TỦ SÁCH TRI THỨC DUY TÂN

NGUYỄN XUÂN HUY

SÁNG TẠO TRONG THUẬT TOÁN VÀ LẬP TRÌNH

với ngôn ngữ Pascal và C#

Tập 2

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

MỤC LỤC .

Chương 1

Các bài toán về đoạn thẳng

Bạn cần chú ý đọc kĩ đề bài. Có những bài mới xem ta thấy tựa tựa như nhau nhưng kết quả là khác nhau. Điển hình là những bài tối ưu hóa, tức là những bài tìm max hay min của một hàm. Các ràng buộc chỉ khác nhau đôi chút nhưng độ khó sẽ thì lại khác xa nhau.

Bài 1.1 Đoạn rời 1

Cho N đoạn thẳng với các điểm đầu a_i và điểm cuối b_i là những số nguyên trong khoảng –1000..1000, $a_i < b_i$. Liệt kê số lượng tối đa K đoạn thẳng không giao nhau. Hai đoạn thẳng [a,b] *và* [c,d] *được coi là không giao nhau nếu xếp chúng trên cùng một trục số, chúng không có điểm chung. Điều kiện này đòi hỏi: b < c hoặc d < a.*

DOAN.INP	DOAN.OUT
8 2 3 4 5 10 12 13 15 1 9 2 5 6 8 7 15	5 1 2 7 3 4

Dữ liệu vào: tệp văn bản **DOAN.INP**

Dòng đầu tiên: số tự nhiên N, 1 < N ≤ 1000.

Dòng thứ i trong N dòng tiếp theo, mỗi dòng chứa hai số nguyên a_i b_i cách nhau qua dấu cách, biểu thị điểm đầu và điểm cuối của đoạn thứ i, i = 1..N.

Dữ liệu ra: tệp văn bản **DOAN.OUT**

Dòng đầu tiên: số tự nhiên K.

K dòng tiếp theo, mỗi dòng một số tự nhiên v thể hiện chỉ số của các đoạn rời nhau tìm được.

Thí dụ bên cho biết tối đa có 5 đoạn rời nhau là 1, 2, 7, 3 và 4.

Thuật toán

Phương pháp: *tham.*

1. Sắp các đoạn tăng theo đầu phải b.
2. Khởi trị: Lấy đoạn 1, đặt r = b_1 là đầu phải của đoạn này
3. Với mỗi đoạn j := 2..N tiếp theo xét:

   Nếu đầu trái của đoạn j, a_j > r thì lấy đoạn j đưa vào kết quả

   và chỉnh r là đầu phải của đoạn j, r := b_j.

Độ phức tạp: cỡ *N*log*N* chi phí cho quick sort.

(* Pascal *)

```
(*==========================================
  Doan roi 1: Liet ke toi da cac doan thang
              khong giao nhau
  ==========================================*)
program DoanRoi1;
uses crt;
const mn = 1001; bl = #32 {Dấu cách}; nl = #13#10 {Xuống dòng};
      fn = 'doan.inp'; gn = 'doan.out';
type { Mô tả một đoạn }
  KieuDoan = record
               a,b: integer;
               id: integer; { Chỉ số đoạn }
             end;
  md1 = array[0..mn] of KieuDoan;
  mi1 = array[0..mn] of integer;
var n,m,r: integer; { n - số lượng đoạn }
                    { m - số lượng đoạn được chọn }
                    { r - đầu phải đang duyệt     }
    d: md1; { các đoạn d[1..n] }
    f,g: text;
    c: mi1; { mảng chứa kết qủa }
procedure Doc;
 var i: integer;
begin
  assign(f,fn); reset(f); readln(f,n);
  for i := 1 to n do
   begin
     read(f,d[i].a,d[i].b); d[i].id := i;
   end;
  close(f);
end;
(*---------------------------------
    Sắp tăng các đoạn d[t..p] theo
    đầu phải b.
  ---------------------------------*)
procedure Qsort(t,p: integer);
var i,j,m: integer;
    x: KieuDoan;
begin
  i := t; j := p; m := d[(i + j) div 2].b;
  while (i <= j) do
   begin
    while (d[i].b < m) do i := i + 1;
    while (m < d[j].b) do j := j - 1;
    if (i <= j) then
     begin
       x := d[i]; d[i] := d[j]; d[j] := x;
       i := i + 1; j := j - 1;
     end;
   end;
   if (t < j) then Qsort(t,j);
   if (i < p) then Qsort(i,p);
end;
procedure XuLi;
var i: integer;
```

```
  begin
    m := 1; c[m] := 1; { Đưa đoạn 1 vào kết quả }
    r := d[m].b; { đầu phải của đoạn cuối trong kết quả }
    for i := 2 to n do
      if (r < d[i].a) then
        begin
          m := m + 1; c[m] := i; r := d[i].b;
        end;
  end;
  procedure Ghi;
    var i: integer;
  begin
    assign(g,gn); rewrite(g); writeln(g,m);
    for i := 1 to m do writeln(g,d[c[i]].id);
    close(g);
  end;
  BEGIN
    Doc; Qsort(1,n); XuLi; Ghi;
  END.
```

// C#

```
  using System;
  using System.IO;
  using System.Collections;
  /*================================================
   *  Doan Roi 1: Liệt kê tối đa k đoạn rời nhau *
   ================================================*/
  namespace SangTao2 {
      class DoanRoi1 {
          static public Doan[] d;
          static int n; // số đoạn
          static int m; // số đoạn được chọn cho kết quả
          static int[] c; // lưu kết quả
          const string fn = "doan.inp";
          const string gn = "doan.out";
          static void Main(string[] args) {
              Doc();  QSortB(d,0,n-1);
              XuLi(); Ghi(); XemKetQua();
              Console.WriteLine("\n Fini "); Console.ReadLine();
          }
          static public void Doc() {
              StreamReader f = File.OpenText(fn);
              string s = f.ReadToEnd(); f.Close();
              String[] ss = s.Split(
                            new char[] { ' ', '\n', '\r', '\t' },
                            StringSplitOptions.RemoveEmptyEntries);
              int[] a = Array.ConvertAll(ss,
                             new Converter<string, int>(int.Parse));
              n = a[0]; // so doan
              d = new Doan[n];
              int j = 1;
              for (int i = 0; i < n; ++i, j += 2) // đọc đoạn i
                  d[i] = new Doan(a[j], a[j + 1], i + 1);
          } // Doc
          static public void XuLi() {
              m = 0;
```

```
            c = new int[n];
            c[m++] = 0; // chọn đoạn 0
            int r = d[0].b; // thiết lập giới hạn phải
            for (int i = 1; i < n; ++i)
                if (r < d[i].a) { c[m++] = i; r = d[i].b; }
        } // XuLi
        // Sắp tăng các đoạn d[t..p] theo đầu phải b
        static public void QSortB(Doan[] d, int t, int p) {
            int i = t, j = p, m = d[(i + j) / 2].b;
            while (i <= j) {
                while (d[i].b < m) ++i;
                while (d[j].b > m) --j;
                if (i <= j) {
                    Doan x = d[i]; d[i] = d[j]; d[j] = x;
                    ++i; --j;
                }
            }
            if (t < j) QSortB(d, t, j);
            if (i < p) QSortB(d, i, p);
        }
        static public void Ghi() {
            StreamWriter g = File.CreateText(gn);
            g.WriteLine(m);
            for (int i = 0; i < m; ++i) g.WriteLine(d[c[i]].id);
            g.Close();
        }
        // Hiển thị lại các files input, output để kiểm tra
        static public void XemKetQua() {
            Console.WriteLine("\n Input " + fn);
            Console.WriteLine(File.ReadAllText(fn));
            Console.WriteLine("\n Output " + gn);
            Console.WriteLine(File.ReadAllText(gn));
        }
    }// DoanRoi1
    public struct Doan  { // Mô tả một đoạn
      public int a, b, id;
      public Doan(int x1,int x2,int z)  // Tạo đoạn mới
      { a = x1; b = x2; id = z; }
    }  // Doan
} //  SangTao2
```

Giải thích chương trình C#

1. Khai báo file text f, mở file tên **fn = "doan.inp"** để đọc toàn bộ dữ liệu vào biến **string s** rồi đóng file lại.

```
StreamReader f = File.OpenText(fn);
string s = f.ReadToEnd(); f.Close();
```

2. Tách **string s** thành mảng các **string ss[i]** theo các dấu ngăn cách khai báo trong **new char []**, loại bỏ các **string** rỗng.

Trong một dòng văn bản thường chứa các dấu ngăn cách sau đây (gọi là các *dấu trắng*)

' ' - dấu cách

'\n' - dấu hết dòng (dấu xuống dòng)

'\r' - dấu về đầu dòng (dấu **ENTER/RETURN**)

'\t' - dấu tab

```
string[] ss = s.Split(new char [] { ' ', '\n', '\r', '\t' },
                      StringSplitOptions.RemoveEmptyEntries);
```

3. Chuyển đổi mỗi **string ss[i]** thành số nguyên và ghi trong mảng nguyên **a**

```
int[] a = Array.ConvertAll(ss,
                        new Converter<string, int>(int.Parse));
```

Sau bước 3 dữ liệu trong file "**doan.inp**" được đọc vào mảng **a[0..n-1]**.

4. Lấy số lượng đoạn : **n = a[0]**;

5. Xin cấp phát **n** con trỏ kiểu **Doan** : **d = new Doan[n];**

6. Cấp phát và khởi trị cho mỗi đoạn **i = 0..n-1.** Đoạn i có chỉ số là i+1:

```
int j = 1;
for (int i = 0; i < n; ++i, j += 2) // doc doan i
      { d[i] = new Doan(a[j], a[j + 1], i + 1); }
```

Có nhiều phương thức đọc/ghi các text file. Bạn cần lựa chọn và ghi nhớ một phương thức mà bạn cảm thấy tiện lợi nhất.

7. Bạn có thể tổ chức dữ liệu **Doan** theo dạng **struct** (bản ghi) hoặc dạng **class** (lớp). Điểm khác nhau căn bản giữa hai cấu trúc này là, theo qui ước ngầm định **struct** được truyền theo trị (by **val**) còn **class** được truyền theo chỉ dẫn (by **ref**).

```
public struct Doan {
   public int a, b;  // diểm đầu và điểm cuối đoạn
   public int id;    // chỉ số đoạn
   // phương thức tạo đoạn
   public Doan(int x1, int x2, int z)
   { a = x1; b = x2; id = z; }
}
```

Bài 1.2 Đoạn gối 1

Cho N đoạn thẳng trên trục số với các điểm đầu a_i và điểm cuối b_i là những số nguyên trong khoảng –1000..1000, $a_i < b_i$. Hãy tìm số lượng tối đa K đoạn thẳng gối nhau liên tiếp. Hai đoạn thẳng [*a*,*b*] *và* [*c*,*d*] *được gọi là gối nhau nếu xếp chúng trên cùng một trục số thì điểm đầu đoạn này trùng với điểm cuối của đoạn kia, tức là c = b hoặc d = a.*

DOAN.INP	DOAN.OUT
5 2 7 1 3 7 9 3 4 4 5	3

Dữ liệu vào: tệp văn bản **DOAN.INP:** *xem Đoạn gối 1*

Dữ liệu ra: tệp văn bản **DOAN.OUT**

chứa duy nhất một số tự nhiên K.

Thí dụ này cho biết có tối đa 3 đoạn gối nhau liên tiếp là [1,3], [3,4] và [4,5].

Thuật toán

Phương pháp: *Quy hoạch động + Tham.*

Giả sử các đoạn được sắp tăng theo đầu phải b. Kí hiệu c(i) là số lượng tối đa các đoạn thẳng gối nhau tạo thành một dãy nhận đoạn i làm phần tử cuối dãy (khi khảo sát các đoạn từ 1..i). Ta có

c(1) =1,

Với i = 2...N: c(i) = max { c(j) | $1 \le j < i$, Đoạn j kề trước đoạn i: $b_j = a_i$ } + 1.

Lợi dụng các đoạn sắp tăng theo đầu phải b, với mỗi đoạn i ta chỉ cần duyệt ngược các đoạn j đứng trước đoạn i cho đến khi phát hiện bất đẳng thức $b_j < a_i$.

Kết quả: K = max { c(i) | i = 1...n }.

Độ phức tạp: cỡ N^2 vì với mỗi đoạn ta phải duyệt tối đa tất cả các đoạn đứng trước đoạn đó.

(* Pascal *)

```
(*============================
 Đoạn Gối 1: Số lượng tối đa
 các đoạn gối nhau.
 ============================*)
program DoanGoi1;
uses crt;
const
  mn = 1001; bl = #32; nl = #13#10;
  fn = 'doan.inp'; gn = 'doan.out';
type
  KieuDoan = record a,b: integer; end;
  md1 = array[0..mn] of KieuDoan;
  mi1 = array[0..mn] of integer;
var n,m: integer; { n - số lượng đoạn, m - số đoạn được chọn }
    d: md1; { các đoạn d[1..n]}
    f,g: text;
    c: mi1; { c[i] = số lượng max các đoạn gối nhau đến i }
procedure Doc; tự viết
procedure Qsort(t,p: integer);tự viết
procedure XuLi;
var i,j: integer;
begin
  c[1] := 1;
  for i := 2 to n do { Tính c[i] }
    begin
      c[i] := 0;
        for j := i-1 downto 1 do
          begin
            if (d[j].b < d[i].a)  { doan j khong noi voi i }
            then break;
            if (d[j].b = d[i].a) then { j noi voi i }
               if (c[j] > c[i]) then c[i] := c[j];
          end;
      c[i] := c[i] + 1;
    end;
end;
procedure Ket; { Tim c max va hien thi ket qua }
var i,imax: integer;
begin
  assign(g,gn); rewrite(g);
  imax := 1;
  for i := 2 to n do
    if (c[imax] < c[i]) then imax := i;
  writeln(g,c[imax]); close(g);
end;
BEGIN
  Doc; Qsort(1,n); XuLi;  Ket;
END.
```

```
// C#
  using System;
  using System.IO;
  using System.Collections;
  /*================================================
    Đoạn Gối 1: Số lượng tối đa các đoạn gối nhau
   ================================================*/
  namespace SangTao2 {
      class DoanGoi1 {
          static public Doan[] d; // các đoạn d[0..n-1]
          static int n; // số đoạn
          static int m;// số max các đoạn gối nhau
          const string fn = "doan.inp"; // input file
          const string gn = "doan.out"; // output file
          static void Main(string[] args) {
              Doc(); QSortB(d,0,n-1);
              XuLi(); Ghi();
              XemKetQua(); Console.WriteLine("\n Fini ");
              Console.ReadLine();
          }
          static public void Doc(): tự viết
       // Sắp tăng các đoạn d[t..p] theo đầu phải b
          static public void QSortB(Doan[]d,int t,int p): tự viết
          static public void XuLi() {
              int[] c = new int[n];
              // c[i] - số lượng max đoạn gối kếtthúc tại đoạn i
              c[0] = 1; // lấy đoạn 0
              int imax = 0;
              for (int i = 1; i < n; ++i) {
                  c[i] = 0;
                  int jmax = i;
                  for (int j = i - 1; j >= 0; --j) {
                      if (d[j].b < d[i].a) break;
                      if (d[j].b == d[i].a)
                          if (c[j] > c[jmax]) jmax = j;
                  }
                  c[i] = c[jmax] + 1;
                  if (c[i] > c[imax]) imax = i;
              }
              m = c[imax];
          }
          static public void Ghi(){
              StreamWriter g = File.CreateText(gn);
              g.WriteLine(m); g.Close();
          }
        // Hien thi lai cac files input, output
          static public void XemKetQua(): tự viết
      }// DoanGoi1
      public struct Doan
      {
          public int a,b;
          public Doan(int x1, int x2) { a = x1; b = x2; }
      }  // Doan
  } // SangTao2
```

Chú thích

Trong bài này ta không cần sử dụng trường chỉ số riêng **id** cho kiểu đoạn.

Trong phương án C# ta tranh thủ tìm giá trị cmax = c[imax] sau mỗi lần tính c[i].

Bài 1.3 Đoạn gối 2

Cho N đoạn thẳng trên trục số với các điểm đầu a_i và điểm cuối b_i là những số nguyên trong khoảng –1000..1000, $a_i < b_i$. Liệt kê tối đa K đoạn thẳng gối nhau liên tiếp.

DOAN.INP	DOAN.OUT
5 2 7 1 3 7 9 3 4 4 5	3 2 4 5

Dữ liệu vào: tệp văn bản **DOAN.INP***: xem Đoạn gối 1*

Dữ liệu ra: tệp văn bản **DOAN.OUT**

Dòng đầu tiên: số tự nhiên K.

Tiếp đến là K dòng, mỗi dòng chứa một số tự nhiên biểu thị chỉ số của đoạn thẳng gối nhau liên tiếp trong dãy tìm được.

Thí dụ này cho biết tối đa có 3 đoạn 2, 4 và 5 tạo thành dãy đoạn gối nhau liên tiếp.

Thuật toán

Tương tự như bài Đoạn gối 1 nhưng cần tạo thêm con trỏ trước. **t[i] = j** có nghĩa là đoạn i được gối sau đoạn j. Thủ tục **GiaiTrinh(i)** liệt kê các đoạn gối liên tiếp từ phải qua trái thực chất là liệt kê theo chiều ngược các phần tử trong mảng con trỏ trước t bắt đầu từ phần tử thứ i. Giả sử t có dạng sau,

```
t[2] = 0; t[4] = 2; t[5] = 4;
```

và giả sử **i = 5** là vị trí đạt trị cmax, ta phải ghi vào file kết quả g dãy các đoạn gối nhau liên tiếp như sau,

```
2 4 5
```

Ta chỉ việc gọi *đệ quy muộn* như sau

```
procedure GiaiTrinh(i: integer);
begin
 if (i <> 0) then
   begin GiaiTrinh(t[i]); writeln(g,d[i].id); end;
end;
```

0	0	1	2	3	4	5
			0		2	4

Độ phức tạp: cỡ N^2.

```
(*  Pascal  *)
  (*===========================================
     Doan Goi 2: Liet ke toi da cac doan thang
                 goi nhau liên tiep
    ===========================================*)
  program DoanGoi2;
  uses crt;
  const
    mn = 1001; bl = #32; nl = #13#10;
    fn = 'doan.inp'; gn = 'doan.out';
  type
    KieuDoan = record a,b,id: integer; end;
    md1 = array[0..mn] of KieuDoan;
    mi1 = array[0..mn] of integer;
  var n,m: integer; { n - so luong doan, m - so doan duoc chon }
      d: md1; // cac doan
```

```
    f,g: text;
    c: mi1; { c[i] = so luong max doan goi voi doan i }
    t: mi1; { tro truoc }
  procedure Doc: tự viết
  procedure Qsort(i1,i2: integer): tự viết
  procedure XuLi;
  var i,j,jmax: integer;
  begin
    fillchar(t,sizeof(t),0); {Khởi trị mảng trỏ trước}
    c[1] := 1;
    for i := 2 to n do
      begin
        c[i] := 0; jmax := i;
          for j := i-1 downto 1 do
            begin
              if (d[j].b < d[i].a) then break;
              if (d[j].b = d[i].a) then
                 if (c[j] > c[jmax]) then jmax := j;
            end;
        c[i] := c[jmax]+1; t[i] := jmax;
      end;
  end;
  procedure GiaiTrinh(i: integer): tự viết;
  procedure Ket;
  var i,imax: integer;
  begin
    assign(g,gn);rewrite(g);
    imax := 1;
    for i := 2 to n do
      if (c[imax] < c[i]) then imax := i;
    writeln(g,c[imax]);
    GiaiTrinh(imax);
    close(g);
  end;
  BEGIN
    Doc; Qsort(1,n); XuLi; Ket;
  END.
```

// C#

```
  using System;
  using System.IO;
  using System.Collections;
  /*================================================
   *  Doan Goi 2: Liet ke toi da cac doan goi nhau *
   ================================================*/
  namespace SangTao2 {
      class DoanGoi2 {
          static public Doan[] d;
          static int n; // tong so doan
          static int[] t; // tro truoc
          const string fn = "doan.inp";
          const string gn = "doan.out";
          static void Main(string[] args){
              Doc(); QSortB(d,0,n-1);
              int m = 0; // so doan tim duoc
              int imax = 0; // chi so doan cuoi trong day max
```

```
            XuLi(ref imax, ref m); Ghi(imax,m);
            XemKetQua(); Console.ReadLine();
        }
        static public void Doc(): tự viết
        static public void XuLi(ref int imax, ref int m) {
            int [] c = new int[n];
            t = new int[n];
            Array.Clear(t, 0, t.Length);
            t[0] = -1;
            // c[i] - so luong doan goi ket thuc tai doan i
            c[0] = 1; // lay doan 0
            imax = 0;
            for (int i = 1; i < n; ++i){
                c[i] = 0;
                int jmax = i;
                for (int j = i - 1; j >= 0; --j){
                    if (d[j].b < d[i].a) break;
                    if (d[j].b == d[i].a)
                        if (c[j] > c[jmax]) jmax = j;
                }
                c[i] = c[jmax] + 1; t[i] = jmax;
                if (c[i] > c[imax]) imax = i;
            }
            m = c[imax];
        }
        // Sap tang cac doan theo dau phai (b)
        static public void QSortB(Doan[] d,int i1,int i2): tự viết
        static void Path(StreamWriter g, int imax) {
            if (imax != -1)
            { Path(g,t[imax]); g.WriteLine(d[imax].id); }
        }
        static public void Ghi(int imax, int m){
            StreamWriter g = File.CreateText(gn);
            g.WriteLine(m); Path(g, imax); g.Close();
        }
        // Hien thi lai cac files input, output
        static public void XemKetQua(): tự viết
    } // DoanGoi2
    public struct Doan: tự viết
} //  SangTao2
```

Bài 1.4 Đoạn gối 3

Cho N đoạn thẳng trên trục số với các điểm đầu a_i và điểm cuối b_i là những số nguyên trong khoảng –1000..1000, $a_i < b_i$. i = 1..n. Liệt kê các đoạn thẳng gối nhau có tổng chiều dài C lớn nhất.

DOAN.INP	DOAN.OUT

Dữ liệu vào: tệp văn bản **DOAN.INP***: xem bài Đoạn gối 1.*

8 2 7 1 3 7 9 3 40 3 5 2 3 5 9 9 16	39 2 4

Dữ liệu ra: tệp văn bản **DOAN.OUT**

Dòng đầu tiên: số tự nhiên C.

Mỗi dòng tiếp theo chứa một số tự nhiên biểu thị chỉ số của đoạn thẳng gối nhau liên tiếp trong dãy tìm được.

Thí dụ này cho biết hai đoạn 2 và 4 tạo thành dãy đoạn gối nhau liên tiếp có tổng chiều dài max là 39.

Thuật toán

Phương pháp: *Quy hoạch động* kết hợp với con trỏ trước t để giải trình kết quả.

Giả sử các đoạn được sắp tăng theo đầu phải b. Kí hiệu c(i) là tổng chiều dài lớn nhất các đoạn thẳng gối nhau liên tiếp tạo thành một dãy nhận đoạn i làm phần tử cuối dãy (khi khảo sát các đoạn từ 1..i). Để ý rằng ($b_i - a_i$) là chiều dài đoạn thứ i, ta có

c(1) = chiều dài đoạn 1 = $b_1 - a_1$,

Với i = 2..N: c(i) = max { c(j) | $1 \le j < i$, đoạn j kề trước đoạn i: $b_j = a_i$ } + ($b_i - a_i$),

Nếu j là chỉ số đạt max thì đặt t_i = j.

Độ phức tạp: N^2.

```
(*  Pascal  *)
  (*===========================================
      Doan Goi 3: Liet ke cac doan goi nhau
                  co tong chieu dai max
    ===========================================*)
  program DoanGoi3;
  uses crt;
  const
    mn = 1001; bl = #32; nl = #13#10;
    fn = 'doan.inp'; gn = 'doan.out';
  type
    KieuDoan = record a,b,id: integer; end;
    md1 = array[0..mn] of KieuDoan;
    mi1 = array[0..mn] of integer;
  var n,m: integer; { n - số lượng đoạn, m - số đoạn được chọn }
      d: md1;
      f,g: text;
      c: mi1; {c[i] = chieu dai max nhan i lam doan cuoi}
      t: mi1; { tro truoc }
  procedure Doc; tự viết
  procedure Qsort(l,r: integer); tự viết
  procedure XuLi;
  var i,j,jmax: integer;
  begin
    fillchar(t,sizeof(t),0);{ Khởi tạo mảng trỏ trước }
    c[1] := d[1].b - d[1].a; { Chieu dai doan 1 }
    for i := 2 to n do
      begin
        c[i] := 0; jmax := i;
          for j := i-1 downto 1 do
            begin
              if (d[j].b < d[i].a) then break;
```

```
            if (d[j].b = d[i].a) then
               if (c[j] > c[jmax]) then
                   jmax := j;
          end;
      c[i] := c[jmax] + (d[i].b - d[i].a); t[i] := jmax;
    end;
end;
procedure GiaiTrinh(i: integer); tự viết
procedure Ket; tự viết
BEGIN
  Doc; Qsort(1,n); XuLi; Ket;
END.
```

// C#

```
using System;
using System.IO;
using System.Collections;
/*================================================
 *  Doan Goi 3: Liet ke toi da cac doan goi nhau *
 *              co tong chieu dai max            *
 ================================================*/
namespace SangTao2 {
    class DoanGoi3 {
        static public Doan[] d;
        static int n; // tong so doan
        static int[] t; // tro truoc
        const string fn = "doan.inp";
        const string gn = "doan.out";
        static void Main(string[] args) {
            Doc(); QSortB(d,0,n-1);
            int maxlen = 0; // so doan tim duoc
            int imax = 0; // chi so doan cuoi trong day max
            XuLi(ref imax, ref maxlen);
            Ghi(imax,maxlen);
            XemKetQua(); Console.ReadLine();
        }
        static public void Doc(): tự viết
        static public void XuLi(ref int imax, ref int maxlen){
            int [] c = new int[n];
            t = new int[n];
            Array.Clear(t, 0, t.Length);
            t[0] = -1;
            // c[i] - so luong doan goi ket thuc tai doan i
            c[0] = d[0].b-d[0].a; // lay doan 0
            imax = 0;
            for (int i = 1; i < n; ++i) {
                c[i] = 0;
                int jmax = i; // Day dai nhat noi voi doan i
                for (int j = i - 1; j >= 0; --j) {
                    if (d[j].b < d[i].a) break;
                    if (d[j].b == d[i].a)
                        if (c[j] > c[jmax]) jmax = j;
                }
                c[i] = c[jmax] + (d[i].b - d[i].a) ; t[i] = jmax;
                if (c[i] > c[imax]) imax = i;
            }
```

```
            maxlen = c[imax];
          }
          // Sap tang cac doan theo dau phai (b)
          static public void QSortB(Doan[] d, int t, int p): tự viết
          static void Path(StreamWriter g, int imax): tự viết
          static public void Ghi(int imax, int maxlen): tự viết
          // Hien thi lai cac files input, output
          static public void XemKetQua():  tự viết
        }// DoanGoi3
        public struct Doan: tự viết
} //  SangTao2
```

Bài 1.5 Đoạn bao nhau 1

Cho N đoạn thẳng trên trục số với các điểm đầu a_i và điểm cuối b_i là những số nguyên trong khoảng –1000..1000, $a_i < b_i$, i = 1..n. Tìm K là số lượng nhiều đoạn nhất tạo thành một dãy các đoạn bao nhau liên tiếp. Hai đoạn [a,b] *và* [c,d] *được gọi là bao nhau nếu đoạn này nằm lọt trong đoạn kia, tức là* $a \le c < d \le b$ *hoặc* $c \le a < b \le d$.

DOAN.INP	DOAN.OUT
6 -1 12 8 10 8 11 2 7 17 18 13 20	3

Dữ liệu vào: tệp văn bản **DOAN.INP***: xem bài trước*

Dữ liệu ra: tệp văn bản **DOAN.OUT**

chứa duy nhất một số tự nhiên K.

Thí dụ này cho biết tối đa có 3 đoạn bao nhau là các đoạn [-1,12] ⊇ [8,11] ⊇ [8,10].

Thuật toán

Phương pháp: *Quy hoạch động.*

Giả sử các đoạn được sắp tăng theo đầu phải b như sau. Nếu hai đoạn có cùng đầu phải thì đoạn nào có đầu trái nhỏ hơn sẽ được đặt sau. Kí hiệu c(i) là số lượng lớn nhất các đoạn bao nhau liên tiếp trong đoạn i. Ta có,

c(1) = 1,

Với i = 2...N: $c(i) = \max \{ c(j) \mid 1 \le j < i$, Đoạn j lọt trong đoạn i: $a_j \ge a_i \} + 1$,

Độ phức tạp: N^2.

Hàm **SanhDoan(x,y)** thiết lập trật tự giữa hai đoạn **x** và **y** như sau:

- Nếu **x.b < y.b** cho kết quả **–1**, nếu không: xét tiếp
- Nếu **x.b > y.b** cho kết quả **1**, nếu không: xét tiếp
- Xét trường hợp **x.b = y.b**.
  - Nếu **x.a < y.a** cho kết quả **1**, nếu không: xét tiếp
  - Nếu **x.a > y.a** cho kết quả **–1**, nếu không: xét tiếp
  - Hai đoạn trùng nhau: **x.a = y.a** và **x.b = y.b** cho kết qủa **0**.

```
(*  Pascal  *)
  uses crt;
  const MN = 1001; bl = #32; nl = #13#10;
```

```
fn = 'doan.inp'; gn = 'doan.out';
type
Doan = record  a,b: integer;  end;
MD1 = array[0..MN] of Doan;
MI1 = array[0..MN] of integer;
var f,g: text;
d: MD1; { cac doan }
n: integer; { so doan }
procedure Doc; tự làm
function SanhDoan(x,y: Doan): integer;
begin
  if (x.b < y.b) then begin SanhDoan := -1; exit end;
  if (x.b > y.b) then begin SanhDoan := 1; exit end;
  if (x.a < y.a) then begin SanhDoan := 1; exit end;
  if (x.a > y.a) then begin SanhDoan := -1; exit end;
  SanhDoan := 0;
end;
procedure QSortB(t,p: integer);
var i,j: integer; m,x: Doan;
begin
  i := t; j := p; m := d[(i+j) div 2];
  while (i <= j) do
  begin
    while (SanhDoan(d[i],m) < 0) do i := i+1;
    while (SanhDoan(d[j],m) > 0) do j := j-1;
    if (i <= j) then
    begin
      x := d[i]; d[i] := d[j]; d[j] := x;
      i := i+1; j := j-1;
    end;
  end;
  if (t < j) then QSortB(t,j);
  if (i < p) then QSortB(i,p);
end;
function XuLi: integer;
   var c: mi1; { c[i] so doan bao nhau max }
     i,j,cmax: integer;
begin
  cmax := 0;
  for i := 1 to n do
  begin
    c[i] := 0;
    for j := i-1 downto 1 do
    begin
      if (d[j].b <= d[i].a) then break;
      if (d[j].a >= d[i].a) then
        if (c[j] > c[i]) then c[i] := c[j];
    end;
    c[i] := c[i] + 1;
    if (cmax < c[i]) then cmax := c[i];
  end;
  XuLi := cmax;
end;
procedure Ghi(k: integer);
begin
   assign(g,gn); rewrite(g); writeln(g,k); close(g);
end;
```

```
BEGIN
 Doc; QSortB(1,n); Ghi(XuLi); readln;
END.
```

// C#

```
using System;
using System.IO;
using System.Collections;
/*=============================================
   Doan Bao 1:  So luong toi da cac doan
                bao nhau
 =============================================*/
namespace SangTao2 {
    class DoanBao1 {
        static public Doan[] d; // cac doan
        static int n; // tong so doan
        const string fn = "doan.inp";
        const string gn = "doan.out";
        static void Main(string[] args){
            Doc(); QSortB(d, 0, n - 1);
            Ghi(XuLi());
            XemKetQua(); Console.WriteLine("\n Fini");
            Console.ReadLine();
        }
        static public void Doc(): tự viết
        static public int XuLi(){
            int [] c = new int [n];
            int cmax = 0;
            for (int i = 0; i < n; ++i){
                c[i] = 0;
                for (int j = i-1; j >= 0; --j){
                    if (d[j].b <= d[i].a) break;
                    if (d[j].a >= d[i].a)
                        if (c[j] > c[i]) c[i] = c[j];
                }
                ++c[i];
                if (cmax < c[i]) cmax = c[i];
            }
            return cmax;
        } // XuLi
        static public int SanhDoan(Doan x, Doan y){
            if (x.b > y.b) return 1;
            if (x.b < y.b) return -1;
            // x.b == y.b
            if (x.a < y.a) return 1;
            if (x.a > y.a) return -1;
            return 0;
        }
        // Sap tang cac doan theo dau b
        // Hai doan cung dau b: doan nao a nho dat sau
        static public void QSortB(Doan[] d, int t, int p){
            int i = t, j = p;
            Doan m = new Doan(d[(i + j) / 2]);
            while (i <= j){
                while (SanhDoan(d[i],m) < 0) ++i;
                while (SanhDoan(d[j],m) > 0) --j;
```

```
                if (i <= j){
                    Doan x = d[i]; d[i] = d[j]; d[j] = x;
                    ++i; --j;
                }
            }
            if (t < j) QSortB(d, t, j);
            if (i < p) QSortB(d, i, p);
        }
        static public void Ghi(int m){
            File.WriteAllText(gn, m.ToString());
        }
        // Hien thi lai cac files input, output
        static public void XemKetQua(): tự viết
    }// DoanBao1
    public struct Doan: tự viết
} //  SangTao2
```

Bài 1.6 Đoạn bao nhau 2

Cho N đoạn thẳng trên trục số với các điểm đầu a_i và điểm cuối b_i là những số nguyên trong khoảng –1000..1000, $a_i < b_i$, i = 1..n. Liệt kê tối đa K đoạn bao nhau.

DOAN.INP	DOAN.OUT
6 -1 12 8 10 17 18 2 7 8 11 13 20	3 1 5 2

Dữ liệu vào: tệp văn bản **DOAN.INP***: xem bài trước*

Dữ liệu ra: tệp văn bản **DOAN.OUT**

Dòng đầu tiên: số tự nhiên K.

Tiếp đến là K dòng, mỗi dòng chứa một số tự nhiên là chỉ số của đoạn trong dãy tìm được. Các chỉ số được liệ kê theo trật tự bao nhau từ lớn đến nhỏ.

Thí dụ này cho biết tối đa có 3 đoạn bao nhau là các đoạn 1, 5 và 2: `[-1,12] ⊇ [8,11] ⊇ [8,10].`

Thuật toán

Giống bài Đoạn bao nhau 1. Để có danh sách đoạn bao nhau ta dùng mảng trỏ t[1..n], t[i] trỏ đến đoạn j là đoạn nằm trong đoạn i và c[j] đạt giá trị max.

Độ phức tạp: N^2.

```
(*  Pascal  *)
  uses crt;
  const MN = 1001; bl = #32; nl = #13#10;
  fn = 'doan.inp'; gn = 'doan.out';
  type
  Doan = record  a,b: integer; id: integer;  end;
  MD1 = array[0..MN] of Doan;
  MI1 = array[0..MN] of integer;
  var f,g: text;
  d: MD1;
  t: MI1; { tro truoc }
  n: integer;
  imax, k: integer;
  procedure Doc; tự làm
  function SanhDoan(x,y: Doan): integer; tự làm
```

```
  procedure QSortB(t,p: integer): tự làm
  procedure XuLi;
  var c: mi1;
  i,j: integer;
  begin
    imax := 1;
    for i := 1 to n do
    begin
      c[i] := 0; t[i] := 0;
      for j := i-1 downto 1 do
      begin
        if (d[j].b <= d[i].a) then break;
        if (d[j].a >= d[i].a) then
          if (c[j] > c[i]) then
          begin c[i] := c[j]; t[i] := j end;
      end;
      c[i] := c[i] + 1;
      if (c[imax] < c[i]) then imax := i;
    end;
    k := c[imax];
  end;
  procedure Path(i: integer);
  begin
    if (i = 0) then exit;
    writeln(g,d[i].id); Path(t[i]);
  end;
  procedure Ghi;
  begin
     assign(g,gn); rewrite(g); writeln(g,k);
     path(imax); close(g);
  end;
  BEGIN
   Doc; QSortB(1,n); XuLi; Ghi; readln;
  END.
```

// C#

```
  using System;
  using System.IO;
  using System.Collections;
  /*================================================
     Doan Bao 2:  Liet ke toi da cac doan
                  bao nhau
   ================================================*/
  namespace SangTao2 {
      class DoanBao2 {
          static public Doan[] d; // Cac doan
          static public int [] t; // Con tro truoc
          static int n; // tong so doan
          const string fn = "doan.inp";
          const string gn = "doan.out";
          static void Main(string[] args){
              Doc(); QSortB(d, 0, n - 1);
              int k, imax;
              XuLi(out k, out imax); Ghi(k, imax);
              XemKetQua(); Console.WriteLine("\n Fini");
              Console.ReadLine();
```

```
        }
        static public void Doc(): tự làm
        static public void XuLi(out int cmax, out int imax){
            int [] c = new int [n];
            t = new int[n];
            imax = 0;
            for (int i = 0; i < n; ++i){
                c[i] = 0; t[i] = -1;
                for (int j = i-1; j >= 0; --j){
                    if (d[j].b <= d[i].a) break;
                    if (d[j].a >= d[i].a)
                        if (c[j] > c[i])
                        {   c[i] = c[j]; t[i] = j; }
                }
                ++c[i];
                if (c[imax] < c[i]) imax = i;
            }
            cmax = c[imax];
        } // XuLi
        static public int SanhDoan(Doan x, Doan y): tự làm
        static public void QSortB(Doan[] d, int t, int p): tự làm
        static void Path(StreamWriter g, int imax){
            if (imax != -1)
            { g.WriteLine(d[imax].id); Path(g, t[imax]); }
        }
        static public void Ghi(int k, int imax){
            StreamWriter g = File.CreateText(gn);
            g.WriteLine(k); Path(g, imax); g.Close();
        }
        static public void XemKetQua(): xem bài trước
    }// DoanBao2
    public struct Doan: tự làm
} //  SangTao2
```

Bài 1.7 Phủ đoạn 1

Cho N đoạn thẳng trên trục số với các điểm đầu a_i và điểm cuối b_i là những số nguyên trong khoảng –1000..1000, $a_i < b_i$. Hãy chỉ ra ít nhất K đoạn thẳng sao cho khi đặt chúng trên trục số thì có thể phủ kín đoạn [*x, y*] *với tọa độ nguyên cho trước.*

DOAN.INP	DOAN.OUT
5 3 23 1 15 3 10 8 20 17 25 2 7	3 1 3 4

Dữ liệu vào: tệp văn bản **DOAN.INP***: xem bài trước*

Dữ liệu ra: tệp văn bản **DOAN.OUT**

Dòng đầu tiên: số K, nếu vô nghiệm K = 0.

Tiếp theo là K số tự nhiên biểu thị chỉ số của các đoạn thẳng phủ kín đoạn [x, y].

Thí dụ này cho biết ít nhất là 3 đoạn 1, 3 và 4 sẽ phủ kín đoạn [3, 23].

Thuật toán

Phương pháp: *Tham*

Sắp các đoạn tăng theo đầu phải b.

k : = 1; { chỉ số đầu tiên }

v := x; { Đầu trái của đoạn [x,y] }

Lặp đến khi $v \geq y$

Duyệt ngược từ N đến k

Tìm đoạn j $[a_j, b_j]$ đầu tiên có đầu trái $a_j \leq v$;

Nếu không tìm được: vô nghiệm;

Nếu tìm được:

Ghi nhận đoạn j;

Đặt lại v := b_j;

Đặt lại k := j+1;

Độ phức tạp: N^2.

(* Pascal *)

```
(*=======================================
                Phu doan 1
      Tìm ít nhất K đoạn có thể phủ kín
         đoạn [x,y] cho trước
  =======================================*)
program PhuDoan1;
uses crt;
const
  mn = 2002; bl = #32; nl = #13#10;
  fn = 'doan.inp'; gn = 'doan.out';
type
  KieuDoan = record
               a,b: integer;
               id: integer; { Chỉ số đoạn }
             end;
  md1 = array[0..mn] of KieuDoan;
  mi1 = array[0..mn] of integer;
var n: integer; { n - so luong doan }
    d: md1; { cac doan }
    f,g: text;
    t: mi1;
    x, y: integer; { Doan can phu }
procedure Doc;
 var i: integer;
begin
  assign(f,fn); reset(f); readln(f,n);
  readln(f,x, y);
  for i := 1 to n do
   begin
     readln(f,d[i].a,d[i].b);
     d[i].id := i;
   end;
  close(f);
end;
procedure Qsort(l,r: integer): tự viết
(*---------------------------------------
   Duyet nguoc cac doan d[s..e]
   tim doan i dau tien thoa d[i].a <= x
  ---------------------------------------*)
```

```
function Tim(s,e,x: integer): integer;
var i: integer;
begin
  Tim := 0;
  for i := e downto s do
    if (d[i].a <= x) then
      begin
        Tim := i;
        exit;
      end;
end;
procedure Ket(k: integer): tự viết
procedure XuLi;
var i,j,k,v: integer; { k - so doan tim duoc }
begin
  v := x;
  k := 0; t[k] := 0;
  repeat
    j := Tim(t[k]+1,n,v);
    if (j = 0) then { Khong tim duoc }
       begin  Ket(0); { vo nghiem } exit;  end;
    v := d[j].b; k := k + 1; t[k] := j;
  until (v >= y);
  Ket(k); { co nghiem }
end;
BEGIN
  doc; qsort(1,n); xuli;
END.
```

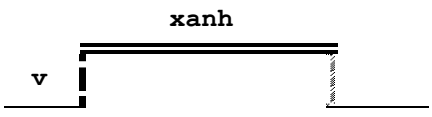
```
//  C#
using System;
using System.IO;
using System.Collections;
/*================================================
 *  Phu Doan 1: Liet ke toi thieu cac doan      *
 *              phu doan [x,y] cho truoc        *
 ================================================*/
namespace SangTao2 {
    class PhuDoan1 {
        static public Doan[] d; // cac doan
        static int n; // tong so doan
        static int m;// so doan tim duoc
        static int[] c; // luu ket qua cac doan duoc chon
        const string fn = "doan.inp";
        const string gn = "doan.out";
        static int x, y; // doan [x,y] can phu

        static void Main(string[] args) {
            Doc(); QSortB(d, 0, n - 1);
            m = XuLi(); Ghi();
            XemKetQua(); Console.WriteLine("\n Fini ");
            Console.ReadLine();
        }
        static public void Doc() {
            int[] a = Array.ConvertAll(File.ReadAllText(fn).
                          Split(new chae[] {' ','\n','\r','\t'},
```

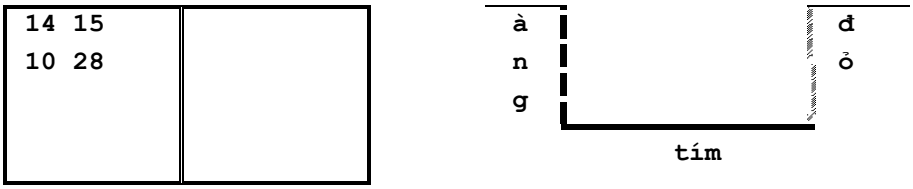
```
                        StringSplitOptions.RemoveEmptyEntries),
                         new Converter<string, int>(int.Parse));
            int j = 0;
            n = a[j++]; // tong so doan
            d = new Doan[n];
            // Doc doan xy can phu
            x = a[j++]; y = a[j++];
            for (int i = 0; i < n; ++i, j += 2) // doc doan i
                d[i] = new Doan(a[j], a[j + 1], i + 1);
        } // Doc
        static public int XuLi() {
            c = new int [n];
            int v = x; // dau trai doan [x,y]
            int k = 0; // dem so doan tim duoc
            int left = 0; // can trai
            do {
                int j = Tim(left, n - 1, v);
                if (j == -1) return 0; // vo nghiem
                c[k++] = j; // Tim duoc
                left = j + 1;
                v = d[j].b; // Chinh lai dau trai

            } while (v < y);
            return k;
        } // XuLi
        // Duyet nguoc cac doan d[s..e]
        // tim doan dau tien i: d[k].a <= x
        static public int Tim(int s, int e, int x) {
            for (int i = e; i >= s; --i)
                if (d[i].a <= x) return i;
            return -1;
        }
        // Sap tang cac doan theo dau phai (b)
        static public void QSortB(Doan[] d, int t, int p): tự viết
        static public void Ghi() :  tự viết
        // Hien thi lai cac files input, output
        static public void XemKetQua():  tự viết
    }// PhuDoan1
    public struct Doan:  tự viết
} //  SangTao2
```

Bài 1.8 Xanh đỏ tím vàng 1

Cho 4 loại đoạn thẳng sơn các màu xanh, đỏ, tím và vàng, bao gồm x đoạn màu xanh mỗi đoạn dài dx, d đoạn màu đỏ mỗi đoạn dài dd, t đoạn màu tím mỗi đoạn dài dt và v đoạn màu vàng mỗi đoạn dài dv. Các đoạn thẳng cùng màu thì có cùng chiều dài. Hãy chọn mỗi loại một số đoạn thẳng rồi xếp nối nhau theo chu vi để thu được một hình chữ nhật có diện tích lớn nhất với các cạnh lần lượt mang các màu tính theo chiều quay của kim đồng hồ là xanh, đỏ, tím, vàng. Các đại lượng trong bài đều là các số nguyên dương.

XDTV1.INP	XDTV1.OUT
15 12 6 21	15120 15 4 12 3





Dữ liệu vào: tệp văn bản **XDTV1.INP** *gồm 4 dòng, mỗi dòng hai số nguyên dương viết cách nhau*

Dòng thứ nhất: x dx

Dòng thứ hai: d dd

Dòng thứ ba: t dt

Dòng thứ tư: v dv

Dữ liệu ra: tệp văn bản **XDTV1.OUT**

Dòng đầu tiên: Diện tích của hình chữ nhật xanh - đỏ - tím - vàng.

Dòng thứ hai: 4 số cho biết số lượng đoạn thẳng cần chọn theo mỗi loại màu để ghép được hình chữ nhật diện tích max.

*Kết quả trên cho biết cần chọn 15 đoạn xanh, 4 đoạn đỏ, 12 đoạn tím và 3 đoạn vàng để ghép thành hình chữ nhật xanh – đỏ – tím – vàng với diện tích max là 15120 = (15*12)*(4*21) = (12*15)*(3*28).*

Thuật toán

Phương pháp: *Tham.*

Ta gọi một bộ xanh - tím là một cặp (nx,nt) trong đó nx là số ít nhất các đoạn màu xanh đặt liên tiếp nhau trên đường thẳng tạo thành đoạn AB và nt là số ít nhất các đoạn màu tím đặt liên tiếp nhau trên đường thẳng tạo thành đoạn CD sao cho AB = CD. Ta có ngay nx*dx = nt*dt. Dễ dàng tìm được nx = Bcnn(dx,dt)/dx và nt = Bcnn(dx,dt)/dt, trong đó Bcnn(a,b) là hàm tính bội chung nhỏ nhất của hai số tự nhiên a và b. Tương tự ta tính cho bộ đỏ - vàng. Tiếp đến ta tính xem tối đa có thể lấy được bao nhiêu bộ xanh - tím và bộ đỏ - vàng. Dễ thấy Số bộ xanh - tím = min(x div nx, t div nt). Tương tự ta tính cho bộ đỏ - vàng.

Độ phức tạp: O(1).

```
(*  Pascal  *)
  (****************************************
            Xanh Do Tim Vang 1
  ****************************************)
  program xdtv1;
  uses crt;
  const
   fn = 'xdtv1.inp'; gn = 'xdtv1.out'; bl = #32;
  var
   n: longint;
   f,g: text;
   x,d,t,v: longint; { so doan X D T V }
   dx,dd,dt,dv: longint; { cheu dai moi doan }
   nx,nt,nd,nv: longint; { so doan X D T V can chon }
  procedure Doc;
  begin
   assign(f,fn); reset(f);
   readln(f,x,dx); readln(f,d,dd); readln(f,t,dt); readln(f,v,dv);
   close(f);
  end;
  function Ucln(a,b: longint): longint;
    var r: longint;
```

```
begin
 while (b <> 0) do
   begin r := a mod b; a := b; b := r; end;
 Ucln := a;
end;
(*-----------------------------------
      Bcnn(a,b) = (a * b)/Ucln(a,b)
-----------------------------------*)
function Bcnn(a,b: longint): longint;
begin Bcnn := a * (b div Ucln(a,b)); end;
function Min(a,b: longint): longint; tự viết
procedure XuLi;
 var b, nxt, ndv: longint;
begin
 b := Bcnn(dx,dt);
 nx := b div dx; { so doan xanh trong 1 bo xanh - tim }
 nt := b div dt; { so doan tim trong 1 bo xanh - tim }
 nxt := Min(x div nx, t div nt); { so bo xanh - tim }
 nx := nxt * nx; { so doan xanh can chon }
 nt := nxt * nt; { so doan tim can chon }
 b := Bcnn(dd,dv);
 nd := b div dd; { so doan do trong 1 bo do - vang }
 nv := b div dv; { so doan vang trong 1 bo do - vang }
 ndv := Min(d div nd, v div nv);{ so bo do vang }
 nd := ndv * nd; { so doan do can chon }
 nv := ndv * nv; { so doan vang can chon }
end;
procedure Ghi;
begin
  assign(g,gn);  rewrite(g);
  writeln(g,nx*dx*nd*dd);{ dien tich }
  writeln(g,nx,bl,nd,bl,nt,bl,nv); { so doan moi loai }
  close(g);
end;
BEGIN
 Doc; XuLi; Ghi;
END.
```

// C#

```
using System;
using System.IO;
using System.Collections;
/*================================================
                  Xanh Do Tim Vang 1
 ================================================*/
namespace SangTao2 {
    class Xdtv1 {
        static int x, dx, d, dd, t, dt, v, dv;
        static int nx, nd, nt, nv; // Dap so
        const string fn = "xdtv1.inp";
        const string gn = "xdtv1.out";
        static void Main(string[] args) {
            Doc(); XuLi(); Ghi();
            XemKetQua(); Console.WriteLine("\n Fini");
            Console.ReadLine();
        }
```

```
        static public void Doc(){
            int[] a = Array.ConvertAll(File.ReadAllText(fn).
                          Split(new chae[] {' ','\n','\r','\t'},
                          StringSplitOptions.RemoveEmptyEntries),
                           new Converter<string, int>(int.Parse));
            int j = 0;
            x = a[j++]; dx = a[j++]; d = a[j++]; dd = a[j++];
            t = a[j++]; dt = a[j++]; v = a[j++]; dv = a[j];
        } // Doc
        static public void XuLi(){
            int b = Bcnn(dx,dt);
            nx = b / dx; // so doan xanh trong 1 bo xanh - tim
            nt = b / dt; // so doan tim trong 1 bo xanh - tim
            int nxt = Min(x / nx, t / nt); // so bo xanh - tim
            nx = nxt * nx; // so doan xanh can chon
            nt = nxt * nt; // so doan tim can chon
            b = Bcnn(dd,dv);
            nd = b / dd; // so doan do trong 1 bo do - vang
            nv = b / dv; // so doan vang trong 1 bo do - vang
            int ndv = Min(d / nd, v / nv);// so bo do vang
            nd = ndv * nd; // so doan do can chon
            nv = ndv * nv; // so doan vang can chon
        } // XuLi
        static public int Ucln(int a, int b){
            int r;
            while (b != 0) { r = a % b; a = b; b = r; }
            return a;
        }
        static public int Bcnn(int a, int b)
        { return a*(b/Ucln(a, b));}
        static public int Min(int a,int b): tự viết
        static public void Ghi(){
            StreamWriter g = File.CreateText(gn);
            g.WriteLine((nx*dx*nd*dd)); // dien tich
            g.WriteLine(nx+" "+nd+" "+nt+" "+nv);//so doan moi loai
            g.Close();
        }
        // Hien thi lai cac files input, output
        static public void XemKetQua(): tự viết
    }// Xdtv1
} //  SangTao2
```

Bài 1.9 Xanh đỏ tím vàng 2

Cho 4 loại đoạn thẳng sơn các màu xanh dài dx, đỏ dài dd, tím dài dt và vàng dài dv. Các đoạn thẳng cùng màu thì có cùng chiều dài và số lượng không hạn chế. Hãy chọn ra không quá N đoạn thẳng rồi xếp nối nhau theo chu vi để thu được một hình chữ nhật có diện tích lớn nhất với các cạnh lần lượt mang các màu theo chiều quay của kim đồng hồ là xanh, đỏ, tím, vàng. Dữ liệu trong bài đều là các số nguyên dương.

XDTV2.INP	XDTV2.OUT

Dữ liệu vào: tệp văn bản **XDTV2.INP**

35 3 2 2 5	480 8 10 12 4

Dòng thứ nhất: số N > 0.

Dòng thứ hai: bốn số dx dd dt dv

Dữ liệu ra: tệp văn bản **XDTV2.OUT**

Dòng đầu tiên: Diện tích của hình chữ nhật xanh - đỏ - tím - vàng.

Dòng thứ hai: 4 số cho biết số lượng đoạn thẳng cần chọn theo mỗi loại màu để ghép được hình chữ nhật diện tích max.

Kết quả trên cho biết cần chọn 8 đoạn xanh, 10 đoạn đỏ, 12 đoạn tím và 4 đoạn vàng để ghép thành hình chữ nhật có diện tích max là 480.

Thuật toán

Phương pháp: *Tham.*

Tương tự như bài Xanh đỏ tím vàng 1, trước hết ta tính các bộ xanh - tím (nx,nt) và bộ đỏ - vàng (nd, nv). Tiếp đến ta tính xem khi lấy i bộ xanh – tím để kết hợp với j bộ đỏ - vàng thì thu được hình chữ nhật có diện tích max là bao nhiêu. Lưu ý rằng i bộ xanh - tím sẽ gồm i(nx+nt) đoạn thẳng. Số đoạn thẳng còn lại khi đó sẽ là N – i(nx+nt). Số này sẽ tạo ra được j = (N – i(nx+nt)) / (nd+nv) bộ đỏ - vàng.

Độ phức tạp: O(n).

```
(*  Pascal  *)
  (*****************************************
            Xanh Do Tim Vang 2
  *****************************************)
  program xdtv2;
  uses crt;
  const
   fn = 'xdtv2.inp';  gn = 'xdtv2.out';  bl = #32;
  var
   n: longint;
   f,g: text;
   x,d,t,v: longint;
   dx,dd,dt,dv: longint;
   nx,nt,nd,nv: longint;
   smax,bmax: longint;
  procedure Doc: Tự viết;
  function Ucln(a,b: longint): Tự viết;
  function Bcnn(a,b: longint): Tự viết;
  function Min(a,b: longint): Tự viết;
  (*-------------------------------------------------------
    1 bo xanh - tim = (sx,st) = (so doan xanh, so doan tim)
    sx = bcnn(dx,dt) div dx
    st = bcnn(dx,dt) div dt
    bxt = sx + st
    -------------------------------------------------------*)
  procedure XuLi;
   var b: longint;
       bxt, bdv: longint;
       s: longint;
       sx,st, sd, sv: longint;
  begin
   b := bcnn(dx,dt);
```

```
  sx := b div dx;
  st := b div dt;
  bxt := sx + st;
  b := Bcnn(dd,dv);
  sd := b div dd;
  sv := b div dv;
  bdv := sd + sv;
  smax := 0; bmax := 0;
  for b := 1 to ((n - bdv) div bxt) do
    begin
      s := (b*sx*dx)*(((n - b*bxt) div bdv)*sd*dd);
      if (s > smax) then
        begin  bmax := b; smax := s;  end;
     end;
   nx := bmax * sx; nt := bmax * st;
   b  := smax div (nx*dx); { Chieu dai canh Do ( = Vang) }
   nd := b div dd; nv := b div dv;
end;
procedure Ghi: Tự viết;
BEGIN
 Doc; XuLi; Ghi;
END.
```

// C#

```
using System;
using System.IO;
using System.Collections;
/*================================================
                Xanh Do Tim Vang 2
 ================================================*/
namespace SangTao2 {
    class Xdtv2 {
        static int n;
        static int dx, dd, dt, dv;
        static int nx, nd, nt, nv; // Dap so: so luong moi loai
        const string fn = "xdtv1.inp";
        const string gn = "xdtv1.out";
        static void Main(string[] args){
            Doc(); XuLi(); Ghi();
            XemKetQua(); Console.WriteLine("\n Fini");
            Console.ReadLine();
        }
        static public void Doc(){
            int[] a = Array.ConvertAll(File.ReadAllText(fn).
                          Split(new chae[] {' ','\n','\r','\t'},
                          StringSplitOptions.RemoveEmptyEntries),
                           new Converter<string, int>(int.Parse));
            int j = 0;
            n = a[j++]; dx = a[j++]; dd = a[j++];
            dt = a[j++]; dv = a[j];
        } // Doc
        static public void XuLi() {
            int b = Bcnn(dx,dt);
            int sx = b / dx; // so luong doan xanh trg 1 bo XT
            int st = b / dt; // so lg doan tim trg 1 bo X-T
            int sxt = sx+st; // tg so doan xanh va tim trg bo XT
```

```
            b = Bcnn(dd,dv);
            int sd = b / dd, sv = b / dv; // do, vang
            int sdv = sd + sv;// tg so doan do va vg trg bo DV
            int smax = 0; // dt max
            int bxtmax = 0; // so bo XT toi da
            int bb = (n-sdv)/sxt; // can tren cua cac bo XT
            for (b = 1; b <= bb; ++b) { // b - so bo XT
                 int s = (b*sx*dx)*(((n-b*sxt)/sdv)*sd*dd);// dt
                if (s > smax) { bxtmax = b; smax = s; }
            }
            nx = bxtmax * sx; nt = bxtmax * st;
            b = smax / (nx * dx); // chieu dai canh Do = Vang
            nd = b/dd; nv = b/dv;
        } // XuLi
        static public int Ucln(int a, int b): Tự viết;
        static public int Bcnn(int a, int b): Tự viết;
        static public int Min(int a,int b): Tự viết;
        static public void Ghi(): Tự viết;
        static public void XemKetQua(): Tự viết;
    }// Xdtv2
} //  SangTao2
```

Bài 1.10 Phủ đoạn 2

Cho n đoạn thẳng <a_i , b_i> với các điểm đầu a_i và điểm cuối b_i là những số nguyên trong khoảng –1000..1000,

$a_i < b_i$ và thuộc một trong 4 dạng sau đây:

[*d*,*c*] - *đoạn đóng: chứa điểm đầu d và điểm cuối c,*

(*d*,*c*] - *đoạn mở trái: không chứa điểm đầu d, chứa điểm cuối c,*

[*d*,*c*) - *đoạn mở phải: chứa điểm đầu d, không chứa điểm cuối c,*

(*d*,*c*) - *đoạn mở: không chứa điểm đầu d và điểm cuối c.*

Hãy chỉ ra ít nhất K đoạn thẳng sao cho khi đặt chúng trên trục số thì có thể phủ kín đoạn <x, y> với tọa độ nguyên cho trước.

DOAN.INP	DOAN.OUT
6 (4,10) (-2, 5) [5 , 7] [6, 7) [7, 8) (8,9) (7 , 10]	3 1 2 6

Dữ liệu vào: tệp văn bản **DOAN.INP**

Dòng đầu tiên: số tự nhiên 1 < N ≤1000.

Dòng thứ hai: Đoạn x y

Từ dòng thứ ba liệt kê các đoạn, mỗi dòng có thể chứa nhiều đoạn, mỗi đoạn được ghi trọn trên một dòng.

Dữ liệu ra: tệp văn bản **DOAN.OUT**

Dòng đầu tiên: số K, nếu vô nghiệm K=0.

Tiếp theo là K số tự nhiên biểu thị chỉ số của các đoạn thẳng phủ kín đoạn x y.

Kết quả trên cho biết ít nhất là 3 đoạn 1, 2 và 6 sẽ phủ kín đoạn (4,10).

Chú ý: Giữa các số và các ký tự trong file input có thể chứa các dấu cách.

Thuật toán

Phương pháp: *Tham*

Để ứng dụng thuật toán của bài Phủ đoạn 1 ta đưa các đoạn về cùng một *dạng đóng* bằng cách chỉnh lại các đầu mở. Cụ thể là thêm/bớt điểm đầu mở của mỗi đoạn một lượng ε = 0.3 như sau,

[d,c] giữ nguyên
(d,c) → [d + ε, c],
[d,c) → [d, c - ε],
(d,c) → [d + ε, c - ε].

Qui tắc trên khá dễ hiểu. Bạn hãy giải thích vì sao chọn ε = 0.3 mà không chọn ε = 0.5?

Hai trường a và b thể hiện các điểm đầu và cuối mỗi đoạn cần được khai báo kiểu **real** (**float**). Các biến liên quan đến các trường này trong thủ tục xử lí cũng cần được khai báo theo các kiểu trên.

Ta đọc tất cả các đoạn và ghi vào mảng d[0..N], trong đó d[0] sẽ chứa đoạn x, y.

Cách đọc các đoạn được tổ chức trên cơ sở giả thiết là các đoạn được viết đúng cú pháp. Mỗi lần ta đọc một kí tự ch từ tệp input. Nếu (ch = '(') hoặc (ch = '[') thì ta gặp kí tự đầu đoạn: ta tiến hành đọc một đoạn. Để đọc một đoạn ta lần lượt đọc số thứ nhất, bỏ qua dấu phảy (',') ngăn cách giữa hai số rồi đọc số thứ hai. Thủ tục đọc một đoạn kết thúc khi gặp một trong hai dấu đóng ngoặc là ']' hoặc ')'. Căn cứ vào các dấu mở và đóng ngoặc đầu và cuối mỗi đoạn ta xác định lượng ε cần thêm hoặc bớt cho mỗi điểm đầu hoặc cuối đoạn, đương nhiên các điểm này cần được biểu diễn dưới dạng số thực.

Độ phức tạp: N^2.

Chú ý

Bạn có thể dùng *kỹ thuật đồng dạng* chuyển trục số sang trục số "phóng đại" gấp 3 lần. Khi đó các đoạn tương ứng sẽ được chuyển như sau:

[d,c] → [3d - 1, 3c + 1],
(d,c] → [3d + 1, 3c + 1],
[d,c) → [3d - 1, 3c - 1],
(d,c) → [3d + 1, 3c - 1].

Tức là giữ lại kiểu nguyên và chọn ε = 1. Cách làm này có đơn giản hơn trong trường hợp giới hạn của d và c là nhỏ.

```
(*  Pascal  *)
 (*=======================================
                Phu doan 2
   =======================================*)
 program PhuDoan2;
 uses crt;
 const
   mn = 2002; bl = #32; nl = #13#10;
   fn = 'doan.inp'; gn = 'doan.out';
   MoVuong = '['; DongVuong = ']'; MoTron = '('; DongTron = ')';
   NgoacMo = [MoVuong, MoTron]; NgoacDong = [DongVuong, DongTron];
   ChuSo = ['0'..'9']; CongTru = ['+','-']; eps = 0.3;
 type
   KieuDoan = record
                a,b: real;
                id: integer; { chỉ số đoạn }
              end;
   md1 = array[0..mn] of KieuDoan;
   mi1 = array[0..mn] of integer;
 var n: integer; { n - so luong doan }
     d: md1;
     f,g: text;
     t: mi1;
     xy: KieuDoan;
```

```
    ch: char;
    k: integer; {dem so doan da doc}
    Ngoac: char;
(* Doc 1 so nguyen tu input file *)
function DocSo: integer;
var s,dau: integer;
begin
  s := 0; dau := 1;
  while not (ch in (CongTru + ChuSo)) do read(f,ch);
  if (ch in CongTru) then
  begin
    if (ch = '-') then dau := -1;
    read(f,ch);
  end;
  while not (ch in ChuSo) do read(f,ch);
  repeat
    s := 10*s + ord(ch) - ord('0');
    read(f,ch);
  until not (ch in ChuSo);
  DocSo := dau * s;
end;
procedure DocDoan;
begin
  k := k + 1; d[k].id := k; Ngoac := ch; d[k].a := DocSo;
  if (Ngoac = MoTron) then d[k].a := d[k].a + eps;
  while (ch <> ',') do read(f,ch);
  d[k].b := DocSo;
  while not(ch in NgoacDong) do read(f,ch);
  if (ch = DongTron) then d[k].b := d[k].b - eps;
end;
procedure Doc;
 var i: integer;
begin
  assign(f,fn); reset(f); readln(f,n); k := -1;
  while (k < n) and (not eof(f)) do
  begin
    read(f,ch);
    if (ch in NgoacMo) then DocDoan;
  end;
  close(f); xy.a := d[0].a; xy.b := d[0].b;
end;
procedure Qsort(l,r: integer): xem bài phủ đoạn 1;
function Tim(id,ic: integer; x: real): xem bài phủ đoạn 1;
procedure Ket(k: integer): xem bài phủ đoạn 1;
procedure XuLi: xem bài phủ đoạn 1;
BEGIN
  Doc; Qsort(1,n); XuLi;
END.
```

// C#

```
using System;
using System.IO;
using System.Collections;
namespace SangTao2 {
    class PhuDoan2  { // Cac doan dong mo
        static public string fn = "Doan.inp";
```

```
static public string gn = "Doan.out";
static public string s; // du lieu vao
static public Doan[] d; // cac doan
static public int[] c; // luu cac doan ket qua
static public Doan xy;
static public int n = 0; // so luong doan
static public int si; // index cho s
static int m; // so luong doan phu
const float eps = 0.3f;
static void Main(string[] args) {
    Doc(); QSortB(d, 0, n - 1);
    m = XuLi(); Ghi(); XemKetQua();
    Console.WriteLine("\n Fini ");
    Console.ReadLine();
}
static public void XemKetQua(): tự làm
static public void Doc() {
    s = (File.ReadAllText(fn)).Trim();
    si = 0; n = DocSo();
    xy = DocDoan(0);
    d = new Doan[n];
    for (int i = 0; i < n; ++i)   d[i] = DocDoan(i+1);
}
static public void Ghi(): tự làm
static public int XuLi(): tự làm
static public int Tim(int i, int j, float x): tự làm
static public void QSortB(Doan[] d, int t, int p): tự làm
// đọc đoạn thứ i
static public Doan DocDoan(int i) {
    Cach();
    char mo = s[si++];
    float a = DocSo();
    if (mo == '(')  a += eps;
    float b = DocSo();
    Cach();
    char dong = s[si++];
    if (dong == ')' b -= eps;
    return new Doan(a, b, i);
}
// bỏ qua các dấu trắng
static public void Cach() {
    while (s[si]==' '||s[si]=='\n'
          ||s[si]=='\t'||s[si]=='\r')  ++si;
}
static public void DenSo() {
    while ((s[si]<'0'||s[si]>'9')&&(s[si]!='+')
            &&(s[si]!='-')) ++si;
}
static public int DocSo(){
    int m = 0, dau = 1;
    DenSo();
    if (s[si] == '+') ++si;
    if (s[si] == '-') { ++si; dau = -1; }
    do {
      m = m * 10 + s[si++] - '0';
    } while (s[si] >= '0' && s[si] <= '9');
    return dau*m;
```

```
        }
      } // PhuDoan2
      public struct Doan  { // Mô tả một đoạn
        public float a, b;
        public id;
        public Doan(float x1, float x2, int z)  // Tạo đoạn mới
        { a = x1; b = x2; id = z; }
      }  // Doan
  } // SangTao2
```

Bài 1.11 Đoạn rời 2

Cho N đoạn thẳng với các điểm đầu a_i và điểm cuối b_i là những số nguyên trong khoảng –1000..1000, $a_i < b_i$. Liệt kê số lượng tối đa các đoạn thẳng rời nhau. Hai đoạn được xem là rời nhau nếu chúng không có điểm chung. Các đoạn có dạng như trong bài Phủ đoạn 2.

DOAN.INP	DOAN.OUT
8 (-2, 3) [3 , 5) [8, 12] [13 ,15) (1 , 9) (2, 5] [5 ,8) [7, 15]	5 1 2 7 3 4

Dữ liệu vào: tệp văn bản **DOAN.INP**

Dòng đầu tiên: số tự nhiên N > 1.

Từ dòng thứ hai: liệt kê các đoạn, mỗi dòng có thể chứa nhiều đoạn, mỗi đoạn được ghi trọn trên một dòng.

Dữ liệu ra: tệp văn bản **DOAN.OUT**

Dòng đầu tiên: số K.

Tiếp theo là K số tự nhiên biểu thị chỉ số của các đoạn thẳng rời nhau.

Kết quả trên cho biết có tối đa 5 đoạn rời nhau là 1, 2, 7, 3 và 4.

Thuật toán

Phương pháp: *Tham.*

Trước hết ta chỉnh lại các đầu hở giống như bài trước sau đó áp dụng thuật toán của bài đoạn rời.

Các điểm đầu và cuối đoạn và các biến liên quan được khai báo kiểu số thực.

Độ phức tạp: N.logN chi phí cho quick sort.

```
(*   Pascal   *)
  (*=======================================
     liet ke toi da cac doan thang roi nhau
    =======================================*)
  program DoanRoi2;
  uses crt;
  Tổ chức dữ liệu: xem bài Phủ đoạn 2
  function DocSo: xem bai Phu doan 2;
  procedure DocDoan: xem bai Phu doan 2;
  procedure Doc: xem bai Phu doan 2;
  procedure Qsort(l,r: integer): xem bai Phu doan 2;
  procedure XuLi : xem bai Doan roi 1;
  procedure Ket: xem bai Doan roi 1 ;
  BEGIN
    Doc;   Qsort(1,n);
    XuLi;  Ket;
  END.
```

```
// C#
  using System;
  using System.IO;
  using System.Collections;
  namespace SangTao2 {
      class DoanRoi2 { // Cac doan dong mo
          Tổ chức dữ liệu: xem bài Phủ đoạn 2
          static void Main(string[] args) {
              Doc(); QSortB(d, 0, n - 1);
              XuLi(); Ghi();
              XemKetQua();
              Console.WriteLine("\n Fini ");
              Console.ReadLine();
          }
          static public void XemKetQua(): xem bai Phu doan 2
          static public void Doc(): xem bai Phu doan 2
          static public void Ghi(): xem bai Doan Roi 1
          static public void XuLi(): xem bai Doan roi 1
          // Sap tang cac doan theo dau phai (b)
          static public void QSortB(Doan[] d, int t, int p): tự làm
          static public Doan DocDoan(int i): xem bai Phu doan 2
          static public int DocSo(): xem bai Phu doan 2
      } // DoanRoi2
      public struct Doan: xem bai Phu doan 2
  } // SangTao2
```

Bài 1.12 Ghép hình chữ nhật

Cho N đoạn thẳng cùng chiều dài d đơn vị. Hãy chọn K đoạn để ghép thành cạnh của hình chữ nhật có diện tích lớn nhất.

Input: Hai số N và d, n ≥ 4.

Output: Hiển thị trên màn hình

- *t: Tổng số đoạn cần chọn t,*
- *a: Số đoạn đặt trên 1 chiều rộng*
- *b: Số đoạn đặt trên 1 chiều dài,*
- *s: Diện tích max.*

Thí dụ,

```
Input: N = 23, d = 2.
Output:   22   5   6   120.
```

Thuật toán

Định lý Trong các hình chữ nhật cùng chu vi thì hình vuông có diện tích lớn nhất.

Chứng minh

Gọi c là chu vi của hình chữ nhật, a là chiều rộng, b là chiều dài, b ≥ a, d là độ lệch giữa chiều dài b và chiều rộng a. Ta có, b = a + d và c = 2(a+a+d) = 4a + 2d, diện tích s = a(a+d). Thay giá trị a = (c–2d)/4 vào biểu thức tính diện tích và rút gọn ta thu được $s = c^2/16 – d^2/4 = (c^2 – 4d^2)/16$. Giá trị s đạt max khi và chỉ khi d = 0, tức là khi a = b. Vậy hình chữ nhật có diện tích lớn nhất là $c^2/16$ chính là hình vuông.

Từ định lý trên ta rút ra heuristics sau đây: muốn xây dựng một hình chữ nhật có diện tích lớn nhất từ một chu vi c cố định với các cạnh nguyên cho trước ta phải chọn độ lệch giữa chiều dài và chiều rộng nhỏ nhất có thể được.

Khởi trị: a = b = N div 4.

Xét các trường hợp số đoạn còn thừa r = N mod 4.

r = 0: bỏ qua

r = 1: bỏ qua

r = 2: thêm chiều dài 1 đoạn, b := b + 1;

r = 3: thêm chiều dài 1 đoạn, b := b + 1;

Tổng quát: a = N div 4; b = a + (N mod 4) div 2;

Khi đó diện tích sẽ là: s = a*b*d*d.

Thí dụ, N = 23, d = 2: a = 23 div 4 = 5; b = a + (N mod 4) div 2 = 5 + (3 div 2) = 5 + 1 = 6;

t = 2*(a+b) = 2*(5+6) = 22; s = a*b*d*d = 5*6*2*2 = 120.

Độ phức tạp: O(1).

(* Pascal *)

```
(*=======================================
   Chon t trong n doan de ghep duoc HCN
   dien tich max
   =======================================*)
program GhepChuNhat;
uses crt;
const bl = #32;
procedure ChuNhatMax(n,d: longint);
var a,b,t,s: longint;
begin
   a := n div 4; b := a + ((n mod 4) div 2);
   t := 2 * (a + b); { tong so doan }
   s := a * b * d * d;
   writeln(t,bl,a,bl,b,bl,s);
end;
BEGIN
  ChuNhatMax(23,2);
END.
Kết quả dự kiến:
22 5 6 120
```

// C#

```
using System;
using System.IO;
using System.Collections;
namespace SangTao2 {
    class GhepHCN {
        static void Main(string[] args){
            ChuNhatMax(23,2);
            Console.WriteLine("\n Fini ");
            Console.ReadLine();
        }
        static public void ChuNhatMax(int n, int d){
            int a = n / 4;
            int b = a + ((n % 4) / 2);
            int t = 2 * (a + b);
            int s = a * b * d * d;
            Console.WriteLine(t + " " + a + " " + b + " " + s);
        }
    } // GhepHCN
} // SangTao2
```

Bài 1.13 Xanh đỏ

Cho x đoạn thẳng màu xanh có chiều dài bằng nhau là dx đơn vị và d đoạn màu đỏ có chiều dài bằng nhau là dd đơn vị. Hãy chọn nx đoạn xanh và nd đoạn đỏ đặt trên chu vi của hình chữ nhật tạo thành một đường viền đan màu (các màu xen kẽ nhau) và diện tích của hình là lớn nhất.

Input: Bốn số x, dx, d và dd, $x \geq 2$, $d \geq 2$.

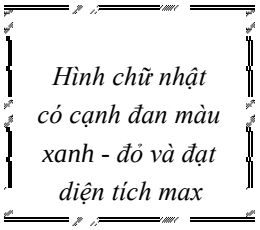
Output: Hiển thị trên màn hình

- nx: Số đoạn xanh được chọn,

- nd: Số đoạn đỏ được chọn,

- s: Diện tích max.

```
* Thí dụ 1
input: (x, dx, d, dd) = (5, 2, 7, 2)
output: (nx, nd,  s) = (5, 5, 24)
* Thí dụ 2
input: (x, dx, d, dd) = (6, 2, 7, 3)
output: (nx, nd,  s) = (6, 6, 56)
* Thí dụ 3
input: (x, dx, d, dd) = (6, 2, 7, 2)
output: (nx, nd, s) = (6, 6, 36)
* Thí dụ 4
input: (x, dx, d, dd) = (7, 2, 7, 3)
output: (nx, nd, s) = (6, 6, 56)
```




```
x = 11, dx = 1,
 d = 10, dd = 2.
nx = 10, nd = 10,
s = 7 × 8 =  56.
```

Thuật toán

Dễ thấy là để thu được hình có đường viền đan màu thì cần chọn số đoạn xanh *nx* và số đoạn đỏ *nd* như nhau, tức là chọn *nx* = *nd* = min(*x*, *d*). Khi đó chu vi của hình sẽ gồm *t* = *nx* + *nd* = 2*nx* = 2*nd* đoạn, và do đó *t* là một số chẵn. Do *t* chẵn nên (*t* mod 4) sẽ bằng 0 hoặc 2.

Kí hiệu *a* là số đoạn (tính cả xanh và đỏ) cần đặt trên một chiều rộng, *b* là số đoạn (tính cả xanh và đỏ) cần đặt trên một chiều dài của hình. Xét hai trường hợp *dx* = *dd* và *dx* ≠ *dd*.

1. Trường hợp thứ nhất: *dx* = *dd*. Ta có, *a* = *b* = *t* div 4. Nếu (*t* mod 4 = 2). Ta thêm vào chu vi một cặp xanh đỏ nữa. Vậy ta cần chọn:

- *nx* = min(*x*,*d*) đoạn xanh,

- *nd* = *nx* đoạn đỏ,

- Diện tích max: *s* = *a* * *b* * *dx* * *dx* với

  a = *t* div 4 là số đoạn đặt trên 1 chiều rộng,

  b = *a* + ((*t* mod 4) div 2) là số đoạn đặt trên 1 chiều dài.

2. Trường hợp thứ hai: *dx* ≠ *dd*. Ta thấy, để đảm bảo tính đan màu thì *a* và *b* phải có cùng tính chẵn lẻ. Từ đó thấy rằng khi (*t* mod 4 = 2) ta phải bỏ qua số dư, vì nếu thêm cho chiều dài *b* 1 đoạn nữa thì tính chẵn lẻ của *a* và *b* sẽ khác nhau. Để ý rằng khi *a* và *b* cùng chẵn thì hình nhận được sẽ vuông và mỗi cạnh chứa đúng *z* = *a* div 2 đoạn xanh và *z* đoạn đỏ. Khi đó diện tích có thể tính theo công thức: *s* = sqr(*z* * (*dx* + *dd*)). Nếu *a* và *b* cùng lẻ thì số lượng đoạn xanh và số lượng đoạn đỏ trong một cạnh sẽ hơn kém nhau 1 đơn vị. Nếu cạnh này nhiều xanh hơn đỏ thì cạnh kề với cạnh đó sẽ nhiều đỏ hơn xanh. Khi đó diện tích sẽ được tính theo công thức

s = (*z* * *dx* + (*z*+1) **dd*) * (*z* * *dd* + (*z*+1) * *dx*) = (*z* * *dx* + *z* * *dd* + *dd*) * (*z* * *dd* + *z* * *dx* + *dx*)

= (*z* * (*dx* + *dd*) + *dd*) * (*z* * (*dx* + *dd*) + *dx*) = (*k* + *dd*) * (*k* + *dx*)

với *z* = *a* div 2, *k* = (*a* div 2)*(*dx* + *dd*).

Kết quả là cần chọn:

$nx = \min(x,d)$. Chu vi $2*nx$ phải là bội của 4, tức là nx phải chẵn. Nếu nx lẻ thì đặt lại $nx := nx - 1$.

$nd = nx$;

tổng cộng $t = nx + nd$ đoạn, trong đó

Số đoạn đặt trên 1 chiều rộng: $a = t$ div 4,

Số đoạn đặt trên 1 chiều dài: $b = a$,

- Diện tích max: $(k+dd) * (k+dx)$, $k = (a \text{ div } 2)*(dx+dd)$.

Độ phức tạp: O(1).

(* Pascal *)

```
program XanhDo;
uses crt;
const bl = #32;
function Min(a,b: longint): tự viết
procedure XD(x,dx,d,dd: longint);
 var a,b,nx,nd,k,t,s: longint;
begin
  if (dx = dd) then
    begin
      nx := Min(x,d); nd := nx; t := nx + nd;
      a := t div 4;
      b := a + ((t mod 4) div 2); s := a * b * dx * dx
    end
   else { dx <> dd }
    begin
      nx := Min(x,d);
      if (nx mod 2 > 0) then nx := nx - 1;
      nd := nx; t := nx + nd; a := t div 4; b := a;
      k := (a div 2) * (dx + dd);
      if (Odd(a)) then s := (k + dx) * (k + dd)
      else s = k*k;
    end;
   writeln(nx,bl,nd,bl,s);
end;
BEGIN
 XD(7, 2, 7, 3);
END.
```

// C#

```
using System;
using System.IO;
using System.Collections;
namespace SangTao2 {
    class XanhDo {
        static void Main(string[] args){
            XD(11, 1, 9, 2);
            Console.WriteLine("\n Fini ");
            Console.ReadLine();
        }
        static public void XD(int x, int dx, int d, int dd){
            int nx = Min(x,d); // so luong doan xanh can chon
            int nd; // so luong doan do can chon
            int t; // tong so: nx + nd
            int a,b; // chieu rong, dai tinh theo so doan
```

```
            int s; // dien tich max
            if (dx == dd){
                nd = nx; t = nx + nd;
                a = t / 4; b = a+((t % 4)/2);
                s = a*b*dx*dx;
            } else {
               if (nx % 2 > 0) --nx;
               nd = nx; t = nx+nd; b = a = t / 4;
               int k = (a/2)*(dx+dd);
               s = (a % 2 == 0) ? k * k : (k+dx)*(k+dd);
            }
            Console.WriteLine("\n " + nx + " " + nd + " " + s);
        }
        static public int Min(int a,int b): tự viết
    } // XanhDo
} // SangTao2
```

Bài 1.14 Xếp đoạn

Cho N đoạn thẳng trên trục số với các điểm đầu x_i là những số nguyên trong khoảng –1000..1000 và độ dài d_i là những số nguyên dương trong khoảng 1..1000, i = 1..N. Tính tổng chiều dài các đoạn đó phủ trên trục số.

DOAN.INP	OUTPUT
5 3 5 -11 3 -20 4 -12 8 2 5	28

Dữ liệu vào: tệp văn bản **DOAN.INP**

Dòng đầu tiên: số tự nhiên $1 < N \leq 1000$.

Dòng thứ i trong N dòng tiếp theo, mỗi dòng chứa hai số nguyên a_i d_i cách nhau qua dấu cách, biểu thị điểm đầu và chiều dài của đoạn thứ i, i = 1..N.

Dữ liệu ra: hiển thị trên màn hình tổng chiều dài t các đoạn phủ trên trục số.

Thuật toán

Phương pháp: *tham.*

Sắp tăng các đoạn theo điểm đầu *x*.

Ta dùng kí hiệu [*x*,*y*] biểu diễn cho đoạn thẳng có điểm đầu *x* và điểm cuối *y*, *x*:*d* biểu diễn cho đoạn thẳng có điểm đầu *x* và chiều dài *d*. Ta định nghĩa một *làn* là đoạn tạo bởi các đoạn giao nhau liên tiếp. Hai đoạn [*a*,*b*] và [*c*,*d*] được gọi là *giao nhau* nếu chúng có điểm chung. Điều kiện này có nghĩa điểm đầu của đoạn này nằm trong đoạn thứ hai, tức là $a \leq c \leq b$ hoặc $c \leq a \leq d$. Do các đoạn đã được sắp tăng theo điểm đầu *x* nên hai đoạn x_i:d_i và x_j:d_j sẽ giao nhau khi và chỉ khi $x_j \leq x_i + d_i$. Để ý rằng $x_i + d_i$ là điểm cuối của đoạn *i*. Nếu hai đoạn *i* và *j* giao nhau thì ta hợp chúng thành một đoạn [*a*,*b*] với $a = x_i$ và $b = \max(x_i + d_i, x_j+d_j)$. Kết hợp các đoạn giao nhau liên tiếp đến mức tối đa ta thu được một làn gọi là *làn tối đại* [*a*,*b*] có chiều dài $b - a$.

Ta khởi trị làn [*a*, *b*] bằng đoạn đầu tiên x_1:d_1, cụ thể là $a := x_1$, $b := x_1+d_1$ $(= a + d_1)$.

Với mỗi đoạn $i := 2..N$ ta xét:

- Nếu đoạn *i* giao với làn [*a*,*b*], tức là $x_i \leq b$ thì ta hợp đoạn *i* với làn để tạo ra làn mới [*a*,*b*] bằng cách chỉnh $b := \max(b, x_i + d_i)$.

- Nếu đoạn *i* không giao với làn [*a*,*b*] thì ta cộng tích lũy chiều dài của làn [*a*,*b*] hiện có vào biến tổng *t* rồi sinh ra làn mới từ đoạn *i*.

```
t := t + (b – a);
```

$a := x_i ; b := a + d_i;$

Sau khi kết thúc duyệt các đoạn ta cộng nốt làn cuối cùng vào tổng *t* bằng thao tác $t := t + (b - a)$.

Độ phức tạp: $O(N.\log N)$ – chi phí cho sắp xếp Qsort.

(* Pascal *)

```
(************************************
            Xep doan
************************************)
program XepDoan;
uses crt;
const
 bl = #32;  fn = 'DOAN.INP';  mn = 1001;
type KieuDoan = record x: integer; d: integer; end;
     md1 = array[0..mn] of KieuDoan;
var  c: md1; { chua cac doan }
     n: integer;
     f: text;
procedure Doc;
 var i: integer;
begin
  assign(f,fn); reset(f); readln(f,n);
  for i := 1 to n do readln(f,c[i].x,c[i].d);
  close(f);
end;
(*--------------------------------
     Sap tang cac doan theo
     diem dau x
 --------------------------------*)
procedure Qsort(t,p: integer): tự viết;
function max(a,b: integer): tự viết
function Tong: longint;
var t: longint; { tong do dai }
    a, b: integer; { lan [a, b] }
    i: integer;
begin
  t := 0;
  a := c[1].x; b := a + c[1].d; { Khoi tri lan }
  for i := 2 to n do
    if (c[i].x <= b) then b := max(b,c[i].x + c[i].d)
     else
     begin
       t := t + (b - a);
       a := c[i].x; b := a + c[i].d;
     end;
  Tong := t + (b - a);
end;
BEGIN
  Doc; Qsort(1,n); writeln(Tong);
END.
```

// C#

```
using System;
using System.IO;
using System.Collections;
namespace SangTao2 {
```

```
    class XepDoan {
        const string fn = "doan.inp";
        const string gn = "doan.out";
        static public int n; // so luong doan
        static public Doan[] c; // cac doan
        static void Main(string[] args) {
            Doc(); QSort(0, n - 1);
            Console.WriteLine("\n \n Dap so: "+CoverSum());
            Console.WriteLine("\n Fini ");
            Console.ReadLine();
           }
        static public void Doc(): tự viết
        static public int CoverSum() {
            int a = c[0].x, b = a + c[0].d, t = 0;
            for (int i = 1; i < n; ++i)
              if (c[i].x <=  b) b = Max(b, c[i].x + c[i].d);
                 else { t += (b-a); a = c[i].x; b = a+c[i].d; }
            return t + (b - a);
           }
           static public int Max(int a, int b): tự viết
           // Sap cac doan tang theo diem dau x
           static public void QSort(int s, int e): tự viết
    } // XepDoan
    public struct Doan: tự viết
} // SangTao2
```

Bài 1.15 Các hình chữ nhật

ACM

Trên mặt phẳng tọa độ cho N hình chữ nhật (HCN) có diện tích khác 0 và có các cạnh song song với các trục tọa độ. Mỗi HCN được mô tả bằng bộ bốn số nguyên (x_1,y_1) và (x_2,y_2) biểu thị tọa độ nguyên của hai đỉnh đối diện.

Yêu cầu: xác định diện tích phần mặt phẳng bị các HCN phủ.

HCN.INP	HCN.OUT
5 0 0 2 4 1 2 4 6 5 3 3 7 4 1 6 5 7 3 9 0	35

Dữ liệu vào: tệp văn bản **HCN.INP**

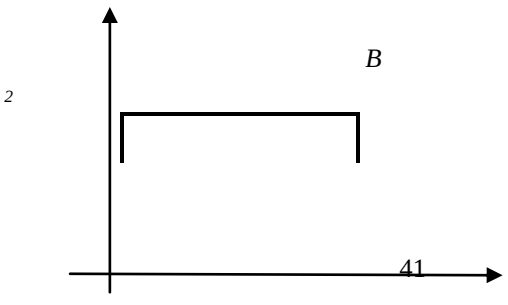
Dòng đầu tiên: số tự nhiên $1 < N \le 1000$.

Dòng thứ i trong N dòng tiếp theo, mỗi dòng chứa 4 số nguyên x_1, y_1, x_2, y_2 cách nhau qua dấu cách.

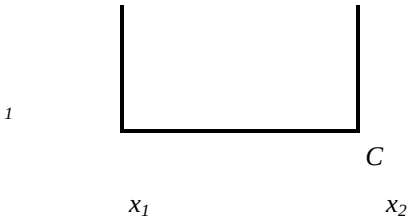
Dữ liệu ra: tệp văn bản **HCN.OUT** *chứa tổng đơn vị diện tích trên mặt phẳng bị các HCN phủ.*

Thuật toán

Phương pháp: *Tham.*



1. Đọc dữ liệu và chỉnh lại các tọa độ sao cho $x1 \le x2$ và $y1 \le y2$. Điều này có nghĩa là ta qui ước mọi HCN *ABCD* đều được xác định qua 2 đỉnh đối diện *D* (đỉnh *Tây-Nam*) và *B* (đỉnh *Đông-Bắc*): *D*(*x1*, *y1*), *B*(*x2*, *y2*).



Khi đọc dữ liệu ta đồng thời lược bớt các giá trị *y* trùng lặp và sắp các giá trị y_1 và y_2 theo chiều tăng để ghi vào một mảng *y*[1..*m*]. Như vậy, giá trị lớn nhất của *m* là 2*n*. Ta gọi phần mặt phẳng giới hạn bởi hai đường thẳng song song với trục hoành và cắt trục tung tại điểm *y*[*i*] và *y*[*i*+1] là *băng i*. Ta có *m*–1 băng mã số từ *1* đến *m*–1. Băng đầu tiên có mã số 1 nằm giữa hai đường *y*[1] và *y*[2], băng cuối cùng có mã số *m*–1 nằm giữa hai đường *y*[*m*–1] và y[*m*]. Nhiệm vụ còn lại là tính diện tích bị phủ trên mỗi băng. Tổng số diện tích bị phủ của các băng sẽ là đáp số cho bài toán.

2. Ta lại sắp các HCN theo chiều tăng của tọa độ *x*1.

3. Với mỗi HCN *h*(*x*1, *y*1, *x*2, *y*2) ta xét các băng *i* trong khoảng từ *y*1 đến *y*2. Với băng *i* giả sử ta đã biết phần bị các HCN 1..(*h*–1) phủ trên băng này. Ta kí hiệu *s*[*i*] và *e*[*i*] là giới hạn hoành độ trái và phải của phần đang bị phủ trên băng *i*. Diện tích hiện bị phủ trên băng *i* sẽ là: (*y*[*i*+1]–*y*[*i*])*(*e*[*i*] – *s*[*i*]), trong đó *y*[*i*+1] – *y*[*i*] là chiều rộng của băng *i*. Ta cần chỉnh lại phần bị phủ trong băng *i* khi xét thêm HCN *h*(*x*1, *y*1, *x*2, *y*2). Dễ thấy, nếu *x*1 nằm giữa *s*[*i*] và *e*[*i*] thì ta cần chỉnh lại *e*[*i*] theo công thức *e*[*i*] := max(*e*[*i*], *x*2). Ngược lại, nếu *x*1 > *e*[*i*] thì ta kết thúc làn (*s*[*i*],*e*[*i*]) này như sau: Tính diện tích làn này và đưa vào biến tích lũy diện tích *dt* sau đó đặt lại cận *s*[*i*] := *x*1; *e*[*i*] := *x*2.

Do các hoành độ *y*1 và *y*2 được sắp tăng nên ta có thể gọi thủ tục tìm kiếm nhị phân BínSearch để xác định băng chứa *y*1.

Độ phức tạp: Các thuật toán sắp xếp và chèn đòi hỏi tối đa N^2, Thủ tục xử lí xét mỗi HCN 1 lần và duyệt 2*n* băng. Tổng hợp: N^2.

```
(* Pascal *)
  (*********************************************
                Cac hinh chu nhat

   *********************************************)
  program HinhChuNhat;
  uses crt;
  const mn = 2001; fn = 'HCN.INP'; gn = 'HCN.OUT';
        bl = #32; nl = #13#10;
  type CN = record
              x1, y1: integer;
              x2, y2: integer;
            end;
        mcn1 = array[0..mn] of CN;
        mi1 = array[0..mn] of integer;
  var
    n,m: integer; { n - so HCN, m - so lan }
    h: mcn1; { cac HCN }
    y: mi1; { ranh gioi cac lan }
    s, e: mi1; { Cac diem dau va cuoi cua bang }
    f,g: text;
    dt: Longint;
  function Max(a,b: integer): tự viết
  (*----------------------------
       Hoán đổi trị để a ≤ b
  ----------------------------*)
  procedure Chinh(var a,b: integer);
    var c: integer;
  begin
    if (a > b) then begin  c := a; a := b; b := c;  end;
```

```
end;
(*-----------------------------------
     Tim nhi phan v trong y[1..m]
-----------------------------------*)
function BinSearch(v: integer): integer;
var d,c,t: integer;
begin
   d := 1 ; c := m;
   while (d < c) do
   begin
     t := (d + c) div 2;
     if (y[t] < v) then d := t + 1
       else c := t;
   end;
   BinSearch := d;
end;
(*----------------------------
   Xen them v vao day
   da sap tang y[1..m]
 ----------------------------*)
procedure Insert(v: integer);
var d: integer;
begin
  if (m = 0) then { danh sach rong }
   begin m := m + 1;  y[m] := v;  exit; end;
  d := BínSearch(v);
  if (y[d] =  v) then exit; { da co trong danh sach }
  if (y[d] < v) then begin m := m + 1; y[m] := v; exit; end;
  move(y[d], y[d+1],sizeof(integer)*(m-d+1));
  y[d] := v;  m := m + 1;
end;
(*-----------------------------
   Doc du lieu va sap theo Y
   luoc bot cac Y bang nhau
  -----------------------------*)
procedure Doc;
var i: integer;
begin
  assign(f,fn); reset(f); readln(f,n); m := 0;
  for i := 1 to n do
    begin
      readln(f,h[i].x1,h[i].y1,h[i].x2,h[i].y2);
      Chinh(h[i].x1, h[i].x2); Chinh(h[i].y1, h[i].y2);
      insert(h[i].y1); insert(h[i].y2);
    end;
  close(f);
end;
procedure SortByX1(d,c: integer);
var i,j,m: integer;
    v: CN;
begin
  i := d; j := c;
  m := h[(i+j) div 2].x1;
  while (i <= j) do
  begin
    while (h[i].x1 < m) do i := i + 1;
    while (m < h[j].x1) do j := j - 1;
```

```
    if (i <= j) then
     begin
        v := h[i]; h[i] := h[j]; h[j] := v;
        i := i + 1; j := j - 1;
     end;
  end;
  if (d < j) then SortByX1(d,j);
  if (i < c) then SortByX1(i,c);
end;
(*----------------------------------
    Xet HCN d, tinh tung phan
    phu trong moi lan
----------------------------------*)
procedure Hinh(x1, x2, y1, y2: integer);
 var i: integer;
begin
  { Tim xuat hien cua y1 trong cac lan }
  i := BinSearch(y1);
  while (y[i] < y2) do
    begin
     if (x1 <= e[i]) then
       e[i] := Max(e[i], x2)
     else
      begin
        dt := dt + (e[i] - s[i]) * (y[i+1] - y[i]);
        s[i] := x1; e[i] := x2;
      end;
     i := i + 1;
   end;
end;
procedure XuLi;
 var d: integer;
begin
  { Khoi tri }
  dt := 0;
  for d := 1 to m-1 do
  begin s[d] := -maxint; e[d] := s[d];  end;
  { Duyet cac HCN }
  for d := 1 to n do
      Hinh(h[d].x1, h[d].x2, h[d].y1, h[d].y2);
  { Tong hop ket qua }
  for d := 1 to m-1 do
     dt := dt +  (e[d] - s[d]) * (y[d+1] - y[d]);
end;
procedure Ghi;
begin
  assign(g,gn); rewrite(g); writeln(g,dt); close(g)
end;
BEGIN
  Doc; SortByX1(1,n);  XuLi;  Ghi;
END.
```

// C#

```
using System;
using System.IO;
using System.Collections;
```

```
namespace SangTao2 {
    class Hcn {
        const string fn = "hcn.inp";
        const string gn = "hcn.out";
        static public int n,m; // n - so luong HCN; m - so bang
        static public HCN[] c;
        static public int dt; // tong dien tich
        static public int[] y;
        static void Main(string[] args) {
            Doc(); QSortByx1(0, n - 1);
            XuLi(); Ghi();  XemKetQua();
            Console.WriteLine("\n Fini ");
            Console.ReadLine();
        }
        static public void Ghi()
        { File.WriteAllText(gn,dt.ToString()); }
        static public void XemKetQua(): tự viết
        static public void XuLi(){
            dt = 0; // tong dien tich
            int[] s = new int[m]; // diem dau cac bang
            int [] e = new int [m]; // diem cuoi cac bang
            for (int i = 0; i < m; ++i)
                s[i] = e[i] = int.MinValue;
            // Duyet cac HCN
            for (int i = 0; i < n; ++i) {   // xu li HCN i
                int sj = BinSearch(c[i].y1);
                int ej = BinSearch(c[i].y2);
                for (int j = sj; j < ej; ++j){   // xet bang j
                    if (c[i].x1 <= e[j]) e[j] = Max(e[j], c[i].x2);
                    else {
                        dt += (e[j] - s[j])*(y[j+1]-y[j]);
                        s[j] = c[i].x1; e[j] = c[i].x2;
                    }
                }
            }
            // Tong hop ket qua
            int m1 = m - 1;
            for (int j = 0; j < m1; ++j)
                dt += (e[j] - s[j])*(y[j+1]-y[j]);
        }
        static public int Max(int a, int b): tự viết
        static public void Doc() {
            int[] v =
             Array.ConvertAll(((File.ReadAllText(fn))).Split(
                         new chae[] { ' ', '\t', '\r', '\n' },
                         StringSplitOptions.RemoveEmptyEntries),
                         new Converter<string, int>(int.Parse));
            int j = 0; n = v[j];
            c = new HCN[n];
            int dx, dy, bx, by, t;
            m = 0;
            y = new int[2*n];
            for (int i = 0; i < n; ++i, j += 4) {
                dx = v[j + 1]; dy = v[j + 2];
                bx = v[j + 3]; by = v[j + 4];
                if (dx > bx) { t = dx; dx = bx; bx = t; }
                if (dy > by) { t = dy; dy = by; by = t; }
```

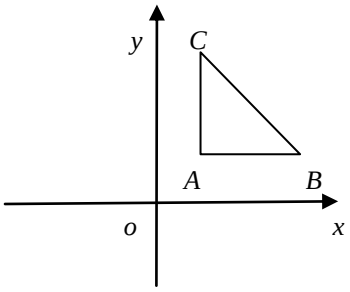
```
                c[i] = new HCN(dx, dy, bx, by);
                Insert(dy); Insert(by);
            }
        }
        // Tim nhi phan gia tri v trong y[0..m-1]
        static public int BinSearch(int v) {
            int left = 0, right = m-1, midle;
            while (left < right) {
                midle = (left + right)/2;
                if (y[midle] < v) left = midle + 1;
                else right = midle;
            }
            return left;
        }
        // Xen toa do v vao danh sach cac lan y
        static public void Insert(int v) {
            if (m == 0) { y[m++] = v; return; }
            int i = BinSearch(v);
            if (y[i] == v) return;
            if (y[i] < v) { y[m++] = v; return; }
            Array.Copy(y, i, y, i + 1, m - i);
            y[i] = v; ++m;
        }
        // Sap cac HCN tang theo x1
        static public void QSortByx1(int s, int e) {
            int i = s, j = e, m = c[(i + j) / 2].x1;
            HCN t;
            while (i <= j) {
                while (c[i].x1 < m) ++i;
                while (c[j].x1 > m) --j;
                if (i <= j) {
                    t = c[i]; c[i] = c[j]; c[j] = t;
                    ++i; --j;
                }
            }
            if (s < j) QSortByx1(s, j);
            if (i < e) QSortByx1(i, e);
        }
    } // Hcn
    public struct HCN {
        public int x1,y1,x2,y2;
        public HCN(int dx, int dy, int bx, int by)
        { x1 = dx; y1 = dy; x2 = bx; y2 = by; }
    }
} // SangTao2
```

Bài 1.16 Các tam giác vuông cân

ACM

Trên mặt phẳng tọa độ cho N tam giác vuông cân, hai cạnh góc vuông song song với hai trục tọa độ, cạnh huyền nằm ở bên phải tam giác. Cụ thể là nếu kí hiệu tam giác là ABC thì ta qui định đỉnh A là đỉnh góc vuông, cạnh góc vuông AB song song với trục hoành ox, cạnh góc vuông AC song song với trục tung oy, AB = AC = d. Mỗi tam giác được mô tả bằng bộ ba số nguyên x, y, d trong đó (x,y) là tọa độ nguyên của đỉnh A, d là chiều dài cạnh góc vuông.

5 6 0 3 1 0 3 2 1 3 4 1 2 4 5 2	16.50



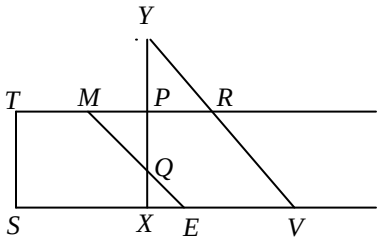
Yêu cầu: Tính diện tích do các tam giác phủ trên mặt phẳng tọa độ.

Dữ liệu vào: text file **TAMGIAC.INP**

Dòng đầu tiên: số tự nhiên N trong khoảng 2 .. 1000.

N dòng tiếp theo: mỗi dòng 3 số x y d cách nhau qua dấu cách, x và y biến thiên trong khoảng –1000..1000, d trong khoảng 1..1000.

Dữ liệu ra: text file **TAMGIAC.OUT** *chứa một số thực duy nhất S là diện tích bị các tam giác phủ trên mặt phẳng tọa độ.*



Thuật toán

Tương tự như bài Các Hình chữ nhật. Với mỗi tam giác (x, y, d) ta tạo ra hai đường song song với trục hoành và cắt trục tung tại các điểm y và y+d, cụ thể là một đường chứa cạnh đáy và một đường đi qua đỉnh của tam giác. Các đường thẳng này sẽ tạo ra các băng. Với mỗi tam giác (TG) ta cũng xét tương tự như HCN, nghĩa là xét các băng chứa trong TG đó. Biến diện tích dt được khai báo kiểu float vì các hình trong băng j sẽ là hình thang có đáy lớn là e[j]–s[j], đáy nhỏ bằng đáy lớn – h, h là chiều cao: h = y[j+1]–y[j] = độ rộng của băng. Khi chuyển từ băng j lên băng j+1 ta phải chỉnh lại đầu phải x2 của cạnh đáy TG thành x2 – h vì TG lúc này sẽ ngắn lại. Khi tính phần giao nhau ta còn phải trừ đi diện tích của TG nằm ngoài phần giao đó. Hình vẽ cho ta thấy khi xét tam giác XYV và băng giới hạn trong 2 đường thẳng TM và SE, thì làn (S, E) sẽ được mở rộng thành làn (S, V). Diện tích trước đó của làn SE chính là diện tích hình thang vuông TMES, nay làn (S,V) có diện tích của hình thang vuông TRVS. Như vậy ta đã tính dôi ra phần diện tích tam giác MPQ. Dễ dàng xác định được diện tích z của tam giác vuông cân này: Đặt k = MP = PQ, ta có z = k^2/2.Ta xác định k như sau. k = MP = TP – TM = SX–TM = x–(e[j]–h), trong đó h là chiều rộng của băng j đang xét, h = y[j+1]–y[j], e[j] là điểm cuối của làn đang xét, x là hoành độ của đỉnh góc vuông trong tam giác XYV đang xét.

Độ phức tạp: N^2.

```
(* Pascal *)
  (*********************************************
            Cac tam giac vuong can
   *********************************************)
  program TamGiacVuongCan;
  uses crt;
  const mn = 2001; bl = #32; nl = #13#10;
```

```
fn = 'TamGiac.inp'; gn = 'TamGiac.out';
type TG = record
            x,y: integer;
            d: integer;
          end;
     mtg1 = array[0..mn] of TG;
     mi1 = array[0..mn] of integer;
     mr1 = array[0..mn] of real;
var
  n,m: integer; { n - so tam giac }
  y: mi1;       { m - so diem y, max(m) = 2n }
  t: mtg1;
  f,g: text;
  dt: real; // tong dien tich
  (*-----------------------------------------
    To chuc du lieu cho Bang i
     s[i], e[i] - can trai va phai cua Bang
  -----------------------------------------*)
  s, e: mi1;
function Max(a,b: integer): tự viết
(*----------------------------
   Xen them v vao day
   da sap tang y[1..m]
 ----------------------------*)
procedure Insert(v: integer); tự viết
procedure Doc;
var i: integer;
begin
  assign(f,fn); reset(f); readln(f,n);
  m := 0;
  for i := 1 to n do
   begin
    readln(f,t[i].x,t[i].y,t[i].d);
    Insert(t[i].y); Insert(t[i].y + t[i].d);
   end;
  close(f);
end;
{ Sap cac TG tang theo x }
procedure SortByX(d,c: integer); tự viết
{ Tim nhi phan v trong y[1..m] }
function BinSearch(v: integer): integer; tự viết
(*--------------------------------------
   Xet hinh Tam giac h: tinh cac o
   hinh h phu tren cac Bang
   --------------------------------------*)
procedure Hinh(x1,y1,d: integer);
 var i,y2,x2,h,k: integer;
begin
  { Tim xuat hien cua toa do y1 trong cac lan }
  i := BinSearch(y1);
  x2 := x1 + d; y2 := y1 + d;
  while (y[i] < y2) do
    begin
     h := y[i + 1] - y[i];
     if (x1 <= e[i]) then
      begin
       k := x1 - (e[i] - h); e[i] := Max(e[i], x2);
```

```
        if (k > 0) then dt := dt - (real(k * k) / 2);
       end
      else
       begin
         if (e[i] > s[i]) then
            dt  := dt + h * (2 * (e[i] - s[i]) - h) / 2;
         s[i] := x1; e[i] := x2;
       end;
      i := i + 1; x2 := x2 - h;
   end;
end;
procedure XuLi;
 var i, h: integer;
begin
 for i := 1 to m do
  begin s[i] := -maxint; e[i] := s[i]; end;
  dt := 0.0;
  { Duyet cac TG }
  for i := 1 to n do Hinh(t[i].x,t[i].y,t[i].d);
  { Tong hop ket qua }
  for i := 1 to m-1 do
    begin
      h := y[i + 1] - y[i];
      if (e[i] > s[i]) then
        dt := dt +  h * (2 * (e[i] - s[i]) - h) / 2;
    end;
end;
procedure Ghi;
begin
  assign(g,gn); rewrite(g); writeln(g,dt:0:2); close(g)
end;
BEGIN
  Doc; SortByX(1,n); XuLi; Ghi;
END.
```

// C#

```
using System;
using System.IO;
using System.Collections;
namespace SangTao2 {
    class TamGiac {
        const string fn = "TamGiac.inp";
        const string gn = "TamGiac.out";
        static public int n, m; // n - so luong HCN; m - so bang
        static public TG[] c;
        static public float dt; // tong dien tich
        static public int[] y;
        static void Main(string[] args){
            Doc(); QSortByx1(0, n - 1); XuLi();
            Ghi(); XemKetQua();
            Console.WriteLine("\n Fini "); Console.ReadLine();
        }
        static public void Ghi(): tự viết
        static public void XemKetQua(): tự viết
        static public void XuLi() {
            dt = 0F;
```

```
        int[] s = new int[m];
        int[] e = new int[m];
        for (int i = 0; i < m; ++i)
            s[i] = e[i] = int.MinValue;
        // Duyet cac TG c[i]
        for (int i = 0; i < n; ++i) {
            int x2 = c[i].x1+c[i].d; // dau phai cua canh day
            int sj = BinSearch(c[i].y1);
            int ej = BinSearch(c[i].y1 + c[i].d);
            for (int j = sj; j < ej; ++j) {
                // xet bang j
                int h = y[j + 1] - y[j]; // do rong cua bang j
                if (c[i].x1 <= e[j]) {
                    int k = c[i].x1 - (e[j] - h);
                    if (k > 0) dt -= (float)(k * k) / 2;
                    e[j] = Max(e[j], x2);
                }
                else {
                  if (e[j] > s[j])
                   dt += (float)(2*(e[j]-s[j])-h)*h/2;
                    s[j] = c[i].x1; e[j] = x2;
                }
                x2 -= h;
            } // xong bang j
        } // xong TG i
        // Tong hop ket qua
        int m1 = m - 1;
        for (int j = 0; j < m1; ++j) {
            int h = y[j+1]-y[j]; // do rong cua bang
            if (e[j] > s[j])
             dt += (float)(2*(e[j]-s[j])-h)* h/2;
    }
    static public int Max(int a, int b): tự viết
    static public void Doc() {
        int[] v =
          Array.ConvertAll(((File.ReadAllText(fn))).Split(
                    new char[] { ' ', '\t', '\r', '\n' },
                    StringSplitOptions.RemoveEmptyEntries),
                    new Converter<string, int>(int.Parse));
        int j = 0; n = v[j];
        c = new TG[n];
        m = 0;
        y = new int[2 * n];
        for (int i = 0; i < n; ++i, j += 3) {
            Insert(v[j + 2]); Insert(v[j + 2] + v[j + 3]);
            c[i] = new TG(v[j + 1], v[j + 2], v[j + 3]);
        }
    }
    // Tim nhi phan gia tri v trong y[0..m-1]
    static public int BinSearch(int v): tự viết
    // Xen toa do v vao danh sach cac lan y
    static public void Insert(int v): tự viết
    // Sap cac TG tang theo x1
    static public void QSortByx1(int s, int e): tự viết
} // TamGiac
public struct TG {
    public int x1, y1, d;
```

```
        public TG(int dx, int dy, int dd)
        { x1 = dx; y1 = dy; d = dd; }
    }
} // SangTao2
```

Chương 2

Các hàm Next

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

```
(* Pascal *) type mi1 = array[0..MN] of integer;
```

trong đó MN là hằng số đủ lớn cho kích thước mỗi bài toán, thí dụ

```
const MN = 2000;
```

Trong C# mảng được khai báo trực tiếp hoặc thông qua class, thí dụ,

```
int [] a = new int [2000];
Class Array { … };
```

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

Bài 2.1 Số sát sau cùng độ cao

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

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

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

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

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

Dữ liệu ra: tệp văn bản **DOCAO.OUT**

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

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

Thuật toán

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

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

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

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

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

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

Độ phức tạp: cỡ N, do mỗi chữ số của x được thăm và xử lí không quá 2 lần.

```
(* Pascal *)
  function Next(var x: mi1; n,b: integer): Boolean;
     var i,j,t,b1: integer;
   begin
     Next := FALSE;
     x[0] := b; j := n;
     while (x[j] = 0) do j := j - 1;
     if (j = 0) then exit; { ko co so sat sau }
     i := j - 1; b1 := b - 1 ;
     while (x[i] = b1) do i := i - 1;
     if (i = 0) then exit; { Ko co so sat sau }
     x[j] := x[j] - 1; x[i] := x[i] + 1;
     i := i + 1; j := n;
     { Lat doan x[i..n] }
     while (i < j) do
       begin
         t := x[i]; x[i] := x[j]; x[j] := t;
         i := i + 1; j := j - 1;
       end;
     Next := TRUE;
   end;
```

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

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

Bài 2.2 Số sát sau cùng chữ số

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

NXT.INP	NXT.OUT
6 239521	1 251239

Dữ liệu vào: tệp văn bản NXT.INP

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

Dòng thứ hai: số x

Dữ liệu ra: tệp văn bản NXT.OUT

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

Thuật toán

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

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

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

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

- Đổi chỗ x_i và x_j,

- Lật lại đoạn x[i+1..n].

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

Sau khi hoán vị x[i] và x[j] ta thu được, x = (2,**5**,9,**3**,2,1)

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

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

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

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

3. Hoán vị x[i] và x[j],

4. Lật đoạn x[i+1..N].

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

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

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

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

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

Bài 2.3 Các hoán vị

Olimpic Moscva

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

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

Dữ liệu ra: tệp văn bản `HV.OUT`

Mỗi dòng một hoán vị.

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

Thuật toán

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

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

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

(* Pascal *)

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

```
  Run;
END.
```

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

```
        }
      } // CacHoanVi
  } // SangTao2
```

Bài 2.4 Tổ hợp

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

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

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

Dòng đầu tiên: hai số N và K cách nhau qua dấu cách, $1 \le N \le 9$, $K \le N$.

Dữ liệu ra: tệp văn bản **TOHOP.OUT**

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

Thuật toán

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

Pha 1. Dỡ. Duyệt ngược từ K qua trái bỏ qua những phần tử mang giá trị …N–2, N–1, N đứng cuối mảng. Nếu sau khi dỡ x không còn phần tử nào thì kết thúc với Next = false với ý nghĩa là sát sau tổ hợp x không còn tổ hợp nào. Thí dụ, nếu N = 7, K = 5, x[1..5] = (2,3,5,6,7) thì sau khi dỡ ba phần tử cuối của x ta thu được i = 2, x[1..2] = (2,3). Điều này cho biết sẽ còn tổ hợp sát sau.

Pha 2. Xếp.

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

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

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

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

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

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

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

Sau đó mỗi lần gọi hàm Next ta kiểm tra

Nếu v = 0 thì dừng hàm Next.

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

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

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

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

```
//  C#
  using System;
```

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

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

Bài 2.5 Số Kapreka

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

KAPREKA.INP	KAPREKA.OUT
10 4	6174

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

Dòng đầu tiên: hai số B và K cách nhau qua dấu cách, $2 \le B \le 10$, $K < B$.

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

Số x viết trong hệ đếm B.

Bộ dữ liệu trên cho biết: Trong hệ đếm thập phân (B = 10), x = 6174 là số Kapreka có 4 chữ số (khác nhau đôi một), x'' - x' = 7641 – 1467 = 6174 = x.

Thuật toán

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

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

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

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

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

Sau đó ta bỏ đi số nhớ cuối cùng.

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

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

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

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

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

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

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

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

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

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

NN	B	K	Đáp số	NN	B	K	Đáp số	NN	B	K	Đáp số
1	4	3	132	6	8	2	25	11	9	7	0
2	5	4	0	7	8	3	374	12	9	8	0
3	6	3	253	8	8	7	6417532	13	10	3	495

4	6	4	0	9	9	5	62853	14	10	4	6174
5	6	5	41532	10	9	6	0	15	10	9	864197532

15 thí dụ về các số Kapreka

(* Pascal *)

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

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

// C#

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

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

Chú thích

Bạn có thể sử dụng thuật toán sau đây:

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

Bài 2.6 Khóa vòng

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

KHOA.INP	KHOA.OUT
1 2 B C 2 3 0 2	12 B20 B21 B22 B30 B31 B32 C20 C21 C22 C30 C31 C32

Dữ liệu vào: tệp văn bản **KHOA.INP**

Dòng đầu tiên: hai số tự nhiên M và N, $1 \le M, N \le 5$.

Dòng thứ i trong số M+N dòng tiếp theo: giới hạn a_i và b_i cho các vòng khóa.

Dữ liệu ra: tệp văn bản **KHOA.OUT**

Dòng đầu tiên: Tổng số khả năng.

Từ dòng thứ hai trở đi: mỗi dòng một giá trị khóa liệt kê tăng dần theo trật tự từ điển. Các kí tự chữ và số trong mỗi khóa được viết liền nhau, không có dấu cách ở giữa. Các giá trị chữ được lấy từ bảng chữ HOA tiếng Anh.

Thuật toán

Phương pháp: duyệt toàn bộ các tổ hợp.

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

```
Khởi trị: c[1..N] là tổ hợp nhỏ nhất chứa toàn cận dưới:
  c[i] := a[i], i = 1..N.
Xử lí:
repeat
  Ghi tổ hợp c[1..N];
until not Next;
```

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

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

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

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

```
bl = #32; nl = #13#10; fn = 'KHOA.INP'; gn = 'KHOA.OUT';
ChuCai = ['A'..'Z'];
type mi1 = array[0..mn] of integer;
var m,n: integer;
   a,b,c: mi1;
   f,g: text;
   m: longint;
procedure Doc;
  var i: integer;
  c: char;
begin
  assign(f,fn); reset(f); readln(f,m,n);
  n := m + n;
  for i := 1 to m do
    begin
      repeat
        read(f,c);
      until c in ChuCai;
      a[i] := ord(c) - ord('A');
      repeat
        read(f,c);
      until c in ChuCai;
      b[i] := ord(c) - ord('A');
    end;
  for i := m + 1 to n do read(f,a[i],b[i]);
  close(f);
  m := 1;
  for i := 1 to n do m := m * (b[i] - a[i] + 1);
end;
function Min(a,b: integer): integer;
begin
  if (a < b) then Min := a else Min := b;
end;
function Next: Boolean;
  var i: integer;
begin
  Next := false;
  i := n;
  while (c[i] = b[i]) do
    begin
      c[i] := a[i];
      i := i - 1;
    end;
  if (i = 0)  then exit;
  c[i] := c[i] + 1;
  Next := true;
end;
procedure Duyet;
  var i: integer;
begin
  for i := 1 to n do c[i] := a[i];
  c[0] := -1;
  assign(g,gn); rewrite(g); writeln(g,m);
  repeat
    for i := 1 to m do write(g,chr(ord('A')+c[i]));
    for i := m + 1 to n do write(g,c[i]);
    writeln(g);
```

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

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

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

Bài 2.7 Trả tiền

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

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

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

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

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

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

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

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

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

Thuật toán

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

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

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

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

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

```
(*  Pascal  *)
  (***************************************
                 Tra tien
  ***************************************)
  program TraTien;
  uses crt;
  const mn = 20; bl = #32; nl = #13#10;
    fn = 'TRATIEN.INP'; gn = 'TRATIEN.OUT';
  type mi1 = array[0..mn] of integer;
```

```
(*----------------------------------
    n - so luong cac loai tien
    v - so tien can tra lai
    vt - gia tri tam thoi
    m[1..n] - cac menh gia
    s[1..n] - so luong to tien
    c[1..n] - so luong can chon
----------------------------------*)
var n,v,vt: integer;
   m,s,c: mi1;
   f,g: text;
procedure Doc;
  var i: integer;
begin
  assign(f,fn); reset(f); readln(f,n,v);
  for i := 1 to n do read(f,m[i]);
  for i := 1 to n do read(f,s[i]);
  close(f);
end;
function Min(a,b: integer): tự viết
function Next: Boolean;
  var i: integer;
begin
  Next := false;
  i := n;
  while (c[i] = s[i]) do
    begin
      vt := vt - c[i] * m[i];
      c[i] := 0;
      i := i - 1;
    end;
  if (i = 0)  then exit;
  c[i] := c[i] + 1;
  vt := vt + m[i];
  Next := true;
end;
function Duyet: Boolean;
  var i: integer;
begin
  { Khoi tri }
  for i := 1 to n do
   begin
     s[i] := min(s[i],v div m[i]);
     c[i] := 0;
   end;
  c[0] := -1;  vt := 0; { tong gia tri cua 1 phuong an }
  Duyet := true;
  repeat
    if (vt = v) then exit;
  until not Next;
  Duyet := false;
end;
procedure Run;
  var i: integer;
begin
  Doc;  assign(g,gn);  rewrite(g);
  if (Duyet) then
```

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

// C#

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

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

Bài 2.8 Dãy Farey

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

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

- m biến thiên trong khoảng 1..N.

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

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

Dữ liệu ra: tệp văn bản **FAREY.OUT**

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

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

Thuật toán

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

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

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

a/b = min { x/y | x/y > t/m, y = 1..n, x ≤ y, (x,y) = 1 }

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

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

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

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

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

- 1 ≤ m ≤ n

- x/y lầ PS đầu tiên lớn hơn t/m.

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

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

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

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

Bình luận

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

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

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

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

Ta có thuật toán sau:

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

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

m = 2: thêm các PS trung bình với mẫu bằng 2: 0/1, 1/2 , 1/1.

m = 3: thêm các PS trung bình với mẫu bằng 3: 0/1, 1/3, 1/2, 2/3, 1/1.

...

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

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

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

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

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

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

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

Từ tính chất 4 ta suy ra ngay cách xác định PS t3/m3 thông qua hai PS sát trước.

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

(* Pascal *)

```
(*-----------------------------------*
     Ba phuong an cho bai Day Farey
*-----------------------------------*)
uses crt;
const bl = #32; nl = #13#10;
var n: integer;
{ Uoc chung lon nhat cua hai so tu nhien a, b }
function Ucln(a,b:integer):integer;
```

```
  var r: integer;
 begin
   while b > 0 do begin r := a mod b; a := b; b:=r end;
   Ucln:=a;
 end;
{ Rut gon PS a/b thanh PS t/m }
procedure RutGon(a,b:integer; var t,m:integer);
var d:integer;
begin d :=Ucln(a,b); t := a div d; m := b div d; end;
{ Tim PS sat sau PS t/m, ket qua dat trong t/m }
function Next(n: integer; var t,m: integer): Boolean;
var a,b,x,y: integer;
begin
  if (t+m=2) then begin Next := false; exit end;
  a := 1; b := 1;
  for y := 2 to n do
  begin
    x := t*y div m + 1;
    if a*y > b*x then begin a := x; b:=y end;
  end;
  RutGon(a,b,t,m);  Next := true;
end;
procedure Farey1(n: integer);
 var t,m,d:integer;
begin
 writeln(nl,'Farey1'); d := 0;
 t := 0; m := 1;
 repeat
   write(t,'/',m,bl); inc(d);
 until not Next(n,t,m);
 writeln(nl,'Total: ',d,' PS');
 readln;
end;
procedure Farey2(n: byte);
const mn = 1000;
type PS = record tu,mau: byte end;
     mps1 = array[0..mn] of PS;
var a,b: mps1; { 2 day PS a , b }
    d,k,i,m:integer;
begin
 writeln(nl,'Farey2'); d := 2;
 a[1].tu := 0; a[1].mau := 1; { PS dau day }
 a[2].tu := 1; a[2].mau := 1; { PS thu hai }
for m:=2 to n do
 begin
   k := 0; inc(k); b[k] := a[k];
   for i := 2 to d do
   begin
     if a[i].mau+a[i-1].mau = m then
     begin
       inc(k); b[k].tu := a[i-1].tu + a[i].tu;
       b[k].mau := a[i-1].mau + a[i].mau;
     end;
     inc(k); b[k] := a[i];
   end;
   a := b; d := k;
 end;
```

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

// C#

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

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

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

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

Bài 2.9 Qúy Mùi

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

Thuật toán

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

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

Bước 2. Thêm Q: Viết thêm kí tự 'Q' vào mọi dãy của T(n–1).

Bước 3. Thêm M: Viết thêm kí tự 'M' vào mọi dãy của T*(n–1).

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

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

Thiết kế các thiếp T(1), T(2) và T(3)

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

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

```
(*  Pascal  *)
```

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

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

Độ phức tạp. n.

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

Mã Gray

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

Các tính chất của mã Gray

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

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

Mã Gray và giá trị của hàm Gline của 16

số tự nhiên đầu tiên k = 0..15.

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

Bước 1. Tính x = Gray(k) = (k shr 1) xor k = (k div 2) xor k.

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

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

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

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

Bài 2.10 Tổng đoạn

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

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

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

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

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

Dữ liệu ra: tệp văn bản **TDOAN.OUT**

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

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

Thuật toán

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

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

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

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

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

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

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

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

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

Bài 2.11 Đoạn không giảm dài nhất

Dijkstra E.

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

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

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

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

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

Dữ liệu ra: tệp văn bản **MDOAN.OUT**

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

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

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

1 5 5 <u>1</u> <u>3</u> <u>3</u> <u>3</u> <u>5</u> <u>7</u> <u>9</u> 1 2

Thuật toán

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

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

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

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

Độ phức tạp: cỡ N.

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

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

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

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

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

Bài 2.12 Đoạn đơn điệu dài nhất

Dijkstra E.

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

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

Dữ liệu vào: tệp văn bản DONDIEU.INP

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

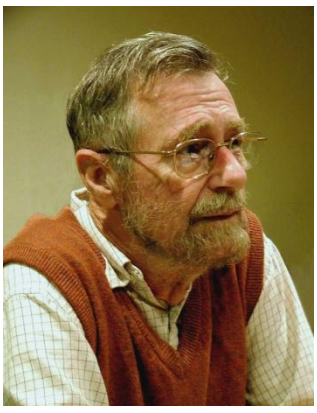
Từ dòng thứ hai trở đi: các phần tử của dãy.

Dữ liệu ra: tệp văn bản DONDIEU.OUT

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

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

Thuật toán



Edsger Wybe Dijkstra (1930-2002) Sinh năm 1930 tại Rotterdam, Holland. 1948-1956 học Toán và Vật lý lý thuyết tại Đại học Leyden. 1952-1962 nghiên cứu tại Trung tâm Toán học Amsterdam. 1962-1973 Giáo sư Toán tại Đại học Bách khoa Eindhoven, Holland và Đại học Texas Austin. Dijkstra là một trong những người đi tiên phong trong lĩnh vực lập trình, người khởi xướng và đặt nền móng cho nguyên lý lập trình cấu trúc.

Edsger Wybe Dijkstra

(*photo* ©2002 *Hamilton Richards*)

Nhận xét:

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

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

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

Độ phức tạp: cỡ N.

Các biến tổng thể:

n: số lượng phần tử,

dt: đếm số phần tử trong dãy tăng,

dg: đếm số phần tử trong dãy giảm.

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

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

(* Pascal *)

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

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

// C#

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

Bài 2.13 Lũy thừa 2, 3 và 5

Dijkstra E.

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

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

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

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

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

N số tìm được, mỗi dòng một số.

Thuật toán

Gọi S là tập các số cần tìm. Ta có

(i) $1 \in S$

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

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

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

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

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

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

```
(*  Pascal  *)
  (**************************************
          LUY THUA cua 2, 3, 5
    ************************************)
  program LuyThua;
  uses crt;
  const bl = #32; mn = 1001; fn = 'LUYTHUA.INP'; gn = 'LUYTHUA.OUT';
  type ml1 = array[0..mn] of longint;
  var f,g: text;
  n: integer;
  a: ml1;
  procedure Doc: tự viết;
  function Min(a,b,c: longint): tự viết;
  function Next(i: integer): tự viết;
  procedure Sinh;
   var i: longint;
  begin
    assign(g,gn); rewrite(g);
```

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

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

Chương 3

Trò chơi

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

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

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

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

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

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

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

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

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

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

Trước hết cần xác định được một tính chất ***T*** *thỏa các điều kiện sau đây:* *a) Thế thua cuối cùng thỏa* ***T,*** *b) Mọi nước đi luôn luôn biến* ***T*** *thành* ***V*** *= not* ***T,*** *c) Tồn tại một nước đi để biến* ***V*** *thành* ***T.***
Tính chất ***T*** *được gọi là bất biến thua của trò chơi.*

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

T *được gọi là bất biến thua nếu* *a') Thế thua cuối cùng thỏa* ***T****,* *b') Mọi nước đi từ* ***T*** *đều lật* ***T*** *thành* ***V****,* *c') Tồn tại một nước đi để lật* ***V*** *thành* ***T****.*	*Đấu thủ nào có cách đẩy đấu thủ khác vào thế thua T thì đấu thủ đó sẽ thắng.* *Đấu thủ nào không thể đẩy đấu thủ khác vào thế thua T thì đấu thủ đó sẽ thua.*

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

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

Bài 3.1. Bốc sỏi A

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

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

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

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

Thí dụ, với M = 3 cho trước và cố định, A là đấu thủ đi trước, ta có

N = 0 là một thế thua, vì A không có cách đi.

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

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

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

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

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

Qui tắc xác định thế thắng / thua
Từ một thế X đang xét, • *nếu tìm được một nước đi dẫn đến thế thua T thì X sẽ là thế thắng V, và* • *nếu mọi nước đi từ X đều dẫn đến thế thắng thì X sẽ là thế thua T.*

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

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

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

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

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

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

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

$$T(N = 0) \leftarrow V(N = 1 \mid 2 \mid 3) \leftarrow T(N = 4)\ldots$$

Đến đây ta có thể dự đoán bất biến thua sẽ là N = 4k, cho trường hợp M = 3, hoặc tổng quát hơn, N = k(M+1), k ≥ 0.

Vậy bất biến thua là:

T: Số viên sỏi trong đống là bội của M+1: N = k(M+1), k ≥ 0.

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

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

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

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

Kết luận Bài Bốc sỏi A
Nếu số sỏi N = k(M+1), k ≥ 0 *thì đấu thủ nào đi trước sẽ thua.*

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

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

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

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

```
    CachDi := r;
end;
```

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

Bài 3.2. Bốc sỏi B

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

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

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

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

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

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

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

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

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

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

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

T(0) = 1;

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

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

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

...

Dự đoán: T(k) = k(M+1)+1.

Chứng mnh bất biến thua: Nếu số sỏi trong đống là T(k) = k(M+1)+1, k ≥ 0 thì ai đi trước sẽ thua.

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

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

Kết luận Bài Bốc sỏi B
Nếu số sỏi N = k(M+1)+1, k ≥ 0 thì đấu thủ nào đi trước sẽ thua.

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

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

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

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

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

Bài 3.3. Bốc sỏi C

Cho đống sỏi N viên, hai đấu thủ A và B lần lượt đi, A đi nước đầu tiên. Tại mỗi nước đi, đấu thủ buộc phải bốc tối thiểu 1 quân, tối đa nửa số quân trong đống. Đấu thủ nào đến lượt mình không đi nổi thì thua. Cả hai đấu thủ đều chơi rất giỏi. Cho biết A thắng hay thua.

Chú ý:

- *Nếu số quân lẻ thì bốc nửa non,*
- *Đống nào còn 1 quân thì không có cách bốc ở đống đó, vì 1 div 2 = 0 trong khi yêu cầu của luật chơi là phải bốc tối thiểu 1 quân.*

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

N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	…
A thắng (1) hay thua (0)?	0	1	0	1	1	1	0	1	1	1	1	1	1	1	0	1	…
Cách đi: số sỏi cần bốc để chắc thắng	#	1	#	1	2	3	#	1	2	3	4	5	6	7	#	1	…

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

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

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

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

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

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

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

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

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

Tổng hợp lại ta có

T(0) = 1,

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

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

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

...

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

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

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

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

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

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

Kết luận Bài Bốc sỏi C
Nếu số sỏi N có dạng 2^k - 1, k ≥ 1 thì đấu thủ nào đi trước sẽ thua.

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

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

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

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

```
(* Pascal *)
  function Ket(N : integer) : integer;
  var m,n1: integer;
  begin
     n1 := N + 1; m := 1;
     while (m < n1) do m := m shl 1;
    { m = 2^k ≥ n1 = N+1 ==> N ≤ 2^k-1 }
     if m = n1 then Ket := 0 else Ket := 1;
  end;
```

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

```
(* Pascal *)
  function CachDi(N : integer) : integer;
  var m, n1: integer;
  begin
     n1 := N + 1; m := 1;
     while (m < n1) do m := m shl 1;
    { m = 2^k ≥ n1 = N+1 ==> N ≤ 2^k-1 }
```

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

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

Bài 3.4. Chia đoạn

Dạng phát biểu khác của Bài Bốc sỏi C

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

Bài 3.5. Bốc sỏi D

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

Thuật toán

Bài này khá dễ giải.

Bất biến thua cho Bài Bốc sỏi D
Số sỏi của hai đống bằng nhau.

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

Ta cũng dễ dàng viết được hàm kết như sau:

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

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

```
(* Pascal *)
procedure CachDi(N, M : integer; var D,S : integer);
  { Dong 1: N vien, Dong 2: M vien soi }
begin
   if N = M then { Se Thua }
      begin
         if N = 0 then D := 0 { Het soi: dau hang }
           else
           begin { Keo dai cuoc choi }
            S := 1; { boc 1 vien }
            D := random(2)+1;{tai 1 dong tuy chon}
           end;
         exit;
      end;
   { Tinh huong thang }
    if N > M then D := 1 else D := 2; { Chon dong nhieu soi }
    S := abs(N-M); { Boc so soi chenh lech }
end;
```

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

Bạn thử nghĩ

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

Bài 3.6. Bốc sỏi E

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

Thuật toán

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

		M						
		⓿	❶	❷	❸	❹	❺	…
N	⓪	1	0	1	1	1	1	…
	①	0	1	1	1	1	1	…
	②	1	1	0	1	1	1	…
	③	1	1	1	0	1	1	…
	④	1	1	1	1	0	1	…
	⑤	1	1	1	1	1	0	…
…								

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Bài 3.7. Bốc sỏi F

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

Thuật toán

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

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

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

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

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

Vậy

Bất biến thua cho Bài Bốc sỏi F
Số sỏi trong hai đống thỏa hệ thức $(N+1) = 2^k(M+1), k \geq 0$ *(*)*

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

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

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

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

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

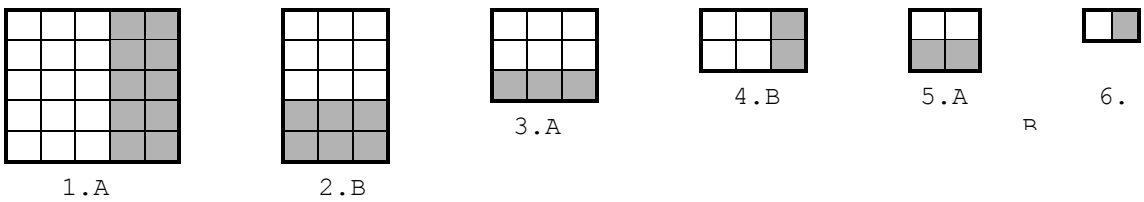
```
procedure CachDi(N,M : integer; var D,S : integer);
    var N1,M1,t: integer;
begin
   { N = M = 1: dau hang }
   if (N = M) and (N = 1) then
    begin
       D := 0; S := 0;
       exit;
    end;
   { Chon dong nhieu soi }
   if N > M then D := 1 else D := 2;
   N1 := N+1; M1 := M+1;
   if N1 <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

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

Bài 3.8. Chia Hình chữ nhật

Olimpic Quốc tế

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



Với hình chữ nhật 5×5 đấu thủ A sẽ thua sau 6 nước đi.
Sau mỗi lần cắt, mảnh trắng có diện tích lớn hơn
sẽ được giao cho đấu thủ tiếp theo,
mảnh xám sẽ được bỏ đi.

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

Bài 3.9. Bốc sỏi G

(Dạng tổng quát).

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

Bài 3.10. Chia Hình hộp

(Cho trường hợp 3 đống sỏi)

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

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

Cơ sở toán học

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

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

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

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

Thí dụ,

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

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

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

Cho 2 số tự nhiên n và m. Xét hệ thức

$$n + 1 = 2^k (m + 1), k = 0, 1, 2, \ldots \qquad (*)$$

Mệnh đề 2

a) Hai số Mersenne bất kì đều thỏa hệ thức (*).

b) Có những cặp số tự nhiên thỏa hệ thức (*) nhưng không phải là số Mersenne.

c) Nếu cặp số tự nhiên n và m thỏa hệ thức (*) và một trong hai số đó là số Mersenne thì số kia cũng phải là số Mersenne.

Gợi ý

a) $n= 2^a-1$, $m = 2^b-1$, $a \ge b \Rightarrow n+1 = (m+1).2^{a-b}$.

b) Thí dụ, 5 và 11: $(11+1) = (5+1).2^1$, Với mọi a, b nguyên: $5 \ne 2^a-1$, $11 \ne 2^b-1$.

c) $n+1 = (m+1).2^k$, $n = 2^a-1 \Rightarrow 2^a = (m+1).2^k$ hay $m = 2^{a-k} - 1$.

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

Thí dụ,

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

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

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

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

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

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

Bài 3.11. Trò chơi NIM

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

Thuật toán

Gọi số viên sỏi trong các đống là $S_1, S_2, \ldots, S_N$.

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

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

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

2) $a \oplus 0 = a$

3) $a \oplus 1 = not\ a$

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

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

6) Tính lũy linh: $a \oplus a = 0$

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

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

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

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

	Dạng nhị phân			
S1 = 13	1	1	0	1
S2 = 14	1	1	1	0
S3 = 6	0	1	1	0
S4 = 7	0	1	1	1
S5 = 2	0	0	1	0
⊕ x = 0	0	0	0	0

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

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

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

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

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

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

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

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

Ta phát hiện được qui luật lí thú sau đây:

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

Chứng minh

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

Ta chứng minh tiếp các mệnh đề sau:

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

Chứng minh

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

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

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

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

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

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

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

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

Chứng minh

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

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

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

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

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

Thủ tục **CachDi** hoạt động như sau:

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

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

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

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

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

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

Bài 3.12. Cờ bảng

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

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

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

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

	❶	❷	❸	❹
①		A_3		
②				
③	B_2			
④			A_1	
⑤				
⑥				
⑦		@		
⑧				

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

1. A chuyển @ từ vị trí @(7,2) sang vị trí A_1(4,3)
2. B chuyển @ từ vị trí A_1(4,3) sang vị trí B_2(3,1)
3. A chuyển @ từ vị trí B_2(3,1) sang vị trí A_3(1,2)

B chịu thua vì hết cách đi!

Thuật toán

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

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

Nhận xét
Nếu [i, j] = 0 thì mọi ô trên dòng i+j, trừ ô (i+j, j) đều nhận trị 1.
[i, j] = 0 ⇒ [i+j, k] = 1, k = 1..M, k ≠ j.

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

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

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

Thuật toán trên đòi hỏi độ phức tạp cỡ $N.M^2$.

	❶	❷	❸	❹
①	0	0	0	0
②	0	0	0	0
③	0	0	0	0
④	0	0	0	0
⑤	0	0	0	0
⑥	0	0	0	0

❶	❷	❸	❹
0	0	0	0
0	1	1	1
1	1	1	1
1	1	0	1
1	1	1	0
0	0	0	0

⑦	0	0	0	0
⑧	0	0	0	0

1	1	1	1
0	0	0	0

Lấp đầy 0 (bảng trái) rồi điền trị (bảng phải)
cho cờ bảng N = 8, M = 4.
Kết quả với x = 7, y = 2: a[7,2] = 1 (A thắng).

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

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

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

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

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

sx = 1, hoặc

sx = 2 và sy = 1.

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

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

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

a[nx, ny] = 0,

nx càng nhỏ càng tốt.

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

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

```
end;
```

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

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

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

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

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

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

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

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

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

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

//C#

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

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

Nguyên tắc của các trò chơi đối kháng

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

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

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

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

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

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

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

	1	2	3	4
1	0	0	0	0

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

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

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

Tab Game với N = 12, M = 4.

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

```
function Ket(M,x,y: integer): integer;
 var i, j: integer;
begin
  fillchar(a,sizeof(a), 0);
  for i := 1 to x do
    for j := 1 to M do
      if (a[i,j] = 0) then
        begin
          TinhNuocThua(M,i,j);
          TinhNuocThang(M,x,i,j);
        end;
   Ket := a[x,y];
end;
Procedure TinhNuocThua(M,i,j: integer);
 var k, vmax: integer;
begin
  vmax := -1;
  for k := 1 to Min(i-1,M) do
    if (k <> j) then
      vmax := Max(vmax, a[i-k,k]);
  a[i,j] := -(vmax + 1);
end;
Procedure TinhNuocThang(M,x,i,j: integer);
 var k, d, v: integer;
begin { Xet dong i+j }
  d := i+j;
  if (d <= x) then { quan co con trong vung can xu ly }
    begin
     v := -a[i,j] + 1;
     for k := 1 to M do
       if (k <> j) then
         if (a[d,k] > 0) then a[d,k] := Min(a[d,k], v)
          else a[d,k] := v;
    end;
end;
```

// C#

```
    static int Ket(int n, int m, int x, int y) {
              Array.Clear(a, 0, a.Length);
              for (int i = 1; i <= x; ++i) { // voi moi dong i
                  for (int j = 1; j <= m; ++j) // xet cot j
```

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

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

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

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

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

Bài 3.13. Cờ đẩy

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

a) A thắng (ghi 1) hay thua (ghi 0).

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

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

Phương án 1: (1, 2, 3).

Phương án 2: (1, 3, 2).

Phương án 3: (1, 4, 1).

Thuật toán

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

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

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

Bài 3.14. Bốc sỏi H

Dạng phát biểu khác của bài Cờ đẩy

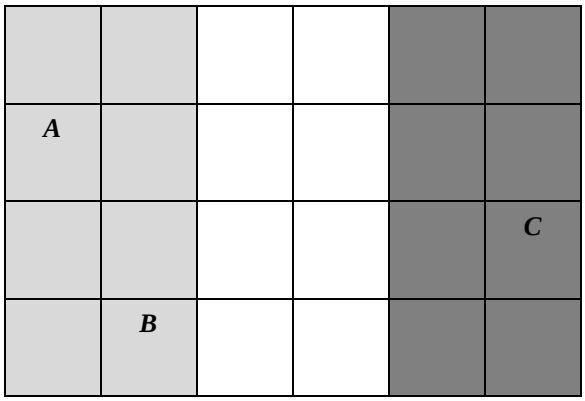
Cho đống sỏi N viên, hai đấu thủ A và B lần lượt đi, A đi nước đầu tiên. Mỗi nước đi đấu thủ buộc phải bốc tối thiểu 1 viên, tối đa M viên trong đống và không được lặp lại nước vừa đi của người trước. Thí dụ, nếu đấu thủ A vừa bốc v viên sỏi thì đến lượt mình, đấu thủ B không được bốc v viên nữa. Đấu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

Chương 4

Các thuật toán sắp đặt

4.1 Cờ tam tài

Olimpic quốc tế



Cờ tam tài 4 × 6

N = 2

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

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

	❶	❷	❸	❹	❺	❻
①	A1:X	A2:X	A3:T	A4:T	C3:D	C2:D
②	A0:X	A1:X	A2:T	A3:T	C2:D	C1:D

	Xanh	Trắng	Đỏ
A	5	4	0

③	A1:X	B1:X	B2:T	C2:T	C1:D	C0:D
④	B1:X	B0:X	B1:T	B2:T	C2:D	C1:D

Cờ tam tài, N = 2, A(2,1), B(4,2), C(3,6)
X: Xanh, T: Trắng, D: Đỏ.

B	3	3	0
C	0	1	8

Kết quả

Thuật toán

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

$$d = abs(i-x) + abs(j-y) + 1$$

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

$$d = abs(i-x) + abs(j-y)$$

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

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

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

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

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

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

Các biến chung của chương trình sẽ là:

```
var
n: integer; { Ban co co kich thuoc 2n×3n }
Ax,Ay,Bx,By,Cx,Cy: integer; { Toa do xuat phat cua A, B, C }
kq: array['A'..'C',0..2] of integer; { Chua ket qua }
```

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

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

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

Chương trình C#

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

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

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

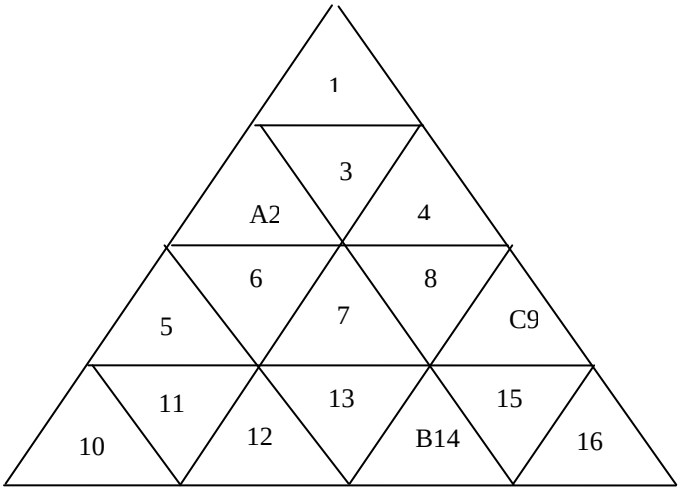
Độ phức tạp

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

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

4.2 Lưới tam giác đều

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



Lưới Tam giác N = 4, NA = 2, NB = 14,
NC = 9.
Kết quả, A: 9, B: 5, C: 2.

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

Tổ chức dữ liệu

Các biến dùng chung:

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

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

Thuật toán

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

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

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

Nếu một TGĐV có đỉnh quay lên trên thì

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

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

Nếu một TGĐV có đỉnh quay xuống dưới thì

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

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

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

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

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

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

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

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

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

```
    t := v1; v1 := v2; v2 := t;
    t := d1; d1 := d2; d2 := t;
  end;
  { v1 <= v2 }
  if odd(v1) then KhoangCach := KCLe(d1,v1,d2,v2)
  else KhoangCach := KCLe(d1-1,v1-1,d2,v2) - 1;
end;
procedure XuLi;
 var d,v,j: longint;
     Ad, Av, Bd, Bv, Cd, Cv: longint;
begin
   fillchar(kq,sizeof(kq),0);
   ToaDo(NA, Ad, Av);
   ToaDo(NB, Bd, Bv);
   ToaDo(NC, Cd, Cv);
   for d := 1 to N do
     for v := 1 to 2*d - 1 do
        inc(kq[Min3(KhoangCach(Ad,Av,d,v),
                    KhoangCach(Bd,Bv,d,v),
                    KhoangCach(Cd,Cv,d,v))]);
end;
```

Chương trình C#

Chương trình C# dưới đây giải bài toán với dữ liệu cho trước N = 4, A, B và C lần lượt xuất phát tại các TGĐV 2, 14 và 9 như thí dụ đã cho.

```
// C#
using System;
using System.Collections.Generic;
using System.Text;
namespace SangTao2 {
    class TamGiacDeu {
        static int n = 4, NA = 2, NB = 14, NC = 9;
        static int[] Kq = new int[3];
        static void Main(string[] args){
            XuLi();
            for (int i = 0; i < 3; ++i)
                Console.Write(KQ[i] + " ");
            Console.ReadLine();
        }
     // Tinh dong va vi tri tren dong
        // theo so hieu cua TGDV
        static void ToaDo(int cell, out int dong, out int viTri){
            dong = 0;
            while (cell > 0){
                ++dong; cell -= (2*dong - 1);
            }
            viTri = cell + (2*dong - 1);
        }
        static int KhoangCach(int d1, int v1, int d2, int v2){
           if (d1 > d2){
                int t;
                t = d1; d1 = d2; d2 = t;
                t = v1; v1 = v2; v2 = t;
            }
        return (v1%2==1)?KCLe(d1,v1,d2,v2):KCLe(d1-1,v1-1,d2,v2)-1;
```

```
        }
        static int KCLe(int d1, int v1, int d2, int v2){
            int c1=v1, d=2*(d2-d1), c2=v1+d;
            // Xet tam giac voi 3 dinh v1 c1 c2
            if (c1 <= v2 && v2 <= c2)
                return (v2 % 2 == 1) ? d : d-1;
            return d + Math.Min(Math.Abs(v2-c1),Math.Abs(v2-c2));
        }
        static int Min3(int a, int b, int c){
            int min = 0;
            if (a > b) { min = 1; a = b;}
            if (a > c) min = 2;
            return min;
        }
        static void XuLi(){
            int Ad, Av, Bd, Bv, Cd, Cv;
            ToaDo(NA,out Ad, out Av);
            ToaDo(NB,out Bd, out Bv);
            ToaDo(NC,out Cd, out Cv);
            Array.Clear(Kq, 0, Kq.Length);
            for (int d = 1; d <= n; ++d){
                int vv = 2*d-1;
                for (int v = 1; v <= vv; ++v)
                    ++KQ[Min3(KhoangCach(Ad,Av,d,v),
                              KhoangCach(Bd,Bv,d,v),
                              KhoangCach(Cd,Cv,d,v))];
            }
        }
    } // Tam Giac Deu
} // SangTao2
```

Độ phức tạp

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

4.3 Dạng biểu diễn của giai thừa

Cho số tự nhiên $n \le 480.000$. Hãy phân tích n! ra tích của các thừa số nguyên tố theo trật tự tăng dần. Thí dụ, $13! = 2^{10}.3^5.5^2.7.11.13$. Kết quả hiển thị dưới dạng các dòng, mỗi dòng một số nguyên tố tiếp đến là số mũ tương ứng. Các số trên cùng dòng cách nhau qua dấu cách. Thí dụ trên cho ta kết quả hiển thị như sau

```
2  10
3  5
5  2
7  1
11 1
13 1
```

Thuật toán

Nhận xét Cho số tự nhiên N và một số nguyên tố p. Khi đó,

Nếu viết dãy thừa số 1, 2, ..., N vào một bảng có p cột thì ta thấy có n_1 = N div p dòng chứa p, 2p,...,n_1.p (ở cột cuối cùng). Nhóm các phần tử này lại ta được,

1p.2p....n_1p = (1.2...n_1).p^{n1}. Thực hiện tương tự với tích 1.2...n_1 ta thu được n_2 = n_1 div p dòng chứa p, 2p,...,n_2.p... Từ đây ta suy ra lũy thừa k của p, p^k trong dạng phân tích của N! sẽ là k = $n_1+n_2+...+n_v$, trong

đó $n_i = n_{i-1}$ div p, n_1 = N div p, n_v = 0, i = 2..v. Hàm tính lũy thừa của p trong dạng phân tích của N! bằng các phép chia liên tiếp khi đó sẽ như sau,

```
function Power(n,p: longint): byte;
var k: byte;
begin
  k := 0;
  while (n <> 0) do
   begin
    n := n div p;
    k := k + n;
   end;
  Power := k;
 end;
```

Ta dùng hàm **NextPrime** để sinh lần lượt các số nguyên tố p trong khoảng 2..N và tính **Power(N,p)**. Nếu giá trị này lớn hơn 0 thì ta hiển thị kết quả.

```
procedure Fac(n: longint);
const bl = #32; { Dau cach }
var p: longint; k: byte;
begin
  writeln;
  p := 2;
  while p <= n do
  begin
    k := Power(n,p);
    if (k > 0) then writeln(p,bl,k);
    p := NextPrime(p);
  end;
end;
```

Hai hàm phụ trợ.

Hàm **IsPrime(p)** kiểm tra p có phải là số nguyên tố hay không bằng cách xét xem trong khoảng từ 2 đến $\sqrt{p}$ có ước nào không.

```
function IsPrime(p: longint): Boolean;
var i: longint;
begin
  IsPrime := false;
  if p < 2 then exit;
  for i := 2 to round(sqrt(p)) do
     if p mod i = 0 then exit;
  IsPrime := True;
end;
```

Hàm **NextPrime(p)** sinh số nguyên tố sát sau p bằng cách duyệt tuần tự các số lẻ sau p là p+2k nếu p lẻ và (p-1) + 2k, nếu p chẵn.

```
function NextPrime(p: longint): longint;
begin
  if p < 2 then
   begin
     NextPrime := 2;
     exit;
   end;
  if not odd(p) then p := p-1;
  repeat
    p := p+2;
  until IsPrime(p);
```

```
  NextPrime := p;
end;
```

Ta có thể cải tiến khá mạnh tốc độ tính toán bằng các kỹ thuật sau.

Sinh sẵn các số nguyên tố trong khoảng từ 1..N bằng giải thuật Sàng mang tên nhà toán học Hi Lạp Eratosthene. Từ vài nghìn năm trước, Eratosthenes đã dạy như sau:

Bài giảng của Eratosthenes

Nếu trò muốn liệt kê toàn bộ các số nguyên tố nằm trong khoảng từ 1 đến N hãy làm như sau

1. Viết dãy số từ 1 đến N.
2. Xóa đi số 1 vì nó không phải là số nguyên tố, cũng không phải là hợp số. Nó là một số đặc biệt.
3. Lần lượt duyệt từ 2 đến $\sqrt{N}$ như sau. Nếu gặp số chưa bị xóa thì đó chính là một số nguyên tố. Trò hãy xóa mọi bội của số này kể từ bình phương của nó trở đi.

Khi kết thúc, những số nào không bị xóa trên tấm bảng sẽ là các số nguyên tố. Đó là kho các số nguyên tố trong khoảng 1..N.

Eratosthenes (276-194 tr. CN) Nhà toán học lỗi lạc Hy Lạp Cổ đại. Ông sinh tại Cyrene, theo học trường phái Plato tại Athens. Hoàng đế Ptolemy II mời ông đến Alexandria để dạy cho hoàng tử.

Sau ông được giao phụ trách thư viên Alexandria, một trung tâm lưu trữ và bảo tồn các tác phẩm văn hóa và khoa học nổi tiếng đương thời. Ngoài các công trình tóan học, Eratosthenes còn có những đóng góp rất giá trị về đo lường. Ông đã tiến hành đo kích thước Trái Đất.

Thời đó chưa có giấy viết nên thày trò phải viết trên những tấm bảng bằng đất sét vào lúc đất còn dẻo, các số bị xóa được đục thủng. Sau khi phơi khô ta thu được những tấm bảng thủng lỗ chỗ như một cái *sàng gạo*.

Với mảng **a[0..MN] of byte** đủ lớn, thí dụ, MN = 60.000 ta có thể ghi nhận các số nguyên tố nằm trong khoảng 1..MN. Ta qui ước a[i] = 0 thì i là số nguyên tố, a[i] = 1 ứng với số i bị dùi thủng nên i không phải là số nguyên tố.

```
procedure Eratosthenes(n: longint);
var i,j: longint;
begin
  fillchar(a,sizeof(a),0);
  for i := 2 to round(sqrt(n)) do
    if a[i]=0 then
       for j := i to (n div i) do a[i*j] := 1;
end;
```

Thủ tục phân tích N! ra thừa số nguyên tố dạng cải tiến sẽ như sau,

```
procedure NewFac(n: longint);
const bl = #32; { Dau cach }
 var i,p: longint;
begin
  Eratosthenes(n);
  writeln;
  for i := 2 to n do
    if a[i] = 0 then
       begin
         p := Power(n,i);
         if P > 0 then writeln(i,bl,p);
       end;
```

```
end;
```

Dùng kỹ thuật đánh dấu bit có thể tạo kho số nguyên tố cỡ 8.MN vì một byte có 8 bit, mỗi bit sẽ quản lí 1 số.

Mảng a vẫn được khai báo như trước: **a[0..MN] of byte** (quan trọng là chỉ số phải tính từ 0 trở đi) nhưng lúc này mỗi phần tử a[i] sẽ quản lí 8 số chứ không phải một số như trước. Tiếp đến bạn cần viết thêm ba thủ tục sau đây:

Thủ tục **BitOn(i)** - đặt trị 1 cho bit thứ i trong dãy bit a (bật bit). Các bit trong dãy a sẽ được mã số từ 0 đến 8MN-1= 480.000-1. Bản thân số 480.000 là hợp số nên ta có thể bỏ qua.

```
procedure BitOn(i: longint);
 var b,p: longint;
begin
  b := i shr 3; { i div 8 }
  p := i and 7; { i mod 8 }
  a[b] := a[b] or (1 shl p);
end;
```

Đặt trị 1 cho bit i trong dãy bit a

1. Xác định xem bit i nằm trong byte nào

```
b := i div 8
```

2. Xác định xem bit i là bit thứ mấy trong byte b (tính theo trật tự 7,6,5,4,3,2,1,0)

```
p := i mod 8
```

3. Lấy số nhị phân 8 bit 00000001 dịch trái p vị trí rồi cộng logic theo bit với a[b].

```
a[b] := a[b] or (1 shl p);
```

Bạn ghi nhớ sự tương đương của các phép toán sau đây

Phép toán	**Phép toán tương đương**
x div 2^k	**x shr k**
x mod 2^k	**x and 2^k-1**
	Tính theo dạng này sẽ nhanh hơn

Thủ tục **BitOff(i)** đặt trị 0 cho bit thứ i trong dãy bit a (tắt bit).

```
procedure BitOff(i: longint);
 var b,p: longint;
begin
  b := i shr 3; { i div 8 }
  p := i and 7; { i mod 8 }
  a[b]:=a[b] and (not(1 shl p));
end;
```

Đặt trị 0 cho bit i trong dãy bit a

1. Xác định xem bit i nằm trong byte nào

```
b := i div 8;
```

2. Xác định xem bit i là bit thứ mấy trong byte b (tính theo trật tự 7,6,5,4,3,2,1,0)

```
p := i mod 8;
```

3. Lấy số nhị phân 6 bit 00000001 dịch trái p vị trí, lật rồi nhân logic theo bit với a[b].

```
a[b]:=a[b] and (not(1 shl p));
```

Hàm **GetBit(i)** cho ra trị (1/0) của bit i trong dãy bit a.

```
function GetBit(i: longint): byte;
 var b,p: longint;
```

Đặt trị 0 cho bit i trong dãy bit a

1. Xác định xem bit i nằm trong byte nào

begin **b := i shr 3;** **p := i and 7; { i mod 8 }** **GetBit := (a[b] shr p) and 1;** **end;**	**b := i div 8;** *2. Xác định xem bit i là bit thứ mấy trong byte b (tính theo trật tự 7,6,5,4,3,2,1,0)* **p := i mod 8;** *3. Dịch a[b] qua phải p vị trí, rồi nhân logic theo bit với 00000001 để lấy bit phải nhất (bit 0).* **GetBit := (a[b] shr p) and 1;**

Các thủ tục cơ bản theo kỹ thuật xử lí bit khi đó sẽ như sau.

```
procedure Eratosthenes_B(n: longint);
var i,j: longint;
begin
  fillchar(a,sizeof(a),0);
  for i:=2 to round(sqrt(n)) do
    for j:=i to (n div i) do
       BitOn(i*j);
end;
procedure BFac(n: longint);
const bl = #32; { Dau cach }
 var i,p: longint;
begin
  Eratosthenes_B(n);
  writeln;
  for i:=2 to n do
    if GetBit(i)=0 then
       begin
         p := Power(n,i);
         if P > 0 then writeln(i,bl,p);
       end;
end;
```

Chương trình C#

```
//  C#
using System;
using System.Collections.Generic;
using System.Text;
namespace SangTao2 {
    class GiaiThua {
        static byte [] a = new byte[40000];
        static void Main(string[] args){
            BFac(13);
            Console.ReadLine();
        }
        static int Power(int n, int p){
            int k = 0;
            while (n != 0){ n /= p; k += n; }
           return k;
        }
        static void Fac(int n) {
            Console.WriteLine();
            int p = 2, k;
            while (p <= n){
                k = Power(n,p);
```

```
        if (k > 0) Console.WriteLine(p+" "+k);
        p = NextPrime(p);
    }
}
static bool IsPrime(int p){
    if (p<2) return false;
    if (p==2) return true;
    if (p % 2 == 0) return false;
    int cp = (int)(Math.Sqrt(p));
    for (int i=3; i <= cp; i+=2)
        if (p % i == 0) return false;
    return true;
}
static int NextPrime(int p){
    if (p < 2) return 2;
    if (p % 2 == 0) --p;
    do { p += 2; } while (!IsPrime(p));
    return p;
}
// Sang Eratosthene dung byte
static void Eratosthenes(int n){
    Array.Clear(a,0,a.Length);
    int sn = (int)Math.Sqrt(n);
    for (int i = 2; i <= sn; ++i)
        if (a[i]==0){
            int ni = n/i;
            for (int j = i; j <= ni; ++j) a[i*j] = 1;
        }
}
// Gan 1 cho bit i
static void BitOn(int i){
    int b = i >> 3;
    int p = i & 7;
    a[b] |= (byte)(1 << p);
}
// Gan 0 cho bit i
static void BitOff(int i){
    int b = i >> 3;
    int p = i & 7;
    a[b] &= (byte)~(1 << p);
}
// Lay tri cua bit i
static byte GetBit(int i) {
    int b = i >> 3;
    int p = (i & 7);
    return (byte)((a[b] >> p)&1);
}
// Sang Eratosthene dung bit
static void Eratosthenes_B(int n){
    Array.Clear(a, 0, a.Length);
    int sn = (int)Math.Sqrt(n);
    for (int i = 2; i <= sn; ++i)
        if (GetBit(i) == 0) {
            int ni = n / i;
            for (int j = i; j <= ni; ++j) BitOn(i * j);
        }
}
```

```
        static void BFac(int n){
            int p;
            Eratosthenes_B(n);
            for (int i = 2; i <= n; ++i)
                if (GetBit(i)==0)
                {
                    p = Power(n,i);
                    if (p > 0) Console.WriteLine(i+" "+p);
                }
        }
    } // GiaiThua
} // SangTao2
```

Độ phức tạp

Để liệt kê các số nguyên tố từ 1..N ta duyệt từ 1 đến $\sqrt{N}$, với mỗi số nguyên tố ta phải gạch tối đa cỡ N các bội của chúng. Vậy độ phức tạp tính toán cỡ *N*. $\sqrt{N}$.

4.4 Xếp sỏi

Cho một bảng chia lưới ô vuông N dòng mã số 1..N tính từ trên xuống và M cột mã số 1..M tính từ trái sang. Mỗi ô được phép đặt không quá 1 viên sỏi. Người ta cho trước giới hạn tổng số sỏi được phép đặt trên dòng i là d_i, i = 1..N và trên mỗi cột j là C_j, j = 1..M. Hãy tìm một phương án xếp được nhiều sỏi nhất trong bảng, biết rằng các dữ liệu đều hợp lệ và bài toán luôn có nghiệm.

Thuật toán

Tổ chức dữ liệu:

```
const MN = 101;
d: array[0..MN] of integer;
c: array[0..MN] of integer;
a: array[1..MN,1..MN] of byte;
```

trong đó d là mảng chứa giới hạn sỏi trên dòng, c - trên cột, a là mảng hai chiều biểu diễn bảng chia lưới ô vuông, a[i,j] = 1 - có viên sỏi đặt tại dòng i, cột j; a[i,j] = 0 - không có sỏi tại ô này. Ta thực hiện kỹ thuật hai pha như sau.

```
procedure XepSoi;
var j: integer;
begin
   fillchar(a,sizeof(a),0);
   d[0] := M+1; { dat linh canh }
   { Pha 1 } XepDong;
   { Pha 2 } for j := 1 to M do ChinhCot(j);
end;
```

Pha thứ nhất: Xếp tối đa sỏi vào mỗi dòng. Mỗi dòng i ta xếp liền nhau d[i] viên sỏi. Đồng thời ta sử dụng lại các biến mảng d và c với ý nghĩa sau đây: d[i] cho biết vị trí cột của viên sỏi cuối cùng trên dòng i. c[j] cho biết số sỏi còn có thể xếp thêm trên cột j. Dĩ nhiên, ta phải chỉnh lại các giá trị c[j] mỗi khi xếp thêm 1 viên sỏi vào cột này. Nếu c[j] < 0 tức là ta cần bớt sỏi ở cột j. Thủ tục xếp dòng khi đó sẽ như sau.

```
procedure XepDong;
  var i,j: integer;
begin
  for i := 1 to N do
     for j := 1 to d[i] do
       begin
         a[i,j] := 1; dec(c[j]);
       end;
end;
```

Pha thứ hai: Sau khi xếp xong N dòng ta tiến hành chỉnh từng cột j có giá trị c[j] < 0 đến khi nào c[j] = 0. Để chỉnh cột j theo phương pháp tham lam ta duyệt để chọn một dòng imin có chứa sỏi tại cột j và đầu phải d[imin] đạt giá trị nhỏ nhất. Sau đó ta chuyển viên sỏi trên dòng imin từ cột j sang cột d[imin]+1 và chỉnh lại các giá trị c[j] và d[imin]. Để tìm dòng imin ta cần dùng phần tử d[0] với giá trị lớn nhất làm phần tử khởi đầu. Ta có thể cho giá trị này là M+1, vì mỗi dòng không thể có qúa M viên sỏi. Bạn cần lưu ý rằng khi d[imin] = M tức là mọi viên sỏi cuối cùng trên mỗi dòng đều chiếm vị trí tại cột M tức là hết chỗ để đặt sỏi.

```
procedure ChinhCot(j: integer);
begin
  while c[j] < 0 do GiamCot(j);
end;
procedure GiamCot(j: integer);
  var i: integer;
begin
  i := DongMin(j);
  a[i,j] := 0; { Bo vien soi } inc(c[j]);
  if d[i] = M then exit;
  inc(d[i]); a[i,d[i]] := 1; { Dat 1 vien vao day }
  dec(c[d[i]]);
end;
function DongMin(j: integer): integer;
  var i,imin: integer;
begin
  imin := 0;
  for i:=1 to N do
    if a[i,j]=1 then
      if d[i] < d[imin] then imin := i;
   DongMin := imin;
end;
```

Thí dụ dưới đây minh họa thuật toán với N = M = 4; d = (3,2,1,2), c = (2,2,2,2).

0	0	0	0	3
0	0	0	0	2
0	0	0	0	1
0	0	0	0	2
2	2	2	2	

Cấu hình ban đầu

1	1	1	0	3
1	1	0	0	2
1	0	0	0	1
1	1	0	0	2
-2	-1	1	2	

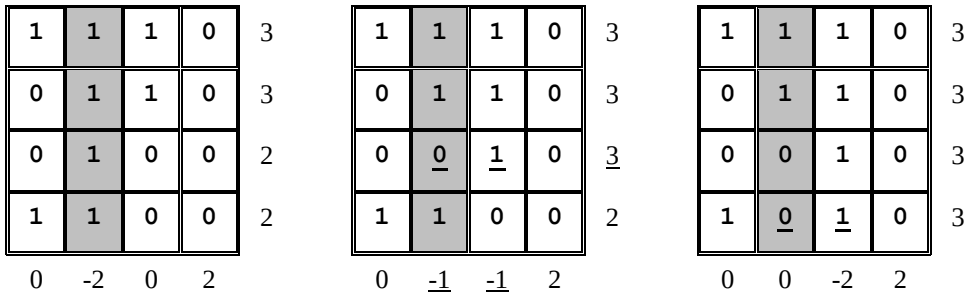
1	1	1	0	3
1	1	0	0	2
1	0	0	0	1
1	1	0	0	2
-2	-1	1	2	

Sau Pha 1

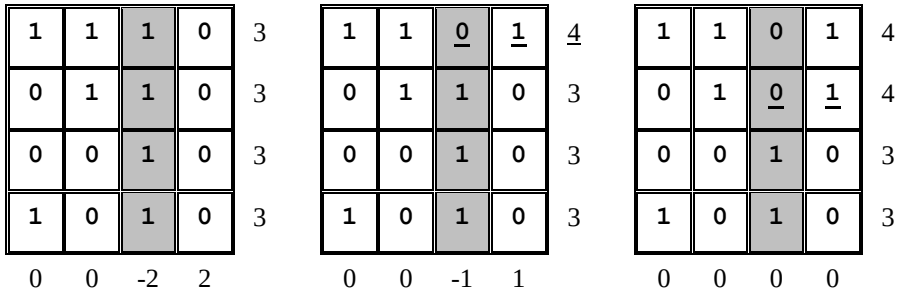
1	1	1	0	3
1	1	0	0	2
0	1	0	0	2
1	1	0	0	2
-1	-2	1	2	

1	1	1	0	3
0	1	1	0	3
0	1	0	0	2
1	1	0	0	2
0	-2	0	2	

Chỉnh cột 1



Chỉnh cột 2



Chỉnh cột 3

Độ phức tạp

Ta cần chỉnh M cột. Mỗi cột ta cần lặp tối đa N lần, mỗi lần giảm được 1 viên sỏi trong cột. Để giảm 1 viên sỏi này ta phải duyệt N dòng để tìm imin. Tổng cộng ta cần cỡ MN^2 thao tác.

Chương trình C#

```
// C#
  using System;
  using System.Collections.Generic;
  using System.Text;
  namespace SangTao2 {
      class XepSoi {
          const int n = 4, m = 4;
          static byte [,] a = new byte[n+1,m+1];
          static int [] d = new int [n+1] {0,3,2,1,2};
          static int [] c = new int [m+1] {0,2,1,2,3};
          static void Main(string[] args) {
              Xep(); Show();
              Console.ReadLine();
          }
          static void Show(){
              for (int i = 1; i <= n; ++i){
                  for (int j = 1; j <= m; ++j)
                      Console.Write(a[i, j]);
                  Console.WriteLine();
              }
          }
          static void Xep(){
              Array.Clear(a,0,a.Length);
              d[0] = m+1;
```

```
            XepDong();
            for (int j = 1; j <= m; ++j) ChinhCot(j);
        }
        static void XepDong(){
            for (int i = 1; i <= n; ++i)
                for (int j = 1; j <= d[i];++j){
                    a[i,j] = 1; --c[j];
                }
        }
        static void ChinhCot(int j) {
            while (c[j] < 0) GiamCot(j);
        }
        static void GiamCot(int j){
            int i = DongMin(j);
            a[i,j] = 0; // Bot 1 vien tai o (i,j)
            ++c[j];
            if (d[i]==m) return; // het cho dat tren dong i
            ++d[i]; a[i,d[i]] = 1; // Dat 1 vien vao o (i,d[i])
            --c[d[i]];
        }
        static int DongMin(int j){
            int imin = 0;
            for (int i = 1; i <= n; ++i)
                if (a[i,j]==1)
                    if (d[i] < d[imin]) imin = i;
            return imin;
        }
    } // XepSoi
} // SangTao2
```

4.5 Dãy các hoán vị

Dãy các hoán vị của N chữ cái HOA đầu tiên trong bảng chữ tiếng Anh được sắp theo trật tự từ điển tăng dần và viết liền nhau thành một dãy kí tự duy nhất. Hãy cho biết kí tự thứ M trong dãy tính từ 1 trở đi, $2 \le N \le 10$, $1 \le M \le N.N!$. *Thí dụ, với N=3, ta có dãy 6 hoán vị xếp theo trật tự từ điển là ABC, ACB, BAC, BCA, CAB, CBA. Sau khi ghép chúng ta thu được dãy duy nhất gồm 18 kí tự ABCACBBACBCACABCBA.* Kí tự thứ M = 15 trong dãy là: *B.*

Thuật toán

Nếu ta viết mỗi hoán vị trên 1 dòng thì kí tự thứ M sẽ nằm trên dòng d = (M-1) div N (tính từ dòng 0) và sẽ chiếm vị trí v = ((M-1) mod N)+1 (tính từ 1) trên dòng d đó. Như vậy ta cần xác định hoán vị trên dòng d rồi lấy kí tự nằm ở vị trí v làm kết quả.

Để xác định hoán vị $(c_1,c_2,...,c_N)$ tại dòng d ta lần lượt tính các kí tự c_i, i = 1..N. Ta phân hoạch các hoán vị theo nhóm. Nếu bỏ kí tự đầu tiên thì ta còn lại (N-1)! hoán vị, khi đó hoán vị tại dòng d sẽ rơi vào nhóm d div (N-1)! và sẽ chiếm dòng d mod (N-1)! trong nhóm đó. Tương tự, ta tính cho các kí tự thứ 2, 3, ..., N-1. Kí tự còn lại sẽ chiếm vị trí thứ N. Nếu biết nhóm d của kí tự thứ i trong hoán vị thì ta tính được chính kí tự đó như sau.

d = 1 ứng với kí tự thứ nhất trong số các kí tự chưa dùng,

d = 2 ứng với kí tự thứ hai trong số các kí tự chưa dùng,

...

Tổng quát, d ứng với kí tự thứ d trong số các kí tự chưa dùng.

Mỗi lần xác định được kí tự nào thì ta đánh dấu kí tự đó bằng thủ tục Mark.

Để tránh việc tính n! ta viết thủ tục ThuongDu(z, n, q, r) cho ra thương q và dư r của phép chia số tự nhiên z cho n!, cụ thể là q = z div n! và r = z mod n!. Thủ tục này khá đơn giản. Ta có

$q_1 = z \text{ div } n$; $r_1 = z \text{ mod } n \Rightarrow z = q_1.n + r_1$;

$q_2 = q_1 \text{ div } (n-1)$; $r_2 = q_1 \text{ mod } (n-1) \Rightarrow q_1 = q_2.(n-1) + r_2$;

…

$q_{n-1} = q_{n-2} \text{ div } 2$; $r_{n-1} = q_{n-2} \text{ mod } 2 \Rightarrow q_{n-2} = q_{n-1}.2 + r_{n-1}$.

$q_n = q_{n-1} \text{ div } 1 = q_{n-1}$; $r_n = q_{n-1} \text{ mod } 1 = 0$.

Thay lần lượt các đại lượng của dòng dưới vào dòng trên ta thu được $q = q_{n-1}$ và $r = r_1 + n.r_2 + (n-1).r_3 + \ldots + 3.r_{n-1} + 2.r_n$. Nhận xét này cho phép ta xây dựng thủ tục theo kỹ thuật chia liên tiếp như sau.

```
procedure ThuongDu(z,n: longint;var q,r: longint);
   var c: longint;
  begin
    r := 0; q := z; c := 1;
    while n > 1 do
      begin
        r := r + (q mod n)*c;
        q := q div n;
        c := n; n := n - 1;
      end;
  end;
```

Thủ tục Test trong chương trình dưới đây tính mọi xuất hiện của các kí tự (M = 1..24*4) trong dãy các hoán vị với N = 4.

Chương trình Pascal

```
(* Pascal *)
    uses crt;
    const MN = 20; bl = #32;
    var   b: array[0..MN] of byte;
    { d = z div n! r = z mod n! }
    procedure ThuongDu(z,n: longint;var q,r: longint);
     Tự viết
    { Danh dau ki tu v thu k
       trong so cac ki tu chua dung }
    procedure Mark(N,k,v: integer);
     var i,d: integer;
     begin
       d := 0;
       for i := 1 to N do
         if b[i] = 0 then
          begin
            d := d+1;
            if d = k then
             begin
               b[i] := v;
               exit;
             end;
          end;
     end;
    { Xac dinh ki tu thu M trong day cac hoan vi }
    function Value(N: integer;M: longint): char;
     var i,j,v: integer;
         th,du,d: longint;
    begin
       fillchar(b,sizeof(b),0);
       d := (M-1) div N; { Dong chua ki tu M }
```

```
      v := (M-1) mod N + 1; { vi tri cua M tren dong d }
      { xac dinh hoan vi tai dong d }
      j := N-1;
      for i := 1 to N-1 do
       begin
         ThuongDu(d,j,th,du);
         Mark(N, th+1,i);
         j := j-1;
         d := du;
       end;
      Mark(N,1,N);
      for i:=1 to N do
        if b[i] = v then
        begin
          Value := chr(ord('A') + i-1);
          exit;
        end;
  end;
  procedure Test;
    var N: integer;
        M: longint;
  begin
    N := 4; writeln;
    for M := 1 to 24*N do
      begin
       write(Value(N,M));
       if M mod N = 0 then
        begin
          if readkey = #27 then halt else writeln;
        end;
      end;
  end;

  BEGIN
    Test;
  END.
```

Chương trình C#

```
// C#
using System;
using System.Collections.Generic;
using System.Text;

namespace SangTao2 {
    class DayHoanVi {
        const int MN = 20;
        static int [] b = new int [MN+1];
        static void Main(string[] args){
            Test();
        }
        // q = z / n!; r = z % n!
        static void ThuongDu(long z, int n,
                             out long q, out long r ){
            q = z; r = 0;
            int c = 1;
```

```
            while (n > 1){
                r += (q % n) * c;
                q /= n;
                c = n; --n;
            }
        }
        static void Mark(int n, long k, int v){
            int d = 0;
            for (int i = 1; i <= n; ++i)
                if (b[i]==0){
                    ++d;
                    if (d==k){ b[i] = v; return; }
                }
        }
        static char Value(int n, long m){
            Array.Clear(b, 0, b.Length);
            long d = (int) (m - 1) / n;
          // d - Dong chua ki tu can tim
            int v = (int)(m - 1) % n + 1;
            // v - vi tri cua ki tu tren dong d
            int j = n - 1;
            long th, du;
            for (int i = 1; i < n; ++i){
                ThuongDu(d, j, out th, out du);
                Mark(n, th + 1, i);
                d = du; --j;
            }
            Mark(n, 1, n);
            for (int i = 1; i <= n; ++i)
                if (b[i]==v) return (char)('A'+i-1);
            return (char)0;
        }
        // test voi n=4, m=1..n.n!
        static void Test(){
            int n = 4;
            int m4 = 24 * n;
            string s;
            for (long m = 1; m <= m4; ++m) {
                Console.Write(Value(n, m));
                if (m % n == 0){
                    s = Console.ReadLine();
                    if (s == "stop") break;
                }
            }
        }
    } // DayHoanVi
} // SangTao2
```

N	N!	N	N!
1	1	11	39916800
2	2	12	479001600
3	6	13	6227020800
4	24	14	87178291200

5	120	13	1307674368000
6	720	14	20922789888000
7	5040	15	355687428096000
8	40320	16	6402373705728000
9	362880	17	121645100408832000
10	3628800	18	2432902008176640000
		19	
		20	

Giai thừa của 20 số nguyên dương đầu tiên

Với C# bạn có thể dùng kiểu int64 hoặc long với 64 bit (8 byte) biểu diễn số nguyên trong khoảng [-9.223.372.036.854.775.808, 9.223.372.036.854.775.807].

Độ phức tạp

Thuật toán chỉ đòi hỏi N = 20 phép chia các số nguyên có tối đa 20 chữ số và gọi thủ tục Mark N lần, mỗi lần gọi phải thực hiện phép duyệt trên dãy N phần tử. Tổng cộng là N^2 phép toán, tức là cỡ 400 phép toán thay vì 2432902008176640000 phép toán nếu ta sinh lần lượt N! hoán vị bằng phương pháp vét cạn với N = 20.

4.6 Bộ bài

Trên bàn đặt một bộ bài gồm n-1 quân bài mã số 1,2,…,n-1, $3 \le n \le 10000$. Trọng tài chỉ định bạn lấy k quân bài. Sau đó trọng tài đưa ra một số tự nhiên s. Bạn cần cố gắng thực hiện ít nhất m thao tác thuộc một trong hai loại sau đây:

- Lấy thêm một quân bài từ trên bàn,

- Bỏ bớt một quân bài trên tay,

để cuối cùng đạt được hệ thức

$$t \bmod n = s \bmod n \qquad (*)$$

trong đó t là tổng số hiệu các quân bài có trên tay bạn sau khi bạn đã hoàn tất m thao tác như trên.

Dữ liệu vào: file văn bản **BAI.INP**

Dòng đầu tiên: 3 số tự nhiên n, k và s.

Từ dòng thứ hai trở đi: k số tự nhiên thể hiện mã số của các quân bài cần lấy lúc đầu.

Dữ liệu ra: Hiển thị trên màn hình

Dòng đầu tiên: số tự nhiên m cho biết số thao tác ít nhất cần thực hiện

Tiếp đến là m dòng, mỗi dòng là một thao tác lấy thêm hoặc bỏ bớt một quân bài v. v > 0 cho biết cần lấy thêm (từ trên bàn) quân bài v; v < 0 cho biết cần bớt (từ trên tay) quân bài v để đạt được hệ thức (*).

Thí dụ, với n = 8, trọng tài cho số s = 22 và chỉ định bạn lấy k = 3 quân bài là 2, 3 và 6.

Nếu bạn bỏ quân bài 2 và lấy quân bài 5 thì tổng t = 3 + 6 + 5 = 14. Khi đó
t mod n = 14 mod 8 = 6 = s mod n = 22 mod 8.
Vậy một lời giải cho bộ dữ liệu này là
Thực hiện 2 thao tác: –2 và +5

BAI.INP	MÀN HÌNH
8 3 22 2 3 6	2 -2 5

Ý NGHĨA

Cho bộ bài gồm 8 quân. Lúc đầu trọng tài chỉ định bạn lấy k = 3 quân bài mã số 2, 3 và 6. Ngoài ra trọng tài đưa ra số s = 22.

Sau đó bạn thực hiện 2 thao tác

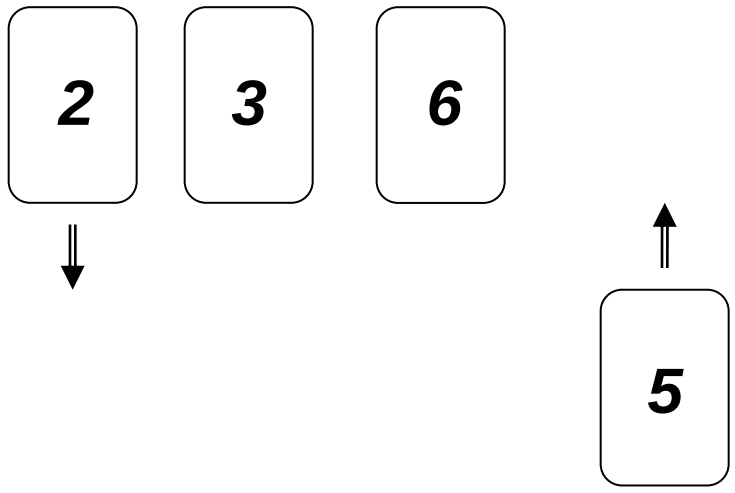
- bỏ quân bài 2

- lấy thêm quân bài 5.

Khi đó tổng số hiệu các quân bài có trên tay bạn sẽ là:

T = 3 + 6 + 5 = 14

T mod N = 14 mod 8 = 6 = s mod 8 = 22 mod 8.



n = 8; s = 22; Trên tay giữ k = 3 quân bài 2, 3, 6.

Lời giải: Bỏ quân bài 2, lấy thêm quân bài 5.

t = 3+6+5 = 14,

t mod 8 = 14 mod 8 = 6 = s mod 8 = 22 mod 8.

Thuật toán

Ta sẽ chứng minh rằng với không quá 2 thao tác (+) lấy thêm / (–) bỏ bớt một quân bài ta có thể đạt được hệ thức (*).

Trước hết ta nhắc lại các phép toán đồng dư. Với số nguyên dương n cho trước ta xét tập các số dư trong phép chia một số tự nhiên x cho n, x mod n, $Z_n = \{0,1,2,\ldots,n-1\}$. Trên Z_n các phép toán cộng và nhân được thực hiện như bình thường sau đó lấy kết quả chia dư cho n. Phép toán lấy số đối của số x cho ta n–x. Phép trừ x–y được đổi thành phép cộng x với số đối của y. Ta có

Cộng: (x + y) mod n

Nhân: x*y mod n

Lấy số đối của x: n – x

Trừ: (x + (n–y)) mod n.

Hãy tưởng tượng các số của Z_n là 0, 1, …, n-1 được bố trí trên một vòng tròn như trên mặt đồng hồ. Để tính tổng x+y ta xuất phát từ x và di chuyển y bước theo chiều kim đồng hồ (còn gọi là *di chuyển xuôi*), mỗi bước ta chuyển qua một số. Kết quả sẽ là điểm dừng cuối cùng. Để tính hiệu x – y ta cũng xuất phát từ x và di chuyển y bước theo chiều ngược lại (*di chuyển ngược*). Để ý rằng, trên vòng tròn gồm n số, di chuyển xuôi y bước sẽ cho cùng kết quả như di chuyển ngược (n–y) bước, và ngược lại, di chuyển ngược y bước sẽ tương đương như di chuyển xuôi (n–y) bước. Điều này có nghĩa là muốn thêm b đơn vị cho đại lượng t ta có thể bớt (n–b) đơn vị và ngược lại, muốn bớt b đơn vị từ đại lượng t ta có thể thêm cho t (n–b) đơn vị. Ta cũng để ý rằng số hiệu của mọi quân bài đều nhỏ thua n và mỗi quân bài hoặc là có trên tay người chơi, hoặc là nằm trên bàn. Vì lẽ trên, đôi khi người ta nói tính toán theo *đồng dư* (modulo) chính là tính toán trên *vòng tròn*.

Bạn cũng cần ghi nhớ tính chất sau đây:

Với mọi số tự nhiên x, y và n, n > 0 và với mọi phép toán số học $\theta \in \{+, -, *\}$ *ta luôn có*

$(x \ \theta \ y) \bmod n = ((x \bmod n) \ \theta \ (y \bmod n)) \bmod n$

Công thức trên cho ta quy tắc dễ hiểu sau đây: Khi tính trị của các biểu thức số học chỉ chứa các phép toán cộng, trừ và nhân trong Z_n ta có thể thực hiện phép lấy số dư *mod n* trên các hạng tử và các kết quả trung gian.

Vì lũy thừa nguyên dương tương đương với phép nhân liên tiếp, ta suy ra hệ quả sau:

$a^k \bmod n = (a \bmod n)^k \bmod n$

Sau khi đã biết các giá trị input là n, k, s và số hiệu các quân bài cần lấy lên tay, ta gán trị cho mảng a[1..n–1] như sau: a[i] = 1 cho biết quân bài i có trên tay, ngược lại, a[i] = 0 cho biết quân bài i còn nằm trên bàn. Với thí dụ đã cho, trọng tài yêu cầu ta lấy 3 quân bài có số hiệu 2, 3 và 6 nên a = (0,1,1,0,0,1,0) ứng với a[2] = a[3] = a[6] = 1, các giá trị a[i] còn lại đều bằng 0.

Trước hết ta tính tổng số hiệu của các quân bài có trong tay lúc đầu và đặt trong biến t. Sau đó ta tính t := t mod n và s := s mod n. Với thí dụ đã cho ta tính được

t = 2+3+6 = 11, do đó t mod n = t mod 8 = 3

và s mod 8 = 22 mod 8 = 6

Tức là t = 3 và s = 6.

Giả sử t ≥ s, ta đặt b = t – s và xét các trường hợp loại trừ nhau sau đây:

1. b = 0: Hệ thức (*) đã thỏa, ta không phải làm gì. Ta thông báo m = 0, trong đó m là số thao tác +/– cần thực hiện.

2. Quân bài b có trên tay, tức là a[b] = 1: Ta chỉ việc bỏ quân bài này xuống, khi đó tổng t sẽ giảm b đơn vị theo mod n.

3. Quân bài (n–b) có trên bàn, tức là a[n–b] = 0: Ta chỉ việc lấy thêm quân bài này. Khi đó tổng t sẽ được thêm (n-b) đơn vị theo mod n, điều này tương đương với việc giảm tổng t đi b đơn vị theo mod n.

4. Nếu không xảy ra các trường hợp 1, 2 và 3 như trên, tức là b ≠ 0, a[b] = 0, a[n–b] = 1, ta tiến hành như sau:

Tìm hai quân bài u và v thỏa các điều kiện sau

Quân bài u có trên tay, a[u] = 1,

Quân bài v có trên bàn, a[v] = 0,

u = (k*b) mod n; v = ((k–1)*b) mod n, k là một số tự nhiên. Điều này có nghĩa là u lớn hơn v b đơn vị theo mod n.

Nếu tìm được hai quân bài u và v như trên ta sẽ thực hiện hai thao tác: bỏ quân bài u (–u) và lấy thêm quân bài v (+v). Khi đó tổng t sẽ được giảm một lượng b theo mod n. Thật vậy,

(u – v) mod n = (k*b – (k–1)*b) mod n = b.

Trường hợp t < s ta phải thêm b = s – t đơn vị cho cho t. Việc này tương đương với giảm t bớt (n-b) đơn vị. Đặt b = n-b rồi lặp lại thủ tục trên sẽ cho ta kết quả tương ứng.

Ta chứng minh rằng nếu gặp tình huống 4 thì bao giờ cũng có thể tìm được hai quân bài u và v như đã mô tả. Trên hai ngàn năm trước nhà toán học Cổ Hy Lạp Diophantus đã phát biểu và chứng minh định lý sau:

Định lý Cho phương trình ax mod n = b mod n, với các hệ số a, b, n là các số tự nhiên, n > 0. Gọi d là ước chung lớn nhất của a và n, d = (a,n). Khi đó

a) Nếu d không là ước của b thì phương trình vô nghiệm.

b) Nếu b = kd thì phương trình có đúng d nghiệm trong tập Z_n. Các nghiệm này có dạng (x + i(n/d)) mod n, trong đó x là một nghiệm tùy ý, i = 0,1,2...(d-1).

Phương trình ax mod n = b mod n được người đời sau gọi là phương trình Diophantus.

Chứng minh

Nếu *x* là nghiệm của phương trình *ax mod n = b mod n* thì *ax* và *b* có cùng số dư theo *mod n* nên hiệu của chúng sẽ chia hết cho *n, ax – b = kn,* hay *ax – kn = b*. Mặt khác, do *d* = (*a,n*) nên *a* và *n* đều chia hết cho *d* và do đó hiệu *ax–kn* cũng chia hết cho *d*, thế tức là *b* phải chia hết cho *d*. Giả sử *b = md* tức là phương trình có nghiệm. Gọi *x* là nghiệm nguyên không âm nhỏ nhất của phương trình trên, ta dễ dàng kiểm tra được rằng *x+i(n/d), i = 0,1,...,*(*d–1*) cũng là nghiệm của phương trình đó. Thật vậy, ta để ý rằng nếu *d* là ước chung lớn nhất của *a* và *n* thì *an/d* chính là bội chung nhỏ nhất của chúng, nghĩa là *an/d* chia hết cho *a* và *n*. Ta có

a(x+i(n/d)) mod n = ((ax mod n) + (i(an)/d) mod n) mod n

= (b mod n + 0) mod n = b mod n.

Ta chứng minh xong.

Thí dụ 1. Giải phương trình sau

6*x mod* 9 = 21 *mod* 9

Phương trình trên tương đương với phương trình sau:

6*x mod* 9 = 3

Ta có *d* = (6,9) = 3. Vì 3 là ước của vế phải nên phương trình đã cho có 3 nghiệm. Dễ thấy *x* = 2 là một nghiệm của phương trình. Vậy các nghiệm của phương trình dưới dạng tổng quát là

x + *i*(*n/d*) = 2 + *i*(9/3) = 2 + 3*i*, *i* = 0, 1, 2

Cụ thể là $x_1 = 2$, $x_2 = 5$ và $x_3 = 8$ là 3 nghiệm trong tập $Z_9 = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$.

Thí dụ 2. Giải phương trình

4*x mod* 12 = 5

Ta có, *d* = (4,12) = 4 không phải là ước của 5. Phương trình vô nghiệm.

Trở lại bài toán trên, khi gặp tình huống 4 ta có a[b] = 0 và a[n–b] = 1. Xét phương trình *bx mod n* = (*n–b*) *mod n*. Vì 1 ≤ *b* < *n* nên 1 ≤ *n–b* < *n* và do đó (*n–b*) *mod n* = *n–b*, phương trình đã cho có thể viết lại là *bx mod n = n–b*.

Theo tính chất: ước chung lớn nhất của hai số tự nhiên (*a,b*) sẽ không đổi nếu ta thay số lớn nhất trong hai số đó bằng hiệu của nó với số thứ hai, đặt *d* = (*b,n*), ta có *d* = (*b,n–b*), tức là *n–b* chia hết cho *d*, do đó phương trình *bx mod n = n–b* luôn có nghiệm. Từ nhận xét này suy ra rằng vòng lặp **repeat** trong đoạn trình dưới đây luôn kết thúc.

```
u := b;
repeat
   v := u;
   u := (u+b) mod n;
until a[u] = 1;
```

Thật vậy, sau *k* lần lặp ta thu được *u = kb* do phương trình *bx mod n = n–b* có nghiệm nên sẽ tồn tại một giá trị *k* để *u = kb mod n = n–b*. Do a[n–b] = 1 nên tối đa sau *k* lần lặp thì vòng lặp phải kết thúc và ta sẽ thu được *u = kb mod n*. Vì *v* mang giá trị sát trước của *u* nên *v = (k–1)b mod n*.

Ta có thuật toán sau đây

1. Đọc dữ liệu vào các biến n, k và s

2. Khởi trị cho mảng a[1..n–1] với a[i] = 1 nếu quân bài i có trên tay, a[i] = 0 nếu quân bài i còn trên bàn.

3. Tính t = tổng số hiệu các quân bài có trên tay.

4. Tính t := t mod n; s := s mod n.

5. Nếu t ≥ s: đặt b := t – s; ngược lại đặt b := n – (s – t).

Ý nghĩa: cần giảm b đơn vị từ tổng t để đạt hệ thức

$$t \bmod n = s \bmod n \qquad (*)$$

6. Xét các trường hợp loại trừ nhau sau đây

6.1 b = 0: Đặt m = 0; Thông báo: "*Không làm gì*"; Stop.

6.2 a[b] = 1 (Quân bài b có trên tay):

Thông báo: "Thực hiện m = 1 thao tác – b: Bỏ quân bài b"; Stop.

6.3 a[b] = 0 và a[n–b] = 0 (Quân bài b không có trên tay, quân bài (n-b) có trên bàn):

Thông báo: "Thực hiện m = 1 thao tác +(n–b): Lấy quân bài (n–b)"; Stop.

6.4 a[b] = 0 và a[n–b] = 1: (Quân bài b không có trên tay, quân bài (n-b) không có trên bàn)

6.4.1 Tính u và v

```
u := b;
repeat
   v := u;
   u := (u+b) mod n;
until a[u] = 1;
```

6.4.2 Thông báo: "Thực hiện m = 2 thao tác

–u: Bỏ quân bài u

+v: Lấy quân bài v."

6.4.3 Stop

Từ chứng minh trên ta rút ra độ phức tạp của thuật toán là O(n) vì trong trường hợp xấu nhất ta duyệt 1 lần mảng a chứa n-1 phần tử.

Tổ chức dữ liệu:

```
const mn = 10000; { Max n }
bl = #32; { Dấu cách }
nl = #13#10; { New line: xuống dòng }
ESC = #27;
fn = 'bai.inp';
type mi1 = array[0..mn] of integer;
var a: mi1; { Dánh dấu các quân bài }
n, k : integer; { n–1: số lượng quân bài }
{ k: số lượng các quân bài trên tay }
t , s: longint; { t: tổng số hiệu các quân bài trên tay }
{ s: số đối chứng của trọng tài }
f: text; { input file }
```

Thủ tục đọc dữ liệu: Mở input file, đọc các giá trị n, k và s, đọc số hiệu và đánh dấu k quân bài được chọn. tính tổng t của chúng }

```
procedure Doc;
var i,j: integer;
begin
  assign(f,fn); reset(f);
  read(f,n,k,s);
  t := 0; fillchar(a,sizeof(a),0);
  for i := 1 to k do
  begin
    read(f,j); a[j] := 1; t := t+j;
  end;
  close(f);
```

```
end;
```

Thủ tục xử lí.

```
procedure XuLi;
var b,u,v: integer;
begin
  t := t mod n; s := s mod n;
  if t >= s then b := t-s else b := n-(s-t);
  if (b = 0) then
    begin
      Ket(0,0,0);
      exit
    end;
  if (a[b] = 1) then
    begin { Quan bai b co tren tay }
      Ket(1,-b,0); { bo xuong }
      exit
    end;
  if (a[n-b] = 0) then
    begin { Quan bai n-b co tren ban }
      Ket(1,n-b,0); { Lay len }
      exit
    end;
  { Quan bai b khong co tren tay
    Quan bai n-b khong co tren ban }
  u := b;
  repeat
    v := u;
    u := (u+b) mod n;
  until (a[u] = 1);
  Ket(2,-u,v); { bo u, lay v }
end;
```

Thủ tục **Ket(m,u,v)** thông báo kết quả ứng với các trường hợp:

m = 1: Bỏ bớt hoặc lấy thêm 1 quân bài u;

m = 2: Bỏ quân bài u, lấy quân bài v.

```
procedure Ket(m,u,v: integer);
begin
  case m of
    0: write(nl,'Khong lam gi',nl);
    1: begin
        write(nl,' Thuc hien 1 thao tac: ');
        if (u > 0) then write('+',u,nl)
        else write(u,nl);
       end;
    2: begin
        write(nl,' Thuc hien 2 thao tac: ');
        if (u > 0) then write('+',u,bl)
        else write(u,bl);
        if (v > 0) then write('+',v,nl)
        else write(v,nl);
       end;
    end;
end;
```

Độ phức tạp tính toán: N.

Chương trình C# dưới đây hiển thị kết quả trên màn hình.

Chương trình C#

```
// C#
  using System;
  using System.Collections.Generic;
  using System.Text;
  using System.IO;
  namespace SangTao2 {
      class BoBai {
          const string fn = "bai.inp";
          const int MN = 20;
          static int[] a;
          static int n; // so luong quan bai
          static int k; // so luong quan bai tren tay
          static int s; // so cho truoc trong khoang [1,n];
          static int t;
          static void Main(string[] args){
              Doc(); XuLi();
              Console.ReadLine();
          }
          static void Doc(){
              int[] c =
           Array.ConvertAll((File.ReadAllText(fn)).Split(
                new char[] { '\0', '\n', '\t', '\r', ' ' },
                   StringSplitOptions.RemoveEmptyEntries),
                  new Converter<String, int>(int.Parse));
              n = c[0]; // so luong quan bai
              k = c[1]; // so luong quan bai tren tay
              s = c[2]; // so cho truoc
              a = new int[n + 1];
              Array.Clear(a, 0, a.Length);
              t = 0;
              for (int i = 3; i < c.Length; ++i){
                  a[c[i]] = 1; t += c[i];
              };
          }
          static void XuLi(){
              t %= n; s %= n;
              int b = (t >= s) ? t - s : n - (s - t);
              if (b == 0) { Ket(0, 0, 0); return; }
              // Sua t: giam b don vi hoac tang n-b don vi
              if (a[b] == 1) { // quan b co tren tay
                  Ket(1, -b, 0); // bo quan b
                  return;
              }
              if (a[n - b] == 0) { // quan n-b co tren ban
                  Ket(1, n - b, 0); // lay quan n-b
                  return;
              }
              // Quan b tren ban, quan n-b trên tay
              int u = b, v = 0;
              do { v = u; u = (u + b) % n; } while(a[u] == 0);
              Ket(2, -u, v); // bo quan u, lay quan v
          }
          static void Ket(int c, int u, int v) {
              switch (c){
```

```
                    case 0: Console.WriteLine(c); break;
                    case 1: Console.WriteLine(c + ": " + u); break;
                    case 2: Console.WriteLine(c + ": "+u+" , "+v);
                            break;
                }
            }
        } // Bo Bai
} // SangTao2
```

4.7 Thuận thế

Dijkstra E.

Cho hoán vị a = (a_1,a_2,...,a_N) của N số nguyên dương đầu tiên 1,2,...,N. Một thuận thế của a là dãy b = (b_1,b_2,...,b_N) trong đó b_i là số lượng các phần tử nhỏ thua a_i và đứng trước a_i, $b_i = ||\{a_j \mid a_j < a_i, j < i\}||$. Biết trước N, $2 \le N \le 1000$.

a) Cho một hoán vị a, tính thuận thế b của a.

b) Cho thuận thế b, tìm hóan vị a.

c) Mọi thuận thế đều có phần tử đầu tiên (trái nhất) là 0 nên ta có thể bỏ phần tử này. Ngoài ra, nếu trong thuận thế còn có phần tử 0 nữa ta bỏ thêm 1 phần tử 0 để thu được một dãy có M = N-1 hoặc M = N-2 phần tử và gọi dãy này là thuận thế thu gọn c. Cho một thuận thế thu gọn. Hãy tìm hoán vị nhỏ nhất theo trật tự từ điển sinh ra thuận thế thu gọn này.

Thí dụ, với N = 5,

a) Cho a = (2,5,1,4,3) ta tính được b = (0,1,0,2,2),

b) Cho b = (0,1,0,2,2) ta tìm được a = (2,5,1,4,3),

c) Cho thuận thế thu gọn c = (1,2,2), N = 5, ta tìm được a = (2,3,5,4,1).

Để ý rằng hai hoán vị (2,5,1,4,3) và (2,3,5,4,1) cùng sinh ra thuận thế thu gọn (1,2,2), nhưng hoán vị (2,3,5,4,1) nhỏ hơn.

Dữ liệu vào: text file **THUANTHE.INP**

Dòng đầu tiên: N

Từ dòng thứ hai trở đi: N phần tử của hoán vị a.

Dòng tiếp theo: M

Trên các dòng tiếp theo: M phần tử của thuận thế thu gọn.

Dữ liệu trên cùng một dòng cách nhau qua dấu cách.

Dữ liệu ra: Hiển thị trên màn hình theo trật tự sau:

Câu a: Cho hoán vị a, tìm thuận thế b.

Câu b: Cho thuận thế b, tìm hoán vị a.

Câu c: Cho thuận thế thu gọn c tìm hoán vị nhỏ nhất a.

Thuật toán

Việc xác định thuận thế b từ hoán vị a là dễ dàng. Hai câu b và c là hơi khó. Chúng ta sẽ sử dụng kỹ thuật đối xứng để trình bày một thuật toán do Dijkstra đề xuất. Theo thuật toán này thì thủ tục cho câu a và b là đối xứng nhau. Thuật toán tiến hành xử lý tại chỗ, nghĩa là không sử dụng mảng phụ mà trực tiếp biến đổi hoán vị a thành thuận thế lưu luôn trong a và ngược lại.

Trước hết ta nhận xét rằng với hoán vị đơn vị e = (1,2,...,N) thì có đúng e_i phần tử không lớn hơn e_i và không đứng sau phần tử e_i, i = 1..N. Vậy, nếu trong một hoán vị a mà ta thấy một phần tử $a_j \ge a_i$ và $j \le i$ thì ta khẳng định rằng chỉ còn đúng a_j-1 phần tử không lớn hơn a_j và không đứng sau phần tử a_j.

Ta khai báo các biến x, y, a là các mảng để chứa các hoán vị và thuận thế:

```
const MN = 1000;
type mi1 = array[0..MN] of integer;
var x,y,a: mi1;
```

Thủ tục biến đổi hoán vị a sang thuận thế a khi đó sẽ như sau:

```
procedure HoanViThuanThe(var a: mi1;n: integer);
var i,j: integer;
begin
  for i := n downto 1 do
    for j:=1 to i do
      if (a[j] >= a[i]) then dec(a[j]);
end;
```

Để thu được hoán vị a từ thuận thế a ta chỉ cần viết thủ tục xử lý theo chiều ngược lại. Hai thủ tục như vậy gọi là *đối xứng nhau*.

```
procedure ThuanTheHoanVi(var a: mi1;n: integer);
var i,j: integer;
begin
  for i := 1 to n do
    for j := i downto 1 do
      if (a[j] >= a[i]) then inc(a[j]);
end;
```

Hai thủ tục này đều có độ phức tạp N^2.

Câu c được giải như sau. trước hết thêm một số 0 vào đầu trái của dữ liệu vào a. Sau đó xét hiệu N-M. Nếu N-M=1 thì chứng tỏ thuận thế thu gọn a chỉ khuyết một số 0. Ta chỉ việc gọi thủ tục **ThuanTheHoanVi(a,N)** là thu được kết quả. Trường hợp N-M=2 thì ta phải bù thêm một số 0 nữa vào một vị trí nào đó trong a. Ta lần lượt đặt số 0 này vào đầu phải (vị trí N) rồi chuyển dần nó về đầu trái, mỗi lần một vị trí và gọi thủ tục **ThuanTheHoanVi** để sinh ra một dãy a[1..N] sau đó kiểm tra xem dãy này có phải là hoán vị của 1..N hay không. Nếu đúng, ta dừng thuật toán và cho ra kết quả. Để kiểm tra một dãy a[1..N] có phải là một hoán vị của 1..N ta sử dụng một mảng d[1..N] đánh dấu xem mỗi phần tử a[i] có xuất hiện đúng 1 lần hay không. Tuy nhiên trước đó ta phải kiểm tra điều kiện $1 \le a[i] \le N$ để đảm bảo rằng a[i] nằm trong giới hạn của chỉ số mảng d.

```
 procedure ThuanTheThuGon(var a: mi1; n,m: integer);
 var b: mi1;
     i: integer;
 begin
  move(a[1],a[2],m*sizeof(integer));
  a[1] := 0; inc(m);
  if (n = m) then
   begin
     ThuanTheHoanVi(a,n);
     exit;
   end;
  b := a;
  for i := n downto 2 do
    begin
      { Them 0 tai vi tri i }
      a := b;
      move(a[i],a[i+1],(n-i)*sizeof(integer));
      a[i] := 0;
      ThuanTheHoanVi(a,n);
      if LaHoanVi(a,n) then exit;
    end;
   end;
function LaHoanVi(var a: mi1; n: integer): Boolean;
 var d: mi1;
     i: integer;
begin
  LaHoanVi := false;
  fillchar(d,sizeof(d),0);
  for i := 1 to n do
```

```
   begin
    if (a[i] < 1) or (a[i] > n) then exit;
    if (d[a[i]] = 1) then exit;
    d[a[i]] := 1;
   end;
  LaHoanVi := true;
end;
```

Chương trình C#

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System;
namespace SangTao2 {
    class ThuanThe {
        const string fn = "thuanthe.inp";
        const int MN = 1001;
        static int[] a = new int[MN];
        static int[] c = new int[MN];
        static int n; // 1..n
        static int m; // so luong phan tu trong thuan the thu gon
        static void Main(string[] args){
            Doc();
            Console.WriteLine("\n n = " + n + "  m = " + m);
            Console.WriteLine("\n Cho Hoan vi: ");
            Show(a, n);
            HoanViThuanThe(a);
            Console.WriteLine("\n Tim Thuan the: ");
            Show(a, n);
            Console.WriteLine("\n Cho Thuan the: ");
            Show(a, n);
            ThuanTheHoanVi(a);
            Console.WriteLine("\n Tim Hoan vi: ");
            Show(a, n);
            Console.WriteLine("\n Cho Thuan the Thu gon: ");
            Show(c,m);
            ThuanTheThuGon(c);
            Console.WriteLine("\n Tim Hoan vi: ");
            Show(c, n);
            Console.ReadLine();
        }
        static void Doc(){
            int[] v =
         Array.ConvertAll((File.ReadAllText(fn)).Split(
              new char[] { '\0', '\n', '\t', '\r', ' ' },
                 StringSplitOptions.RemoveEmptyEntries),
                new Converter<String, int>(int.Parse));
            int i = 0;
            n = v[i++];
            for (int j = 1; j <= n; ++j) a[j] = v[i++];
            m = v[i++];
            for (int j = 1; j <= m; ++j) c[j] = v[i++];
        }
        static void HoanViThuanThe(int[] a){
            for (int i = n; i > 0; --i)
```

```
                for (int j = 1; j <= i; ++j)
                    if (a[j] >= a[i]) --a[j];
            }
            static void ThuanTheHoanVi(int[] a){
                for (int i = 1; i <= n; ++i)
                    for (int j = i; j > 0; --j)
                        if (a[j] >= a[i]) ++a[j];
            }
            static void ThuanTheThuGon(int[] c){
                Array.Copy(c, 1, c, 2, m);
                c[1] = 0; ++m;
                if (m == n) { ThuanTheHoanVi(c); return; }
                int [] b = new int [n+1];
                Array.Copy(c,1,b,1,m);
                for (int i = n; i >= 2 ; --i){
                    Array.Copy(b,1,c,1,i-1);
                    Array.Copy(b, i, c, i + 1, m - i + 1);
                    c[i] = 0;
                    ThuanTheHoanVi(c);
                    if (LaHoanVi(c)) return;
                }
            }
            static bool LaHoanVi(int[] c){
                int[] d = new int[n + 1];
                Array.Clear(d,0,d.Length);
                for (int i = 1; i <= n; ++i)
                {
                    if (c[i] < 1 || c[i] > n) return false;
                    if (d[c[i]] > 0) return false;
                    d[c[i]] = 1;
                }
                return true;
            }
            static void Show(int[] a, int n){
                for (int i = 1; i <= n; ++i)
                    Console.Write(a[i] + " ");
            }
        } // ThuanThe
    } // SangTao2
```

Độ phức tạp

Thủ tục move(a,b,M) copy m byte từ mảng a sang mảng b. Thủ tục ThuanTheThuGon có độ phức tạp N^3 vì nó gọi thủ tục ThuanTheHoanVi N lần. Hàm kiểm tra một dãy a[1..N] có phải là hoán vị đòi hỏi N thao tác và sử dụng một mảng phụ d để đánh dấu các phần tử đã xuất hiện.

4.8 Các nhà khoa học

Olimpic Quốc tế

Trong một hội nghị khoa học có n nhà khoa học (KH) tổ chức thư giãn dưới hình thức sau. Họ đặt một máy tính trong căn phòng hẹp, ngoài máy ra chỉ có thể chứa thêm 1 người, màn hình máy tính hiện số 0. Sau đó mỗi nhà khoa học buộc phải thực hiện n thao tác loại 1 và n thao tác loại 2 đan xen nhau, trong đó thao tác đầu tiên phải là loại 1.

Thao tác loại 1: Đọc	*Thao tác loại 2: Ghi*

♦ *Vào phòng;* ♦ *Đọc và ghi nhớ số trên màn hình;* ♦ *Ra khỏi phòng.*	♦ *Vào phòng;* ♦ *Lấy số nhớ trong đầu cộng thêm 1 và hiển thị kết quả trên màn hình;* ♦ *Ra khỏi phòng.*

Khi hiển thị số mới trên màn hình thì số cũ trên màn hình tự động bị xóa và người thực hiện thao tác loại 2 cũng quên luôn số đã nhớ.

Cho trước các giá trị n và m. Hãy bố trí một lịch thực hiện để các nhà khoa học hoàn thành trọn vẹn cuộc chơi theo đúng yêu cầu và số cuối cùng hiển thị trên màn hình là m.

Dữ liệu vào: Tệp văn bản **KH.INP** *chứa 2 số n và m trên một dòng cách nhau qua dấu cách.*

Dữ liệu ra: Tệp văn bản **KH.OUT** *chứa một lịch thực hiện gồm một dãy tuần tự các dòng lệnh thuộc một trong hai dạng sau:*

Dạng thứ nhất gồm hai số tự nhiên ghi cách nhau qua dấu cách, i t cho biết nhà khoa học i thực hiện thao tác t; $i \in \{1,2,...,n\}$; $t \in \{1,2\}$.

Dạng thứ hai gồm 4 số tự nhiên ghi cách nhau qua dấu cách, i t_1 t_2 k cho biết nhà khoa học i thực hiện k lần liên tiếp các thao tác t_1 và t_2 đan xen nhau; $i \in \{1,2,...,n\}$; $t \in \{1,2\}$; $k > 0$.

Nếu không có cách nào bố trí lịch thì ghi duy nhất một số 0.

Thí dụ,

KH.INP	KH.OUT	Ý nghĩa
3 4		*Có 3 nhà khoa học tham gia trò chơi thư giãn với nhiệm vụ sinh ra kết quả trên màn hình (MH) là số 4. Lịch thực hiện sẽ như sau:*
	1 1	*Người số 1: Đọc. MH = 0. Đầu(1) = 0.*
	3 1 2 3	*Người số 3: (Đọc; Ghi) 3 lần. MH = 3.*
	1 2	*Người số 1: Ghi. MH = Đầu(1)+1 = 0+1 = 1.*
	2 1	*Người số 2: Đọc. Đầu(2) = 1, MH = 1.*
	1 1 2 2	*Người số 1: (Đọc; Ghi) 2 lần. MH = 3.*
	2 2	*Người số 2 Ghi. MH = Đầu(2)+1 = 1+1 = 2.*
	2 1 2 2	*Người số 2 (Đọc;Ghi) 2 lần. MH = 2+2 = 4.*
		Chú thích: Đầu(i) - số nhớ trong đầu nhà khoa học thứ i.

Thuật toán

Ta sẽ chỉ ra rằng với mọi $n \geq 2$ và mọi m trong khoảng $2..n^2$ luôn luôn có một lịch thỏa yêu cầu của đầu bài (ta gọi là *lịch hợp lệ*) để màn hình (MH) đạt giá trị m.

Sau khi mở tệp KH.INP và đọc hai giá trị n, m ta tiến hành xếp lịch và ghi dần kết quả vào tệp KH.OUT mở sẵn. Trước hết ta nhận xét rằng nếu một người thực hiện liên tiếp k lần một cặp thao tác (Đọc - Ghi, viết tắt là ĐG), tức là sau 2k dòng lệnh

```
i  1
i  2
...
i  1
i  2
```

thì giá trị của MH được tăng thêm k đơn vị. Dãy 2k lệnh trên có thể viết gộp lại thành một lệnh 4 thành phần là **i 1 2 k.**

Ta tính $k = m \text{ div } n$ và $r = m \text{ mod } n$, ta có $m = k.n+r$, $0 \le r < n$, với ý nghĩa là để đạt được giá trị m trên MH thì phải có k người thực hiện đầy đủ và liên tiếp n cặp thao tác ĐG, ngoài ra phải có ít nhất một người thực hiện thêm r cặp thao tác ĐG. Do yêu cầu mỗi người buộc phải thực hiên n thao tác Đ và n thao tác G, một lẽ tự nhiên, ta phải sử dụng 2 người ghi nhận giá trị hiện có trên MH để khi cần sẽ trả lại giá trị đó (dĩ nhiên là phải cộng thêm 1) nhằm đảm bảo cho các thao tác cần thiết được thực hiện liên tục. Xét 3 trường hợp sau đây.

Trường hợp 1: r = 0, tức là m = k.n. Ta cần k người thực hiện liên tiếp n cặp thao tác ĐG. Tuy nhiên, do những người khác cũng phải thực hiện đầy đủ n cặp thao tác ĐG cho mỗi người, nên ta chia các thao tác thành hai loại là các *thao tác vô ích* là những thao tác đến một thời điểm nào đó sẽ có một thao tác khác đặt lại giá trị cho MH. Các thao tác còn lại được gọi là *thao tác có ích.* Như vậy trường hợp này cần có k người thực hiện tổng cộng m = k.n cặp thao tác có ích và mọi thao tác còn lại là *vô ích*. Lịch khi đó sẽ như sau.

1 1 - Người số 1 Đọc và nhớ số 0 trên màn hình. Đầu(1) = 0.

i 1 2 n ; i = k+1..n - Những người còn lại (mang mã số k+1..n) ĐG n lần vô ích.

1 2 - Người số 1 ghi số 1 lên màn hình (có ích), MH = Đầu(1)+1 = 0 + 1 = 1.

1 1 2 n-1 - Người số 1 ĐG nốt n-1 lần có ích, MH = n.

i 1 2 n ; i = 2..k - Những người số 2..k ĐG n lần có ích, MH = n+(k-1).n = k.n = m.

Trường hợp 2: r ≠ 0 và k > 0. Ta có m = k.n+r, 0 < r < n, k > 0. Trường hợp này cần n-1 người thực hiện các thao tác vô ích, 1 người thực hiện r cặp thao tác ĐG có ích và cũng chính người đó phải thực hiện (n-r) cặp thao tác vô ích. Ta sử dụng 2 người , số 1 và số 2 như sau.

1 1 - Người số 1 Đọc và nhớ số 0. Đầu(1) = 0.

i 1 2 n ; i = k+2..n - Những người mã số từ k+2 đến n ĐG n lần vô ích.

1 2 - Người số 1 Ghi 1 lên MH. MH = Đầu(1)+1 = 0 + 1 = 1.

2 1 - Người số 2 Đọc và nhớ số 1. Đầu(2) = 1.

1 1 2 n-r - Người số 1 ĐG n-r lần vô ích.

Tiếp đến là những thao tác có ích:

2 2 - Người số 2 Ghi số 2 lên MH, MH = Đầu(2)+1 = 1 + 1 = 2.

1 1 2 r-1 - Người số 1 ĐG nốt r-1 lần có ích, MH = 2+(r-1).

2 1 2 n-1 - Người số 2 ĐG nốt n-1 lần có ích, MH = 2+(r-1)+(n-1) = n+r.

i 1 2 n ; i = 3..k+1 - Những người số 3..k+1 ĐG n lần có ích, MH = n+r+(k-1).n = k.n+r.

Trường hợp 3: r ≠ 0 và k = 0, do đó m = r ≥ 2. Trường hợp này cần n-1 người thực hiện các thao tác vô ích, 1 người thực hiện r cặp thao tác ĐG có ích và cũng chính người đó phải thực hiện (n-r) căp thao tác vô ích. Ta sử dụng 2 người , số 1 và số 2 như sau.

1 1 - Người số 1 Đọc và nhớ số 0 trên MH. Đầu(1) = 0.

i 1 2 n ; i = 3..n - Những người từ số 3 đến n ĐG n lần vô ích.

2 1 2 n-1 - Người số 2 ĐG n-1 lần vô ích.

1 2 - Người số 1 Ghi số 1 lên MH. MH = Đầu(1)+1 = 0+1 = 1.

2 1 - Người số 2 Đọc số 1 trên MH. Đầu(2) = 1.

1 1 2 n-r+1 - Người số 1 ĐG n-r+1 lần vô ích.

2 2 - Người số 2 Ghi số 2 lên MH. MH = Đầu(2)+1 = 1+1 = 2.

1 1 2 r-2 - Người số 1 ĐG r-2 lần có ích, MH = 2 + (r-2) = r.

Phần dưới đây trình bày cấu trúc dữ liệu và các thủ tục đọc và xếp lịch. Hai thủ tục phụ trợ Lenh2 và Lenh4 dùng để ghi một lệnh 2 tham biến dạng i t và lệnh 4 tham biến dạng i t_1 t_2 k vào output file g KH.OUT. Trong các chú thích dưới đây d[i] là số trong đầu nhà KH thứ i, t[i] là số thao tác ĐG nhà KH i đã thực hiện, MH – màn hình, kí hiệu MH = x cho biết ta không quan tâm đến giá trị của MH vì sớm muộn giá trị này sẽ bị xóa.

```
uses crt;
const
```

```
fn = 'kh.inp'; gn = 'kh.out';
bl = #32; nl = #13#10; mn = 100;
{ bl - dấu cách; nl - xuống dòng }
type
  mi1 = array[0..mn+1] of integer;
var
f,g: text;
n, m, mh: integer;
d,t: mi1;
{ mh - Màn hình }
  d[i] - so nho trong dau,
  t[i] - con dem lenh cua nguoi i }
procedure Doc;
begin
 assign(f,fn); reset(f);
 read(f,n,m); close(f);
end;
procedure Lenh2(i,tt: integer);
begin writeln(g,i,bl,tt); end;
procedure Lenh4(i,tt1,tt2,k: integer);
begin if k > 0 then writeln(g,i,bl,tt1,bl,tt2,bl,k); end;
procedure XepLich;
var k,r,i: integer;
begin
  assign(g,gn); rewrite(g);
  if (n < 2) or (m < 2) or (m > n*n) then
  begin writeln(g,0); close(g); exit; end;
  k := m div n; { k nguoi co ich }
  r := m mod n; { va r thao tac co ich }
  if (r = 0) then
  begin
    Lenh2(1,1); {MH=0,d[1]=0,t[1]=1}
    for i := k+1 to n do Lenh4(i,1,2,n);
    {MH=x,t[i]=2n,i=k+1..n}
    Lenh2(1,2); {MH=1}
    Lenh4(1,1,2,n-1); {MH=n,t[1]=2n}
    for i := 2 to k do Lenh4(i,1,2,n);
    { MH =n+(k-1)n=kn=m,t[i]=2n,i=2..k}
    close(g); exit;
  end;
  { r > 0 }
  if k > 0 then
   begin { r,k > 0 }
    Lenh2(1,1); {MH=0,d[1]=0,t[1]=1}
    { Bo nhung nguoi vo ich }
    for i:=k+2 to n do Lenh4(i,1,2,n);
    {MH=x,t[i]=2n,i=k+2..n}
    Lenh2(1,2); {1 Ghi;MH=1,t[1]=2}
    Lenh2(2,1); {2 Doc;MH=1;d[2]=1,t[2]=1}
    { Cac thao tac vo ich cua 1 }
    Lenh4(1,1,2,n-r); {MH=x,t[1]=2+2(n-r)=2(n-r+1)}
    { Tu day la cac thao tac co ich }
    Lenh2(2,2); {MH=2,t[2]=2}
    Lenh4(1,1,2,r-1);
    {MH=2+r-1,t[1]=2(n-r+1)+2(r-1)=2n}
    Lenh4(2,1,2,n-1);{MH=2+r-1+n-1=n+r,t[2]=2n}
    for i := 3 to k+1 do Lenh4(i,1,2,n);
```

```
    {MH=n+r+(k-1)n=kn+r=m,t[i]=2n,i=3..k+1}
    close(g); exit;
   end;
  { k = 0, r > 0 }
  Lenh2(1,1); {1 Doc,d[1]=0,t[1]=1}
  { Bo nhung nguoi vo ich }
  for i:=3 to n do Lenh4(i,1,2,n);{MH=x,t[i]=2n,i=3..n}
  { n-1 thao tac vo ich cua 2 }
  Lenh4(2,1,2,n-1);{MH=x,t[2]=2(n-1)}
  Lenh2(1,2); {1 Ghi,MH=1,t[1]=2}
  Lenh2(2,1); {2 Doc,MH=1,d[2]=1,t[2]=2n-2+1=2n-1}
  { Cac thao tac vo ich cua 1 }
  Lenh4(1,1,2,n-r+1);{MH=x,t[1]=2+2(n-r+1)=2(n-r+2)}
  Lenh2(2,2); {MH=2,t[2]=2n}
  Lenh4(1,1,2,r-2);{MH=2+r-2=r=m,t[1]=2(n-r+2)+2(r-2)=2n}
  close(g);
end;
```

Bạn có thể viết thêm thủ tục test để kiểm tra xem lịch đã xếp và ghi trong tệp KH.OUT có thỏa các yêu cầu của đầu bài hay không. Thủ tục sử dụng các mảng sau đây. Mảng d[1..n], d[i] là số nhớ trong đầu người số i. Mảng t[1..n], t[i] là số lần người thứ i thực hiện các thao tác Đọc (1), Ghi (2). Do thao tác Đọc phải thực hiện trước và hai thao tác Đọc - Ghi phải đan xen nên thời điểm sát trước thao tác Đọc của người thứ i ta phải có t[i] là *số chẵn* và thời điểm sát trước thao tác Ghi phải có t[i] là *số lẻ*. Mỗi lần đọc 1 dòng lệnh thủ tục phải xét xem dòng lệnh đó chứa 2 hoặc 4 số. Thủ tục phải thực hiện các kiểm tra sau đây.

Kiểm tra lệnh dạng i v: $1 \le i \le n$, v = 1 hoặc 2. Nếu v = 1 thì t[i] phải là số chẵn, nếu v = 2 thì t[i] phải lẻ.

Kiểm tra lệnh dạng i v1 v2 k: tương tự như trên.

Thực hiện lệnh i v: Nếu v = 1 (thao tác đọc) thì gán d[i] := MH; ngược lại, nếu v = 2 (ghi) thì gán MH := d[i] + 1. Trong cả hai trường hợp đều tăng con đếm lệnh t[i] thêm 1 đơn vị.

Sau khi đọc xong tệp KH.OUT phải duyệt lại các con đếm để đảm bảo rằng d[i] = 2.n với mọi i = 1..n. Cuối cùng kiểm tra xem MH = m?

```
procedure DocLenh(var i,t1,t2,k,v: integer);
begin
  read(g,i,t1); v := 2;
  if seekeoln(g) then exit;
  readln(g,t2,k); v := 4;
end;
procedure XemLenh(i,t1,t2,k,KieuLenh: integer);
begin
  if KieuLenh = 2 then writeln(i,bl,t1)
   else writeln(i,bl,t1,bl,t2,bl,k);
end;
function Lenh(i,c: integer): Boolean;
begin
  Lenh := false;
  if (i < 1) or (i > n) then exit;
  case c of
   1: begin
        if odd(t[i]) then exit;
        inc(t[i]); d[i] := mh;
      end;
   2: begin
        if not(odd(t[i])) then exit;
        inc(t[i]); mh := d[i]+1;
      end;
```

```
    else exit;
    end;
    Lenh := true;
  end;
  function KiemTraLenh(i,t1,t2,k,v: integer): Boolean;
   var j: integer;
  begin
    if v = 2 then KiemTraLenh := Lenh(i,t1)
    else
    begin
      KiemTraLenh := false;
      for j := 1 to k do
        begin
          if not Lenh(i,t1) then exit;
          if not Lenh(i,t2) then exit;
        end;
      KiemTraLenh := true;
    end;
  end;
  procedure Test;
   var i,t1,t2,k,v,n2: integer;
  begin
    mh := 0;
    fillchar(d,sizeof(d),0);
    fillchar(t,sizeof(t),0);
    assign(g,gn); reset(g);
    while not Seekeof(g) do
    begin
       DocLenh(i,t1,t2,k,v);
       XemLenh(i,t1,t2,k,v);
       if not KiemTraLenh(i,t1,t2,k,v) then
        begin
          writeln('Sai '); close(g);
          exit;
        end;
    end;
    n2 := n+n;
    for i:=1 to n do
     if (t[i] <> n2) then
       begin
         writeln('Sai '); close(g);
         exit;
       end;
    if (mh <> m) then
       begin
         writeln('Sai '); close(g);
         exit;
       end;
    writeln('Dung');
    close(g);
  end;
```

Chương trình C#

```
// C#
  using System;
```

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

namespace SangTao2 {
    class KhoaHoc {
        const string fn = "KH.INP";
        const string gn = "KH.OUT";
        const int MN = 1002;
        static int[] d = new int[MN]; // So nho trong dau
        static int[] t = new int[MN]; // con dem thao tac
        static int n; // nha khoa hoc
        static int m; // man hinh cuoi
        static int mh;
        static StreamWriter g;
        static StreamReader f;

        static void Main(string[] args){
            Run();
        }
        static void Run() { // Kiem tra tren 1 file
            Doc();
            Console.WriteLine("n = " + n + "   m = " + m);
            XepLich();
            Console.WriteLine("Output file " + gn);
            Console.WriteLine(File.ReadAllText(gn));
            Console.WriteLine("Now Testing...");
            Test();
            Console.ReadLine();
        }
        // Kiem tra file output KH.OUT
        static void Test(){
            f = File.OpenText(gn);
            Array.Clear(d, 0, d.Length);
            Array.Clear(t, 0, t.Length);
            mh = 0;
            int i, t1, t2, k, v;
            while (DocLenh(out i, out t1, out t2, out k, out v)){
                XemLenh(i, t1, t2, k, v);
                if (!KiemTraLenh(i, t1, t2, k, v)){
                    Console.WriteLine("SAI LENH");
                    return;
                }
            }
            f.Close();
            for (int j = 1, nn = n + n; j <= n; ++j)
                if (t[j] != nn){
                    Console.WriteLine("SAI SO THAO TAC" + t[j]);
                    return;
                }
            if (mh != m) Console.WriteLine("SAI KET QUA MH");
            else Console.WriteLine(" LAP LICH DUNG "); }
        static bool KTLenh2(int i, int tt){
            switch (tt){
                case 1: if (t[i] % 2 != 0) return false;
                    ++t[i]; d[i] = mh; return true;
                case 2: if (t[i] % 2 == 0) return false;
```

```
            ++t[i]; mh = d[i] + 1; return true;
        default: return false;
    }
}
static bool KiemTraLenh(int i, int t1, int t2,
                                  int k, int v){
    if (i < 1 || i > n) return false;
    if (v == 2) return KTLenh2(i, t1);
    for (int j = 1; j <= k; ++j){
        if (!KTLenh2(i, t1)) return false;
        if (!KTLenh2(i, t2)) return false;
    }
    return true;
}
static void XemLenh(int i, int t1, int t2, int k, int v)
{
    if (v == 2) Console.WriteLine(i + " " + t1);
    else Console.WriteLine(i+" "+t1+" "+t2+" "+k);
}
static bool DocLenh(out int i, out int t1,
                    out int t2, out int k, out int v){
    i = t1 = t2 = k = v = 0;
    if (f.EndOfStream) return false;
    int[] c = Array.ConvertAll((f.ReadLine()).Split(
                new char[] { '\t', ' ' },
                StringSplitOptions.RemoveEmptyEntries),
                new Converter<String, int>(int.Parse));
    foreach (int x in c) ++v;
    Console.Write(" v = " + v + ":  ");
    if (v != 2 && v != 4) return false;
    i = c[0]; t1 = c[1];
    if (v == 4) { t2 = c[2]; k = c[3]; }
    return true;
}
static void Doc(){
    int[] v =
 Array.ConvertAll((File.ReadAllText(fn)).Split(
      new char[] { '\0', '\n', '\t', '\r', ' ' },
         StringSplitOptions.RemoveEmptyEntries),
        new Converter<String, int>(int.Parse));
    n = v[0]; // n  nha khoa hoc
    m = v[1]; // Gia tri man hinh
}
static void XepLich(){
    g = File.CreateText(gn);
    if (n < 2 || m < 2 || m > n * n){
        g.WriteLine("0");
        g.Close();
        return;
    }
    int k = m / n;
    int r = m % n;
    Console.WriteLine("k = " + k + "   r = " + r);
    if (r == 0){
        Lenh2(1, 1);
        for (int i = k + 1; i <= n; ++i)
           Lenh4(i, 1, 2, n);
```

```
                Lenh2(1, 2);
                Lenh4(1, 1, 2, n - 1);
                for (int i = 2; i <= k; ++i) Lenh4(i, 1, 2, n);
                g.Close();
                return;
            }
            if (k > 0){
                Lenh2(1, 1); // 1 Doc
                // Bo nhung nguoi vo ich
                for (int i = k + 2; i <= n; ++i)
                   Lenh4(i, 1, 2, n);
                Lenh2(1, 2); // 1 Ghi
                Lenh2(2, 1); // 2 Doc
                Lenh4(1,1,2,n-r);//cac thao tac vo ich cua 1
                // Tu day la cac thao tac co ich
                Lenh2(2, 2); // 2 Ghi
                Lenh4(1, 1, 2, r - 1); // 1 DG r-1 lan
                Lenh4(2, 1, 2, n - 1); // 2 DG n-1 lan
                for (int i = 3, k1 = k + 1; i <= k1; ++i)
                    Lenh4(i, 1, 2, n);
                g.Close();
                return;
            }
            // k = 0
            Lenh2(1, 1); // 1 Doc. Tu 3..n vo ich
            for (int i = 3; i <= n; ++i) Lenh4(i, 1, 2, n);
            Lenh4(2, 1, 2, n - 1);
            Lenh2(1, 2); // 1 Ghi
            Lenh2(2, 1); // 2 Doc
            Lenh4(1,1,2,n-r+1); // Cac thao tac vo ich cua 1
            Lenh2(2, 2); // 2 Ghi co ich
            Lenh4(1, 1, 2, r - 2); // 1 Ghi not co ich
            g.Close();
        }
        static void Lenh2(int i, int t)
        { g.WriteLine(i + " " + t); }
        static void Lenh4(int i, int t1, int t2, int k) {
           if (k > 0)
            g.WriteLine(i + " " + t1 + " " + t2 + " " + k);
        }
    } // KhoaHoc
} // SangTao2
```

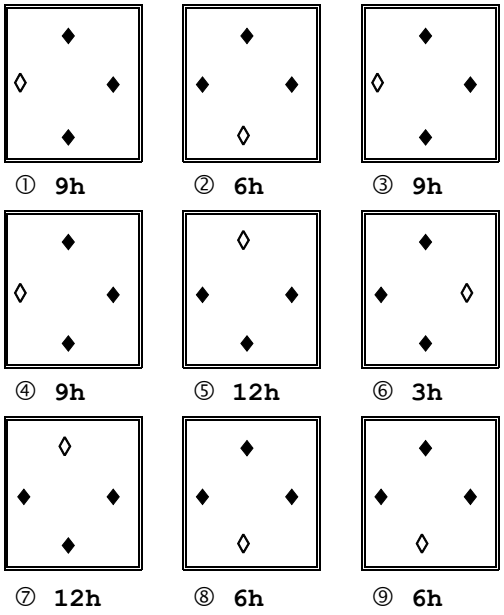
Độ phức tạp

Thuật toán phát sinh và ghi vào file kết quả tối đa $2.n^2$ dòng lệnh.

4.9 Chín chiếc đồng hồ

Olimpic Quốc tế

Có 9 chiếc đồng hồ điện tử treo trên một bảng theo sơ đồ 3 × 3. Các đồng hồ được mã số từ 1 đến 9 theo trật tự từ trái qua phải, từ hàng trên xuống hàng dưới. Mỗi đồng hồ có 4 điểm chỉ giờ ứng với các giờ 12, 3, 6 và 9. Giờ hiện trỏ của mỗi đồng hồ ứng với điểm sáng 0. Để điều khiển các đồng hồ người ta sử dụng 9 phím với các chức năng được mô tả như trong hình vẽ. Khi nhấn vào một phím thì có 4 đồng hồ đồng loạt nhảy điểm sáng đi 90^o theo chiều kim đồng hồ để cộng thêm 3 giờ tính từ giờ hiện trỏ. Thí dụ, khi nhấn phím 1 thì 4 đồng hồ 1, 2, 4 và 5 sẽ được chỉnh giờ theo luật trên. Biết giờ hiện tại của các đồng hồ. Hãy xác định một dãy ngắn nhất các phím cần nhấn để các đồng hồ đồng loạt trỏ 12 giờ.



9 đồng hồ, ◊ giờ đang trỏ

Sơ đồ bố trí 9 đồng hồ

①	②	③
④	⑤	⑥
⑦	⑧	⑨

9 phím điều khiển

1	2	3
4	5	6
7	8	9

Dữ liệu vào: tệp văn bản **DONGHO.INP** *gồm 12 dòng*

3 dòng đầu tiên gồm 9 số trỏ giờ tại thời điểm đầu bố trí theo ma trận 3×3.

Dòng thứ i trong số 9 dòng tiếp theo, i = 1, 2, 3, 4 ghi 4 số tự nhiên thể hiện mã số của 4 đồng hồ sẽ nhảy điểm sáng 90^o khi ta nhấn phím i.

Các số trên cùng dòng cách nhau qua dấu cách.

Dữ liệu ra: Tệp văn bản **DONGHO.OUT** *gồm hai thông tin:*

- Bài toán có nghiệm (ghi số 1) hoặc vô nghiệm (ghi số 0).

- Dãy phím ngắn nhất cần nhấn cho trường hợp có nghiệm.

DONGHO.INP	DONGHO.OUT
9 6 9 9 12 3 12 6 6 1 2 4 5 2 3 5 6 4 5 7 8 5 6 8 9 1 2 3 5 1 4 5 7 3 5 6 9 5 7 8 9	1 1 2 4 4

2 5 6 8	

Ý nghĩa: Nhấn lần lượt các phím 1, 2, 4, 4
cả 9 đồng hồ đều đồng loạt trỏ 12 giờ.

Thuật toán

Ta nhận thấy rằng do các đồng hồ hoạt động độc lập với nhau nên dãy phím cần nhấn là giao hoán, nghĩa là có thể liệt kê dãy phím đó theo trật tự tùy ý. Thí dụ, nhấn các phím 1, 2, 3 sẽ mang lại kết quả như khi nhấn dãy phím 2, 1, 3 hoặc theo bất kì hoán vị nào của 3 phím đó. Ngoài ra, nếu một phím được nhấn đến một bội lần của 4 thì điểm sáng trỏ giờ sẽ quay lại đúng vị trí ban đầu, do đó ta không dại gì mà nhấn một phím qúa 3 lần. Từ hai nhận xét trên ta thấy rằng có thể dùng kỹ thuật vét cạn, cụ thể là xét các tổ hợp dãy phím p[1..9], p[i] = 0..3 biểu thị số lần bấm phím i. Với mỗi tổ hợp p ta tính xem các đồng hồ được cập nhật ra sao. Nếu cả 9 đồng hồ đều nhảy về thời điểm 12 giờ thì ta chọn phương án có số lần bấm phím ít nhất trong số các tổ hợp ứng viên.

Dưới đây là một vài chi tiết sử dụng trong chương trình. Ta khởi tạo sẵn mảng hai chiều mô tả chức năng của các phím điều khiển, **phim[i,j]** cho biết khi nhấn phím i thì đồng hồ j sẽ được chỉnh.

```
var
phim: array[1..9,1..4] of
integer;
```

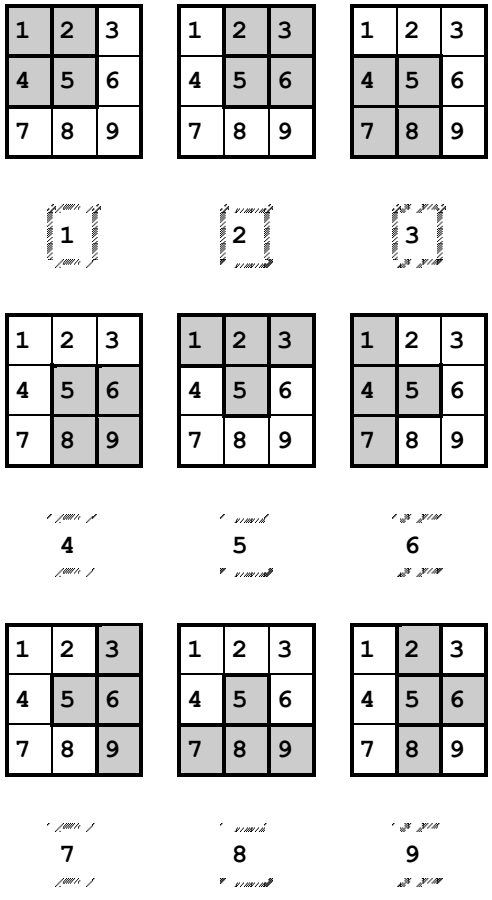
Sau khi đọc dữ liệu ứng với thí dụ cụ thể thì mảng **phim** sẽ được gán trị như sau:

```
phim  =  ((1,2,4,5),
      (2,3,5,6),
      (4,5,7,8),
      (5,6,8,9),
      (1,2,3,5),
      (1,4,5,7),
      (3,5,6,9),
      (5,7,8,9),
      (2,5,6,8));
```

Bạn cũng nên mô tả sẵn một kiểu mảng 9 phần tử để biểu thị các phím và các đồng hồ.

```
type mi1 = array[1..9] of
                       integer;
var
 dongHo: mi1;
 f: text;
 imin, imax: longint;
```

Bạn có thể sử dụng 9 vòng **for** lồng nhau ứng với 9 phím, mỗi vòng biến thiên từ 0 đến 3 ứng với số lần nhấn phím.



Chức năng của các phím điều khiển

Biến ts dùng để tính tổng số lần nhấn phím. Dễ thấy, mỗi phím được nhấn tối đa 3 lần, vậy 9 phím sẽ được nhấn tối đa 9.3 = 27 lần. Ta lấy 28 làm gía trị khởi đầu cho việc tính tsmin - tổng số lần nhấn phím ít nhất. Ta cũng nên chuyển các số trên mặt đồng hồ là (12,3,6,9) sang các số hệ 4 là (0,1,2,3) để cho tiện tính toán. Hàm Sum(p) tính tổng 9 phần tử của mảng p - đó chính là tổng số lần nhấn phím của phương án p. Hàm KiemTra(q) thực hiện việc kiểm tra xem 9 đồng hồ có cùng trỏ đến 12h hay không, q[i] là số lần đồng hồ i được cập nhật khi thực hiện phương án p.

Bạn cũng có thể sử dụng hệ 4 để xử lí như sau: Các phương án nhấn 9 phím biến thiên từ (0,0,0,0,0,0,0,0,0) đến (3,3,3,3,3,3,3,3,3) ứng với imin = 0 và imax = 4^9-1 = 2^{18}-1 = (1 shl 18) - 1 = 262143. Ta cho i biến thiên từ imin đến imax. Với mỗi i ta xây dựng phương án nhấn phím bằng cách gọi thủ tục Split(i,p) chuyển số i sang dạng biểu diễn hệ 4 để ghi vào mảng p, trong đó p[j] sẽ là số lần nhấn phím j. Biết p ta dễ dàng cập nhật các đồng hồ.

Chương trình dưới đây cài đặt 2 phương án vét cạn, Run1 – chín vòng for lồng nhau và Run2 - tính toán theo hệ 4.

(* Pascal *)

```
(* Dong ho *)
const bl = #32; fn = 'DONGHO.INP'; gn = 'DONGHO.OUT';
type mi1 = array[1..9] of integer;
var
 dongHo,kq: mi1;
 f,g: text;
 imin, imax: longint;
 s, tsmin: integer;
var
phim: array[1..9,1..4] of integer;
procedure Split(x: longint; var a: mi1);
var i: integer;
begin
  for i := 1 to 9 do
  begin
    a[i] := x mod 4;
    x := x div 4;
  end;
end;
procedure Doc;
var i,j: integer;
begin
 assign(f,fn); reset(f);
 for i:=1 to 9 do read(f,dongHo[i]);
 for i:=1 to 9 do
   for j:=1 to 4 do read(f,phim[i,j]);
 close(f);
end;
procedure Ghi;
var i,j: integer;
begin
   assign(g,gn); rewrite(g);
   if tsmin = 28 then writeln(g,0) { vo nghiem }
   else
   begin { co nghem }
     writeln(g,1);
     for i := 1 to 9 do
       for j := 1 to c[i] do write(g,i,bl);
     writeln(g);
   end;
   close(g);
end;
procedure Init;
var i: integer;
begin
 for i:=1 to 9 do dongHo[i] := (dongHo[i] div 3) mod 4;
 imin := 0;
```

```
 imax := 1;
 imax := (imax shl 18 - 1);
end;
function KiemTra(var q: mi1): Boolean;
var i: integer;
begin
    KiemTra := false;
    for i:=1 to 9 do
     if (dongHo[i]+q[i]) mod 4 <> 0 then exit;
    KiemTra := true;
end;
function Sum(var q: mi1): integer;
 var i,s: integer;
begin
  s := 0;
  for i:=1 to 9 do s := s+q[i];
  Sum := s;
end;
procedure XuLiFor;
var j,k,ts: integer;
    q,p: mi1; { p[i] - so lan nhan phim i }
    i,ikq: longint;
begin
 tsmin := 28; { = 3*9+1 }
 for p[1] := 0 to 3 do
  for p[2] := 0 to 3 do
   for p[3] := 0 to 3 do
    for p[4] := 0 to 3 do
     for p[5] := 0 to 3 do
      for p[6] := 0 to 3 do
       for p[7] := 0 to 3 do
        for p[8] := 0 to 3 do
         for p[9] := 0 to 3 do
           begin
             fillchar(q,sizeof(q),0);
             for j := 1 to 9 do
              begin
               for k := 1 to 4 do
               inc(q[phim[j,k]],p[j]);
              end;
             if KiemTra(q) then
              begin
               ts := Sum(p);
               if ts < tsmin then
               begin
                tsmin := ts;
                kq := p;
               end;
              end;
            end;
end;
procedure XuLi;
var j,k,ts: integer;
    q,p: mi1;
    i,ikq: longint;
begin
 tsmin := 28; { = 3*9+1 }
```

```
 for i:=imin to imax do
   begin
     Split(i,p); { bam phim j p[j] lan }
     fillchar(q,sizeof(q),0);
     for j := 1 to 9 do
      begin
        for k := 1 to 4 do
          inc(q[phim[j,k]],p[j]);
      end;
      if KiemTra(q) then
      begin
        ts := Sum(p);
        if ts < tsmin then
        begin
          tsmin := ts;
          ikq := i;
        end;
      end;
   end;
   Split(ikq,kq);
end;
procedure Run1;
begin
 Doc; Init;
 XuLiFor; Ghi;
end;
procedure Run2;
begin
 Doc; Init;
 XuLi; Ghi;
end;
BEGIN
 Run1;
END.
```

Chương trình C#

```
// C#
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
namespace SangTao2 {
    class DongHo {
        const string fn = "DONGHO.INP";
        const string gn = "DONGHO.OUT";
        const int cc = 9;
        static int[,] phim = new int [9,4];
        static int[] dongHo = new int[cc];
        static int [] kq = new int [cc];
        static int smin = 0; // Tong so lan nhan phim
        static void Main(string[] args) {
            Run2();
            Console.ReadLine();
        }
        static void Run1(){
```

```
    Doc(); Init(); XuLi(); Ghi(); XemKq();
}
static void Run2(){
    Doc(); Init();
    XuLiFor(); Ghi(); XemKq();
}
static void XuLiFor(){
    int [] p = new int [cc];// 9 phim
    int[] dh = new int[cc];//so lan nhay kim cua 9 DH
    int s = 0;
    smin = 28;
    for (p[0]=0; p[0] < 4; ++p[0])
     for (p[1]=0; p[1] < 4; ++p[1])
      for (p[2]=0; p[2] < 4; ++p[2])
       for (p[3]=0; p[3] < 4; ++p[3])
        for (p[4]=0; p[4] < 4; ++p[4])
         for (p[5]=0; p[5] < 4; ++p[5])
          for (p[6]=0; p[6] < 4; ++p[6])
           for (p[7]=0; p[7] < 4; ++p[7])
            for (p[8]=0; p[8] < 4; ++p[8]){
                Array.Clear(dh,0,dh.Length);
                for (int j = 0; j < cc; ++j)
                   //phim j nhan p[j] lan
                    for (int k = 0; k < 4; ++k)
                       // 4 DH chuyen kim
                        dh[phim[j, k]] += p[j];
                if (KiemTra(dh)){
                    s = Sum(p);
                    if (s < smin){
                        smin = s;
                        p.CopyTo(kq,0);
                    }
                }
            }
}
static int Sum(int []p){ // Tong so lan nhan phim
    int s = 0;
    for (int i = 0; i < cc; ++i) s += p[i];
    return s;
}
static void Doc() { // Doc du lieu tu input file
    int[] v =
        Array.ConvertAll((File.ReadAllText(fn)).Split(
        new char[] { '\0', '\n', '\t', '\r', ' ' },
         StringSplitOptions.RemoveEmptyEntries),
        new Converter<String, int>(int.Parse));
    int k = 0;
    for (int j = 0; j < cc; ++j) dongHo[j] = v[k++];
    for (int i = 0; i < cc; ++i)
        for (int j = 0; j < 4; ++j)
            phim[i, j] = v[k++];
}
static void Init() { // Khoi tri
    for (int j = 0; j < cc; ++j)
        dongHo[j] = (dongHo[j] / 3) % 4;
    for (int i = 0; i < cc; ++i)
        for (int k = 0; k < 4; ++k)
```

```
                --phim[i, k];
        }
        static void XuLi(){
            int imax = ((int)1 << 18) - 1;
            int[] p = new int[cc]; smin = 28;
            int[] q = new int[cc];
            for (int i = 0; i <= imax; ++i){
                int s = Split(i, p);
                Array.Clear(q, 0, q.Length);
                for (int j = 0; j < cc; ++j)
                    for (int k = 0; k < 4; ++k)
                        q[phim[j, k]] += p[j];
                if (KiemTra(q))
                    if (s < smin){ smin = s; p.CopyTo(kq,0); }
            }
        }
        static void Ghi(){
            StreamWriter g = File.CreateText(gn);
            if (smin < 28) { // co nghiem
                g.WriteLine(1);
                for (int i = 0; i < cc; ++i)
                    for (int j = 1; j <= c[i]; ++j)
                        g.Write((i + 1) + " ");
            }
            else  g.WriteLine(0); // vo nghiem
            g.Close();
        }
        static bool KiemTra(int[] q)// Kiem tra ca 9 DH tro ve 0?
        {
            for (int i = 0; i < cc; ++i)
                if ((dongHo[i] + q[i]) % 4 > 0) return false;
            return true;
        }
        // Tach x thanh cac chu so he 4 va tinh tong
        static int Split(int x, int[] p){
            int s = 0;
            for (int i = 0; i < cc; ++i)
            { p[i] = x % 4; s += p[i]; x /= 4; }
            return s;
        }
        static void XemKq(){
            Console.WriteLine("\n Input file " + fn);
            Console.WriteLine(File.ReadAllText(fn));
            Console.WriteLine("\n Output file " + gn);
            Console.WriteLine(File.ReadAllText(gn));
        }
    } // DongHo
} // SangTao2
```

4.10 Số duy nhất

Olimpic Baltics

Tệp văn bản UNIQUE.INP chứa dãy số, mỗi số chiếm một dòng dài không quá 20 chữ số. Trong dãy này có duy nhất một số xuất hiện đúng một lần, các số còn lại đều xuất hiện đúng k lần. Hãy tìm số duy nhất đó. Số k không cho trước, nhưng biết rằng đó là một số chẵn khác 0. Kết quả hiển thị trên màn hình.

Thuật toán

Ta dựa vào một kiến thức có từ hàng ngàn năm trước, đó là biểu diễn số theo vị trí. Ta lần lượt đọc từng dòng vào một biến string sau đó ghi vào một mảng a số lần xuất hiện của từng chữ số tại từng vị trí, a[c,i] cho biết số lần xuất hiện của chữ số c tại cột i tính từ trái qua phải.

Với thí dụ đã cho, trên cột 1 ta tính được a[‘1’,1] = 5, a[‘2’,1] = 4, a[‘3’,1] = 0,... , a[‘8’,1] = 4...

Như vậy mảng a có kích thước 10 hàng đủ chứa 10 chữ số ‘0’..’9’ và 20 cột đủ chứa các chữ số dài nhất. Sau khi đọc xong dữ liệu, ta thấy mọi phần tử của mảng a hoặc là chia hết cho k do đó là số chẵn, hoặc là số lẻ có dạng m.k + 1, m = 0, 1, 2,... Nếu a[c,i] là số lẻ thì c sẽ là chữ số xuất hiện tại cột i của số duy nhất cần tìm.

Chương trình Pascal

```
(* Pascal *)
  procedure XuLi;
  const mn = 20;
     ChuSo = ['0'..'9'];
        fn = 'unique.inp';
  var  a: array['0'..'9',1..mn] of integer;
       f: text;
       i: integer;
       s: string;
      cs: char;
  begin
    fillchar(a,sizeof(a),0);
    assign(f,fn); reset(f);
     while not seekeof(f) do
     begin
       readln(f,s);
       for i:=1 to length(s) do
         if s[i] in ChuSo then inc(a[s[i],i]);
     end;
    close(f);
    s := ''; { Ket qua }
    for i := 1 to mn do
      for cs := '0' to '9' do
        if Odd(a[cs,i]) then s := s+cs;
    writeln(s);
  end;
  BEGIN
    XuLi;
    readln;
  END.
```

UNIQUE.INP	MÀN HÌNH
1357 2008 80 1357 2008 80 2008 1357 10203040 80 2008 80 1357	10203040

Giải thích: Số duy nhất cần tìm là **10203040**. *Các số còn lại đều xuất hiện 4 lần.*

Chương trình C#

```
// C#
  using System;
  using System.Collections.Generic;
  using System.Linq;
  using System.Text;
  using System.IO;
```

```
namespace SangTao2 {
    class Unique {
        const string fn = "UNIQUE.INP";
        static void Main(string[] args){
            Console.WriteLine(XuLi());
            Console.ReadLine();
        }
        static string XuLi(){
            string s;
            StreamReader f = File.OpenText(fn);
            int mn = 20;
            int [,] a = new int[10,mn];
            Array.Clear(a, 0, a.Length);
            while (true){
                s = f.ReadLine().Trim();
                if (s.Length == 0) break;
                for (int i = 0; i < s.Length;++i)
                    if (s[i] >= '0' && s[i] <= '9')
                        ++a[s[i]-'0',i];
            }
            f.Close();
            string kq = "";
            for (int i = 0; i < mn; ++i)
                for (int cs = 0; cs <= 9; ++cs)
                    if (a[cs, i] % 2 == 1)
                        kq += cs;
            return kq;
        }
    } // Unique
} // SangTao2
```

Độ phức tạp

Nếu có n số dài tối đa m chữ số thì ta cần xét mỗi chữ số 1 lần, nghĩa là tổng cộng cần n.m thao tác. Duyệt mảng a cần 10.m thao tác là số rất nhỏ so với n.m.

Các biến thể của bài UNIQUE

1. Nếu đầu bài cho biết số k thì không cần đòi hỏi k là số chẵn.

2. Biết duy nhất một số xuất hiện đúng m lần, các số còn lại đều xuất hiện k lần như nhau, $k \neq m$ và k và m nguyên tố cùng nhau. Bạn thử suy nghĩ xem có cần biết cụ thể các giá trị của m và k không? Bạn thử tìm một số điều kiện của k và m?

3. Thay các số bằng các dãy kí tự A..Z dài tối đa m.

21.01.2010

NXH