // HS i chon ngay j
            for (q = s[j-1]+1; q <= s[j]; ++q) // duyet theo phien
                c[i][q] = 1;
        }
    }
    f.close();
}

```

### 3.5 Cấp ghép cực đại: Chị Hằng 2

Nội dung giống bài cấp ghép với một thay đổi như sau:  $c[i][j] = v$  cho biết em i yêu thích món quà j với mức độ  $v_i$ ,  $0 \leq v_i \leq 10$ . Yêu cầu ghép cấp sao cho tổng độ yêu thích đạt max.

<b>autum2.inp</b>	<b>autum2.out</b>	<i>Input text file: autum2.inp</i>
5 5	60	Dòng đầu tiên: hai số n và m, trong đó n là số em nhỏ; m là số quà. Tiếp đến là n dòng của ma trận $c[1..n, 1..m]$ , mỗi dòng m số; $m \leq n$ .
1 2 3 10 5	1 4	<i>Output text file: autum2.out</i>
1 2 3 4 11	2 5	Dòng đầu tiên: giá trị max tổng độ yêu thích đạt được theo phương án ghép cấp đã chọn. Tiếp đến là n dòng, mỗi dòng 2 số $i j$ cho biết em i nhận quà j.
12 2 3 4 5	3 1	
1 13 3 4 5	4 2	
1 2 14 4 5	5 3	

#### Thuật toán

Ta quan lí một giá trị dmax là giá trị gia tăng nhiều nhất trong các phương án xếp quà cho em i. Thí dụ, sau khi đã xếp quà cho  $i-1$  em đầu tiên ta chuyển qua xếp quà cho em i. Giả sử ta có 2 phương án xếp quà cho em i. Phương án thứ nhất có thể tăng giá trị dmax lên thêm  $v_1$  đơn vị, phương án thứ hai có thể tăng giá trị dmax lên thêm  $v_2 > v_1$  đơn vị. Ta sẽ chọn phương án thứ hai. Để thực hiện điều này ta cần lưu ý mấy giá trị sau đây.

- $a[i]$  là quà em i hiện giữ trên tay,  $b[j]$  là em đang giữ quà j. Ta đã biết  $b[a[i]] = i$  và  $a[b[j]] = j$ , ngoài ra, nếu em i chưa được chia quà thì ta đặt  $a[i] = 0$ , và nếu món quà j chưa được chia cho em nào thì  $b[j] = 0$ .
- $c[i,j]$  là *độ yêu thích* của em i đối với món quà j mà ta tạm gọi là *giá trị của món quà j đối với em i* hay vẫn tắt là *giá trị*. Đè ý rằng cùng một món quà j nhưng em i sẽ đánh giá khác với em k, tức là nói chung  $c[i,j] \neq c[k,j]$ .

Khác với phương pháp ghép cặp không phân biệt mức độ yêu thích trước kia, mỗi khi tìm được một dãy liên hoàn em  $t_1$  nhận quà từ em  $t_2$ ,  $t_2$  nhận quà từ em  $t_3$ , ...,  $t_k$  sẽ nhận món quà q chưa chia

$$i = t_1 \leftarrow t_2 \leftarrow \dots t_{k-1} \leftarrow t_k = j \quad (*)$$

thì ta đánh giá xem nếu quyết định kết thúc thủ tục Xep(i) bằng cách thực hiện dãy liên hoàn (\*) thì giá trị *gia tăng dmax* của phương án này là bao nhiêu và cập nhật giá trị đó.

Gọi d là giá trị gia tăng của phương án chia quà. Mỗi khi em j nhường quà  $a[j]$  của mình cho bạn i để nhận quà mới q thì giá trị của phương án sẽ thay đổi như sau:

- d giảm một lượng  $c[j,a[j]]$  khi em j đặt quà  $a[j]$  xuống;
- d tăng một lượng  $c[j,q]$  khi em j nhận quà mới q;
- d tăng một lượng  $c[i,a[j]]$  khi em i nhận quà  $a[j]$  từ bạn j.

Tổng hợp lại, giá trị gia tăng của việc em j nhường quà cho em i để nhận quà mới q sẽ là  $c[j,q] + c[i,a[j]] - c[i,a[j]]$ . Ta có  $d := d + c[j,q] + c[i,a[j]] - c[i,a[j]]$ .

Kí hiệu  $d(j)$  là giá trị gia tăng của phương án khi em j được đưa vào danh sách nhường quà. Phương án này bắt đầu bằng việc tìm quà cho em i, do đó  $d(i) = 0$ . Mỗi khi đưa j vào danh sách nhường quà cho k ta tính được  $d(j) = d(k) + c[k,a[j]] - c[k,a[j]]$ .

Giả sử ta quyết định kết thúc việc tìm quà cho em i, tức là kết thúc thủ tục xét tại dãy (\*). Khi đó do em j cuối cùng trong dãy nhận quà mới là q thì giá trị gia tăng của phương án sẽ được cộng thêm một lượng  $c[j,q]$  và bằng  $d(j) + c[j,q]$ . Ta so sánh đại lượng này với dmax để ghi nhận tình huống tối ưu.

```
(* Autum2.pas *)
uses crt;
const fn = 'autum2.inp'; gn = 'autum2.out';
mn = 201; { so nguoi va so qua toi da }
bl = #32; nl = #13#10;
type mil = array[0..mn] of integer;
      mb2 = array[0..mn, 0..mn] of byte;
var a: mil; { a[i] = j: em i nhan qua j }
      b: mil; { b[j] = i: qua j trong tay em i }
      t: mil; { t[j] = i : em j nhuong qua cho ban i }
```

```

c: mb2; { c[i][j] = 1: i thich qua j }
d: mil; { d[j]: gia tri gia tang tich luy den em j }
n: integer; { so em nho }
m: integer; { so qua }
st: mil; { stack }
p: integer; { tro ngon st }
imax, jmax, dmax: integer;
procedure Ghi(v: integer);
var vmax, i: integer;
g: text;
begin
vmax := 0;
for i := 1 to n do
  if (a[i] > 0) then vmax := vmax + c[i,a[i]];
assign(g,gn); rewrite(g);
writeln(g,vmax);
for i := 1 to n do
  if (a[i] > 0) then writeln(g,i,bl,a[i]);
close(g);
end;
procedure PrintArray(var a: mil; d,c: integer);
{ Hien thi mang a[d..c] }
var i: integer;
begin for i := d to c do write(a[i],bl); end;
procedure Update(i,j: integer);
var c: integer;
begin
repeat
  c := a[i]; { i bo qua c }
  a[i] := j; b[j] := i; { i nhan qua moi j }
  i := t[i]; j := c; { chuyen qua nguoi khac }
until c = 0;
end;
procedure Mark(i,j: integer);
{ ghi nhan phuong an i nhan qua j }
var sum: integer;
begin
sum := d[i] + c[i,j]; { neu i nhan qua moi j }
if (sum > dmax) then { ghi nhan phuong an tot hon }
begin
  dmax := sum;
  imax := i; jmax := j;
end
end;
procedure Nap(i,j: integer);
{ Nap em j vao danh sach nhuong qua cho em i }
begin
  if (t[j] > 0) then exit; { j da co trong st }
  inc(p); st[p] := j; t[j] := i; { j se nhuong qua cho i }
  { dieu chinh gia tri tich luy }
  d[j] := d[i] + c[i,a[j]] - c[j,a[j]];
end;
function Xep(i: integer): integer; { tim qua cho em i }
var j: integer;
begin
fillchar(t,sizeof(t),0); d := t;
{ d[j] - do gia tang tich luy den em j }
dmax := 0; p := 0; Nap(i,i);
while (p > 0) do
begin
  i := st[p]; dec(p); { lay ngon stack }
  for j := 1 to m do { duyet cac mon qua }
    if (b[j] = 0) then Mark(i,j) { qua j chua chia }

```

```

        else Nap(i,j); { danh sach nhuong qua }
    end;
    Xep := 0;
    if (dmax = 0) then exit;
    Update(imax, jmax);
    Xep := 1; { Xep duoc qua cho em i }
end;
function Par: integer; { Ghep cap }
    var i, v: integer;
begin
    v := 0;
    fillchar(a,sizeof(a),0); b := a;
    for i := 1 to n do v := v + Xep(i);
    Par := v;
end;
procedure Print; { Hien thi ma tran c }
    var i,j: integer;
begin
    writeln(nl,' Input: ');
    for i := 1 to n do
    begin
        writeln;
        for j := 1 to m do write(c[i,j],bl);
    end;
end;
procedure Doc;
    var f: text;
    i,j,k,r: integer;
begin
    fillchar(c,sizeof(c),0);
    assign(f,fn); reset(f);
    readln(f,n,m);
    for i := 1 to n do
        for j := 1 to m do
            read(f,c[i,j]);
    close(f);
end;
procedure Run;
    var v: integer;
begin
    Doc; Print;
    v := Par;
    Ghi(v);
    writeln(nl, ' ket qua: '); PrintArray(a,1,n);
end;
BEGIN
    Run;
    writeln(nl,' Fini');
    readln;
END.

// Dev-C++: Autum2
#include <fstream>
#include <iostream>
using namespace std;
// DATA AND VARIABLES
const char * fn = "autum2.inp";
const char * gn = "autum2.out";
const int mn = 201; // so nguoi va so qua toi da
int a[mn]; // a[i] = j: em i nhan qua j
int b[mn]; // b[j] = i: qua j trong tay em i
int t[mn]; // t[j] = i : em j nhuong qua cho ban i
int c[mn][mn]; // c[i][j] = 1: i thich qua j

```

```

int d[mn]; // d[j] khi j nhuong qua cho i
int n; // so em nho
int m; // so qua
int st[mn]; // stack
int p; // con tro stack
int imax, jmax, dmax;
// P R O T O T Y P E S
int main();
void Doc();
void Print();
int Par();
int Xep(int);
void Update(int, int);
void PrintArray(int [], int , int );
void Ghi(int);
void Nap(int, int);
void Mark(int, int);
// I M P L E M E N T A T I O N
int main() {
    Doc();
    Print();
    int v = Par();
    Ghi(v);
    cout << endl << "    Ket qua: "; PrintArray(a,1,n);
    cout << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
void Ghi(int v) {
    int vmax = 0;
    for (int i = 1; i <= n; ++i)
        if (a[i] > 0) vmax += c[i][a[i]];
    ofstream g(gn);
    g << vmax << endl;
    for (int i = 1; i <= n; ++i)
        if (a[i] > 0) g << i << " " << a[i] << endl;
    g.close();
}
void PrintArray(int a[], int d, int c) {
    int i;
    for (i = d; i <= c; ++i) cout << a[i] << " ";
}
void Update(int i, int j){
    int c;
    do {
        c = a[i]; // i bo qua c
        a[i] = j; b[j] = i; // i nhan qua moi j
        i = t[i]; j = c; // chuyen qua nguoi khac
    } while (c > 0);
}
void Mark(int i, int j) {
    int sum = d[i]+c[i][j];// neu i nhan qua moi j
    if (sum > dmax) { // ghi nhan phuong an tot hon
        dmax = sum;
        imax = i; jmax = j;
    }
}
void Nap(int i, int j) { // Nap j vao st
// em j se nhuong qua a[j] cho em i
// em i bo qua a[i] de lay a[j]
    if (t[j]>0) return; // j da co
    st[++p] = j; t[j] = i;
    d[j] = d[i] + c[i][a[j]] - c[j][a[j]] ;// do lech
}

```

```

}

int Xep(int i) { // tim qua cho em i
    memset(t, 0, sizeof(t));
    memset(d, 0, sizeof(d)); // d[i] - do lech neu i nhuong qua
    int j;
    dmax = 0; p = 0; Nap(i,i);
    while(p > 0) {
        i = st[p--]; // lay ngon stack
        for (j = 1; j <= m; ++j) // duyet cac mon qua
            if (b[j] == 0) Mark(i,j); // qua j chua chia
            else Nap(i,j); // danh sach nhuong qua
    } // while
    if (dmax = 0) return 0;
    Update(imax, jmax);
    return 1; // Xep duoc qua cho em i
}
int Par(){ // Cap ghep
    int v = 0, i;
    memset(a,0,sizeof(a)); memset(b,0,sizeof(b));
    for (i = 1; i <= n; ++i) v += Xep(i);
    return v;
}
void Print() { // Hien thi ma tran c
    cout << endl << " input: ";
    cout << endl << n << " " << m << endl;
    int i,j;
    for (i = 1; i <= n; ++i){
        for (j = 1; j <= m; ++j)
            cout << c[i][j] << " ";
        cout << endl;
    }
}
void Doc(){
    memset(c,0,sizeof(c));
    ifstream f(fn);
    f >> n >> m;
    int i,j;
    for (i = 1; i <= n; ++i)
        for (j = 1; j <= m; ++j)
            f >> c[i][j];
    f.close();
}

```

## Chương 4 Các phép lật và chuyển vị

---

### 4.1 Lật xâu

Cho xâu kí tự s. Hãy lật xâu s, tức là sắp các kí tự của xâu s theo trật tự ngược lại. Thí dụ, với s = "abcd", thì sau khi đảo ta thu được s = "dcba".

#### Thuật toán

Để lật một đoạn s[d..c] trong dãy s bắt kì ta thực hiện liên tiếp các phép đổi chỗ hai phần tử cách nhau đầu và cuối tính dần từ ngoài vào giữa dãy.

#### Độ phức tạp

Nếu đoạn cần lật có chiều dài n, mỗi lần đổi chỗ hai phần tử ta cần 3 phép gán. Tổng cộng có  $n/2$  cặp phần tử do đó số phép gán sẽ là  $3n/2$ .

#### Chương trình

Hàm Rev trong các chương trình sau đây nhận vào một xâu s và hai chỉ số đầu d và cuối c, sau đó thực hiện phép đảo đoạn s[d..c] rồi cho ra chính xâu s đó.

```
(* Reverse.pas *)
uses crt;
const nl = #13#10;
var s: string;
function Rev(var s: string; d,c: integer): string;
var ch: char;
begin
  while (d < c) do
  begin
    ch := s[d]; s[d] := s[c]; s[c] := ch;
    inc(d); dec(c);
  end;
  Rev := s;
end;
BEGIN
  s := 'I have a dream';
  writeln('Given: ',s);
  Rev(s,1,length(s));
  writeln(' => ',s);
  writeln(nl,' Now, the source string is: ', Rev(s,1,length(s)));
  readln;
END.

// DevC++: Reverse.cpp
#include <iostream>
using namespace std;
// P R O T O T Y P E S
int main();
char * Rev(char *s, int d, int c);
// I M P L E M E N T A T I O N
int main() {
  char * s = strdup("I have a dream");
```

```

    cout << endl << " Given: " << s;
    cout << " => " << Rev(s,0,strlen(s)-1);
    cout << endl << " Now, the source string is => "
    << Rev(s,0,strlen(s)-1);
    cout << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
char * Rev(char * s, int d, int c) {
    char ch;
    while (d < c) {
        ch = s[d]; s[d] = s[c]; s[c] = ch;
        ++d; --c;
    }
    return s;
}

```

### Giải thích

Hàm `s = strdup("I have a dream")` cấp phát miền nhớ cho xâu s và khởi trị xâu này bằng dãy kí tự "I have a dream".

## 4.2 Lật số nguyên

Viết hàm `Rev(x)` cho ra số nguyên dạng lật của số nguyên x. Thí dụ,  $Rev(-1234) = -4321$ .

### Thuật toán

Gọi y là kết quả. Ta khởi trị y = 0 sau đó lấy lần lượt các chữ số đầu phải của x ghép vào bên phải số y.

### Độ phức tạp

Nếu số đã cho có n chữ số, mỗi lần chuyển một chữ số ta cần thực hiện một phép chia dư và hai phép nhân. Nếu ta coi thời gian thực hiện phép nhân, chia và chia dư là xấp xỉ bằng nhau thì thuật toán lật số nguyên cần thời gian  $3n$ . Mỗi số nguyên x có  $\lg(x) + 1$  chữ số dạng thập phân. Vậy độ phức tạp cỡ  $\lg(x)$ .

```

(* RevInt.pas *)
uses crt;
var x,y: integer;
function Rev(x: longint): longint;
var y: longint;
begin
    y := 0;
    while x <> 0 do
    begin
        y := y*10 + (x mod 10);
        x := x div 10;
    end;
    Rev := y;
end;
BEGIN
    x := -1234;
    y := Rev(x);
    writeln(' Given: ',x,' => ',y);
    writeln(' Now, the source number is ', Rev(y));
    readln;
END.

// DevC++: RevInt.cpp
#include <fstream>
#include <iostream>
using namespace std;
// P R O T O T Y P E S
int main();
int Rev(int x);
// I M P L E M E N T A T I O N
int main() {

```

```

int y, x = -1234;
cout << endl << " Given: " << x;
cout << " => " << (y = Rev(x));
cout << endl << " Now, the source number is => " << Rev(y);
cout << endl;
system("PAUSE");
return EXIT_SUCCESS;
}
int Rev(int x) {
    int y = 0;
    while (x) {
        y = y*10 + (x % 10);
        x /= 10;
    }
    return y;
}

```

### 4.3 Sân bay vũ trụ

Muốn đưa các con tàu vũ trụ vào đúng quỹ đạo trên không gian người ta cần chọn địa điểm thích hợp để xây dựng đường băng. Nếu đường băng đặt tại vị trí thuận lợi, phù hợp với hướng vận hành của các hành tinh thì sẽ tiết kiệm được nhiều nhiên liệu. Người đã ta xây dựng xong một đường băng tại sân bay vũ trụ. Đường băng gồm  $n$  tám bê tông lớn được đặt tại một vị trí cố định. Trong các tám bê tông chĩa nhiều linh kiện và đường nối tinh vi do đó sơ đồ liên kết rất phức tạp. Tuy nhiên, lúc kiểm tra người ta đã phát hiện ra sự nhầm lẫn lớn:  $i$  tám bê tông đầu đường băng đặt sai vị trí: chúng cần được chuyển về phía cuối đường băng. Rất may là trên công trường lúc này còn một xe đặc chủng có sức chở  $1$  tám bê tông và một càn trục có sức nâng  $1$  tám bê tông. Xe chạy trên đường ray song song với đường băng. Mỗi tám bê tông cần chuyển được tháo các khớp nối và được càn trục cẩu lên đặt trên xe rồi được xe chuyển đến vị trí cần đặt lại. Tại vị trí đó càn trục lại cẩu tám bê tông khỏi xe và đặt vào vị trí thích hợp. Càn trục cũng có thể cẩu trực tiếp  $1$  tám bê tông từ một vị trí đến vị trí còn trống nào đó. Thời gian cẩu và vận chuyển một tám bê tông là đáng kể. Hãy đề xuất một phương án khắc phục sự cố với thời gian ngắn nhất, cụ thể là cần giảm tối đa số lần cẩu bê tông.

#### Thí dụ

1	2	3	4	5	6	7
4	5	6	7	1	2	3

Đường băng gồm  $7$  tám bê tông mã số lần lượt từ  $1$  đến  $7$ .  $3$  tám bê tông đầu tiên là  $1, 2$  và  $3$  bị đặt sai vị trí. Sau khi chuyển lại  $3$  tám này ta thu được đường băng đặt đúng là  $(4, 5, 6, 7, \underline{1}, \underline{2}, \underline{3})$ .

#### Thuật toán

**Phương án 1.** Ta gọi mỗi lần cầu một tấm bê tông là một thao tác. Để chuyển i tấm bê tông từ đầu đường băng về cuối đường băng ta chuyên dần từng tấm. Để chuyển một tấm t từ đầu về cuối đường băng ta thực hiện n+1 thao tác sau đây:

- 1 thao tác: Chuyển tấm t ra xe x;
- n-1 thao tác: dịch dần n-1 tấm trên đường băng lên 1 vị trí;
- 1 thao tác: Chuyển tấm t từ xe vào vị trí cuối đường băng.

Tổng hợp lại, để chuyển i tấm từ đầu về cuối đường băng ta cần  $T_1 = i(n+1)$  thao tác.

Giả sử đường băng có 1000 tấm bê tông và ta cần chuyển 500 tấm bê tông từ đầu về cuối đường băng thì ta cần  $T = 500(1000+1) = 500.1001 = 500500$  thao tác. Lại giả sử mỗi ngày ta có thể thực hiện được 100 thao tác thì thời gian cần thiết để khắc phục sự cố sẽ là:

$$500500/(100 \times 365(\text{ngày})) \approx 13 \text{ năm}$$

**Phương án 2.** Ta vận dụng phép đổi xứng (phép lật) để giải bài toán này. Kí hiệu  $u'$  là dãy lật của dãy  $u$ . Thí dụ,  $u = 1234$  thì  $u' = (1234)' = 4321$ .

Phép lật có các tính chất sau:

1. *Tính khả nghịch hay lũy đẳng:*  $u'' = u$ . *Lật đi lật rồi lật lại một dãy sẽ cho ta dãy ban đầu;*
2. *Cộng tính ngược:*  $(uv)' = v'u'$ . *Lật một dãy gồm hai khúc u và v sẽ cho kết quả là một dãy gồm hai khúc lật riêng rẽ: khúc lật thứ hai v' kết nối với khúc lật thứ nhất u'.*

Gọi  $u$  là khúc đầu gồm  $i$  tấm bê tông đầu đường băng,  $v$  là khúc cuối gồm  $n-i$  tấm bê tông còn lại. Bài toán đặt ra là biến đổi  $uv$  thành  $vu$ :  $uv \Rightarrow vu$ . Vận dụng hai tính chất của phép lật ta có:

$$(u'v')' = v''u'' = vu \quad (*)$$

Ta xét lại thí dụ đường băng gồm 7 tấm bê tông và cần chuyển  $i = 3$  tấm bê tông từ đầu về cuối đường băng. Ta có  $u = 123$ ;  $v = 4567$ .

Nhiệm vụ:  $uv = 1234567 \Rightarrow vu = 4567123$ .

Vận dụng đẳng thức (\*) ta có

$$(u'v')' = ((123)'(4567)')' = (3217654)' = 4567123 = vu$$

Nếu **Rev(s, d, c)** là thủ tục lật đoạn từ chỉ số d đến chỉ số c trong dãy s(1..n) thì biểu thức (\*) nói trên được cài đặt qua ba phép gọi thủ tục Rev như sau:

```
Rev(s, 1, i); { u' }
Rev(s, i+1, n); { v' }
Rev(s, 1, n); { s' = (u'v')' = vu }
```

Ta đã biết, để lật một khúc gồm m phần tử ta cần đổi chỗ lần lượt mỗi cặp phần tử cách đều đầu và cuối. Tổng cộng có  $m/2$  cặp. Mỗi lần đổi chỗ hai phần tử trong một cặp ta cần thực hiện 3 phép gán tương ứng với 3 thao tác cầu. Vậy thuật toán chuyển vị theo công thức (\*) sẽ đòi hỏi:

- $3i/2$  thao tác cho  $u'$ ;
- $3(n-i)/2$  thao tác cho  $v'$ ;
- $3n/2$  thao tác cho  $s'$ ;

Tổng cộng ta cần  $T_2 = 3/2 \cdot (i+(n-i)+n) = 3n$  thao tác.

Với thí dụ đã cho,  $n = 1000$ ,  $i = 500$  ta tính được  $T_2 = 3.1000 = 3000$ , tức là  $3000/100 = 30$  ngày.

Phương án 1 đòi hỏi 13 năm trong khi phương án 2 chỉ cần 1 tháng!

**Chú ý** Nếu m là số lẻ thì khi lật đoạn gồm m phần tử sẽ chỉ cần  $3(m-1)/2$  phép gán, do đó công thức  $T_2$  có thể còn nhỏ hơn  $3n$  tối đa là 6 phép gán.

**Phương án 3.** Có thể vận dụng phép *lấy tích các hoán vị* để giải bài toán với  $n+d$  phép chuyển, trong đó  $d$  là ước chung lớn nhất của  $n$  và  $i$ . Giả sử đường băng a gồm  $n = 15$  tấm bê tông,  $a = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)$  và ta cần chuyển  $i = 6$  tấm từ đầu về cuối đường băng theo yêu cầu của đầu bài. Kết quả cuối cùng phải thu được là  $b = (7, 8, 9, 10, 11, 12, 13, 14, 15, 1, 2, 3, 4, 5, 6)$ . Như vậy ta có phép hoán vị  $a \rightarrow b$  như sau:

<b>a</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>
<b>b</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>

#### Ba hoán vị vòng quanh

- $xe \leftarrow 1 \leftarrow 7 \leftarrow 13 \leftarrow 4 \leftarrow 10 \leftarrow 1$  (xe);
- $xe \leftarrow 2 \leftarrow 8 \leftarrow 14 \leftarrow 5 \leftarrow 11 \leftarrow 2$  (xe);
- $xe \leftarrow 3 \leftarrow 9 \leftarrow 15 \leftarrow 6 \leftarrow 12 \leftarrow 3$  (xe).

Ta sẽ có gắng mỗi lần chuyển 1 tấm vào đúng vị trí cần thiết. So sánh hai dòng a và b của bảng ta có thể thực hiện như sau:

#### Pha thứ nhất

1. Cầu tấm 1 ra xe; vị trí 1 trở thành trống,

2. Cầu tám 7 vào vị trí 1; vị trí 7 trở thành trống,
3. Cầu tám 13 vào vị trí 7; vị trí 13 trở thành trống,
4. Cầu tám 4 vào vị trí 13; vị trí 4 trở thành trống,
5. Cầu tám 10 vào vị trí 4; vị trí 10 trở thành trống,
6. Cầu tám 1 từ xe vào vị trí 10.

Sau 6 thao tác chuyển ta thu được:

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

Ta tiếp tục:

**Pha thứ hai:**

7. Cầu tám 2 ra xe; vị trí 2 trở thành trống,
8. Cầu tám 8 vào vị trí 2; vị trí 8 trở thành trống,
9. Cầu tám 14 vào vị trí 8; vị trí 14 trở thành trống,
10. Cầu tám 5 vào vị trí 14; vị trí 5 trở thành trống,
11. Cầu tám 11 vào vị trí 5; vị trí 11 trở thành trống,
12. Cầu tám 2 từ xe vào vị trí 11.

Đến đây ta thu được:

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

Ta lại chuyên tiếp:

**Pha thứ ba:**

13. Cầu tám 3 ra xe; vị trí 3 trở thành trống,
14. Cầu tám 9 vào vị trí 3; vị trí 9 trở thành trống,
15. Cầu tám 15 vào vị trí 9; vị trí 15 trở thành trống,
16. Cầu tám 6 vào vị trí 15; vị trí 6 trở thành trống,
17. Cầu tám 12 vào vị trí 6; vị trí 12 trở thành trống,
18. Cầu tám 3 từ xe vào vị trí 12.

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

Sau  $T_3 = 18$  lần cầu ta thu được kết quả. Phương án 2 đòi hỏi  $T_2 = 3n = 3.15 = 45$  lần cầu.

Tổng quát, ta hãy tưởng tượng các tám bê tông được xếp thành vòng tròn như trên mặt số đồng hồ, nếu xuất phát từ vị trí  $s_0$  sau ít nhất là k lần chuyên (không tính lần chuyên  $s_0$  ra xe) ta sẽ thu được dãy

$$xe \leftarrow s_0 \leftarrow s_1 \leftarrow s_2 \leftarrow \dots \leftarrow s_k = s_0 \text{ (xe)}$$

Trong đó tám bê tông đầu tiên  $s_0$  được chuyên ra xe và cuối cùng, tại bước thứ k tám đó lại được chuyên vào vị trí  $s_0$ . Ta dễ dàng nhận thấy  $s_{j+1} = (s_j + i) \bmod n$ ,  $j = 0, 1, \dots, k-1$ . Từ đây ta suy ra  $(s_0 + ki) \bmod n = s_0$ , hay  $ki \bmod n = 0$ . Gọi d là ước chung lớn nhất của n và i,  $d = (n, i)$ . Ta có ngay,  $n = dn'$  và  $i = di'$ . Đặt  $k = n' = n/d$ , ta có  $kd = n'd = n$  và  $ki \bmod n = n'i \bmod n = (n/d)i' \bmod n = (ni/d) \bmod n = ni' \bmod n = 0$ . Số pha chuyên là d.

Như vậy tổng cộng sẽ có tất cả  $T_3 = (k+1)d = kd + d = n + d$  lần chuyên, trong đó  $d = (n, i)$ . Với  $n = 15$ ,  $i = 6$  ta tính được  $d = (15, 6) = 3$ , do đó ta chuyên trong  $d = 3$  pha và tổng số lần chuyên sẽ là  $T_3 = 15 + 3 = 18$ .

Hàm Move dưới đây nhận vào hai giá trị: tổng số tám bê tông n và số bê tông đầu tiên cần chuyên về cuối i sau đó giải trình trạng tự chuyên các tám bê tông theo từng pha.

```
(* SanBay.pas *)
uses crt;
const BL = #32; NL = #13#10;
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;
function Move(n, i: integer): integer;
var
```

```

d: integer;
tamdau, tam, vitri: integer;
t: integer; { tong so lan chuyen }
j, p: integer;
a: array[0..1000] of integer;
begin
  d := Ucln(n,i);
  writeln(NL,' Se chuyen trong ', d, ' pha', NL);
  t := 0; tamdau := 1;
  for j := 0 to n do a[j] := j;
  for p := 1 to d do
  begin
    writeln(NL, ' Pha thu ', p, ':', NL);
    while(a[tamdau] <> tamdau) do inc(tamdau);
    tam := tamdau; inc(t); a[tam] := 0;
    write(NL, t, '. Chuyen tam ', tam, ' ra xe');
    readln;
    while (true) do
    begin
      vitri := tam; tam := tam + i;
      if (tam > n) then tam := tam - n;
      inc(t); a[vitri] := tam;
      if (tam <> tamdau) then
      begin
        write(NL,t,'. Chuyen tam ',tam, ' den vi tri ',vitri);
        a[tam] := 0;
      end
      else
      begin
        write(NL,t,'. Chuyen tam ',tamdau,
              ' tu xe vao vi tri ', vitri);
        write(NL,' Xong pha ',p,NL);
        break;
      end;
      readln;
    end { while }
    end ; { end for }
    write(NL,' Ket qua: ');
    for i := 1 to n do write(a[i],BL);
    Move := t;
  end;
var t: integer;
BEGIN
  t := Move(15,6);
  writeln(NL, NL, ' Tong cong ', t, ' phep chuyen');
  writeln(NL,NL,' Fini');
  readln
END.

```

```

// DevC++: SanBay.cpp
#include <string.h>
#include <fstream>
#include <iostream>
using namespace std;
// P R O T O T Y P E S
int Ucln(int a, int b);
int Move(int n, int i);
// I M P L E M E N T A T I O N
main() {
  cout << endl << endl
    << " Tong cong " << Move(15,6) << " phep chuyen";
  cout << endl << " Fini "; cin.get();

```

```

}

int Ucln(int a, int b) {
    int r;
    while (b) {
        r = a % b; a = b; b = r;
    }
    return a;
}

int Move(int n, int i) {
    int d = Ucln(n,i);
    cout << endl << " Se chuyen trong " << d << " pha" << endl;
    int tam, vitri;
    int t = 0; // tong so lan chuyen
    int tamdau = 1; // tam be tong can chuyen dau tien cua moi pha
    int a[n+1];
    int j, p;
    for (j = 0; j <= n; ++j) a[j] = j;
    for (p = 1; p <= d; ++p) {
        cout << endl << " Pha thu " << p << ":" << endl;
        while(a[tamdau] != tamdau) ++tamdau;
        tam = tamdau;
        ++t; a[tam] = 0;
        cout << endl << t << ". Chuyen tam " << tam << " ra xe";
        if (cin.get()=='.') return t;
        while (1) {
            vitri = tam; tam += i;
            if (tam > n) tam -= n;
            ++t; a[vitri] = tam; // a[tam] = 0;
            if (tam != tamdau) {
                a[tam] = 0;
                cout << endl << t << ". Chuyen tam "
                    << tam << " den vi tri " << vitri;
            }
            else {
                cout << endl << t << ". Chuyen tam "
                    << tamdau << " tu xe vao vi tri " << vitri;
                cout << ". Xong pha " << p << endl;
                break;
            }
            if (cin.get()=='.') return t;
        } // end while
    } // end for
    cout << endl << endl << " Ket qua: ";
    for (i = 1; i <= n; ++i) cout << a[i] << " ";
    return t;
}

```

#### 4.4 Cân

Olimpic các nước vùng Baltic, 2004

Người ta cần cân một vật có khối lượng là một số tự nhiên  $n$  gam bằng một bộ quả cân khối lượng  $1, 3, 9, \dots, 3^k, \dots$  gam,  $k = 0, 1, 2, \dots$ , mỗi loại có đúng một quả cân. Vật cần cân được đặt trên đĩa trái. Hãy chọn các quả cân đặt trên hai đĩa để cân thăng bằng.

can.inp	can.out
69	2 3 9 1 81

#### Giải thích

Input text file: số  $n$ ;  $1 \leq n \leq 1000000000$  (1 tỷ).

Output text file: 2 dòng

Dòng 1: số quả cân đặt trên đĩa trái, tiếp đến là các quả cân cụ thể.

Dòng 2: số quả cân đặt trên đĩa phải, tiếp đến là các quả cân cụ thể.

Thí dụ, với khối lượng vật cần  $n = 69$  g đặt trên đĩa trái, ta cần đặt thêm 2 quả cân trên đĩa trái là 3 và 9g; 1 quả cân trên đĩa phải là 81 g. Ta có:

$$69 + 3 + 9 = 81.$$

#### Thuật toán

Đầu tiên ta tạm giả thiết là số lượng quả cân mỗi loại là đủ nhiều để có thể cân mọi khối lượng trong giới hạn cho trước. Khi đó ta biểu diễn n dưới dạng hệ đếm 3 rồi đặt vật cần cân trên đĩa trái và đặt các quả cân tương ứng trên đĩa phải.

Để biểu diễn n dưới dạng hệ b tùy ý ta chia liên tiếp n cho b và ghi nhận các số dư. Trong các phiên bản dưới đây p là mảng nguyên chứa các chữ số trong dạng biểu diễn ngược của số n dưới dạng hệ đếm b, đầu ra của các hàm ToBase là số chữ số trong dạng biểu diễn đó.

```
type mil = array[0..30] of integer;
(* Pascal *)
function ToBase(n: longint; b: integer; var p: mil): longint;
var i: longint;
begin
  fillchar(p, sizeof(p), 0);
  i := 0;
  repeat
    p[i] := n mod b;
    n := n div b;
    inc(i);
  until n = 0;
  ToBase := i;
end;

// DevC++
int ToBase(int n, int b, int *p) {
  int i = 0;
  memset(p, 0, sizeof(p));
  do {
    p[i++] = n % b;
    n /= b;
  } while (n != 0);
  return i;
}
```



### Claude Gaspar Bachet de Méziriac

9/10/1581 – 26/2/1638

Ảnh trái là bìa cuốn Sô học nổi tiếng của Diophantus viết vào khoảng năm 250 tại Trung tâm văn hóa Alexandria do Bachet dịch và xuất bản năm 1621.

### Xuất xứ

**Claude Gaspar Bachet de Méziriac** (1581-1638) nhà ngôn ngữ học, nhà thơ và học giả Pháp chuyên nghiên cứu các tác phẩm cổ điển. Bachet cũng rất đam mê các bài toán đó. Ông đã xuất bản cuốn sách "Những bài toán vui và lý thú về các con số". Ông cũng là người phát biểu bài toán lý thú về chiếc cân đĩa như sau:

*Xác định tối thiểu một bộ quả cân để có thể cân mọi vật có khối lượng từ 1 đến 40g trên một chiếc cân đĩa.*

Bachet cho biết chỉ cần dùng 4 quả cân là 1, 3, 9 và 27.

Cuốn *Sô học* nổi tiếng của Diophantus do Bachet dịch đã tạo cảm hứng cho rất nhiều thế hệ các nhà toán học trên Thế giới.

Nguồn: Internet; Simon Sigh, Định lý cuối cùng của Fermat (Phạm Văn Thiều, Phạm Việt Hưng biên dịch)

Để tiện lập luận ta tạm qui ước gọi quả cân  $3^i$  là quả cân loại i, tức là ta gọi theo số mũ của hệ số 3. Ví dụ dã cho  $n = 69$ , lời gọi  $k = \text{ToBase}(69, 3, \text{phai})$  sẽ cho  $k = 4$  và mảng phai[0..3] = [0, 2, 1, 2] chính là các chữ số hệ đếm 3 trong dạng biểu diễn của số 69. Cụ thể là số 69 được biểu diễn ngược trong hệ đếm 3 sẽ là một số gồm  $k = 4$  chữ số lần lượt tính từ chữ số hàng đơn là 0, 2, 1 và 2, cụ thể là:

$$69 = \underline{0} \cdot 3^0 + \underline{2} \cdot 3^1 + \underline{1} \cdot 3^2 + \underline{2} \cdot 3^3 = 2120_3.$$

Loại quả cân	$0$ $(3^0=1)$	$1$ $(3^1=3)$	$2$ $(3^2=9)$	$3$ $(3^3=27)$	$4$ $(3^4=81)$	Vật cần cân khối lượng $n = 69$ được đặt trên đĩa trái. Trên đĩa phải đặt 3 quả cân: phai[1] = 2 quả cân loại 1, $2 \cdot 3^1 = 6$ g, phai[2] = 1 quả cân loại 2, $1 \cdot 3^2 = 9$ g, phai[3] = 2 quả cân loại 3, $2 \cdot 3^3 = 2.27 = 54$ g, $69 = 6 + 9 + 54$ . Đĩa trái tạm thời để trống
Đĩa trái (69+...)	0	0	0	0	0	
Đĩa phải	0	2	1	2	0	

Vì mỗi loại quả cân chỉ có đúng 1 quả nên ta cần tìm cách thay 2 quả cân cùng loại i bằng tổ hợp khác. Ta có

$$2 \cdot 3^i = 3 \cdot 3^i - 3^i = 3^{i+1} - 3^i.$$

Hệ thức trên cho ta thấy rằng có thể thay 2 quả cân loại i ở *đĩa cân phải* bằng cách đặt 1 quả cân loại  $i+1$  trên *đĩa phải* và 1 quả cân loại i trên *đĩa trái*. Với thí dụ dã cho, phai[1] = 2 nên ta thay 2 quả cân loại 1 bằng 1 quả cân loại 2 trên *đĩa phải* và 1 quả cân loại 1 trên *đĩa trái*. Vì trên *đĩa phải* đã có sẵn 1 quả cân loại 2 nên số quả cân loại này sẽ được tăng thêm 1 và bằng 2. Ta thu được:

Loại quả cân	$0$ $(3^0=1)$	$1$ $(3^1=3)$	$2$ $(3^2=9)$	$3$ $(3^3=27)$	$4$ $(3^4=81)$	phai = (0,2,1,2) $\Rightarrow$ (0,0,2,2); trai = (0,1,0,0). Đĩa trái ( $69$ g)+1 quả cân loại 1 = $69 + 3 = 72$ g; Đĩa phải: 2 quả cân loại 2 + 2 quả cân loại 3 = $2 \cdot 3^2 + 2 \cdot 3^3 = 2.9 + 2.27 = 18 + 54 = 72$ g.
Đĩa trái (69+...)	0	1	0	0	0	
Đĩa phải	0	0	2	2	0	

Lại thực hiện phép thay 2 quả cân loại 2 trên *đĩa phải* bằng 1 quả cân loại 2 trên *đĩa phải* và 1 quả cân loại 3 trên *đĩa trái* ta thu được:

Loại quả cân	$0$ $(3^0=1)$	$1$ $(3^1=3)$	$2$ $(3^2=9)$	$3$ $(3^3=27)$	$4$ $(3^4=81)$	phai = (0,0,2,2) $\Rightarrow$ (0,0,0,3);

Đĩa trái (69+...)	0	1	1	0	0	trai = (0,1,1,0).
Đĩa phải	0	0	0	3	0	Đĩa trái (69g) + 1 quả cân loại 1 + 1 quả cân loại 2 = $69 + 1 \cdot 3^1 + 1 \cdot 3^2 = 69 + 3 + 9 = 81$ g; Đĩa phải: 3 quả cân loại 3 = $3 \cdot 27 = 81$ g.

Cuối cùng ta thay 3 quả cân loại 3 trên đĩa phải bằng 1 quả cân loại 4 là hoàn tất.

$$\text{phai} = (0,0,0,3) \Rightarrow (0,0,0,0,1).$$

$$\text{trai} = (0,1,1,0).$$

Kết quả ta thu được: Để cân vật  $n = 69$  g ta đặt vật đó trên đĩa trái và

Đặt tiếp trên đĩa trái 2 quả cân 3 và 9 g;

Đặt trên đĩa phải 1 quả cân 81 g.

Tổng hợp lại ta có thuật toán Replace thực hiện phép thay các quả cân loại  $i$  trên đĩa phải như sau:

- Nếu trên đĩa phải có 2 quả cân loại  $i$  thì thay bằng 1 quả loại  $i+1$  trên đĩa phải và 1 quả loại  $i$  trên đĩa trái;
- Nếu trên đĩa phải có 3 quả cân loại  $i$  thì thay bằng 1 quả loại  $i+1$  trên đĩa phải.

Hàm Replace nhận vào là dãy  $k$  quả cân trên đĩa phải  $p[0..k-1]$ , cho ra dãy  $m$  quả cân trên đĩa trái  $t[0..m-1]$ :

```
(* Pascal *)
function Replace(var p: mil; var k: integer; var t: mil): longint;
var m, i: longint;
begin
  fillchar(t, sizeof(t), 0);
  for i := 0 to k do
    if p[i] = 3 then begin p[i] := 0; inc(p[i+1]) end
    else if p[i] = 2 then
      begin p[i] := 0; inc(p[i+1]); inc(t[i]) end;
  m := k;
  if p[k] > 0 then inc(k);
  Replace := m;
end;

// DevC++
int Replace(int * p, int &k, int * t) {
  int i, m;
  memset(t, 0, sizeof(t));
  for (i = 0; i < k; ++i)
    if (p[i] == 3) { p[i] = 0; ++p[i+1]; }
    else if (p[i] == 2) { p[i] = 0; ++p[i+1]; ++t[i]; }
  m = k;
  if (p[k] > 0) ++k;
  return m;
}
```

Trước khi ghi file chúng ta cần thu gọn sơ bộ dữ liệu. Ta duyệt các phần tử của hai dãy trái và phải để đếm xem có bao nhiêu quả cân và qui các loại quả cân đó thành giá trị cụ thể, tức là thay vì viết  $i$  ta phải viết  $3^i$ .

```
(* can.pas *)
uses crt;
const fn = 'can.inp'; gn = 'can.out';
  mn = 30; bl = #32; nl = #13#10;
type mil = array[0..mn] of longint;
function Doc: longint;
var n: longint;
  f: text;
begin
  assign(f,fn); reset(f);
  readln(f,n); close(f);
  Doc := n;
end;
function ToBase(n: longint; b: longint; var p: mil): longint;
var i: longint;
begin
```

```

fillchar(p, sizeof(p),0);
i := 0;
repeat
  p[i] := n mod b;
  n := n div b;
  inc(i);
until n = 0;
ToBase := i;
end;
function Replace(var p: mil; var k: longint; var t: mil): longint;
var m, i: longint;
begin
  fillchar(t, sizeof(t),0);
  for i := 0 to k do
    if p[i] = 3 then begin p[i] := 0; inc(p[i+1]) end
    else if p[i] = 2 then
      begin p[i] := 0; inc(p[i+1]); t[i] := 1 end;
  m := k;
  if p[k] > 0 then inc(k);
  Replace := m;
end;
procedure Ghi(var t: mil; dt: longint; var p: mil; dp: longint);
var i, v, nt, np: longint;
  g: text;
begin
  v := 1; nt := 0;
  for i := 0 to dt-1 do
  begin
    if t[i] > 0 then begin inc(nt); t[i] := v; end;
    v := v * 3;
  end;
  v := 1; np := 0;
  for i := 0 to dp-1 do
  begin
    if p[i] > 0 then begin inc(np); p[i] := v; end;
    v := v * 3;
  end;
  assign(g,gn); rewrite(g);
  write(g,nt,bl);
  for i := 0 to dt-1 do
    if t[i] > 0 then write(g,t[i],bl);
  write(g,nl,np,bl);
  for i := 0 to dp-1 do
    if p[i] > 0 then write(g,p[i],bl);
  close(g);
end;
procedure Run;
var trai, phai: mil;
  n, dt, dp: longint;
begin
  n := Doc;
  dp := ToBase(n,3,phai);
  dt := Replace(phai, dp, trai);
  Ghi(trai,dt,phai,dp);
end;
BEGIN
  Run;
  write(nl,' Fini');
  readln;
END.

// Devcpp Can.cpp
#include <iostream>

```

```

#include <fstream>
using namespace std;
// D A T A   A N D   V A R I A B L E
const char * fn = "CAN.INP";
const char * gn = "CAN.OUT";
// P R O T O T Y P E S
int main();
int Doc();
void Ghi(int *t, int n, int *p, int m);
int ToBase(int n, int b, int *p);
int Replace(int *p, int &n, int *t);
void Run();
// I M P L E M E N T A T I O N
int main() {
    Run();
    cout << endl << endl << " Fini ";
    cin.get();
    return 0;
}
void Run() {
    const int mn = 20;
    int phai[mn], trai[mn];
    int dp, dt, n;
    n = Doc();
    dp = ToBase(n, 3, phai);
    dt = Replace(phai, dp, trai);
    Ghi(trai,dt,phai,dp);
}
int Doc() {
    int n;
    ifstream f(fn);
    f >> n;
    f.close();
    return n;
}
void Ghi(int *t, int dt, int *p, int dp) {
    int i, v, nt, np;//dt,dp: so qua can tren dia trai va phai
    nt = np = 0;
    for (v = 1,i = 0; i < dt; ++i, v*= 3)
        if (t[i] > 0) { ++nt; t[i] = v; }
    for (v = 1,i = 0; i < dp; ++i, v*= 3)
        if (p[i] > 0) { ++np; p[i] = v; }
    ofstream g(gn);
    g << nt << " ";
    for (i = 0; i < dt; ++i)
        if (t[i] > 0) g << t[i] << " ";
    g << endl << np << " ";
    for (i = 0; i < dp; ++i)
        if (p[i] > 0) g << p[i] << " ";
    g.close();
}
// Bieu dien so n qua he b
// return i - chieu dai so trong he b
// n = p[0].b^0 + p[1].b^1 + ... + p[i].bi
int ToBase(int n, int b, int *p) {
    int i = 0;
    memset(p, 0, sizeof(p));
    do {
        p[i++] = n % b;
        n /= b;
    } while (n != 0);
    return i;
}

```

```

int Replace(int * p, int &k, int * t) {
    int i, m;
    memset(t, 0, sizeof(t));
    for (i = 0; i < k; ++i)
        if (p[i] == 3) { p[i] = 0; ++p[i+1]; }
        else if (p[i] == 2) { p[i] = 0; ++p[i+1]; ++t[i]; }
    m = k;
    if (p[k] > 0) ++k;
    return m;
}

```

**Độ phức tạp:** cỡ  $\log_3(n)$ ; trong đó  $\log_3(n) + 1$  là số chữ số trong dạng biểu diễn của n theo hệ đếm 3.

#### 4.5 Biprime

Cặp số tự nhiên x và số lật của nó, x' nếu đồng thời là hai số nguyên tố khác nhau thì được gọi là cặp song nguyên tố. Hãy liệt kê các cặp song nguyên tố trong khoảng 1..N = 500000.

biprime.inp	biprime.out	Giải thích
100	4 13 31 17 71 37 73 79 97	Input text file: số N Output text file: Dòng đầu tiên: M – số cặp song nguyên tố. Tiếp đến là M dòng, mỗi dòng một cặp song nguyên tố. Với n = 100 ta tìm được 4 cặp song nguyên tố: (13, 31), (17, 71), (37, 73) và (79, 97). Các số cùng dòng cách nhau qua dấu cách.

#### Thuật toán

Trước hết dùng thuật toán sàng để tìm và ghi nhận các số nguyên tố trong khoảng 1..N. Dùng mảng a đánh dấu các số nguyên tố. Nếu bit thứ i bằng 0 thì i là số nguyên tố. Các thủ tục xử lí bit bao gồm:

- **BitOn(i)**: Đặt bit thứ i trong a bằng 1 (bật bit i);
- **BitOff(i)**: Đặt bit thứ i trong a bằng 0 (tắt bit i);
- **GetBit(i)**: cho giá trị 0/1 của bit thứ i trong dãy bit a.

Với **Nmax** = 500000 thì mảng a có kích thước  $(Nmax+7)/8 = 625000$  byte. Bit thứ i trong dãy a sẽ ứng với bit thứ  $i \% 8$  trong byte **b** =  $i/8$ . Chú ý rằng  $i \% 8 = i \& 7$  và  $i/8 = (i >> 3)$ .

Sau khi gọi thủ tục **Sang** ta duyệt lại dãy bit a, với mỗi số nguyên tố i (**GetBit(i)=0**) ta tìm số lật **ip = Rev(i)**. Nếu **ip ≠ i**, **ip ≤ N** và **ip** cũng là số nguyên tố thì ta đếm số cặp. Ta sử dụng bảng quyết định để xác định khi nào thì cần đánh dấu (đặt **BitOn(i)** hoặc **BitOn(ip)**). Nếu i và số lật ip của nó là cặp song nguyên tố thì ta chỉ cần đánh dấu một trong hai số đó bằng thủ tục **BitOn**. Lần duyệt thứ hai ta chỉ quan tâm những bit i nhận giá trị 0 và ghi lại các cặp i và **Rev(i)**.

Bảng quyết định xóa i và số lật						<b>ip = Rev(i).</b> <i>Xóa x tức là đặt BitOn(x).</i>
Điều kiện	i nguyên tố	yes	yes	yes	yes	
	ip ≤ N	yes	yes	yes	no	
	ip ≠ i	yes	yes	no	–	
	ip nguyên tố	yes	no	–	–	
Quyết định	Xóa i ( <b>BitOn(i)</b> )	no	yes	yes	yes	
	Xóa ip ( <b>BitOn(ip)</b> )	yes	no	no	no	

#### Độ phức tạp

Thủ tục sàng đòi hỏi  $\sqrt{n}$  phép chia dư và n lần duyệt cho mỗi số nguyên tố do đó bài toán đòi hỏi độ phức tạp tính toán cỡ  $n\sqrt{n}$ .

```

(* Biprime.pas *)
uses crt;
const mn = (500000+7) shr 3;
      fn = 'biprime.inp'; gn = 'biprime.out';
      bl = #32; nl = #13#10;

```

```

type mb1 = array[0..mn] of byte;
var a: mb1;
procedure BitOn(i: longint); { bật bit i }
  var p, b: longint;
begin
  b := i shr 3; p := i and 7;
  a[b] := a[b] or (1 shl p);
end;
procedure BitOff(i: longint); { tắt bit i }
  var p, b: longint;
begin
  b := i shr 3; p := i and 7;
  a[b] := a[b] and (not(1 shl p));
end;
function GetBit(i: longint): integer; { nhận giá trị của bit i }
  var p,b: longint;
begin
  b := i shr 3; p := i and 7;
  GetBit := (a[b] shr p) and 1;
end;
procedure Sang(n: longint);
  var i,j: longint;
begin
  fillchar(a,sizeof(a),0);
  for i := 2 to round(sqrt(n)) do
    if GetBit(i) = 0 then
      for j := i to (n div i) do BitOn(i*j);
end;
function Rev(x: longint): longint; { số lật của x }
  var y: longint;
begin
  y := 0;
  while x <> 0 do
  begin
    y := y*10 + (x mod 10);
    x := x div 10;
  end;
  Rev := y;
end;
function Doc: longint;
  var n: longint;
    f: text;
begin
  assign(f,fn); reset(f);
  readln(f,n); close(f);
  Doc := n;
end;
procedure Run;
  var n, i, ip, d: longint;
    g: text;
begin
  n := Doc;
  Sang(n);
  d := 0;
  for i := 13 to n do
    if GetBit(i) = 0 then { i nguyên tố }
    begin
      ip := Rev(i);
      if (ip <= n) and (ip <> i) then
      begin
        if GetBit(ip) = 0 then { ip nguyên tố }
        begin
          inc(d);
        end;
      end;
    end;
  end;
end;

```

```

        BitOn(ip); { xóa ip }
    end else BitOn(i); { xóa i }
end else BitOn(i);
end;
{ Ghi file }
assign(g,gn); rewrite(g);
writeln(g,d);
for i := 13 to n do
    if GetBit(i) = 0 then
        writeln(g,i,bl,Rev(i));
close(g);
end;
BEGIN
Run;
write(nl,' Fini'); readln;
END.
// Devcpp biprime.cpp
#include <iostream>
#include <fstream>
#include <math.h>
using namespace std;
// DATA AND VARIABLE
const char * fn = "biprime.inp";
const char * gn = "biprime.out";
const int mn = (500000+7)>>3;
char a[mn];
// PROTOTYPES
int main();
int Doc();
void BitOn(int i);
void BitOff(int i);
int GetBit(int i);
int Rev(int x);
void Sang(int n);
void Run();
// IMPLEMENTATION
int main() {
    Run();
    cout << endl << endl << " Fini ";
    cin.get();
    return 0;
}
int Doc() {
    int n;
    ifstream f(fn);
    f >> n;
    f.close();
    return n;
}
void BitOn(int i) { // bật bit i
    int b = i >> 3, p = i & 7;
    a[b] |= (1 << p);
}
void BitOff(int i) { // tắt bit i
    int b = i >> 3, p = i & 7;
    a[b] &= ~(1 << p);
}
int GetBit(int i) { // nhận trị của bit i
    int b = i >> 3, p = i & 7;
    return (a[b] >> p) & 1;
}
int Rev(int x) { // số lật của x
    int y = 0;

```

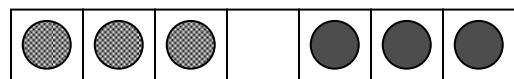
```

        do {
            y = y*10 + (x % 10);
            x /= 10;
        } while (x);
        return y;
    }
    void Sang(int n) {
        int can = int(sqrt(n));
        int i, j, ni;
        for (i = 2; i <= can; ++i)
            if (GetBit(i) == 0)
                for (ni = n/i, j = i; j <= ni; ++j) BitOn(i*j);
        for (i = 13; i <= n; ++i)
            if (GetBit(i) == 0) cout << i << " ";
    }
    void Run() {
        int i, n, d, ip;
        n = Doc();
        memset(a, 0, sizeof(a));
        cout << endl << " n = " << n << " mn = " << mn << endl;
        Sang(n);
        for (d = 0, i = 13; i <= n; ++i)
            if (GetBit(i) == 0) {
                ip = Rev(i);
                if ((ip <= n) && (ip != i)) {
                    if (GetBit(ip) == 0) {
                        ++d; BitOn(ip); // xóa ip
                    } else BitOn(i); // xóa i
                } else BitOn(i); // xóa i
            }
        // Ghi file
        ofstream g(gn);
        g << d << endl;
        for (i = 13; i <= n; ++i)
            if (GetBit(i) == 0)
                g << i << ' ' << Rev(i) << endl;
        g.close();
    }
}

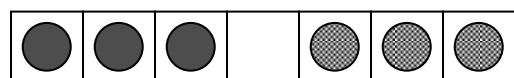
```

#### 4.6 Chuyển bi

Trên một bảng chia  $2n+1$  ô người ta đặt  $n$  viên bi xanh liền nhau, mỗi ô 1 viên, sau đó bỏ một ô trống và đặt tiếp  $n$  viên bi đỏ như hình a. Hãy tìm cách chuyển với số lần ít nhất để thu được hình b, trong đó các viên bi xanh được chuyển qua trái của ô trống, các viên bi đỏ được chuyển qua phải ô trống. Mỗi lần được phép chuyển một viên bi vào ô trống kè viên bi đó hoặc cách viên bi đó 1 ô.



(a)



(b)

<b>balls.inp</b>	<b>balls.out</b>	<i>Giải thích</i>
3	<b>bbborrr bbobrrr bbrborr</b>	Input text file: số N.

	<b>bbrboror</b> <b>bbrorbr</b> <b>borbrbr</b> <b>obrbrbr</b> <b>rbobrbr</b> <b>rbrbобр</b> <b>rbrbrbo</b> <b>rbrbrrob</b> <b>rbrorbb</b> <b>rorbrbb</b> <b>rrobrbb</b> <b>rrrbobb</b> <b>rrrobbb</b> 15	Output text file: Dòng đầu tiên - câu hình xuất phát là một xâu gồm N kí tự 'b' biểu thị bi xanh (blue), tiếp đến là 1 kí tự 'o' biểu thị ô trống, tiếp đến là N kí tự 'r' biểu thị bi đỏ (red). Tiếp đến là M dòng, mỗi dòng là một câu hình thu được sau mỗi lần chuyển. Dòng cuối cùng: M - tổng số lần chuyển.
--	--	--

## Thuật toán

Ta kí hiệu  $x(n)$  là dãy gồm  $n$  chữ cái  $x$ . Bài toán khi đó được phát biểu như sau:

$$b(n)or(n) \Rightarrow r(n)ob(n)$$

Nếu chuyển dàn từng viên bi xanh đến vị trí cần thiết ở bên phải thì mỗi lần chuyển một bi xanh qua phải một vị trí ta phải theo sơ đồ với 2 bước chuyển như sau:

$$\dots bor\dots \Rightarrow \dots obr\dots \Rightarrow \dots rbo\dots$$

Để thực hiện phép chuyển một bi xanh về cuối dãy  $b(n)or(n) = b(n-1)bor(n) \Rightarrow b(n-1)r(n)bo$  ta cần  $2n$  bước chuyển. Sau đó ta lại phải thực hiện  $n+1$  bước để chuyển ô trống về đầu trái của dãy  $r(n)$  theo sơ đồ:

$$b(n-1)r(n)bo \Rightarrow b(n-1)or(n)b$$

Vậy để chuyển một bi xanh về cuối dãy sau đó đưa ô trống về đầu trái của dãy  $r(n)$  theo sơ đồ

$$\dots bor(n) \Rightarrow \dots r(n)bo \Rightarrow \dots or(n)b$$

ta cần  $3n+1$  bước chuyển.

Để chuyển  $n-1$  bi xanh qua phải theo sơ đồ

$$b(n)or(n) = bb(n-1)or(n) \Rightarrow bor(n)b(n-1)$$

ta cần  $(n-1)(3n+1)$  bước chuyển.

Với viên bi xanh còn lại cuối cùng ta sẽ chuyển theo sơ đồ sau

$$bor(n)b(n-1) \Rightarrow r(n)bob(n-1) \quad (2n \text{ bước chuyển}) \Rightarrow r(n)obb(n-1) \quad (1 \text{ bước chuyển}) = r(n)ob(n).$$

Vậy tổng cộng ta cần  $(n-1)(3n+1)+2n+1 = 3n^2+n-3n-1+2n+1 = 3n^2$  bước chuyển.

Với  $n = 3$  ta cần  $3 \cdot 3^2 = 27$  bước chuyển cụ thể như sau:

bbborrr  $\Rightarrow$  bbobrrr  $\Rightarrow$  bbrborr  $\Rightarrow$  bbrrbor  $\Rightarrow$  bbrrobr  $\Rightarrow$  bbrrrbo  $\Rightarrow$  bbrrrob  $\Rightarrow$   
bbrrorb  $\Rightarrow$  bbrrorb  $\Rightarrow$  bborrrb  $\Rightarrow$  brborrb  $\Rightarrow$  brobrbb  $\Rightarrow$  brrborb  $\Rightarrow$  brrobrb  $\Rightarrow$   
brrrbob  $\Rightarrow$  brrrobb  $\Rightarrow$  brrrorbb  $\Rightarrow$  borrrbb  $\Rightarrow$  rborrb  $\Rightarrow$  robrrbb  $\Rightarrow$   
rrborbb  $\Rightarrow$  rrobrbb  $\Rightarrow$  rrrbobb  $\Rightarrow$  rrrobbb.

Ta sẽ cải tiến thuật toán trên để thu được một thuật toán với số bước chuyển là  $n(n+2)$ . Ta gọi thuật toán này là *thuật toán quả lắc* vì cơ chế hoạt động của nó rất giống với dao động của quả lắc. Trước hết ta đề xuất một số heuristics trợ giúp cho việc tối ưu hóa số lần chuyển:

- Không bao giờ chuyển bi đi lùi, nghĩa là bi xanh phải luôn luôn được chuyển *qua phải*, bi đỏ *qua trái*,
- Phải chuyển bi đi *nhanh nhất có thể*, nghĩa là phải tìm cách chuyển bi qua 2 ô thay vì qua một ô mỗi bước.

Ta theo dõi sự di chuyển của ô trống. Ta kí hiệu  $-1$  nếu ô trống được chuyển qua trái 1 vị trí,  $+1$  nếu ô trống được chuyển qua phải 1 vị trí;  $+2(k)$  nếu ô trống được chuyển qua phải  $k$  lần, mỗi lần 2 vị trí và  $-2(k)$  nếu ô trống được chuyển qua trái  $k$  lần, mỗi lần 2 vị trí. Với  $n = 3$  như thí dụ đã cho, ta có dãy gồm 15 phép chuyển ô trống như sau:

Câu hình ban đầu: bbborrr

$-1; +2(1)$ : bbobrrr, bbrborr - dịch ô trống qua trái 1 ô sau đó dịch ô trống qua phải 1 lần nhảy 2 ô,  
 $+1; -2(2)$ : bbrbror, bbrorbr, borbrbr - dịch ô trống qua phải 1 ô sau đó dịch ô trống qua trái 2 lần, mỗi lần 2 ô,  
 $-1; +2(3)$ : obrbrbr, rbobrbr, rbrbopr, rbrbrbo - dịch ô trống qua trái 1 ô sau đó dịch ô trống qua phải 3 lần, mỗi lần 2 ô,  
 $-1; -2(2)$ : rbrbopr, rbrorbb, rorbrbb - dịch ô trống qua trái 1 ô sau đó dịch tiếp ô trống qua trái 2 lần, mỗi lần 2 ô,  
 $+1; +2(1)$ : rrobrbb, rrrbobb, - dịch ô trống qua phải 1 ô sau đó dịch tiếp ô trống qua phải 1 lần nhảy 2 ô,  
 $-1$ : rrrobbb - dịch ô trống qua trái 1 ô. Hoàn thành.

Bạn dễ dàng phát hiện rằng thuật toán trên vận dụng tối đa 2 heuristics nói trên.

Tổng quát, ta mô tả thuật toán theo sơ đồ sau:

Pha 1:  $(-1)^i ; (-1)^{i+1} \cdot 2(i)$ ;  $i = 1, 2, \dots, n$  - hai phép chuyển trái dấu nhau;

Pha 2:  $(-1)^{i+1} ; (-1)^{i+1} \cdot 2(i)$ ;  $i = n-1, \dots, 1$  - hai phép chuyển cùng dấu.

Cuối cùng thực hiện phép chuyển  $-1$ .

Ta sử dụng thủ tục Move(h,k) : chuyển ô trống k lần, mỗi lần h ô,  $h = 1, -1, 2, -2$ . Nếu  $h > 0$  thì chuyển qua phải, ngược lại, khi  $h < 0$  thì chuyển qua trái. Khi đó sơ đồ hai pha nói trên được triển khai như sau:

Pha 1: **Move**(( $-1^i$ , 1); **Move**(( $-1^{i+1} \cdot 2$ , i);  $i = 1, 2, \dots, n$ ).

Pha 2: **Move**(( $-1^{i+1}$ , 1); **Move**(( $-1^{i+1} \cdot 2$ , i);  $i = n-1, \dots, 1$ ).

Nếu ta để ý rằng  $(-1)^i$  và  $(-1)^{i+1}$  trái dấu nhau và  $(-1)^{i+1}$  và  $(-1)^{i+1}$  cùng dấu thì hai pha nói trên có thể cài đặt thông qua một biến nguyên sign quản lý dấu như sau:

```
(* Pascal *)
{ Pha 1 }
sign := -1;
for i := 1 to n do
begin Move(sign, 1); sign := -sign; Move(sign*2,i) end;
{ Pha 2 }
sign := -sign;
for i := n-1 downto 1 do
begin Move(sign, 1); Move(sign*2,i); sign := -sign end;

// Devcpp
// Pha 1
sign = -1;
for (i = 1; i <= n; ++i) {
    Move(sign, 1); sign = -sign; Move(sign*2,i);
}
// Pha 2
sign = -sign;
for (i = n-1; i > 0; --i){
    Move(sign, 1); Move(sign*2,i); sign = -sign;
}
```

### Chương trình

```
(* balls.pas *)
uses crt;
const fn = 'balls.inp'; gn = 'balls.out'; nl = #13#10;
var n, v: integer;
    d: longint;
(* n - so vien bi xanh = so vien bi do
   v - vi tri o trong
   d - tong so lan dich chuyen
*)
    a: string;
    f,g: text;
procedure Init;
    var i: integer;
begin
    a := '';
    for i := 1 to n do a := a + 'b';
    a := a + 'o';
    for i := 1 to n do a := a + 'r';
    d := -1; v := n+1; { vi tri o trong o }
end;
procedure PP; begin writeln(g,a); inc(d) end; { Ghi file }
procedure Swap(i,j: integer);
    var c: char;
begin c := a[i]; a[i] := a[j]; a[j] := c; PP end;
(* chuyen bi
h > 0: qua phai h o
h < 0: qua trai h o
k: so lan
```

```

*)
procedure Move(h,k: integer);
  var i: integer;
begin
  for i := 1 to k do
  begin
    Swap(v, v+h) ;
    v := v + h;
  end;
end;
procedure Balls;
  var i, sign: integer;
begin
  assign(f,fn); reset(f); readln(f,n); close(f);
  assign(g,gn); rewrite(g);
  Init;
  writeln(n);
  PP;
  sign := -1;
  for i := 1 to n do
  begin Move(sign, 1); sign := -sign; Move(sign*2,i) end;
  sign := -sign;
  for i := n-1 downto 1 do
  begin Move(sign, 1); Move(sign*2,i); sign := -sign end;
  Move(-1,1);
  writeln(g,d); close(g);
end;
BEGIN
  Balls;
  writeln(nl,' Total ',d, ' move(s)',nl);
  write(nl,' Fini');
  readln;
END.

// DEV-C++: balls.cpp
#include <string.h>
#include <fstream>
#include <iostream>
using namespace std;
// DATA AND VARIABLE
const int mn = 202;
char a[mn];
int n, v, n2, d;
/* n - so vien bi xanh = so vien bi do
   v - vi tri o trong
   n2 = 2n+1 - tong so o
   d - tong so lan dich chuyen
*/
// PROTOTYPES
void Init();
void PP();
void Run();
void Balls();
void Swap(int, int);
void Move(int, int);
ofstream f("BALLS.OUT");
// IMPLEMENTATION
int main(){
  Balls();
  f << d; f.close();
  cout << endl << " Total " << d << " move(s)" << endl;
  cout << endl << " Fini";
  system("PAUSE");
}

```

```

        return EXIT_SUCCESS;
    }
    void Init() {
        int i;
        for (i = 1; i <= n; ++i) a[i] = 'b'; // blue
        a[n+1] = 'o';
        n2 = n+n+1;
        for (i = n+2; i <= n2; ++i) a[i] = 'r'; // red
        d = -1; v = n+1; // vi tri o trong o
    }
    void PP() { // Ghi file
        for (int i = 1; i <= n2; ++i) f << a[i];
        f << endl;
        ++d;
    }
    void Swap(int i, int j) {
        char c = a[i]; a[i] = a[j]; a[j] = c;
        PP();
    }
    /* chuyen bi
    h > 0: qua phai h o; h < 0: qua trai h o; k: so lan
    */
    void Move(int h, int k) {
        for (int i = 0; i < k; v += h, ++i) Swap(v, v+h);
    }
    void Balls() {
        ifstream inf("BALLS.INP");
        inf >> n; inf.close();
        Init();
        cout << n ; PP();
        cout << endl;
        int i, sign = -1;
        for (i = 1; i <= n; ++i) {
            Move(sign, 1); sign = -sign; Move(sign*2,i);
        }
        sign = -sign;
        for (i = n-1; i > 0; --i){
            Move(sign, 1); Move(sign*2,i); sign = -sign;
        }
        Move(-1,1);
    }
}

```

### Độ phức tạp

Tại pha 1 ta chuyển ô trống 1 vị trí n lần và 2 vị trí  $(1 + 2 + \dots + n)$  lần, tổng cộng  $n + n(n+1)/2$  lần.

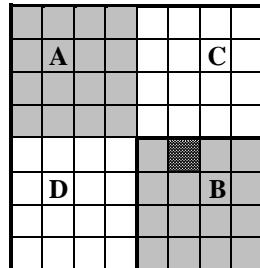
Tại pha 2 ta chuyển tương tự như trên nhưng với  $n-1$  lần: Ta có tổng cộng  $(n-1) + n(n-1)/2$ .

Lần cuối cùng ta chuyển 1 lần ô trống qua trái.

Vậy tổng cộng số lần chuyển là:  $t = n + n(n+1)/2 + (n-1) + n(n-1)/2 + 1 = n(n+2)$ . Với  $n = 3$  ta cần  $3.5 = 15$  lần thay vì 27 lần như thuật toán thứ nhất.

### 4.7 Lát nền 2

Người ta cần lát kín một nền nhà hình vuông cạnh dài  $n = 2^t$ , ( $t$  là một số tự nhiên trong khoảng 1..6) khuyết một ô thoát nước tại vị trí  $(x,y)$  bằng những viên gạch màu hình thước thợ (chữ L) tạo bởi 3 ô vuông đơn vị như trong hình 4.7.1(b). 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.



a) Nền nhà

Nền nhà kích thước  $8 \times 8$   
( $t = 3$ ). Lỗ thoát nước tại  
dòng 5 cột 6



b) Gạch lát

Hình 4.7.1

Dữ liệu vào: tệp văn bản **SQUARE.INP**:

Dòng đầu tiên: số tự nhiên  $n$ ;

Dòng thứ hai: hai số tự nhiên  $x$   $y$  cách nhau qua dấu cách, trong đó  $x$  là tọa độ dòng,  $y$  là tọa độ cột của lỗ thoát nước.

Nền nhà kích thước  $n$  được mã số dòng 1, 2, ...,  $n$  tính từ trên xuống và mã số cột 1, 2, ...,  $n$  tính từ trái qua phải.

Dữ liệu ra: tệp văn bản **SQUARE.OUT**:

Hai dòng đầu tiên ghi lại các dữ liệu của input file;

Dòng thứ ba: số màu cần dùng cho việc lát nền.

Tiếp đến là một phuong 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. Ô thoát nước được ghi số 0.

Thí dụ, với  $n = 8$  và ô thoát nước tại vị trí  $x = 5$ ,  $y = 6$  ta có một cách lát nền như hình vẽ.

### Thuật toán

1	1	3	3	1	1	3	3
1	2	2	3	1	2	2	3
3	2	1	1	3	3	2	1
3	3	1	2	2	3	1	1
1	1	3	2	1	0	3	3
1	2	3	3	1	1	2	3
3	2	2	1	3	2	2	1
3	3	1	1	3	3	1	1

Hình 4.7.2 Nền nhà với  $n = 8$   
( $t = 3$ )

Về số màu, với  $n = 2$  thì chỉ cần 1 viên gạch màu 1. Với mọi  $n$  lớn hơn 2 ta sẽ trình bày một thuật toán cần tối đa ba màu. Ta sẽ giải bài toán qua hai pha. Trước hết ta lát nền nhà để chừa một ô trống tại góc cuối cùng  $(n,n)$  của nền nhà. Sau đó ta sẽ tìm cách dịch chuyển ô trống này đến vị trí  $(x,y)$  cần thiết.

NEN . INP	NEN . OUT
8	8
5 6	5 6
	3
	1 1 3 3 1 1 3 3
	1 2 2 3 1 2 2 3
	3 2 1 1 3 3 2 1
	3 3 1 2 2 3 1 1
	1 1 3 2 1 0 3 3
	1 2 3 3 1 1 2 3
	3 2 2 1 3 2 2 1
	3 3 1 1 3 3 1 1

Phản lát nền đã trình bày chi tiết ở tập 1. Thủ tục này có tên là **Fill** và hoạt động như sau. Đầu tiên ta khởi trị với hình vuông cạnh  $k = 2$  nằm ở góc trên trái của nền nhà được biểu diễn dưới dạng một mảng hai chiều a: ba ô trong hình vuông  $2 \times 2$  sẽ được điền giá trị 1, ô nằm ở góc dưới phải được điền giá trị 2 (phản tô đậm). Như vậy, sau khi khởi trị ta coi như đã lát xong nền nhà cạnh  $n = 2$  bằng 1 viên gạch màu 1, lỗ thoát nước nằm ở góc dưới-phải ( $(1,1)$ ). 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 xuống theo đường chéo để thu được hình B (xem thủ tục **Copy**).
- Lật A sang phải (tức là lấy đối xứng gương qua trực tung) để thu được hình C (xem thủ tục **Right**).
- Lật A xuống dưới (tức là lấy đối xứng gương qua trực hoành) để thu được hình D (xem thủ tục **Down**).

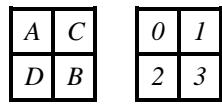
Chú ý rằng khi lật ta cần thực hiện thêm phép hoán đổi trị 1 và 3 cho nhau.

Hình 4.7.3

Mỗi lần lặp như vậy ta sẽ thu được hình vuông có cạnh tăng gấp đôi hình trước. Khi  $k = n$  ta thu được nền nhà được lát bằng các viên gạch chữ L với tối đa 3 màu 1, 2 và 3 cho trường hợp  $n > 2$ . Riêng ô (n,n) mang giá trị 2 sẽ được sửa thành 0.

Bây giờ ta chuyển qua pha thứ hai: Dịch chuyển ô trống tại (n,n) đến vị trí (x,y). Ta sẽ sử dụng thao tác cơ bản sau đây: dịch chuyển ô (d,c) tại góc một hình vuông cạnh k tới tâm của hình vuông cụ thể là tới một trong 4 ô nằm tại tâm của hình vuông này.

*Ô trống (8,8) được dịch chuyển đến tâm, tới ô (4,4). Hướng dịch chuyển  $dx = -1$  (theo dòng) và  $dy = -1$  (theo cột).*



1	1	3	3	1	1	3	3
1	2	2	3	1	2	2	3
3	2	1	1	3	3	2	1
3	3	1	0	2	3	1	1
1	1	3	2	1	1	3	3
1	2	3	3	1	2	2	3
3	2	2	1	3	2	1	1
3	3	1	1	3	3	1	0

1	1	3	3	1	1	3	3
1	2	2	3	1	2	2	3
3	2	1	1	3	3	2	1
3	3	1	0	2	3	1	1
1	1	3	2	2	1	3	3
1	2	3	3	1	1	2	3
3	2	2	1	3	2	2	1
3	3	1	1	3	3	1	1

Hình 4.7.4

Thủ tục này có tên là **ToCenter(k)**. Độ dịch chuyển theo dòng và cột phụ thuộc vào hướng dịch chuyển. Ta gọi  $dx$  là độ dịch chuyển (mỗi bước 1 ô) theo dòng và  $dy$  là độ dịch chuyển theo cột. Khi cần dịch ô trống về tâm theo hướng B → A ta đặt  $dx = dy = -1$ ; theo hướng D → C ta đặt  $dx = -1$ ,  $dy = 1$ ; theo hướng C → D ta đặt  $dx = 1$ ,  $dy = -1$ .

Muốn đưa ô trống (d,c) về vị trí (x,y) trước hết ta xác định xem hai ô này rơi vào mảnh nào trong các mảnh phân tư của hình vuông cạnh k. Ta dùng 2 bit để biểu diễn mệnh đề số hiệu dòng và cột lớn hơn hay nhỏ hơn  $k/2$ . Như vậy giá trị 01 ứng với mệnh đề: tọa độ dòng của ô (x,y) đang xét  $x \leq k/2$  và tọa độ cột của ô đó  $y > k/2$ . Vậy ô (x,y) đang xét nằm trong mảnh phân tư 1. Theo cách mã hóa nhị phân này, mỗi mảnh sẽ được mã số như sau: A = 0 = 00<sub>2</sub>, B = 3 = 11<sub>2</sub>, C = 1 = 01<sub>2</sub> và D = 2 = 10<sub>2</sub>. Sau khi đưa được ô (c,d) về tâm ta còn phải thực hiện một bước nhỏ nữa là chuyển tiếp ô này trong phạm vi 4 ô ở tâm để ô (c,d) rơi vào cùng mảnh với ô (x,y). Thủ tục này có tên là **Equalize**.

Với mỗi hình vuông cạnh k ta chia làm 4 phần A, B, C và D rồi gọi thủ tục **Equalize** để đưa hai ô (c,d) và (x,y) về cùng một mảnh phân tư. Sau một số lần lặp ta thu được  $k = 2$ . Khi đó trong hình vuông 4 ô chứa hai ô (c,d) và (x,y), trong đó (c,d) là ô trống, 3 ô còn lại cùng màu, ta chỉ làm phép đổi chỗ hai ô (c,d) và (x,y) là thu được kết quả.

### Độ phức tạp

Thủ tục lát nền duyệt mỗi ô 1 lần nên đòi hỏi  $n^2$  phép gán.

Giả sử  $n = 2^t$ . Thủ tục chuyển ô (c,d) tại góc dưới phải của nền nhà, tức là từ vị trí (n,n) về vị trí lỗ thoát nước (x,y) đòi hỏi t lần lặp, mỗi lần lặp ta phải dịch chuyển tối đa  $v/2$  lần, trong đó  $v$  là chiều dài cạnh của mảnh nền nhà hình vuông đang xét. Mỗi lần dịch chuyển ta đổi chỗ 2 ô, do đó cần 3 phép gán. Tổng cộng thủ tục này đòi hỏi cỡ  $3t(n/2+n/2^2+\dots+n/2^t) = 3tn(1/2+1/2^2+\dots+1/2^t) = 3tn(1-1/2^{t+1})/(1-1/2) = 6tn(1/2^{t+1})$ . Vì  $n = 2^t$  nên đại lượng trên được rút gọn thành  $6tn(1/2n) = 3t$  với  $t = \log_2(n)$ .

Tổng hợp lại, độ phức tạp của bài toán vào cỡ  $n^2$ .

### Chương trình

```
(* square.pas *)
```

```

uses crt;
const fn = 'square.inp'; gn = 'square.out';
      bl = ' '; nl = #13#10; mn = 101;
type mil = array[0..mn] of integer;
      mi2 = array[0..mn] of mil;
var
  a: mi2;
  n: integer; { chieu dai nen nha }
  x,y : integer; { Vi tri lo thoat nuoc }
  colors: integer; { so mau gach lat }
  sx, sy: integer; { goc tren trai cua manh dang xet }
  d, c: integer; { dong (d) va cot (c) chua o trong }
  qdc, qxy: integer;
{ qdc: manh chua o trong d c; qxy: manh chua o x y }
  dx, dy: integer; { huong dich chuyen o trong }
procedure ReadData;
var f: text;
begin
  assign(f,fn); reset(f);
  readln(f,n,x,y);
  close(f);
  writeln(nl, n, bl, x, bl, y);
end;
procedure Down(k: integer); { Lát xuống }
var i, j, ii, k2: integer;
begin
  ii := k; k2 := 2*k;
  for i := k+1 to k2 do
  begin
    for j := 1 to k2 do
      a[i][j] := 4-a[ii][j];
    dec(ii);
  end;
end;
procedure Right(k: integer); { Lật phải }
var i, j, jj, k2: integer;
begin
  jj := k; k2 := 2*k;
  for j := k+1 to k2 do
  begin
    for i := 1 to k2 do
      a[i][j] := 4-a[i][jj];
    dec(jj);
  end;
end;
procedure Copy(k: integer); { Tính tiến theo đường chéo }
var i, j: integer;
begin
  for i := 1 to k do
    for j := 1 to k do
      a[i+k][j+k] := a[i][j];
end;
procedure Fill; { Lát nền n×n }
var k: integer;
begin
  a[1][1] := 1; a[1][2] := 1;
  a[2][1] := 1; a[2][2] := 2;
  k := 2;
  while (k > n) do
  begin
    Down(k); Right(k); Copy(k);
    k := k*2;
  end;
end;

```

```

    a[n][n] := 0;
end;
procedure ToCenter(k: integer); { Dịch ô trống (c,d) về tâm }
  var nd, nc, i: integer;
begin
  nd := d + sx; nc := c + sy; k := k div 2;
  for i := 1 to k do
    begin
      a[nd][nc] := a[nd+dx][nc+dy];
      nd := nd + dx; nc := nc + dy;
      a[nd][nc] := 0;
    end;
  d := d + k*dx; c := c + k*dy; { Chính lại tọa độ (c,d) }
end;
{ Manh chua (x,y) trong hinh [1..n,1..n].
  0 1
  2 3 }
function Quater(n, x, y: integer): integer;
  var q, n2: integer;
begin
  q := 0; n := n div 2;
  if (x > n) then q := q + 2;
  if (y > n) then q := q + 1;
  Quater := q;
end;
procedure NewPos(n: integer); { tọa độ mới của (x,y) và (c,d) }
begin
  if (x > n) then x := x - n;
  if (y > n) then y := y - n;
  if (d > n) then d := d - n;
  if (c > n) then c := c - n;
  case qxy of
    1: sy := sx + n;
    2: sx := sy + n;
    3: begin sx := sx + n; sy := sy + n; end;
  end;
end;
{ doi cho a[x][y] va a[u][v] }
procedure ISwap(x, y, u, v: integer);
  var t: integer;
begin
  t := a[sx+x][sy+y]; a[sx+x][sy+y] := a[sx+u][sy+v];
  a[sx+u][sy+v] := t;
end;
procedure Equalize(k: integer); { Đưa (c,d) về cùng mảng với (x,y) }
  var k2: integer;
begin
  k2 := k div 2;
  if d = k then dx := -1 else dx := 1;
  if c = k then dy := -1 else dy := 1;
  ToCenter(k); { d, c den vi tri moi; a[d][c] = 0 }
  case qdc of
    0: if (qxy = 1) then
      begin
        ISwap(d,c,k2,k2+1);
        d := k2; c := k2+1;
      end
    else if (qxy = 2) then
      begin
        ISwap(d,c,k2+1,k2);
        d := k2+1; c := k2;
      end;
    1: if (qxy = 0) then
      begin

```

```

        ISwap(d,c,k2,k2);
        d := k2; c := k2;
    end else if (qxy = 3) then
        begin
            ISwap(d,c,k2+1,k2+1);
            d := k2+1; c := d;
        end;
    2: if (qxy = 0) then
        begin
            ISwap(d,c,k2,k2);
            d := k2; c := d;
        end else if (qxy = 3) then
            begin
                ISwap(d,c,k2+1,k2+1);
                d := k2+1; c := d;
            end;
    3: if (qxy = 1) then
        begin
            ISwap(d,c,k2,k2+1);
            d := k2; c := k2+1;
        end else if (qxy = 2) then
            begin
                ISwap(d,c,k2+1,k2);
                d := k2+1; c := k2;
            end;
        end { case };
        qdc := qxy;
    end;
procedure Move;
    var k: integer;
begin
    k := n; d := n; c := n;
    sx := 0; sy := 0;
    while (k > 2) do
    begin
        if (x = d) and (y = c) then exit;
        qdc := Quater(k,d,c); { manh chua (d,c) }
        qxy := Quater(k,x,y); { manh chua (x,y) }
        if (qdc <> qxy) then
            { chinh dong d va cot c cho cung manh voi (x,y) }
            Equalize(k);
            k := k div 2; NewPos(k);
    end;
    { k = 2. Final }
    ISwap(d,c,x,y);
end;
procedure WriteResult;
    var i, j: integer;
    g: text;
begin
    ReadData;
    assign(g,gn); rewrite(g);
    writeln(g,n); writeln(g, x, bl, y); writeln(g, colors);
    for i := 1 to n do
    begin
        writeln(g);
        for j := 1 to n do write(g,a[i][j],bl);
    end;
    close(g);
end;
procedure Run;
begin
    ReadData;

```

```

Fill; Move;
if (n = 2) then colors := 1 else colors := 3;
WriteResult;
end;
BEGIN
  Run;
  writeln(nl,' Fini');
  readln;
END.

// DevC++: square.cpp
#include <string.h>
#include <fstream>
#include <iostream>
using namespace std;
// D A T A   A N D   V A R I A B L E
const int mn = 101; // kich thuoc toi da cua nen nha
const char * fn = "square.inp";
const char * gn = "square.out";
const char bl = ' ';
int a[mn][mn]; // nen nha
int n; // chieu dai nen nha
int x, y; // vi tri lo thoat nuoc;
int colors; // so mau
int sx,sy; // goc tren trai cua manh dang xet
int d,c; // dong (d) va cot (c) chua o trong
int qdc, qxy; // qdc: manh chua o trong d c; qxy: manh chua o x y
int dx, dy; // huong dich chuyen o trong
// P R O T O T Y P E S
void ReadData(); // doc du lieu: n, x, y
void Fill(); // lat nen nha
void Move(); // di chuyen o trong tu vi tri (d,c) den vi tri (x,y)
void WriteResult(); // Ghi file
void Down(int); // Lat len
void Right(int); // Lat phai
void Copy(int); // dich cheo
void Run();
void ToCenter(int); // di chuyen o trong den giua manh
int Quater(int n, int x, int y); // manh phan tu chua o (x,y) trong
manh nn
void NewPos(int); // toa do moi
void Equalize(int); // can bang 2 manh
void ISwap(int, int, int, int);
// I M P L E M E N T A T I O N
int main() {
  Run();
  cout << endl << " Fini" << endl;
  cin.get();
  return 0;
}
void ReadData() {
  ifstream f(fn);
  f >> n >> x >> y;
  f.close();
}
void Down(int k) { // Lật xuồng
  int i,ii = k, j, k2 = k*2;
  for (i = k+1; i <= k2; ++i,--ii)
    for (j = 1; j <= k; ++j)
      a[i][j] = 4-a[ii][j];
}
void Right(int k) { // Lật phải
  int i, j, jj = k, k2 = 2*k;

```

```

        for (j = k+1; j <= k2; ++j,--jj)
            for (i = 1; i <= k; ++i)
                a[i][j] = 4 - a[i][jj];
    }
    void Copy(int k) { // Dịch chéo
        int i, j ;
        for (i = 1; i <= k; ++i)
            for (j = 1; j <= k; ++j)
                a[i+k][j+k] = a[i][j];
    }
    void Fill() { // Lát nền n×n
        int k;
        a[1][1] = a[1][2] = a[2][1] = 1;
        a[2][2] = 2;
        for (k = 2; k < n ; k *= 2) {
            Copy(k); Right(k); Down(k);
        }
        a[n][n] = 0;
    }
    void ToCenter(int k) { // đưa ô (c,d) về tâm
        int nd = d+sx, nc = c+sy;
        k = k/2;
        for (int i = 1; i <= k; ++i) {
            a[nd][nc] = a[nd+dx][nc+dy];
            nd += dx; nc += dy;
            a[nd][nc] = 0;
        }
        d += k*dx; c += k*dy; // tọa độ mới của (c,d)
    }
    // Manh chua (x,y) trong hinh [1..n,1..n]
    // 0 1
    // 2 3
    int Quater(int n, int x, int y) {
        int q = 0;
        n /= 2;
        if (x > n) q += 2;
        if (y > n) ++q;
        return q;
    }
    void NewPos(int n) { // Cập nhật tọa độ (x,y) và (c,d)
        if (x > n) x -= n;
        if (y > n) y -= n;
        if (d > n) d -= n;
        if (c > n) c -= n;
        switch(qxy) {
            case 1: sy += n; break;
            case 2: sx += n; break;
            case 3: sx += n; sy += n; break;
        }
    }
    // doi cho a[x][y] va a[u][v]
    void ISwap(int x, int y, int u, int v) {
        int t = a[sx+x][sy+y]; a[sx+x][sy+y] = a[sx+u][sy+v];
        a[sx+u][sy+v] = t;
    }
    void Equalize(int k) { // di chuyen o trong (d,c)
        // den manh chua qxy
        int k2 = k/2;
        dx = (d == k) ? -1 : 1;
        dy = (c == k) ? -1 : 1;
        ToCenter(k); // d, c den vi tri moi; a[d][c] = 0
        switch(qdc) {
            case 0: if (qxy==1) {

```

```

        ISwap(d,c,k2,k2+1);
        d = k2; c = k2+1;
    }
    else if (qxy==2) {
        ISwap(d,c,k2+1,k2);
        d = k2+1; c = k2;
    };
    break;
case 1: if (qxy==0) {
    ISwap(d,c,k2,k2);
    d = c = k2;
}
else if (qxy==3) {
    ISwap(d,c,k2+1,k2+1);
    d = c = k2+1;
};
break;
case 2: if (qxy==0) {
    ISwap(d,c,k2,k2);
    d = c = k2;
}
else if (qxy==3) {
    ISwap(d,c,k2+1,k2+1);
    d = c = k2+1;
};
break;
case 3: if (qxy==1) {
    ISwap(d,c,k2,k2+1);
    d = k2; c = k2+1;
}
else if (qxy==2) {
    ISwap(d,c,k2+1,k2);
    d = k2+1; c = k2;
};
break;
}
qdc = qxy;
}
void Move() {
    int k;
    k = d = c = n;
    sx = sy = 0;
    while (k > 2) {
        if (x==d && y == c) return;
        qdc = Quater(k,d,c); // manh chua (d,c)
        qxy = Quater(k,x,y); // manh chua (x,y)
        if (qdc != qxy)
            // chinh dong d va cot c cho cung manh voi (x,y)
            Equalize(k);
        k /= 2; NewPos(k);
    }
    // k = 2. Final
    ISwap(d,c,x,y);
}
void WriteResult() {
    ReadData();
    ofstream g(gn);
    g << n ;
    g << endl << x << bl << y;
    g << endl << colors;
    int i,j;
    for (i = 1; i <= n; ++i) {
        g << endl;

```

```

        for (j = 1; j <= n; ++j)
            g << a[i][j] << bl;
    }
    g.close();
}
void Run() {
    ReadData();
    Fill();
    Move();
    colors = (n == 2) ? 1 : 3;
    WriteResult();
}

```

#### 4.8 Test

Bạn hãy giúp Ban Giám khảo cuộc thi viết một chương trình kiểm tra bài giải Lát nền 2 nói trên. Dữ liệu vào chính là output file square.out. Dữ liệu ra ghi trong output file test.out các thông tin sau đây:

- 0 – nếu kết quả đúng;
- 1 – nếu đặt sai lỗ thoát nước;
- 2 – nếu đặt gạch sai;
- 3 – nếu số màu công bố khác với số màu thực lát.

#### Thuật toán

Trước hết mở file square.out đọc thông tin bao gồm các giá trị: n – chiều dài cạnh của nền nhà; (x,y) – vị trí (dòng, cột) của ô trống; colors – số màu đã dùng. Tiếp đến đọc dữ liệu kết quả về nền nhà đã lát vào mảng hai chiều a rồi chuyển qua pha kiểm lỗi.

– Nếu  $a[x][y] \neq 0$  ta ghi nhận lỗi 1: đặt sai lỗ thoát nước;

Với mỗi ô (i,j) trên nền nhà ta gọi thủ tục **Loang(i, j, c, d)**, trong đó c là màu của ô (i,j), c =  $a[i][j]$ . Thủ tục này đánh dấu và đếm số ô cùng màu c và liên thông cạnh với ô (i,j). Gọi số ô đếm được là s. Ta xét:

– Nếu  $s > 3$  tức là có hơn hai viên gạch chữ L cùng màu và kề cạnh nhau, vì mỗi viên gạch chữ L chỉ được phép chiếm tối đa 3 ô cùng màu. Trường hợp này ta ghi lỗi 2: đặt gạch sai.

– Nếu  $s = 3$  nhưng 3 ô cùng màu đó nằm thẳng hàng thì báo lỗi 2: đặt gạch sai.

Thủ tục Loang còn đảm nhiệm thêm chức năng tích lũy vào biến d số màu gạch lát khác nhau đã dùng. Nếu  $d \neq \text{colors}$  ta ghi nhận lỗi 3: số màu thực dùng khác với số màu đã công bố.

Khi loang ta đánh dấu ô đã xét bằng giá trị đối của nó.

```

(* tsquare.pas *)
uses crt;
const mn = 100; bl = #32; nl = #13#10;
      gn = 'square.out'; hn = 'test.out';
type mi1 = array[0..mn] of integer;
      mi2 = array[0..mn] of mi1;
var
  a: mi2;
  n, x, y, colors: integer;
procedure Loang(i, j, c: integer; var d: integer);
begin
  if (a[i][j] <> c) then exit;
  a[i][j] := -a[i][j]; inc(d);
  Loang(i+1,j,c,d);
  Loang(i-1,j,c,d);
  Loang(i,j+1,c,d);
  Loang(i,j-1,c,d);
end;
(*
0: khong co loi
1: Dat sai lo thoat
2: Dat sai gach
3: sai so mau
*)
procedure Test;
  var i, j, c, dc, d: integer;
      g,h: text;

```

```

        col: mil; { danh dau so mau da dung }
begin
    assign(g,gn); reset(g);
    read(g,n,x,y,colors);
    write(nl,' n = ', n, ' x = ', x, ' y = ', y);
    fillchar(a,sizeof(a),0); fillchar(col,sizeof(col),0);
    for i := 1 to n do
begin
    writeln;
    for j := 1 to n do
begin
    read(g,a[i][j]); write(a[i][j],bl);
end;
end;
close(g);
writeln;
assign(h,hn); rewrite(h); dc := 0;
if (a[x][y] <> 0) then
begin writeln(h,1); close(h); exit; end;
for i := 1 to n do
    for j := 1 to n do
begin
    c := a[i][j];
    if (c > 0) then
begin
    if (col[c] = 0) then
begin col[c] := 1; inc(dc) end;
    d := 0; Loang(i,j,c,d);
    if (d <> 3) then
begin writeln(h,2); close(h); exit end;
    { d = 3: xet 3 phan tu lien tiep }
    if ( (a[i][j]=a[i][j+1])and(a[i][j]=a[i][j+2]) )
        or ( (a[i][j]=a[i+1][j])and(a[i][j]=a[i+2][j]) )
    then begin writeln(h,2); close(h); exit end;
end;
end ; { for }
    if (d <> colors) then begin writeln(h,3); close(h); exit end;
writeln(h,0); close(h);
end;
BEGIN
Test;
writeln(nl,' Fini');
readln;
END.
```

```

// DevC++: tsquare.cpp
#include <string.h>
#include <fstream>
#include <iostream>
using namespace std;
// D A T A   A N D   V A R I A B L E
const int mn = 101;
const char * gn = "square.out";
const char * hn = "test.out";
const char bl = ' ';
int a[mn][mn];
int n, x, y, colors;
// P R O T O T Y P E S
void Test();
void Loang(int, int, int, int& );
// I M P L E M E N T A T I O N
int main() {
    Test();
```

```

cout << endl << " Fini" << endl;
cin.get();
return 0;
}
/*
0: khong co loi
1: Dat sai lo thoat
2: Dat sai gach
3: sai so mau
*/
void Test() {
    ifstream g(gn);
    g >> n >> x >> y >> colors;
    cout << endl << " n = " << n << ", x = " << x << ", y = " << y
        << ", colors = " << colors << endl;
    int i, j, c, dc, d; // dc dem so mau
    int col[100]; // danh dau mau da dung
    memset(a,0,sizeof(a)); memset(col,0,sizeof(col));
    for (i = 1; i <= n; ++i) {
        cout << endl;
        for (j = 1; j <= n; ++j) {
            g >> a[i][j]; cout << a[i][j] << bl;
        }
    }
    g.close();
    cout << endl;
    ofstream h(hn); dc = 0;
    if (a[x][y] != 0) { h << 1; h.close(); return; }
    for (i = 1; i <= n; ++i)
        for (j = 1; j <= n; ++j) {
            c = a[i][j];
            if (c > 0) {
                if (col[c] == 0) {col[c] = 1; ++dc;} // them mau moi
                d = 0; Loang(i,j,c,d);
                if (d != 3) { h << 2; h.close(); return; }
                // d = 3: xet 3 phan tu lien tiep
                if ( ( (a[i][j]==a[i][j+1])&&(a[i][j]==a[i][j+2]) )
                    || ( (a[i][j]==a[i+1][j])&&(a[i][j]==a[i+2][j]) ) )
                    { h << 2; h.close(); return; }
                } // if c > 0
            } // for j
            if (d != colors) { h << 3; h.close(); return; }
            h << 0;
            h.close();
        }
    void Loang(int i, int j, int c, int &d) {
        if (a[i][j] != c) return;
        a[i][j] = -a[i][j]; ++d;
        Loang(i+1,j,c,d);
        Loang(i-1,j,c,d);
        Loang(i,j+1,c,d);
        Loang(i,j-1,c,d);
    }
}

```

#### 4.9 Giải mã

Cho mã nhị phân của n chữ cái đầu bảng chữ tiếng Anh A, B, C,... Biết rằng không có mã nào là khúc đầu của mã khác và chiều dài tối đa của mỗi mã là 10. Lập chương trình giải mã một văn bản cho trước.

Input text file: **code.inp**

Dòng đầu tiên: số n;  $1 \leq n \leq 26$ .

Tiếp đến là n dòng, mỗi dòng chứa mã nhị phân của một chữ cái theo trật tự A, B, C,...

Cuối cùng là dòng chứa mã cần giải.

Output file: **code.out** chứa văn bản đã giải.

<b>code.inp</b>	<b>code.out</b>
<pre> 5 0000 0001 0010 0011 110 0000000100010000 </pre>	<b>ABBA</b>

Thí dụ trên cho mã của  $n = 5$  kí tự đầu trong bảng chữ cái tiếng Anh là **A = 0000**, **B = 0001**, **C = 0010**, **D = 0011** và **E = 110**. Đoạn mã văn bản cần giải là **s = 0000000100010000**. Sau khi giải mã ta phải thu được kết quả: **ABBA**.

### Thuật toán

Trước hết ta xây dựng cây nhị phân  $h$  với các nhánh trái ứng với giá trị 0 của mỗi mã, nhánh phải ứng với giá trị 1. Tại các lá của cây ta ghi kí tự tương ứng của mã đó. Như vậy, dãy nhãn trên mỗi đường đi từ gốc cây  $h$  đến lá của  $h$  sẽ lập thành mã của kí tự trên lá. Thí dụ, đường đi 0011 sẽ kết thúc tại lá D, do đó 0011 là mã nhị phân của D.

Sau đó dựa vào dãy bit 0/1 của mã văn bản s ta giải mã thông qua phép duyệt cây  $h$  như sau:

Duyệt cây $h$ để giải mã văn bản
1. Xuất phát từ gốc $h$ .
2. Duyệt lần lượt mỗi bit $s_i$ , xét 2 tình huống:
2.1 Nếu $s_i = 0$ : rẽ trái; Nếu $s_i = 1$ : rẽ phải.
2.2 Nếu gặp lá thì: - Ghi kí tự trên lá vào kết quả; - Quay lại gốc $h$ .
3. end.

Ta có thể cài đặt nhanh bằng cách sử dụng heap (chùm, đống) thay cho cây. Trong bài này ta hiểu heap  $h$  là một cây nhị phân đầy đủ (gọi là cây cân bằng hoàn toàn). Nếu gọi đỉnh đầu tiên của  $h$  là 1 thì hai đỉnh kề của nó sẽ nhận mã số là 2 và 3..., tổng quát, hai đỉnh kề của đỉnh  $i$  sẽ có mã số  $2i$  và  $2i+1$ , trong đó ta qui định nhánh  $i \rightarrow 2i$  sẽ là nhánh trái, do đó nhận nhãn 0; nhánh  $i \rightarrow 2i+1$  là nhánh phải, do đó nhận nhãn 1. Nếu độ dài tối đa của mã là  $k$  thì heap sẽ có  $2^{k+1}-1$  phần tử. Thật vậy, do heap có  $k$  nhánh tính từ gốc đến lá nên sẽ có  $k+1$  mức với số lượng đỉnh theo từng mức lần lượt là 1, 2, 4, ...,  $2^1, \dots, 2^k$ . Tổng số đỉnh của heap khi đó là

$$1 + 2 + 2^2 + \dots + 2^k = 2^{k+1} - 1$$

Áp dụng công thức trên cho độ dài max của mã  $k = 10$  ta tính được số phần tử của heap sẽ là  $2^{11}-1 = 2047$ . Như vậy ta có thể sử dụng một mảng  $h$  để chứa các đỉnh của cây. Thủ tục tạo cây trên heap khi đó sẽ khá đơn giản.

Tạo cây  $h$  như một cây con của heap

1. Khởi trị mọi phần tử của mảng h bằng 0 với ý nghĩa
  - $h[i] = 0$ : đỉnh i là đỉnh trung gian hoặc không thuộc cây;
  - $h[i] = C$ : đỉnh i là lá và đường đi từ gốc đến đỉnh i sẽ là mã của kí tự C.
2. Với mỗi mã  $(u_1, u_2, \dots, u_m)$  của kí tự C ta xét
  - 2.1 Gọi v là số hiệu của đỉnh trong cây h.  
Khởi trị  $v := 1$ ;
  - 2.2 Với mỗi bit  $u_i$  xét.  
Nếu  $u_i = 0$ : tính  $v := 2*v$ ;  
Nếu  $u_i = 1$ : tính  $v := 2*v + 1$ ;
  - 3.2 Gán  $h[v] := C$ .
3. end.

Và thủ tục giải mã trên heap h khi đó sẽ là:

Giải mã s trên heap h																	
. Xuất phát từ gốc h với $v := 1$ .																	
2. Với mỗi bit $s_i$ của mã văn bản s xét:																	
2.1 Nếu $s_i = 0$ : tính $v := 2*v$ ; Nếu $s_i = 1$ : tính $v := 2*v + 1$ .																	
2.2 Nếu $h[v] \neq 0$ tức là đã gặp lá thì: - Ghi kí tự trên lá vào kết quả; - Quay lại gốc h: đặt $v := 1$ .																	
3. end.																	

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	0	0	0	0	0	0	0	0	0	0	0	E	0	A	B	C	D

Heap h thu được sau khi đọc dữ liệu

### Độ phức tạp

Có n mã cho n kí tự, mỗi mã được duyệt 1 lần để tạo heap. Gọi chiều dài tối đa của mã là k. Vậy để tạo heap ta cần n.k thao tác. Để giải mã ta duyệt mỗi bit trong bản mã 1 lần trên cây nhị phân. Vậy khi giải mã ta cần  $m = \text{len}(s)$  thao tác trên cây nhị phân. Thao tác trên cây nhị phân v đỉnh cần  $\log_2(v) = k$  phép so sánh. Tổng hợp lại, tính theo chiều dài m của mã văn bản ta cần  $mk$  phép so sánh.

### Chương trình

```
(* code.pas *)
uses crt;
const fn = 'code.inp'; gn = 'code.out';
      mn = 2050; bl = #32; nl = #13#10;
var h: array[0..mn] of char;
procedure Decode;
var i, j, v, n: integer;
    ch: char;
    f,g: text;
    x: string;
begin
    ch := 'A';
    assign(f,fn); reset(f);
    assign(g,gn); rewrite(g);
    readln(f,n); writeln(n,bl,bl);
    fillchar(h,sizeof(h),0);
    {Tạo heap h }
    for i := 1 to n do
```

```

begin
  readln(f,x); writeln(x);
  v := 1;
  for j := 1 to length(x) do
    if x[j] = '0' then v := v*2 else v := v*2 + 1;
    h[v] := ch; inc(ch);
end;
writeln(nl);
for i := 1 to mn do
  if (h[i] <> #0) then writeln(i,bl,h[i]);
v := 1; { Giai ma }
while not eof(f) do
begin
  read(f,ch);
  if (ch = '0') or (ch = '1') then
begin
  v := 2*v;
  if ch = '1' then v := v+1;
  if (h[v] <> #0) then
  begin
    write(g,h[v]);
    v := 1;
  end;
  end;
end;
close(f); close(g);
end;
BEGIN
  Decode;
  writeln(nl, ' Fini ');
  readln;
END.

```

```

// DevC++: code.cpp
#include <iostream>
#include <fstream>
using namespace std;
const char * fn = "code.inp";
const char * gn = "code.out";
const int mn = 2050;char h[mn]; // heap
int main();
void Decode();
int main() {
  Decode();
  cout << endl << " Fini ";
  cin.get();
  return 0;
}
void Decode() {
  int i, j, v, n;
  char x[15]; // doan ma
  char ch = 'A';
  ifstream f(fn); f >> n; cout << endl << n;
  f.getline(x,mn,'\n');
  ofstream g(gn);
  memset(h,0,sizeof(h));
  cout << endl << endl;
  // Tao heap h
  for (i = 0; i < n; ++i) {
    f.getline(x,mn,'\n'); cout << endl << x;
    v = 1;
    for (j = 0; x[j]; ++j)

```