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

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

MỤC LỤC

Lời nói đầu *4*
Chương 1 Các thuật toán trên String *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**
Chương 2 Xử lí dãy lệnh và biểu thức *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**
Chương 3 Cặp ghép *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**
Chương 4 Các phép lật và chuyển vị *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**
Chương 5 Luyện tập từ các đề thi ***110***
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 vắn 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];
```

Cả hai khai báo trên là tương đương nhau và x, y đều có *dung lượng* hay sức chứa tới 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 thống 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 = 'abcdegh'`** thì **`s[2..5] = 'bcde'`** là một đoạn. Đoạn s[1..i] 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.

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'

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.

data.inp

```
Tinh tong cua 12 so sau day:
1 -2 3 -4 5 6
7 8 9 10 -11 -12
```

data.out

```
Tong cua 12 so:
 +1 -2 +3 -4 +5 +6 +7 +8 +9 +10 -11 -12 = 20
```

Yêu cầu: xác định giá trị của n và tính 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');
```

```
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
   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;
// D A T A  A N D  V A R I A B L E
const char * fn = "data.inp";
const char * gn = "data.out";
int n;
// P R O T O T Y P E S
void Tong();
bool LaChuSo(char c);
// I M P L E M E N T A T I O N
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; // khoi tri so n
    while (LaChuSo(s[i])) { // doc so  n
      n = n*10 + int(s[i]-'0');
      ++i;
    }
    t = 0; // khoi 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] ≠ y[j] thì s(i,j) = 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 sứ 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] ≠ y[j] thì b[j] = 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 = len(x), n = 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('xabcxxxd','aybcydyy')); { 4 }
  readln;
END.

// Dev-C++: XauChung.cpp
```

```
int Max(int a, int b) { rturn (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("xaxxbcxd","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

1. 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.

2. 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 = "xabcxxabcdxd", y = "aybcyabcdydy" 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], $k \to \max$. 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≤ i < j ≤ 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* = '*xababababayyy*' 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 → max. Ta có,

- Nếu x[i] = x[j] thì s(i,j) = s(i–1,j–1) + 1;
- Nếu x[i] ≠ x[j] thì s(i,j) = 0.

Đáp số sẽ là Max { s(i,j) | 1 ≤ i ≤ len(x), 1 ≤ j ≤ 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 = 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;
end;
BEGIN
  DoanLap('xabababayy',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

$$\text{Max } \{ \text{ ComLen(id[i], id[i-1]) } | \text{ i = 2..len(s) } \}$$

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 mi1 = 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: mi1; 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: mi1;
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 = 'xyabcdefghhikabcdefghhikzt'. Đáp số i = 3, j = 2 + 5 + 6 + 1 = 14, k = 5+6 = 11.

Test 2. s = XABABABY = 'xyabcdefghhikabcdefghhikabcdefghhikabcdefghhikzt'. 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 đôi một. 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 saintpavnamtranaisnotsaintabba	5	*Sau khi xóa 5 chữ cái (gạch dưới)* saintpavnamtranaisnotsaintabba *ta thu được dãy ghép của các từ 5, 6, 3, 2, 5, 1* saintpanamaisnotsaintabba

Thuật toán

Gỉa 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–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–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;
// D A T A  A N D  V A R I A B L E
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];
}
// Nhung 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. Cỡ 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 cải 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.

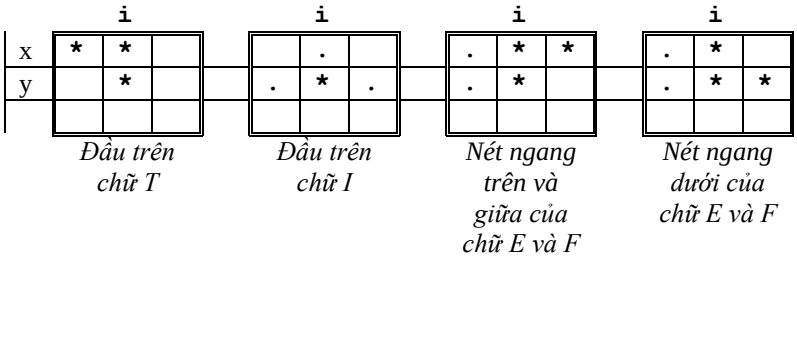
1.8 TEFI

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 vach 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.



dnt = số nét ngang trên.
dnd = số nét ngang dưới.
dx = 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 – dòng trên; y – dòng dưới.

Điều kiện	`x[i] = '*'`	1	0	1	1
	`x[i-1] = '*'`	1	–	0	0
	`x[i+1] = '*'`	–	–	1	–
	`y[i] = '*'`	1	1	1	1
	`y[i-1] = '*'`	–	0	0	0
	`y[i+1] = '*'`	–	0	–	1
Quyết định	T (chữ T)	1			
	Γ (nét ngang trên)			1	
	L (nét ngang dưới)				1
	I (chữ I)		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 Γ 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;
}
```

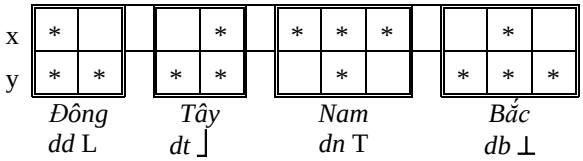
```
void II(int i) {
  if (y[i] == ss && y[i-1] == cc && y[i+1] == cc) ++di;
}
void TEFI() {
   int i, j;
  dt = di = dnt = dnd = 0;
  ifstream f(fn);
  f >> n >> m; f.get();
  x = cc;
  for (i = 0; i < 8; ++i) x = x + x;
  for (j = 0 ; j < n; ++j) {
    f >> y; y = cc + y + cc;
    for (i = 1; i <= 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 xiếc

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 xiếc. 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* ⊥ (E *Bắc). Mỗi E Đông có 2 nét dạng* L *và mỗi E Tây có 2 nét dạng* ⌋. *Ngoài ra, mỗi chữ E Bắc còn có 1 nét* L *và 1 nét* ⌋. *Ta có*

Số chữ E Đông = (dd – db)/2,

Số chữ E Tây = (dt–db)/2,

Số chữ E Nam = dn,

Số chữ E Bắc = db.

Độ phức tạp. cỡ 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;
```

```
     cout << endl << endl << " Fini "; // 3
     cin.get();
     return 0;
}
void DongTayNamBac(int i) {
  if (y[i+1] == ss) DongBac(i);
  else TayNam(i);
}
void DongBac(int i){
   if (y[i-1] == ss) ++db;
    else ++dd;
}
void TayNam(int i) {
  if (y[i-1] == ss) ++dt;
  else if (x[i-1] == ss && x[i+1] == ss) ++dn;
}
void EXIEC() {
   int i, j;
  dd = dt = dn = db = 0;
  ifstream f(fn);
  f >> n >> m; 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 && y[i] == ss) DongTayNamBac(i);
    x = y;
  }
  f.close();
  dd = (dd - db)/2; dt = (dt - db)/2;
}
```

Chương 2 Xử lí dãy lệnh và biểu thức

2.1 Val

*Cho các biến được gán trị a = 0, b = 1, c = 2,..., z = 25. Tính trị của biểu thức số học được viết đúng cú pháp, chứa các tên biến, các phép toán +, –, *, và / (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]) ≤ 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 đến 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 Lucasievics 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. cỡ n, trong đó n là số kí hiệu trong biểu thức.

```
(* Val.pas *)
```

```
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 LaPhepToan(c: char): Boolean;
begin LaPhepToan := (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; { nap phep toan p }
end;
procedure XuLiNgoac;
begin
 while (c[ic] <> '(') do begin Tinh(c[ic]); dec(ic) end;
 dec(ic); { Bo ngoac }
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 LaPhepToan(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 Dư lieu va bien
const int mn = 500;
char s[mn]; // bieu thuc
char c[mn]; //ngan xep phep toan va dau (
int ic; // con trỏ ngăn xếp c
int v[mn]; //ngan xep tinh toan
int iv; // con trỏ ngăn xếp v
int kq; // ket qua
int n; // len - số ki tự trong biểu thức
// Khai báo các hàm
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 phép 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 ABBBCCDCCDABBBCCDCCDAAA.

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 kí 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 ngăn 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

ABBBCCDABBBCCDABBBCCDABBBCCD

Cuối cùng, sau khi thực hiện dòng lệnh 7 ta thu được kết quả

ABBBCCDABBBCCDAAA

Độ phức tạp Cỡ n, trong đó n là số kí tự trong xâu input.

```
(* XauGon.pas *)
uses crt;
const mn = 500; BL = #32; NL = #13#10;
ChuSo = ['0'..'9']; ChuCai = ['A'..'Z'];
type mi1 = array[0..mn] of integer;
     mc1 = array[0..mn] of char;
var M, T, R, st: mi1; { 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 }
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;
  // D A T A  A N D  V A R I A B L E
  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
  // P R O T O T Y P E S
  void KhaiTrien(char *);
  void LenhDon(char);
  void LenhLap();
  int DocSo();
  void Cach();
  bool LaChuSo(char);
  bool LaChuCai(char);
  void Pha2();
  // I M P L E M E N T A T I O N
  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 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 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:

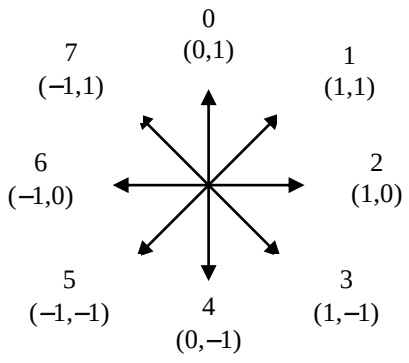
Gn – đi thẳng n bước qua các điểm nguyên,
Rn – quay phải n lần 45 độ,
Ln – quay trái n lần 45 độ.

Robot được đặt tại vị trí xuất phát là gốc 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) mod 8, còn khi quay trái k lần ta sẽ có h = (h + n−(k mod 8)) mod 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 *Cỡ 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 mi1 = array[0..mn] of integer;
     mc1 = array[0..mn] of char;
var M, T, R, st: mi1; { 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;
 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 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;
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;
// D A T A  A N D  V A R I A B L E
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, không có tham biến, qui ước = 0
ucln(x) = |x|, ucln của số x là giá trị tuyệt đối của chính số đó
ucln(a,b) = ucln(b,a),
ucln(a,0) = a,
ucln(a,b) = ucln(a mod b, b)
$ucln(x_1, x_2,...,x_n) = ucln\ (ucln(x_1, x_2,...,x_{n-1}), |x_n|)$, $n \geq 2$.

Thí dụ 2: Hàm sum – tính tổng của các số nguyên:

sum( ) = 0,
sum(x) = x,
$sum(x_1, x_2,...,x_n) = sum(sum(x_1, x_2,...,x_{n-1}), x_n)$, $n \geq 2$.

Ngoài ra còn các hàm lấy min, max của dãy phần tử...
*Cho một biểu thức được viết đúng cú pháp, chứa các hằng nguyên, các biến a, b,... được gán sẵn các trị a = 0, b = 1,..., 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 với phép toán trừ. Kỹ thuật này có thể gây nhập nhằng, thí dụ, khi xử lí biểu thức a–b thì dấu – gặp 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 li 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 phảy dùng để ngăn cách các tham biến. Ta cũng nạp dần các dấu ngăn 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 phảy: 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) end;
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;
// D A T A  A N D  V A R I A B L E
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
// P R O T O T Y P E S

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 dao dau
bool LaBien(char c) { return (c >= 'a') && (c <= 'z'); }
int Val(char x) { return int(v-'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

files.inp	files.out

Người ta cần xử lí một số file trong số các file có tên 1, 2, ..., 48 hiện

LOAD 3 **MODI** **SAVE 50** **LOAD 4** **MODI** **SAVE 51** **LOAD 50** **INS 51** **MODI** **SAVE 52** **LOAD 1** **INS 2** **MODI** **INS 52** **SAVE 49** **END**	**LOAD 4** **MODI** **SAVE 51** **LOAD 3** **MODI** **INS 51** **MODI** **SAVE 50** **LOAD 1** **INS 2** **MODI** **INS 50** **SAVE 49** **END**

lưu trên đĩa cứng để tạo ra một file kết quả có tên 49. Qui trình xử lí được lập trình sẵn. Chương trình được viết bằng ngôn ngữ quản lý file gồm các lệnh

LOAD i *: đọc file i vào miền nhớ RAM*

MODI*: sửa nội dung hiện có trên RAM*

INS j*: xen file j vào file trên RAM*

SAVE j: *ghi nội dung trên RAM vào file tên j.*

END*: kết thúc chương trình.*

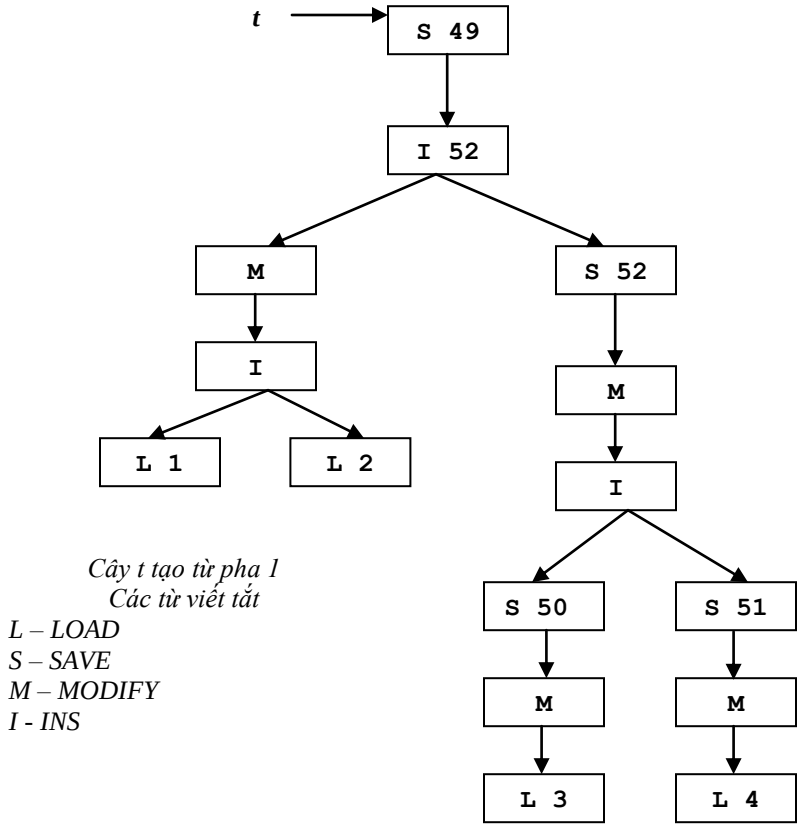
Trừ lệnh SAVE 49 cuối cùng, các lệnh SAVE đều được sử dụng để lưu tạm các kết quả trung gian với các tên file từ 50 trở đi. Với các file kích thước lớn, người ta muốn hạn chế số lần ghi đĩa bằng lệnh SAVE nhằm tiết kiệm thời gian xử lí. Hãy thay chương trình ghi trong files.inp bằng một chương trình tương đương ghi trong files.out với ít lệnh SAVE. Hai chương trình được xem là tương đương nếu file 49 chứa kết quả cuối cùng có nội dung giống nhau.

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, ..., 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
LOAD 3	**13**	**13**	**LOAD 4**	**14**	**14**
MODI		**31**	**MODI**		**41**
SAVE 50	**31**	**31**	**SAVE 51**	**41**	**41**
LOAD 4	**14**	**14**	**LOAD 3**	**13**	**13**
MODI		**41**	**MODI**		**31**
SAVE 51	**41**	**41**	**INS 51**	**41**	**3411**
LOAD 50	**31**	**31**	**MODI**		**1143**
INS 51	**41**	**3411**	**SAVE 50**	**1143**	**1143**
MODI		**1143**	**LOAD 1**	**11**	**11**
SAVE 52	**1143**	**1143**	**INS 2**	**12**	**1121**
LOAD 1	**11**	**11**	**MODI**		**1211**
INS 2	**12**	**1121**	**INS 50**	**1143**	**11143211**
MODI		**1211**	**SAVE 49**	**11143211**	**11143211**
INS 52	**1143**	**11143211**	**END**		
SAVE 49	**11143211**	**11143211**			
END					

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**] = [1**13**2], trong khi [**13**] INS [12] = [**1**12**3**], ngoài ra ([12] INS [13]) INS [14] = [1132] INS [14] = [114132], trong khi [12] INS ([13] INS [14]) = [12] INS [1143] = [111432].

Thuật toán



Cây t tạo từ pha 1
Các từ viết tắt
L – LOAD
S – SAVE
M – MODIFY
I - INS

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 **L**OAD, **S**AVE, **M**ODI, **E**ND, 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 **L**OAD, **S**AVE, **M**ODI, **I**NS, **E**ND 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 = 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 = NewElem('I',f,v,0), ngược lại, nếu file f đã xuất hiện thì tạo nút mới v = 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;
// D A T A  A N D  V A R I A B L E
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 cấy 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*: cấy ghép gen hiện có trên T với gen trong tủ chứa j, kết quả thu đượ 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 rất phức tạp nên người ta cần hạn chế đến mức tối đa các thao tác này. Ngoài ra, lưu ý rằng việc cấy ghép gen i vào gen j cho kết quả khác với việc cấy 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 ADD y SAVE 100 LOAD x SUB y SAVE 101 LOAD 100 MULT 101 SAVE z END	LOAD x SUB y SAVE 100 LOAD x ADD y MULT 100 SAVE z END

Trong các bộ xử lí một địa chỉ các thao tác xử lí 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) kí 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 trển 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) với h > 1, và
s(k,1) = 1, k = 1, 2, …, với k ≥ 1cặp 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, với 0 cặp ngoặc không thể xây dựng được biểu thức mức h > 0;
s(0,0) = 1, với 0 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:

$$b[1] = s(k,1) = 1$$

$$b[j] = s(k,j) = s(k–1,j–1) + 2s(k–1,j) = a[j–1] + 2a[j],\ j = 2..h$$

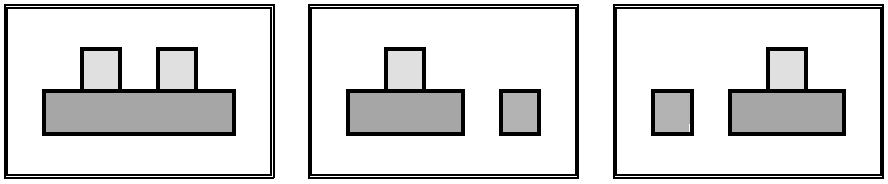
Độ phức tạp: k.h.

```
(* 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[1] = 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 Mi 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 2 2 3 3 6 9	3 3 3 2 1

Ý nghĩa: Cần 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 chỗ hụt còn đủ ta lại kéo từ i–1 từ dòng trên xuống, độ hụt khi đó sẽ là h = h–v[i–1],... 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 trỏ ngược t[1..n] trong đó t[i] = j cho ta biết các từ j, j+1,...,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.

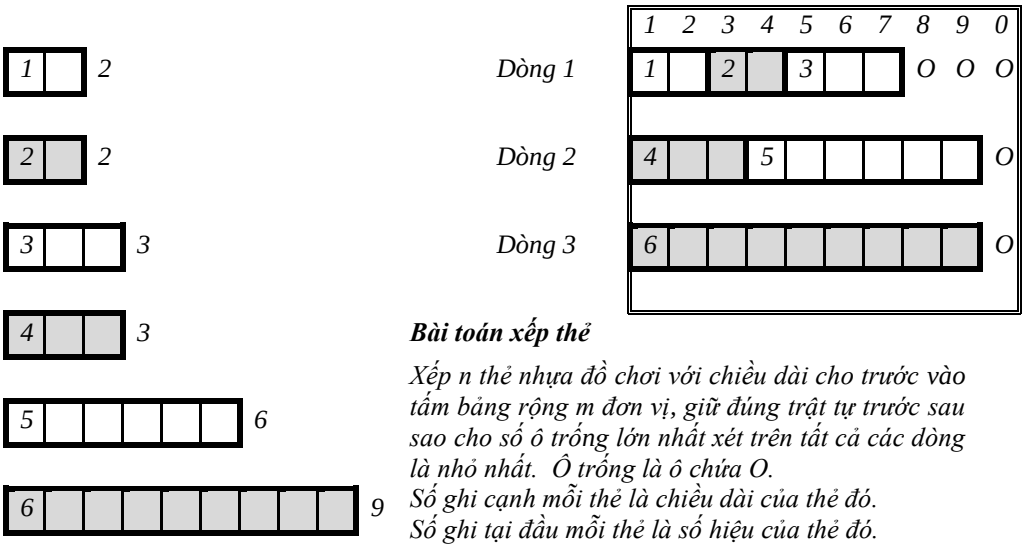
```
(* Pages.pas *)
uses crt;
const mn = 200; bl = #32; nl = #13#10;
      fn = 'pages.inp'; gn = 'pages.out';
type mi1 = array[0..mn] of integer;
var n,m,k,h: integer;
    f,g: text;
    v, d, t, sl: mi1;
    (* ---------------------------------------
      n - number of words; m - width of page;
      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: mi1; 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]);
```

```
  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

(Bài tương tự)



Bài toán xếp thẻ

Xếp n thẻ nhựa đồ chơi với chiều dài cho trước vào tấm bảng rộng m đơn vị, giữ đúng trật tự trước sau sao cho số ô trống lớn nhất xét trên tất cả các dòng là nhỏ nhất. Ô trống là ô chứa O.
Số ghi cạnh mỗi thẻ là chiều dài của thẻ đó.
Số ghi tại đầu mỗi thẻ là số hiệu của thẻ đó.

2.12 Xếp xe

(Bài tương tự)
Tại Hội thi Tin học trẻ có n đoàn học sinh đã xếp hàng đầy đủ tại địa điểm tập trung để chờ Ban tổ chức đưa xe ô tô đến chở các em đi tham quan thủ đô. Mỗi xe có m ghế dành cho mỗi em một ghế. Các đoàn được bố trí lên xe theo đúng trật tự từ đoàn số 1 đến đoàn số n. Mỗi xe có thể xếp vài đoàn liên tiếp nhau nhưng mỗi đoàn cần được xếp gọn trên 1 xe. Sau khi xếp đầy đủ các đoàn, người ta đếm số ghế trống trên mỗi xe và công bố số ghế trống lớn nhất h đã đếm được. Biết số học sinh trong mỗi đoàn, hãy tim một cách xếp xe để giá trị h là nhỏ nhất.

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→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 1−1 f từ tập nguồn *vào* tập đích, f: A → 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 rỗi (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 ∈ 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 ≠ 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 dang giữ quà mà bạn i thích, đặt quà đó vào một túi riêng bên ngoài túi có đề chữ i 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 dảm 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

$$t_1 \leftarrow t_2 \leftarrow \ldots t_{k-1} \leftarrow t_k$$

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 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,…, 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: trỏ tới ngăn 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 mi1 = 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: mi1;
    t: mi1;
    st: mi1; p: integer;
    c: mb2;
    f,g: text; {f: input file; g: otput 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 qùa q }
  a[i] := j; b[j] := i; { để nhận quà 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 qùa j }
      begin
        if b[j] = 0 then { qùa 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;
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[++p] = 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, ... , n×m theo trật tự từ dòng trên xuống dòng dưới, trên mỗi dòng tính từ trái qua phải. Người ta đặt v 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.

domino.inp	domino.out
4 5 8 2 3 4 5 6 7 9 10 10 15 7 12 19 20 13 18	10 1 2 3 4 5 10 6 11 7 8 9 14 12 13 15 20 16 17 18 19

Dữ liệu

Input file: Text file domino.inp.

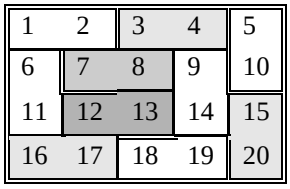
Dòng đầu tiê`n: 3 số n – số dòng, m – số cột, v – số vách ngăn.

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.



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 → 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 kề*. Thí dụ, 8 có 4 số kề 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 kề và ghi vào mảng ke với ý nghĩa như sau ke[i][j] = 1 khi và chỉ khi i có số kề tại vị trí j; j = 1, 2, 3, 4. Nếu i có vách ngăn phải thì i sẽ mất bạn kề phải. Tương tự, nếu i có vách ngăn dưới thì i sẽ mất bạn kề dưới. Ngoài ra, dễ hiểu rằng các số trên dòng 1 thì không có số kề trên, các số trên dòng n thì không có số kề dưới. Tương tự, các số trên cột 1 thì không có số kề trái, các số trên cột m thì không có số kề 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 kề 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  mi1 = 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: mi1; { stack } p: integer; { ngon st }
    a, t: mi1;
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 <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 = "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
// 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);
// I M P L E M E N T A T I O N
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 ngon 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 chở được 2 người có nhiệm vụ khảo cứu một đề 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 đên 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_1 c_2 ... c_k *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.*

Discover.inp	Discover.out
5 3 3 1 4 2 2 3 3 4 2 1 2 1 3 3 5 4 2 3 2 3 1 3 1 4 3 2 5 2 3 4 2 1 3 3 1 2	1 1 2 5 4 2 3 4 5 3

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)

	Học sinh trường A					***Học sinh trường 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 mảnh 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 mảnh B trong bảng.

```
(* Pas: Discover *)
uses crt;
const
fn = 'Discover.inp'; gn = 'Discover.out';
mn = 201; bl = #32; nl = #13#10;
type mi1 = 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: mi1; { xep xe truong a, b }
t, x: mi1; { t - tro truoc }
st: mi1; { 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: mi1; 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: mi1; 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: mi1): 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;
// D A T A  A N D  V A R I A B L E S
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

// P R O T O T Y P E S
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();

// I M P L E M E N T A T I O N
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 "; cinget();
   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_i 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 2 2 1 1 2 2 1 3 2 2 3 1 1 1 1	4 5 1 3 2

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 két 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;
(* D A T A   A N D   V A R I A B L E S *)
const fn = 'show.inp'; gn = 'show.out';
mn = 101; bl = #32; nl = #13#10;
type mi1 = array[0..mn] of integer;
     mb2 = array[0..mn,0..mn] of byte;
var
n, m: integer; { n - so hs; m - so ngay }
s: mi1; { s[i] - phien cuoi cua ngay i }
a,b: mi1;
c: mb2; { c[i,j] = 1 khi va chi khi hs i chon phien j }
t, st: mi1; { stack st }
p: integer; { ngon st }
sp: integer; { so phien lam viec cua BGK }
f,g: text;
(* I M P L E M E N T A T I O N *)
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: mi1; 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 <fstream>
#include <iostream>
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 viêc 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[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;
            }
        }// if
      } // for
    } // while
    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 Cặp ghép cực đại: Chị Hằng 2

Nội dung giống bài cặp 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, $0 \le v_i \le 10$. Yêu cầu ghép cặp sao cho tổng độ yêu thích đạt max.

autum2.inp	autum2.out
5 5 1 2 3 10 5 1 2 3 4 11 12 2 3 4 5 1 13 3 4 5 1 2 14 4 5	60 1 4 2 5 3 1 4 2 5 3

Input text file: autum2.inp
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 ≤ n.
Output text file: autum2.out
Dòng đầu tiên: giá trị max tổng độ yêu thích đạt được theo phương án ghép cặp đã chọn. Tiếp đến là n dòng, mỗi dòng 2 số i j cho biết em i nhận quà j.

Thuật toán

Ta quản 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 vắn tắt là *giá trị*. Để ý rằng cùng một món quà j nhưng em i sẽ đánh giá khác với em k, tức là nói chung c[i,j] ≠ 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

$$i = t_1 \leftarrow t_2 \leftarrow \dots t_{k-1} \leftarrow t_k = j \qquad (*)$$

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 mi1 = array[0..mn] of integer;
     mb2 = array[0..mn,0..mn] of byte;
var a: mi1; { a[i] = j: em i nhan qua j }
    b: mi1; { b[j] = i: qua j trong tay em i }
    t: mi1; { t[j] = i : em j nhuong qua cho ban i }
```

```
    c: mb2; { c[i][j] = 1: i thich qua j }
    d: mi1; { d[j]: gia tri gia tang tich luy den em j }
    n: integer; { so em nho }
    m: integer; { so qua }
    st: mi1; { 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: mi1; 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;
// D A T A  A N D  V A R I A B L E S
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 bất kì ta thực hiện liên tiếp các phép đổi chỗ hai phần tử cách đều đầu và cuối tính dần từ ngoài vào giữa dãy.

Độ phức tạp

Nếu đoạn cần lật có chiều dài n, mỗi lần đổi chố 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';
 write(nl,'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 <fstream>
#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 thich 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 tấm bê tông lớn được đặt tại một vị trí cố định. Trong các tấm bê tông chứa 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 tấm bê tông đầu đường băng đặt sai vị trí: chúng cần được chuyển về phia cuối đường băng. Rất may là trên công trường lúc này còn một xe đặc chủng có sức chở 1 tấm bê tông và một cần trục có sức nâng 1 tấm bê tông. Xe chạy trên đường ray song song với đường băng. Mỗi tấm bê tông cần chuyển được tháo các khớp nối và được cần trục cẩu lên đặt trên xe rồi được xe chuyển đến vị trí cần đặt lại. Tại vị trí đó cần trục lại cẩu tấm bê tông khỏi xe và đặt vào vị trí thích hợp. Cần trục cũng có thể cẩu trực tiếp 1 tấm bê tông từ một vị trí đến vị trí còn trống nào đó. Thời gian cẩu và vận chuyển một tấm 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 tấm bê tông mã số lần lượt từ 1 đến 7. 3 tấm bê tông đầu tiên là 1, 2 và 3 bị đặt sai vị trí. Sau khi chuyển lại 3 tấm này ta thu được đường băng đặt đúng là (4, 5, 6, 7, 1, 2, 3).

Thuật toán

Phương án 1. Ta gọi mỗi lần cẩu một tấm bê tông là một thao tác. Để chuyển i tấm bê tông từ đầu đường băng về cuối đường băng ta chuyển 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:

$$500500/(100\times365(\text{ngày})) \approx 13 \text{ năm}$$

Phương án 2. Ta vận dụng phép đối 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 qủa 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:

$$(u'v')' = v''u'' = vu \qquad (*)$$

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: $uv =$ **123**4567 $\Rightarrow vu =$ 4567**123**.
Vận dụng đẳng thức (*) ta có

$$(u'v')' = ((\mathbf{123})'(4567)')' = (\mathbf{321}7654)' = 4567\mathbf{123} = vu$$

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 cẩu. Vậy thuật tóan 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 $T_2 = 3/2.(i+(n–i)+n) = 3n$ 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 tính 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:

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

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 cố 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. Cẩu tấm 1 ra xe; vị trí 1 trở thành trống,

2. Cẩu tấm 7 vào vị trí 1; vị trí 7 trở thành trống,
3. Cẩu tấm 13 vào vị trí 7; vị trí 13 trở thành trống,
4. Cẩu tấm 4 vào vị trí 13; vị trí 4 trở thành trống,
5. Cẩu tấm 10 vào vị trí 4; vị trí 10 trở thành trống,
6. Cẩu tấm 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. Cẩu tấm 2 ra xe; vị trí 2 trở thành trống,
8. Cẩu tấm 8 vào vị trí 2; vị trí 8 trở thành trống,
9. Cẩu tấm 14 vào vị trí 8; vị trí 14 trở thành trống,
10. Cẩu tấm 5 vào vị trí 14; vị trí 5 trở thành trống,
11. Cẩu tấm 11 vào vị trí 5; vị trí 11 trở thành trống,
12. Cẩu tấm 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 chuyển tiếp:

Pha thứ ba:

13. Cẩu tấm 3 ra xe; vị trí 3 trở thành trống,
14. Cẩu tấm 9 vào vị trí 3; vị trí 9 trở thành trống,
15. Cẩu tấm 15 vào vị trí 9; vị trí 15 trở thành trống,
16. Cẩu tấm 6 vào vị trí 15; vị trí 6 trở thành trống,
17. Cẩu tấm 12 vào vị trí 6; vị trí 12 trở thành trống,
18. Cẩu tấm 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 cẩu ta thu được kết quả. Phương án 2 đòi hỏi $T_2 = 3n = 3.15 = 45$ lần cẩu.

Tổng quát, ta hãy tưởng tượng các tấm 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 chuyển (không tính lần chuyển s_0 ra xe) ta sẽ thu được dãy

$$xe \leftarrow s_0 \leftarrow s_1 \leftarrow s_2 \leftarrow ... \leftarrow s_k = s_0 \text{ (xe)}$$

Trong đó tấm bê tông đầu tiên s_0 được chuyển ra xe và cuối cùng, tại bước thứ k tấm đó lại được chuyển vào vị trí s_0. Ta dễ dàng nhận thấy $s_{j+1} = (s_j + i) \bmod n$, $j = 0, 1, ..., k–1$. 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 chuyển là d.

Như vậy tổng cộng sẽ có tất cả $T_3 = (k+1)d = kd + d = n + d$ lần chuyển, trong đó $d = (n,i)$. Với $n = 15$, $i = 6$ ta tính được $d = (15,6) = 3$, do đó ta chuyển trong $d = 3$ pha và tổng số lần chuyển sẽ là $T_3 = 15 + 3 = 18$.

Hàm Move dưới đây nhận vào hai giá trị: tổng số tấm bê tông n và số bê tông đầu tiên cần chuyển về cuối i sau đó giải trình trật tự chuyển các tấm 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, ..., 3^k, ... gam, k = 0, 1, 2, ... , 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 ≤ n ≤ 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 cân 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  mi1 = array[0..30] of integer;
(* Pascal *)
function ToBase(n: longint; b: integer; var p: mi1): 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 Số 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 đố. 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 *Số 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. Với thí dụ dã cho n = 69, lời gọi k = **`ToBase(69, 3, 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:

$$69 = \underline{0}.3^0 + \underline{2}.3^1 + \underline{1}.3^2 + \underline{2}.3^3 = 2120_3.$$

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, 2.3^1 = 6 g, phai[2] = 1 quả cân loại 2, 1.3^2 = 9 g, phai[3] = 2 quả cân loại 3, 2.3^3 = 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

$$2.3^i = 3.3^i - 3^i = 3^{i+1} - 3^i.$$

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 đã 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) ⇒ (0,0,2,2); trai = (0,1,0,0). Đĩa trái (69 g)+1 quả cân loại 1 = 69 + 3 = 72g; Đĩa phải: 2 quả cân loại 2 + 2 quả cân loại 3 = $2.3^2+2.3^3$ = 2.9 + 2.27 = 18 + 54 = 72g.
Đĩa trái (69+...)	0	1	0	0	0	
Đĩa phải	0	0	2	2	0	

Lại thức 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) ⇒ (0,0,0,3);

Đĩa trái (69+...)	0	1	1	0	0	trai = (0,1,1,0). Đĩa trái (69g) + 1 quả cân loại 1 + 1 quả cân loại 2 = 69 + 1.3^1 + 1.3^2 = 69 + 3 + 9 = 81 g; Đĩa phải: 3 quả cân loại 3 = 3.27 = 81 g.
Đĩa phải	0	0	0	3	0	

Cuối cùng ta thay 3 quả cân loại 3 trên đĩa phải bằng 1 quả cân loại 4 là hoàn tất.

phai = (0,0,0,3) ⇒ (0,0,0,0,1).

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 cản trên đĩa trái t[0..m−1]:

```
(* Pascal *)
function Replace(var p: mi1; var k: integer; var t: mi1): 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 dọn 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 mi1 = 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: mi1): 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: mi1; var k: longint; var t: mi1): 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: mi1; dt: longint; var p: mi1; 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: mi1;
     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);
- **`BitOf(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						**`ip = Rev(i)`.** *Xóa x tức là đặt* **`BitOn(x)`.**
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 (**`BitOn(i)`**)	no	yes	yes	yes	
	Xóa ip (**`BitOn(ip)`**)	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);
```

```
          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;
//  D A T A   A N D  V A R I A B L E
const char * fn = "biprime.inp";
const char * gn = "biprime.out";
const int mn = (500000+7)>>3;
char a[mn];
// P R O T O T Y P E S
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();
// 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;
}
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 kề viên bi đó hoặc cách viên bi đó 1 ô.

(*a*)

(*b*)

balls.inp	balls.out	*Giải thích*
3	bbborrr bbobrrr bbrborr	Input text file: số N.

	bbrbror **bbrorbr** **borbrbr** **obrbrbr** **rbobrbr** **rbrbobr** **rbrbrbo** **rbrbrob** **rbrorbb** **rorbrbb** **rrobrbb** **rrrbobb** **rrrobbb** **15**	Output text file: Dòng đầu tiên - cấu 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 cấu 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 dần 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:

$$...bor... \Rightarrow ...obr... \Rightarrow ...rbo...$$

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

$$...bor(n) \Rightarrow ...r(n)bo \Rightarrow ...or(n)b$$

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

$bor(n)b(n-1) \Rightarrow r(n)bob(n-1)$ (2n bước chuyển) $\Rightarrow r(n)obb(n-1)$ (1 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 $3.3^2 = 27$ bước chuyển cụ thể như sau:

bbborrr $\Rightarrow$ bbobrrr $\Rightarrow$ bbrborr $\Rightarrow$ bbrobrr $\Rightarrow$ bbrrbor $\Rightarrow$ bbrrobr $\Rightarrow$ bbrrrbo $\Rightarrow$ bbrrrob $\Rightarrow$
bbrrorb $\Rightarrow$ bbrorrb $\Rightarrow$ bborrrb $\Rightarrow$ bobrrrb $\Rightarrow$ brborrb $\Rightarrow$ brobrrb $\Rightarrow$ brrborb $\Rightarrow$ brrobrb $\Rightarrow$
brrrbob $\Rightarrow$ brrrobb $\Rightarrow$ brrorbb $\Rightarrow$ brorrbb $\Rightarrow$ borrrbb $\Rightarrow$ obrrrbb $\Rightarrow$ rborrbb $\Rightarrow$ 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:

Cấu 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, rbrbobr, rbrbrbo - dịch ô trống qua trái 1 ô sau đó dịch ô trống qua phải 3 lần, mỗi lần 2 ô,

–1; –2(2): rbrbrob, 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: $(-1)^i$; $(-1)^{i+1}.2(i)$; i = 1, 2, ..., n - hai phép chuyển trái dấu nhau;
Pha 2: $(-1)^{i+1}$; $(-1)^{i+1}.2(i)$; i = n–1, ..., 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)$^{i+1}$*2, i); i = 1, 2, ..., n.**
Pha 2: **Move((–1)$^{i+1}$, 1); Move((–1)$^{i+1}$*2, i); i = n–1, ..., 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;
// D A T A  A N D  V A R I A B L E
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
*/
// P R O T O T Y P E S
void Init();
void PP();
void Run();
void Balls();
void Swap(int, int);
void Move(int, int);
ofstream f("BALLS.OUT");
// I M P L E M E N T A T I O N
int main(){
    Balls();
    f << d; f.close();
    cout << endl << " Total " << d << "  move(s)" << endl;
    cout << endl << " Fini";
    system("PAUSE");
```

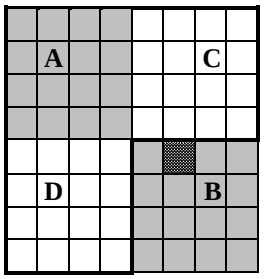
```
    return EXIT_SUCCESS;
}
void Init() {
 int i;
 for (i = 1; i <= n; ++i) a[i] = 'b'; // blue
 a[n+1] = 'o';
 n2 = n+n+1;
 for (i = n+2; i <= n2; ++i) a[i] = 'r'; // red
 d = -1; v = n+1; // vi tri o trong o
}
void PP() { // Ghi file
  for (int i = 1; i <= n2; ++i) f << a[i];
  f << endl;
  ++d;
}
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 +...+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 8×8 (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 phương án lát nền tìm được, trong đó mỗi viên gạch lát được tạo bởi ba chữ số giống nhau thể hiện màu của viên gạch đó. Các số trên mỗi dòng cách nhau qua dấu cách. Ô thoát nước được ghi số 0.

Thí dụ, với *n* = 8 và ô thóat nước tại vị trí x = 5, y = 6 ta có một cách lát nền như hình vẽ.

NEN.INP	NEN.OUT
8 5 6	8 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

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

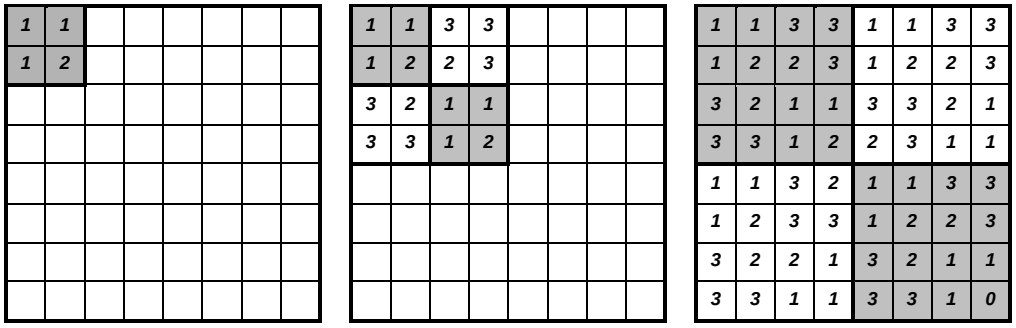
Hình 4.7.2 *Nền nhà với n = 8* (*t = 3*)

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.

Phần 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 × 2 sẽ được điền giá trị 1, ô nằm ở góc dưới phải được điền giá trị 2 (phần 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 (ô (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 trục tung) để thu được hình C (xem thủ tục **Right**).
- Lật A xuống dưới (tức là lấy đối xứng gương qua trục 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

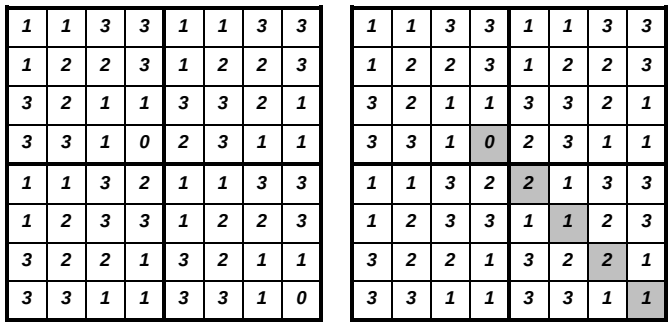
Mỗi lần lặp như vậy ta sẽ thu được hình vuông có cạnh tăng gấp đôi hình trước. Khi $k = n$ ta thu được nền nhà được lát bằng các viên gạnh chữ L với tối đa 3 màu 1, 2 và 3 cho trường hợp n > 2. Riêng ô (n,n) mang giá trị 2 sẽ được sửa thành 0.

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).

A	C
D	B

0	1
2	3



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 $\rightarrow$ A ta đặt dx = dy = –1; theo hướng D $\rightarrow$ C ta đặt dx = –1, dy = 1; theo hướng C $\rightarrow$ 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 phần 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 $x \le k/2$ và tọa độ cột của ô đó $y > k/2$. Vậy ô (x,y) đang xét nằm trong mảnh phần tư 1. Theo cách mã hóa nhị phân này, mỗi mảnh sẽ được mã số như sau: $A = 0 = 00_2$, $B = 3 = 11_2$, $C = 1 = 01_2$ và $D = 2 = 10$. 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 phần 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 $3t(n/2+n/2^2+\ldots+n/2^t) = 3tn(1/2+1/2^2+\ldots+1/2^t) = 3tn(1-1/2^{t+1}) /(1-1/2) = 6tn(1/2^{t+1})$. 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 mi1 = array[0..mn] of integer;
     mi2 = array[0..mn] of mi1;
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); { Lat 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); { Tịnh 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;
```

```
  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; { Chỉnh 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 mảnh 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 nxn
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 xuống
  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] ≠ 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 ≠ 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;
```

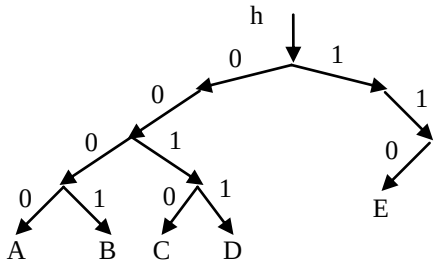
```
     col: mi1; { danh dau so mau da dung }
begin
  assign(g,gn); reset(g);
  read(g,n,x,y,colors);
  write(nl,'  n = ', n, '  x = ', x, '   y = ', y);
  fillchar(a,sizeof(a),0); fillchar(col,sizeof(col),0);
  for i := 1 to n do
  begin
    writeln;
    for j := 1 to n do
    begin
       read(g,a[i][j]); 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 ≤ n ≤ 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.*

code.inp	code.out
5 0000 0001 0010 0011 110 0000000100010000	ABBA



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^i,…, 2^k. Tổng số đỉnh của heap khi đó là

$$1 + 2 + 2^2 + \ldots + 2^k = 2^{k+1} - 1$$

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,…,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] ≠ 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 = 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,nl,nl);
  fillchar(h,sizeof(h),0);
  {Tao 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;
  // Tạo heap h
    for (i = 0; i < n; ++i) {
     f.getline(x,mn,'\n'); cout << endl <<  x;
     v = 1;
     for (j = 0; x[j]; ++j)
```

```
      v = (x[j]=='0') ? 2*v : 2*v+1;
    h[v] = ch; ++ch;
 }
 cout << endl << endl;
// Giải mã
 v = 1;
 while (!f.eof()) {
     f >> ch;
     if (ch == '0' || ch == '1') {
     cout << ch;
     v = (ch=='0') ? 2*v : 2*v+1;
     if (h[v] != 0) {
        g << h[v];
        v = 1;
     }
     }
 }
     f.close(); g.close();
 }
```

Chương 5 Luyện tập từ các đề thi

5.1 Số nguyên tố cùng độ cao

Olimpic Duy Tân 2009

Độ cao của một số tự nhiên là tổng các chữ số của số đó. Với mỗi cặp số tự nhiên n và h cho trước hãy liệt kê các số nguyên tố không vượt quá n và có độ cao h, 10 ≤ n ≤1000000; 1 ≤h ≤54.

hprimes.inp	hprimes.out
n h	mỗi dòng 1 số

Dữ liệu test

```
n = 1000, h = 16. Kết quả 15 số nguyên tố độ cao 16: 79, 97, 277,
349, 367, 439, 457, 547, 619, 673, 691, 709, 727, 853, 907.
```

Thuật toán

Thuật toán liệt kê các số nguyên tố độ cao h trong khoảng 1..n
1. Gọi thủ tục Sang(n) (do Eratosthenes đề xuất, xem Tập 2) xác định các số nguyên tố trong khoảng 1..n và đánh dấu vào mảng byte p: p[i] = 0 khi và chỉ khi i là số nguyên tố. 2. Duyệt lại các số nguyên tố i trong danh sách p, tính độ cao của số i. Nếu **Height(i) = h** thì ghi nhận. 3. end.

Để tính độ cao của số i ta tách dần các chữ số hàng đơn của i bằng phép chia dư cho 10 rồi cộng dồn vào một biến tổng.

Độ phức tạp

Thủ tục sàng duyệt $\sqrt{n}$ lần, mỗi lần lại duyệt n phần tử do đó cần cỡ $n\sqrt{n}$ thao tác cơ sở (phép nhân, phép gán).

Chương trình

```
(* Hprimes.pas - So nguyen to cung do cao *)
uses crt;
const fn = 'hprimes.inp'; gn = 'hprimes.out';
      nl = #13#10; bl = #32; mn = 1000000;
type mb1 = array[0..mn] of byte;
var p: mb1;
n,h: longint;
procedure Sang(n: longint);
var i,j: longint;
begin
     fillchar(p,sizeof(p),0);
     for i := 2 to round(sqrt(n)) do
       if (p[i] = 0) then
         for j := i to (n div i) do p[i*j] := 1;
end;
function Height(x: longint): integer;
var sum : integer;
begin
  sum := 0;
  while x <> 0 do
  begin
    sum := sum + (x mod 10);
    x := x div 10;
  end;
```

```
  Height := sum;
end;
procedure Ghi;
 var i: longint;
     G: text;
begin
  assign(g,gn); rewrite(g);
  for i := 2 to n do
    if p[i] = 0 then
      if Height(i) = h then writeln(g,i);
  close(g);
end;
procedure Doc;
 var f: text;
begin
  assign(f,fn); reset(f);
  readln(f,n,h); close(f);
end;
BEGIN
  Doc; Sang(n); Ghi;
  writeln(nl,' Fini'); readln;
END.

// DevC++: hprimes.cpp - So nguyen to cung do cao
#include <string.h>
#include <fstream>
#include <iostream>
#include <stdio.h>
#include <math.h>
using namespace std;
// D A T A  A N D  V A R I A B L E
const int mn = 1000001;
char p[mn];
int n, h;
const char * fn = "hprimes.inp";
const char * gn = "hprimes.out";
// P R O T O T Y P E S
void Doc();
void Sang();
void Ghi();
int Height(int x);
// I M P L E M E N T A T I O N
int main() {
     Doc(); Sang(); Ghi();
     cout << endl << endl <<  " Fini" << endl;
     return 0;
}
void Doc() {
   ifstream f(fn);
   f >> n >> h;
   f.close();
}
// Sang Eratosthenes
void Sang() { // p[i] = 0 <=> i nguyen to
  int can = (int) sqrt(n);
  int i, j, ni;
  memset(p,0,sizeof(p));
  for (i = 2; i <= can ; ++i)
    if (p[i] == 0)
      for (ni = n/i, j = i; j <= ni; ++j)
        p[i*j] = 1;
}
int Height(int x) { // tong so bit 1 cua x
```

```
  int d = 0;
  for (; x ; x /= 10) d += (x % 10);
  return d;
}
void Ghi() {
   int i;
   ofstream g(gn);
   for (i = 2; i <= n; ++i)
     if (p[i] == 0)
        if (Height(i) == h)
           g << endl << i;
   g.close();
}
```

5.2 Số nguyên tố cùng số bít 1

Olimpic Đại học Kỹ thuật Công nghệ Tp HCM 2009

Với mỗi n và h cho trước hãy cho biết có bao nhiêu số nguyên tố không vượt quá n và ở dạng nhị phân chứa đúng h bit 1, $10 \le n \le 1000000$; $1 \le h \le 30$.

bprimes.inp	bprimes.out	*Giải thích*
100 4	7	*Có 7 số nguyên tố trong khoảng 1..100 chứa đúng h = 4 bit 1. Đó là 23 = 10111_2 ; 29 = 11101_2 ; 43 = 101011_2 ; 53 = 110101_2; 71 =1000111_2; 83 = 1010011_2 ; 89 =1011001_2.*

Bài này là dạng đặc biệt của bài trước vì trong dạng nhị phân (chỉ chứa các chữ số 0/1) thì tổng các chữ số chính là số bit 1. Thủ tục Sum dưới đây tính tổng các chữ số của số tự nhiên x khi biểu diễn x theo dạng hệ số b bất kỳ, b > 1. Như vậy, để tính tổng các chữ số của x trong hệ đếm 10 ta gọi **Height(x,10)**. Nếu cần tính tổng các bít 1 của x ta gọi **Height(x,2)**.

```
(* Pascal *)
function Height(x: longint, b: integer): integer;
var sum : integer;
begin
  sum := 0;
  while x <> 0 do
  begin
    sum := sum + (x mod b);
    x := x div b;
  end;
  Height := sum;
end;

// DevC++
int Height(int x, int b) {
  int d = 0;
  for (; x ; x /= b) d += (x % b);
  return d;
}
```

Chương trình giải bài này giống như bài trước ngoại trừ thay đổi nhỏ là thay lời gọi hàm một tham số **Height(x)** bằng lời gọi 2 tham số **Height(x,2)**.

Dữ liệu test

```
n  = 100, h = 4: 7
    n = 1000000, h = 11:  14176
    n = 1000, h =  4: 29
```

5.3 Cắt hình

Việt Nam, 2008

Một tấm bìa dạng lưới vuông cạnh dài $n = 2^k$ tạo bởi các ô vuông đơn vị đánh số theo dòng 1.. n tính từ trên xuống và theo cột 1..n tính từ trái sang. Người ta thực hiện lần lượt hai thao tác đan xen nhau sau đây cho đến khi nhận được một cột gồm n^2 ô vuông đơn vị:

1. Cắt ngang hình theo đường kẻ giữa sau đó chồng nửa trên lên trên nửa dưới;

2. Cắt dọc hình theo đường kẻ giữa sau đó chồng nửa trái lên trên nửa phải.

Tại cột cuối cùng người ta đánh số các ô vuông đơn vị 1, 2,..., n^2 tính từ trên xuống.
Hãy viết hai thủ tục sau đây

a) **`ViTri(k, i, j) = v`** *cho ra số thứ tự* **`v`** *của ô* **`(i,j)`** *trên cột cuối cùng.*

b) **`ToaDo(k, v, i, j)`** *Cho trước k và vị trí* **`v`** *trên cột cuối cùng, tính tọa độ* **`(i,j)`** *của ô ban đầu.*

Thí dụ

`ViTri(2, 4, 3) = 8.`

`ToaDo(2,8,i, j)` *cho kết quả* **`i = 4, j = 3.`**

Thuật toán

Ta khảo sát bài toán với k = 2. Ta có $n = 2^k = 2^2 = 4$. Ta kí hiệu ô nằm trên dòng i, cột j là [i,j].

Nhận xét: Ta gọi một lần cắt ngang và một lần cắt dọc liên tiếp nhau là ND. Sau mỗi lần ND ta thu được 4 mảnh vuông và bằng nhau A, B, C và D được chồng lên nhau thành một cột theo thứ tự tính từ trên xuống là A, B, C và D (mảnh A trên cùng, mảnh D dưới cùng). Gọi d là độ dày (số tầng) của khối này. Ta có, lúc đầu d = 1 và cột chỉ có 1 tầng gồm 1 tấm bìa duy nhất ban đầu. Gọi n là chiều dài cạnh của một mảnh hình vuông. Sau mỗi lần ND, n giảm 2 lần. Vị trí v của các ô đơn vị trong mảnh A sẽ được bảo lưu, trong mảnh B được cộng thêm d, mảnh C được cộng thêm 2d và mảnh D được cộng thêm 3d. Sau mỗi lần ND số tầng d sẽ tăng thêm 4 lần. Biết tọa độ (i,j) trong hình ABCD ta dễ dàng tính được tọa độ mới (i',j') trong từng mảnh.

A	C	Tầng 1	[1,1] [1,2] [1,3] [1,4] [2,1] [2,2] [2,3] [2,4] [3,1] [3,2] [3,3] [3,4] [4,1] [4,2] [4,3] [4,4]
B	D		

4 mảnh thu được sau một lần ND

Trước khi cắt cột có duy nhất 1 tầng

Tầng 1	[1,1] [1,2] [1,3] [1,4] [2,1] [2,2] [2,3] [2,4] AC	Tầng 1: A	[1,1] [1,2] [2,1] [2,2]
		Tầng 2: B	[3,1] [3,2] [4,1] [4,2]
Tầng 2	[3,1] [3,2] [3,3] [3,4] [4,1] [4,2] [4,3] [4,4] BD	Tầng 3: C	[1,3] [1,4] [2,3] [2,4]
		Tầng 4: D	[3,3] [3,4] [4,3] [4,4]

Cắt ngang lần 1 và chồng nửa trên lên trên nửa dưới thu được 2 tầng

Cắt dọc lần 1 và chồng nửa trái lên trên nửa phải thu được 4 tầng

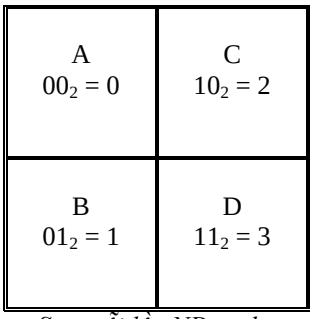
Tầng 1	[1,1] [1,2]
Tầng 2	[3,1] [3,2]
Tầng 3	[1,3] [1,4]
Tầng 4	[3,3] [3,4]
Tầng 5	[2,1] [2,2]
Tầng 6	[4,1] [4,2]
Tầng 7	[2,3] [2,4]
Tầng 8	[4,3] [4,4]

Cắt ngang lần 2 và chồng nửa trên lên trên nửa dưới thu được 8 tầng

Tầng 1	[1,1]
Tầng 2	[3,1]
Tầng 3	[1,3]
Tầng 4	[3,3]
Tầng 5	[2,1]
Tầng 6	[4,1]
Tầng 7	[2,3]
Tầng 8	[4,3]
Tầng 9	[1,2]
Tầng 10	[3,2]
Tầng 11	[1,4]
Tầng 12	[3,4]
Tầng 13	[2,2]
Tầng 14	[4,2]
Tầng 15	[2,4]
Tầng 16	[4,4]

Cắt dọc lần 2 và chồng nửa trái lên trên nửa phải

Ta chọn cách mã số các mảnh A, B, C và D một cách khôn ngoan. Cụ thể là ta gán mã số cho các mảnh này theo số lần cộng thêm độ dày d sau mỗi lần ND, tức là ta đặt A = 0, B = 1, C = 2 và D = 3. Trong dạng nhị phân ta có A = 00_2, B = 01_2, C = 10_2 và D = 11_2.



Sau mỗi lần ND ta thu được 4 mảnh A, B, C, D.

```
function ViTri(k,i,j: integer): integer;
var d, v, m, manh, n: integer;
begin
  d := 1; { so tang }
  v := 1; { tang chua o [i,j] }
  n := 1 shl k; { n = 2^k }
  for m := 1 to k do
    begin
      n := n div 2; manh := 0;
      if (j > n) then
      begin
        manh := 2; j := j - n;
      end;
      if (i > n) then
      begin
         manh := manh + 1; i := i - n;
      end;
      v := v + manh*d; d := 4*d;
    end;
   ViTri := v;
 end;
```

Thủ tục **ToaDo** là đối xứng với thủ tục **ViTri**.

```
procedure ToaDo(k,v: integer; var i,j: integer);
var n,nn,m,d, manh: integer;
begin
  n := 1 shl k; d := n*n;
  n := 1 ; { kich thuoc 1 tang }
  i := 1; j := 1;
  for m := 1 to k do
  begin
    d := d div 4;
    manh := (v-1) div d;
    if (manh >= 2) then j := j+n;
    if odd(manh) then i := i+n;
    v := v - manh*d; n := n * 2;
  end;
end;

// DevC++
#include <iostream>
using namespace std;
// P R O T O T Y P E S
int ViTri(int, int, int);
void ToaDo(int, int, int &, int &);
void Test(int);
// I M P L E M E N T A T I O N
int main() {
    Test(2);
    cout << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
// Cho biet k va toa do (i,j). Tim vi tri tren cot
int ViTri(int k, int i, int j) {
```

```
    int n = 1 << k; // kich thuoc hinh: n = 2^k
    int d = 1; // be day 1 tang
    int v = 1; // vi tri;
    int manh;
    for (int m = 0; m < k; ++m) {
      n >>=1; // n = n/2
      manh = 0;
      if (j > n) { manh = 2; j -= n; }
      if (i > n) { manh += 1; i -= n; }
      v += manh * d;  d *= 4;
    }
     return v;
}
// Cho vi tri o tren cot v, tinh toa do (i,j). n = 2^k
void ToaDo(int k, int v, int &i, int &j) {
 int n = 1, d = 1 << k, manh = 0;
  d *= d;
  i = j = 1;
  for (int m = 0; m < k; ++m) {
    d >>= 2; // d /= 4;
    manh = (v - 1) / d;
    if (manh >= 2) j += n;
    if (manh % 2 == 1) i += n;
    v -= manh * d; n *= 2;
  }
}
```

Độ phức tạp

Với $n = 2^k$ chương trình cần k lần lặp, mỗi lần lặp cần thực hiện không quá 10 phép toán số học trên các số nguyên.

Thủ tục Test dưới đây hiển thị vị trí v của mọi ô (i,j), i = 1..n, j = 1..n sau đó xuất phát từ vị trí v để tính lại tọa độ i, j.

```
(* Pascal *)
procedure Test(k: integer);
var n, v, ii, jj : integer;
begin
  n := 1 shl k; { n = 2^k }
  writeln; writeln( ' k = ', k, '  n = ', n);
  for i := 1 to n do
    for j := 1 to n do
    begin
      v := ViTri(k, i, j); ToaDo(k, v, ii, jj);
      writeln; write('(',i, ',', j, ') => ', v);
      writeln(' => ', '(', ii, ',', jj, ')');
      readln;
    end;
end;

// DevC++
void Test(int k) {
   int n = 1 << k;
   cout << endl << "k = " << k << "  n = " << n
   int v, ii, jj;
   for (int i = 1; i <= n; ++i)
      for (int j = 1; j <= n; ++j) {
        v = ViTri(k, i, j); ToaDo(k, v, ii, jj);
        cout << endl << "(" << i << "," << j << ") => "
             << v << " => (" << ii << "," << jj << ")";
        cin.get();
      }
}
```

5.4 Tổng nhỏ nhất

Cho hai dãy số nguyên a_i và b_i, i = 1, 2, ..., n chứa các gía trị trong khoảng -1 tỷ đến 1 tỷ, $1 \le n \le 1000$. Tìm giá trị nhỏ nhất của $||a_i+b_j||$, $1 \le i, j \le n$.

MINSUM.INP	MINSUM.OUT
5 7 5 1 -3 6 8 9 1 5 -9	2

Thuật toán

Trước hết ta sắp tăng hai dãy a và b. Sau đó, với mảng a ta duyệt xuôi theo chỉ số i biến thiên từ 1..n, với mảng b ta duyệt ngược theo chỉ số j biến thiên từ n về 1. Đặt $t(i,j) = a_i + b_j$ ta có nhận xét sau:

1. t là một hàm 2 biến theo i và j,
2. Nếu cố định i, cho j giảm dần thì t là hàm đồng biến theo j, nghĩa là $t(i,j) \ge t(i,j')$ nếu $j > j'$.
3. Nếu cố định j thì t là hàm đồng biến theo i, nghĩa là $t(i,j) \le t(i',j)$ nếu $i < i'$.

Từ nhận xét trên ta suy ra chiến lược tìm kiếm trị nhỏ nhất của ||t(i,j)|| trong hàm XuLi như sau:

- Nếu t(i,j) = 0 ta nhận giá trị này và kết thúc thuật toán.
- Nếu t(i,j) < 0 ta cần tăng giá trị của hàm này bằng cách tăng i thêm 1 đơn vị.
- Nếu t(i,j) > 0 ta cần giảm giá trị của hàm này bằng cách giảm j bớt 1 đơn vị.

Sau một số bước lặp ta gặp tình huống, hoặc i = n hoặc j = 1. Ta xử lý tiếp như sau:

- Nếu i = n, ta cần duyệt nốt phần còn lại b[j..1], nghĩa là tính t(n,k) với k = j..1. Vì đây là hàm đồng biến nên khi $t(n,k) \le 0$ ta dừng thuật toán.
- Tương tự, nếu j = 1, ta cần duyệt nốt phần còn lại của a[j..n], nghĩa là tính t(k,1) với k = i..n. Vì đây là hàm đồng biến nên khi $t(k,1) \ge 0$ ta cũng dừng thuật toán.

Tại mỗi bước lặp ta tính và cập nhật giá trị min m của || t ||.

Độ phức tạp

Thuật toán sắp xếp nhanh cần $n.\log_2(n)$ phép so sánh. Khi tính giá trị min, mỗi bước lặp ta thay đổi đúng 1 trong 2 chỉ số i hoặc j do đó số phép so sánh là 2n. Tổng hợp lại ta cần cỡ $n.\log_2(n)$ phép so sánh.

Chương trình

```
(* Pascal *)
const
 bl = #32; nl = #13#10;
 fn = 'minsum.inp';
 gn = 'minsum.out';
 mn = 1001;
type ml1 = array[1..mn] of longint;
var
 a,b: ml1;
 n: integer;
 f,g: text;
procedure Print(var a: ml1; d,c: integer);
 var i: integer;
begin
  writeln;
  for i:= d to c do write(a[i],bl);
end;
procedure Sort(var a: ml1; d, c: integer);
var i,j: integer; t,m: longint;
begin
   i := d; j := c; m := a[(d+c) div 2];
   while (i <= j) do
   begin
     while (a[i] < m) do inc(i);
     while (a[j] > m) do dec(j);
     if (i <= j) then
     begin
       t := a[i]; a[i] := a[j]; a[j] := t;
       inc(i); dec(j);
     end;
```

```
   end;
   if (d < j) then Sort(a,d,j);
   if (i < c) then Sort(a,i,c);
end;
procedure Ghi(m: longint);
begin
   assign(g,gn); rewrite(g);
   writeln(g,m);
   close(g);
end;
function XuLi: longint;
 var i,j,v: integer;
     t,m: longint;
begin
  XuLi := 0;
  Sort(a,1,n);  Sort(b,1,n);
  writeln(nl,'Sau khi sap tang ');
  Print(a,1,n); Print(b,1,n);
  i := 1; j := n; m := MaxLongInt;
  while (i <= n) and (j >= 1) do
  begin
    t := a[i]+b[j]; v := abs(t);
    if (m > v) then m := v;
    if t = 0 then exit;
    if (t < 0) then inc(i) else dec(j);
  end;
  { i = n+1 or j = 0 }
  for i := i to n do
  begin
    t := a[i]+b[1]; v := abs(t);
    if (m > v) then m := v;
    if (t >= 0) then begin XuLi := m; exit end;
  end;
  for j := j downto 1 do
  begin
    t := a[n]+b[j]; v := abs(t);
    if (m > v) then m := v;
    if (t <= 0) then begin XuLi := m; exit end;
  end;
  XuLi := m;
end;
procedure Doc;
var i: integer;
begin
   assign(f,fn); reset(f);
   readln(f,n);
   for i := 1 to n do read(f,a[i]);
   for i := 1 to n do read(f,b[i]);
   close(f);
end;
procedure Run;
begin
    Doc;
    writeln(nl,n);
    Print(a,1,n); Print(b,1,n);
    Ghi(XuLi);
end;
BEGIN
 Run;
 readln;
END.
```

```
// DevC++
#include <string.h>
#include <fstream>
#include <iostream>
#include <stdio.h>
#include <Math.h>
using namespace std;
// D A T A  A N D  V A R I A B L E
const char * fn = "minsum.inp";
const char * gn = "minsum.out";
const int mn = 1001;
int a[mn], b[mn];
int n;
// P R O T O T Y P E S
void Doc();
void Print(int [], int , int);
int XuLi();
void Ghi(int);
void Sort(int [], int, int);
void Run();
// I M P L E M E N T A T I O N
int main(){
    Run();
    cout << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
void Run(){
    Doc();
    cout << endl << n;
    Print(a,1,n); Print(b,1,n);
    Ghi(XuLi());
}
void Ghi(int m){
   ofstream g(gn);
   g << m;
   g.close();
}
int XuLi(){
  Sort(a,1,n);
  Sort(b,1,n);
  cout << endl << "Sau khi sap tang ";
  Print(a,1,n); Print(b,1,n);
  int i = 1, j = n, m = 0x7fffffff, t, v;
  while (i <= n && j >= 1){
    t = a[i]+b[j]; v = abs(t);
    if (m > v) m = v;
    if (t == 0) return 0;
    if (t < 0) ++i;
         else --j;
  }
  // i = n+1 || j = 0
  for (;i <= n; ++i){
    t = a[i]+b[1]; v = abs(t);
    if (m > v) m = v;
    if (t >= 0) return m;
  }
  for (;j >= 1;--j){
    t = a[n]+b[j]; v = abs(t);
    if (m > v) m = v;
    if (t <= 0) return m;
  }
  return m;
```

```
}
void Sort(int a[], int d, int c){
   int i = d, j = c, m = a[(d+c)/2], t;
   while (i <= j){
     while (a[i] < m) ++i;
     while (a[j] > m) --j;
     if (i <= j) {
       t = a[i]; a[i] = a[j]; a[j] = t;
       ++i; --j;
      }
   }
   if (d < j) Sort(a,d,j);
   if (i < c) Sort(a,i,c);
}
void Doc(){
   ifstream f(fn);
   f >> n;
   int i;
   for (i = 1; i <= n; ++i) f >> a[i];
   for (i = 1; i <= n; ++i) f >> b[i];
   f.close();
}
void Print(int a[], int d, int c){
  int i;
  cout << endl;
  for (i = d; i <= c; ++i) cout << a[i] << " ";
}
```

5.5 Lò cò

Việt Nam, 2008

Cho n vòng tròn, mỗi vòng tròn chứa một giá trị nguyên dương có thể trùng nhau. Gọi a_i là giá trị chứa trong vòng tròn i. Nếu ba vòng tròn i. j, k khác nhau và thỏa đẳng thức $a_i+a_j = a_k$ thì ta vẽ 2 cung (có hướng) $i \to k$ và $j \to k$. Cho biết đường đi dài nhất có thể qua bao nhiêu đỉnh.

loco.inp	loco.out	***Giải thích*** *Với 10 vòng tròn chứa các giá trị liệt kê trong file loco.inp, một đường đi dài nhất qua 6 đỉnh chứa giá trị sau:* ①→③→④→⑤→⑥→⑧
10 4 6 5 8 1 2 3 2 3 4	6	

Thuật toán

Sau khi đọc dữ liệu vào mảng a và sắp tăng mảng a ta thấy rằng mỗi cung $i \to k$ trong đồ thị có hướng G xây dựng theo yêu cầu của đầu bài luôn luôn kèm với một cung $j \to k$ khi và chỉ khi a[i]+a[j] = a[k]; $i \neq j$; i,j < k. G chính là đồ thị một chiều (có cung hướng từ đỉnh số hiệu nhỏ tới đỉnh số hiệu lớn), không chu trình. Ngoài ra, do dãy a chỉ gồm các giá trị nguyên dương và được sắp tăng nên với i, j cho trước thì đẳng thức a[i] + a[j] = a[k] chỉ có hy vọng đạt được với những k > max(i,j). Ta sử dụng một mảng d với các phần tử d[k] ghi nhận số đỉnh nằm trên đường dài nhất đi đến đỉnh k. Ta có

$$d[k] = \text{Max} \{ \text{Max}(d[i]+1, d[j]+1) \mid 1 \le i < j < k, a[i] + a[j] = a[k] \}$$

Ta khởi trị cho d[k] = 1 với mọi k = 1..n với ý nghĩa là khi chỉ có 1 đỉnh độc lập thì đường dài nhất chỉ chứa 1 đỉnh đó. Mỗi khi đạt hệ thức a[i]+a[j] = a[k] ta chỉnh lại d[k] là giá trị max của 3 giá trị: d[k], d[i]+1 và d[j]+1, đồng thời ta cũng xác định giá trị dmax cho toàn bài.

Do a là dãy sắp tăng nên với mỗi k ta chỉ duyệt các cặp đỉnh i, j thỏa $1 \le i < j < k$. Ngoài ra, để tính d[k], với mỗi i đã chọn, tức là khi đã biết k và i, nếu tồn tại j để a[i] + a[j] = a[k] thì ta không cần xét tiếp các giá trị a[j] khác.

Hàm chính của chương trình MaxPath hoạt động như sau:

Với mỗi k = 2..n hàm tính giá trị d[k] là số đỉnh nhiều nhất trong số các đường đi kết thúc tại đỉnh k. Đồng thời hàm ghi nhận giá trị max của các d[k].

Hàm Tinhd(k) thực hiện chức năng tính giá trị d[k]. Hàm này hoạt động như sau:

Phác thảo thuật toán tính d[k]
Biết trước k; Với mỗi i = 1..(k–1) và với mỗi j = (i+1)..(k–1): Kiểm tra điều kiện có hướng: Nếu a[k] = a[i]+a[j] chứng tỏ có đường đi $i \to k$ và

j→k thì ta chọn đường dài nhất (qua i hoặc j) đến k và cập nhật lại d[k] thông qua lời gọi hàm d[k] := Max(d[k], d[i]+1,d[j]+1).

end.

Độ phức tạp

Ta có 3 vòng lặp: theo k trong hàm MaxPath và 2 vòng lặp theo i và j trong hàm Tind(k) nên độ phức tạp cỡ n^3.

Chương trình

```
(* Loco.cpp *)
uses crt;
const
  fn = 'loco.inp'; gn = 'loco.out';
  mn = 10001; bl = #32; nl = #13#10;
type mi1 = array[0..mn] of integer;
var
 f,g: text;
 a,d: mi1;
 n: integer;
procedure Doc;
var i: integer;
begin
  assign(f,fn); reset(f);
  readln(f,n);
  for i := 1 to n do read(f,a[i]);
  close(f);
end;
procedure Sort(d,c: integer);
  var i, j, t, m: integer;
begin
  i := d; j := c; m := a[(d+c) div 2];
  while (i <= j) do
  begin
   while (a[i] < m) do inc(i);
   while (a[j] > m) do dec(j);
   if (i <= j) then
   begin
     t := a[i]; a[i] := a[j]; a[j] := t;
     inc(i); dec(j);
   end;
  end;
  if (d < j) then Sort(d,j);
  if (i < c) then Sort(i,c);
end;
{ Max cua 3 so nguyen a, b, c }
function Max(a, b, c: integer): integer;
begin
  if a < b then a := b;
  if a < c then Max := c else Max := a;
end;
procedure Tinhd(k: integer);
 var i,j,t: integer;
begin
  d[k] := 1;
  for i := 1 to k-1 do
    for j := i + 1 to k - 1 do
    begin
      t := a[i] + a[j];
      if t > a[k] then break;
      if t = a[k] then
      begin
```

```
        d[k] := Max(d[k], d[i]+1, d[j]+1);
        break;
      end;
    end;
end;
function MaxPath: integer;
 var k, dmax: integer;
begin
  MaxPath := 0;
  if (n = 0) then exit;
  d[1] := 1; dmax := 1;
  for k := 2 to n do
  begin
    Tinhd(k);
    if d[k] > dmax then dmax := d[k];
  end;
  MaxPath := dmax;
end;
procedure Ghi(m: integer);
begin
  assign(g,gn); rewrite(g);
  writeln(g,m); close(g);
end;
BEGIN
  Doc; Sort(1,n); Ghi(MaxPath);
END.

// DevC++: Loco.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 char * fn = "loco.inp";
const char * gn = "loco.out";
const int mn = 10001;
int a[mn], d[mn];
int n;
// P R O T O T Y P E S
void Doc();
void Sort(int, int);
int Max(int, int, int);
int MaxPath();
void Tinhd(int);
void Ghi(int);
// I M P L E M E N T A T I O N
int main(){
    Doc();  Sort(1,n); Ghi(MaxPath());
    cout << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
void Ghi(int m){
  ofstream g(gn); g << m; g.close();
}
// Max cua 3 so nguyen a, b, c
int Max(int a, int b, int c){
  if (a < b) a = b;
  return (a > c) ? a : c;
}
int MaxPath(){
   int k, dmax = 1;
   if (n == 0) return 0;
```

```
   d[1] = 1;
   for (k = 2; k <= n; ++k){
      Tinhd(k);
      if (dmax < d[k]) dmax = d[k];
    }
   return dmax;
}
void Tinhd(int k){
   int i, j, t;
   d[k] = 1;
   for (i = 1; i < k ; ++i)
     for (j = i+1 ; j < k ; ++j){
        t = a[i]+a[j];
        if (t > a[k]) break;
        if (t == a[k]){
           d[k] = Max(d[k],d[i]+1,d[j]+1);
           break;
        }
     }
}
void Doc(){
     ifstream f(fn);
     f >> n;
     int i;
     for (i=1; i <= n; ++i)
       f >> a[i];
     f.close();
}
void Sort(int d, int c){
     int i = d, j = c, m = a[(i+j)/2];
     int t;
     while (i <= j){
       while (a[i] < m) ++i;
       while (a[j] > m) --j;
       if (i <= j){
         t = a[i]; a[i] = a[j]; a[j] = t;
         ++i; --j;
       }
     }
     if (d < j) Sort(d,j);
     if (i < c) Sort(i,c);
}
```

Cải tiến 1

Khi tính d[k] ta nên tìm kiếm nhị phân trên đoạn được sắp tăng a[i+1..k–1] như sau. Biết trước a[k], với mỗi i ta biết a[i] và cần tìm a[j] trong đoạn a[i+1..k–1] thoả điều kiện a[i] + a[j] = a[k]. Đặt t = a[k] – a[i], bài toán quy về việc tìm phần tử a[j] = t trong đoạn sắp tăng a[i+1..k–1]. Nếu tìm được thì ta cập nhật d[k]. Cải tiến này giảm độ phức tạp xuống còn (log n).n^2.

Hàm Binsearch(t,s,e) dưới đây thực hiện việc tìm kiếm nhị phân giá trị t trong đoạn sắp tăng a[s..e] của mảng a. Hàm cho ra chỉ số j trong khoảng s..e nếu a[j] = t; ngược lại, khi không tồn tại chỉ số j như vậy thì hàm cho ra giá trị 0. Hàm Binsearch hoạt động như sau:

Phác thảo thuật toán cho hàm Binsearch
1. Xác định phần tử m = a[(s+e)/2] là phần tử giữa của đoạn a[s..e]; 2. So sánh t với m: Nếu t > m: sẽ tìm kiếm tiếp trên đoạn từ s = m+1 đến e; Nếu t ≤ m: sẽ tìm kiếm tiếp trên đoạn từ s đến e = m. 3. Các bước 1 và 2 sẽ được lặp đến khi s = e. 4. Khi s = e: Nếu a[s] = t thì return s; nếu không: return 0; 5. end.

```
(* Pascal: Loco, Cai tien 1  *)
uses crt;
const
  fn = 'loco.inp'; gn = 'loco.out';
  mn = 10001; bl = #32; nl = #13#10;
type mi1 = array[0..mn] of integer;
var
 f,g: text;
 a,d: mi1;
 n: integer;
procedure Doc; Tự viết
procedure Sort(d,c: integer); Tự viết
function Max(a, b, c: integer): integer; Tự viết
{ Tim j trong khoang s..e thoa: a[j] = t }
function Binsearch(t,s,e: integer): integer;
  var m: integer;
begin
  Binsearch := 0;
  if s > e then exit;
  while (s < e) do
  begin
   m := (s+e) div 2;
   if (a[m] <  t) then s := m+1 else e := m;
  end;
  if (a[s] = t) then Binsearch := s else Binsearch := 0;
end;
procedure Tinhd(k: integer);
 var i,j,t: integer;
begin
  d[k] := 1;
  for i := 1 to k-1 do
  begin
    t := a[k] - a[i];
    if (t > 0) then
    begin
      j := Binsearch(t, i+1, k-1);
      if j > 0 then
        d[k] := Max(d[k], d[i]+1, d[j]+1);
    end;
  end;
end;
function MaxPath: integer;
 var k, dmax: integer;
begin
if (n = 0) then begin MaxPath := 0; exit end;
if (n = 1) then begin MaxPath := 1; exit end;
  dmax := 1; d[1] := 1; d[2] := 1;
  for k := 3 to n do
  begin
    Tinhd(k);
    if d[k] > dmax then dmax := d[k];
  end;
  MaxPath := dmax;
end;
procedure Ghi(m: integer); Tự viết
BEGIN
  Doc; Sort(1,n); Ghi(MaxPath);
END.


// DevC++, Loco.cpp: Cai tien 1
```

```
#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 char * fn = "loco.inp";
const char * gn = "loco.out";
const int mn = 10001;
int a[mn], d[mn];
int n;
// P R O T O T Y P E S
void Doc();
void Sort(int, int);
int Max(int, int, int);
int Binsearch(int, int, int);
int MaxPath();
void Tinhd(int);
void Ghi(int);
// I M P L E M E N T A T I O N
int main(){
    Doc(); Sort(1,n); Ghi(MaxPath());
    return EXIT_SUCCESS;
}
void Ghi(int m) tự viết
int Max(int a, int b, int c) tự viết
// Tim nhi phan vi tri xua hien cua x trong a[s.e]
int Binsearch(int t, int s, int e){
    int m;
    if (s > e) return 0;
    while (s < e) {
       m = (s+e) / 2;
       if (a[m] < t) s = m + 1; else e = m;
    }
    return (a[s] == t) ? s : 0;
}
int MaxPath(){
   if (n == 0) return 1;
   if (n == 1) return 1;
   d[1] = d[2] = 1;
   int k, dmax = 1;
   for (k = 3; k <= n; ++k){
      Tinhd(k);
      if (dmax < d[k]) dmax = d[k];
    }
   return dmax;
}
void Tinhd(int k){
   int i, j, t;
   d[k] = 1;
   for (i = 1; i < k ; ++i) {
        t = a[k]-a[i];
        if (t > 0){
          j = Binsearch(t,i+1,k-1);
          if (j > 0)
           d[k] = Max(d[k],d[i]+1,d[j]+1);
        }
     }
}
void Doc()  tự viết;
void Sort(int d, int c)  tự viết;
```

Cải tiến 2

Để ý rằng nếu trong dãy a có hai phần tử a[i] = a[j], tức là có hai đỉnh chứa cùng một giá trị thì trong kết quả cuối cùng ta phải có d[i] = d[j], tức là số lượng đỉnh trên các đường đi dài nhất kết thúc tại i và j phải bằng nhau. Tuy nhiên ta phải xử lý trường hợp a[i] = a[j], i ≠ j và tồn tại k để a[i]+a[j] = 2a[i] = a[k]. Khi đó ta vẫn có 2 cung i→k và j→k; và d[k] dĩ nhiên cần được cập nhật. Nhận xét này cho phép ta lược bớt các gía trị trùng lặp chỉ giữ lại tối đa hai phần tử bằng nhau.

Cải *tiến 3*

Nếu các giá trị của dãy a là đủ nhỏ, thí dụ nằm trong khoảng 1.. 20000 thì ta có thể dùng mảng để đánh dấu. Ta sẽ mã số đỉnh bằng chính giá trị chứa trong đỉnh. Ta đặt s[i] = 1 nếu i xuất hiện duy nhất 1 lần trong input file; s[i] = 2 nếu i xuất hiện nhiều lần trong input file; ngược lại, nếu i không xuất hiện trong dãy thì ta đặt s[i] = 0.

i	1	2	3	4	5	6	7	8
s[i]	1	2	2	2	1	1	0	1

Mảng s sau khi đọc dữ liệu:
s[i] = 1 nếu i xuất hiện duy nhất 1 lần;
s[i]= 2 nếu i xuất hiện hơn 1 lần;
s[i] = 0 nếu i không xuất hiện trong dãy.

Sau đó, để tính d[k] ta chỉ xét những giá trị k xuất hiện trong dãy đã cho, nghĩa là s[k] > 0. Với mỗi i = 1..k/2 ta xét, nếu i và j = k–i đều có trong dãy, tức là s[i] > 0 và s[k–i] > 0 thì ta đặt d[k] = max(d[k], d[i] + 1, d[k–i] + 1). Cuối cùng ta xét thêm trường hợp k là số chẵn và s[k/2] = 2, nghĩa là k là tổng của hai số bằng nhau và có mặt trong dãy.

i	1	2	3	4	5	6	7	8
s[i]	1	2	2	2	1	1	0	1
d[i]	1	1	2	3	4	5	0	6

Mảng d thu được sau khi xử lý theo s
d[i] – số đỉnh trên đường dài nhất kết thúc tại đỉnh i.

Cải tiến này rút độ phức tạp tính toán xuống còn n^2.
Bạn lưu ý sử dụng Free Pascal để có đủ miền nhớ.

```
(* Loco.pas: Cai tien 3  *)
uses crt;
const
  fn = 'loco.inp'; gn = 'loco.out';
  mn = 20000; bl = #32; nl = #13#10;
type mi1 = array[0..mn+1] of integer;
var
 f,g: text;
 s,d: mi1;
 n: integer;
 maxv: integer; { Gia tri max trong input file }
procedure Doc;
var i,j: integer;
begin
  assign(f,fn); reset(f); maxv := 0;
  readln(f,n);  fillchar(s,sizeof(s),0);
  for i := 1 to n do
  begin
    read(f,j);
    if (maxv < j) then maxv := j;
    if s[j] > 0 then s[j] := 2 else s[j] := 1;
  end;
  close(f);
end;
function Max(a, b, c: integer): integer;  tự viết
procedure Tinhd(k: integer);
 var i,k2: integer;
begin
```

```
  d[k] := 1; k2 := (k-1) div 2;
  for i := 1 to k2 do
    if (s[i] > 0) and (s[k-i] > 0) then
          d[k] := Max(d[k], d[i]+1, d[k-i]+1);
  if (not odd(k)) then
  begin
    inc(k2);
    if (s[k2] = 2) then
      d[k] := Max(d[k], d[k2]+1, d[k2]+1);
  end;
end;
function MaxPath: integer;
 var k, dmax: integer;
begin
  fillchar(d,sizeof(d),0); dmax := 0;
  for k := 1 to maxv do
   if s[k] > 0 then
  begin
    Tinhd(k);
    if d[k] > dmax then dmax := d[k];
  end;
  MaxPath := dmax;
end;
procedure Ghi(m: integer);  tự viết
BEGIN
  Doc; Ghi(MaxPath);
END.


// DevC++: Loco.cpp, Phuong an Cai tien 3
#include <string.h>
#include <fstream>
#include <iostream>
using namespace std;
// D A T A  A N D  V A R I A B L
const char * fn = "loco.inp";
const char * gn = "loco.out";
const int mn = 10001;
int s[mn], d[mn];
int n;
int maxv; // Gia tri max trong input file
// P R O T O T Y P E S
void Doc();
int Max(int, int, int);
int MaxPath();
void Tinhd(int);
void Ghi(int);
// I M P L E M E N T A T I O N
int main(){
    Doc();
    Ghi(MaxPath());
    cout << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
void Ghi(int m) Tự viết
int Max(int a, int b, int c)  Tự viết
int MaxPath(){
   int k, dmax = 0;
   memset(d,0,sizeof(d));
   for (k = 1; k <= maxv; ++k)
   if (s[k] > 0){
      Tinhd(k);
```

```
        if (dmax < d[k]) dmax = d[k];
      }
    return dmax;
}
void Tinhd(int k){
   int i, k2 = (k-1)/2;
   d[k] = 1;
   for (i = 1; i <= k2 ; ++i)
     if (s[i] > 0 && s[k-i] > 0)
        d[k] = Max(d[k],d[i]+1,d[k-i]+1);
   k2 = k/2;
   if ((2*k2 == k) && (s[k2] > 1))
       d[k] = Max(d[k],d[k2]+1,d[k2]+1);
}
void Doc(){
     ifstream f(fn);
     f >> n;
     memset(s,0,sizeof(s));
     int i, j;
     for (i=1; i <= n; ++i){
       f >> j;
       if (maxv < j) maxv = j;
       if (s[j] > 0) s[j] = 2; else s[j] = 1;
     }
     f.close();
}
```

5.6 Chuyển tin

Bách khoa Đà Nẵng, 2001

Cần chuyển hết n gói tin trên một mạng có m kênh truyền. Biết chi phí để chuyển i gói tin trên kênh j là C(i,j). Cho biết một phương án chuyển với chi phí thấp nhất.

Dữ liệu vào: file văn bản messages.inp:

Dòng đầu tiên: 2 số n và m;

Dòng thứ i trong số n dòng tiếp theo: Dãy m số nguyên dương b_1, b_2, ..., b_m trong đó b_j là chi phí chuyển i gói tin trên kênh j.

Dữ liệu ra: file văn bản messages.out:

Dòng đầu tiên: tổng chi phí thấp nhất theo phương án tìm được;

Dòng thứ j trong số m dòng tiếp theo: số lượng gói tin chuyển trên kênh j.

messages.inp	messages.out
5 4 31 10 1 1 1 31 12 13 4 10 31 1 6 1 20 19 10 5 10 10	2 0 4 1 0

Ý nghĩa

Với n = 5 gói tin, m = 4 kênh và chi phí c(i,j) cho trước, trong đó i là chỉ số dòng (số gói tin), j là chỉ số cột (kênh) thì cách chuyển sau đây sẽ cho chi phí thấp nhất là 2

Kênh	*Số gói tin*	*Chi phí*
1	0	0
2	4	1
3	1	1
4	0	0

Thuật toán Quy hoạch động

Gọi s(i,j) là chi phí thấp nhất của phương án chuyển hết i gói tin trên mạng gồm j kênh đầu tiên 1, 2, ..., j. Ta xét kênh thứ j và thử chuyển lần lượt k = 0, 1, ..., i gói tin theo kênh này. Ta thấy, nếu chuyển k gói tin theo kênh j thì chi phí cho kênh j này sẽ là c(k,j), còn lại i–k gói tin phải chuyển trên j–1 kênh đầu tiên với chi phí là s(i–k,j–1), vậy phương án này cho chi phí là c(k,j) + s(i–k,j–1), k = 0, 1, 2,...,i.

Ta có

$$s(i,j) = \max \{ c(k,j) + s(i-k,j-1) \mid k = 0, 1, ..., i \}$$

Để cài đặt, ta có thể có thể dùng 1 mảng 1 chiều p với m bước lặp. Tại bước lặp thứ j ta có p[i] = s(i,j) là chi phí thấp nhất khi chuyển hết i gói tin trên mạng với j kênh đầu tiên. Vậy

$$p[i] = \max \{ c(k,j) + p[i-k] \mid k = i, i-1,...,0 \}; i = n..1$$

Chú ý rằng để khỏi ghi đè lên giá trị còn phải dùng để xử lý bước sau, với mỗi kênh j ta cần duyệt i theo chiều ngược từ n đến 1.

Điểm khó là phải giải trình cụ thể phương án chuyển tin tối ưu, nghĩa là cho biết phải chuyển theo mỗi kênh bao nhiêu gói tin. Ta sẽ sử dụng một mảng 2 chiều SoGoi, trong đó phần tử SoGoi[i][j] cho biết trong phương án tối ưu chuyển i gói tin theo j kênh đầu tiên thì riêng kênh j sẽ được phân phối bao nhiêu gói tin. Trước khi ghi kết quả ta tính ngược từ kênh m đến kênh 1 số gói tin cần chuyển theo mỗi kênh.

Độ phức tạp

Thủ tục XuLi tính m.n lần, mỗi lần lại tính tối đa m phần tử, vậy độ phức tạp là $O(nm^2)$.

Chương trình

```
(* messages.pas *)
const fn = 'messages.inp'; gn = 'messages.out';
      mn = 101; bl = #32; nl = #13#10;
type mi1 = array[0..mn] of integer;
     mi2 = array[0..mn] of mi1;
var f,g: text;
    c,sogoi: mi2; { c - ma tran chi phi }
    { sogoi[i,j] - so goi tin chuyen theo kenh j }
    p: mi1;
    n,m: integer; { n - tong so goi tin; m - so kenh }
procedure Ghi;
 var i,j: integer;
begin
  assign(g,gn); rewrite(g); writeln(g,p[n]);
  i := n;
  for  j := m downto 1 do
  begin
    p[j] := SoGoi[i,j]; { so goi chuyen theo kenh j }
    i := i - SoGoi[i,j]; { so goi con lai }
  end;
  for i := 1 to m do writeln(g,p[i]);
  close(g);
end;
{ Chi phi chuyen het i goi tin
  theo j kenh dau tien: 1, 2, ... j }
procedure Tinhp(i,j: integer);
 var k: integer;
begin
  SoGoi[i,j] := 0;
  for k := 1 to i do { thu chuyen k goi theo kenh j }
     if (p[i] > p[i-k] + c[k][j]) then
     begin
           p[i] := p[i-k] + c[k][j]; { cap nhat tri min }
           SoGoi[i,j] := k; { chuyen k goi theo kenh j }
     end;
end;
procedure XuLi;
 var i, j, k: integer;
begin
  fillchar(SoGoi,sizeof(SoGoi),0);
  { Khoi tri cho kenh 1 }
  { i goi tin chi chuyen tren kenh 1 voi chi phi c[i][1] }
  for i := 1 to n do
  begin p[i] := c[i,1]; SoGoi[i,1] := i; end;
  for j := 2 to m-1 do { xet kenh j }
    for i := n downto 1 do
     Tinhp(i,j); { chi phi chuyen i goi tin theo j kenh dau tien }
  { Xu li buoc cuoi, kenh m }
  SoGoi[n,m] := 0;
  for k := 1 to n do
```

```
    if (p[n] > p[n-k] + c[k][m]) then
     begin
      SoGoi[n,m] := k;
      p[n] := p[n-k] + c[k][m];
     end;
end;
procedure Print;
 var i,j: integer;
begin
  writeln(nl, n, bl, m);
  for i := 1 to n do
  begin writeln;
    for j := 1 to m do write(c[i,j],bl);
  end;
end;
procedure Doc;
 var i,j: integer;
begin
   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;
BEGIN
    Doc; Print;
    XuLi; Ghi;
    writeln(nl,' Fini ');
    readln;
END.

// DevC++ messages.cpp
#include <string.h>
#include <fstream>
#include <iostream>
#include <stdio.h>
using namespace std;
// D A T A  A N D  V A R I A B L E
const char * fn = "messages.inp";
const char * gn = "messages.out";
const int mn = 101;
int c[mn][mn]; // ma tran chi phi
int SoGoi[mn][mn];//SoGoi[i][j]-so goi tin chuyen theo kenh j
int n, m;
int p[mn];
// P R O T O T Y P E S
void Doc();
void Print();
void XuLi();
void Tinhp(int, int);
void Ghi();
// I M P L E M E N T A T I O N
int main() {
    Doc(); Print();
    XuLi(); Ghi();
    cout << endl; system("PAUSE");
    return EXIT_SUCCESS;
}
void Ghi(){
 ofstream g(gn);
 g << p[n]; // chi phi min
 int i = n, j;
```

```
 for (j = m; j > 0; --j) {
   p[j] = SoGoi[i][j];
   i = i - SoGoi[i][j];// so goi con lai
 }
 for (i = 1; i <= m; ++i) g << endl << p[i];
 g.close();
}
// Chi phi chuyen het i goi tin
// theo j kenh dau tien: 1, 2, ... j
void Tinhp(int i, int j) {
   int k;
     SoGoi[i][j] = 0;
     for (k = 1; k <= i; ++k)  // thu chuyen k goi theo kenh j
        if (p[i] > p[i-k] + c[k][j]) {
           p[i] = p[i-k] + c[k][j]; // cap nhat tri min
           SoGoi[i][j] = k; // so goi can chuyen theo kenh j
        }
}
void XuLi(){
  int i, j, k;
  memset(SoGoi, 0,sizeof(SoGoi));
  // Khoi tri cho kenh 1
  // i goi tin chi chuyen tren kenh 1 voi chi phi c[i][1]
  p[0] = 0; // chuyen 0 goi tin tren kenh 1
  for (i = 1; i <= n; ++i) {
      p[i] = c[i][1]; SoGoi[i][1] = i;
  }
  for (j = 2; j < m; ++j) // xet kenh j = 2..m-1
    for (i = n; i > 0; --i) // chuyen i goi tin tren kenh j
      Tinhp(i,j);
  // Xu li buoc cuoi, kenh m
  SoGoi[n][m] = 0;
  for (k = 1; k <= n; ++k) // thu chuyen k goi theo kenh m
    if (p[n] > p[n-k] + c[k][m]) {
      p[n] = p[n-k] + c[k][m];
      SoGoi[n][m] = k;
    }
}
void Print(){
  int i,j;
  cout << endl << n << " goi tin, " << m << " kenh";
  for (i = 1; i <= n; ++i) {
      cout << endl;
      for (j = 1; j <= m; ++j)
        cout << " " << c[i][j];
  }
}
void Doc(){
   ifstream f(fn);
   memset(c,0,sizeof(c));
   f >> n >> m; // n goi tin, m kenh
   int i,j;
   for (i = 1; i <= n; ++i)
     for (j = 1; j <= m; ++j)
       f >> c[i][j];
   f.close();
}
```

5.7 Mã BW

Olimpic Quốc tế (Bài dự tuyển)

Mã BW do Michael Burrows and David Wheeler đề xuất dùng để mã hóa xâu kí tự s thành cặp (u,d) như sau.

1. Quay xâu s qua trái mỗi lần 1 vị trí để thu được n xâu tính cả bản thân xâu s,
2. Sắp tăng các xâu thu được theo trật tự từ điển,
3. Lấy các kí cuối của các xâu được sắp ghép thành từ u,
4. Xác định d là vị trí xuất hiện của xâu s trong dãy được sắp.

Thí dụ, với s = "panama" ta có kết quả tại các bước như sau:

1. Sinh các xâu theo cách quay: "panama", "anamap", "namapa", "amapan", "mapana", "apanam".
2. Sắp các xâu: "amapan", "anamap", "apanam", "mapana", "namapa","panama".
3. u = "npmaaa",
4. d = 6.
Kết quả: "panama" được mã BW thành: ("npmaaa", 6).

Cho s, hãy tính (u,d) và biết (u,d), hãy xác định s. Chiều dài tối đa của s là 200.

Thuật toán

Ta gọi BW là thuật toán mã hóa và WB là thuật toán giải mã. Như vậy, với thí dụ đã cho ta có,
`BW("panama") = ("npmaaa",6)` và **`WB("npmaaa",6) = "panama"`**.

Bài toán xuôi BW chỉ có một điểm khó là sinh ra n xâu, tính cả xâu nguồn và sắp tăng các xâu đó. Với xâu nguồn s có chiều dài tối đa là 200 thì sẽ đòi hỏi miền nhớ 40000 byte để có thể lưu trữ các xâu. Ta sẽ triển khai thuật toán với chỉ 1 xâu nguồn s duy nhất. Giả sử chiều dài của xâu s là n. Với mỗi i = 1..n ta kí hiệu [i] là "*xâu vòng*" bắt đầu tính từ kí tự s[i] đến hết xâu, tức là đến kí tự s[n] rồi lại tính tiếp đến s[1] và kết thúc tại s[i–1]. Tóm lại, xâu vòng [i] là xâu sinh ra khi ta duyệt xâu nguồn s theo vòng tròn kể từ vị trí i qua phải. Mỗi xâu dài n sẽ có đúng n xâu vòng và mỗi xâu vòng có đúng n kí tự. Thí dụ, với s[1..6] = "panama" thì [2] = "anamap" là một xâu vòng. Ta dễ dàng sắp xếp các xâu vòng và ghi nhận trật tự trong một mảng chỉ dẫn id. Với xâu đã cho, sau khi sắp tăng theo chỉ dẫn ta thu được:

id[1] = 4 – xâu vòng [4] = "amapan" đứng đầu tiên trong dãy sắp tăng,
id[2] = 2 – xâu vòng [2] = "anamap" đứng thứ hai trong dãy sắp tăng,
id[3] = 6 – xâu vòng [6] = "apanam" đứng thứ ba trong dãy sắp tăng,
id[4] = 5 – xâu vòng [5] = "mapana" đứng thứ tư trong dãy sắp tăng.
id[5] = 3 – xâu vòng [3] = "namapa" đứng thứ năm trong dãy sắp tăng.
id[6] = 1 – xâu vòng [1] = "panama" đứng cuối cùng, thứ sáu, trong dãy sắp tăng.

Dễ thấy, nếu [i] là xâu vòng thì kí tự cuối của xâu này sẽ là kí tự sát trái kí tự thứ i trong s, cụ thể là s[(i+(n–1)) % n] nếu s biểu diễn trong C++ với chỉ số tính từ 0 và là s[(i–1+(n–1)) mod n + 1] nếu s biểu diễn trong Pascal với chỉ số tính từ 1.

Khi sắp xếp ta gọi hàm Sanh(s,i,j) để so sánh hai xâu vòng [i] và [j] của s. Hàm cho ra giá trị 0 nếu hai xâu bằng nhau, và 1 nếu xâu vòng [i] lớn hơn xâu vòng [j], –1 nếu xâu vòng [i] nhỏ thua xâu vòng [j].

Hoạt động của hàm Sanh 2 xâu vòng [i] và [j]
1. Duyệt n lần các phần tử s[i] và s[j] theo vòng tròn. Xét: - Nếu s[i] > s[j]: return 1; - Nếu s[i] < s[j]: return -1; 2. return 0; 3. end.

Bài toán ngược, WB khôi phục xâu nguồn s từ xâu mã u và giá trị d được triển khai như sau. Trước hết ta để ý rằng xâu u là một hoán vị của các kí tự trong xâu s. Cũng do các xâu vòng được sắp tăng nên sau khi gọi hàm BS để sắp tăng xâu u theo chỉ dẫn id ta dễ dàng tính được xâu s như sau. Ta coi u như là cột chứa các kí tự cuối cùng của các xâu vòng trong dãy được sắp và u' là xâu sắp tăng của xâu u. Do dãy các xâu vòng được sắp tăng nên cột đầu tiên của dãy các xâu này cũng phải được sắp tăng. Vậy cột đó chính là u'. Xét kí tự u[i]. Đây là kí tự cuối của một xâu vòng v nào đó trong dãy được sắp. Vậy trước khi quay, xâu v sẽ nhận u[i] làm kí tự đầu tiên, nghĩa là kí tự u[i] sẽ thuộc và là kí tự j nào đó trong xâu được sắp u', ta có u[i] = u'[j], hay là i được đặt tương ứng với j. Nhờ tương ứng này ta có thể khôi phục xâu s Kí tự này thuộc cả xâu u'. Nếu kí tự cuối trong dãy này là u[j] thi trước khi quay trái một vị trí, kí tự u[j] phải nằm trong cột đầu tiên được sắp.

```
 (* Pascal *)
procedure WB(var u: string; d: integer; var s: string);
var i: integer;
begin
```

```
  n := length(u);
  BS(u); { sắp tăng xâu u theo chỉ dẫn }
  s := ''; d := id[d];
  for i := 1 to n do
  begin
    s := s + u[d]; d := id[d];
  end;
end;

// DevC++
void WB(char *u, int d, char * s) { // (u,d) ==> v
   n = int(strlen(u));
   BS(u); // sắp tăng xâu u theo chỉ dẫn
   int i;
   d = id[d-1];
   for (i = 0; i < n; ++i) {
      s[i] = u[d]; d = id[d];
   }
   s[n] = '\0';
}
```

Bạn phải chọn phương thức sắp xếp không xáo trộn các phần tử bằng nhau mà chỉ tịnh tiến các phần tử đó. Các phương pháp sắp xếp như quick sort, min sort đều vi phạm tiêu chí này. Chương trình dưới đây sử dụng phương pháp sắp nổi bọt (bubble sort).

```
(* BW.PAS *)
uses crt;
const bl = #32; nl = #13#10;
var
n: integer;
id: array[0..256] of integer;
procedure iswap(i, j: integer);
var t: integer;
begin
 t := id[i]; id[i] := id[j]; id[j] := t;
end;
function Sanh(var s: string; 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;
    i := (i mod n) + 1;
    j := (j mod n) + 1;
  end;
  Sanh := 0;
end;
// sap tang cac xau vong cua s theo chi dan
procedure BubbleSort(var s: string);
 var i,j: integer;
begin
 for i := 1 to n do id[i] := i;
 for i := 1 to n-1 do
  for j := n downto i+1 do
    if Sanh(s,id[j],id[j-1]) < 0 then iswap(j,j-1);
end;
procedure BW(var s: string; var u: string; var d: integer);
var i: integer;
begin
 n := length(s);
```

```
 BubbleSort(s);
 u := '';
 for i := 1 to n do
 begin
   u := u + s[(id[i]-1+n-1) mod n + 1];
   if id[i] = 1 then d := i;
 end;
end;
procedure BS(var u: string);
var i,j: integer;
begin
 for i := 1 to n do id[i] := i;
 for i := 1 to n-1 do
  for j := n downto i+1 do
    if u[id[j]] < u[id[j-1]] then iswap(j,j-1);
end;
procedure WB(var u: string; d: integer; var s: string);
var i: integer;
begin
  n := length(u);
  BS(u);
  s := ''; d := id[d];
  for i := 1 to n do
  begin
    s := s + u[d]; d := id[d];
  end;
end;
procedure Test;
 var s, u, v: string;
     d: integer;
begin
  s := 'panama';
  BW(s,u,d);
  writeln(s,' => (',u,',',d,')');
  WB(u,d,v);
  writeln('(',u,',',d,') => ',v);
end;
BEGIN
 Test;
 readln;
END.

// DevC++: Bw.cpp
#include <iostream>
using namespace std;
// D A T A  A N D  V A R I A B L E
const char * fn = "bw.inp";
const char * gn = "bw.out";
const int mn = 256;
char s[mn];
char u[mn];
char v[mn];
int id[mn];
int n,d; // n = len(s); d: vi tri cua s trong day duoc sap
// P R O T O T Y P E S
// BW(s) = (u,d)
// WB(u,d) = v ( = x)
void BW(char * s, char * u, int & d);
void WB(char * u, int d, char * s);
int Sanh(char * s, int i, int j);
void BubbleSort(char * s);
void BS(char * u); // BubbleSort tren u
void iswap(int i , int j ); // doi cho id[i], id[j]
```

```
// I M P L E M E N T A T I O N
int main() {
    strcpy(s,"panama");
    BW(s,u,d);
    cout << endl << "BW(" << s << ") = (" << u
         << ", " << d << ")" << endl;
    WB(u,d,v);
    cout << endl << "WB(" << u << ", " << d << ") = " << v;
    cout << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
void BW(char * s, char * u, int & d) { // s ==> (u,d)
   n = int(strlen(s));
   BubbleSort(s);
   for (int i = 0; i < n; ++i) {
       u[i] = s[(id[i] + n - 1) % n];
       if (id[i] == 0) d = i+1;
   }
   u[n] = '\0';
}
void BubbleSort(char * s){
// sap tang cac xau vong cua s theo chi dan
     int i,j, n1 = n - 1;
     for (i = 0; i < n; ++i) id[i] = i;
     for (i = 0; i < n1; ++i) {
          for (j = n1; j > i; --j)
            if (Sanh(s,id[j],id[j-1]) < 0) iswap(j,j-1);
     }
}
int Sanh(char * s, int i, int j) { // sanh 2 xau vong [i] va [j]
   int k;
   for (k = 0; k < n; ++k, i = (i+1)%n, j = (j+1)%n)
      if (s[i] != s[j]) return (s[i] > s[j]) ? 1 : -1;
   return 0;
}
void iswap(int i, int j) { int t = id[i]; id[i]= id[j]; id[j] = t; }
void WB(char *u, int d, char * s) { // (u,d) ==> v
   n = int(strlen(u));
   BS(u);
   int i;
   d = id[d-1];
   for (i = 0; i < n; ++i) {
      s[i] = u[d]; d = id[d];
   }
   s[n] = '\0';
}
void BS(char *u){ // sap tang xau u theo chi dan
     int i,j,n1 = n-1;
     for (i = 0; i < n; ++i) id[i] = i;
     for (i = 0; i < n1 ; ++i) {
       for (j = n1; j > i; --j)
         if (u[id[j]] < u[id[j-1]]) iswap(j,j-1);
     }
}
```

5.8 Tam giác Pascal

Tam giác Pascal bậc n là một bảng gồm n dòng, dòng thứ i là một dãy gồm i+1 số tự nhiên được xây dựng như sau.

Dòng i = 1 chứa hai số 1, 1.

Gọi $x = (x_1, x_2, ..., x_{i+1})$ *là dòng thứ i, thì dòng thứ i+1,* $y = (y_1, y_2, ..., y_{i+2})$ *được xây dựng như sau:*

$y_1 = y_{i+2} = 1$,

$y_j = x_{j-1} + x_j$, $j = 2, 3, ..., i+1$.

Các phần tử của dãy được gọi là <u>hệ số</u>. Hai phần tử đầu và cuối của mỗi dãy luôn luôn mang giá trị 1 và được gọi là hai <u>hệ số ngoài</u>. Các phần tử còn lại được gọi là các <u>hệ số trong</u>.

Blaise Pascal *Nhà vật lý học, toán học và triết gia người Pháp. Ông được tiếp thu nền giáo dục từ người cha. Ngay từ thời trẻ Pascal đã nổi tiếng là thần đồng. Các tác phẩm đầu tiên của Pascal là về tự nhiên và các khoa học ứng dụng, nơi ông đã có những đóng góp quan trọng vào việc xây dựng một máy tính cơ khí, các nghiên cứu về chất lỏng, trình bày các khái niệm về áp suất và chân không bằng việc khái quát tác phẩm của Evangelista Torricelli.* *Pascal là một trong những người đặt nền móng cho hai lĩnh vực nghiên cứu mới của toán học. Ông đã viết một luận án quan trọng về hình học xạ ảnh ở độ tuổi 16. Cùng với Pierre de Fermat xây dựng lý thuyết xác suất. Đây là công trình có ảnh hưởng lớn tới sự phát triển của kinh tế học hiện đại và các khoa học xã hội.* *Sau đó ông dành tâm sức vào triết học và thần học với hai tác phẩm nổi tiếng trong thời kỳ đó.* *Nguồn: Wikipedia*	 ***Blaise Pascal*** 19/6/1623-19/8/1662

Tam giác Pascal (PT – Pascal's Triangle) có nhiều ứng dụng nổi tiếng. Dòng thứ n trong PT chính là các hệ số trong dạng khai triển của đa thức $(a+b)^n$. *Định lý sau đây cho ta một dấu hiệu nhận biết số nguyên tố dựa trên PT.*

Định lý: *n là số nguyên tố khi và chỉ khi mọi hệ số trong của dòng n trong tam giác Pascal chia hết cho n.*

Chúng ta thử giải hai bài toán sau đây liên quan đến tam giác Pascal.

PT1: Với mỗi số tự nhiên n hãy tính các hệ số của tam giác Pascal và tổng của chúng.

PT2: Kí hiệu a(i,n) là hệ số thứ i trên dòng thứ n của m giác Pascal. Hãy tính tổng

$\alpha(n) = sum\ \{\ a(i, i/2+1) \mid i = 1..n\ \} = a(1,1) + a(2,2) + a(3,2) + ... + a(i, i/2+1) + ... + a(n, n/2+1)$

trong đó x/y cho ta phần nguyên của thương trong phép chia số tự nhiên x cho số tự nhiên y. Giới hạn của n là $1 \le n \le 20$.

Thí dụ, với n = 6 ta có

PT1: Dòng 6 của tam giác Pascal: (1, 6, 15, 20, 15, 1); Tổng là: 64.

PT2: $\alpha(6) = \underline{1} + \underline{2} + \underline{3} + \underline{6} + \underline{10} + \underline{20} = 42$ (các số gạch dưới trong bảng).

n		***Ứng dụng***
1 2 3 4 5 6	**<u>1</u> 1** **1 <u>2</u> 1** **1 <u>3</u> 3 1** **1 <u>4</u> <u>6</u> 4 1** **1 5 <u>10</u> 10 5 1** **1 6 15 <u>20</u> 15 6 1** . . .	***Dòng 4:*** $(a+b)^4 = 1a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + 1b^4$. ***Dòng 5****: Mọi hệ số trong chia hết cho 5, vậy n = 5 nguyên tố.* ***Dòng 6****: Hệ số 15 không chia hết cho 6, vậy 6 không phải nguyên tố.*

Thuật toán

Hệ số thứ i trên dòng n, p(n, i) của PT chính là tổ hợp chặp i–1 của n phần tử do đó được tính theo công thức:

$$p(n,i) = C_n^{i-1} = n!/(i-1)!(n-i+1)! = (n-i+2)...n/(i-1)!, \qquad i = 1..\ n+1.$$

Ta qui ước 0! = 1, do đó

$$p(n, 1) = p(n, n+1) = C_n^0 = C_n^n = 1$$

Biết p(n, i) ta tính được p(n, i+1) theo công thức sau:

$$p(n, i+1) = p(n, i).(n-i+1)/i \qquad (*)$$

Ta khởi trị a[1] = 1 ứng với p(n, 1). Ngoài ra ta sử dụng hai biến phụ tu và mau với giá trị khởi đầu là tu = n, mau = 1. Sau đó, với mỗi i = 2..n+1 ta tính p(n, i) = a[i] = a[i-1]*tu/mau và giảm tu, tăng mau 1 đơn vị. Thủ tục PT dưới đây tính và hiển thị dòng thứ n trong tam giác Pascal:

```
(* Passcal *)
procedure PT(n: integer);
var x: longint;
    i, tu, mau: integer;
begin
  x := 1; tu := n; mau := 1;
  for i := 2 to n+1 do
  begin
    write(x,bl); { bl là dấu cách, bl = #32 }
    x := (x * tu) div mau;
    dec(tu); inc(mau);
  end;
  write(1);
end;

// DevC++
void PT(int n) {
   int x = 1;
   int i, n1 = n+1, tu = n, mau = 1;
   for (i = 2 ; i <= n1; ++i, --tu, ++mau) {
      cout << x << " ";
      x = x*tu/mau;
   }
   cout << 1;
}
```

Để tính tổng các hệ số trên dòng thứ n của PT, T(n), ta xét dạng khai triển của đa thức $(a+b)^n$:

$$(a+b)^n = p(n, 1)a^n + p(n, 2)a^{n-1}b + ... + p(n, i)a^{n-i+1}b^{i-1} + ... + p(n, n+1)b^n$$

Thay $a = b = 1$ vào biểu thức trên ta thu được:

$$T(n) = (1+1)^n = 2^n = p(n, 1) + p(n, 2) + ... + p(n, i) + ... + p(n, n+1)$$

Vậy $T(n) = 2^n$. Do các số nguyên trong máy tính được biểu diễn dưới dạng nhị phân nên hàm T(n) nhận giá trị là số 1 dịch qua trái n bit:

```
(* Passcal *)
function T(n: integer): longint;
begin T := longint(1) shl n; end;

// DevCPP
int T(int n) {  return (1 << n); }
```

Để tính hàm $\alpha(n)$ trước hết ta cần tính được p(i, j) với i = 1..n và j = i/2 + 1. Ta gọi x là giá trị của p(i,i/2+1) tính được tại bước i. Ta có nhận xét sau đây:

- Khi i = 1 ta có x = p(1,1) = 1;
- Nếu i là số chẵn thì giá trị của x được tăng gấp đôi: x = 2 *x.
- Nếu i là số lẻ thì x = x + x', trong đó x là giá trị tính được ở bước sát trước, bước i–1, x = p(i–1,v), v = (i–1)/2; x' là giá trị sát sau x tại bước i–1, x' = p(i–1, v+1). Sử dụng công thức (*) ta dễ dàng tính được x' từ x. Khi cài đặt ta sử dụng biến v để ghi nhận chỉ số của hệ số cần tính tại bước i. v được khởi trị 1, sau đó tăng thêm 1 nếu gặp dòng chẵn. Ta cũng sử dụng biến bool even để xác định dòng chẵn.

Thủ tục Test trong các chương trình dưới đây tính đồng thời với mỗi giá trị n = 1..20 các đại lượng sau đây:

- Dòng thứ n trong tam giác Pascal và tổng các hệ số trên dòng;
- Hàm α.

Để kết thúc chương trình bạn hãy bấm dấu chấm (.).
Độ phức tạp: O(n) cho cả hai thủ tục PT và Alpha**.**

Chương trình

```
(* Triangle.pas *)
uses crt;
const bl = #32; nl = #13#10;
procedure PT(n: integer);
var x: longint;
    i, tu, mau: integer;
begin
  x := 1; tu := n; mau := 1;
  for i := 2 to n+1 do
  begin
    write(x,bl);
    x := (x * tu) div mau;
    dec(tu); inc(mau);
  end;
  write(1);
end;
function Alpha(n: integer): longint;
 var x, sum, i, v: integer;
     even: Boolean;
 begin
  x := 1; sum := 1; even := false; v := 1;
  for i := 2 to n do
  begin
     even := not even;
     if (even) then
     begin
       inc(v); x := x * 2;
     end
     else x := x + x * (i-v) div v;
     sum := sum + x;
  end;
  Alpha := sum;
 end;
procedure Test;
 var n: integer;
begin
  for n := 1 to 20 do
  begin
    writeln(nl, ' n = ',n);
    write(' a = '); PT(n);
    writeln(nl, ' Tong = ', longint(1) shl n);
    writeln(' Alpha = ',Alpha(n));
    write(nl,' Bam . de ket thuc: ');
    if readkey = '.' then exit;
  end;
```

```
end;
BEGIN
 Test;
END.

// DevC++: Triangle.cpp
#include <iostream>
using namespace std;
// P R O T O T Y P E S
void PT(int);
int Alpha(int );
void Test();
// I M P L E M E N T A T I O N
int main() {
    Test();
    return 0;
}
void Test(){
    int n;
    for (n = 1; n <= 20; ++n) {
        cout << endl << " n = " << n << ":";
        cout << endl << "a = "; PT(n);
        cout << endl << " Tong = " << (1 << n);
        cout << endl << " Alpha = " << Alpha(n) << "  ";
        cout << endl << endl << " Bam . de ket thuc: ";
        if (cin.get() == '.') return;
    }
}
// p(i, v+1) = p(i, v).(i-v+1)/v, v = i/2 + 1
int Alpha(int n) {
  int sum = 1, x = 1, v = 1, i;
  bool even = false;
   for (i = 2 ; i <= n; ++i) {
     even = !even;
     if (even) { // i chan
       v++; x *= 2;
     }  else  x += x*(i-v)/v;
     sum += x;
   }
  return sum;
}
void PT(int n) {
   int x = 1;
   int i, n1 = n+1, tu = n, mau = 1;
   for (i = 2 ; i <= n1; ++i, --tu, ++mau) {
      cout << x << " ";
      x = x*tu/mau;
   }
   cout << 1;
}
```

5.9 Sơn mô hình

Moscow

Bạn Minh làm một mô hình như sau: Trên một tấm bảng nhựa hình chữ nhật chia lưới kích thước n × m đơn vị Minh dùng keo gắn tại một số ô của bảng một cột gỗ vuông cạnh đáy 1 đơn vị, chiều cao là một số nguyên. Các cột gỗ kề nhau cũng được gắn keo giữa hai mặt tiếp giáp. Sau đó Minh sơn các mặt gỗ tiếp xúc với không khí. Tính diện tích cần sơn.

son.inp	son.out	***Giải thích***
2 3 1 2 0 5 4 5	49	*file son.inp: dòng đầu tiên: 2 số nguyên dương n và m.* *Phần tử thứ j trên dòng i trong số n dòng tiếp theo là chiều cao của cột đặt tại ô (i, j).* *file son.out: diện tích cần sơn.*

Thuật toán

Bài này khá dễ nhưng hầu hết các bạn đều qnuên sơn mặt trên. Với mỗi cột ta tính diện tích cần sơn trên 4 mặt cột đó nếu tại mặt đó phần tường cao hơn cột bên cạnh. Ta viết hàm Hieu(x,y) cho ra hiệu x−y nếu x > y, ngược lại hàm cho ra giá trị 0. Như vậy hàm này cho ra diện tích cần sơn trên phần lộ của bức tường cao x và bức tường cao y kề nó (x > y). Sau đó, nếu ô nào có cột (độ cao lớn hơn 0) bạn nhớ cộng thêm diện tích sơn mái là 1 đơn vị. Khi đọc dữ liệu vào mảng bạn nên điền các ô viền phia ngoài bảng giá trị 0.

Độ phức tạp: O(n.m) vì ta duyệt mỗi ô 1 lần.

Chương trình

```
(* son.pas *)
uses crt;
const mn = 101; bl = #32; nl = #13#10;
      fn = 'son.inp'; gn = 'son.out';
var n, m: integer;
    a: array[0..mn,0..mn] of integer;
procedure Doc;
 var i,j: integer;
     f: text;
 begin
   assign(f,fn); reset(f);
   readln(f, n, m);
   fillchar(a, sizeof(a), 0);
   for i := 1 to n do
    for j := 1 to m do
      read(f,a[i,j]);
   close(f);
 end;
procedure Ghi(d: longint);
  var g: text;
begin
  assign(g,gn); rewrite(g);
  writeln(g,d); close(g);
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(a[i,j], bl);
  writeln;
 end;
end;
function Hieu(x, y: integer): integer;
begin
  if x > y then Hieu := x - y else Hieu := 0;
end;
function Son: longint;
var d: longint;
    i, j: integer;
begin
  d := 0;
  for i := 1 to n do
   for j := 1 to m do
     begin
     d := d + Hieu(a[i,j], a[i+1,j]) + Hieu(a[i,j], a[i-1,j]) +
              Hieu(a[i,j], a[i,j+1]) + Hieu(a[i,j], a[i,j-1]);
     if a[i,j] > 0 then inc(d);
     end;
  Son := d;
```

```
end;
BEGIN
  Doc; Print; Ghi(Son);
  readln;
END.
```

```
  // DevC++: son.cpp
  #include <iostream>
  #include <fstream>
  using namespace std;
  // D A T A   A N D  V A R I A B L E S
  const char * fn = "son.inp";
  const char * gn = "son.out";
  const int mn = 102;
  int a[mn][mn];
  int n,m;
  // P R O T O T Y P E S
  void Doc();
  int Son();
  int Hieu(int, int);
  void Print(); // xem du lieu
  void Ghi(int);
  // I M P L E M E N T A T I O N
  int main() {
      Doc(); Ghi(Son());
      cout << endl;
      system("PAUSE");
      return EXIT_SUCCESS;
  }
  void Print() { // Hiển thị bảng a
    int i,j;
    cout << endl << " n = " << n << "   m = " << m;
    for (i = 0; i <= n+1; ++i) {
        cout << endl;
        for (j = 0; j <= m+1; ++j)
           cout << a[i][j] << " ";
    }
  }
  void Doc() {
     memset(a,0,sizeof(a));
     ifstream f(fn);
     f >> n >> m;
     int i,j;
     for (i = 1; i <= n; ++i)
       for (j = 1; j <= m; ++j)
         f >> a[i][j];
     f.close();
     Print();
  }
  void Ghi(int d) {
    ofstream g(gn);
    g << d;
    g.close();
  }
  int Son() {
    int i,j, d = 0; // d: dien tich can son
    for (i = 1; i <= n; ++i)
      for (j = 1; j <= m;++j) {
         d += Hieu(a[i][j],a[i][j+1])+Hieu(a[i][j],a[i][j-1])
              + Hieu(a[i][j],a[i-1][j]) + Hieu(a[i][j],a[i+1][j]);
         if (a[i][j] > 0) ++d;
      }
     return d;
```

```
}
int Hieu(int x, int y) { return (x > y) ? (x - y) : 0;  }
```

5.10 Nhúng mô hình

Bạn Minh làm một mô hình như sau: Trên một tấm bảng nhựa hình chữ nhật chia lưới kích thước n × m đơn vị Minh dùng keo gắn tại một số ô của bảng một cột nhựa vuông cạnh đáy 1 đơn vị, chiều cao là một số nguyên. Các cột kề nhau cũng được gắn keo giữa hai mặt và hai cạnh tiếp giáp. Sau đó Minh nhúng mô hình ngập vào nước rồi cẩn thận nhấc lên. Tính thể tích của khối nước đọng trong mô hình.

model.inp	model.out	***Giải thích***
4 5 5 5 4 5 5 5 4 0 3 0 5 0 5 2 5 5 5 4 5 5	8	*file model.inp: dòng đầu tiên: 2 số nguyên dương n và m.* *Phần tử thứ j trên dòng i trong số n dòng tiếp theo là chiều cao của cột đặt tại ô (i, j).* *file model.out: thể tích nước đọng trong mô hình.*

Thuật toán

Gọi a là tấm bảng nền mô hình với kích thước n×m và a(i,j) là chiều cao của cột nhựa vuông đặt trên ô (i,j). Thoạt tiên ta giả sử chiều cao tối đa của các cột là 1, như vậy các ô trên nền nhà chỉ chứa các giá trị 0 hoặc 1.

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

Nền tầng 1 với các ô đặc mang giá trị 1, ô chứa nước đọng màu trắng (0) và ô thoát nước màu xám (0).

Có 5 ô trắng bị giam nên thể tích nước đọng sẽ là 5.

Các ô liên thông với ô trống trên biên sẽ tạo thành nột cống thoát nước ra ngoài mô hình.

0	ô thoát nước
0	ô nước đọng
1	ô đặc

Ô mang giá trị 0 chính là mặt sàn, còn ô mang giá trị 1 chính là cột. Bây giờ bạn thử rót nước vào mô hình đơn giản này cho ngập các ô và quan sát điều gì sẽ xảy ra. Dễ thấy là chỉ có các ô trống bị giam tại giữa mô hình là còn chứa nước. Có 5 ô trống bị giam nên thể tích nước đọng sẽ là 5. Các ô trống (mang giá trị 0) liên thông cạnh với một ô trống trên biên sẽ tạo thành một cống thoát nước ra ngoài mô hình. Nhận xét này cho ta thuật toán tính lượng nước đọng tại tầng nền (tầng 0) của mô hình như sau:

Duyệt đường biên, nếu gặp ô trống (trên biên và chứa trị 0) thì gọi thủ tục loang đánh dấu các ô này bằng giá trị 1.

Duyệt các ô lọt trong mô hình, đếm các ô trống (chứa giá trị 0) và đồng thời đánh dấu các ô này bằng giá trị 1. Đó là thể tich nước đọng.

Tiếp theo, sau khi đã duyệt và tính lượng nước đọng tại tầng nền (tầng 0), bạn dễ dàng suy ra cách tính lượng nước đọng tại tầng 1 nếu như coi các ô trống đã duyệt tại tầng 0 đã được lấp bằng các khối nhựa chiều cao 1. Ta gọi phương thức này là *đóng băng các ô đã xử lí*. Tổng quát, tại tầng h ta thực hiện thủ tục tương tự như tại tầng nền với chút điều chỉnh là thay giá trị đánh dấu bằng h+1 thay vì bằng 1. Gọi chiều cao tối đa của các cột là hmax, cho h biến thiên từ 0 đến hmax−1 ta tính được tổng khối nước đọng của mô hình.

Để thực hiên thuật toán loang bạn nhớ khởi trị mọi ô của nền nhà là −1.

Thủ tục Loang(i,j,h)
1. Xét trị của ô (i,j):

Nếu a(i,j) = h thì

1.1 Thay a(i,j) bằng h+1;

1.2 Loang tiếp sang 4 ô kề: (i,j+1), (i,j-1), (i+1,j) và (i-1,j)

2. end.

Độ phức tạp

Mỗi tầng ta duyệt n.m ô vậy tổng số ô phải duyệt vào cỡ hmax.n.m.

Chương trình

```
(* Model.Pas *)
uses crt;
const mn = 101; bl = #32; nl = #13#10;
      fn = 'model.inp'; gn ='model.out';
var
 n, m: integer;
 a: array[0..mn, 0..mn] of integer;
 hmax: integer;
procedure Doc;
 var f: text;
    i,j: integer;
begin
 assign(f,fn); reset(f);
 fillchar(a,sizeof(a),255); // gan -1
 readln(f,n,m); hmax := 0;
 for i := 1 to n do
   for j := 1 to m do
   begin
     read(f,a[i,j]);
     if hmax < a[i,j] then hmax := a[i,j];
   end;
 close(f);
end;
procedure Ghi(v: integer);
var g: text;
begin
  assign(g,gn); rewrite(g);
  writeln(g,v); close(g);
end;
procedure Loang(i,j,h: integer);
begin
 if a[i,j] <> h then exit;
 a[i,j] := h+1;
 Loang(i+1,j,h); Loang(i-1,j,h);
 Loang(i,j+1,h); Loang(i,j-1,h);
end;
function Tang(h: integer): longint;
 var i, j: integer;
     v: longint;
begin
 for i := 1 to n do { canh trai, phai }
 begin
   Loang(i,1,h); Loang(i,m,h);
 end;
 for j := 2 to m-1 do { canh tren, duoi }
 begin
   Loang(1,j,h); Loang(n,j,h);
 end;
 v := 0; { Duyet trong }
 for i := 2 to n-1 do
```

```
  for  j := 2 to m-1 do
    if a[i,j] = h then
     begin
      inc(v);
      a[i,j] := h+1;
     end;
 Tang := v;
end;
function TheTich: longint;
 var h: integer;
     v: longint;
begin
  v := 0;
  for h := 0 to hmax-1 do  v := v + Tang(h);
  TheTich := v;
end;
BEGIN
 Doc; Ghi(TheTich);
 readln;
END.
```

```
  // DevC++: Model.cpp
  #include <iostream>
  #include <fstream>
  using namespace std;
  // D A T A   A N D  V A R I A B L E S
  const char * fn = "model.inp";
  const char * gn = "model.out";
  const int mn= 102;
  int a[mn][mn];
  int n,m;
  int hmax;
  // P R O T O T Y P E S
  void Doc();
  void Ghi(int);
  int TheTich();
  int Tang(int);
  void Loang(int, int, int);
  // I M P L E M E N T A T I O N
  int main(){
      Doc(); Ghi(TheTich());
      cout << endl;
      system("PAUSE");
      return EXIT_SUCCESS;
  }
  void Doc(){
     memset(a,0xff,sizeof(a));
     ifstream f(fn);
     f >> n >> m; hmax = 0;
     int i,j;
     for (i = 1; i <= n; ++i)
       for (j = 1; j <= m; ++j){
         f >> a[i][j];
         if (a[i][j] > hmax) hmax = a[i][j];
       }
     f.close();
  }
  void Ghi(int v) {
      ofstream g(gn);
      g << v;
      g.close();
  }
  void Loang(int i, int j, int h){
```

```
   if (a[i][j] != h) return;
   ++a[i][j];
   Loang(i,j+1,h);
   Loang(i,j-1,h);
   Loang(i+1,j,h);
   Loang(i-1,j,h);
}
int Tang(int h){ // The tich nuoc bi giam tai tang h
    // Loang ngoai
    int i,j;
    for (i = 1; i <= n; ++i){
       Loang(i,1,h); // bien trai
       Loang(i,m,h); // bien phai
    }
    for (j = 2; j < m; ++j){
       Loang(1,j,h); // bien tren
       Loang(n,j,h); // bien duoi
    }
    // Tong khoi nuoc bi giam ben trong
    int v = 0;
    for (i = 2; i < n; ++i)
      for (j = 2; j < m; ++j)
        if (a[i][j] == h) {
           v++; ++a[i][j];
        }
    return v;
}
int TheTich(){
  int h, v = 0; // v: The tich nuoc bi giam
  for (h = 0; h < hmax; ++h)
    v += Tang(h);
   return v;
} // end
```

5.11 Số sát sau nhị phân

Cho số tự nhiên x trong dạng thập phân có tối đa 200 chữ số. Tìm số tự nhiên y đầu tiên lớn hơn x và ở dạng nhị phân x và y có cùng số chữ số 1.

Dữ liệu vào: file văn bản binnext.inp:

Dòng đầu tiên: N – số chữ số của x

Dòng thứ hai: số x dạng thập phân với các chữ số viết liền nhau.

Dữ liệu ra: file văn bản binnext.out:

Dòng đầu tiên: M – số chữ số của y

Dòng thứ hai: số y dạng thập phân với các chữ số viết liền nhau.

binnext.inp	binnext.out	***Giải thích***
2 22	2 25	*Trong dạng nhị phân số **22** có dạng **10110** và chứa 3 bit 1. Các số 23, 24 và **25** có dạng nhị phân lần lượt là 10111, 11000, **11001**. Vậy **25** là số sát sau số 22 chứa đúng 3 bit 1 trong dạng nhị phân.*

Thuật toán

Trước hết ta phác thảo khung bài giải dưới dạng

Thuật toán BinNext
1. Đọc số x từ input file; 2. ToBin(x,y): Chuyển số x sang dạng nhị phân và ghi vào biến y; 3. Next(y): Tìm số nhị phân sát sau số y và có cùng số bit

1 với y; Kết quả ghi ngay trong y; 4. ToDec(y,z): Chuyển số nhị phân y sang dạng thập phân; Kết quả ghi trong z; 5. Ghi số z vào output file; 6. end.

Thuật toán ToBin(x , y)	Để chuyển một số x sang dạng biểu diễn nhị phân y ta chia liên tiếp x cho 2 và ghi các số dư theo trật tự ngược tức là từ bit thấp đến bit cao.

`i := 0;`
`repeat`
`  Đặt bit y[i] := x%2;`
`  i := i + 1;`
`  x := x / 2;`
`until x = 0;`
`return y;`	

Thuật toán ToDec(y , x)	Để chuyển một số nhị phân y sang dạng biểu diễn thập phân x ta sử dụng thuật toán Horne duyệt ngược các bit từ cao đến thấp: nhân liên tiếp x với 2, cộng thêm bit vừa duyệt.

`x := 0;`
`Duyệt (các bit y[i] từ cao đến thấp)`
`  x := x*2 + y[i];`
`return x;`	

Vì x là số lớn nên ta cần biểu diễn chúng dưới dạng mảng. Ta sử dụng mảng nguyên x[0..n] để biểu diễn một số (nhị phân hay thập phân) chiều dài n, trong đó phần tử x[0] được dùng để lưu chiều dài n, tức là ta đặt x[0] := n. Ta đã biết, trong hệ đếm a, số x có $\log_a(x)+1$ chữ số do đó khi chuyển đổi x sang hệ đếm b sẽ có $\log_b(x) + 1$ chữ số. Theo định nghĩa của logarit ta có $\log_b(x) = \log_a(x).\log_b(a)$. Với a = 10, b = 2 ta có:

$$\log_2(x) = \log_2(10).\log_{10}(x) \le 4.\log_{10}(x)$$

Như vậy, với số thập phân x có tối đa 200 chữ số thì khi chuyển đổi x qua dạng nhị phân sẽ có tối đa 800 chữ số.

Để ý rằng, khi làm các phép toán số học, các kết quả có thể lớn dần, tức là số chữ số của kết quả phát triển dần về bên trái (hàng cao), do đó ta nên lưu các chữ số của chúng theo chiều ngược, từ hàng đơn vị trở đi. Thí dụ, số 157 sẽ được biểu diễn trong mảng x như sau: x[0..3] = (**3**,7,5,1), trong đó x[0] = **3** là số chữ số của x. Với cách biểu diễn này, khi số chữ số tăng lên thì chúng sẽ được bổ sung vào bên phải mảng, do đó ta không phải dịch chuyển các chữ số.

Dưới đây sẽ giải trình một số thủ tục.

`Div2(x)`: Thực hiện phép chia nguyên số x cho 2, x := x / 2. Để chia nguyên một số lớn x cho 2 ta chia nguyên lần lượt từng chữ số của x cho 2, tính từ chữ số hàng đơn. Nếu gặp chữ số lẻ thì ta cộng thêm 5 vào chữ số kết quả thu được tại bước trước đó. Cuối cùng ta bỏ bớt các chữ số 0 ở hàng cao. Thí dụ, để thực hiện phép chia nguyên số x = 157 cho 2 ta lưu ý rằng x được biểu diễn ngược như sau x[0..3] = (**3**, 7, 5, 1). Khi đó

Xét x[1] = 7: x[1] := x[1] / 2 = 7 / 2 = 3;

Xét x[2] = 5:

Vì x[2] = 5 là số lẻ nên ta chỉnh lại x[1] := x[1] + 5 = 3 + 5 = 8;

x[2] := x[2] / 2 = 5 / 2 = 2;

Xét x[3] = 1:

Vì x[3] = 1 là số lẻ nên ta chỉnh lại x[2] := x[2] + 5 = 2 + 5 = 7;

x[3] := x[3] / 2 = 1 / 2 = 0.

Ta thu được x[0..3] = (**3**, 8, 7, 0). Sau khi bỏ số 0 ở đầu phải (hàng cao) ta thu được x[0..2] = (**2**, 8, 7). Điều này có nghĩa 157 / 2 = 78.

`Next(x)` Xác định số nhị phân sát sau x và có cùng số bit 1 như x, kết quả ghi tại x. Tình huống này được gọi là *xử lí tại chỗ*. Trường hợp duy nhất vô nghiệm là khi x = 0. Thủ tục này thực hiện như sau:

Hàm Next(x): Sửa số nhị phân x thành số nhị phân sát sau x và có cùng số bit 1 như x
1. Nếu x = 0: Gi kết quả vô nghiệm; Thóat khỏi thủ tục; 2. Duyệt x từ bit thấp (i = 1) trở đi, tìm bit i đầu tiên nhận giá trị 1, x[i] = 1. 3. Duyệt tiếp từ i+1 tìm bít j đầu tiên nhận giá trị 0, x[j] = 0; 4. Lật hai bit i và j: x[i] = 0; x[j] = 1; 5. Đảo lại đoạn x[1..j–1]. 6. end.

Thí dụ, với x = 100110 ta có dạng biểu diễn ngược của x là x[0..6] = (**6**,0,1,1,0,0,1). Ta có:

i = 2, x[i] = x[2] = 1;

j = 4, x[j] = x[4] = 0;

Đặt lại x[2] = 0; x[4] = 1 ta thu được x[0..6] = (**6**,0,0,1,1,0,1);

Lật đoạn x[1..3] ta thu được x[0..6] = (**6**,1,0,0,1,0,1).

Vậy 101001 là số sát sau số x = 100110 và có cùng 3 bít 1 như x.

`MultPlus(x,c)` Thực hiện việc tính x = 2*x + c. Thủ tục này dùng trong thuật toán ToDec(x,y) – chuyển đổi số nhị phân x sang dạng thập phân y. Thủ tục hoạt động như sau: duyệt các chữ số của x từ hàng đơn trở đi. Với mỗi chữ số x[i] ta tính x[i] := (2*x[i] + c) mod 10. Ta coi c như là số nhớ của phép nhân, do đó ta cần tính lại c = (2*x[i] + c) div 10.

Nhờ thủ tục này, khi c = 0 ta tính được ngay thủ tục x = x*2 với lời gọi **`MultPlus(x,0).`**

`bool IsZero(char * x)`: Kiểm tra số lớn x = 0?

`bool Odd(char c)`: Kiểm tra số lớn x là số lẻ?

```
(* Binnext.pas *)
uses crt;
const fn = 'binnext.inp'; gn = 'binnext.out';
      mn = 801; bl = #32; nl = #13#10;
type mi1 = array[0..mn] of integer;
var x,y: mi1;
    n: integer;
(* Ghi file *)
procedure Save(var x: mi1; fn: string);
 var g: text; i: integer;
begin
   assign(g,gn); rewrite(g);
   writeln(g,x[0]); { x[0] – chiều dài số x }
   for i := x[0] downto 1 do write(g,x[i]); { Ghi ngược }
   close(g);
end;
function IsZero(var x: mi1): Boolean;
 begin IsZero := (x[0] = 1) and (x[1] = 0) end;
{ so nhi phan sat sau x }
function Next(var x: mi1): Boolean;
 var i, j, k: integer;
begin
   Next := false;
   if (IsZero(x)) then exit;
  { tim so 1 dau tien }
   i := 1;
   while x[i] = 0 do inc(i);
   { x[i] = 1 }
   { tim so 0 dau tien sau i }
   j := i+1;
   while x[j] = 1 do inc(j);
   if (j > x[0]) then
   begin
     inc(x[0]); { them 1 vi tri }
     j := x[0];
   end;
   x[i] := 0; x[j] := 1;
```

```
   i := 1; j := j-1;
   while (i < j) do
   begin
     k := x[i]; x[i] := x[j]; x[j] := k;
     inc(i); dec(j);
   end;
   Next := true;
end;
procedure MultPlus(var x: mi1; c: integer); { x := 2*x + c }
 var i: integer;
begin
   for i := 1 to x[0] do
   begin
      c := 2*x[i] + c; { c la so nho }
      x[i] := c mod 10;
      c := c div 10;
   end;
   if (c > 0) then begin inc(x[0]); x[x[0]] := c; end;
end;
procedure ToDec(var x,y: mi1);
 var i: integer;
begin
   y[0] := 1; y[1] := 0; { Khoi tri y = 0 }
   for i := x[0] downto 1 do
      MultPlus(y,x[i]);
end;
procedure Div2(var x: mi1); { x := x div 2 }
 var i: integer;
begin
   x[1] := x[1] div 2;
   for i := 2 to x[0] do
   begin
      if Odd(x[i]) then inc(x[i-1],5);
      x[i] := x[i] div 2;
   end;
   i := x[0];
   while (x[i] = 0) do dec(i);
   if i = 0 then i := 1;
   x[0] := i;
end;
procedure ToBin(var x,y: mi1);
 var i: integer;
begin
   i := 0;
   repeat
     inc(i);
     if Odd(x[1]) then y[i] := 1 else y[i] := 0;
     Div2(x);
   until IsZero(x);
   y[0] := i;
end;
procedure Init(var x: mi1; fileName: string);
 var i: integer; c: char; f: text;
begin
 fillchar(x,sizeof(x),0);
 assign(f,fn); reset(f);
 readln(f,x[0]);
 writeln(x[0]);
 for i := x[0] downto 1 do
 begin
    read(f,c); x[i] := ord(c)-ord('0');
    write(x[i]);
 end;
```

```
 close(f);
end;
procedure Print(var x: mi1);
 var i: integer;
begin
   for i := x[0] downto 1 do write(x[i]);
end;
procedure Run;
begin
   Init(x,fn); { Doc so x tu file fn }
   write(nl, ' x = '); Print(x);
   ToBin(x,y); { Chuyen x sang dang nhi phan y }
   write(nl, ' y = '); Print(y);
   if (Next(y)) then
   begin { So nhi phan sat sau y }
       write(nl, ' y = '); Print(y);
       ToDec(y,x); { Doi y sang x }
       write(nl, ' x = '); Print(x);
   end else { x = 0: vo nghiem }
   begin x[0] := 1; x[1] := 0; end;
   Save(x,gn);
end;
BEGIN
    Run;
    writeln(nl,' Fini '); readln;
END.
```

```
// DevC++: binnext.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 char * fn = "binnext.inp";
const char * gn = "binnext.out";
const int mn = 1001;
int x[mn], y[mn];
int n;
// P R O T O T Y P E S
void Init(int *, const char *);
void Save(int *, const char *);
void Print(int *);
bool Odd(int);
bool Even(int);
bool Odd(int *);
bool Even(int *);
void MultPlus(int *, int);
void Div2(int *);
void ToBin(int *, int *);
bool IsZero(int *);
void ToDec(int *, int *);
bool Next(int * x);
void Run();

// I M P L E M E N T A T I O N
int main() {
    Run();
    cout << endl << "  Fini "; cin.get();
    return 0;
}
void Run() {
```

```
   Init(x,fn); // Doc so x tu file fn
   cout << endl << " x = "; Print(x);
   ToBin(x,y); // Chuyen x sang dang nhi phan y
   cout << endl << " y = "; Print(y);
   if (Next(y)) { // So nhi phan sat sau y
       cout << endl << " y = "; Print(y);
       ToDec(y,x); // Doi y sang x
       cout << endl << " x = "; Print(x);
   }
    else { // vo nghiem: 0
           x[0] = 1; x[1] = 0;
         }
   Save(x,gn);
}
// Ghi file
void Save(int * x, const char * fileName) {
   ofstream g(gn);
   g << x[0] << endl;
   for (int i = x[0]; i > 0; --i) g<< x[i];
   g.close();
}
// so nhi phan sat sau x
bool Next(int *x) {
   int i, j, k;
   if (IsZero(x)) return false;
   // tim so 1 dau tien
   for (i = 1; i <= x[0]; ++i)
      if (x[i] == 1) break;
   // tim so 0 dau tien sau i
   for (j = i+1; j <= x[0]; ++j)
       if (x[j]==0) break;
   if (j > x[0]) ++x[0]; // them 1 chu so
   x[i] = 0; x[j] = 1;
    i = 1; --j;
    while (i < j) {
      k = x[i]; x[i] = x[j]; x[j] = k;
      ++i;--j;
    }
    return true;
}
bool IsZero(int * x) { return (x[0]==1 && x[1]==0); }
void ToDec(int *x, int *y){
   int i;
   y[0] = 1; y[1] = 0;
   for (i = x[0]; i > 0; --i)
      MultPlus(y,x[i]);
}
void ToBin(int * x, int *y) {
   int i = 0;
   do {
        y[++i] = Odd(x);
        Div2(x);
       } while (! IsZero(x));
   y[0] = i;
}
void Div2(int * x) {
   int i;
   x[1] /= 2;
   for (i = 2; i <= x[0]; ++i) {
      if (Odd(x[i])) x[i-1] += 5;
      x[i] /= 2;
   }
   i = x[0];
```

```
   while (x[i] == 0 && i > 1) --i;
   x[0] = i;
}
// x = x*2 + c
void MultPlus(int * x, int c) {
   int i;
   for (i = 1; i <= x[0]; ++i) {
      c = 2*x[i] + c; // c la so nho
      x[i] = c % 10;
      c /= 10;
   }
   if (c > 0) { ++x[0]; x[x[0]] = c; }
}
bool Odd(int c) { return (c%2) == 1; }
bool Even(int c) { return !Odd(c); }
bool Odd(int * x) { return (x[1]%2) == 1; }
bool Even(int * x) { return !Odd(x[1]);  }
void Init(int *x, const char *fileName) {
 int i;
 char c;
 memset(x,0,sizeof(x));
 ifstream f(fileName);
 f >> x[0]; cout << endl << x[0] << endl;
 for (i = x[0]; i > 0; --i) {
    f >> c; x[i] = c - '0'; cout << x[i];
  }
 f.close();
}
void Print(int * x) {
   for (int i = x[0]; i > 0; --i) cout << x[i];
}
```

5.12 Hàm f(n)

Tính gía trị của hàm f(n) với biến số nguyên n cho trước, $0 \le n \le 1.000.000.000$ (1 tỷ). Biết
f(0) = 0; f(1) = 1; f(2n) = f(n); f(2n+1) = f(n) + f(n+1).

Thuật toán

Xét hàm 3 biến *g(n,a,b) = af(n) + bf(n+1)*. Ta có,

1) *g(n,1,0) = 1.f(n) + 0.f(n+1) = f(n).*

2) *g(0,a,b) = af(0) + bf(1) = a.0 + b.1 = b*.

3) *g(2n,a,b) = af(2n) + bf(2n+1) = af(n) + bf(n) + bf(n+1) = (a+b)f(n) + bf(n+1) = g(n,a+b,b).*

4) *g(2n+1,a,b) = af(2n+1)+bf(2n+2) = af(n) + af(n+1) + bf(2(n+1)) = af(n) + af(n+1) + bf(n+1) = af(n) + (a+b)f(n+1) = g(n,a,a+b).*

Từ bốn tính chất trên ta thiết kế được hàm *f(n)* như sau:

Để tính *f(n)* ta tính *g(n,a,b)* với *a = 1, b = 0*. Để tính *g(n)* ta lặp đến khi *n = 0*. Nếu *n* chẵn ta gọi hàm *g(n/2,a+b,b)*; ngược lại, nếu *n* lẻ ta gọi hàm *g(n/2,a,a+b)*. Khi *n = 0* ta thu được *f = g(0,a,b) = b*.

```
(* Pascal *)
function f(n: longint): longint;
var a,b: longint;
begin
   a := 1; b := 0;
   while (n <> 0) do
   begin
     if odd(n) then b := b + a else a := a + b;
     n := n div 2;
   end;
   f := b;
end;

// DevC++
```

```
int f(int n) {
   int a = 1, b = 0;
   while (n) {
         if (n % 2 == 0) a += b;
         else b += a;
         n /= 2;
   }
   return b;
}
```

Độ phức tạp

$\log_2(n)$ vòng lặp.

Dữ liệu test

$f(100) = 7$; $f(101) = 19$; $f(1000000) = 191$; $f(1000000000) = 7623$.

5.13 Hàm h(n)

Tính hàm h(n) với giá trị n cho trước $0 \le n \le 1.000.000$ (1 triệu). Biết
$h(0) = 3$; $h(1) = 1$; $h(2n) = 2h(n)$; $h(2n+1) = h(n) – 2h(n+1)$.

Thuật toán

Xét hàm 3 biến $g(n,a,b) = ah(n) + bh(n+1)$. Ta có,

1) $g(n,1,0) = h(n)$.

2) $g(0,a,b) = 3a + b$.

3) $g(2n,a,b) = g(n,2a+b,–2b)$.

4) $g(2n+1,a,b) = g(n,a,2b–2a)$.

Dữ liệu test

$h(100) = –176$; $h(101) = 128$; $h(1000000) = 3162112$; $h(1000001) = –1933312$.

5.14 Rhythm

Viết hàm Rhythm(s) cho ra dáng điệu của xâu kí tự $s = (s_1, s_2, \ldots, s_n)$ như sau

- *Rhythm(s) = 1, nếu các kí tự trong s đều bằng nhau: $s_1 = s_2 = \ldots = s_n$,*
- *Rhythm(s) = 2, nếu các kí tự trong s tạo thành dãy tăng chặt: $s_1 < s_2 < \ldots < s_n$,*
- *Rhythm(s) = 3, nếu các kí tự trong s tạo thành dãy đồng biến: $s_1 \le s_2 \le \ldots \le s_n$,*
- *Rhythm(s) = 4, nếu các kí tự trong s tạo thành dãy giảm chặt: $s_1 > s_2 > \ldots > s_n$,*
- *Rhythm(s) = 5, nếu các kí tự trong s tạo thành dãy nghịch biến: $s_1 \ge s_2 \ge \ldots \ge s_n$,*
- *Rhythm(s) = 0, nếu không xảy ra các tình huống trên,*

Kết quả được chọn ưu tiên cho các giá trị nhỏ. Thí dụ, Rhythm("aaaaa") = 1, trong khi các tình huống 3 và 5 cũng thỏa. Biết $2 \le n \le 255$.

Thuật toán

Ta sử dụng 3 biến đếm b (bằng), t (tăng), g (giảm) và duyệt tuần tự xâu s để ghi nhận các tình huống khi di chuyển từ phần tử s_i sang phần tử tiếp theo s_{i+1}. Cụ thể là ta gán

b = 1 nếu $s_i = s_{i+1}$,
t = 1 nếu $s_i < s_{i+1}$,
g = 1 nếu $s_i > s_{i+1}$.

Giá trị của hàm Rhythm chính là số nguyên có dạng biểu diễn nhị phân (g,t,b). Cụ thể là với 5 trường hợp đầu ta có

*Rhythm = 4*g + 2*t + b*

Hai trường hợp cuối ứng với các trị 6 và 7 được chuyển về 0.
Tóm lại, ta có

*Rhythm = ((4*g + 2*t + b) mod 7) mod 6*

g	t	b	Rhythm
0	0	1	**1**
0	1	0	**2**
0	1	1	**3**
1	0	0	**4**
1	0	1	**5**
1	1	0	**6 → 0**
1	1	1	**7 → 0**

Độ phức tạp. cỡ n.

```
(* Rhythm.pas *)
uses crt;
function Rhythm(s: string): integer;
var i, g, t, b: integer;
begin
  g := 0; t := 0; b := 0;
  for i := 2 to length(s) do
    if s[i] = s[i-1] then b := 1
     else if s[i] > s[i-1] then t := 1
      else { s[i] < s[i-1] } g := 1;
  Rhythm := ((4*g + 2*t + b) mod 7 mod 6);
end;
BEGIN
 writeln(Rhythm('aabccdx')); { 3 }
 readln;
END.
```

```
// Rhythm.cpp
#include <string.h>
#include <fstream>
#include <iostream>
#include <stdio.h>
using namespace std;
// P R O T O T Y P E S
int Rhythm(char *);
// I M P L E M E N T A T I O N
int main() {
     cout << endl << Rhythm("aabccdxxxz") << endl; // 3
     cout << endl << " Fini ";
     cin.get();
     return 0;
}
int Rhythm(char * s) {
   int g, t, b, i, n;
   g = t = b = 0; n = strlen(s);
   for (i = 1; i < n; ++i)
     if (s[i] == s[i-1]) b = 1;
       else if (s[i] > s[i-1]) t = 1;
         else g = 1;
   return ((4*g + 2*t + b) % 7) % 6;
}
```

5.15 Cóc

Một chú cóc máy có thể nhảy k bước với độ dài khác nhau (b_1, b_2, ..., b_k) trên đoạn đường thẳng. Đặt cóc trên đoạn đường thẳng tại vạch xuất phát 0. Cho biết số cách nhảy để cóc đến được điểm N.
Thí dụ, số bước k = 2, b_1 = 2, b_2 = 3, đoạn đường dài N = 8.
Có 4 cách: (2,2,2,2) (2, 3, 3) (3, 3, 2) (3, 2, 3).

Thuật toán: Quy hoạch động.

Gọi S(n) là số cách để cóc vượt đoạn đường dài n. Dựa theo bước nhảy đầu tiên ta chia toàn bộ các phương án nhảy của cóc thành k nhóm không giao nhau. Như vậy,

- Nhóm 1 sẽ gồm các phương án bắt đầu bằng bước nhảy độ dài b_1, tức là gồm các phương án dạng (b_1,...). Sau bước nhảy đầu tiên, cóc vượt đoạn đường b_1, đoạn đường còn lại sẽ là $n–b_1$, do đó tổng số phương án của nhóm này sẽ là $S(n–b_1)$.
- Nhóm 2 sẽ gồm các phương án bắt đầu bằng bước nhảy độ dài b_2, tức là gồm các phương án dạng (b_2,...). Sau bước nhảy đầu tiên, cóc vượt đoạn đường b_2, đoạn đường còn lại sẽ là $n–b_2$, do đó tổng số phương án của nhóm này sẽ là $S(n–b_2)$.
- ...
- Nhóm i sẽ gồm các phương án bắt đầu bằng bước nhảy độ dài b_i, tức là gồm các phương án dạng (b_i,...). Sau bước nhảy đầu tiên, cóc vượt đoạn đường b_i, đoạn đường còn lại sẽ là $n–b_i$, do đó tổng số phương án của nhóm này sẽ là $S(n–b_i)$.
- ...
- Nhóm k sẽ gồm các phương án bắt đầu bằng bước nhảy độ dài b_k, tức là gồm các phương án dạng (b_k,...). Sau bước nhảy đầu tiên cóc vượt đoạn đường b_k, đoạn đường còn lại sẽ là $n–b_k$, do đó tổng số phương án của nhóm này sẽ là $S(n–b_k)$.

Dĩ nhiên, bước đầu tiên là b_i sẽ được chọn nếu $b_i \le n$.
Vậy ta có,

$$S(n) = \Sigma\{ S(n–b_i) \mid n \ge b_i \; i = 1, 2, \dots , k \} \qquad (*)$$

Để ý rằng khi đoạn đường cần vượt có chiều dài n = 0 thì cóc có 1 cách nhảy (là đứng yên), do đó S(0) = 1. Ngoài ra ta quy định S(n) = 0 nếu n < 0.

Sử dụng mảng s ta tính dần các trị s[0] = 1; s[1], s[2], ..., s[n] theo hệ thức (*) nói trên. Kết quả được lưu trong s[n].

Độ phức tạp Với mỗi giá trị *n* ta tính *k* bước nhảy, vậy độ phức tạp thời gian cỡ *k.n*.

Các chương trình dưới đây đọc dữ liệu từ input file "coc.inp" gồm các số k, n và độ dài các bước nhảy b_i, i = 1, 2, ..., k. Kết quả được hiển thị trên màn hình.

```
(* Coc.pas *)
uses crt;
const
 fn = 'coc.inp';
 maxk = 10; maxn = 50;
type mi1 = array[0..maxk] of integer;
     ml1 = array[0..maxn] of longint;
var
 k, n: integer;
 b: mi1;
 s: ml1;
procedure  Doc;
 var f: text; i: integer;
begin
   assign(f,fn); reset(f);
   read(f,k, n);
   for i := 1 to k do read(f,b[i]);
   close(f);
end;
procedure TinhS(d: integer);
 var i: integer; v: longint;
begin
  v := 0;
  for i := 1 to k do
      if (d >= b[i]) then  v := v + s[d-b[i]];
   s[d] := v;
end;
function XuLi: longint;
 var d: integer;
```

```
begin
   s[0] := 1;
   for d := 1 to n do TinhS(d);
   XuLi := s[n];
end;
BEGIN
    Doc;
    writeln(#13#10, XuLi);
    readln;
END.
```

```
// DevC++: Coc.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 char BL = ' ';
int b[10];
int s[50];
int n, k;
void Doc();
int XuLi();
void TinhS(int);
int main(){
    Doc();
    cout << endl << XuLi() << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
void Doc() {
   ifstream f("Coc.inp");
   f >> k >> n;
   for (int i = 1; i <= k; ++i) f >> b[i];
}
int XuLi() {
   s[0] = 1;
   for (int d = 1; d <= n; ++d)
     TinhS(d);
   return s[n];
}
void TinhS(int d) {
     int v = 0;
     for (int j = 1; j <= k; ++j)
        if (d >= b[j]) v += s[d-b[j]];
   s[d] = v;
}
```

Chú ý

Nếu ta sắp tăng mảng b thì chỉ cần duyệt đến khi b[i] > d.

5.16 Trả tiền

Ucraina

Ngân hàng có m loại tiền giấy mệnh giá khác nhau $b_1 = 1, b_2, ..., b_k$ với số lượng không hạn chế. Cần chọn ít nhất là bao nhiêu tờ tiền để có tổng bằng t cho trước.

Thí dụ. Có m = 4 loại tiền mệnh giá lần lượt là 1, 2, 7 và 10 quan tiền. Cần trả lại t = 14 quan. Ta chọn 2 tờ tiền mệnh giá 7 quan.

Thuật toán

Gọi s(t) là số tờ tiền ít nhất có tổng bằng t. Nếu trong số này có tờ tiền mệnh giá b_i thì tổng số còn lại là $t–b_i$ và do đó số tờ tiền còn phải trả nốt là $s(t–b_i)$. Phương án tối ưu khi đó sẽ là

$$s(t) = \min \{ s(t-b_i)+1 \mid i = 1, 2, \dots , m; b_i \le t \}$$

Để cài đặt ta dùng mảng s và tính lần lượt các giá trị s[1], s[2], ... , s[t]. Kết quả được hiển thị trên màn hình là s[t].

Độ phức tạp: tm.

Chú ý Điều kiện có loại tiền mệnh giá 1 với số lượng không hạn chế đảm bảo rằng bài toán luôn luôn có nghiệm. Thật vậy, trong trường hợp xấu nhất ta có thể dùng t tờ tiền mệnh giá 1 quan để trả lại.

```
(* money.pas *)
uses crt;
const fn = 'money.inp'; bl = #32; nl = #13#10;
var b: array[0..31] of integer;
    s: array[0..1001] of integer;
    m,t: integer;
procedure Doc;
var f: text; i: integer;
begin
  assign(f,fn); reset(f);
  read(f,m,t);
  writeln(nl,m,bl,t);
  for i := 1 to m do
  begin
    read(f,b[i]);
    write(b[i],bl);
  end;
  close(f);
end;
function Min(a,b: integer): integer;
begin if a < b then Min := a else Min := b end;
procedure Tinhs(d: integer);
 var i: integer;
begin
  s[d] := d;
  for i := 2 to m do
    if (d >= b[i]) then s[d] := Min(s[d],s[d-b[i]]+1);
end;
function XuLi: integer;
 var i: integer;
begin
 for i := 1 to t do Tinhs(i);
 XuLi := s[t];
end;
BEGIN
  Doc;
  writeln(nl,XuLi);
  writeln(nl,' Fini'); readln;
END.

// DevC++: money.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 char * fn = "money.inp";
int b[31]; // cac loai menh gia
int s[1001];
int m, t;
// P R O T O T Y P E S
void Doc();
int XuLi();
void Tinhs(int);
```

```
// I M P L E M E N T A T I O N
int main() {
    Doc();
    cout << endl << XuLi() << endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
void Doc() {
  int i;
  ifstream f(fn);
  f >> m >> t ;
  cout << endl << m << " " << t << endl;
  for ( i = 1; i < m; ++i) {
    f >> b[i];
    cout << b[i] << " ";
  }
  f.close();
}
int XuLi() {
  for (int i = 1; i <= t; ++i) Tinhs(i);
  return s[t];
}
int Min(int a, int b) {
    return (a < b) ? a : b;
}
void Tinhs(int d) {
  s[d] = d;
  for (int i = 2; i <= m; ++i)
     if (d >= b[i]) s[d] = Min(s[d], s[d-b[i]]+1);
}
```

Chú ý

Nếu ta sắp tăng mảng b thì chỉ cần duyệt đến khi b[i] > d.

5.17 Game

Bulgaria

Đây là trò chơi một người tạm gọi là bạn Vova với hai dãy số nguyên dương: a gồm n > 1 phần tử và b gồm m > 1 phần tử. Vova phải thực hiện các thao tác sau đây:

- *Chia mỗi dãy a và b thành k ≥ 1 nhóm, mỗi nhóm gồm dãy các phần tử đứng liền nhau, số phần tử của mỗi nhóm có thể khác nhau nhưng tối thiểu phải là 1. Số k do Vova tự quyết định; Mã số các nhóm là 1, 2, ... , k;*
- *Với mỗi nhóm i trong dãy a Vova phải tính tổng* $c = c_1 + c_2 + \ldots + c_k$,

trong đó

$c_i = (s_i - u_i)(t_i - v_i)$, *i = 1, 2, …, k;*
s_i *là tổng các phần tử của nhóm i trong dãy a;*
u_i *là số phần tử của nhóm i trong dãy a;*
t_i *là tổng các phần tử của nhóm i trong dãy b;*
v_i *là số phần tử của nhóm i trong dãy b;*
Hãy cho biết giá trị min của c.

Dữ liệu vào: text file **VOVA.INP**
Dòng đầu tiên: 2 số n và m;
Tiếp đến là n phần tử của dãy a;
Tiếp đến là m phần tử của dãy b.
Các số cùng dòng cách nhau qua dấu cách.

Dữ liệu ra: text file **VOVA.OUT**
Chứa giá trị min c.

VOVA.INP	VOVA.OUT
3 2 3 7 4 5 2	17

Giải thích

Với thí dụ trên, Vova xét các phương án sau:

Phương án 1: Chọn k = 1. Khi đó mỗi dãy tự thạo thành một nhóm: a = (3,7,4); b = (5,2).

Thao tác trên dãy a: $s_1 = 3+7+4 = 14$; $u_1 = 3$;

Thao tác trên dãy b: $t_1 = 5+2 = 7$; $v_1 = 2$;

$c = c_1 = (s_1 - u_1)(t_1 - v_1) = (14-3)(7-2) = 11\times5 = \mathbf{55}$.

Phương án 2: Chọn k = 2. Khi đó mỗi dãy có đúng 2 nhóm: a = (3)(7,4); b = (5)(2) hoặc a = (3,7)(4); b = (5)(2) . Ta xét lần lươt từng phương án con 2.1 và 2.2.

Phương án 2.1: a = (3)(7,4); b = (5)(2)

Thao tác trên dãy a: $s_1 = 3$; $u_1 = 1$; $s_2 = 7+4 = 11$; $u_2 = 2$;

Thao tác trên dãy b: $t_1 = 5$; $v_1 = 1$; $t_2 = 2$; $v_2 = 1$;

$c_1 = (s_1 - u_1)(t_1 - v_1) = (3-1)(5-1) = 2\times4 = 8$;

$c_2 = (s_2 - u_2)(t_2 - v_2) = (11-2)(2-1) = 9\times1 = 9$;

$c = c_1 + c_2 = 8 + 9 = \mathbf{17}$.

Phương án 2.2: a = (3,7)(4); b = (5)(2)

Thao tác trên dãy a: $s_1 = 3+7 = 10$; $u_1 = 2$; $s_2 = 4$; $u_2 = 1$;

Thao tác trên dãy b: $t_1 = 5$; $v_1 = 1$; $t_2 = 2$; $v_2 = 1$;

$c_1 = (s_1 - u_1)(t_1 - v_1) = (10-2)(5-1) = 8\times4 = 32$;

$c_2 = (s_2 - u_2)(t_2 - v_2) = (4-1)(2-1) = 3\times1 = 3$;

$c = c_1 + c_2 = 32 + 3 = \mathbf{35}$.

Vậy cmin = min(**55**, **17**, **35**) = **17**.

Thuật toán

Nhận xét 1. Ta có thể giảm mọi phần tử của a và b xuống 1 đơn vị. Khi đó công thức tính c_i sẽ được rút gọn thành $c_i = s_i \times t_i$.

Nhận xét 2. Muốn đạt trị tối ưu cmin thì trong nhóm cuối cùng, nhóm thứ k của a và b không thể chứa đồng thời hơn 1 phần tử.

Thật vậy, gỉa sử hai nhóm cuối của dãy a và b, nhóm thứ k, đều chứa hơn 1 phần tử. Ta gọi phương án này là P_1. Ta xây dựng phương án P_2 gồm k+1 nhóm như sau. Tách hai nhóm cuối của a và b thành hai nhóm nhỏ và gọi tổng của các nhóm đó lần lượt là A+B cho nhóm của dãy a và C+D cho nhóm của dãy b. Khi đó (A+B)(C+D) = AC + AD + BC + BD > AC + BD. Từ đây suy ra phương án phân nhóm P_2 với k+1 nhóm sẽ cho kết quả nhỏ hơn phương án phân nhóm P_1 thành k nhóm như trong bảng dưới đây:

	Phương án P_1: k nhóm	Phương án P_2: k+1 nhóm
Dãy a	$s_1, s_2, \ldots, s_{k-1}$, (A+B)	$s_1, s_2, \ldots, s_{k-1}$, A, B
Dãy b	$t_1, t_2, \ldots, t_{k-1}$, (C+D)	$t_1, t_2, \ldots, t_{k-1}$, C, D

(A+B)(C+D) = AC + AD + BC + BD > AC + BD

Như vậy, để đạt trị cmin, nhóm thứ k của một trong 2 dãy a hoặc b phải chứa đúng 1 phần từ. Ta kí hiệu cho các tình huống này là:

- 1:1 – cả hai nhóm cuối của a và b đều có đúng 1 phần tử;
- 1:N – nhóm cuối của dãy a có đúng 1 phần tử , của dãy b có hơn 1 phần tử;

hoặc

- N:1 – nhóm cuối của dãy a có hơn 1 phần tử, của dãy b có đúng 1 phần tử.

Kí hiệu c(n,m) là giá trị cmin của bài toán với hai dãy a(1..n) và b(1..m).

Với tình huống 1:1 ta có hai nhóm cuối là (a_n) và (b_m), do đó

$$c(n,m) = c(n-1,m-1) + a_n b_m$$

Với tình huống 1:N ta giả sử có hai nhóm cuối là (a_n) và $(b_i, b_{i+1}, \ldots, b_{m-1}, b_m)$, $i < m$. Ta có

$$a_n(b_i + b_{i+1} + \ldots + b_{m-1} + b_m) = a_n(b_i + b_{i+1} + \ldots + b_{m-1}) + a_n b_m \text{ , do đó}$$

$$c(n,m) = c(n,m-1) + a_n b_m$$

Với tình huống N:1 ta giả sử có hai nhóm cuối là $(a_j, a_{j+1}, \ldots, a_{n-1}, a_n)$, $j < n$ và (b_m) . Xét tương tự như tình huống 1:N ta thu được

$$c(n,m) = c(n-1,m) + a_n b_m$$

Vậy

$$c(n,m) = \min \{ c(n-1,m-1) + a_n b_m,\ c(n,m-1) + a_n b_m,\ c(n-1,m) + a_n b_m \}, \text{ hay}$$

$$c(n,m) = \min \{ c(n-1,m-1),\ c(n,m-1),\ c(n-1,m) \} + a_n b_m$$

Khi n = 1, với mọi m ≥ 1, mỗi dãy tạo thành đúng 1 nhóm nên ta có

$$c(n,m) = a_1(b_1 + b_2 + \ldots + b_{m-1} + b_m).$$

Tương tự, khi m = 1, với mọi n ≥ 1, mỗi dãy tạo thành đúng 1 nhóm nên ta có

$$c(n,m) = (a_1 + a_2 + \ldots + a_{n-1} + a_n) b_1.$$

Để cài đặt ta dùng 2 mảng 1 chiều c và d.

Khi đọc dữ liệu bạn nhớ giảm đồng loạt 1 đơn vị cho các số trong hai dãy a và b.

Mảng c lúc đầu được khởi trị với n = 1, c[j] = a[1]*(b[1] + ... + b[j]), j = 1..m với ý nghiã dãy a có đúng 1 phần tử a[1] do đó lúc đầu chỉ có 2 nhóm (a[1]) và (b[1..j]).

Sau đó ta lặp n–1 lần mỗi lần lặp thứ i ta tính trị cho hàm c(i,j), j = 1..m và ghi vào mảng d với

$$d[1] = c[1] + a[i]*b[1]$$

$$d[j] = \min \{ c[j-1], c[j], d[j-1] \} + a[i]*b[j],\ j = 2..m$$

```
(* Game.pas *)
uses crt;
const mn = 201;
      fn = 'vova.inp'; gn = 'vova.out';
       bl = #32; nl = #13#10;
type mi1 = array[0..mn] of integer;
var a,b,c,d: mi1;
    n, m: integer;
function Min(a, b, c: integer): integer;
begin
  if a > b then a := b;
  if a < c then Min := a else Min := c;
end;
function Cmin: integer;
var i,j: integer;
begin
  fillchar(c,sizeof(c),0); d := c;
  { Khoi tri: 2 day a[1] , b[1..m] }
  for j := 1 to m do c[j] := c[j-1] + a[1]*b[j];
  for i := 2 to n do
  begin
    d[1] := c[1] + a[i]*b[1];
    for j := 2 to m do
        d[j] := Min(c[j-1], c[j],d[j-1]) + a[i]*b[j];
    c := d;
  end;
  Cmin := d[m];
end;
procedure Doc;
var i: integer;
f: text;
begin
  assign(f,fn); reset(f);
  read(f,n,m); writeln(nl,n, bl, m);
  for i := 1 to n do
  begin
   read(f,a[i]); dec(a[i]); write(a[i],bl);
  end;
  writeln;
  for i := 1 to m do
  begin
   read(f,b[i]); dec(b[i]); write(b[i],bl);
  end;
  close(f);
end;
procedure Ghi(v: integer);
 var g: text;
```

```
begin
  assign(g,gn); rewrite(g);
  writeln(g,v); close(g);
end;
BEGIN
  Doc;
  Ghi(Cmin);
  writeln(nl,' Fini');
  readln;
END.

// DevC++: game.cpp
#include <string.h>
#include <fstream>
#include <iostream>
using namespace std;
const int mn = 201;
const char * fn = "vova.inp";
const char * fn = "vova.out";
int a[mn], b[mn], c[mn], d[mn];
int n, m;
void Doc();
int Cmin();
int Min(int, int, int);
void Ghi(int);
main() {
  Doc(); Ghi(Cmin());
  cout << endl << " Fini "; cin.get();
  return 0;
}
int Min(int x, int y, int z) {
    if (x > y) x = y;
    return (x < z) ? x : z;
}
int Cmin() {
  int i,j;
  memset(c,0,sizeof(c)); memset(d,0,sizeof(d));
  for (j = 1; j <= m; ++j) c[j] = c[j-1] + a[1]*b[j];
  for (i = 2; i <= n; ++i) {
    d[1] = c[1] + a[i]*b[1];
    for (j = 2; j <= m; ++j)
        d[j] = Min(c[j-1], c[j],d[j-1]) + a[i]*b[j];
    memcpy(c,d,sizeof(d));
  }
   return d[m];
}
void Doc() {
  ifstream f(fn);
  f >> n >> m; cout << endl << n << " " << m << endl;
  int i;
  for (i = 1; i <= n; ++i) {
    f >> a[i]; --a[i];
    cout << a[i] << " ";
  }
  cout << endl;
  for (i = 1; i <= m; ++i) {
    f >> b[i]; --b[i];
    cout << b[i] << " ";
  }
  f.close();
}
void Ghi(int v) {
  ofstream g(gn);
```

```
  g << v;
  g.close();
}
```

5.18 Robots

Một khu nghiên cứu Robot có 3 phòng thí nghiệm là A, B và C bố trí trên một tuyến đường thẳng theo thứ tự đã nêu. Mỗi phòng đều có thể có 3 loại robot thông minh là robot mũ đỏ R, robot mũ xanh B và robot mũ xanh lá G. Người ta ra lệnh cho các robot phải tự bàn nhau để thực hiện nhiệm vụ sau đây: di chuyển về các phòng sao cho mỗi phòng chỉ có một loại robot và năng lượng chi phí cho việc di chuyển là ít nhất. Hãy cùng bàn với các robot để chọn một phương án tối ưu.

Dữ liệu vào: text file ROBOTS.INP

Dòng thứ nhất: hai số nguyên dương biểu thị khoảng cách AB và AC;

Dòng thứ hai: ba số nguyên dương r b g cho biết chi phí năng lượng của mỗi loại robot R, B và G để di chuyển 1 đơn vị khoảng cách;

Dòng thứ ba: ba số nguyên không âm cho biết số lượng robot mỗi loại R, B và G trong phòng A;

Dòng thứ tư: tương tự như dòng thứ ba, chứa thông tin về phòng B;

Dòng thứ năm: tương tự như dòng thứ ba, chứa thông tin về phòng C.

Dữ liệu ra: text file ROBOTS.OUT

Dòng thứ nhất: tổng chi phí năng lượng tối thiểu cho các robot di chuyển;

Dòng thứ hai: một hoán vị của ba chữ cái R, B, G viết liền nhau cho biết phương án tối ưu bố trí các robot vào các phòng.

Dữ liệu trên cùng dòng cách nhau qua dấu cách.

ROBOTS.INP	ROBOTS.OUT
1 3 3 1 2 2 1 2 1 4 3 2 2 1	40 RGB

Giải thích *Khoảng cách AB = 1; AC = 3 trên đường thẳng ABC;*
robot R cần 3, B cần 1 và G cần 2 đơn vị năng lượng để di chuyển 1 đơn vị khoảng cách;
Phòng A có 2 robot R, 1 robot B và 2 robot G;
Phòng B có 1 robot R, 4 robot B và 3 robot G;
Phòng C có 2 robot R, 2 robot B và 1 robot G.
Kết quả: *Nếu tập hợp các robot mũ đỏ R về phòng A, các robot mũ xanh lá G về phòng B và các robot mũ xanh B về phòng C thì chi phí năng lượng là tối thiểu và bằng 40.*

Thuật toán

Bài này khá dễ giải vì chỉ đòi hỏi khoảng vài chục lần tính toán đơn giản. Kí hiệu XYZ là một phương án bố trí robot loại X vào phòng A, loại Y vào phòng B và loại Z vào phòng C. Ta có tất cả 6 phương án là RBG, RGB, BRG, BGR, GRB và GBR. Ta tính chi phí cho mỗi phương án rồi chọn phương án với chi phí min. Ta sử dụng các hàm hỗ trợ sau đây:

`Move(r, i, j)` – chi phí di chuyển toàn bộ robot loại r hiện có trong phòng i sang phòng j ≠ i, i, j = A, B, C .

`PhuongAn(r1, r2, r3)` – chi phí cho phương án đưa các robot loại r1 về phòng A, loại r2 về phòng B và robot loại r3 về phòng C.

```
(* ROBOTS.PAS *)
uses crt;
const fn = 'ROBOTS.INP';  gn = 'ROBOTS.OUT';
      bl = #32; nl = #13#10; NN = 3;
      A = 1; B = 2; C = 3; { cac phong }
      RR = 1; RB = 2; RG = 3; { mau mu cua robots }
type mi1 = array[0..NN] of integer;
     mi2 = array[0..NN] of mi1;
var AB,AC: integer; { Khoảng cách }
    energy: mi1; { energy[i] - chi phí năng lượng của robot loại i }
    number: mi2;
{ number[i,j] - so luong robot loai j trong phong i }
procedure Doc;
var i,j: integer;
  f: text;
begin
```

```
  assign(f,fn); reset(f);
  read(f,AB,AC);
  write(nl,' Khoang cach: ',AB,bl,AC);
  write(nl,' Nang luong: ');
  for i := 1 to NN do
  begin read(f,energy[i]); write(energy[i],bl); end;
  writeln(nl, ' Phan bo: ');
  for i := 1 to NN do
  begin
    writeln;
    for j := 1 to NN do
    begin  read(f,number[i,j]); write(number[i,j],bl) end;
  end;
  close(f);
end;
(* Chi phi chuyen robot loai r tu phong i sang phong j *)
function Move(r, i, j: integer): integer;
 var e: integer;
begin
  e := energy[r]*number[i][r];
  case i of
   A: if (j = B) then e := e*AB
      else if (j = C) then e := e*AC else e := 0;
   B: if (j = A) then e := e*AB
      else if (j = C) then e := e*(AC-AB) else e := 0;
   C: if (j = A) then e := e*AC
      else if (j = B) then e := e*(AC-AB) else e := 0;
  end;
  Move := e;
end;
function PhuongAn(r1, r2, r3: integer): integer;
begin
  PhuongAn :=  Move(r1,B,A) + Move(r1,C,A)+
               Move(r2,A,B) + Move(r2,C,B)+
               Move(r3,A,C) + Move(r3,B,C);
end;
procedure XuLi;
var i, imin, p: integer; pa: array[1..6] of integer;
    g: text;
begin
  p := 1; { phuong an 1: RBG  }
  pa[p] :=  PhuongAn(RR,RB,RG);
  inc(p); { phuong an 2: RGB  }
  pa[p] :=  PhuongAn(RR,RG,RB);
   inc(p); { phuong an 3: BRG }
  pa[p] :=  PhuongAn(RB,RR,RG);
  inc(p); { phuong an 4: BGR }
  pa[p] :=  PhuongAn(RB,RG,RR);
  inc(p); { phuong an 5: GRB }
  pa[p] :=  PhuongAn(RG,RR,RB);
  inc(p); { phuong an 6: GBR }
  pa[p] :=  PhuongAn(RG,RB,RR);
  imin := 1;
  for i := 1 to 6 do
    if (pa[i] < pa[imin]) then imin := i;
  assign(g,gn); rewrite(g);
  writeln(g,pa[imin]);
  case imin of
     1: write(g,'RBG');
     2: write(g,'RGB');
     3: write(g,'BRG');
     4: write(g,'BGR');
     5: write(g,'GRB');
```

```
     6: write(g,'GBR');
  end;
  close(g);
end;
BEGIN
  Doc;
  XuLi;
  writeln(nl,' Fini'); readln;
END.


// DecC++: Robots.cpp
#include <fstream>
#include <string.h>
#include <iostream>
using namespace std;
const int A = 0; // Phong 0: A
const int B = 1; // Phong 1: B
const int C = 2; // Phong 2: C
const int RR = 0; // Red Robot
const int RB = 1; // Blue Robot
const int RG = 2; // Green Robot
const int NN = 3; // So loai Robot = 3
const char * fn = "robots.inp";
const char * gn = "robots.out";
const char bl = 32;
int energy[NN]; // chi phi nag luong
int number[NN][NN];
// numer[i][j] - so luong robot loai j trong phong i
int AB, AC; // Khoang cach AB, AC
int pa[6]; // cac phuong an
void Doc() {
  int i, j;
  ifstream f(fn);
  f >> AB >> AC;
  cout << endl << "Khoang cach: " << AB << bl << AC;
  cout << endl << " Nang luong: ";
  for (i = 0; i < NN; ++i) {
      f >> energy[i];
      cout << energy[i] << bl;
  }
  cout << endl << " Phan bo: ";
  for (i = 0; i < NN; ++i) {
    cout << endl;
    for (j = 0; j < NN; ++j) {
      f >> number[i][j];
      cout << number[i][j] << bl;
    }
 }
  f.close();
}
int Move(int r, int from, int to) {
    // chi phoi chuyen robot r tu phong i sang phong j
  int e = energy[r]*number[from][r];
  switch(from) {
   case A: if (to == B) return e*AB;
           else if (to == C) return e*AC;
   case B: if (to == A) return e*AB;
           else if (to == C) return e*(AC-AB);
   case C: if (to == A) return e*AC;
           else if (to == B) return e*(AC-AB);
  }
}
```

```
int PhuongAn(int r1, int r2, int r3) {
  return (Move(r1,B,A)+Move(r1,C,A)+
          Move(r2,A,B)+Move(r2,C,B)+
          Move(r3,A,C)+Move(r3,B,C));
}
void XuLi() {
  int p, i, imin;
  p = 0; // phuong an 0: RBG
  pa[p] =  PhuongAn(RR,RB,RG);
  cout << endl << "RBG: " << pa[p];
  ++p; // phuong an 1: RGB
  pa[p] =  PhuongAn(RR,RG,RB);
  cout << endl << "RGB: " << pa[p];
  ++p; // phuong an 2: BRG
  pa[p] =  PhuongAn(RB,RR,RG);
  cout << endl << "BRG: " << pa[p];
  ++p; // phuong an 3: BGR
  pa[p] =  PhuongAn(RB,RG,RR);
  cout << endl << "BGR: " << pa[p];
  ++p; // phuong an 4: GRB
  pa[p] =  PhuongAn(RG,RR,RB);
  cout << endl << "GRB: " << pa[p];
  ++p; // phuong an 5: GBR
  pa[p] =  PhuongAn(RG,RB,RR);
  cout << endl << "GBR: " << pa[p];
  imin = 0;
  for (i = 1; i < 6; ++i)
    if (pa[i] < pa[imin]) imin = i;
  ofstream g(gn);
  g << pa[imin] << endl;
  switch (imin) {
   case 0: g << "RBG"; break;
   case 1: g << "RGB"; break;
   case 2: g << "BRG"; break;
   case 3: g << "BGR"; break;
   case 4: g << "GRB"; break;
   case 5: g << "GBR"; break;
  }
  g.close();
}
main(){
  Doc();
  XuLi();
  cout << endl << " Fini"; cin.get();
  return 0;
}
```
