TÀI LIỆU

BỒI DƯỠNG HỌC SINH GIỎI

MÔN: TIN HỌC

(Lưu hành nội bộ)

MỤC LỤC

PHẦN I. NHỮNG VẤN ĐỀ CHUNG VỀ KIỂM TRA ĐÁNH GIÁ

CHƯƠNG II. KỸ THUẬT CƠ BẢN KHI LẬP TRÌNH

I. Các chú ý

❖ Hiện nay trong kỳ thi học sinh giỏi Quốc gia thì người ta không còn sử dụng trình biên dịch Turbo Pascal và thay vào đó là trình biên dịch và soạn thảo Free Pascal. Đây là mã nguồn mở, vì vậy các bạn có thể download miễn phí trên mạng. Đây là công cụ mạnh hơn nhiều so với Turbo Pascal. Thực tế về giao diện nó cũng tương tự như Turbo Pascal. Nhưng nó có một số ưu điểm sau:

- Với kiểu dữ liệu số nguyên ngoài kiểu byte, integer, word, longint, longword. Thì còn có thêm kiểu int64 và qword đều là kiểu số nguyên 64 bit:

+ int64: độ lớn của nó thuộc phạm vi [-2^{63}, $2^{63}-1$]

+ qword: độ lớn của nó thuộc phạm vi [0, $2^{64}-1$]

- Vì kiểu dữ liệu xâu ngoài kiểu string ra thì nó còn có kiểu ansistring có độ lớn đến khoảng 2 GB kí tự. Nên khi làm việc với xâu có độ dài lớn hơn 255 kí tự thì ta không được khai báo bằng string. Nếu khai báo bằng string và thêm dẫn hướng biên dịch {&H+} vào đầu chương trình thì nó cũng như ansistring. Ví dụ sau đây là cách khái báo cũng giống sẽ như khai báo bằng ansistring

```
{$H+} // Chúng ta để nó ở đầu chương trình
Var   st: string;
```

- Đối với hàm Upcase() và Lowercase(): Trong free pascal nó là 2 hàm chuyển cả xâu đó thành xâu in hoa hay in thường. Còn trong Turbo pascal thì đây là 2 hàm chuyển kí tự thành in hoa hay in thường:

+ Ví dụ: `st:='abca'; st:=upcase(st);` //lúc này xâu st sẽ thành in hoa

Còn trong Turbo Pascal phải làm như thế này thì mới chuyển cả xâu st thành xâu in hoa: `for i:= 1 to length(st) do st[i]:= upcase(st[i]);`

❖ Khi lập trình không được khai báo số phần tử của mảng quá 5.10^8 phần tử. Ví dụ: `var a: array[<chỉ số đầu>..<chỉ số cuối>] of <kiểu phần tử>;`

Số phần tử của mảng = <chỉ số cuối> - <chỉ số đầu> +1 ≤ 5.10^8

- Phải biết đánh giá được độ phức tạp của thuật toán và độ lớn của các kiểu dữ liệu. Chú ý: các đề thi thông thường cho giới hạn chạy 1giây/ 1test. Trong 1 giây máy tính chấm bài của chúng ta chạy được khoảng 5.10^8 phép tính. Ví dụ: Nếu một bạn làm bài với thuật toán O(n) mà $n=10^9$ thì không thể ăn được 100% số test, nếu có ăn được 100% số test thì test đó người ta gọi là test quá yếu.

- Khi làm việc với số thực thực ta không được so sánh bằng. Nếu muốn kiểm tra X = R thì ta phải so sánh `abs(X-R)<= delta` (trong đó delta là sai số).

- Trong các kỳ thi học sinh giỏi, thì dữ liệu vào và ra đều làm việc với tệp. Trong cách dùng tệp thì thông thường có 2 cách: dùng biến tệp hoặc cách không dùng biến tệp (thay vào đó người ta dùng input và output). Cách dùng chi tiết như các code ở các phần sau.

- Khi lập trình, phải biết phong cách lập trình, ví dụ như: tên biến phải đặt phù hợp với ý nghĩa của nó, các câu lệnh tương đương nhau thì phải thẳng cột,…

Bài tập phần làm việc với tệp

1. Số nguyên tố

Cho dãy số gồm có N số nguyên dương a_1, a_2,..., a_N và một số nguyên dương K.

Yêu cầu: Hãy cho biết số lượng các phần tử có giá trị nhỏ hơn K là số nguyên tố của dãy số trên.

Dữ liệu: Vào từ File văn bản PRIME.INP gồm:

- Dòng đầu tiên là hai số N và K.
- Dòng tiếp theo lần lượt là N số nguyên của dãy số.

Kết quả: Ghi ra File PRIME.OUT gồm duy nhất số M là số lượng các phần tử của dãy số thoả mãn yêu cầu đề bài.

Giới hạn: $0 < N < 10^8$; $0 < K < 10^6$, $|a_i| < 10^9 < \forall i = 1..N$;

Ví dụ:

PRIME.INP	PRIME.OUT
7 8 1 2 3 8 7 6 11	3

***Thuật toán*:** Dùng thuật toán sàng nguyên tố để tìm đánh dấu tất cả các số nguyên tố $<10^6$. Rồi duyệt tất cả các phần tử của mảng, chỉ xét các phần tử có giá trị < K, nếu nó là số nguyên tố thì tăng kết quả lên.

Code tham khảo:

2. Tổng bình phương

Cho một số nguyên dương N. Tính tổng lập phương các chữ số của số N.

Dữ liệu: Từ tệp TONGLP.INP: gồm 1 dòng là số nguyên dương N ($N <= 10^{18}$).

Kết quả: Ghi ra tệp TONGLP.OUT một số duy nhất là tổng lập phương các chữ số của số N

Ví dụ:

TONGLP.INP	TONGLP.OUT
13	28

Code tham khảo:

```
const   fi='tonglp.inp';
        fo='tonglp.out';
var s,n: int64;
    a: longint;
BEGIN
```

```
    assign(input, fi); reset(input);
    readln(n);
    close(input);
    assign(output, fo); rewrite(output);
    s:=0;
    while n<>0 do
        begin
            a:= n mod 10;
            n:= n div 10;
            s:= s+ a*a*a;
        end;
    writeln(s);
    close(output);
END.
```

3. Cộng phân số

Cho hai phân số $\frac{A}{B}$ và $\frac{C}{D}$.

Hãy xác định 2 số nguyên dương E và F thỏa mãn 2 điều kiện sau:

+ Điều kiện 1: $\frac{E}{F} = \frac{A}{B} + \frac{C}{D}$

+ Điều kiện 2: $\frac{E}{F}$ là phân số tối giản

***Dữ liệu vào**:* từ tệp văn bản PHANSO.INP, có cấu trúc

+ Dòng 1 chứa hai số A và B.

+ Dòng 2 chứa hai số C và D.

(A, B, C, D là các số nguyên dương và không lớn hơn 10^4)

Dữ liệu ra: kết quả đưa ra tệp văn bản PHANSO.OUT

Có một dòng chứa hai số E và F tìm được thỏa mãn hai điều kiện trên. (Các số ghi trên một dòng được cách nhau bởi một dấu cách trống)

Ví dụ

PHANSO.INP	PHANSO.OUT
2 5 3 6	9 10

Code tham khảo:

```
const   fi='phanso.inp';
        fo='phanso.out';
var  a,b,c,d,e,f,u:longint;
function   ucln(m,n:longint): longint;
        begin
            while (m<>0) and (n<>0) do
                if m>n then m:=m mod n
                else n:=n mod m;
            ucln:= m+n;
        end;
BEGIN
        assign(input,fi);reset(input);
        assign(output,fo); rewrite(output);
        read(a,b,c,d);
        e:=a*d+b*c;
        f:=b*d;
        u:=ucln(e,f);
        write(e div u,' ',f div u);
```

```
        close(output);
END.
```

4. Xâu đối xứng

Một xâu được gọi là xâu đối xứng, nếu đọc từ trái qua phải cũng giống như từ phải qua trái.

Dữ liệu vào: từ tệp văn bản XAUDX.INP gồm:

+ Dòng đầu là số nguyên dương N ($n <= 10^3$)

+ N dòng tiếp theo, mỗi dòng chứa một xâu (có độ dài <= 500)

Dữ liệu ra: Ghi ra file văn bản XAUDX.OUT gồm một số duy nhất là số lượng xâu đối xứng của N xâu trên

Ví dụ:

XAUDX.INP	XAUDX.OUT
4 abca a abba adfda	3

Code tham khảo:

```
const   fi='xaudx.inp';
        fo='xaudx.out';
var     d, i, n: longint;
        st: ansistring;

function    kiemtra(st: ansistring): boolean;
    var  i, n: longint;
    begin
        n:= length(st);
        for i:= 1 to length(st) div 2 do
            if st[i]<> st[n - i+1] then exit(false);
        exit(true);
    end;
BEGIN
    assign(input, fi); reset(input);
    readln(n);d:=0;
    for i:=1 to n do
        begin
            readln(st);
            if kiemtra(st) then inc(d);
        end;
    assign(output, fo); rewrite(output);
    writeln(d);
    close(input);
    close(output);
END.
```

5. Đếm số fibonaci

Số fibonaci được tính theo công thức: $f_1 = 1$; $f_2 = 2$; $f_n = f_{n-1} + f_{n-2}$

Cho số nguyên dương N. Đếm xem có bao nhiêu số fibonaci nhỏ hơn bằng N.

***Dữ liệu vào**:* từ tệp văn bản FIBONACI.INP gồm 1 số nguyên dương N (N <=2.10^8)

Dữ liệu ra: ghi ra file FIBONACI.OUT gồm 1 số duy nhất là kết quả của bài toán.

Ví dụ:

FIBONACI.INP	FIBONACI.OUT
5	4

Code tham khảo:

```
const   fi='fibonaci.inp';
        fo='fibonaci.out';
        maxn= round(2e8);
var f0,f1,f2,n,k:longint;
    a: array[1..maxn] of boolean;
    d,i: longint;
BEGIN
 assign(input, fi); reset(input);
 assign( output, fo); rewrite(output);
 readln(n);
 fillchar(a, sizeof(a), false);
 a[1]:= true;
 f0:=1;
 f1:=1;
 while f2 <= round(2e8) do
  begin
   f2:=f0+f1;
   if f2 <= maxn then
   a[f2]:= true;
   f0:=f1;
   f1:=f2;
  end;
 d:= 0;
 for i:= 1 to n do
    if a[i] then inc(d);
 writeln(d);
 close(input); close(output);
END.
```

6. Dãy số đẹp

Người ta định nghĩa một dãy số nguyên dương $a_1, a_2,\ldots,a_n$ là dãy số đẹp như sau: nếu a_i là số chẵn thì a_{i+1} là số lẻ, ngược lại nếu a_i là số lẻ thì a_{i+1} là số chẵn ($1 \le \forall i \le n-1$).

Ví dụ như dãy số (2, 3, 4, 5) là dãy số đẹp, dãy số (2, 3, 4, 6) không phải là dãy số đẹp

Yêu cầu: Cho dãy số nguyên dương $a_1, a_2,\ldots,a_n$. Dãy số trên có phải là dãy số đẹp hay không.

Dữ liệu vào: File văn bản DAYSO.INP gồm:

- Dòng 1: số nguyên n ($2 \le n \le 10^5$)
- Dòng 2: n số nguyên dương $a_1, a_2,\ldots,a_n$ ($a_i \le 10^5$)

***Dữ liệu ra*: Ghi ra file DAYSO.OUT là YES nếu dãy số là dãy số đẹp, ngược lại không phải dãy số đẹp thì in ra NO.**

DAYSO.INP	DAYSO.OUT
4 2 3 4 5	YES

Code tham khảo:

```
const   fi='dayso.inp';
        fo='dayso.out';
var a:array[1..100000] of longint;
    n,i: longint;
    kt:boolean;

BEGIN
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(n);
    for i:=1 to n do read(a[i]);
    kt:=true;
    for i:=1 to n-1 do
        if ((a[i] mod 2 =0) and (a[i+1] mod 2=0))
            or ((a[i] mod 2 =1) and (a[i+1] mod 2 =1)) then
                begin
                        kt:=false;
                        break;
                end;
    if kt then write('YES')
        else write('NO');
    close(output);
END.
```

7. Số phong phú

Trong số học, số phong phú là số mà tổng các ước số của số đó (không kể chính nó) lớn hơn số đó. Ví dụ, số 12 có tổng các ước số (không kể 12) là 1 + 2 + 3 + 4 + 6 = 16 > 12. Do đó 12 là một số phong phú.

Yêu cầu : Hãy đếm xem có bao nhiêu số phong phú trong đoạn [L, R]

Dữ liệu: File văn bản NUMBER.INP gồm 2 số nguyên L, R (1 <= L <= R <= 10^3).

Kết quả: Ghi ra file NUMBER.OUT gồm:

Ví dụ:

NUMBER.INP	NUMBER.OUT
1 50	9

Code tham khảo:

```
const   fi='number.inp';
        fo='number.out';
var     i,count,m,n:longint;
        a:array[1..100000] of longint;

procedure    solve;
    var  s,hs,i:longint;
    begin
```

```
        for i:=1 to n div 2 do
          begin
            hs:=2;
            while (i*hs<=n) do
            begin
                a[i*hs]:=a[i*hs]+i;
                inc(hs);
            end;
          end;
    end;

BEGIN
    assign(input,fi);reset(input);
    readln(m,n);
    close(input);
    solve;
    assign(output,fo);rewrite(output);
    count:=0;
    for i:=m to n do
        if a[i]>i then inc(count);
    write(count);
    close(output);
END.
```

8. Mã bài: BCFACTOR

Cho số nguyên dương n ($2 \le n \le 10^9$), hãy phân tích n ra thừa số nguyên tố.

***Dữ liệu vào**:* File văn bản BCFACTOR.INP gồm số nguyên dương n

Dữ liệu ra: Ghi vào file BCFACTOR.OUT mỗi dòng ghi một thừa số nguyên tố và số mũ tương ứng cách nhau bởi dấu cách.Các thừa số nguyên tố in ra theo thứ tự tăng dần.

Ví dụ:

BCFACTOR.INP	BCFACTOR.OUT
4	2 2
168	2 3 3 1 7 1

Code tham khảo:

```
Const fi='bcfactor.inp';
      fo='bcfactor.out';
Var  n, i,s,j : longint;
     a:array[2..100000]of longint;
BEGIN
        Assign(input,fi);Reset(input);
        Assign(output,fo);Rewrite(output);
        Read(n);
        i:=2;a[2]:=0;
        While n<>1 do
            begin
                While n mod i<>0 do
                        begin
                                i:=i+1;
                                a[i]:=0;
                        end;
                a[i]:=a[i]+1;
                n:=n div i;
```

```
            end;
        For j:=2 to i do
                If a[j]>0 then writeln(j,' ',a[j]);
        Close(output);
END.
```

9. Nguồn bài: http://vn.spoj.com/problems/NKDIVSEQ/

Thầy Hoàng xây dựng một dãy số vô hạn A từ dãy các số nguyên dương bằng cách lần lượt xét các số tự nhiên bắt đầu từ 1 và lần lượt chọn các số cho dãy A theo quy tắc: chọn một số chia hết cho 1 (hiển nhiên là số 1), sau đó là hai số chia hết cho 2, tiếp theo là 3 số chia hết cho 3, 4 số chia hết cho 4, 5 số chia hết cho 5…. Như vậy các số đầu tiên của dãy A là: 1, 2, 4, 6, 9, 12, 16, 20, 24, 28, 30, 35, 40, 45, 50, 54, ...

Thầy Hoàng tìm ra quy luật xác định một cách nhanh chóng các phần tử của dãy. Bạn là người lập trình giỏi, hãy giúp các bạn Đội tuyển Toán viết chương trình kiểm tra quy luật mà Thầy Hoàng tìm ra có đúng hay không.

***Yêu cầu*:** Cho số tự nhiên N. Hãy xác định số thứ N của dãy số.

***Dữ liệu*:** Vào từ file NKDIVSEQ.INP chứa duy nhất số N ($1 \leq N \leq 100000$).

***Kết quả*:** Ghi ra file NKDIVSEQ.OUT là số thứ N tìm được.

Ví dụ:

NKDIVSEQ.INP	NKDIVSEQ.OUT
10	28
13	40

Code tham khảo:

```
const fi='NKDIVSEQ.INP';
      fo='NKDIVSEQ.OUT';
var   n:longint;
      a:qword;
procedure   docfile;
   begin
      assign(input,fi);
      reset(input);
      readln(n);
      assign(output,fo);
      rewrite(output);
   end;
procedure   dongfile;
   begin
      close(input);
      close(output);
   end;
procedure   xuli;
   var dem,k,i:longint;
   begin
      k:=1;   a:=1;   dem:=1;
      repeat
         if dem>=n then break;
         inc(k);
         repeat
            inc(a);
         until a mod k=0;
```

```
         inc(dem);
         if dem>=n then break;
         for i:=1 to k-1 do
            begin
               a:=a+k;
               inc(dem);
               if dem>=n then break;
            end;
      until false;
   end;
procedure   ghifile;
   begin
      writeln(a);
   end;
BEGIN
   docfile;
   xuli;
   ghifile;
   dongfile;
END.
```

10. Xóa số

Cho một số nguyên dương có N chữ số. Hãy xóa đi K số để số sau khi xóa là số bé nhất

Dữ liệu: Vào từ file văn bản XOASO.INP gồm

+ Dòng đầu tiên là số nguyên dương có N chữ số ($N <= 10^4$)

+ Dòng thứ hai là số nguyên dương K (K< N)

Kết quả: Ghi ra file XOASO.OUT gồm số nguyên bé nhất sau khi xóa K chữ số.

Ví dụ:

XOASO.INP	XOASO.OUT
151323343121223453	11233312122345

Thuật toán: Lặp K lần mỗi lần xóa đi 1 kí tự số. Nếu mà st[i] > st[i+1] thì xóa ký tự st[i] đi và ngắt.Nếu ko có trường hợp nào thỏa mãn thì xóa đi ký tự cuối cùng của xâu.

Code tham khảo:

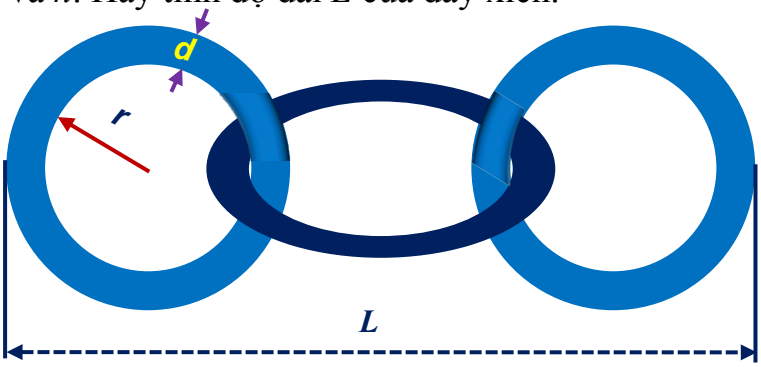
```
const fi='xoaso.inp';
      fo='xoaso.out';
var   st: ansistring;
      i,j,n,k: longint;
      kt: boolean;
BEGIN
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    readln(st);
    readln(k);
    for i:=1 to k do
      begin
          kt:= true;
          for j:= 1 to length(st)-1 do
            if st[j] > st[j+1] then
               begin
                  delete(st,j,1);
```

```
                kt:= false;
                break;
              end;
          if kt then delete(st,length(st),1);
      end;
    while (st[1]='0')and(length(st)>1) do delete(st,1,1);
    writeln(st);
    close(input); close(output);
END.
```

11. Dây xích. (Nguồn: Bài 1 Đề thi hsg Tỉnh Thanh Hóa năm học 2014 – 2015)

Người ta dùng dây thép tròn có đường kính thiết diện ngang là *d* làm *n* vòng tròn, bán kính vòng tròn trong là *r*, móc nối với nhau thành một dây xích, mỗi vòng tròn là một mắt xích. Nếu dây xích có nhiều hơn một mắt xích thì tồn tại hai vòng tròn mà mỗi vòng chỉ nối với đúng với một vòng tròn khác, đó là các mắt xích đầu và cuối. Cầm 2 mắt xích đầu và cuối, kéo căng ra, ta có dây xích độ dài *L*.

Yêu cầu: Cho *d*, *r* và *n*. Hãy tính độ dài *L* của dây xích.



Dữ liệu vào: Vào từ file văn bản BAI1.INP gồm một dòng chứa 3 số nguyên dương *d*, *r* và *n* ($d < r \le 100$; $n \le 10^9$).

Kết quả: Ghi ra file văn bản BAI1.OUT một số nguyên là độ dài *L* tìm được.

Ví dụ:

BAI1.INP	BAI1.OUT
2 10 3	64

Thuật toán: Dễ dàng nhận thấy công thức tính: ***L = 2*d + 2*r*n.***

Code tham khảo:

```
const   fi='bai1.inp';
        fo='bai1.out';
var  l: real;
     d,r,i,n: real;
BEGIN
    assign(input,fi); reset(input);
      assign(output, fo); rewrite(output);
    readln(d,r,n);
    close(input);
    l:= 2*d+2*r*n;
    writeln(l:0:0);
    close(output);
END.
```

12. (Nguồn: Bài 2 Đề thi hsg Tỉnh Thanh Hóa năm học 2014 – 2015)

Việc bảo vệ máy tính của mình để hạn chế người khác thâm nhập vào là một vấn đề đặt ra cho mọi người sử dụng máy tính. Để tăng tính an toàn trong lưu trữ Lan đã quyết định đặt mật khẩu truy cập máy tính của mình vào một xâu T với một quy ước sao cho khi cần cô ta có thể lấy lại được mật khẩu từ xâu T như sau:

Là một người yêu thích số học cô ta thường chọn mật khẩu P là một số nguyên tố và đem giấu vào trong một xâu ký tự T sao cho P chính là số nguyên tố có giá trị lớn nhất trong số các số nguyên tố được tạo từ các xâu con của T (xâu con của một xâu ký tự T là một chuỗi liên tiếp các ký tự trong T).

Ví dụ: xâu T= "Test1234#password5426" chứa mật khẩu là 23 vì T chứa các xâu con ứng với các số nguyên tố 2, 3, 23 và 5.

Yêu cầu: cho một xâu ký tự T có chiều dài không quá 500 ký tự. Tìm mật khẩu P đã dấu trong xâu T biết P có giá trị nhỏ hơn 10^5. Dữ liệu cho đảm bảo luôn có P.

***Dữ liệu vào*:** vào từ file văn bản BAI2.INP gồm 1 dòng duy nhất là xâu T.

***Kết quả*:** ghi ra file văn bản BAI2.OUT là số P tìm được.

Ví dụ:

BAI2.INP	BAI2.OUT
Test1234#password5426	23

Thuật toán:

Với bài này học sinh chú ý đọc kỹ đề bài để lấy hết dữ kiện của bài toán. Xâu T có chiều dài không quá 500 ký tự thì chúng ta có thể khai báo một mảng có tối đa 500 phần tử kiểu char hoặc khai báo kiểu xâu ansistring hoặc kiểu string (có thêm {$H+} lên đầu chương trình)

Vì bài toán luôn có đáp số và kết quả của bài toán luôn < 10^5 nên ta có thuật toán đơn giản như sau : cho i chạy từ 10^5 - 1 về 2, nếu i là số nguyên tố và số i có trong xâu thì in ra i và kết thúc luôn chương trình. Để giảm độ phức tạp của thuật toán, chúng ta có thể sử dụng thuật toán sàng nguyên tố để kiểm tra tính nguyên tố của một số.

Code tham khảo:

```
const   fi='bai2.inp';
        fo='bai2.out';
        maxn= round(1e5);
var st: ansistring;
    st1: string;
    i,j: longint;
    kt: array[1..maxn] of boolean;
procedure   sangnt;
begin
   fillchar(kt, sizeof(kt), true);
   kt[1]:= false;
   for i:=2 to trunc(sqrt(maxn)) do
      if kt[i] then
```

```
          for j:= 2 to maxn div i do
             kt[i*j]:= false;
end;
BEGIN
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    readln(st);
    sangnt;
    for i:= 100000-1 downto 2 do
        if kt[i] then
            begin
                str(i,st1);
                if pos(st1,st)<>0 then
                    begin
                        write(st1);
                        halt;
                    end;
            end;
    close(input);
    close(output);
END.
```

13. (Nguồn: Bài 3 Đề thi hsg Tỉnh Thanh Hóa năm học 2014 – 2015)

Một phân số luôn luôn có thể được viết dưới dạng số nguyên hoặc số thập phân hữu hạn hoặc số thập phân vô hạn tuần hoàn.

Ví dụ:

$$\frac{8}{2}=4;\ \frac{-23}{5}=-4.6;\ \frac{-3}{-8}=0.375;\ \frac{1}{3}=0.(3);\ \frac{-45}{56}=-0.803(571428)$$

Trong các ví dụ trên thì các chữ số đặt trong dấu ngoặc chỉ phần tuần hoàn của số thập phân.

Dữ liệu vào: vào từ file văn bản BAI3.INP gồm 2 số nguyên m và n $(|m|\le 10^8;\ |n|\le 10^8;\ n\ne 0)$.

Kết quả: ghi ra file văn bản BAI3.OUT là số nguyên hoặc số thập phân hữu hạn hoặc số thập phân vô hạn tuần hoàn của phân số $\frac{m}{n}$.

Ví dụ:

BAI3.INP	BAI3.OUT	BAI3.INP	BAI3.OUT
8 2	4	–45 56	–0.803(571428)

Thuật toán :

- Trước tiên ta phải hiểu tại sao –45 / 56 = –0.803(571428)
- Đầu tiên ta lấy 45 chia 56 được 0 dư 45

45	56
45	0

- Tiếp theo ta lấy phần dư là 45 nhân với 10 được 450, rồi chia cho 56 được 8 dư 2.

450	56
2	8

- Tương tự vậy cứ theo quy tắc trên thì phần thương chính là phần cần ghi ra đáp án còn phần tuần hoàn xuất hiện khi số dư nó xuất hiện lại lần 2 thì đó là phần tuần hoàn (nghĩa là ta không cần chia tiếp nữa vì có chia nữa nó lại lặp đi lặp lại như vậy.)
- Cụ thể thuật toán như sau : Bây giờ ta chỉ cần xét m, n là nguyên dương còn đến lúc in phần dấu – thì ta biện luận.Ta lưu các số dư vào mảng *a* và các thương vào mảng *b*. Dùng mảng *dem* để kiểm tra số lần xuất hiện của số dư a[i]. Chúng ta thực hiệu lặp để tìm ra thương và số dư theo quy tắc trên cho đến khi phép chia hết hoặc số dư nó xuất hiện lần 2 thì ta dừng. Công việc còn lại là chúng ta biện luận theo mảng *a* để in ra mảng *b* kèm theo dấu . và dấu tuần hoàn ( ) nếu có.

Code tham khảo:

```
const   fi='bai3.inp';
        fo='bai3.out';
        nmax= round(1e8);
var  n,m: longint;
     dem,a,b: array[0..nmax+10] of longint;
     i,j,k: longint;
BEGIN
    assign(input, fi); reset(input);
    readln(m, n);
    close(input);
    assign(output, fo); rewrite(output);
    fillchar(dem, sizeof(dem),0);
    if((n>0)and (m<0)) or((n<0) and(m>0)) then write('-');
    m:= abs(m); n:= abs(n);
    a[0]:= m mod n;
    b[0]:= m div n;
    dem[a[0]]:=1;
```

```
    i:=0;
    while true do
        begin
            inc(i);
            a[i]:= (a[i-1]* 10) mod n;
            b[i]:= (a[i-1] *10) div n;
            if a[i] = 0 then break;
            inc(dem[a[i]]);
            if dem[a[i]] = 2 then break;
        end;
    write(b[0]);
    if a[0] = 0 then halt;
    if a[0]<>0 then write('.');
    if a[i] = 0 then
        for j:= 1 to i do write(b[j])
    else
       begin
        for j:= 0 to i-1  do if a[i] = a[j] then break;
        for k:=1 to j do write(b[k]);
        write('(');
        for k:= j+1 to i do write(b[k]);
        write(')');
       end;
    close(output);
END.
```

PHẦN III. LẬP TRÌNH NÂNG CAO
CHƯƠNG I. ĐỆ QUY

1. Khái niệm

Chương trình con đệ quy là chương trình con mà trong thân chương trình con có lời gọi tới chính chương trình con đó.

2. Hiểu đệ quy thông qua một số ví dụ

2.1 Tính N!

```
function   gt(n:byte): qword;
begin
    if n=0 then gt:=1
    else gt:= n*gt(n-1);
end;
```

Nếu khi gọi chương trình con gt(4) thì nó sẽ thực hiện như thế nào?

+ Đầu tiên chương trình con sẽ truyền giá trị cho n=4

+ Vì 4 ≠ 0 nên gt:= 4*gt(4-1); ở đây có lời gọi hàm gt(3) lúc này nó lại truyền giá trị lên cho n = 3

+ Vì 3 ≠ 0 nên gt:= 3*gt(3-1); ở đây có lời gọi hàm gt(2) lúc này nó lại truyền giá trị lên cho n = 2

+ Vì 2 ≠ 0 nên gt:= 2*gt(2-1); ở đây có lời gọi hàm gt(1) lúc này nó lại truyền giá trị lên cho n = 1

+ Vì 1 ≠ 0 nên gt:= 1*gt(1-1); ở đây có lời gọi hàm gt(0) lúc này nó lại truyền giá trị lên cho n = 0

+ Vì 0 = 0 nên gt:= 1; Hay nói cách khác là nó trả giá trị 1 lại cho hàm. Lúc này có bao nhiêu lần lời gọi hàm trong chương trình con thì nó sẽ nhớ và thực hiện từ dưới lên (hay nói cách khác là nó quay lui lên)

2.2. Chương trình sau in ra file giaithua.out có nội dung gì? Vì sao?

```
const fo='giaithua.out';
var n: byte;
function   gt(n:byte): qword;
begin
    if n=0 then gt:=1
    else gt:= n*gt(n-1);
    writeln(n);
end;
BEGIN
   assign(output, fo); rewrite(output);
   writeln(gt(5));
   close(input); close(output);
END.
```

2.3. Chương trình A và B sau in ra màn hình nội dung gì? Vì sao?

```
Program  A;
Var   x,i: integer;
Procedure   try(i:integer);
Begin
   inc(x);
```

```
   if  i= 10 then writeln(x)
   else  try(i+1);
   Writeln(x);
End;
BEGIN
   x:= 1;
   try(7);
   readln;
END.
------------------------------
Program  B;
Var   x,i: integer;
Procedure   try(i:integer);
Begin
      inc(x);
      if  i= 14 then exit
      else  try(i+1);
      Writeln(i,' ',x);
End;
BEGIN
      x:=4 ;
      try(11);
      readln;
END.
```

2.4. Chương trình sau sẽ chạy như thế nào? Và cho kết quả trong file fibo.out?

```
const fo='fibo.out';
var   n: longint;
function fibonaci(n: byte): int64;
begin
   if n<=1 then fibonaci:= 1
   else fibonaci:= fibonaci(n-1) + fibonaci(n-2);
   writeln(n);
end;
BEGIN
    assign(output, fo); rewrite(output);
    writeln(fibonaci(5));
    close(output);
END.
```

2.5. Biểu thức Zero (Nguồn Bài 4 đề thi hsg Tỉnh Thanh Hóa năm 2009)

Cuội viết liên tiếp các số tự nhiên từ 1 đến N thành dãy: 1 2 3 ... N. Cuội đố Bờm điền các dấu phép toán + hoặc - vào giữa 2 số tự nhiên liên tiếp sao cho biểu thức thu được có kết quả bằng 0.

Yêu cầu: Bạn hãy giúp Bờm viết chương trình liệt kê tất cả các cách điền dấu phép toán thích hợp.

Dữ liệu: Vào từ file văn bản BAI4.INP gồm 1 dòng duy nhất ghi số N. N<10

Kết quả: Ghi ra file văn bản có tên BAI4.OUT:

- Dòng đầu tiên ghi số M là số cách điền dấu vào biểu thức.
- M dòng tiếp theo, mỗi dòng ghi một kết quả tìm được.

Ví dụ:

BAI4.INP	BAI4.OUT
2	0

BAI4.INP	BAI4.OUT
3	1 1+2-3=0

Thuật toán: Dùng đệ quy để sinh dãy nhị phân có độ dài N-1. Quy ước số 0 mang dấu -, còn số 1 mang dấu +. Được dãy nhị phân nào thì kiểm tra xem biểu thức đó có bằng 0 hay không. Nếu bằng 0 thì tăng biến đếm lên và lưu lại biểu thức đó dưới dạng xâu.

Code tham khảo:

```
const fi='bai4.inp';
      fo='bai4.out';
var   x: array[0..9] of 0..1;
      ch: array[1..9] of char =('1','2','3','4','5','6','7','8','9');
      xau: array[1..200] of string;
      i,j,n,res: longint;
procedure   vietch;
var   s,i: longint; st: string;
begin
   s:=1;
   st:='1';
   for i:=1 to n-1 do
      if x[i] = 0 then
         begin
            s:= s-(i+1);
            st:= st+'-'+ch[i+1];
         end
      else
         begin
            s:= s+(i+1);
            st:= st+'+'+ch[i+1];
         end;
   if s= 0 then
      begin
         inc(res);
         xau[res]:= st;
      end;
end;

procedure   try(i: longint);
var   j: longint;
begin
   for j:= 0 to 1 do
      begin
          x[i]:= j;
          if i = n -1 then vietch
          else try(i+1);
      end;
end;
BEGIN
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    readln(n);
    try(1);
    writeln(res);
    for i:=1 to res do writeln(xau[i]);
    close(input); close(output);
END.
```

CHƯƠNG II. MỘT SỐ THUẬT TOÁN SẮP XẾP

Không mất tính tổng quát, ta sẽ trình bày các thuật toán sắp xếp cho bài toán sau: Cho dãy gồm n số nguyên a_1, a_2, a_3, …a_n . Yêu cầu hãy sắp xếp thành dãy tăng không ngặt (dãy không giảm).

Tùy vào từng bài toán chúng ta đánh giá được độ phức tạp của thuật toán, để lựa chọn cách sắp xếp hợp lý.

1. Thuật toán sắp xếp nổi bọt (bubble sort)

```
for i:= 2 to n do
      for j:= n downto i do
            if a[j] < a[j-1] then doicho(a[j],a[j-1])
```

Độ phức tạp thuật toán là $O(n^2)$

Tương tự thuật toán sắp xếp nổi bọt, thuật toán sắp xếp chọn cũng có độ phức tạp là $O(n^2)$.

```
For i:= 1 to n-1 do
      For j:= i+1 to n do
            If (a[i] > a[j]) then doicho(a[i],a[j]);
```

2. Thuật toán sắp xếp bằng đếm phân phối

Ý tưởng của thuật toán là dùng mảng dem để đếm số lần xuất hiện của số a[i] trong dãy.

```
Fillchar(dem, sizeof(dem),0);
For i:= 1 to n do inc(dem[a[i]]); // dem[a[i]]:= dem[a[i]] + 1;
For i:= <giá trị nhỏ nhất của a[i]> to <giá trị lớn nhất của a[i]> do
      For j:= 1 to dem[i] do Write(i,' ');
```

3. Thuật toán sắp xếp nhanh (Quick sort)

Ý tưởng: Chọn một phần tử làm chốt (ở đây ta chọn phần tử ở vị trí giữa). Từ trái sang tìm phần tử có vị trí i lớn hơn hoặc bằng phần tử chốt, từ phải sang tìm phần tử có vị trí j bé hơn hoặc bằng phần tử chốt. Nếu i <= j thì đổi chỗ hai phần tử. Làm cho đến khi i > j. Lúc này sẽ chia ra được 2 nhóm cần sắp xếp. Làm tương tự như vậy với mỗi nhóm cho đến khi đã sắp xếp hết dãy.

```
procedure   quicksort(l,r: longint);
var i,j,tg,mid: longint;
begin
    i:=l; j:=r;
    mid:= a[(l+r) div 2];
    repeat
        while a[i]< mid do inc(i);
        while a[j] > mid do dec(j);
        if i<=j then
            begin
                tg:=a[i];
                a[i]:=a[j];
                a[j]:=tg;
                inc(i); dec(j);
            end;
    until   i> j;
    if i<r then quicksort(i,r);
    if j>l then quicksort(l,j);
```

```
end;
```
Độ phức tạp thuật toán là O(NlogN)

4. Một số bài tập ứng dụng

4.1. Số khác nhau

Cho dãy số a_1, a_2,…, a_N. Hãy đếm xem trong dãy số trên có bao nhiêu số đôi một khác nhau (hay nói cách khác là có bao nhiêu số khác nhau).

Dữ liệu: Vào từ file NUMBER.INP gồm

+ Dòng đầu là số nguyên dương N. ($N <= 10^4$)
+ Dòng thứ hai là N số nguyên dương a_1, a_2,…, a_N. ($|a_i| <= 10^9$)

Kết quả: Ghi ra file NUMBER.OUT – số lượng các các số khác nhau

Ví dụ:

NUMBER.INP	NUMBER.OUT
7 1 -1 1 2 3 4 2	5

Giải thích test: Có 5 số khác nhau là -1, 1, 2, 3, 4

Thuật toán:

+ Sắp xếp dãy số tăng dần (dãy không giảm)
+ Khởi tạo một biến dem = 1.
+ Duyệt từ vị trí 1 đến vị trí thứ n-1, nếu a[i] <> a[i+1] thì tăng dem lên 1.
+ Kết quả bài toán: in ra dem.

Code tham khảo:

```
const fi='number.inp';
      fo='number.out';
      maxn= 10000;
var   i,n: longint;
      dem: longint;
      a: array[0..maxn+1] of longint;
procedure   sapxep;
var   i,j,tg: longint;
begin
    for i:= 1 to n-1 do
      for j:= i+1 to n do
         if a[i] > a[j] then
            begin
                tg:= a[i];
                a[i]:=a[j];
                a[j]:= tg;
            end;
end;
BEGIN
   assign(input, fi); reset(input);
   assign(output, fo); rewrite(output);
   readln(n);
   for i:= 1 to n do read(a[i]);
   dem:=1;
   sapxep;
   for i:= 1 to n-1 do
      if a[i] <> a[i+1] then inc(dem);
```

```
   writeln(dem);
   close(input); close(output);
END.
```

4.2. Nguồn: http://vn.spoj.com/problems/NUMCON/

Vaxia đã viết được một số lớn trên một cuộn giấy dài và muốn khoe với anh trai Petia về thành quả vừa đạt được. Tuy nhiên, khi Vaxia vừa ra khỏi phòng để gọi anh trai thì cô em Kachia chạy vào phòng và xé rách cuộn giấy thành một số mảnh. Kết quả là trên mỗi mảnh có một hoặc vài kí số theo thứ tự đã viết.

Bây giờ Vaxia không thể nhớ chính xác mình đã viết số gì. Vaxia chỉ nhớ rằng đó là một số rất lớn.

Để làm hài lòng cậu em trai, Petia quyết định truy tìm số nào là lớn nhất mà Vaxia đã có thể viết lên cuộn giây trước khi bị xé. Bạn hãy giúp Petia làm việc này.

Dữ liệu: Vào từ file văn bản NUMCON.INP: Ghi một hoặc nhiều dòng. Mỗi dòng ghi một dãy kí số. Số dòng không vượt quá 100. Mỗi dòng ghi từ 1 đến 100 kí tự số. Bảo đảm rằng có ít nhất một dòng mà kí số đầu tiên khác 0.

Kết quả: Ghi ra file NUMCON.OUT - số lớn nhất đã có thể viết trên cuộn giấy trước khi bị xé rách.

Ví dụ:

NUMCON.INP	NUMCON.OUT
2 20 004 66	66220004
3	3

Thuật toán:

+ Bài này ta chỉ cần sắp xếp theo tiêu chí sau: Nếu xâu a[i] đứng trước xâu a[j] thì a[i] + a[j] > a[j] + a[i]. Chú ý phép cộng này chính là phép ghép hai xâu. Kết quả của bài toán là ghi lần lượt các xâu a[i] theo thứ tự đã sắp xếp.

Code tham khảo:

```
const   fi='';//'NUMCON.INP';
        fo='';//'NUMCON.OUT';
var     i,j,n:integer;
        a:array[1..100] of string;
        x,y,tg:string;
Begin
    assign(input,fi);reset(input);
    assign(output,fo);rewrite(output);
    i:=0;
    while not eof(input) do
      begin
```

```
         inc(i);
         readln(input,a[i]);
      end;
    n:=i;
    for i:= 1 to n-1 do
      for j:=i+1 to n do
         begin
            if a[i]+a[j]<a[j]+a[i] then
               begin
                  tg:=a[j];
                  a[j]:=a[i];
                  a[i]:=tg;
               end;
         end;
    for i:=1 to n do
    write(output,a[i]);
    close(output); close(input);
END.
```

4.3. Dãy số

Cho N số nguyên dương. Với mỗi cách sắp xếp N số trên thì ta được một dãy các số a_1, a_2, ... a_N. Ta định nghĩa tổng của 2 số cạnh nhau là $a_i + a_{i+1}$ ($1 \le \forall i \le N-1$). Với mỗi cách sắp xếp như vậy thì ta tìm được tổng lớn nhất của 2 số tự nhiên cạnh nhau.

Yêu cầu: Trong tất cả các tổng lớn nhất đó thì đưa ra giá trị nhỏ nhất.

Dữ liệu: Vào từ file văn bản DAYSO.INP gồm:

- Dòng đầu: Ghi số nguyên dương N ($2 \le N \le {}^{1}04$).
- N dòng tiếp theo: Mỗi dòng ghi một số tự nhiên a_i ($a_i < 10^9$).

Kết quả: Ghi ra file văn bản DAYSO.OUT một số duy nhất là kết quả của bài toán.

Ví dụ:

DAYSO.INP	DAYSO.OUT
4 2 3 4 5	7

Thuật toán:

+ Sắp xếp dãy số a[i] tăng dần, rồi tạo mảng b có cấu trúc như sau: b[1] = a[n]; b[2] = a[1]; b[3] = a[n-1]; b[4] = a[2]…

+ Mảng b vừa tạo sẽ là mảng có tổng lớn nhất của hai số cạnh nhau là nhỏ nhất.

Code tham khảo:

```
const fi='dayso.inp';
      fo='dayso.out';
var   a,b:array[1..100000] of longint;
      i,n,j:longint;
      max,min,tg,kt:longint;
```

```
BEGIN
   assign(input,fi);reset(input);
   assign(output,fo);rewrite(output);
   readln(n);
   for i:=1 to n do readln(a[i]);
   for i:=1 to n-1 do
      for j:=i+1 to n do
         if a[i] > a[j] then
            begin
               tg:=a[i];
               a[i]:=a[j];
               a[j]:=tg;
            end;
   for i:=1 to n div 2 do
      begin
         b[2*i-1]:= a[n-i+1];
         b[2*i]:= a[i];
      end;
   if n mod 2=1 then b[n]:=a[(n+1) div 2];
   max:=0;
   for i:=1 to n-1 do
      if max < b[i]+b[i+1] then max:= b[i]+b[i+1];
   Write(max);
   close(input);
   close(output);
END.
```

4.4. Qùa tặng

Nhân ngày sinh nhật, hai anh em sinh đôi Tèo và Tý nhận được N món quà, N là một số chẵn. Mỗi người gán cho mỗi món quà một giá trị ưa thích - là một số nguyên dương nhỏ hơn hoặc bằng 100. Giá trị này thể hiện mức độ hạnh phúc họ có được nếu có được món quà đó. Sau đó, 2 anh em quyết định chia quà, hai người có số lượng quà bằng nhau và bằng N/2.

Yêu cầu: Hãy xác định cách chia quà sao cho tổng mức độ hạnh phúc của hai anh em là lớn nhất.

Dữ liệu: Vào từ file văn bản GIFT.INP trong đó:

+ Dòng đầu là số nguyên dương N ≤ 500000.

+ N dòng tiếp theo, mỗi dòng ghi 2 số nguyên dương a và b tương ứng là giá trị ưa thích của Tèo và Tý với từng món quà.

Kết quả: Ghi ra file văn bản GIFT.OUT duy nhất một số là tổng mức độ hạnh phúc lớn nhất.

Ví dụ:

GIFT.INP	GIFT.OUT
4 1 2 2 3 3 5 2 1	11

Thuật toán:

+ Đọc vào 2 mảng tương ứng là a[i] và b[i]

+ Tạo mảng c[i] = a[i] – b[i]

+ Sắp xếp lại a[i] và b[i] theo chiều tăng dần của c[i]

+ Kết quả là tổng N/2 số đầu của b[i] và tổng N/2 số cuối của a[i]. Lấy như vậy mới được tổng mức độ hạnh phúc lớn nhất.

Code tham khảo:

```
Const   fi='gift.inp';
        fo='gift.out';
Var     a,b,c:array[0..500001]of longint;
        i,n:longint;
        s:int64;
Procedure   swap(var x,y:longint);
var   tg:longint;
begin
   tg:=x; x:=y; y:=tg;
end;
Procedure   quicksort(l,r:longint);
var   i,j,x:longint;
begin
   i:=l;j:=r;
   x:=c[(l+r) div 2];
   repeat
      While c[i]<x do inc(i);
      While c[j]>x do dec(j);
      If i<=j then
         begin
            swap(a[i],a[j]);
            swap(b[i],b[j]);
            swap(c[i],c[j]);
            inc(i);dec(j);
         end;
   until i>j;
   if i < r then quicksort(i,r);
   if j > l then quicksort(l,j);
end;
BEGIN
   assign(input,fi); reset(input);
   assign(output,fo); rewrite(output);
   readln(n);
   for i:=1 to n do
      begin
         read(a[i],b[i]);
         c[i]:=a[i]-b[i];
      end;
   quicksort(1,n);
   s:=0;
   for i:=1 to n div 2 do s:=s+b[i];
   for i:=n div 2+1 to n do s:=s+a[i];
   write(s);
   Close(output);
END.
```

4.5. Nguồn: http://vn.spoj.com/problems/INSUL/

Cho một dãy N viên gạch lần lượt có độ cách nhiệt là các số a_1.. a_N. Nếu xếp lần lượt các viên gạch theo trình tự đó thì độ cách nhiệt cả khối là $a_1 + a_2 + ...$

+ a_N + max(0, a_2 - a_1) + max(0, a_3 - a_2) + ... + max(0, a_N - a_{N-1}). Nhiệm vụ của bạn là tìm cách xếp sao cho độ cách nhiệt của cả khối là lớn nhất có thể.

Dữ liệu: Vào từ file INSUL.INP:

- Dòng đầu ghi số nguyên dương N (0 < n ≤ 10^5).
- N dòng sau mỗi dòng ghi một số a_i (1 ≤ i ≤ N và 1 ≤ a_i ≤ 10000).

Kết qủa: Ghi ra file INSUL.OUT - một dòng kết quả cần tìm.

***Ví dụ*:**

INSUL.INP	INSUL.OUT
4 5 4 1 7	24

Thuật toán:

+ Sắp xếp mảng a theo thứ tự tăng dần

+ Kết quả là cách xếp theo thứ tự như sau: viên gạch thứ n, viên gạch 1, viên gạch n-1, viên ghạch 2…

Code tham khảo:

```
const   fi='';//'insul.inp';
        fo='';//'insul.out';
        nmax=100000;
var     a: array[1..nmax] of longint;
        res,n,i,j: longint;
procedure   doicho(var a,b: longint);
var   tmp: longint;
begin
    tmp:= a;
    a:= b;
    b:=tmp;
end;
procedure   quicksort(l,r: longint);
var  i,j,x: longint;
begin
    i:=l;
    j:=r;
    x:= a[random(r-l+1)+l];
    repeat
        while a[i] < x do inc(i);
        while a[j] > x do dec(j);
        if i<=j then
            begin
                doicho(a[i],a[j]);
                inc(i); dec(j);
            end;
    until  i> j;
    if i<r then quicksort(i,r);
    if j > l then quicksort(l,j);
end;

BEGIN
    res:= 0;
    assign(output, fo); rewrite(output);
```

```
    assign(input,fi); reset(input);
    readln(n);
    for i:=1 to n do
        begin
            readln(a[i]);
            res:=res+a[i];
        end;
    close(input);
    quicksort(1,n);
    for i:= 1 to n div 2 do
        res:=res+ a[n-i+1] - a[i];
    writeln(res);
    close(output);
END.
```

4.6. Nguồn: http://vn.spoj.com/problems/NOIXICH/

Người ta có N đoạn dây xích (N <= 200000), mỗi đoạn dây xích là chuỗi các mắt xích được nối với nhau. Các đoạn dây xích này tách rời nhau. Mỗi đoạn mắt xích có không quá 20000 mắt xích. Bằng cách cắt ra một mắt xích, sau đó hàn lại, ta có thể nối hai dây xích thành một đoạn. Thời gian để cắt và hàn mỗi mắt xích là 1 đơn vị thời gian và được xem là bằng nhau với mọi mắt xích. Nhiêm vụ của bạn là phải nối chúng lại thành một đoạn dây xích duy nhất với thời gian ít nhất(hay số mắt xích bị cắt và hàn lại là ít nhất).

Dữ liệu: Vào từ file văn bản NOIXICH.INP

+ Dòng đầu tiên là số N, số đoạn xích.

+ Những dòng tiếp theo ghi N số nguyên dương, số thứ i là số mắt xích có trong đoạn xích thứ i(i <= i <= N) Hai số cạnh nhau trên cùng một dòng cách nhau ít nhất một dấu cách.

Kết quả: Một dòng duy nhất là số đơn vị thời gian mà bạn cần để nối N đoạn xích đã cho.

Ví dụ:

NOIXICH.INP	NOIXICH.OUT
3 2 3 4	2

Thuật toán:

+ Sắp xếp các đoạn xích theo thứ tự tăng dần (không giảm)

+ Dùng phương pháp tham lam: Khi xét để cắt các mắt xích của đoạn thứ i thì các đoạn còn lại sẽ (n-i) đoạn xích. Như vậy cần (n-i-1) mắt xích để nối (n-i) đoạn. Nếu số mắt xích của i đoạn từ 1 đến i mà bằng đúng (n-i-1) thì đáp số là (n-i-1). Ngược lại nếu mà lớn hơn thì phải cần (n-i-1) mắt xích để nối (n-i) đoạn sau và cần thêm 1 mắt xích để nối đoạn i nữa.

Code tham khảo:

```
const   nmax= 200000;
    fi='';//'noixich.inp';
    fo='';//'noixich.out';
var a:array[1..nmax] of longint;
    n,i,j,res: longint;
procedure doicho(var a,b: longint);
var tmp: longint;
begin
    tmp:=a;
    a:=b;
    b:=tmp;
end;
{procedure sapxep;
begin
    for i:=1 to n-1 do
        for j:=i+1 to n do
            if a[i]>a[j] then doicho(a[i],a[j]);
end;
}
procedure   quicksort(l,r:longint);
var i,j,x: longint;
begin
    i:=l;
    j:=r;
    x:= a[random(r-l+1)+l];
    repeat
        while a[i] < x do inc(i);
        while a[j] > x do dec(j);
        if i<=j then
            begin
                doicho(a[i],a[j]);
                inc(i);
                dec(j);
            end;
    until  i> j ;
    if i< r then quicksort(i,r);
    if j>l then quicksort(l,j);
end;
procedure   xuly;
begin
    //sapxep;
    quicksort(1,n);
    for i:=1 to n do
        begin
            res:=res+ a[i];
            if res >= n-i-1 then break;
        end;
    if res = n-i-1 then writeln(res);
    if res > n-i-1 then writeln(n-i);
end;
BEGIN
    res:=0;
    assign(input,fi); reset(input);
    readln(n);
    for i:=1 to n do read(a[i]);
    close(input);
    assign(output,fo); rewrite(output);
    xuly;
    close(output);
END.
```

4.7. Nguồn: http://vn.spoj.com/problems/BWPOINTS/

Trên trục số thực cho n điểm đen và n điểm trắng hoàn toàn phân biệt. Các điểm đen có tọa độ nguyên a1, a2, …, an còn các điểm trắng có tọa độ nguyên b1, b2, …, bn. Người ta muốn chọn ra k điểm đen và k điểm trắng để nối mỗi một điểm đen với một điểm trắng sao cho k đoạn thẳng tạo được đôi một không có điểm chung.

***Yêu cầu*:** Cho tọa độ của n điểm đen a1, a2, …, an và tọa độ của điểm trắng b1, b2, …, bn. Hãy tìm giá trị k lớn nhất thỏa mãn yêu cầu trên.

***Dữ liệu*:** Vào từ file văn bản BWPOINTS.INP gồm

- Dòng thứ nhất chứa số nguyên dương n (n <= 10^5).
- Dòng thứ hai chứa các số a1, a2, …, an (|ai| <= 10^9, i = 1, 2,…, n)
- Dòng thứ ba chứa các số b1, b2, …, bn (|bi| <= 10^9, i = 1, 2,…, n)

Các số trên cùng một dòng được ghi cách nhau ít nhất một dấu cách.

***Kết quả*:** Ghi ra file BWPOINTS.OUT một số nguyên duy nhất là số k lớn nhất tìm được

***Ví dụ*:**

BWPOINTS.INP	BWPOINTS.OUT
3 0 3 1 -3 5 -1	2

Thuật toán:

+ Ghép 2 mảng a và b lại thành một mảng kiểu bản ghi với quy ước có hai thuộc tính: một thuộc tính là giá trị và một thuộc tính là điểm đen hay điểm trắng.

+ Sắp xếp mảng kiểu bản ghi tăng dần theo giá trị.

+ Duyệt từ điểm 1 đến điểm thứ 2*n-. Với mỗi điểm i ta xem nó có thể ghép được với điểm thứ i+1 không. Điều kiện để ghép được là: điểm i và điểm thứ i+1 phải khác màu và điểm thứ i đấy phải chưa được ghép với điểm trước đó. Để kiểm tra xem điểm thứ i đã ghép chưa thì ta dùng một mảng f để đánh dấu.

Code tham khảo:

```
uses math;
const   fi='';//'BWPOINTS.inp';
        fo='';//'BWPOINTS.out';
        nmax=2*100000;
type    mang=record
            x:longint;
            y:0..1;
        end;
var  a:array[1..nmax] of mang;
     i,j,n:longint;
     f:array[0..nmax] of 0..1;
     dem:longint;
procedure   enter;
begin
   assign(input,fi);reset(input);
```

```
   readln(n);
   for i:=1 to n do
      begin
         read(a[i].x);
         a[i].y:=0;
      end;
   for i:=n+1 to 2*n do
      begin
         read(a[i].x);
         a[i].y:=1;
      end;
   close(input);
end;

Procedure Sort (L, R : longint);
Var i,j:longint;
    TG,mid: mang;
begin
   mid:=A[(L+R) div 2];
   i:=L;
   j:=R;
   repeat
      while (a[i].x<mid.x) do inc (i);
      while (a[j].x>mid.x) do dec (j);
      if i<=j then
         begin
            tg:= A[i];
            A[i] := A[j];
            A[j] :=tg;
            inc(i); dec(j);
         end;
   Until i>j;
   if L < j then Sort (L,j);
   if i < R then Sort (i,R);
end;

procedure   solve;
begin
   fillchar(f,sizeof(f),0);
   for i:=1 to 2*n-1 do
      if (a[i].y<>a[i+1].y)and (f[i]=0) then
         begin
            inc(dem);
            f[i+1]:=1;
         end;
end;
BEGIN
    assign(output,fo);rewrite(output);
    enter;
    sort(1,2*n);
    solve;
    writeln(dem);
    close(output);
END.
```

4.8. Nguồn: http://vn.spoj.com/problems/NKSGAME/

Hai bạn học sinh trong lúc nhàn rỗi nghĩ ra trò chơi sau đây. Mỗi bạn chọn trước một dãy số gồm n số nguyên. Giả sử dãy số mà bạn thứ nhất chọn là:

$b_1, b_2, ..., b_n$

còn dãy số mà bạn thứ hai chọn là

$c_1, c_2, \ldots, c_n$

Mỗi lượt chơi mỗi bạn đưa ra một số hạng trong dãy số của mình. Nếu bạn thứ nhất đưa ra số hạng b_i ($1 \le i \le n$), còn bạn thứ hai đưa ra số hạng c_j ($1 \le j \le n$) thì giá của lượt chơi đó sẽ là $|b_i+c_j|$.

Ví dụ: Giả sử dãy số bạn thứ nhất chọn là 1, -2; còn dãy số mà bạn thứ hai chọn là 2, 3. Khi đó các khả năng có thể của một lượt chơi là (1, 2), (1, 3), (-2, 2), (-2, 3). Như vậy, giá nhỏ nhất của một lượt chơi trong số các lượt chơi có thể là 0 tương ứng với giá của lượt chơi (-2, 2).

***Yêu cầu*:** Hãy xác định giá nhỏ nhất của một lượt chơi trong số các lượt chơi có thể.

Dữ liệu: Vào từ file văn bản NKSGAME.INP gồm

- Dòng đầu tiên chứa số nguyên dương n ($n \le 10^5$)
- Dòng thứ hai chứa dãy số nguyên $b_1, b_2, ..., b_n$ ($|b_i| \le 10^9$, i=1, 2, ..., n)
- Dòng thứ hai chứa dãy số nguyên $c_1, c_2, ..., c_n$ ($|c_i| \le 10^9$, i=1, 2, ..., n)

Hai số liên tiếp trên một dòng được ghi cách nhau bởi dấu cách.

Kết quả: Ghi ra file NKSGAME.OUT giá nhỏ nhất tìm được.

***Ví dụ*:**

NKSGAME.INP	NKSGAME.OUT
2 1 -2 2 3	0

Thuật toán:

+ Sắp xếp mảng b và mảng c tăng dần. Trong code tôi có viết 2 thủ tục sắp xếp, nhưng trên thực tế chúng ta chỉ cần 1 thủ tục sắp xếp bằng các chúng ta khai báo thêm tham biến.

+ Cho i chạy từ 1 và cho j chạy từ n về: Nếu b[i] + c[j] > 0 thì ta giảm j xuống, nếu b[i] + c[j] < 0 thì ra tăng i lên, còn nếu b[i] + c[j] = 0 thì đó chính là kết quả.

+ Kết quả của bài toán là min trong tất cả các trường hợp

Code tham khảo:

```
uses math;
const fi='';//NKSGAME.INP';
      fo='';//NKSGAME.OUT';
var   n:longint;
      b,c:array[0..100001] of longint;
procedure   docfile;
   var i:longint;
   begin
      assign(input,fi);
      reset(input);
```

```
      readln(n);
      for i:=1 to n do
         read(b[i]);
      for i:=1 to n do
         read(c[i]);
      assign(output,fo);
      rewrite(output);
   end;
procedure   dongfile;
   begin
      close(input);
      close(output);
   end;
procedure   quicksort1(l,r:longint);
   var i,j,tg,tg1:longint;
   begin
      if l>=r then exit;
      i:=l; j:=r;
      tg:=b[(i+j) div 2];
      repeat
         while b[i]<tg do inc(i);
         while b[j]>tg do dec(j);
         if i<=j then
            begin
               tg1:=b[i];
               b[i]:=b[j];
               b[j]:=tg1;
               inc(i);
               dec(j);
            end;
      until i>j;
      quicksort1(l,j); quicksort1(i,r);
   end;
procedure   quicksort2(l,r:longint);
   var i,j,tg,tg1:longint;
   begin
      if l>=r then exit;
      i:=l; j:=r;
      tg:=c[(i+j) div 2];
      repeat
         while c[i]<tg do inc(i);
         while c[j]>tg do dec(j);
         if i<=j then
            begin
               tg1:=c[i];
               c[i]:=c[j];
               c[j]:=tg1;
               inc(i);
               dec(j);
            end;
      until i>j;
      quicksort2(l,j); quicksort2(i,r);
   end;
procedure   xuli;
   var kq,j,f,l,i,x,y,tg:longint;
   begin
      quicksort1(1,n);
      quicksort2(1,n);
      kq:= maxlongint;
      i:=1; j:=n;
      while (i<=n) and (j>=1) do
        begin
            if kq > abs(b[i]+ c[j])then kq:= abs(b[i] +c[j]);
            if kq= 0 then break;
```

```
            if b[i] +c[j]>0 then dec(j)
            else inc(i);
         end;
       writeln(kq);
   end;
BEGIN
   docfile;
   xuli;
   dongfile;
END.
```

4.9. Nguồn: http://vn.spoj.com/problems/TFIELD/

Ở các vùng cao hiếm đất cùng mặt bằng để canh tác, khi tiến hành trồng trọt trên các sườn đồi núi có đất màu, người ta phải bạt tam cấp để tạo thành những vạt đất bằng. Khu vực đất dốc dùng để canh tác như vậy gọi là ruộng bậc thang. Hình ảnh các khu ruộng bậc thang vẫn luôn là một hình ảnh đẹp ở các vùng cao khiến du khách và các nhà nhiếp ảnh đam mê và tốn không ít phim ảnh. Gia đình Hoàng có một khu ruộng bậc thang bao quanh một ngọn đồi được chia thành các khoang bậc thang, mỗi khoang trồng một loại cây. Khi nhìn từ trên cao xuống, ta thấy các khoang bậc thang này có hình dạng của các đa giác lồi lồng nhau. Ngoại trừ khoang chứa đỉnh đồi có biên là một đa giác lồi chứa đỉnh đồi, mỗi khoang còn lại được xác định bởi hai đa giác lồng nhau: đa giác có diện tích lớn hơn được gọi là biên ngoài của khoang còn đa giác có diện tích nhỏ hơn được gọi là biên trong của khoang. Mỗi khoang có màu đặc trưng của loại cây được trồng ở khoang đó. Vốn là một người say mê chụp ảnh, muốn có một bức ảnh đẹp, Hoàng tìm cách thay đổi không quá k loại cây được trồng ở k khoang để khi nhìn từ trên cao xuống sẽ thấy một vùng cùng màu có diện tích lớn nhất. Hoàng đã ghi nhận được danh sách m đa giác lồi mô tả biên ngoài của m khoang và màu tương ứng của chúng. Do sơ xuất, Hoàng đã để các thông tin về các khoang trong danh sách bị xáo trộn, không còn được liệt kê theo đúng trình tự từ khoang trong đến khoang ngoài.

***Yêu cầu*:** Cho biết thông tin về danh sách mà Hoàng đã ghi nhận và số nguyên k, hãy tìm cách thay đổi không quá k loại cây được trồng ở k khoang để khi nhìn từ trên cao xuống sẽ thấy một vùng cùng màu có diện tích lớn nhất.

Dữ liệu: Vào từ file TFIELD.INP gồm

- Dòng đầu chứa hai số nguyên dương m, k ($k \le m$);
- Dòng thứ i trong số m dòng tiếp theo chứa thông tin về khoang thứ i trong danh sách mà Hoàng ghi nhận bao gồm:
    - Đầu tiên là số nguyên ni là số đỉnh của đa giác lồi mô tả biên ngoài của khoang;
    - Tiếp theo là số nguyên ci thể hiện màu của khoang ($1 \le c_i \le m$);

- Cuối cùng là *ni* cặp số nguyên, mỗi số có trị tuyệt đối không quá 10^9, là tọa độ của một đỉnh của đa giác. Các đỉnh của đa giác được liệt kê theo thứ tự ngược chiều kim đồng hồ.

Hai số liên tiếp trên cùng dòng được ghi cách nhau bởi dấu cách.

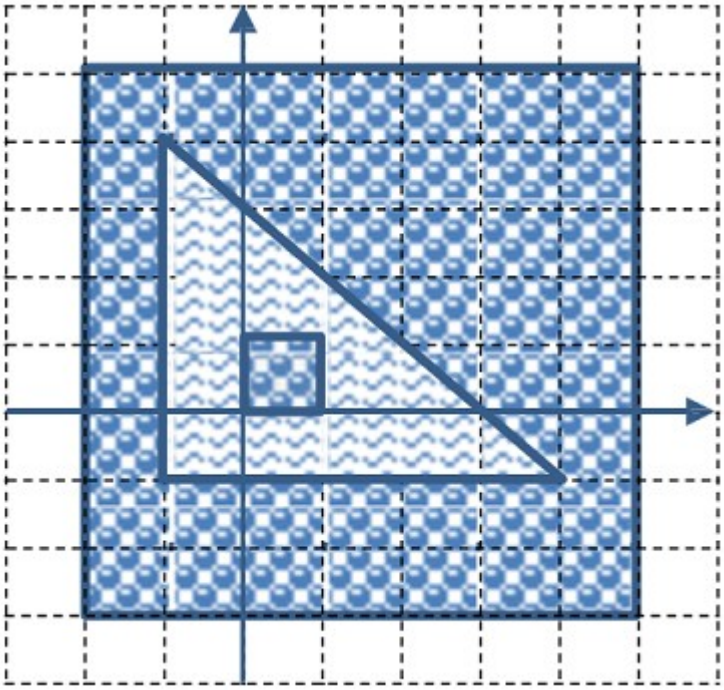
***Kết quả*:** Ghi ra file TFIELD.OUT - một số thực là diện tích vùng cùng màu lớn nhất sau khi thay đổi không quá *k* loại cây được trồng ở *k* khoang (kết quả đưa ra với độ chính xác 1 chữ số sau dấu chấm thập phân).

***Ràng buộc*:**

- Có 40% số test ứng với 40% số điểm của bài thỏa mãn điều kiện: $m \leq 10$; $k = 1$; các đa giác mô tả biên ngoài của các khoang là hình chữ nhật;
- Có 40% số test khác ứng với 40% số điểm của bài thỏa mãn điều kiện: $m \leq 10$; các đa giác mô tả biên ngoài của các khoang là tam giác;
- Có 20% số test còn lại ứng với 20% số điểm của bài thỏa mãn điều kiện: $m, ni \leq 1000$.

Ví dụ:

TFIELD.INP	TFIELD.OUT
3 1 4 1 0 0 1 0 1 1 0 1 4 1 -2 -3 5 -3 5 5 -2 5 3 2 -1 -1 4 -1 -1 4	56.0



Thuật toán:

+ Sử dụng công thức tính diện tích đa giác lồi khi biết tọa độ n điểm

$S = \frac{1}{2}|\sum_{1}^{m}(x_i - x_{i+1})(y_i + y_{i+1})|$ Trong đó tọa độ điểm thứ m+1 chính là tọa độ điểm thứ nhất.

+ Sắp xếp diện tích các đa giác theo thứ tự giảm dần

+ Sau khi sắp xếp, ta có thể tính được diện tích từng khoang (diện tích khoang thứ i là S[i]-S[i-1]).

+ Bài toán đặt ra bây giờ là tìm đoạn khoang có diện tích lớn nhất nằm giữa 2 đa giác. Đầu tiên, ta cố định điểm ngoài cùng của vùng cần xét. Từ điểm này, tìm điểm bên trong xa nhất có số khoang cần đổi màu không quá k (nói cách khác là c-st ≤ k, với c là tổng số khoang của vùng, st là số khoang cùng màu của vùng).

+ Kết quả bài toán là kết quả tối ưu trong các giá trị đã xét. Chú ý kỹ thuật lập trình của bài này ta không cần phải làm việc trên số thực, mặc dù kết quả bài toán là số thực. Các phép toán làm việc với số thực sẽ có thời gian chạy lâu hơn khi làm việc với số nguyên.

Code tham khảo:

```
Const   fi='';//'TFIELD.inp';
        fo='';//'TFIELD.out';
type    bg = record
             s: int64;
             mau: integer;
             end;
Var     a: array [0..1010] of bg;
        x,y: array [0..1010] of longint;
        i,j : integer;
        maxk, m, n, k: integer;
        res: int64;
        sl: array[0..1010] of integer;
Procedure sapxep;
var  tg: bg;
     i,j: integer;
begin
    for i:= 1 to m-1 do
      for j:= i+1 to m do
         if a[i].s < a[j].s then
            begin
                tg:= a[i];
                a[i]:= a[j];
                a[j]:= tg;
            end;
end;
function    dientich( n : integer): int64;
var   i:integer;
begin
   dientich :=0;
   For i:=1 to n do
      dientich := dientich + (x[i]-x[i+1])*(y[i+1]+y[i]);
end;
BEGIN
   Assign(input,fi);Reset(input);
   Assign(output,fo);Rewrite(output);
   Readln(m,k);
```

```
   For i:=1 to m do
      begin
         Read(n,a[i].mau);
         For j:=1 to n do read(x[j],y[j]); readln;
         x[n+1]:=x[1]; y[n+1]:=y[1];
         a[i].s := dientich(n);
      end;
   sapxep;
   res:= 0;
   a[m+1].s :=0;
   for i:= 1 to m do
      begin
         maxk:= 0; j:=i;
         fillchar(sl, sizeof(sl),0);
         while j <= m do
            begin
               inc(sl[a[j].mau]);
               if sl[a[j].mau] > maxk then maxk:= sl[a[j].mau];
               if j-i+1 - maxk <= k then
                  if res < a[i].s - a[j+1].s then
                     res:= a[i].s - a[j+1].s;
               inc(j);
            end;
      end;
   Write(res div 2);
   if res mod 2 = 0 then writeln('.0')
   else writeln('.5');
   Close(output);close(input);
END.
```

CHƯƠNG III. TÌM KIẾM NHỊ PHÂN

1. Thuật toán

Phát biểu bài toán: Cho dãy số a_1, a_2, ...a_n đã được sắp xếp không giảm. Yêu cầu: Tìm xem giá trị X có nằm trong dãy số hay không

Nếu ta dùng thuật toán tìm kiếm tuần tự thì độ phức tạp của bài toán là O(n). Vì bài toán đã cho dãy đã được sắp xếp tăng nên chúng ta dùng tìm kiếm nhị phân để giảm độ phức tạp của thuật toán xuống còn O(logN)

```
//Đoạn code tham khảo
   dau:= 1; cuoi:= n;
   while dau <= cuoi do
      begin
          giua:= (dau+ cuoi) div 2;
          if x = a[giua] then
            begin
               writeln(x,' co trong day');
               break;
            end
          else if x < a[giua] then cuoi:= giua -1
               else dau:= giua +1;
      end;
   if dau > cuoi then writeln(x,' khong co trong day');
```

2. Một số bài tập ứng dụng

2.1 Ghép cặp

Trong một bữa tiệc, có N người tham dự. Người thứ i có chiều cao là Hi. Người tổ chức bữa tiệc muốn đếm xem ông có thể ghép được bao nhiêu cặp từ N người này. Ông là một người khá vui tính, vì không muốn để cho các cặp đôi trông quá chênh lệch về chiều cao, ông đã đưa ra 1 yêu cầu: Người thứ i và người thứ j ($i \neq j$) có thể ghép cặp được với nhau nếu như thỏa mãn điều kiện sau: $90\% * Hj \leq Hi \leq Hj$. Nếu có cặp người thứ i đã ghép với người thứ j thì đảo vị trí ghép của 2 người này, thì đây cũng chỉ tính là 1 cách ghép (ví dụ: nếu có cách ghép người thứ 2 với người thứ 3 thì cũng giống như ghép người thứ 3 với người thứ 2)

Với số lượng người tham dự nhỏ ông có thể dễ dàng tính ra được số cặp có thể ghép, nhưng bữa tiệc có rất nhiều người và việc tính toán của ông trở nên khó khăn hơn.

Yêu cầu: Hãy giúp ông tính số cặp có thể ghép được.

Dữ liệu: Vào từ file văn bản GHEPCAP.INP gồm:

- Dòng đầu tiên chứa số nguyên dương N là số người tham dự bữa tiệc. ($N \leq 10^5$)
- Dòng thứ 2 chứa N số nguyên dương Hi là độ cao của N người. ($Hi \leq 10^9$)

Kết quả: Ghi ra file văn bản **GHEPCAP.OUT** gồm một dòng duy nhất là kết quả của bài toán.

Ví dụ :

GHEPCAP.INP	GHEPCAP.OUT
6 100 89 90 101 91 99	11

Thuật toán:

+ Sắp xếp tăng dần theo chiều cao

+ Duyệt từ i từ 1 đến n-1 với mỗi vị trí i thì ta chặt nhị phân để tìm người có vị trí j xa nhất mà thỏa mãn 9*h[j] <= 10*h[i]. Nếu j > i thì ta được thêm số cặp là j-i.

Code tham khảo:

```
const fi='GHEPCAP.INP';
      fo='GHEPCAP.OUT';
var h:array[0..100001] of longint;
    n:longint;
    res:int64;
    i,j: longint;
procedure   quicksort(l,r: longint);
var i,j,tmp,tg: longint;
begin
    i:= l; j:=r;
    tmp:= h[(l+r) div 2];
    repeat
       while h[i]< tmp do inc(i);
       while h[j] > tmp do dec(j);
       if i<= j then
        begin
            tg:= h[i];
            h[i]:= h[j];
            h[j]:= tg;
            inc(i); dec(j);
        end;
    until   i>j;
    if i<r then quicksort(i,r);
    if j>l then quicksort(l,j);
end;
function  check(x,y: longint): boolean;
begin
    if 9*h[y] <= 10*h[x] then exit(true);
    exit(false);
end;
procedure   xuli;
var tim, dau,giua, cuoi: longint;
begin
    res:=0;
    quicksort(1,n);
    for i:=1 to n-1 do
        begin
            tim:=0;
            dau:=i+1; cuoi:=n;
            while dau<= cuoi do
                begin
                    giua:= (dau + cuoi) shr 1;
                    if check(i,giua) then
                        begin
```

```
                            tim:=giua;
                            dau:= giua +1;
                        end
                    else cuoi:= giua -1;
                end;
            if tim >i then res:= res + tim - i;
        end;
    writeln(res);
end;
BEGIN
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    readln(n);
    for i:=1 to n do read(h[i]);
    xuli;
    close(input); close(output);
END.
```

2.2. Nguồn: http://www.spoj.com/PTIT/problems/PTIT126J/

Có N cây gỗ, có chiều cao lần lượt là A[1], A[2], .., A[n]. Bạn cần lấy một lượng gỗ độ cao tối thiểu là M bằng cách chặt từ N cây theo cách như sau: chặt tất cả những phần thừa của các cây có độ cao lớn hơn H. Hãy tìm giá trị H lớn nhất để bạn có thể lấy được lượng gỗ tối thiểu là M.

Dữ liệu: Vào từ file văn bản PTIT126J.INP

+ Dòng 1 chứa 2 số nguyên N ($1<=N<=10^6$) và M ($1 <= M <= 2.10^9$).

+ Dòng 2 chứa N số nguyên A[1], A[2], …, A[n], là chiều cao mỗi cây gỗ tương ứng ($A[i] <= 10^9$, i=1..N). Giả sử luôn tồn tại cách chặt.

Kết quả: Ghi ra file PTIT126J.OUT số H duy nhất.

Ví dụ:

PTIT126J.INP	PTIT126J.OUT
4 7 20 15 10 17	15
5 20 4 42 40 26 46	36

Giải thích test thứ nhất:

Cây 1 chặt được (20-15)=5.

Cây 4 chặt được (17-15)=2.

Tổng số gỗ chặt được nếu H = 15 là 7.

Thuật toán:

+ Bài này thực chất là bài đơn giản: Chặt nhị phân theo kết quả. Ta biết rằng kết quả của bài toán nằm trong khoảng (0, max(a[i])). Với mỗi giá trị là H thì ta kiểm tra xem tổng gỗ chặt ra có lớn hơn hoặc bằng M hay không. Nếu đúng thì ta lại chặt nhị phân trên đoạn sau. Nếu sai thì ta lại chặt nhị phân trên đoạn đầu.

Code tham khảo:

```
Const fi='';//PTIT126J.inp';
      fo='';//PTIT126J.out';
Var   i,n,h,max,j,dau,cuoi,t,k,m:longint;
      a:array[0..100001]of longint;
Function kt(h:longint):boolean;
Var s:int64;
    i:longint;
begin
   s:=0;
   For i:=1 to n do If a[i]>h then s:=s+a[i]-h;
   If s>=m then exit(true);
   exit(false);
end;
BEGIN
   Assign(input,fi);Reset(input);
   Assign(output,fo);Rewrite(output);
   Read(n,m);
   For i:=1 to n do read(a[i]); max:=a[1];
   For i:=1 to n do If a[i]>max then max:=a[i];
   dau:=0; cuoi:=max;
   while dau<=cuoi do
      begin
         h:=(dau+cuoi) div 2;
         If kt(h)=true then
            begin
               t:=h;
               dau:=h+1;
            end
         else cuoi:=h-1;
      end;
   Write(t);
   Close(output);
END.
```

2.3. Nguồn: http://www.spoj.com/PTIT/problems/P145SUMG/

Tại sân bay Nội Bài, một hành khách gồm M người chuẩn bị tham gia chuyến bay. Vì số lượng khách quá lớn nên điểm kiểm soát của sân bay đã được tăng lên thành N điểm. Tại điểm kiểm soát thứ i, cần mất T_i (s) để có thể kiểm tra xong một người (tính cả thời gian đi bộ từ địa điểm xếp hàng tới điểm kiểm tra này).

Các hành khách sắp xếp theo một hàng đợi. Lần lượt từng người vào một. Hành khách ở đầu hàng đợi được phép đi vào một trạm kiểm soát nào đó nếunhư trạm kiểm soát đó đang trống. Tuy nhiên, người đó cũng có quyền đứng chờ để đợi một trạm kiểm soát khác trống để đi tới trạm đó, vì có thể giảm thiểu chi phí thời gian cho cả đoàn (xem ví dụ 1).

Các bạn hãy tính toán xem thời gian nhỏ nhất có thể để đoàn hành khách kiểm tra xong hành lý là bao nhiêu?

Dữ liệu: Vào từ file P145SUMG.INP gồm:

+ Dòng đầu tiên gồm 2 số nguyên N và M, lần lượt là số quầy gửi đồ và số vị khách ($N <= 10^5$, $M <= 10^9$).

+ N dòng tiếp theo, mỗi dòng là số nguyên T_i (1 <= T_i <= 10^9).

Kết quả: Ghi ra file P145SUMG.OUT đáp số của bài toán.

Ví dụ:

P145SUMG.INP	P145SUMG.OUT
2 6 7 10	28
7 10 3 8 3 6 9 2 4	8

Giải thích test 1: Có 2 trạm kiểm soát và 6 hành khách.

Tại t = 0, hành khách 1 và 2 bước vào trạm kiểm tra I và II. Tại t = 7, trạm I trống, và người thứ 3 được phép đi. Tại t = 10, trạm II trống, người thứ 4 bước vào kiểm tra. Tại t = 14, trạm I trống, người thứ 5 tới kiểm tra. Tại t = 20, trạm II trống, nhưng hành khách thứ 6 sẽ đợi thêm một chút, tới t = 21, trạm I trống và hành khách này sẽ tới kiểm tra, tổng chi phí thời gian bằng 28(s).

Thuật toán:

+ Chặt nhị phân theo kết quả.

+ Kiểm tra xem với mỗi giá trị thời gian là X xem nó có thỏa mãn hay không. Nếu thỏa mãn thì ta lại chặt nhị phân trên đoạn đầu, còn nếu không thỏa mãn thì ta lại chặt nhị phân trên đoạn sau. (cách kiểm tra giá trị X có thỏa mãn hay không như code)

Code tham khảo:

```
const   fi='p145sumg.inp';
        fo='p145sumg.out';
        nmax= round(1e5);
var  n,m,dau,cuoi,giua,kq: qword;  i:longint;
     t:array[1..nmax] of longint;

function check(x:int64):boolean;
var j:longint;s:int64;
begin
   s:=0;
   for j:=1 to n do s:=s+ x div t[j];
   if s>=m then exit(true);
```

```
   exit(false);
end;
BEGIN
   assign(input,fi);reset(input);
   assign(output,fo);rewrite(output);
   readln(n,m);
   for i:=1 to n do read(t[i]);
   dau:=1; cuoi:=round(1e18);
   while dau<=cuoi do
      begin
         giua:=(dau+cuoi) div 2;
         if check(giua) then
            begin
               kq:=giua;
               cuoi:=giua-1;
            end
         else dau:=giua+1;
      end;
   write(kq);
   close(input);close(output);
END.
```

2.4. Chia đồ chơi

Nhà trẻ vừa nhận được một số lượng lớn các đồ chơi với M màu khác nhau. Các cô trông trẻ sẽ chia các đồ chơi này cho N em bé. Có các nguyên tắc khi chia đồ chơi như sau:

- Mỗi em bé sẽ chỉ nhận những đồ chơi giống màu nhau.
- Tất cả các đồ chơi đều phải được sử dụng.
- Lượng đồ chơi của em bé có nhiều đồ chơi nhất phải nhỏ nhất có thể.

Ví dụ trong trường hợp có 5 em bé, 4 đồ chơi màu đỏ và 7 đồ chơi màu xanh, một cách chia hợp lý sẽ như sau: 2 đỏ, 2 đỏ, 2 xanh, 2 xanh, 3 xanh.

Yêu cầu: Bạn không cần đưa ra cách chia cụ thể mà chỉ cần đưa ra số đồ chơi của em bé có nhiều đồ chơi nhất.

Dữ liệu vào: File văn bản TOY.INP gồm:

- Dòng 1: Gồm hai số nguyên dương N, M ($1 <= N <= 10^9$, $1 <= M <= 3.10^5$) lần lượt là số em bé và số màu đồ chơi.

- M dòng tiếp theo: Dòng i ghi một số nguyên dương a_i ($1 <= a_i <= 10^9$) là số đồ chơi có màu i.

***Dữ liệu ra*: Ghi ra file TOY.OUT là số đồ chơi của em bé có nhiều đồ chơi nhất trong cách chia tối ưu. Chú ý: dữ liệu của bài toán luôn có đáp án.**

TOY.INP	TOY.OUT
5 2 4 7	3

Thuật toán:

+ Bài này thuật toán cũng tương tự, các bạn tham khảo ở code dưới.

Code tham khảo:

```
Uses math;
Const fi='TOY.INP';
      fo='TOY.OUT';
Var tong, n, m, maxA: longint;
    A: array[0..300000] of longint;

Procedure Nhapdl;
Var i: longint;
Begin
   Assign(input,fi); Reset(input);
   Assign(output,fo); Rewrite(output);
   Readln(n,m);
   For i:=1 to m do
      Begin
         Read(A[i]);
         maxA:=max(maxA,A[i]);
      End;
   Close(input)
End;

Function kt(so: longint): boolean;
Var i, j: longint;
Begin
   j:=0;
   For i:=1 to m do
      j:=j+(a[i] div so);
   kt:=(j=n);
End;

Procedure done;
Var i, dau, cuoi, giua, kq: longint;
Begin
   dau:=1;
   cuoi:=maxA;
   While dau<=cuoi do
      Begin
         giua:=(dau+cuoi) div 2;
         If kt(giua) then
            Begin
               kq:= giua;
               dau:=giua+1
            End
         Else cuoi:=giua-1
      End;
   tong:=0;
   For i:=1 to m do
      tong:= max (tong,kq+a[i] mod cuoi);
   Writeln(tong);
   Close(output)
End;

BEGIN
   Nhapdl;
   done;
END.
```

2.5. CAKES

Nghệ nhân nấu ăn Tư Mập có thể sử dụng hệ thống gồm n bếp điện để thực hiện nấu món ăn khiến ông được vinh danh, đó là món "gatô hải sản". Thời gian

để thực hiện nấu một suất ăn như vậy trên các bếp điện tương ứng là t1, t2, …, tn giây.

Yêu cầu: Cho biết s là số lượng thực khách cần phục vụ, hãy xác định thời gian tối thiểu cần thiết để Nghệ nhân Tư Mập có thể nấu xong s suất ăn trên hệ thống bếp điện của khách sạn. Để nấu mỗi suất ăn chỉ được sử dụng một bếp.

Dữ liệu: Vào từ file văn bản CAKES.INP:

+ Dòng đầu tiên chứa số lượng suất ăn s ($0 < s < 10^{15}$) và số lượng bếp điện n ($0 < n < 20$).

+ Dòng thứ hai chứa n số nguyên dương t1, t2, …, tn, mỗi số nhỏ hơn 500.

Kết quả: Ghi ra file văn bản CAKES.OUT duy nhất một số nguyên là thời gian tối thiểu tìm được tính bằng giây.

Ví dụ:

CAKES.INP	CAKES.OUT
3 2 50 70	100

Thuật toán:

+ Tương tự cũng là bài toán chặt nhị phân theo kết quả, các bạn tham khảo thuật toán như code ở dưới

Code tham khảo:

```
const fi='CAKES.INP';
      fo='CAKES.OUT';
var res,kq,s,dau,cuoi,giua:int64;
    a:array[0..100001] of longint;
    i,n:longint;
BEGIN
   assign(input,fi); reset(input);
   assign(output,fo); rewrite(output);
   read(s,n);
   for i:=1 to n do read(a[i]);
   dau:=1;
   cuoi:=s*500;
   while dau <= cuoi do
      begin
         giua:=(dau+cuoi) shr 1;
         res:=0;
         for i:=1 to n do
            res:= res + (giua div a[i]);
         if res >= s then
            begin
               kq:=giua;
               cuoi:=giua-1;
            end
         else dau:=giua+1;
      end;
   writeln(kq);
   close(output);
END.
```

2.6. Nguồn: http://vn.spoj.com/problems/CRUELL2/

Tính lũy thừa là chưa đủ, cô giáo còn có bài tập "khủng" nữa, đó là tìm nghiệm của phương trình có dạng đa thức. Biết rằng các đa thức này có bậc cao nhất là D (1 <= D <= 11) , D là số lẻ và đa thức chỉ có duy nhất 1 nghiệm X trong khoảng -1,000,000 <= X <= 1,000,000; X là nghiệm nếu nó làm cho giá trị của đa thức xấp xỉ 0 hoặc đúng bằng 0.

Cho đa thức cùng với các hệ số thực (-500 <= hệ số bậc i <= 500), hãy tìm giá trị X với độ chính xác đến 0.0005. Khi tìm được X rồi thì ghi ra phần nguyên của X*1,000 (chú ý không làm tròn X).

Ví dụ, đa thức bậc 3: 1.5*x^3 - 10 = 0 có 1 nghiệm X = 1.88207. Như vậy phải ghi ra 1882.

Các bậc của đa thức là tính từ 0..D.

Không có đáp án nào có nhiều hơn 6 chữ số có nghĩa và mỗi đáp án đều đủ nhỏ để tăng 0.0001 (với kiểu dữ liệu double) thì cũng không làm mất đi tính chính xác.

Gợi ý: Tìm 1 chiến lược để thu hẹp khoảng của nghiệm cần tìm kiếm mỗi lần "thử" 1 giá trị X nào đó.

Dữ liệu: Vào từ file CRUELL2.INP gồm:

+ Dòng 1: Một số nguyên: D

+ Dòng 2..D+2: Dòng i+2 chứa 1 số thực: hệ số của bậc i

Kết quả: Ghi ra file CRUELL2.OUT một số nguyên là phần nguyên của 1,000 nhân với nghiệm X tìm được.

Ví dụ:

CRUELL2.INP	CRUELL2.OUT
3 -10.0 0.0 0.0 1.50	1882

Thuật toán:

+ Theo đề bài: chúng ta biết giá trị của nghiệm thuộc đoạn $[-10^6, 10^6]$ và chỉ có đúng một nghiệm.

+ Bài này là bài toán chặt nhị phân trên số thực.

+ Ta so sánh giá trị của đa thức tại $X= -10^6$ và $X= 10^6$. Nếu $f(-10^6) > f(10^6)$ thì đồ thị trong khoảng đó đi từ trên xuống dưới. Ngược lại thì đồ thị đi từ dưới lên trên. Với mỗi trường hợp đó thì ta chặt nhị phân theo nó.

Code tham khảo:

```
const   fi='';//'cruell2.inp';
        fo='';//'cuell2.out';
        nmax= 12;
```

```
        delta=0.0005 ;
var     f:real;
        a:array[0..nmax] of real;
        i,n:0..12;
        tg,giua,dau,cuoi:real;
function    giatri(x:real):real;
    var kq,tg:real;
    begin
           kq:=0; tg:=1;
            for i:=0 to n do
              begin
                kq:=kq+tg*a[i];
                tg:=tg*x;
              end;
            giatri:=kq;
    end;
BEGIN
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    readln(n);
    for i:=0 to n do
        begin
            readln(a[i]);
        end;
    close(input);
    dau:=-1000000;
    cuoi:=1000000;
    if giatri(dau)<giatri(cuoi) then
       while dau<=cuoi do
         begin
             giua:=(dau+cuoi)/2;
             f:=0; tg:=1;
             for i:=0 to n do
               begin
                 f:=f+tg*a[i];
                 tg:=giua*tg;
               end;
             if f>delta then cuoi:=giua - 0.0001
             else if f<-delta then dau:=giua+0.0001
             else  break;
         end
     else
       while dau<=cuoi do
         begin
             giua:=(dau+cuoi)/2;
             f:=0; tg:=1;
             for i:=0 to n do
               begin
                 f:=f+tg*a[i];
                 tg:=giua*tg;
               end;
             if f>delta then dau:=giua +0.0001
             else if f<delta then cuoi:=giua-0.0001
             else  break;
         end;
    writeln(trunc(giua*1000));
    close(output);
END.
```

CHƯƠNG IV. MỘT SỐ BÀI TOÁN QUY HOẠCH ĐỘNG TIÊU BIỂU

1. Khái niệm về phương pháp quy hoạch động:

Phương pháp quy hoạch động dùng để giải bài toán tối ưu có bản chất đệ quy, tức là việc tìm phương án tối ưu cho bài toán đó có thể đưa về tìm phương án tối ưu của một số hữu hạn các bài toán con.

Đối với một số bài toán đệ quy, nguyên lý chia để trị (divide and conquer) thường đóng vai trò chủ đạo trong việc thiết kế thuật toán. Để giải quyết một bài toán lớn, ta chia nó thành nhiều bài toán con cùng dạng với nó để có thể giải quyết độc lập.

Trong phương án quy hoạch động, nguyên lý chia để trị càng được thể hiện rõ: Khi không biết phải giải quyết những bài toán con nào, ta sẽ đi giải quyết toàn bộ các bài toán con và lưu trữ những lời giải hay đáp số của chúng với mục đích sử dụng lại theo một sự phối hợp nào đó để giải quyết những bài toán tổng quát hơn.

Đó chính là điểm khác nhau giữa Quy hoạch động và phép phân giải đệ quy và cũng là nội dung phương pháp quy hoạch động:

- Phép phân giải đệ quy bắt đầu từ bài toán lớn phân ra thành nhiều bài toán con và đi giải từng bài toán con đó. Việc giải từng bài toán con lại đưa về phép phân ra tiếp thành nhiều bài toán nhỏ hơn và lại đi giải các bài toán nhỏ hơn đó bất kể nó đã được giải hay chưa.

- Quy hoạch động bắt đầu từ việc giải tất cả các bài toán nhỏ nhất (bài toán cơ sở) để từ đó từng bước giải quyết nhưng bài toán lớn hơn, cho tới khi giải được bài toán lớn nhất (bài toán ban đầu).

Bài toán giải theo phương pháp quy hoạch động gọi là bài toán quy hoạch động.

Công thức phối hợp nghiệm của các bài toán con để có nghiệm của bài toán lớn gọi là công thức truy hồi của quy hoạch động.

Tập các bài toán có ngay lời giải để từ đó giải quyết các bài toán lớn hơn gọi là cơ sở quy hoạch động.

Không gian lưu trữ lời giải các bài toán con để tìm cách phối hợp chúng gọi là bảng phương án của quy hoạch động.

Trước khi áp dụng phương pháp quy hoạch động ta phải xét xem phương pháp đó có thỏa mãn những yêu cầu dưới đây không:

- Bài toán lớn phải phân rã được thành nhiều bài toán con, mà sự phối hợp lời giải của các bài toán con đó cho ta lời giải của bài toán lớn.

- Vì quy hoạch động là đi giải tất cả các bài toán con, nên nếu không đủ không gian vật lý lưu trữ lời giải (bộ nhớ, đĩa, …) để phối hợp chúng thì phương pháp quy hoạch động cũng không thể thực hiện được.

- Quá trình từ bài toán cơ sở tìm ra lời giải bài toán ban đầu phải qua hữu hạn bước. Các bước cài đặt một chương trình sử dụng quy hoạch động:

+ Giải tất cả các bài toán cơ sở (thông thường rất dễ), lưu các lời giải vào bảng phương án.

+ Dùng công thức truy hồi phối hợp những lời giải của các bài toán nhỏ đã lưu trong bảng phương án để tìm lời giải của các bài toán lớn hơn rồi lưu chúng vào bảng phương án. Cho tới khi bài toán ban đầu tìm được lời giải.

+ Dựa vào bảng phương án, truy vết tìm ra nghiệm tối ưu.

Cho tới nay, vẫn chưa có một định lý nào cho biết một cách chính xác những bài toán nào có thể giải quyết hiệu quả bằng quy hoạch động. Tuy nhiên để biết được bài toán có thể giải bằng quy hoạch động hay không, ta có thể đặt câu hỏi:

1. "Một nghiệm tối ưu của bài toán lớn có phải là sự phối hợp các nghiệm tối ưu của các bài toán con hay không?"

2. "Liệu có thể nào lưu trữ được nghiệm các bài toán con dưới một hình thức nào đó để phối hợp tìm được ngiệm bài toán lớn?".

2. Ví dụ về các bài toán có thể giải bằng phương pháp quy hoạch động

2.1. Bài toán Tính N! **GT.PAS**

Ta có định nghĩa như sau: $n! = \begin{cases} 1 & \text{nếu } n = 0 \\ n*(n-1)! & n > 0 \end{cases}$

Cho một số nguyên dương n ($0 \le n \le 20$).

***Yêu cầu*:** Hãy tính $n!$ bằng phương pháp quy hoạch động (lập bảng phương án).

***Dữ liệu*:** Vào từ file văn bản GT.INP gồm 1 số nguyên dương n.

Kết quả: Ghi ra file văn bản GT.OUT gồm 1 dòng là giá trị của $n!$

Ví dụ:

GT.INP	GT.OUT
3	6

***Thuật toán*:**

Gọi f[i] là giá trị của i!

Vì ta biết n! = n*(n-1)!. Từ đây ta có công thức quy hoạch động sau:

```
f[i] := f[i-1]*i;
```

Kết quả của bài toán là f[n].

Code tham khảo:

```
// Độ phức tạp O(n)
const fi='GT.INP';
      fo='GT.OUT';
var   n: integer;
      f: array[0..30] of qword;

procedure   giaithua;
var   i: integer;
begin
   f[0]:=1;
   for i:=1 to n do f[i]:= f[i-1]*i;
end;
BEGIN
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    readln(n);
    giaithua;
    writeln(f[n]);
    close(input); close(output);
END.
```

2.2. Bài toán Tính dãy Fibonaci FIBO.PAS

Ta có định nghĩa như sau: $F(n) = \begin{cases} 1 & \text{nếu } n = 0 \text{ or } n = 1 \\ F(n-1) + F(n-2) & n > 1 \end{cases}$

Cho một số nguyên dương n ($0 \le n \le 50$).

Yêu cầu: Hãy tính $F(n)$ bằng phương pháp quy hoạch động (lập bảng phương án).

Dữ liệu: Vào từ file văn bản FIBO.INP gồm 1 số nguyên dương n.

Kết quả: Ghi ra file văn bản FIBO.OUT là giá trị tính được của $F(n)$.

Ví dụ:

FIBO.INP	FIBO.OUT
5	8

Thuật toán:

Gọi F[i] là giá trị Fibonaci thứ i.

Bài toán cơ sở: F[0] = 1; F[1] = 1;

Ta có công thức quy hoạch động như sau:

F[i] := F[i-1] + F[i-2];

Kết quả của bài toán chính là F[n].

Code tham khảo:

```
// Độ phức tạp O(n)
const fi='fibo.inp';
      fo='fibo.out';
var   n: longint;
      i: longint;
      f: array[0..51] of int64;
BEGIN
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    readln(n);
    f[0]:= 1;  f[1]:= 1;
    for i:= 2 to n do f[i]:= f[i-1] + f[i-2];
```

```
    writeln(f[n]);
    close(input); close(output);
END.
```

2.3. Bài toán Tính tổng của dãy số

Cho dãy số nguyên gồm n phần tử $a_1, a_2, \ldots, a_n$ và hai số nguyên dương p và q ($1 \le p \le q \le n$).

Yêu cầu: Hãy tính tổng của các phần tử liên tiếp từ $a_p \ldots a_q$.

Dữ liệu: Vào từ file văn bản SUM.INP có cấu trúc như sau:

- *Dòng 1:* Ghi số nguyên dương n và k, hai số được ghi cách nhau một dấu cách. ($1 \le k, n \le 10^5$)
- *Dòng 2:* Ghi n số nguyên $a_1, a_2, \ldots, a_n$, các số được ghi cách nhau ít nhất một dấu cách ($-32000 \le a_i \le 32000$).
- *Dòng thứ i trong k dòng tiếp theo:* Mỗi dòng ghi hai số nguyên dương p_i và q_i, hai số được ghi cách nhau một dấu cách ($1 \le p_i \le q_i \le n$).

Kết quả: Ghi ra file văn bản SUM.OUT theo cấu trúc như sau:

- *Dữ liệu được ghi trên k dòng:* Dòng thứ i ghi một số nguyên là tổng giá trị của các phần tử trong đoạn $a_{p_i} \ldots a_{q_i}$

Ví dụ:

SUM.INP	SUM.OUT
5 3 2 9 -3 5 8 1 5 2 3 4 4	21 6 5

Thuật toán:

Gọi S[i] là tổng giá trị các phần tử $a_1, a_2, \ldots, a_i$ ($1 \le i \le n$).

Ta có công thức quy hoạch động để tính S[i] như sau:

S[i] := S[i - 1] + A[i];

Như vậy, việc tính T[n] được thực hiện bằng vòng lặp:

```
S[0] := 0;
For i:=1 to n do
    S[i] := S[i - 1] + A[i];
```

Kết quả: Tổng các phần tử liên tiếp từ a_p đến a_q được tính theo công thức:

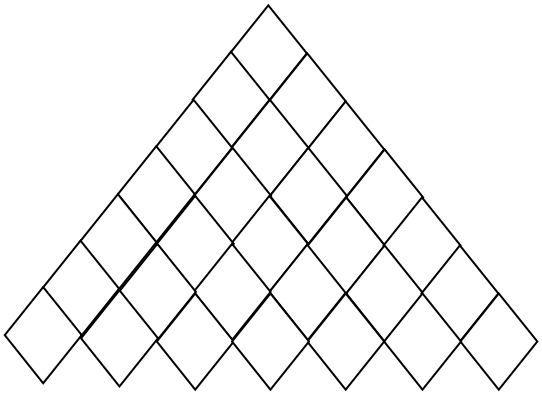
Sum := S[q] - S[p-1];

Code tham khảo:

```
// Độ phức tạp O(n+k)
const fi='sum.inp';
      fo='sum.out';
var   n,k: longint;
```

```
      i, p, q: longint;
      s: array[0..100001] of int64;
      a: array[1..100001] of integer;
BEGIN
   assign(input, fi); reset(input);
   assign(output, fo); rewrite(output);
   readln(n,k);
   s[0]:= 0;
   for i:= 1 to n do
      begin
          read(a[i]);
          s[i]:= s[i-1]+ a[i];
      end;
   for i:= 1 to k do
      begin
          readln(p, q);
          writeln(s[q] - s[p-1]);
      end;
   close(input); close(output);
END.
```

2.4. TRIANGLE PASCAL (Tam giác Pascal)



Tam giác Pascal là một mô hình dùng để đưa ra các hệ số của khai triển nhị thức Newton bậc N $(x+1)^N$.

Ví dụ: trong khai triển $(x+1)^2 = x^2 + 2x + 1$ có các hệ số là 1 2 1

Trong khai triển $(x+1)^3 = x^3 + 3x^2 + 3x + 1$ có các hệ số là 1 3 3 1

Yêu cầu: Hãy tìm các hệ số trong khai triển nhị thức Newton $(x + 1)^N$.

Dữ liệu: Vào file văn bản TRIANPAS.INP có cấu trúc như sau:

Dòng 1: Ghi số nguyên dương N ($1 \le N \le 100$).

***Kết quả*:** Ghi ra file văn bản TRIANNUM.OUT theo cấu trúc:

Dòng 1: Ghi ra các số nguyên dương lần lượt là các hệ số trong khai triển nhị thức Newton $(x + 1)^N$, các số được ghi cách nhau một dấu cách.

Ví dụ:

TRIANPAS.INP	TRIANPAS.OUT
5	1 5 10 10 5 1

Thuật toán:

+ Ta xây dựng mảng hai chiều có kích thước f[0..100, 0..101]

+ Ta nhận thấy f[0, 1] = 1 và f[i, 1] = f[i, i+1] = 1 . với mọi i = 1..n

+ Công thức quy hoạch động

$$f[i, j] = f[i-1, j-1] + f[i-1, j]$$

+ Kết quả được lưu trữ ở dòng N, cụ thể:

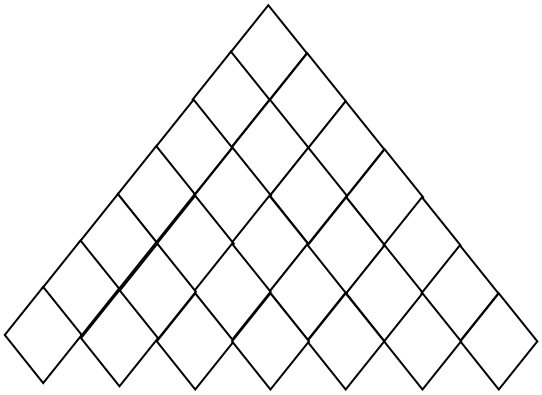
f[N, 1], f[N, 2], f[N, 3], ..., f[N, N], f[N,N+1]

Code tham khảo:

```
// Độ phức tạp O(n^2)
const fi='trianpas.inp';
      fo='trianpas.out';
var   i,j,n: longint;
      f: array[0..100, 0..101] of longint;

BEGIN
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    readln(n);
    for i:= 0 to n do f[i,1]:= 1;
    for i:=  1 to n do f[i,i+1]:= 1;
    for i:= 2 to n do
      for j:= 2 to i do f[i,j]:= f[i-1,j-1] + f[i-1,j];
    for j:= 1 to n+1 do write(f[n,j],' ');
    close(input); close(output);
END.
```

2.5. TRIANGLE NUMBER (Tam giác số)



Cho tam giác số như hình vẽ. Ta định nghĩa một đường đi trong tam giác số là đường đi xuất phát từ hình thoi ở đỉnh tam giác và đi xuống các hình thoi có chung cạnh với nó, đường đi kết thúc khi gặp một hình thoi ở đáy tam giác.

***Yêu cầu*:** Hãy tìm một đường đi trong tam giác số sao cho tổng giá trị của các ô trong đường đi có giá trị lớn nhất.

Dữ liệu: Vào từ file văn bản TRIANNUM.INP có cấu trúc như sau:

Dòng 1: Ghi số nguyên dương N là số hàng của tam giác ($1 \le N \le 100$).

Dòng thư i trong N dòng tiếp theo: Ghi *i* số nguyên dương lần lượt là giá trị của các ô trên dòng thứ i tương ứng trong tam giác (Các số có giá trị không quá 32000). Các số được ghi cách nhau một dấu cách.

***Kết quả*:** Ghi ra file văn bản TRIANNUM.OUT gồm 1 số nguyên dương S là tổng giá trị lớn nhất của đường đi tìm được.

Ví dụ:

TRIANNUM.INP	TRIANNUM.OUT
6 7 2 8 5 9 3 4 6 9 2 7 8 5 6 6 1 3 4 9 8 1	48

Thuật toán:

+ Ta xây dựng mảng hai chiều có kích thước f[1..100, 1..100]

+ Ta nhận thấy f[1, 1] = a[1, 1];

+ Công thức quy hoạch động

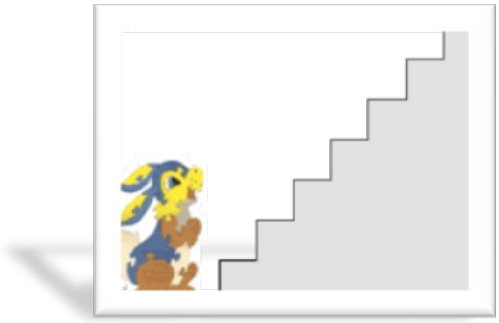
$$f[i, j] = Max(f[i-1, j-1] + f[i-1, j]) + a[i, j]$$

+ Kết quả là giá trị lớn nhất ở dòng thứ N: res = max(f[n,j]) $\forall j = 1..n$

Code tham khảo:

```
// Độ phức tạp O(n^2)
uses math;
const fi='triannum.inp';
      fo='triannum.out';
var   a: array[0..101, 0..101] of longint;
      i,j,n: longint;
      f: array[0..101, 0..101] of int64;
      res: int64;
BEGIN
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    readln(n);
    for i:= 1 to n do
      for j:= 1 to i do read(a[i, j]);
    f[1,1]:= a[1,1];
    for i:= 2 to n do
      begin
         f[i,1]:= f[i-1,1] + a[i,1];
         f[i,i]:= f[i-1,i-1] + a[i,i];
      end;
    for i:= 3 to n do
      for j:= 2 to i-1 do
         f[i,j]:= max(f[i-1,j-1], f[i-1,j]) + a[i, j];
    res:= f[n,1];
    for j:= 2 to n do
      if res < f[n,j] then res:= f[n,j];
    writeln(res);
    close(input); close(output);
END.
```

2.6. Bài toán Thỏ nhặt cà rốt



Các con thú nuôi trong chuồng ở vườn bách thú thường ít có điều kiện vận động. Điều này vừa có hại cho sức khỏe của thú nuôi, vừa làm làm giảm hứng thú của khách tham quan. Để khắc phục điều đó, Ban giám đốc cho đặt một cái thang có ***n*** bậc trong chuồng thỏ. Đến giờ cho ăn người ta đặt cà rốt - thứ khoái khẩu nhất của thỏ, lên bậc trên cùng của thang. Thỏ phải nhảy theo các bậc thang để lấy cà rốt. Mỗi bước nhảy thỏ có thể không được vượt quá ***k*** bậc ($1 \le k \le n \le 300$). Có thể có nhiều cách nhảy để lấy cà rốt. Hai cách nhảy gọi là khác nhau nếu tồn tại một bậc thỏ tới được ở một cách nhảy và bị bỏ qua ở cách kia. Ví dụ, với ***n*** = 4 và ***k*** = 3 có tất cả 7 cách lấy cà rốt khác nhau: 1+1+1+1, 1+1+2, 1+2+1, 2+1+1, 2+2, 1+3, 3+1.

Yêu cầu: Cho ***k*** và ***n***. Hãy xác định số cách khác nhau thỏ có thể thực hiện để lấy cà rốt.

Dữ liệu: Vào từ file văn bản CARROT.INP có cấu trúc như sau:

- *Dòng 1:* Ghi số nguyên dương *t* là số lượng cặp ***k*** và ***n*** ($1 \le t \le 50$).
- *Dòng thứ i trong t dòng tiếp theo:* Mỗi dòng ghi 2 số nguyên ***k*** và ***n***.

Kết quả: Ghi ra file văn bản CARROT.OUT, kết quả mỗi test đưa ra trên một dòng dưới dạng số nguyên.

Ví dụ:

CARROT.INP	CARROT.OUT
3 **1 3** **2 7** **3 10**	**1** **21** **274**

Thuật toán:

+ Gọi f[i] là số cách mà thỏ có thể nhảy tới bậc thứ i của thang.
+ Ta có bài toán cơ sở f[0] = 1;
+ Công thức f[i] = f[i-k] + f[i-k+1] + …+ f[i-1]
+ Kết quả bài toán là f[n]

Code tham khảo:

```
// Độ phức tạp O(n)
const fi='carrot.inp';
      fo='carrot.out';
var   t,k, n: longint;
      f: array[0..301] of int64;
      i: longint;
procedure   xuli;
```

```
var   i,j: longint;
begin
   f[0]:= 1;
   for i:= 1 to k do
      begin
          f[i]:=0;
          for j:= 0 to i-1 do f[i]:= f[i] + f[j];
      end;
   for i:= k+1 to n do
      begin
          f[i]:=0;
          for j:= i-k to i-1 do f[i]:= f[i] + f[j];
      end;
   writeln(f[n]);
end;
BEGIN
   assign(input, fi); reset(input);
   assign(output, fo); rewrite(output);
   readln(t);
   for i:= 1 to t do
      begin
         readln(k, n);
         xuli;
      end;
   close(input); close(output);
END.
```

2.7. Nguồn: http://vn.spoj.com/problems/LATGACH/

Cho một hình chữ nhật kích thước 2xN (1<=N<=100). Hãy đếm số cách lát các viên gạch nhỏ kích thước 1x2 và 2x1 vào hình trên sao cho không có phần nào của các viên gạch nhỏ thừa ra ngoài, cũng không có vùng diện tích nào của hình chữ nhật không được lát.

Dữ liệu: Vào từ file văn bản LATGACH.INP gồm nhiều test:

+ Dòng đầu ghi số lượng test T (T<=100).

+ T dòng sau mỗi dòng ghi một số N.

Kết quả: Ghi ra file LATGACH.OUT gồm T dòng là số cách lát tương ứng.

Ví dụ:

LATGACH.INP	LATGACH.OUT
3 1 2 3	1 2 3

Thuật toán:

Nhận thấy N = 1 thì có 1 cách xếp. N = 2 thì có 2 cách xếp. N = 3 thì có 3 các xếp. N = 4 thì có 5 cách xếp. Từ đó ta suy ra được quy luật của đáp án chính là dãy số fibonaci: f[i] = f[i-1] + f[i-2]. Nhưng do N <= 100 nên kết quả của bài toán có thể lên tới số có khoảng 21 chữ số. Vì vậy chúng ta phải xử lí số lớn.

Code tham khảo:

```
const fi='';//LATGACH.INP';
      fo='';//LATGACH.OUT';
var   t,maxn:byte;
      n:array[0..101] of byte;
      f:array[0..101] of string;
procedure   docfile;
   var i:byte;
   begin
      assign(input,fi);
      reset(input);
      readln(t);
      maxn:=0;
      for i:=1 to t do
         begin
            readln(n[i]);
            if n[i]>maxn then maxn:=n[i];
         end;
      assign(output,fo);
      rewrite(output);
   end;
procedure   dongfile;
   begin
      close(input);
      close(output);
   end;
function cong(a,b:string):string;
   var x,y,d,nho,s,i:byte;
       p:string;
   begin
      while length(a)<>length(b) do
         if length(a)<length(b) then a:='0'+a
         else  b:='0'+b;
      cong:='';   nho:=0;
      for i:=length(a) downto 1 do
         begin
            val(a[i],x);
            val(b[i],y);
            s:=x+y+nho;
            nho:=s div 10;
            s:=s mod 10;
            str(s,p);
            cong:=p+cong;
         end;
      if nho<>0 then
         begin
            str(nho,p);
            cong:=p+cong;
         end;
      while cong[1]='0' do delete(cong,1,1);
   end;
procedure  tinh;
   var i:byte;
   begin
      f[1]:='1';   f[0]:='1';
      for i:=2 to maxn do
         f[i]:=cong(f[i-1],f[i-2]);
   end;
procedure  xuli;
   var i:byte;
   begin
      tinh;
      for i:=1 to t do
```

```
        writeln(f[n[i]]);
   end;
BEGIN
   docfile;
   xuli;
   dongfile;
END.
```

2.8. Nguồn: http://vn.spoj.com/problems/QBMAX/

Cho một bảng A kích thước m x n (1 <= m, n <= 100), trên đó ghi các số nguyên a_{ij} ($|a_{ij}|$ <= 100). Một người xuất phát tại ô nào đó của cột 1, cần sang cột n (tại ô nào cũng được).

Quy tắc đi: Từ ô (i, j) chỉ được quyền sang một trong 3 ô (i, j + 1); (i - 1, j + 1); (i + 1, j + 1)

Dữ liệu: Vào từ file văn bản QBMAX.INP gồm:

+ Dòng đầu: Ghi hai số m, n là số hàng và số cột của bảng.

+ M dòng tiếp theo, dòng thứ i ghi đủ n số trên hàng i của bảng theo đúng thứ tự từ trái qua phải

Kết quả: Ghi ra file QBMAX.OUT gồm 1 dòng duy nhất ghi tổng lớn nhất tìm được

Ví dụ:

QBMAX.INP	QBMAX.OUT
5 7 9 -2 6 2 1 3 4 0 -1 6 7 1 3 3 8 -2 8 2 5 3 2 1 -1 6 2 1 6 1 7 -2 6 2 1 3 7	41

Thuật toán:

+ Gọi f[i,j] là tổng giá trị lớn nhất khi đi từ cột 1 đến ô (i,j).

+ Bài toán cơ sở f[1, n] = a[1, n]

+ Công thức quy hoạch động:

f[i, j] = max(f[i-1, j-1], f[i, j-1],f[i+1, j-1]) + a[i, j]

Chú ý khi lập trình để không phải xét đường biên trên và đường biên dưới thì ta nên khởi tạo f[0, i] = f[m+1, i] = -oo

+ Kết quả của bài toán là max(f[i, n]).

Code tham khảo:

```
uses  math;
const fi='';//QBMAX.INP';
      fo='';//QBMAX.OUT';
var   a:array[0..101,0..101] of integer;
```

```
      n,m,i,j:integer;
      f:array[0..101,0..101] of longint;
procedure   docfile;
   begin
      assign(input,fi);
      reset(input);
      readln(m,n);
      for i:=1 to m do
         begin
            for j:=1 to n do
               read(a[i,j]);
               readln;
         end;
      for i:=1 to n do
         begin
            f[0,i]:=-1000000;
            f[m+1,i]:=-1000000;
         end;
      assign(output,fo);
      rewrite(output);
   end;
procedure   dongfile;
   begin
      close(input);
      close(output);
   end;
procedure   xuli;
   begin
      for i:=1 to m do
         f[i,1]:=a[i,1];
      for j:=2 to n do
         for i:=1 to m do
            f[i,j]:=max(f[i-1,j-1],max(f[i,j-1],f[i+1,j-1]))+a[i,j];
   end;
procedure   ghifile;
   var   max:longint;
   begin
      max:=f[1,n];
      for i:=2 to m do
         if f[i,n]>max then max:=f[i,n];
      writeln(max);
   end;
BEGIN
   docfile;
   xuli;
   ghifile;
   dongfile;
END.
```

2.9. Nguồn: http://vn.spoj.com/problems/VSTEPS/

Bờm chơi trò chơi điện tử Lucky Luke đến màn phải điều khiển Lucky leo lên một cầu thang gồm n bậc.

Các bậc thang được đánh số từ 1 đến n từ dưới lên trên. Lucky có thể đi lên một bậc thang, hoặc nhảy một bước lên hai bậc thang. Tuy nhiên một số bậc thang đã bị thủng do cũ kỹ và Lucky không thể bước chân lên được. Biết ban đầu, Lucky đứng ở bậc thang số 1 (bậc thang số 1 không bao giờ bị thủng).

Chơi đến đây, Bờm chợt nảy ra câu hỏi: có bao nhiêu cách để Lucky leo hết được cầu thang? (nghĩa là leo đến bậc thang thứ n). Bờm muốn nhờ bạn trả lời câu hỏi này.

***Dữ liệu*:** Vào từ file văn bản VSTEPS.INP:

- Dòng đầu tiên: gồm 2 số nguyên n và k, là số bậc của cầu thang và số bậc thang bị hỏng ($0 \le k < n \le 100000$).
- Dòng thứ hai: gồm k số nguyên cho biết chỉ số của các bậc thang bị hỏng theo thứ tự tăng dần.

Kết qủa: Ghi ra file VSTEPS.OUT gồm phần dư của số cách Lucky leo hết cầu thang khi chia cho 14062008.

Ví dụ:

VSTEPS.INP	VSTEPS.OUT
4 2 2 3	0
90000 1 49000	4108266

Thuật toán:

+ Gọi f[i] là số cách nhảy từ bậc 1 đến bậc thứ i
+ Ta có bài toán cơ sở: f[1] = 1
+ Công thức quy hoạch động

  f[i] = f[i-1] + f[i-2] mod base; nếu bậc thang thứ i không bị hỏng

  f[i] = 0; nếu bậc thang thứ i bị hỏng
+ Kết quả của bài toán là f[n].

Code tham khảo:

```
const   fi='';//'vsteps.inpt';
        fo='';//'vsteps.out';
        base=14062008;
var     a:array[1..100000] of longint;
        f:array[0..100000] of longint;
        n,b,i,k:longint;
BEGIN
    assign(input,fi);reset(input);
    readln(n,k); fillchar(a,sizeof(a),0);
    for i:=1 to k do
      begin
        read(b);
        a[b]:=b;
      end;
    close(input);
    fillchar(f,sizeof(f),0);
    f[1]:=1;
    for i:=2 to n do
        if a[i]=0 then
```

```
            f[i]:=((f[i-2])+(f[i-1])) mod base;
    assign(output,fo);rewrite(output);
    writeln(f[n]);
    close(output);
END.
```

2.10. Nguồn: http://vn.spoj.com/problems/QBTICKET/

Tuyến đường sắt từ thành phố A đến thành phố B đi qua một số nhà ga. Tuyến đường có thể biểu diễn bởi một đoạn thẳng, các nhà ga là các điểm trên đó. Tuyến đường bắt đầu từ A và kết thúc ở B, vì thế các nhà ga sẽ được đánh số bắt đầu từ A (có số hiệu là 1) và B là nhà ga cuối cùng.

Giá vé đi lại giữa hai nhà ga chỉ phụ thuộc vào khoảng cách giữa chúng. Cách tính giá vé như sau:

Khoảng cách giữa hai nhà ga (X)

Khoảng cách 0 < X <= L1 -> Giá vé C1

Khoảng cách 0 < X <= L2 -> Giá vé C2

Khoảng cách 0 < X <= L3 -> Giá vé C3

Nghĩa là với các giá vé C1, C2, C3 tương ứng bạn sẽ đi quảng đường tối đa là L1, L2, L3.

Vé để đi thẳng từ nhà ga này đến nhà ga khác chỉ có thể đặt mua nếu khoảng cách giữa chúng không vượt quá L3. Vì thế nhiều khi để đi từ nhà ga này đến nhà ga khác ta phải đặt mua một số vé. Hơn thế nữa, nhân viên đường sắt yêu cầu hành khách chỉ được giữ đúng một vé khi đi trên tàu và vé đó sẽ bị huỷ khi hành khách xuống tàu.

Yêu cầu: Tìm cách đặt mua vé để đi lại giữa hai nhà ga cho trước với chi phí mua vé là nhỏ nhất

Dữ liệu: Vào từ file văn bản QBTICKET.INP gồm

+Dòng đầu tiên ghi các số nguyên L1, L2, L3, C1, C2, C3 ($1 <= L1 <= L2 <= L3 <= 10^9$; $1 <= C1 <= C2 <= C3 <= 10^9$) theo đúng thứ tự liệt kê ở trên.

+ Dòng thứ hai chứa số lượng nhà ga N (2 <= N <= 100000)

+ Dòng thứ ba ghi hai số nguyên s, f là các chỉ số của hai nhà ga cần tìm cách đặt mua vé với chi phí nhỏ nhất để đi lại giữa chúng.

+ Dòng thứ i trong số N - 1 dòng tiếp theo ghi số nguyên là khoảng cách từ nhà ga A (ga 1) đến nhà ga thứ i + 1.

Kết quả: Ghi ra file QBTICKET.OUT gồm 1 dòng duy nhất ghi chi phí nhỏ nhất tìm được

Ví dụ:

QBTICKET.INP	QBTICKET.OUT
3 6 8 20 30 40 7 2 6 3 7 8 13 15 23	70

Thuật toán:

+ Ta có w[i] là khoảng cách từ nhà ga 1 đến nhà ga i.

+ Gọi f[i] là chi phí nhỏ nhất để đi từ nhà ga s đến nhà ga thứ i

+ Bài toán cơ sở f[s] = 0

+ Công thức quy hoạch động

f[i] = min (f[j1] + c1, f[j2] + c2, f[j3] + c3); Trong đó j1 là nhà ga xa nhà ga i nhất mà thỏa mãn j1>=s và w[i] – w[j1] <= L1. j2 là nhà ga xa nhà ga i nhất mà thỏa mãn j2>=s và w[i] – w[j2] <= L2. j3 là nhà ga xa nhà ga i nhất mà thỏa mãn j3>=s và w[i] – w[j3] <= L3

+ Kết quả bài toán chính là f[t]. Trong đó t là nhà ga kết thúc

Code tham khảo:

```
uses  crt;
Const fi='';//'qbticket.inp';
      fo='';//'qbticket.out';
Var   c1,c2,c3,l1,l2,l3:int64;
      i,j,n,s,t:longint;
      f,w:array[1..100000] of int64;
procedure Enter;
 begin
      assign(input,fi);reset(input);
      readln(l1,l2,l3,c1,c2,c3);
      readln(n);
      readln(s,t);
      for i:=2 to n do readln(w[i]);
      close(input);
 end;
procedure Solve;
 begin
      f[s]:=0;
      for i:=s+1 to t do
        begin
            f[i]:=high(int64);
            j:=i-1;
            while (j>=s) and (w[i]-w[j]<=l1) do dec(j);
            if f[i]>f[j+1]+c1 then f[i]:=f[j+1]+c1;
            while (j>=s) and (w[i]-w[j]<=l2) do dec(j);
            if f[i]>f[j+1]+c2 then f[i]:=f[j+1]+c2;
```

```
            while (j>=s) and (w[i]-w[j]<=l3) do dec(j);
            if f[i]>f[j+1]+c3 then f[i]:=f[j+1]+c3;
         end;
 end;
procedure Solution;
 begin
      assign(output,fo);rewrite(output);
      writeln(f[t]);
      close(output);
 end;
BEGIN
      Enter;
      Solve;
      Solution;
END.
```

2.11. Nguồn: http://vn.spoj.com/problems/PTRANG/

Văn bản là một dãy gồm N từ đánh số từ 1 đến N. Từ thứ i có độ dài là wi (i=1, 2,... N). Phân trang là một cách xếp lần lượt các từ của văn bản vào dãy các dòng, mỗi dòng có độ dài L, sao cho tổng độ dài của các từ trên cùng một dòng không vượt quá L.Ta gọi hệ số phạt của mỗi dòng trong cách phân trang là hiệu số (L-S), trong đó S là tổng độ dài của các từ xếp trên dòng đó. Hệ số phạt của cách phân trang là giá trị lớn nhất trong số các hệ số phạt của các dòng. Tìm cách phân trang với hệ số phạt nhỏ nhất.

Dữ liệu: Vào từ file văn bản PTRANG.INP

- Dòng 1 chứa 2 số nguyên dương N, L (N<=6000,L<=1000)
- Dòng thứ i trong số N dòng tiếp theo chứa số nguyên dương wi (wi<=L), i=1, 2,.., N

Kết quả: Ghi ra file PTRANG.OUT là hệ số phạt nhỏ nhất

Ví dụ:

PTRANG.INP	PTRANG.OUT
4 5 3 2 2 4	2

Thuật toán:

+ Gọi w[i] là tổng độ dài của từ thứ 1 đến từ thứ i

+ gọi g[i] là hệ số phạt nhỏ nhất khi xét đến từ thứ i

+ Công thứ quy hoạch động

G[i] = min(max(g[j], L – (w[i] – w[j])) nếu w[i] – w[j] <=L. Với mọi j = i-1 .. 0

+ Kết quả của bài toán chính là g[n].

Code tham khảo:

```
uses  math;
const fi='ptrang.inp';
      fo='ptrang.out';
var   n,l,i,j:longint;
        w,g:array[0..6000] of longint;
procedure Solve;
 begin
      w[0]:=0;
      for i:=1 to n do w[i]:=w[i]+w[i-1];
      for i:=1 to n do
         begin
            g[i]:=maxlongint;
            for j:=i-1 downto 0 do
              begin
                  if w[i]-w[j]>l then break;
                  g[i]:=min(g[i],max(g[j],l-(w[i]-w[j])));
              end;
         end;
      write(g[n]);
 end;
BEGIN
  assign(input, fi); reset(input);
  assign(output, fo); rewrite(output);
  readln(n,l);
  for i:=1 to n do readln(w[i]);
  solve;
  close(input); close(output);
END.
```

2.12. Nguồn: http://vn.spoj.com/problems/LINEGAME/

Trò chơi với băng số là trò chơi tham gia trúng thưởng được mô tả như sau: Có một băng hình chữ nhật được chia ra làm *n* ô vuông, đánh số từ trái qua phải bắt đầu từ 1. Trên ô vuông thứ *i* người ta ghi một số nguyên dương a_i, $i = 1, 2, ..., n$. Ở một lượt chơi, người tham gia trò chơi được quyền lựa chọn một số lượng tùy ý các ô trên băng số. Giả sử theo thứ tự từ trái qua phải, người chơi lựa chọn các ô $i_1, i_2, ..., i_k$. Khi đó điểm số mà người chơi đạt được sẽ là: $a_{i1} - a_{i2} + ... + (-1)^{k-1}a_{ik}$

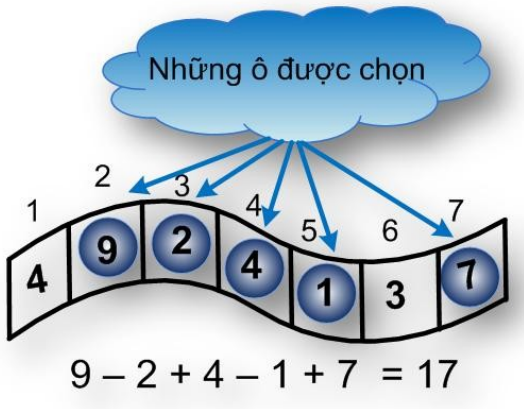
***Yêu cầu*:** Hãy tính số điểm lớn nhất có thể đạt được từ một lượt chơi.

Dữ liệu: Vào từ file văn bản LINEGAME.INP

- Dòng đầu tiên chứa số nguyên dương n ($n \leq 10^6$) là số lượng ô của băng số;
- Dòng thứ hai chứa n số nguyên dương $a_1, a_2, ..., a_n$ ($a_i \leq 10^4$, $i = 1, 2, ..., n$) ghi trên băng số. Các số liên tiếp trên cùng dòng được ghi cách nhau bởi ít nhất một dấu cách.

Kết quả: Ghi ra file LINEGAME.OUT một số nguyên duy nhất là số điểm lớn nhất có thể đạt được từ một lượt chơi.

Ví dụ:



LINEGAME.INP	LINEGAME.OUT
7 4 9 2 4 1 3	17

Thuật toán:

+ Gọi f[i] là tổng số điểm lớn nhất khi xét đến số thứ i và dấu cuối cùng là dấu cộng. Gọi g[i] là tổng số điểm lớn nhất khi xét đến số thứ i và dấu cuối cùng là dấu trừ.

+ Bài toán cơ sở f[0] = g[0] = 0;

+ Công thức quy hoạch động

f[i]:=max(g[i-1]+a[i],f[i-1]);

g[i]:=max(g[i-1],f[i-1]-a[i]);

+ Kết quả của bài toán là max(f[n], g[n]).

Code tham khảo:

```
Uses    math;
Const   fi='';//'linegame.inp';
        fo='';//'linegame.out';
Var     maxx,l,t,tg,k,m,n,i:longint;
        a,f,g:array[0..10000000]of int64;

BEGIN
        Assign(input,fi);Reset(input);
        Assign(output,fo);Rewrite(output);
        Read(n);
        g[0]:=0;f[0]:=0;
        For i:=1 to n do read(a[i]);
        For i:=1 to n do
                begin
                     f[i]:=max(g[i-1]+a[i],f[i-1]);
                     g[i]:=max(g[i-1],f[i-1]-a[i]);

                end;
        Write(max(g[n],f[n]));
        Close(output);
END.
```

2.13. Nguồn: http://vn.spoj.com/problems/QBSQUARE/

Cho một bảng kích thước MxN, được chia thành lưới ô vuông đơn vị M dòng N cột (1 <= M, N <= 1000)

Trên các ô của bảng ghi số 0 hoặc 1. Các dòng của bảng được đánh số 1, 2... M theo thứ tự từ trên xuống dưới và các cột của bảng được đánh số 1, 2..., N theo thứ tự từ trái qua phải

Yêu cầu: Hãy tìm một hình vuông gồm các ô của bảng thoả mãn các điều kiện sau:

1 - Hình vuông là đồng nhất: tức là các ô thuộc hình vuông đó phải ghi các số giống nhau (0 hoặc 1)

2 - Cạnh hình vuông song song với cạnh bảng.

3 - Kích thước hình vuông là lớn nhất có thể

Dữ liệu: Vào từ file QBSQUARE.INP

+ Dòng 1: Ghi hai số m, n

+ M dòng tiếp theo, dòng thứ i ghi N số mà số thứ j là số ghi trên ô (i, j) của bảng

Kết quả: Ghi ra file QBSQUARE.OUT gồm 1 dòng duy nhất ghi kích thước cạnh của hình vuông tìm được.

Ví dụ:

QBSQUARE.INP	QBSQUARE.OUT
11 13 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 1	7

Thuật toán:

+ Gọi f[i, j] là kích thước của hình vuông thỏa mãn bài toán mà có ô (i, j) là đỉnh phải dưới.

+ Bài toán cơ sở f[1, j] = f[i, 1] = 1.

+ Công thức quy hoạch động

f[i,j]:=min(f[i-1,j-1],f[i,j-1],f[i-1,j])+1 nếu 4 ô (i,j), (i-1,j-1), (i-1,j), (i,j-1) cùng số 0 hoặc cùng số 1.

Ngược lại f[i,j] = 1;

+ Kết quả của bài toán là max(f[i,j]). Với mọi i= 1..m, j= 1..n

Code tham khảo:

```
uses math;
const   fi='QBSQUARE.inp';
        fo='QBSQUARE.out';
var     a:array[1..1000,1..1000] of 0..1;
        getmax,m,n,i,j:integer;
        f:array[1..1000,1..1000] of integer;

procedure enter;
    begin
        assign(input,fi);reset(input);
        readln(m,n);
        for i:= 1 to m do
            for j:=1 to n do read(a[i,j]);
        close(input);
    end;
procedure   xuly;
    BEGIN
        for i:= 1 to n do f[1,i]:=1;
        getmax:=1;
        for i:=1 to m do f[i,1]:=1;
        for i:=2 to m do
            for j:= 2 to n do
              begin
                f[i,j]:=1;
                if ((a[i,j-1]=1) and (a[i-1,j-1]=1) and (a[i-1,j]=1)and
(a[i,j]=1))
                or ((a[i,j-1]=0) and (a[i-1,j-1]=0) and (a[i-1,j]=0)and
(a[i,j]=0))
                then f[i,j]:=min(f[i-1,j-1],min(f[i,j-1],f[i-1,j]))+1;
                if f[i,j]>getmax then getmax:=f[i,j];
              end;
    end;
procedure   solution;
    begin
         assign(output,fo);rewrite(output);
         writeln(getmax);
         close(output);
    end;
BEGIN
     enter;
     xuly;
     solution;
END.
```

2.14. Nguồn: http://vn.spoj.com/problems/BONUS/

Tuấn là người chiến thắng trong một cuộc thi "tìm hiểu kiến thức vũ trụ" và được nhận các phần thưởng do công ty XYZ tài trợ. Các phần thưởng được bố trí trên một bảng hình vuông nxn có dạng một lưới ô vuông kích thước đơn vị. Các dòng của bảng được đánh số từ 1 đến n, từ trên xuống dưới và các cột của bảng được đánh số từ 1 đến n, từ trái qua phải. Ô nằm trên giao của dòng i và cột j được gọi là ô (i,j) và trên ô đó chứa một món quà có giá trị là a[i,j] (1 <= i, j <= n)

Để nhận phần thưởng, Tuấn được phép chọn một hình vuông kích thước k x k chiếm trọn trong một số ô của bảng và nhận tất cả các phần quà có trong các ô nằm trong hình vuông đó.

Yêu cầu: Hãy xác định tổng giá trị lớn nhất của món quà mà Tuấn có thể nhận được.

1	9	1	1
9	**9**	**9**	**9**
1	**9**	**9**	**9**
1	**9**	**9**	**14**

Dữ liệu: Vào từ file BONUS.INP gồm:

- Dòng thứ nhất chứa hai số nguyên dương n, k (n <= 1000, n/3 <= k <= n).
- Dòng thứ i trong số n dòng tiếp theo chứa n số nguyên dương, số thứ j là a[i,j] (a[i,j] <= 1000)

***Kết quả*:** Ghi ra một số nguyên duy nhất là tổng giá trị lớn nhất của các món quà mà Tuấn có thể nhận được.

Ví dụ:

BONUS.INP	BONUS.OUT
4 3 1 9 1 1 9 9 9 9 1 9 9 9 1 9 9 14	86

Thuật toán:

+ Gọi L[i,j] là tổng các số từ ô (1,1) đến ô (i,j). F[i, j] là tổng các số trên hàng từ ô (i,1) đến ô (i,j).

+ Công thức quy hoạch động

F[i,j] = F[i, j-1] + a[i,j];

L[i,j] = L[i-1,j] + F[i,j];

+ Kết quả của bài toán là max(L[i,j]-L[i-k,j]-L[i,j-k]+L[i-k,j-k]);

Code tham khảo:

```
const   fi='';//'bonus.inp';
        fo='';//'bonus.out';
        nmax=1000;
        nmin=nmax div 3;
var     a:array[-nmax+nmin..nmax,-nmax+nmin..nmax] of integer;
        l,f:array[-nmax+nmin..nmax,-nmax+nmin..nmax] of longint;
        l1,m,i,j,n,k:integer;
        tong,kq:longint;
BEGIN
    assign(input,fi);reset(input);
    assign(output,fo);rewrite(output);
    readln(n,k);
    for i:=1 to n do
```

```
        for j:=1 to n do
                read(a[i,j]);
    close(input);
    kq:=a[1,1];
    fillchar(l,sizeof(l),0);
    for i:=1 to n do
        for j:=1 to n do
            f[i,j]:=f[i,j-1]+a[i,j];
    for i:=1 to n do
        for j:=1 to n do
            l[i,j]:=l[i-1,j]+f[i,j];
    for i:=1 to n do
        for j:=1 to n do
            if l[i,j]-l[i-k,j]-l[i,j-k]+l[i-k,j-k]>kq then kq:=l[i,j]-l[i-
k,j]-l[i,j-k]+l[i-k,j-k];
    writeln(kq);
    close(output);
END.
```

2.15. Nguồn: http://vn.spoj.com/problems/PARIGAME/

Trò chơi chẵn lẻ là trò chơi hai đối thủ được mô tả như sau: Xuất phát từ bảng trò chơi là một bảng vuông kích thước n x n gồm n dòng và n cột. Các dòng của bảng được đánh số từ 1 đến n, từ trên xuống dưới. Các cột của bảng được đánh số từ 1 đến n, từ trái sang phải. Trên mỗi ô của bảng ghi một số nguyên. Hai đối thủ luân phiên thực hiện nước đi. Đối thủ đến lượt chơi của mình được phép xóa dòng cuối cùng nếu tổng các số trên dòng đó là số chẵn hoặc là cột cuối cùng nếu tổng các số trên cột đó là số chẵn.

Đối thủ thắng cuộc là người xóa được ô cuối cùng của bảng hoặc sau khi thực hiện nước đi của mình thì tổng các số trên dòng cuối cùng và tổng các số trên cột cuối cùng của bảng đều là số lẻ.

Yêu cầu: Cho biết bảng số của trò chơi, hãy xác định xem người đi trước có cách chơi giành phần thắng hay không?

Dữ liệu: Vào từ file PARIGAME.INP

- Dòng thứ nhất chứa số nguyên dương k là số lượng bộ dữ liệu.
- Tiếp theo là k nhóm dòng, mỗi nhóm dòng tương ứng với một bộ dữ liệu có dạng:

o Dòng thứ nhất chứa số nguyên dương n (n <= 500).

o Dòng thứ i trong số n dòng tiếp theo chứa n số nguyên dương (mỗi số không vượt quá 10^9) là các số trên dòng thứ i của bảng trò chơi, i = 1, 2, …, n.

Các số trên cùng một dòng được ghi cách nhau ít nhất một dấu cách.

Kết quả: Ghi ra file PARIGAME.OUT gồm k dòng, mỗi dòng là kết quả tương ứng với một bộ dữ liệu theo thứ tự xuất hiện trong file dữ liệu vào: ghi thông báo **"YES"** nếu người đi trước có cách chơi giành chiến thắng và **"NO"** trong trường hợp ngược lại.

Ví dụ:

PARIGAME.INP	PARIGAME.OUT
2	YES
3	NO
1 2 2	
1 2 3	
2 3 1	
4	
2 2 2 2	
2 2 2 2	
2 2 2 2	
2 2 2 2	

Thuật toán:

+ Gọi tonghang[i,j] là tổng các số trên hàng i từ ô (i,1) đến ô (i,j). tongcot[i,j] là tổng các số trên cột j từ ô (1,j) đến ô (i,j)

+ Gọi F[i,j] = true là thắng cuộc nếu khi xét bảng từ ô (1,1) đến ô (i,j). Ngược lại F[i,j] = false là thua cuộc.

+ Khởi tạo tất cả F[i,j] = false

+ Công thức tính F[i,j] = true nếu (F[i-1,j] = false và tonghang[i,j] là số chẵn) hoặc (F[i,j-1] = false và tongcot[i,j] là số chẵn)

+ Kết quả bài toán: Nêu F[n,n] = true thì in ra YES. Ngược lại in ra NO

Code tham khảo:

```
const   fi='';//'PARAGAME.inp';
        fo='';//'PARAGAME.out';
        nmax=500;
var     n,k,i,j:integer;
        a:array[1..nmax,1..nmax] of longint;
        f:array[0..nmax,0..nmax] of boolean;
        tonghang,tongcot:array[0..nmax,0..nmax] of int64;
procedure   solve;
    begin
        fillchar(f,sizeof(f),false);
        for i:=1 to n do
            for j:=1 to n do
                if ((f[i-1,j]=false) and (tonghang[i,j] mod 2=0))
                or ((f[i,j-1]=false) and (tongcot[i,j] mod 2 =0)) then
                f[i,j]:=true;
        if f[n,n] then writeln('YES')
        else writeln('NO');
    end;
procedure   enter;
    var ii,j:integer;
    begin
        assign(input,fi);reset(input);
        assign(output,fo);rewrite(output);
        readln(k);
        for ii:=1 to k do
            begin
                readln(n);
```

```
                for i:=1 to n do
                    for j:=1 to n do read(a[i,j]);
                fillchar(tonghang,sizeof(tonghang),0);
                fillchar(tongcot,sizeof(tongcot),0);
                for j:=1 to n do
                    for i:=1 to n do
                        tonghang[i,j]:=tonghang[i,j-1]+a[i,j];
                for i:=1 to n do
                    for j:=1 to n do
                        tongcot[i,j]:=tongcot[i-1,j]+a[i,j];
                solve;
            end;
        close(input);
        close(output);
    end;
BEGIN
    enter;
END.
```

2.16. Nguồn: http://vn.spoj.com/problems/QBSTR/

Xâu ký tự X được gọi là xâu con của xâu ký tự Y nếu ta có thể xoá đi một số ký tự trong xâu Y để được xâu X.

Cho biết hai xâu ký tự A và B, hãy tìm xâu ký tự C có độ dài lớn nhất và là con của cả A và B.

Dữ liệu: Vào từ file QBSTR.INP gồm

+ Dòng 1: chứa xâu A (độ dài xâu <= 10^3)

+ Dòng 2: chứa xâu B (độ dài xâu <= 10^3)

***Kết quả*: Ghi ra file QBSTR.OUT gồm một dòng ghi độ dài xâu C tìm được**

Ví dụ:

QBSTR.INP	QBSTR.OUT
abc1def2ghi3 abcdefghi123	10

Thuật toán:

+ Gọi độ dài của xâu A là n, đôi dài của xâu B là m.

+ Gọi F[i,j] là độ dài xâu con chung dài nhất khi xét xâu A từ vị trí 1 đến vị trí i, xâu B từ vị trí 1 đến vị trí j.

+ Bài toán cơ sở F[i,0] = F[0,j] = 0;

+ Công thức quy hoạch động:

F[i,j] = F[i-1,j-1] + 1 nếu A[i] = B[j]

F[i,j] = max(F[i-1,j], F[i,j-1]) nếu ngược lại

+ Kết quả bài toán là F[n,m]

Code tham khảo:

```
const fi='';//QBSTR.INP';
      fo='';//QBSTR.OUT';
var   n,m:longint;
      a,b:ansistring;
```

```
      f:array[0..1000,0..1000] of integer;
procedure   docfile;
   begin
       assign(input,fi);
       reset(input);
       readln(A);
       n:=length(A);
       readln(B);
       m:=length(B);
       assign(output,fo);
       rewrite(output);
   end;
procedure   dongfile;
   begin
      close(input);
      close(output);
   end;
function max(x,y:longint):longint;
   begin
      if x>y then exit(x) else exit(y);
   end;
procedure   xuli;
   var   i,j:longint;
   begin
      for i:=1 to n do
         f[i,0]:=0;
      for j:=1 to m do
         f[0,j]:=0;
      for i:=1 to n do
         for j:=1 to m do
            if a[i]=b[j] then f[i,j]:=f[i-1,j-1]+1
            else
               f[i,j]:=max(f[i-1,j],f[i,j-1]);
   end;
procedure   ghifile;
   begin
      writeln(f[n,m]);
   end;
BEGIN
   docfile;
   xuli;
   ghifile;
   dongfile;
END.
```

2.17. Palindrome

Cho một xâu S chỉ gồm các kí tự latin in thường. Hãy tìm số kí tự ít nhất cần thêm vào để S trở thành xâu đối xứng

Dữ liệu: Vào từ tệp palindro.inp: gồm 1 dòng duy nhất là xâu S có độ dài không quá 10^4

Kết quả: Ghi ra tệp palindro.out: gồm 1 số duy nhất là kết quả của bài toán.

Ví dụ:

Palindro.inp	Palindro.out
adca	1

Thuật toán:

+ Gọi f[i, j] là số kí tự ít nhất cần thêm vào để xâu S[i..j] trở thành xâu đối xứng.

+ Bài toán cơ sở: f[i, i] = 0;

+ Xây dựng công thức:

- Nếu S[i] = S[j] thì f[i, j] = f[i+1, j-1];
- Nếu S[i] <> S[j] thì f[i, j] = min(f[i, j-1], f[i+1,j]) + 1;

+ Kết quả bài toán là f[1, n]. Trong đó n là độ dài của xâu

Code tham khảo:

```
uses  math;
const fi='palindro.inp';
      fo='palindro.out';
var   st: ansistring;
      i,j,k,n: longint;
      f: array[0..10000, 0..10000] of longint;
BEGIN
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    readln(st);
    n:= length(st);
    for i:= 1 to n do f[i,i]:= 0;
    for k:= 1 to n-1 do
      for i:= 1 to n-k do
         begin
             j:= i + k;
             if st[i] = st[j] then f[i,j]:= f[i+1, j-1]
             else f[i,j]:= min(f[i,j-1],f[i+1,j]) +1;
         end;
    writeln(f[1,n]);
    close(input); close(output);
END.
```

2.18. Nguồn: http://vn.spoj.com/problems/MBLAST/

Cho hai xâu A, B. Mở rộng của 1 xâu X là xâu thu được bằng cách chèn (0,1,2 ..) kí tự trống vào xâu. Ví dụ : X là ‘abcbcd’, thì 'abcb-cd', '-a-bcbcd-' và 'abcb-cd-' là các mở rộng của X. (Dấu cách kí hiệu bằng ‘-‘).

A1,B1 là mở rộng của A và B, và giả sử chúng cùng độ dài. Khoảng cách giữa A1 và B1 là tổng khoảng cách giữa các kí tự cùng vị trí. Nếu hai kí tự không là dấu cách thì khoảng cách giữa 2 kí tự này là trị tuyệt đối mã ASCII của chúng. Còn ngược lại, khoảng cách là 1 số K cố định.

Yêu cầu: Cho hai xâu A, B. Tìm khoảng cách nhỏ nhất giữa hai xâu mở rộng của nó.

Dữ liệu: Vào từ file MBLAST.INP gồm

+ Dòng đầu chứa A, dòng 2 chứa B, chỉ gồm chữ thường a-z và số kí tự <=2000.

+ Dòng thứ 3 là số K, khoảng cách của 1 kí tự bất kỳ với kí tự trống, $1 \leq K \leq 100$.

Kết quả: Ghi ra fiel MBLAST.OUT khoảng cách nhỏ nhất.

Ví dụ:

MBLAST.INP	MBLAST.OUT
cmc	10

snmn 2	
koiv ua 1	5
mj jao 4	12

Thuật toán:

+ Gọi m là độ dài xâu A, n là độ dài xâu B

+ Gọi F[i,j] là khoảng cách nhỏ nhất giữa hai xâu mở rộng mà khi xét xâu A đến vị trí thứ i và xâu B đến vị trí thứ j.

+ Nhận thấy F[0,0] = 0; F[i,0] = i*k; F[0,j] = j*k;

+ Công thức quy hoạch động

F[i,j]:=min(F[i,j-1] + k, F[i-1,j] + k, F[i-1,j-1] + 2*k, F[i-1,j-1] + abs(ord(a[i])-ord(b[j]);

+ Kết quả của bài toán là F[m,n]

Code tham khảo:

```
uses math;
const   fi='';//'mblast.inp';
        fo='';//'mblast.out';
var   a,b:ansistring;
      k,m,n,i,j:integer;
      f:array[0..2000,0..20000] of integer;
procedure   enter;
    begin
        assign(input,fi);reset(input);
        readln(a);
        readln(b);
        read(k);
        close(input);
    end;
procedure   solve;
    begin
        m:=length(a);
        n:=length(b);
        for i:= 1 to m do f[i,0]:=i*k;
        for j:=1 to n do f[0,j]:=j*k;
        f[0,0]:=0;
        for i:=1 to m do
            for j:=1 to n do
                f[i,j]:=min(f[i,j-1]+k,min(f[i-1,j]+k,min(f[i-1,j-
1]+2*k,f[i-1,j-1]+abs(ord(a[i])-ord(b[j])))));
    end;
BEGIN
    assign(output,fo);
    rewrite(output);
    enter;
```

```
    solve;
    writeln(f[m,n]);
    close(output);
END.
```

2.19. Nguồn: http://vn.spoj.com/problems/NKPALIN/

Một chuỗi được gọi là đối xứng (palindrome) nếu như khi đọc chuỗi này từ phải sang trái cũng thu được chuỗi ban đầu.

Yêu cầu: tìm một chuỗi con đối xứng dài nhất của một chuỗi s cho trước. Chuỗi con là chuỗi thu được khi xóa đi một số ký tự từ chuỗi ban đầu.

Dữ liệu: Vào từ file NKPALIN.INP gồm một dòng duy nhất chứa chuỗi s, chỉ gồm những chữ cái in thường. Chuỗi s có độ dài không vượt quá 2000.

Kết qủa: Ghi ra file NKPALIN.OUT gồm một dòng duy nhất là một xâu con đối xứng dài nhất của xâu s. Nếu có nhiều kết quả, chỉ cần in ra một kết quả bất kỳ.

Ví dụ:

NKPALIN.INP	NKPALIN.OUT
lmevxeyzl	level

Thuật toán:

+ Nhập vào xâu A. Gọi B là xâu lật ngược của xâu A. Lúc này bài toán quy về bài tìm xâu con chung dài nhất của 2 xâu (giống mã bài QBSTR) nhưng có thêm phần truy vết ngược ra kết quả.

Code tham khảo:

```
const fi='';//NKPALIN.inp';
      fo='';//NKPALIN.out';
var   a,b:ansistring;
      p:array[1..2000] of char;
      f:array[0..2000,0..2000] of integer;
      m,n:integer;
procedure   docfile;
   begin
      assign(input,fi);
      reset(input);
      assign(output,fo);
      rewrite(output);
   end;
procedure   dongfile;
   begin
      close(input);
      close(output);
   end;
function max(x,y:integer):integer;
   begin
      if x>y then max:=x else max:=y;
   end;
procedure   thuchien;
   var  i,j:integer;
   begin
      for i:=0 to n do
         f[i,0]:=0;
      for j:=1 to n do
         f[0,j]:=0;
```

```
      for i:=1 to n do
         for j:=1 to n do
            if a[i]=b[j] then f[i,j]:=f[i-1,j-1]+1
            else  f[i,j]:=max(f[i-1,j],f[i,j-1]);
   end;
procedure   trace;
   var   x,y,k,dem:integer;
   begin
      x:=n; y:=n;
      dem:=0;
      while (x>0) and (y>0) do
         begin
            if a[x]=b[y] then
               begin
                  inc(dem);
                  p[dem]:=a[x];
                  dec(x);
                  dec(y);
               end
            else
               if f[x,y]=f[x-1,y] then dec(x)
               else dec(y);
         end;
      for k:=dem downto 1 do
          write(p[k]);
   end;
procedure   xuli;
   var i:integer;
   begin
      readln(a);
      n:=length(a);
      b:='';
      for i:=n downto 1 do b:=b+a[i];
      thuchien;
      trace;
   end;
BEGIN
   docfile;
   xuli;
   dongfile;
END.
```

2.20. Xâu đẹp. Nguồn: Lê Xuân Mạnh – chuyên Tin

Ta định nghĩa một xâu kí tự X có độ dài K được gọi là xâu đẹp nếu như nó thõa mãn các điều kiện sau:

- K chẵn (xâu X có độ dài chẵn).
- X[1] = X[K].
- X[2] ≠ X[K-1]
- .....

(Tổng quát: $\forall i \le \frac{K}{2}$ thì nếu i lẻ X[i]=X[k-i+1], nếu i chẵn X[i] ≠ X[k-i+1]).

Yêu cầu: Cho một xâu độ dài N chỉ gồm các kí tự là chữ cái latin thường. Hãy tìm xâu con (không cần liên tiếp) dài nhất của xâu đã cho sao cho nó là một xâu đẹp.

Dữ liệu vào*:* File văn bản STR.INP gồm:

- Dòng 1: Số nguyên dương N <=1000 độ dài của xâu đã cho.
- Dòng 2: Chứa một xâu gồm N kí tự chỉ gồm các chữ cái latin in thường.

Dữ liệu ra: Ghi vào file STR.OUT một số duy nhất là độ dài của xâu tìm được.

Ví dụ:

STR.INP	STR.OUT
7 Windows	4

Thuật toán:

+ Gọi f[i,j,1] là độ dài dài xâu đẹp thỏa mãn theo đề bài khi xét xâu từ ký tự vị trí i đến ký tự vị trí j. f[i,j,0] là độ dài có tính chất như xâu đẹp nhưng có hai ký tự hai đầu là khác nhau.

+ Ta nhận thấy f[i,i+1,1] = 2 nếu st[i] = st[i+1]. Ngược lại f[i,i+1,0]=2

+ Công thức quy hoạch động

- Nếu st[i] = st[j] thì f[i,j,0]:=max(f[i+1,j,0],f[i,j-1,0]);
  f[i,j,1]:=max(f[i+1,j,1],f[i,j-1,1],f[i+1,j-1,0]+2);
- Nếu st[i] <> st[j] thì f[i,j,1]:=max(f[i+1,j,1],f[i,j-1,1])
  f[i,j,0]:=max(f[i+1,j,0],f[i,j-1,0],f[i+1,j-1,1]+2);

+ Kết quả bài toán là f[1,n,1]

Code tham khảo:

```
{$H+}
uses math;
const   fi='STR.INP';
        fo='STR.OUT';
var     n:longint;
        st:string;
        f:array[0..1001,0..1001,0..1] of longint;
procedure   docfile;
    begin
        assign(Input,fi);reset(input);
        readln(n);  readln(st);
        assign(output,fo);rewrite(output);
    end;
procedure   dongfile;
    begin
        close(input);   close(output);
    end;
procedure   xuli;
    var i,j,ii:longint;
    begin
        for i:=1 to n-1 do
            if st[i]=st[i+1] then
                f[i,i+1,1]:=2
            else
                f[i,i+1,0]:=2;
        for ii:=2 to n-1 do
            for i:=1 to n-ii do
                begin
                    j:=i+ii;
                    f[i,j,0]:=max(f[i+1,j,0],f[i,j-1,0]);
                    f[i,j,1]:=max(f[i+1,j,1],f[i,j-1,1]);
                    if st[i]<>st[j] then
                        f[i,j,0]:=max(f[i,j,0],f[i+1,j-1,1]+2)
                    else
                        f[i,j,1]:=max(f[i,j,1],f[i+1,j-1,0]+2);
                end;
        writeln(f[1,n,1]);
    end;
BEGIN
    docfile;
```

```
    xuli;
    dongfile;
END.
```

2.21. Nguồn: http://vn.spoj.com/problems/LIQ/

Cho một dãy số nguyên gồm N phần tử A[1], A[2], ... A[N]. Biết rằng dãy con tăng đơn điệu là 1 dãy $A[i_1]$,... $A[i_k]$ thỏa mãn $i_1 < i_2 < ... < i_k$ và $A[i_1] < A[i_2] < .. < A[i_k]$. Hãy cho biết dãy con tăng đơn điệu dài nhất của dãy này có bao nhiêu phần tử?

Dữ liệu: Vào từ file văn bản LIQ.INP gồm

- Dòng 1 gồm 1 số nguyên là số N ($1 \le N \le 1000$).
- Dòng thứ 2 ghi N số nguyên A[1], A[2], .. A[N] ($1 \le A[i] \le 10000$).

Kết quả: Ghi ra file LIQ.OUT - độ dài của dãy con tăng đơn điệu dài nhất.

Ví dụ:

LIQ.INP	LIQ.OUT
6 1 2 5 4 6 2	4

Thuật toán:

+ Gọi f[i] là độ dài dãy con đơn điệu tăng mà có phần tử a[i] là phần tử cuối cùng

+ Nhận thấy f[1] = 1;

+ Công thức quy hoạch động

f[i] = max(f[j]) + 1 nếu a[j]<a[i] Với mọi j =1..i-1

Ngược lại f[i] = 1;

+ Kết quả bài toán là max(f[i])

+ Trong lúc cài đặt nếu cần đưa ra dãy con đó thì chúng ta sử dụng phần truy vết như code.

Code tham khảo:

```
const   fi='LIQ.INP';
        fo='LIQ.OUT';
var     a,f,tr:array[1..100] of integer;
        jmax,max,i,j,n,ii:byte;

procedure   enter;
    begin
        assign(input,fi);reset(input);
        readln(n);
        for i:=1 to n do read(a[i]);
        close(input);
    end;

procedure   solve;
    begin
        ii:=1;
        for i:=1 to n do
            begin
```

```
                f[i]:=1; jmax:=i;
                for j:=1 to i-1 do
                    if (a[j]<a[i]) and (f[j]+1>f[i])then
                        begin
                            f[i]:=f[j]+1;
                            jmax:=j;
                            if max<f[i] then
                                begin
                                    max:=f[i];
                                    ii:=i;
                                end;
                        end;
                tr[i]:=jmax;
            end;
        end;

procedure   solution;
    begin
        assign(output,fo);rewrite(output);
        writeln(max);
        { Truy vet ra day con neu can
        i:=ii; f[i]:=0;
        while i<>tr[i]do
            begin
                i:=tr[i];
                f[i]:=0;
            end;
        for i:=1 to n do
            if f[i]=0 then write(a[i],' ');}
        close(output);

    end;

BEGIN
    enter;
    solve;
    solution;
END.
```

2.22. Ổ GÀ. Nguồn: Đề thi chọn đội tuyển Tin học Thanh Hóa năm 2013

Sau một thời gian dài, con đường XYZ đã có dấu hiệu xuống cấp. Nhận được nhiều lời phàn nàn của người tham gia giao thông, nhà quản lý tiến hành kiểm tra chất lượng con đường trước khi đưa ra quyết định về việc xây đường mới.

Con đường có độ dài n mét và được chia thành n phần bằng nhau liên tiếp (mỗi phần dài 1 mét), đánh số từ 1 đến n. Người ta đã tính toán độ cao trung bình của mỗi phần. Cụ thể, phần thứ i sẽ có độ cao trung bình là A_i.

Kích thước của ổ gà lớn nhất là một trong những yếu tố quan trọng ảnh hưởng đến chất lượng của con đường. Một ổ gà là một đoạn liên tiếp các phần đường từ u đến v thỏa mãn các tính chất sau:

- $v \geq 2+u$ (đoạn đường dài ít nhất 3 mét).
- Tồn tại m ($u<m<v$) sao cho $A_i > A_{i+1}$ với mọi $u \leq i < m$ và $A_i < A_{i+1}$ với mọi $m \leq i < v$

- Giá trị v-u+1 được gọi là kích thước của ổ gà.

Ví dụ: Trên con đường có độ dài 5 mét với độ cao trung bình của mỗi phần lần lượt là (1, **3, 2, 4, 5**), đoạn đường từ phần đường thứ 2 đến phần đường thứ 5 là một ổ gà có kích thước bằng 4.

Yêu cầu: Tìm kích thước của ổ gà lớn nhất.

Dữ liệu: Vào từ file văn bản OGA.INP gồm:

- Dòng đầu: Ghi số nguyên n ($1 \le n \le {}^{1}05$).
- n dòng tiếp theo: Mỗi dòng ghi một số nguyên A_i ($0 \le A_i \le 10^9$).

Kết quả: Ghi ra file văn bản OGA.OUT một số nguyên duy nhất là kích thước lớn nhất của một ổ gà trên đường. Nếu không tìm được ổ gà nào, ghi ra 0.

Ví dụ:

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

Thuật toán:

+ Bài toán quy về tìm dãy con tăng dài nhất của dãy số

+ Gọi b[i] là dãy con giảm khi xét từ phần tử thứ nhất đến phần tử thứ i mà có phần tử cuối cùng là a[i]. Tương tự gọi c[i] là dãy con giảm khi xét từ phần tử thứ n về phần tử thứ i.

+ Ta có b[1] = 1; b[n] = 1;

+ Công thức tính b[i] và c[i] tương tự như bài dãy con tăng LIQ

+ Kết quả bài toán là max(b[i] + c[i] – 1). Rồi biện luận để in ra kết quả.

Code tham khảo:

```
const fi='OGA.inp';
      fo='OGA.out';
var a,b,c:array[0..100001] of longint;
    i,j,n,max:longint;
BEGIN
   assign(input,fi); reset(input);
   readln(n);
   for i:=1 to n do readln(a[i]);
   close(input);
   assign(output,fo); rewrite(output);
   for i:=1 to n do
      begin
         b[i]:=1;
         c[i]:=1;
      end;
   for i:=2 to n do
```

```
         for j:=i-1 downto 1 do
            if (a[i] < a[j]) and (b[j]+1>b[i]) then b[i]:=b[j]+1;
   for i:=n-1 downto 1 do
      for j:=i+1 to n do
         if (a[i] < a[j]) and (c[j]+1 > c[i]) then c[i]:=c[j]+1;
   max:=0;
   for i:=1 to n do
      if b[i]+c[i]-1>max then max:=b[i]+c[i]-1;
      if (max=1) or (max=2) then max:=0;
   write(max);
   close(output);
END.
```

2.23. Nguồn: http://vn.spoj.com/problems/NKTICK/

Có N người xếp hàng mua vé dự buổi hoà nhạc. Ta đánh số họ từ 1 đến N theo thứ tự đứng trong hàng. Mỗi người cần mua một vé, song người bán vé được phép bán cho mỗi người tối đa hai vé. Vì thế, một số người có thể rời hàng và nhờ người đứng trước mình mua hộ vé. Biết t_i là thời gian cần thiết để người i mua xong vé cho mình. Nếu người i+1 rời khỏi hàng và nhờ người i mua hộ vé thì thời gian để người thứ i mua được vé cho cả hai người là r_i.

Yêu cầu: Xác định xem những người nào cần rời khỏi hàng và nhờ người đứng trước mua hộ vé để tổng thời gian phục vụ bán vé là nhỏ nhất.

Dữ liệu: Vào từ file văn bản NKTICK.INP gồm:

- Dòng đầu tiên chứa số N ($1 \le N \le 60000$).
- Dòng thứ 2 ghi N số nguyên dương $t_1, t_2, ..., t_N$. ($1 \le t_i \le 30000$)
- Dòng thứ ba ghi N-1 số nguyên dương $r_1, r_2, ..., r_{N-1}$. ($1 \le r_i \le 30000$)

Kết qủa: Ghi ra file NKTICK.OUT là tổng thời gian phục vụ nhỏ nhất.

Ví dụ:

NKTICK.INP	NKTICK.OUT
5 2 5 7 8 4 4 9 10 10	18
4 5 7 8 4 50 50 50	24

Thuật toán:

+ Gọi F[i] là tổng thời gian mua vé khi xét đến người thứ i.
+ Bài toán cơ sở: F[0] = 0; F[1]= t[1];
+ Công thức: F[i] = min(f[i-1]+t[i],f[i-2]+r[i-1]);
+ Kết quả bài toán là F[n].

Code tham khảo:

```
Uses    math;
Const   fi='';//'nktick.inp';
        fo='';//'nktick.out';
Var     l,tg,k,m,n,i,s,j,maxx:longint;
```

```
        a,f,g,r,t:array[-1..10000000]of int64;
BEGIN
        Assign(input,fi);Reset(input);
        Assign(output,fo);Rewrite(output);
        Read(n);
        f[0]:=0;
        For i:=1 to n do read(t[i]);
        For i:=1 to n-1 do read(r[i]);
        f[1]:=t[1];
        For i:=2 to n do
                f[i]:=min(f[i-1]+t[i],f[i-2]+r[i-1]);
        Write(f[n]);
        Close(output);
END.
```

2.24. Nguồn: http://vn.spoj.com/problems/NKCABLE/

Các học sinh khi đến thực tập trong phòng máy tính thường hay chơi trò chơi điện tử trên mạng. Để ngăn ngừa, người trực phòng máy đã ngắt tất cả các máy tính ra khỏi mạng và xếp chúng thành một dãy trên một cái bàn dài và gắn chặt máy xuống mặt bàn rồi đánh số thứ tự các máy từ 1 đến N theo chiều từ trái sang phải. Các học sinh tinh nghịch không chịu thua, họ đã quyết định tìm cách nối các máy trên bàn bởi các đoạn dây nối sao cho mỗi máy được nối với ít nhất một máy khác. Để tiến hành công việc này, họ đã đo khoảng cách giữa hai máy liên tiếp. Bạn hãy giúp các học sinh này tìm cách nối mạng thoả mãn yêu cầu đặt ra sao cho tổng độ dài cáp nối phải sử dụng là ít nhất.

Dữ liệu: Vào từ file văn bản NKCABLE.INP gồm:

- Dòng đầu tiên chứa số lượng máy N ($1 \le N \le 25000$).
- Dòng thứ i trong số N-1 dòng tiếp theo chứa các khoảng cách từ máy i đến máy i+1 (i=1,2,...,N-1). Giả thiết rằng khoảng cách từ máy 1 đến máy N không vượt quá 10^6.

***Kết quả*:** Ghi ra file NKCABLE.OUT gồm độ dài của cáp nối cần sử dụng.

Ví dụ:

NKCABLE.INP	NKCABLE.OUT
6 2 2 3 2 2	7

Thuật toán:

+ Giải thích test đề bài tại sao ra 7. Ta nối máy 1 với máy 2, máy 3 với máy 4, máy 5 với máy 6.

+ Gọi F[i] là độ dài của cách nối mạng ngắn nhất xét từ máy 1 … i.

+ Rõ ràng nhận thấy F[1] = 0; F[2] = a[1]; F[3] = a[1] + a[2]

+ Công thức quy hoạch động F[i]= min(F[i-1], F[i-2]) + a[i-1]

+ Kết quả bài toán là F[n]

Code tham khảo:

```
uses math;
const fi='';//'nkcable.inp';
      fo='';//'nkcable.out';
      maxn=25000;
var   a: array[0..maxn] of longint;
      f: array[0..maxn] of longint;
      i,n: longint;
BEGIN
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    readln(n);
    for i:=1 to n-1 do readln(a[i]);
    f[1]:=0;
    f[2]:=a[1];
    f[3]:= a[1] + a[2];
    for i:= 4 to n do
      f[i]:= min(f[i-1], f[i-2]) + a[i-1];
    writeln(f[n]);
    close(input); close(output);
END.
```

2.25. Nguồn: http://vn.spoj.com/problems/THEME/

Trong một bản nhạc thường có những đoạn nhạc mà tác giả sử dụng nó nhiều lần (ít nhất 2 lần). Những đoạn đó gọi là "đoạn cao trào". Do có thể sử dụng nhiều giọng khác nhau (son, la, si...) nên nốt đầu tiên của các lần xuất hiện có thể khác nhau, nhưng chệnh lệnh độ cao giữa hai nốt liên tiếp thì chắc chắn giống.

VD: hai đoạn sau 1 2 5 4 10 và 4 5 8 7 13 được coi là một đoạn cao trào, vì chúng cùng sự chênh lệch độ cao : +1,+3,-1,+6

Cho một bản nhạc, yêu cầu tìm độ dài đoạn cao trào dài nhất.

+ Đoạn cao trào phải có từ 5 nốt nhạc trở lên.

+ Những lần xuất hiện của đoạn không được chồng lên nhau (không có nốt nhạc chung).

Dữ liệu: Vào từ file văn bản THEME.INP gồm

+ Dòng 1 - số nguyên dương n là số nốt nhạc <= 5000

+ Một số dòng sau là n nốt nhạc, mỗi nốt được quy ra số tự nhiên trong phạm vi 1..88.

Kết quả: Ghi ra file THEME.OUT gồm 1 số duy nhất là độ dài đoạn cao trào dài nhất. Nếu không tìm được đoạn nhạc nào, in ra 0.

Ví dụ:

THEME.INP	THEME.OUT
30	5

25 27 30 34 39 45 52 60 69 79 69 60 52 45 39 34 30 26 22 18 82 78 74 70 66 67 64 60 65 80	

Thuật toán:

+ Gọi X[i] là độ chênh lệch giữa hai nốt nhạc liên tiếp
+ Gọi F[i,j] là độ dài đoạn cao trào
+ Khởi tạo F[i,j] = 0
+ Công thức quy hoạch động

F[i,j] = F[i-1,j-1] + 1 nếu X[i] = X[j]

+ Kết quả của bài toán max(min(f[i,j], j-i))

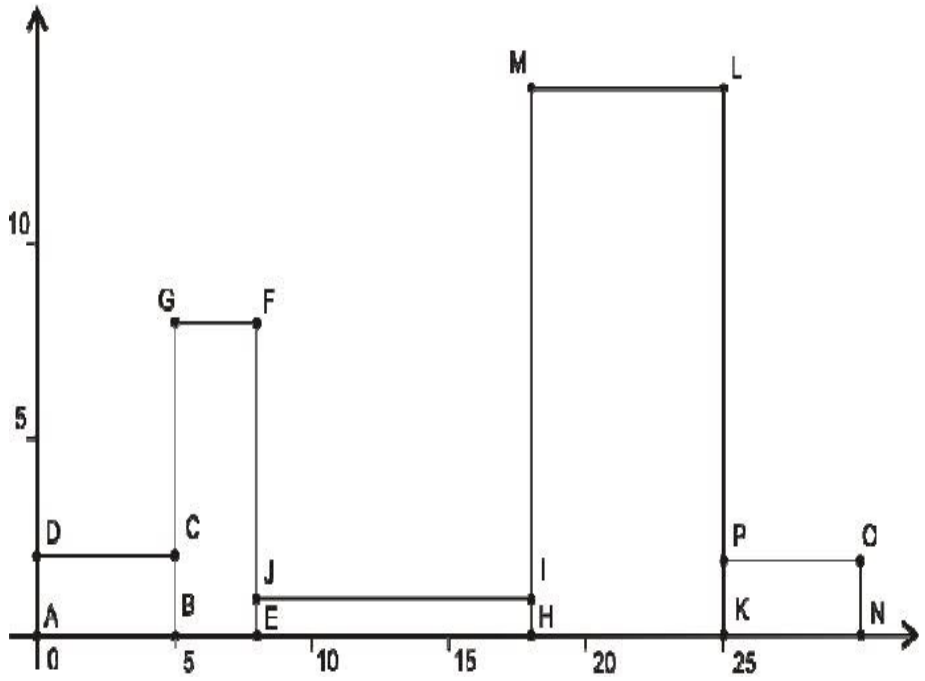
Code tham khảo:

```
USES MATH;
Const fi='theme.inp';
      fo='theme.out';
Var   a,g,x:array[0..1000000]of longint;
      i,n,k,j,o,l,tg,t,h,u,sum,kq:longint;
      f:array[0..10000,0..10000] of longint;
BEGIN
        Assign(input,fi);Reset(input);
        Assign(output,fo);Rewrite(output);
        Readln(n);
        For i:=1 to n do read(a[i]);
        For i:=1 to n-1 do x[i]:=a[i+1]-a[i];
        n:=n-1;kq:=0;
        For i:=1 to n-5 do
                For j:=i+5 to n do If x[i]=x[j] then
                        begin
                                f[i,j]:=f[i-1,j-1]+1;
                                kq:=max(kq,min(f[i,j],j-i));
                        end
                else f[i,j]:=0;
        inc(kq);
        If kq>=5 then Write(kq)
        else write(0);
        Close(output);
END.
```

2.26. Nguồn: http://vn.spoj.com/problems/MMAXPER/

Cho n hình chữ nhật đánh số từ 1 đến n, các hình chữ nhật này được đặt tiếp xúc với trục OX và nằm kề nhau từ trái qua phải theo thứ tự chỉ số

Mỗi hình chữ nhật có thể tiếp xúc với trục Ox theo bất kỳ cạnh cạnh (xem hình). Cần tính độ dài lớn nhất của đường gấp phía trên (xem hình)



Dữ liệu: Vào từ file MMAXPER.INP

+ Dòng đầu tiên ghi số hình chữ nhật n

+ n dòng tiếp theo mỗi dòng ghi hai số a_i và b_i, độ dài cạnh của hình chữ nhật thứ i.

Ràng buộc : $0 < n < 1000$; $0 < a_i < b_i < 1000$, với i = 1, 2, …, n.

Kết quả: Ghi ra file MMAXPER.OUT độ dài lớn nhất tìm được

Ví dụ:

MMAXPER.INP	MMAXPER.OUT
5 2 5 3 8 1 10 7 14 2 5	68

Giải thích: Cách xếp mà thu được chiều dài lớn nhất là hình trên. Cạnh phía trên gồm các đoạn DC, CG, GF, FJ, JI, IM, ML, LP, và PO. Độ dài của đoạn này là 5 + 6 + 3 + 7 + 10 + 13 + 7 + 12 + 5 = 68

Thuật toán:

+ Gọi f[1,i] là tổng chiều dài đường gấp khúc phía trên và bên phải ở ngoài cùng khi xét đến hình chữ nhật thứ i mà chiều đứng là a[i]. Gọi f[2,i] là tổng chiều dài đường gấp khúc phía trên và bên phải ở ngoài cùng khi xét đến hình chữ nhật thứ i mà chiều đứng là b[i]

+ Ta dễ nhận thấy: f[1,1] = f[2,1] = a[i] + b[i]

+ Công thức quy hoạch động

f[1,i]:=max(f[1,i-1]-a[i-1]+abs(a[i]-a[i-1])+a[i]+b[i],f[2,i-1]-b[i-1]+abs(a[i]-b[i-1])+a[i]+b[i]);

f[2,i]:=max(f[1,i-1]-a[i-1]+abs(b[i]-a[i-1])+a[i]+b[i],f[2,i-1]-b[i-1]+abs(b[i]-b[i-1])+a[i]+b[i]);

+ Kết quả bài toán: max(f[1,n]-a[n],f[2,n]-b[n])

Code tham khảo:

```
uses math;
const fi='';//MMAXPER.INP';
      fo='';//MMAXPER.OUT';
var   n:integer;
      a,b:array[1..1000] of integer;
      f:array[1..2,1..1000] of longint;
procedure   docfile;
   var i:integer;
   begin
      assign(input,fi);
      reset(input);
      readln(n);
      for i:=1 to n do
         readln(a[i],b[i]);
      assign(output,fo);
      rewrite(output);
   end;
procedure   dongfile;
   begin
      close(input);
      close(output);
   end;
procedure   xuli;
   var i:integer;
   begin
      f[1,1]:=a[1]+b[1];
      f[2,1]:=a[1]+b[1];
      for i:=2 to n do
         begin
            f[1,i]:=max(f[1,i-1]-a[i-1]+abs(a[i]-a[i-1])+a[i]+b[i],f[2,i-1]-
b[i-1]+abs(a[i]-b[i-1])+a[i]+b[i]);
            f[2,i]:=max(f[1,i-1]-a[i-1]+abs(b[i]-a[i-1])+a[i]+b[i],f[2,i-1]-
b[i-1]+abs(b[i]-b[i-1])+a[i]+b[i]);
         end;
      writeln(max(f[1,n]-a[n],f[2,n]-b[n]));
   end;
BEGIN
   docfile;
   xuli;
   dongfile;
END.
```

2.27. Nguồn: http://vn.spoj.com/problems/QBMSEQ/

Cho dãy số nguyên dương a_1, a_2, ..., a_n.

Dãy số: a_i, a_{i+1}, ..., a_j thỏa mãn $a_i \le a_{i+1} \le ... \le a_j$. Với $1 \le i \le j \le n$ được gọi là dãy con không giảm của dãy số đã cho và khi đó số j-i+1 được gọi là độ dài của dãy con này.

Yêu cầu: Trong số các dãy con không giảm của dãy số đã cho mà các phần tử của nó đều thuộc dãy số $\{u_k\}$ xác định bởi $u_1 = 1$, $u_k = u_{k-1} + k$ ($k \geq 2$), hãy tìm dãy con có độ dài lớn nhất.

Dữ liệu: Vào từ file văn bản QBMSEQ.INP gồm

+ Dòng đầu tiên chứa một số nguyên dương n ($n \leq 10^4$).

+ Dòng thứ i trong n dòng tiếp theo chứa một số nguyên dương a_i ($a_i \leq 10^8$) là số hạng thứ i của dãy số đã cho, i = 1, 2, ..., n.

Kết quả: Ghi ra file QBMSEQ.OUT gồm 1 dòng duy nhất ghi số nguyên d là độ dài của dãy con không giảm tìm được (quy ước rằng nếu không có dãy con nào thỏa mãn điều kiện đặt ra thì d = 0).

Ví dụ:

QBMSEQ.INP	**QBMSEQ.OUT**
8 2 2007 6 6 15 16 3 21	3

Thuật toán:

+ Thực tế bài này chẳng qua là biến đổi từ bài dãy con không giảm liên tiếp dài nhất.

+ Ta chứng minh được dãy $U_k = 1+2+..+k$. Ta dùng mảng kt để đánh dấu tất cả các phần tử của dãy U_k .

+ Gọi f[i] là độ dài dãy con liên tiếp dài nhất thỏa mãn bài toán mà có phần tử cuối cùng là phần tử a[i].

+ Nhận thấy f[0] = 0

+ Công thức quy hoạch động

f[i] = f[i-1] +1 nếu a[i] >= a[i-1] và a[i] thuộc dãy U_k

Ngược lại f[i] = 1 nếu a[i] thuộc dãy U_k

Ngược lại f[i] = 0 nếu a[i] không thuộc dãy U_k

+ Kết quả của bài toán là max(f[i])

Code tham khảo:

```
const   fi='';//'QBMSEQ.INP';
        fo='';//'QBMSEQ.OUT';
```

```
        nmax=10000;
        amax=100000000;
var     a,f:array[0..nmax] of longint;
        n,i,max:longint;
        kt:array[1..amax] of 0..1;
        t:longint;
procedure enter;
   begin
     assign(input,fi);reset(input);
     readln(n);
     for i:=1 to n do  readln(a[i]);
     a[0]:=0;
     close(input);
     fillchar(kt,sizeof(kt),0);
     kt[1]:=1; t:=1;i:=1;
     while t<amax do
         begin
             inc(i);
             t:=t+i;
             if t>amax then break;
             kt[t]:=1;
         end;
   end;
procedure solve;
    begin
        f[0]:=0;
        for i:=1 to n do
            begin
               if kt[a[i]]=1 then
                   if a[i]>=a[i-1] then f[i]:=f[i-1]+1
                   else f[i]:=1
               else f[i]:=0;
               if f[i]>max then max:=f[i];
            end;
    end;
procedure   solution;
   begin
     assign(output,fo);rewrite(output);
     writeln(max);
     close(output);
   end;
BEGIN
   max:=0;
   enter;
   solve;
   solution;
END.
```

2.28. THẮNG BỜM. Nguồn: Thầy Nguyễn Đức Nghĩa – ĐH Bách Khoa

Bờm thắng Phú Ông trong một cuộc đánh cược và buộc Phú Ông phải tặng quà cho Bờm. Bờm biết nhà Phú Ông có một hầm rượu nên rất thích được Phú Ông tặng rượu để mang về nhà trưng bày. Biết được suy nghĩ của Bờm nên Phú Ông bèn bày ra một dãy N chai rượu (đơn vị tính của mỗi chai là lít) và nói với Bờm rằng có thể lấy bao nhiêu tùy ý, nhưng không được lấy cả K chai liền nhau bởi đó là điều xui xẻo.

Yêu cầu: Bạn hãy chỉ cho Bờm cách lấy được nhiều lít rượu nhất.

Dữ liệu: Vào từ file văn bản THANGBOM.INP gồm:

- Dòng đầu chứa hai số nguyên N, K ($1 \le N \le 4.10^5$; $2 \le K \le 10^3$)

- Dòng 2 chứa N số nguyên dương (nhỏ hơn hoặc bằng 10^9) là dung tích của các chai rượu Phú Ông bày ra, theo thứ tự liệt kê từ chai thứ nhất tới chai thứ N.

(Các số trên một dòng được cách nhau bởi một dấu cách)

Kết quả: Ghi ra file văn bản THANGBOM.OUT một số duy nhất là kết quả của bài toán.

Ví dụ:

THANGBOM.INP	THANGBOM.OUT
4 3 10 10 2 3	23

Thuật toán:

+ Lưu các chai rượu vào a[i]. Gọi sum là tổng tất cả lít rượu của n chai

+ Để giải bài toán được đơn giản ta thêm chai rượu thứ n+1 là a[n+1] = 0 (chai thứ n+1 chứa 0 lít rượu). Gọi f[i] là số lít rượu bỏ đi ít nhất khi xét đến chai thứ i sao cho các chai còn lại ở trên bàn không có k chai nào liên tiếp và chai thứ i đó cũng phải bỏ đi.

+ Ta có f[0] = 0; f[i] = a[i] Với mọi i = 1..k

+ Công thức quy hoạch động

f[i] = min (f[i-k], f[i-k+1],...f[i-1]) + a[i]

+ Kết quả bài toán là sum – f[n+1]

Code tham khảo:

```
Const fi='thangbom.inp';
      fo='thangbom.out';
Var   a,f,g:array[0..1000]of longint;
      i,n,k,j,o,l,tg,t,h,u,sum:longint;
Function min(m,n:longint):longint;
Var i:longint;
begin
        If m<=0 then m:=1;
        If n-m+1<k then exit(0);
        min:=sum;
        For i:=m to n do if f[i]<min then min:=f[i];
end;
BEGIN
        Assign(input,fi);Reset(input);
        Assign(output,fo);Rewrite(output);
        Readln(n,k);
        For i:=1 to n do read(a[i]);
        For i:=1 to n do sum:=sum+a[i];
        f[0]:=0; a[n+1]:=0;
        For i:=1 to k do f[i]:=a[i];
        For i:=k+1 to n+1 do
            begin
               f[i]:=min(i-k,i-1)+a[i];
            end;
        Write(sum-f[n+1]);
        Close(output);
END.
```

2.29. Nguồn: http://vn.spoj.com/problems/STMERGE/

Cho 2 xâu ký tự $X = x_1, x_2, .., x_m$ và $Y = y_1, y_2, ..., y_n$. Cần xây dựng xâu $T = t_1, t_2, t_3, .., t_{n+m.}$ gồm tất cả các ký tự trong xâu X và tất cả các ký tự trong xâu Y, sao

cho các ký tự trong X xuất hiện trong T theo thứ tự xuất hiện trong X và các ký tự trong Y xuất hiện trong T theo đúng thứ tự xuất hiện trong Y, đồng thời với tổng chi phí trộn là nhỏ nhất. Tổng chi phí trộn hai xâu X và Y để thu được xâu T được tính bởi công thức c(T) = sum(c(t_k, t_{k+1})) với k = 1, 2, .., n+m-1; trong đó, các chi phí c(t_k, t_{k+1}) được tính như sau:

- Nếu hai ký tự liên tiếp t_k, t_{k+1} được lấy từ cùng một xâu X hoặc Y thì c(t_k, t_{k+1}) = 0
- Nếu hai ký tự liên tiếp t_k, t_{k+1} là x_i y_j thì chi phí phải trả là c(x_i, y_j). Nếu hai ký tự liên tiếp t_k, t_{k+1} là y_j, x_i thì chi phí phải trả là c(y_j, x_i) = c(x_i, y_j)

Dữ liệu: Vào từ file STMERGE.INP gồm

+ Dòng đầu tiên chứa Q là số lượng bộ dữ liệu. tiếp đến là Q nhóm dòng, mỗi nhóm cho thông tin về 1 bộ dữ liệu theo khuôn dạng sau:

+ Dòng thứ nhất chứa 2 số nguyên duong m, n (m, n <= 1000);

+ Dòng thứ i trong m dòng tiếp theo chứa n số nguyên dương, mỗi số không vượt quá 10^9: c(x_i, y_1), c(x_i, y_2), …, c(x_i, y_n), i = 1, 2,…, m.

Kết quả: Ghi ra file STMERGE .OUT gồm Q dòng, mỗi dòng chứa một số nguyên là tổng chi phí theo cách xây dựng xâu ***T*** tìm được tương ứng với bộ dữ liệu vào.

Ví dụ:

STMERGE.INP	STMERGE.OUT
1 2 3 3 2 30 15 5 4	6

Thuật toán:

+ Gọi f[i,j,0] là tổng nhỏ nhất tạo thành từ xâu X[1..i] và Y[1..j], trong đó kí tự cuối là X[j]. f[i,j,1] là tổng nhỏ nhất tạo thành từ xâu X[1..i] và Y[1..j], trong đó kí tự cuối là Y[j].

+ Nhận thấy f[i,0,0] = 0; f[0,j,1] = 0;

+ Công thức: f[i,j,0]:=min(f[i-1,j,0],f[i-1,j,1]+c[i,j]);
f[i,j,1]:=min(f[i,j-1,1],f[i,j-1,0]+c[i,j]);

+ Kết quả bài toán min(f[m,n,0], f[m,n,1]).

Code tham khảo:

```
uses math;
const fi='';//STMERGE.INP';
      fo='';//STMERGE.OUT';
      maxv=round(1e18);
var   f:array[0..1001,0..1001,0..1] of int64;
```

```
      c:array[0..1001,0..1001] of int64;
      m,n:longint;
procedure   docfile;
   begin
      assign(input,fi);
      reset(input);
      assign(output,fo);
      rewrite(output);
   end;
procedure   dongfile;
   begin
      close(input);  close(output);
   end;
procedure   xuly;
   var i,j:longint;
   begin
      readln(m,n);
      for i:=1 to m do
         for j:=1 to n do
            read(c[i,j]);
      for i:=0 to m do
         for j:=0 to n do
            begin
               f[i,j,0]:=maxv;   f[i,j,1]:=maxv;
            end;
      for i:=0 to m do
         f[i,0,0]:=0;
      for i:=0 to n do
         f[0,i,1]:=0;
      for i:=1 to m do
         for j:=1 to n do
            begin
               f[i,j,0]:=min(f[i-1,j,0],f[i-1,j,1]+c[i,j]);
               f[i,j,1]:=min(f[i,j-1,1],f[i,j-1,0]+c[i,j]);
            end;
      writeln(min(f[m,n,0],f[m,n,1]));
   end;
procedure   xuli;
   var test,t:longint;
   begin
      readln(test);
      for t:=1 to test do
         xuly;
   end;
BEGIN
   docfile;
   xuli;
   dongfile;
END.
```

2.30. Nguồn: http://vn.spoj.com/problems/QBPAL/

Trong một buổi học viết chữ, Bờm phát hiện trong một số từ khi bỏ đi một số ký tự thì đọc ngược hay đọc xuôi đều giống nhau.

Ví dụ từ IOICAMP, khi xóa đi các chữ cái C,A,M,P, thì còn lại IOI là một từ đối xứng.

Bờm cảm thấy thú vị, và cậu tiếp tục thử xóa các ký tự khác, kết quả là có thêm nhiều từ đối xứng nữa: II, I, O, C… Nhưng nếu với một từ dài, cứ thử từng cách xóa như vậy thì thật mất thời gian. Bạn hãy viết chương trình giúp Bờm tính

số cách xóa sao cho từ thu được đối xứng. Hai cách xóa chỉ khác nhau bởi thứ tự xóa các ký tự thì coi như trùng nhau.

Dữ liệu: Vào từ file văn bản QBPAL.INP một dòng duy nhất là từ cần tính số cách xóa, từ này chỉ chứa các chữ cái in hoa A…Z. (Độ dài từ không quá 120)

Kết quả: Ghi ra file QBPAL.INP - một số duy nhất là số cách xóa.

Ví dụ:

QBPAL.INP	QBPAL.OUT
IOICAMP	9

Thuật toán:

+ Gọi f[i,j] số xâu con đối xứng khi xét đoạn con từ ký tự thứ i đến ký tự thứ j.

+ Nhận thấy f[i,i] = 1

+ Công thức quy hoạch động:

f[i,j] = f[i+1,j] + f[i,j-1] + 1 nếu st[i] = st[j]

Ngược lại f[i,j] = f[i+1,j] + f[i,j-1] – f[i+1,j-1]

+ Kết quả bài toán là f[1,n]. Vì kết quả ra số lớn nên ta sử dụng cộng và trừ hai số lớn.

Code tham khảo:

```
const fi='';//QBPAL.INP';
      fo='';//QBPAL.OUT';
var   n:byte;
      st:string;
      f:array[0..120,0..120] of string;
procedure   docfile;
   begin
      assign(input,fi);
      reset(input);
      readln(st);
      n:=length(st);
      assign(output,fo);
      rewrite(output);
   end;
procedure   dongfile;
   begin
      close(input);
      close(output);
   end;
function cong(a,b:string):string ;
   var p1:string;
       x,y,nho,s,d,i:byte;
   begin
      while length(a)<>length(b) do
         if length(a)<length(b) then a:='0'+a
         else
         b:='0'+b;
      i:=length(a);  nho:=0;  cong:='';
      repeat
         val(a[i],x);
         val(b[i],y);
         dec(i);
```

```
         s:=x+y+nho;
         nho:=s div 10;
         s:=s mod 10;
         str(s,p1);
         cong:=p1+cong;
      until i<=0;
      if nho>0 then
         begin
            str(nho,p1);
            cong:=p1+cong;
         end;
      while (cong[1]='0') and (length(cong)>1) do delete(cong,1,1);
   end;
function tru(a,b:string):string;
   var p1:string;
       x,y,s,nho,d,i:integer;
   begin
      while length(b)<length(a) do
         b:='0'+b;
      i:=length(a);  nho:=0;  tru:='';
      repeat
         val(a[i],x);
         val(b[i],y);
         dec(i);
         s:=x-y+nho;
         if s<0 then
            begin
               nho:=-1;
               s:=s+10;
            end
         else
            nho:=0;
         s:=s mod 10;
         str(s,p1);
         tru:=p1+tru;
      until i<=0;
      while (tru[1]='0') and (length(tru)>1) do delete(tru,1,1);
   end;
procedure   xuli;
   var i,j:byte;
   begin
      for i:=1 to n do
         f[i,i]:='1';
      for i:=1 to n-1 do
         for j:=1 to n-i do
            begin
               if st[j]=st[i+j] then
                  begin
                    f[j,i+j]:=cong(f[j+1,i+j],f[j,i+j-1]);
                    f[j,i+j]:=cong(f[j,i+j],'1');
                  end
               else
                  begin
                     f[j,i+j]:=cong(f[j+1,i+j],f[j,i+j-1]);
                     f[j,i+j]:=tru(f[j,i+j],f[j+1,i+j-1]);
                  end;
            end;
      writeln(f[1,n]);
   end;
BEGIN
   docfile;
   xuli;
   dongfile;
END.
```

2.31. Nguồn: http://www.spoj.com/PTIT/problems/P154PROD/

Cooper đang chơi trận địa xe tăng với TARS, các anh chơi trên tấm bản đồ là ma trận a với m hàng và n cột, ô a[i][j] có một số điểm mà khi ai di chuyển xe tăng qua sẽ được cộng số điểm ấy vào số điểm của mình sau khi kết thúc trò chơi.

Ban đầu cả số điểm cả hai đều bằng 0, Cooper di chuyển từ ô a[1][1] đến ô a[m][n], sau khi ăn điểm ở ô a[i][j] thì anh sẽ di chuyển đến một trong hai ô a[i][j+1] hoặc a[i+1][j], TARS di chuyển từ ô a[m][1] xuống ô a[1][n], sau khi đến ô a[i][j] thì anh sẽ di chuyển đến một trong hai ô a[i][j+1] hoặc a[i-1][j], các xe tăng luôn di chuyển bên trong bản đồ. Có một quy tắc là hai xe tăng chỉ có thể cùng đi qua một ô duy nhất và ô đấy sẽ không tính điểm cho cả 2.

CASE đứng ngoài theo dõi và anh đang tự đặt ra một câu hỏi là tổng số điểm cao nhất của cả Cooper và TARS cộng lại là bao nhiêu.

Hãy giúp anh ấy.

Dữ liệu: Vào từ file văn bản P154PROD.INP gồm

+ Dòng đầu tiên chứa hai số tự nhiên n và m ($3 \le m, n \le 1000$).

+ N dòng sau dòng thứ i chứa m số nguyên không âm, số thứ j là a[i][j] - số điểm ở ô (i, j). ($0 \le a[i][j] \le 100\ 000$).

Kết quả: Ghi ra file P154PROD.OUT gồm một dòng duy nhất chứa kết quả của bài toán.

Ví dụ:

P154PROD.INP	P154PROD.OUT
3 3 100 100 100 100 1 100 100 100 100	800

Giải thích: Cooper sẽ di chuyển từ $a[1][1] \rightarrow a[1][2] \rightarrow a[2][2] \rightarrow a[3][2] \rightarrow a[3][3]$, TARS sẽ di chuyển từ $a[3][1] \rightarrow a[2][1] \rightarrow a[2][2] \rightarrow a[2][3] \rightarrow a[1][3]$

Thuật toán:

+ Dữ liệu quan trọng của bài toán Cooper và TARS cùng đi qua 1 ô duy nhất và ô đó sẽ không được tính điểm cho cả hai.

+ Gọi f1[i,j] là tổng số điểm đi từ ô (1,1) đến ô (i,j)

+ Gọi f2[i,j] là tổng số điểm đi từ ô (m,n) đến ô (i,j)

+ Gọi g1[i,j] là tổng số điểm đi từ ô (m,1) đến ô (i,j)

+ Gọi g2[i,j] là tổng số điểm đi từ ô (1,n) đến ô (i,j)

+ Tính f1[i,j], f2[i,j], g1[i,j], g2[i,j] như trong code.

+ Kết quả của bài toán là max(f1[i,j-1]+f2[i,j+1]+g1[i+1,j]+g2[i-1,j],f1[i-1,j]+f2[i+1,j]+g1[i,j-1]+g2[i,j+1])

+ Chú ý khi cài đặt: đối với những ô ở viền thì nó không bao giờ là ô giao nhau của hai xe tăng.

Code tham khảo:

```
const   fi='';//'p154prod.inp';
        fo='';//'p154prod.out';
var    a: array[1..1000,1..1000] of longint;
       f1,f2,g1,g2: array[0..1001,0..1001] of int64;
       i,j,m,n: longint; res: int64;
function    max(a,b: int64): int64;
    begin
        if a <b then exit(b);
        exit(a);
    end;
BEGIN
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(m,n);
    fillchar(f1, sizeof(f1),0);
    fillchar(f2, sizeof(f2),0);
    fillchar(g1, sizeof(g1),0);
    fillchar(g2, sizeof(g2),0);
        for i:= 1 to m do
          begin
           for j:= 1 to n do read(a[i,j]);
           readln(input);
          end;
    for i:=1 to m do
        for j:= 1 to n do
            f1[i,j]:= max(f1[i-1,j],f1[i,j-1])+a[i,j];
    for i:=m downto 1 do
        for j:= n downto 1 do
            f2[i,j]:= max(f2[i+1,j],f2[i,j+1])+a[i,j];
    for i:= m downto 1 do
        for j:=1 to n do
            g1[i,j]:= max(g1[i,j-1],g1[i+1,j])+a[i,j];
    for i:= 1 to m do
        for j:= n downto 1 do
            g2[i,j]:= max(g2[i-1,j],g2[i,j+1])+a[i,j];
    res:=0;
    for i:=2 to m-1 do
        for j:=2 to n-1 do res:=max(res,max(f1[i,j-1]+f2[i,j+1]+g1[i+1,j]
+g2[i-1,j],f1[i-1,j]+f2[i+1,j]+g1[i,j-1]+g2[i,j+1]));
    writeln(res);
    close(input); close(output);
END.
```

2.32. Cắt vải

Cho một mảnh vải hình chữ nhật có kích thước là m*n. Hãy tìm cách cắt mảnh vải đó thành các hình vuông sao cho số hình vuông là ít nhất.

Dữ liệu*:* Vào file CATVAI.INP: gồm 2 số nguyên dương m, n (m, n <= 100)

Kết quả: ghi ra file CATVAI.OUT: gồm 1 số duy nhất là số lượng hình vuông ít nhất

Ví dụ:

CATVAI.INP	CATVAI.OUT
4 3	4

Thuật toán:

+ Gọi f[i,j] là số hình vuông ít nhất khi cắt mảnh vải có kích thước i*j

+ Khởi tạo tất cả các f[i,j] = 0;

+ Công thức quy hoạch động: f(i,j) = min (f(i-c,j-c)+ f(i-c,c)+ f(c, j-c))+1; với mọi c= 1.. min(i,j).

Ngoài công thức này ra chúng ta có thể xây dựng công thức khác như sau: f(i,j)= Min(f(i-c,j) + f(i,j-c)- f(i-c,j-c))+1 với mọi c = 1..min(i,j)

+ Kết quả của bài toán là f[m,n].

Code tham khảo:

```
Uses math;
Const fi='CATVAI.INP';
      fo='CATVAI.OUT';
Var c,i,j,m,n: longint;
    min1,s: longint;
    f: array[0..100,0..100] of longint;
BEGIN
   Assign(input,fi);Reset(input);
   Assign(output,fo);Rewrite(output);
   Read(m,n);
   If m = n then write(1)
   Else
    Begin
     Fillchar(f,sizeof(f),0);
     For i:=1 to m do
      For j:=1 to n do
       Begin
        min1:=maxlongint;
        For c:=1 to min(i,j) do
         Begin
          s:=f[i-c,j-c]+f[i-c,c]+f[c,j-c]+1;
          If s<min1 then min1:=s;
         End;
       f[i,j]:= min1;
       End;
    End;
   Write(f[m,n]);
   Close(output);
END.
```

2.33. Nguồn: http://vn.spoj.com/problems/V11WATER/

Năm 2011, tình trạng ngập lụt trong thành phố trở lên nghiêm trọng hơn. Vì vậy, mọi người quyết định xây dựng hệ thống mái che cho toàn thành phố.

Mái che có bề rộng là N, được chia làm N phần có độ dài như nhau. Độ cao của mỗi phần là h1, h2, ..., hn. Khi trời mưa, một phần nước sẽ đọng lại trên mái và một phần sẽ thoát ra ngoài theo hai bên trái và phải của mái che. Do đó, thành phố sẽ không phải chịu cảnh mưa lụt như trước.

Nhằm mục đích bảo trì mái che, bạn cần viết chương trình tính lượng nước lớn nhất có thể đọng lại trên mái che.

Dữ liệu: Vào từ file V11WATER.INP gồm

- Dòng đầu ghi số N. (1 <= N <= 100000)
- Dòng sau ghi N số tự nhiên h1, h2, ..., hn. (1 <= hi <= 100000)

Kết quả: Ghi ra file V11WATER.OUT gồm một số duy nhất thể hiện lượng nước tìm được.

Ví dụ:

V11WATER.INP	V11WATER.OUT
5 1 3 1 2 3	3

Thuật toán:

+ Đọc dữ liệu các độ cao vào a[i]

+ Gọi maxl[i] là độ cao lớn nhất khi xét từ phần 1 đến phần thứ i. Gọi maxr là độ cao lớn nhất khi xét từ phần thứ thứ i đến phần thứ N.

+ Ta nhận thấy maxr[n+1] = maxl[0] = 0; maxr[i] = maxl[i] = a[i];

+ Công thức tính : maxr[i] = maxr[i+1] nếu maxr[i] <maxr[i+1]

maxl[i] = maxl[i-1] nếu maxl[i] < maxl[i-1]

+ Kết quả của bài toán là $\sum_{i=2}^{n-1} \min(\text{maxl}[i-1], \text{maxr}[i+1]) - a[i]$

Code tham khảo:

```
const   fi='';//'v11wate.inp';
        fo='';//'v11wate.out';
var a:array[0..100000] of longint;
    maxl,maxr:array[0..100001] of longint;
    k,i,n,j:longint;
    tong: int64;
procedure max;
    var k:longint;
    begin
        for i:=n downto 1 do
            begin
               maxr[i]:=a[i];
               if maxr[i]<maxr[i+1] then maxr[i]:=maxr[i+1];
            end;
    end;
function    min(a,b:longint):longint;
    begin
        min:=a;
        if a>b then min:=b;
    end;
BEGIN
    assign(input,fi);reset(input);
    assign(output,fo);rewrite(output);
    readln(n);
    for i:=1 to n do
      begin
        read(a[i]);
```

```
        maxl[i]:=a[i];
        if maxl[i]<maxl[i-1] then maxl[i]:=maxl[i-1];
      end;
    close(input);
    max;
    tong:=0;
    for i:=2 to n-1 do
      if min(maxl[i-1],maxr[i+1])>a[i] then tong:=tong+min(maxl[i-
1],maxr[i+1])-a[i];
    writeln(tong);
    close(output);
END.
```

CHƯƠNG V. ĐỒ THỊ

1. Khái niệm đồ thị

- Đồ thị G (V, E) bao gồm một tập hữu hạn V các đỉnh và một tập hữu hạn E các cặp đỉnh (gọi là cung hay cạnh)

- Nếu giữa hai đỉnh có đường nối một chiều và không có đường nối với chính nó thì đồ thị đó được gọi là đơn đồ thị có hướng.

- Nếu giữa hai đỉnh có đường nối không quy định hướng và không có đường nối một đỉnh với chính nó thì gọi là đơn đồ thị vô hướng

- Hai đỉnh u, v có đường nối trực tiếp vô hướng thì ta nói đồ thị có cạnh (u, v). Nếu là đường nối có hướng thì ta gọi là cung (u, v).

- Nếu đồ thị có cạnh (cung) nối (u, v) thì v gọi là kề đỉnh u.

- Bậc của một đỉnh là số cạnh đi ra từ đỉnh đó. Kí hiệu là deg (u): trong đó u là số hiệu của đỉnh.

- Nếu là đồ thị có hướng thì có khái niệm bán bậc ra bằng số cung đi ra khỏi đỉnh đó, bán bậc vào bằng số cung đi vào đỉnh đó:

  + deg^+ (u): bán bậc ra của đỉnh u

  + deg^- (u): bán bậc vào của đỉnh u

- Đường đi có đỉnh xuất phát trùng với đỉnh đích gọi là chu trình

- Đồ thị vô hướng được gọi là liên thông nếu luôn tìm được đường đi giữa hai đỉnh bất kì của nó.

2. Biểu diễn đồ thị trên máy tính

Thông thường chúng ta có thể biểu diễn đồ thị trên máy tính bằng 3 phương pháp sau:

2.1. Dùng ma trận kề

- Sử dụng ma trận A[i, j] để biểu diễn đồ thị gồm n đỉnh. A[i, j] = 0 nếu (i, j) không có cạnh (cung) nối, A[i, j] = 1 nếu (i, j) có cạnh nối.

- Nếu G là đồ thị vô hướng thì A là ma trận đối xứng A[i, j] = A[j, i]

- Nếu mỗi cạnh hoặc cung (u, v) được gán với 1 số c(u, v) gọi là trọng số (hay giá trị) của (u, v) thì G được gọi là đồ thị có trọng số:

$$\begin{cases} A[u, v] = 0; +\infty \text{ Nếu } (u, v) \notin E \\ A[u, v] = c[u, v] \text{ Nếu } (u, v) \in E \end{cases}$$

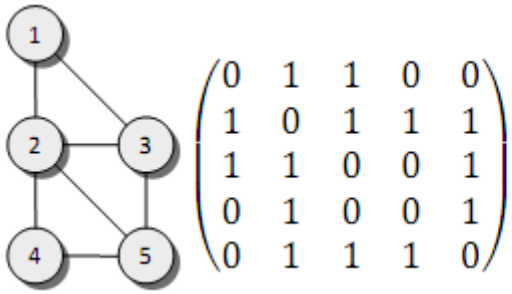
Ưu điểm: dễ biểu diễn, chỉ mất một phép so sánh để kiểm tra 2 đỉnh có kề nhau hay không

Nhược điểm:

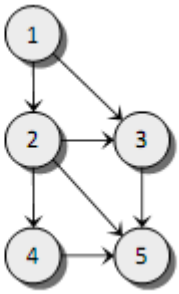
+ Bất kể số cạnh của đồ thị nhiều hay ít thì ta cũng cần n^2 ô nhớ để lưu các phần tử, điều đó gây lãng phí bộ nhớ.

+ Liệt kê tất cả các đỉnh v kề với đỉnh u thì phải duyệt qua n đỉnh

Ví dụ:



$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$



$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2.2. Dùng danh sách cạnh

Với đồ thị G (V, E) có n đỉnh và m cạnh (cung), ta có thể liệt kê tất cả các cạnh của đồ thị trong một danh sách. Với mỗi phần tử là một cặp (u, v) tương ứng là cạnh (cung của đồ thị).

Khai báo: dùng một mảng có m phần tử mà mỗi phần tử có 2 hoặc 3 thuộc tính: u, v, w (w là trọng số nếu có)

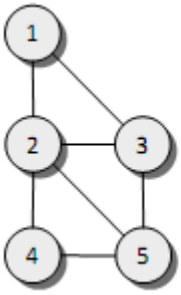
Ưu điểm:

+ Khi đồ thị ít cạnh thì sẽ tiết kiệm được không gian lưu trữ

+ Trong trường hợp xét tất cả các cạnh của đồ thị thì duyệt đơn giản

Nhược điểm: Kiểm tra xem hai đỉnh u, v có kề nhau hay không thì phải duyệt qua tất cả các cạnh của đồ thị, nếu là đồ thị dày thì điều đó khá tốn thời gian

Ví dụ:



1	2	3	4	5	6	7
(1,2)	(1,3)	(2,3)	(2,4)	(2,5)	(3,5)	(4,5)

2.3 Dùng danh sách kề

- Để khắc phục được nhược điểm của hai cách trên thì người ta dùng danh sách kề. Có hai cách cài đặt danh sách kề phổ biến.

+ Forward Star: Với mỗi đỉnh u, lưu trữ một danh sách ke[u] chứa các đỉnh nối từ u

+ Reverse Star: Với mỗi đỉnh v, lưu trữ một danh sách ke[v] chứa các đỉnh nối tới v

- Dùng mảng ke[1..m] chứa các đỉnh (nếu là đồ thị vô hướng thì m = 2*m), dùng thêm mảng head[1..n+1] với ý nghĩa head[u] là chỉ số liền trước của đoạn thứ u hay nói cách khác là chỉ số cuối của đoạn thứ u-1

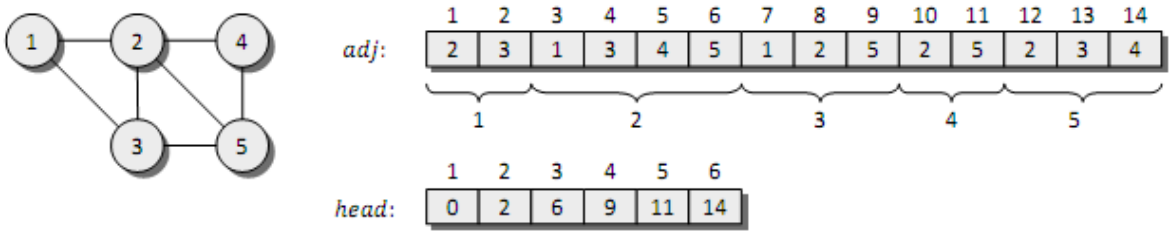
Ưu điểm:

+ Duyệt tất cả các đỉnh kề với đỉnh u thật đơn giản

+ Duyệt tất cả các cạnh cũng đơn giản

Nhược điểm: Muốn kiểm tra (u, v) có kề nhau hay không thì nó phải duyệt tất cả các đỉnh kề với đỉnh u hoặc v

Ví dụ:



3. Chuyển đổi giữa các cách biểu diễn đồ thị

Giả sử ta có đồ thị G (V, E) có n đỉnh và m cạnh thì ta có các cách khai báo sau:

- Ma trận kề:

```
var   a: array[1..n, 1..n] of 0..1;
```

- Danh sách cạnh

```
type  te = record
      u, v: longint;
      end;
var   e: array[1..m] of te;
```

- Danh sách kề:

```
var   ke: array[1..m*2] of longint;
      head: array[1.. n+1] of longint;
```

3.1. Ma trận kề sang danh sách cạnh

```
k:= 0;
for i:= 1 to n do
      for j:= 1 to n do
            if a[i,j] = 1 then
                  begin
                        inc(k);
                        e[k].u:= i; e[k].v:= j;
                  end;
```

3.2. Danh sách cạnh sang ma trận kề

```
Fillchar(a,sizeof(a),0);
For k:= 1 to m do a[e[k].u, e[k].v]:= 1;
```

3.3. Ma trận kề sang danh sách kề

```
head[n+1]:= m;
for i:= n downto 1 do
   begin
      head[i]:= head[i+1];
      for j:= n downto 1 do
         if a[i,j] = 1 then
            begin
                  ke[head[i]]:= j;
```

```
                head[i]:= head[i] - 1;
            end;
    end;
```

3.4. Danh sách kề sang ma trận kề

```
Fillchar(a,sizeof(a),0);
For u:= 1 to n do
      For k:= head[u]+1 to head[u+1] do a[u, ke[k]]:= 1;
```

3.5. Danh sách cạnh sang danh sách kề

```
for u:= 1 to n do head[u]:= 0;
// Tính head[u] là bậc của đỉnh u
for i:= 1 to m do inc(head[e[i].u]);
// head[u] là vị trí cuối của đoạn u
for u:= 2 to n do head[u]:= head[u-1] + head[u];
for i:= m downto 1 do
      begin
            ke[head[e[i].u]:= e[i].v;
            dec(head[e[i].u]);
      end;
head[n+1]:= m;
```

3.6. Danh sách kề sang danh sách cạnh

```
i:= 0;
for u:= 1 to n do
      for k:= head[u]+1 to head[u+1] do
            begin
                  inc(i);
                  e[i].u:= u; e[i].v:= ke[k];
            end;
```

4. Tìm kiếm trên đồ thị

4.1. Thuật toán tìm kiếm theo chiều sâu

```
procedure DFSVisit(u∈V); //Thuật toán tìm kiếm theo chiều sâu từ đỉnh u
begin
  avail[u] := False; //avail[u] = False ⇔ u đã thăm
  Output ← u; //Liệt kê u
  for ∀v∈V:(u, v)∈E do //Duyệt mọi đỉnh v chưa thăm nối từ u
    if avail[v] then
      begin
        trace[v] := u; //Lưu vết đường đi, đỉnh liền trước v trên đường đi
từ s tới v là u
        DFSVisit(v); //Gọi đệ quy để tìm kiếm theo chiều sâu từ đỉnh v
      end;
end;
begin //Chương trình chính
  Input → Đồ thị G, đỉnh xuất phát s, đỉnh đích t;
  for ∀v∈V do avail[v] := True; //Đánh dấu mọi đỉnh đều chưa thăm
  DFSVisit(s);
  if avail[t] then //s đi tới được t
    «Truy theo vết từ t để tìm đường đi từ s tới t»;
end.
```

4.2. Thuật toán tìm kiếm theo chiều rộng

```
Queue := (s); //Khởi tạo hàng đợi chỉ gồm một đỉnh s
for ∀v∈V do
  avail[v] := True;
avail[s] := False; //Đánh dấu chỉ có đỉnh s được xếp hàng
repeat //Lặp tới khi hàng đợi rỗng
  u := Pop; //Lấy từ hàng đợi ra một đỉnh u
  Output ← u; //Liệt kê u
  for ∀v∈V:avail[v] and (u, v)∈E do //Xét những đỉnh v kề u
chưa được đẩy vào hàng đợi
    begin
      trace[v] := u; //Lưu vết đường đi
      Push(v); //Đẩy v vào hàng đợi
      avail[v] := False; //Đánh dấu v đã xếp hàng
    end;
until Queue = Ø;
if avail[t] then //s đi tới được t
  «Truy theo vết từ t để tìm đường đi từ s tới t»;
```

5. Thuật toán Dijstra tìm đường đi ngắn nhất trên đồ thị có trọng số không âm

Trong các ứng dụng thực tế, chẳng hạn như trong mạng lưới giao thông, người ta không chỉ quan tâm đến việc tìm đường đi giữa hai địa điểm mà còn phải lựa chọn một hành trình tiết kiệm nhất (theo tiêu chuẩn không gian, thời gian hay một đại lượng mà ta cần giảm thiểu theo hành trình). Khi đó người ta có thể quy về đồ thị, với mỗi cạnh của đồ thị là một giá trị phản ánh chi phí đi qua cạnh đó và cố gắng tìm một con đường mà tổng chi phí đi qua là nhỏ nhất.

Dữ liệu vào: Vào từ file Dijkstra.inp gồm

❖ Dòng đầu là 4 số nguyên dương *n, m, s, t*. Trong đó n là số đỉnh, m là số cạnh của đồ thị, *s* là đỉnh xuất phát, *t* là đỉnh đích. ($n \le 10^4$, $m \le 10^5$).

❖ *m* dòng tiếp theo, mỗi dòng chứa 3 số *u, v, w*. Trong đó *w* là trọng số của cạnh (*u, v*). ($0 \le w \le 10^5$).

Kết quả ra: Ghi ra file Dijkstra.out gồm:

❖ Dòng đầu là độ dài đường đi ngắn nhất (nếu không có đáp án thì in ra -1)

❖ Dòng thứ hai là thứ tự đường đi từ s đến t (nếu có nhiều đáp án chỉ in ra 1 đáp án)

Dijkstra.inp	**Dijkstra.out**
4 5 1 2 1 2 10 2 3 4 1 3 5 1 4 1 4 3 2	7 1 4 3 2

Thuật toán:

Bước 1. Gọi f[*u*] là độ dài đường đi ngắn nhất từ đỉnh xuất *s* phát đến đỉnh *u*. Như vậy ta có f[*s*] := 0 và f[*u*] := $+\infty$; // $\forall u \neq s$. Đầu tiên ta đánh dấu tất cả các đỉnh chưa được lấy.

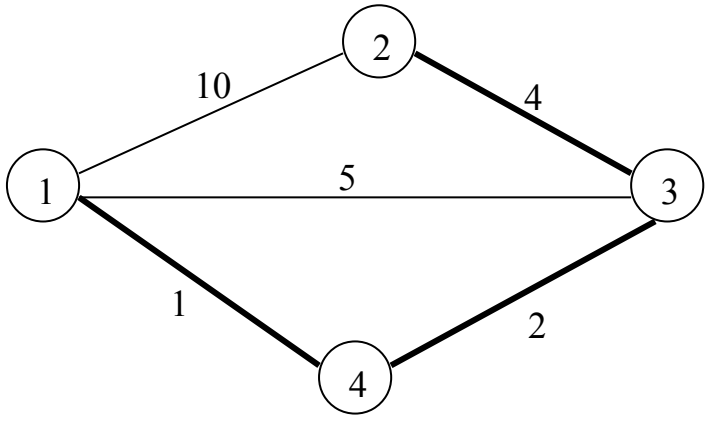
Bước 2. Lặp:

❖ Chọn trong các đỉnh chưa được lấy, lấy ra một đỉnh *u* có f[*u*] nhỏ nhất và đánh dấu đỉnh u đã được lấy.

❖ Nếu đỉnh *u* là đỉnh đích hoặc tất cả các đỉnh của đồ thị đã được lấy thì thoát khỏi vòng lặp.

❖ Dùng đỉnh *u*, xét tất cả đỉnh *v* kề của đỉnh *u* và thực hiện phép co cạnh (*u, v*) để cực tiểu hóa nhãn f[*v*] theo công thức f[*v*]:= Min(f[*v*], f[*u*] + *w(u, v)*).

Bước 3: Ghi ra kết quả: Nếu f[*t*] = $+\infty$ thì không tồn tại đường đi, ngược lại thì f[*t*] là kết quả của bài toán.



Quá trình tìm đường đi ngắn nhất từ đỉnh 1 đến đỉnh 2 được xây dựng như sau, trong đó kí hiệu * là đỉnh được chọn.

Bảng 3.3. Bảng kết quả tính toán theo thuật toán Dijkstra

Lặp	Tập đỉnh	Đến đỉnh	F[1]	F[2]	F[3]	F[4]
Khởi tạo	ϕ	1	0*	$+\infty$	$+\infty$	$+\infty$
1	{1}	4	-	10	5	1*
2	{1, 4}	3	-	10	3*	-
3	{1, 4, 3}	(2)	-	7*	-	-

Code tham khảo:

```
const fi='dijkstra.inp';
      fo='dijkstra.out';
      maxn=10000;
      maxc= round(1e9);
var   c: array[1..maxn, 1..maxn] of longint;
      u,v,i,j,n,m,s,t: longint;
      kt: array[1..maxn] of boolean;
      f: array[1.. maxn] of longint;
      tr: array[1..maxn] of longint;
procedure   dijkstra;
var i,j,min,u,v: longint;
begin
   for i:=1 to n do f[i]:= maxc;
   f[s]:= 0;
   fillchar(kt,sizeof(kt), true);
   while true do
      begin
         u:=0; min:= maxc;
         for i:=1 to n do
            if (kt[i]) and (f[i] < min) then
               begin
                   min:= f[i];
                   u:=i;
               end;
         if (u=0) or (u=t) then break;
         kt[u]:= false;
         for v:= 1 to n do
            if kt[v] and(f[v] > f[u] + c[u,v]) then
               begin
                  f[v]:= f[u]+ c[u,v];
                  tr[v]:= u;
               end;
      end;
```

```
end;
procedure   inkq;
var   i,j: longint;
      a: array[1.. maxn] of longint;
begin
   if f[t] = maxc then writeln(-1)
   else
      begin
          writeln(f[t]);
          i:=0;
          while t<>s do
            begin
                inc(i);
                a[i]:= t;
                t:= tr[t];
            end;
          write(s,' ');
          for j:= i downto 1 do write(a[j],' ');
      end;
end;

BEGIN
   assign(input, fi); reset(input);
   assign(output, fo); rewrite(output);
   readln(n,m,s,t);
   for i:=1 to n do
      for j:=1 to n do
         if i=j then c[i,j]:= 0
         else c[i,j]:= maxc;
   for i:=1 to m do
      begin
         readln(u,v,c[u,v]);
         c[v,u]:= c[u,v];
      end;
   dijkstra;
   inkq;
   close(input);
   close(output);
END.
```

Chú ý: Ngoài cách dùng ma trận kề thì chúng ta có thể dùng danh sách kề để cài đặt dijkstra thì chương trình sẽ chạy nhanh hơn.

6. Bài tập ứng dụng

6.1. Miền 0 (Nguồn: Đề thi học sinh giỏi tỉnh Thanh Hóa năm 2009)

Cho một hình chữ nhật gồm M hàng, N cột, được chia thành MxN ô vuông. Mỗi ô vuông được ghi một trong hai số nguyên 0 hoặc 1.

Miền 0 là một miền liên tục các số 0 thuộc các ô chung cạnh với nhau. Diện tích miền là số lượng các ô vuông cùng giá trị thuộc miền đó.

Yêu cầu: Tính diện tích miền 0 lớn nhất của hình chữ nhật đã cho.

Dữ liệu: Vào từ file văn bản BAI5.INP:

- Dòng đầu tiên ghi hai số M, N.
- M dòng tiếp theo, mỗi dòng ghi N số lần lượt là giá trị các ô trong bảng số.

Kết quả: Ghi ra file văn bản BAI5.OUT số nguyên duy nhất là diện tích miền 0 lớn nhất.

Giới hạn: $1 < M, N < 100$.

Ví dụ:

BAI5.INP	BAI5.OUT
3 4 0 0 0 1 1 1 0 1 0 0 1 0	4

***Thuật toán**:* Thực chất bài này là tìm miền liên thông lớn nhất, ta sử dụng thuật toán DFS hay BFS để tìm thành phần phần liên thông. Duyệt toàn bộ bảng, nếu ô nào có số 0 thì ta tìm thành phần liên thông toàn số 0 đó. Trong lúc tìm thì đếm luôn số 0 của thành phần liên thông, rồi cập nhật lại kết quả của bài toán. Trong lúc cài đặt, để không phải xét vùng biên thì ta nên tạo thêm 4 viền xung quanh đều có chứa giá trị là 1.

Code tham khảo:

```
const fi='bai5.inp';
      fo='bai5.out';
var   a: array[0..101, 0..101] of 0..1;
      hang: array[1..4] of longint = (0,0,1,-1);
      cot: array[1..4] of longint = (-1,1,0,0);
      kt: array[0..101, 0..101] of boolean;
      i,j: longint;
      res,dem,m,n: longint;
procedure   dfs(u,v: longint);
var   i: longint;
begin
   kt[u,v]:=false;
   inc(dem);
   for i:=1 to 4 do
      if a[u+hang[i],v+cot[i]] =0 then
         if kt[u+hang[i],v+cot[i]]=true then dfs(u+hang[i],v+cot[i]);
end;
BEGIN
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    readln(m,n);
    fillchar(a, sizeof(a),1);
    for i:=1 to m do
      for j:=1 to n do read(a[i,j]);
    res:=0;
    fillchar(kt, sizeof(kt), true);
    for i:=1 to m do
      for j:=1 to n do
         if (a[i,j] = 0) and (kt[i,j] = true) then
            begin
               dem:=0;
               dfs(i,j);
               if dem > res then res:=dem
            end;
    writeln(res);
    close(input); close(output);
END.
```

6.2. Bãi cỏ

Bessie dự định cả ngày sẽ nhai cỏ xuân và ngắm nhìn cảnh xuân trên cánh đồng của nông dân John, cánh đồng này được chia thành các ô vuông nhỏ với R

(1 <= R <= 100) hàng và C (1 <= C <= 100) cột. Bessie ước gì có thể đếm được số bãi cỏ trên cánh đồng.

Mỗi bãi cỏ trên cánh đồng là các ô chứa ký tự '#' nằm kề nhau theo cạnh. Cho bản đồ của cánh đồng, hãy nói cho Bessie biết có bao nhiêu bãi cỏ trên cánh đồng.

Dữ liệu: vào từ file văn bản VBGRASS.INP

- Dòng đầu: 2 số nguyên cách nhau bởi dấu cách: R và C
- Dòng 2..R+1: Dòng i+1 mô tả hàng i của cánh đồng với C ký tự, các ký tự là '#' hoặc '.' .

Kết quả: Ghi ra file VBGRASS.OUT một số nguyên cho biết số lượng bãi cỏ trên cánh đồng.

***Ví dụ*:**

VBGRASS.INP	VBGRASS.OUT
5 6 .#.... ..#... ..#..# ...##. .#....	5

***Thuật toán*:** Bài này thực tế là đếm số thành phần liên thông. Tương tự như bài trên

***Code tham khảo*:**

```
Const   fi='';//'vbgrass.inp';
        fo='';//'vbgrass.out';
Var     i,j,r,c,dem,res,w:longint;
        a:array[0..101,0..101]of char;
        free:array[0..101,0..101]of boolean;
        hang:array[1..4]of longint = (0,0,1,-1);
        cot:array[1..4]of longint = (-1,1,0,0);
        st:string;
Procedure       DFS(u,v:longint);
var     i:longint;
begin
        free[u,v]:=false;
        For i:=1 to 4 do
            If free[u+hang[i],v+cot[i]] then dfs(u+hang[i],v+cot[i]);
end;
BEGIN
        Assign(input,fi);Reset(input);
        Assign(output,fo);Rewrite(output);
        Fillchar(free,sizeof(free),false);
        Fillchar(w,sizeof(w),0);
        Readln(r, c);
        For i:=1 to r do
                begin
                        readln(st);
                        For j:=1 to c do
                        begin
```

```
                                a[i,j]:=st[j];
                                If a[i,j]='#' then free[i,j]:=true;
                        end;
                end;
        res:=0;
          For i:=1 to r do
                  For j:=1 to c do
                          If free[i,j] then
                                  begin
                                          inc(res);
                                          dfs(i,j);
                                  end;
          Write(res);
          Close(output);
END.
```

6.3. Nguồn: http://vn.spoj.com/problems/MESSAGE/

Một lớp gồm N học sinh, mỗi học sinh cho biết những bạn mà học sinh đó có thể liên lạc được (chú ý liên lạc này là liên lạc một chiều : u có thể gửi tin tới v nhưng v thì chưa chắc đã có thể gửi tin tới u).

Thầy chủ nhiệm đang có một thông tin rất quan trọng cần thông báo tới tất cả các học sinh. Để tiết kiệm thời gian, thầy chỉ nhắn tin tới 1 số học sinh rồi sau đó nhờ các học sinh này nhắn lại cho tất cả các bạn mà các học sinh đó có thể liên lạc được, và cứ lần lượt như thế làm sao cho tất cả các học sinh trong lớp đều nhận được tin .

Yêu cầu: Hãy tìm một số ít nhất các học sinh mà thầy chủ nhiệm cần nhắn.

Dữ liệu: Vào từ file MESSAGE.INP gồm

- Dòng đầu là N, M (N <= 800, M là số lượng liên lạc 1 chiều)
- Một số dòng tiếp theo mỗi dòng gồm 2 số u , v cho biết học sinh u có thể gửi tin tới học sinh v

Kết quả: Ghi ra file MESSAGE.OUT gồm 1 dòng ghi số học sinh cần thầy nhắn tin.

Ví dụ:

MESSAGE.INP	MESSAGE.OUT
12 15 1 3 3 6 6 1 6 8 8 12 12 9 9 6 2 4	2

4 5 5 2 4 6 7 10 10 11 11 7 10 9	

Thuật toán:

+ Thực tế khi DFS hay BFS thì danh sách kề sẽ chạy nhanh hơn khi ta dùng ma trận kề hay danh sách cạnh. Nên bài này ta chuyển đồ thị từ danh sách cạnh sang danh sách kề.

+ Thuật toán chỉ là DFS tham khảo như code.

Code tham khảo:

```
Const   fi='';//'message.inp';
        fo='';//'message.out';
        maxn=1000000;
        maxm=1000001;
Type    TE=record
                u,v:longint;
        end;
Var     e:array[1..maxm]of TE;
        k:array[1..maxm] of longint;
        head:array[1..maxn+1]of longint;
        d:array[1..maxn+1]of longint;
        i,n,m,s:longint;
Procedure       chuyendoi;
var     i:longint;
begin
        Fillchar(head,sizeof(head),0);
        For i:=1 to m do inc(head[e[i].u]);
        For i:=2 to n do head[i]:=head[i-1]+head[i];
        For i:=m downto 1 do
                begin
                        k[head[e[i].u]]:=e[i].v;
                        dec(head[e[i].u]);
                end;
        head[n+1]:=m;
end;
Procedure       DFS(a:longint);
var     b:longint;
begin
        for b:=head[a]+1 to head[a+1] do
        if d[k[b]]=0 then
                begin
                    d[k[b]]:=i;
                    dfs(k[b]);
                end
            else    if d[d[k[b]]]<>0 then d[d[k[b]]]:=i;
end;
BEGIN
        Assign(input,fi);Reset(input);
        Assign(output,fo);Rewrite(output);
        Readln(n,m);
        For i:=1 to m do readln(e[i].u,e[i].v);
```

```
        chuyendoi;
        Fillchar(d,sizeof(d),0);
        s:=0;
        For i:=1 to n do If d[i]=0 then dfs(i);
        For i:=1 to n do if (d[i]=i)or(d[i]=0) then
                begin
                        //Writeln(i);
                        inc(s);
                end;
        Write(s);
END.
```

6.4. BẢO TỒN ĐỘNG VẬT HOANG DÃ

Một khu bảo tồn động vật có n địa điểm và các đường đi hai chiều nối các địa điểm đó, địa điểm thứ i có nhiệt độ là t_i, giữa hai địa điểm bất kỳ có nhiều nhất là một đường đi nối chúng.

Người ta muốn di chuyển một loài động vật quý hiếm từ địa điểm A tới địa điểm B, tuy nhiên nếu chênh lệch về nhiệt độ giữa hai địa điểm liên tiếp trên đường đi là quá cao thì loài động vật này rất có thể bị chết.

Yêu cầu: Hãy chỉ ra một hành trình mà độ lệch nhiệt độ lớn nhất giữa hai địa điểm liên tiếp bất kỳ trên đường đi là cực tiểu.

Dữ liệu: Vào từ file văn bản MOVE.INP

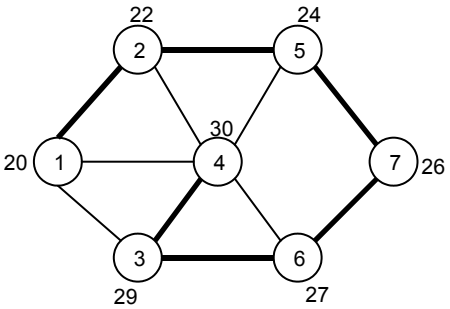
- Dòng 1: Chứa ba số n, A, B ($2 \le n \le 200$; $A \ne B$)
- Dòng 2: Chứa n số tự nhiên $t_1, t_2, ..., t_n$ ($\forall i: 0 \le t_i \le 2.10^6$)
- Các dòng tiếp theo, mỗi dòng chứa hai số nguyên dương u, v cho biết giữa hai địa điểm u và v có đường đi nối chúng.

Kết quả: Ghi ra file văn bản MOVE.OUT là độ lệch nhiệt độ lớn nhất giữa hai địa điểm liên tiếp bất kỳ trên đường đi tìm được, nếu không tồn tại đường đi thì dòng này ghi số -1.

Các số trên một dòng của Input/ Output file được ghi cách nhau ít nhất một dấu cách.

Ví dụ:

MOVE.INP	MOVE.OUT
7 1 4 20 22 29 30 24 27 26 1 2 1 3 1 4 2 4 2 5 3 4 3 6 4 5	2



4 6 5 7 6 7	

Thuật toán: chúng ta chặt nhị phân theo kết quả. Với mỗi giá trị ta DFS xem nó có thỏa mãn di chuyển được từ A đến B không. Nếu được thì ta chặt trên đoạn sau, nếu không thỏa mãn thì ta lại chặt trên đoạn đầu.

Code tham khảo:

```
const   fi='move.inp';
        fo='move.out';
        maxn= 200;
type    te = record
             u,v: longint;
             end;
var     t: array[1..maxn] of longint;
        e: array[1.. maxn*maxn] of te;
        head: array[1..maxn+1] of longint;
        ke: array[1.. 2*maxn*maxn] of longint;
        cx: array [1..maxn] of boolean;
        hh,hh1,a,b,m,n,i,j,u,v: longint;
        dau, cuoi, giua, res: longint;
procedure   dfs(u: longint);
    var v: longint;
    begin
        cx[u]:= false;
        if cx[b]= false then exit;
        for v:= head[u]+1 to head[u+1] do
            if cx[ke[v]] and (abs(t[u]- t[ke[v]])<= giua) then dfs(ke[v]);
    end;
BEGIN
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    readln(n,a,b);
    for i:= 1 to n do read(t[i]);
    m:=0;
    while not eof(input) do
      begin
        inc(m);
        readln(hh, hh1);
        if hh  = -1 then break;
        e[m].u:= hh; e[m].v:= hh1;
      end;
    fillchar(head, sizeof(head),0);
    // tinh bac cua dinh u
    for u:= 1 to m do
        begin
            inc(head[e[u].u]);
            inc(head[e[u].v]);
        end;
    // head[u] la cuoi cua nhom u
    for u:= 2 to n do   head[u]:= head[u-1]+head[u];
    for i:= 1 to m do
        begin
            ke[head[e[i].u]] := e[i].v;
            ke[head[e[i].v]] := e[i].u;
            dec(head[e[i].u]);
            dec(head[e[i].v]);
        end;
```

```
    head[n+1]:= 2*m;
    dau:= 0; cuoi:= 200000; res:=-1;
    while dau <= cuoi do
        begin
            giua:= (dau+cuoi) shr 1;
            fillchar(cx, sizeof(cx), true);
            dfs(a);
            if cx[b]= false then
                begin
                    res:= giua;
                    cuoi:=giua -1;
                end
            else dau := giua +1;
        end;
    writeln(res);
    close(input);
    close(output);
END.
```

6.5. Mê cung

Mê cung hình chữ nhật kích thước m*n gồm các ô vuông đơn vị (m, n ≤ 10^3). Trên mỗi ô ghi một trong ba kí tự:

+ O: Nếu ô đó an toàn

+ X: Nếu ô đó có cạm bẫy

+ E: Nếu là ô có một nhà thám hiểm đang đứng

Duy nhất chỉ có 1 ô ghi chữ E. Nhà thám hiểm có thể từ một ô đi sang một trong số các ô chung cạnh với ô đang đứng. Một cách đi thoát khỏi mê cung là một hành trình đi qua các ô an toàn ra một ô biên. Hãy chỉ giúp cho nhà thám hiểm một hành trình thoát ra khỏi mê cung đi qua ít ô nhất kể cả ô xuất phát.

***Dữ liệu*:** Vào từ file văn bản MECUNG.INP gồm

+ Dòng đầu là 2 số nguyên dương m, n

+ m dòng tiếp theo mỗi dòng là n ký tự O, X, E

Kết quả: Ghi ra file MECUNG.OUT là kết của bài toán. Nếu không có cách thoát hiểm thì in ra -1

Ví dụ:

MECUNG.INP	MECUNG.OUT
5 6 XXXOOX OXXXXX XOOOOX OOOXEX OXOXXX	6

Thuật toán: Bài này phải dùng BFS để làm. Chú ý trong BFS ta dùng mảng d[u,v] với ý nghĩa là đếm số ô mà đi từ ô xuất phát đến ô (u, v), hay nói cách

khác khi có một ô (x, y) là ô an toàn mà chưa bị đánh dấu kề với ô (u, v) thì ta sẽ có d[x,y] = d[u,v] +1.

Code tham khảo:

```
const   fi='mecung.inp';
        fo='mecung.out';
type ptr=record
     x,y:longint;
     end;
var     f1,f2:text;
        hx: array[1..4] of longint = (-1,0,1,0);
        hy: array[1..4] of longint = (0,1,0,-1) ;
        i,j,m,n,first,last,ex,ey,x1,y1:longint;
        kq,st1,st2:ansistring;
        t:ptr;
        cx:array[0..1001,0..1001] of boolean;
        t1:char;
        q:array[0..1000001] of ptr;
        trace,d:array[0..1001,0..1001] of longint;
procedure doc;
begin
    assign(f1,fi);
    reset(f1);
    assign(f2,fo);
    rewrite(f2);
    readln(f1,m,n);
    fillchar(cx, sizeof(cx),false);
    for i:=1 to m do
        begin
            for j:=1 to n do
                begin
                    read(f1,t1);
                    if t1='O' then
                            cx[i,j]:=true;
                    if t1='E' then
                        begin
                            cx[i,j]:=true;
                            d[i,j]:=1;
                            ex:=i;
                            ey:=j;
                        end;
                end;
            readln(f1);
        end;

end;
procedure   bfs(u,v: longint);
begin
    first:=1;
    last:=1;
    q[1].x:=u; q[1].y:=v;
    cx[u,v]:=false;
    repeat
        t:=q[first];
        inc(first);
        for i:=1 to 4 do
          if (cx[t.x+ hx[i], t.y+ hy[i]]) then
           begin
               x1:= t.x+hx[i];
               y1:= t.y+hy[i];
               cx[x1,y1]:=false;
               inc(last);
```

```
                q[last].x:=x1;
                q[last].y:=y1;
                d[x1,y1]:=d[t.x,t.y]+1;
                if (x1=1)  or (y1=1 ) or (x1=m) or (y1=n) then exit;
            end;
   until  first> last;
end;
procedure inkq;
begin
   if (x1=1)  or (y1=1 ) or (x1=m) or (y1=n) then
      writeln(f2,d[x1,y1]);
  else writeln(f2,-1);
end;
BEGIN
   doc;
   bfs(ex,ey);
   inkq;
   close(f1);
   close(f2);
END.
```

6.6. Các mã bài có thể tham khảo thêm:

http://vn.spoj.com/problems/NUMBER/

http://www.spoj.com/THPTCBT/problems/MTNTRAI/

http://www.spoj.com/PTIT/problems/BCROBOT/cstart=10

http://www.spoj.com/PTIT/problems/BCLKCOUN/

http://www.spoj.com/PTIT/problems/BCDAISY/

http://www.spoj.com/PTIT/problems/BCACM11D/

http://vn.spoj.com/problems/C11BC2/

http://www.spoj.com/PTIT/problems/BCISLAND/

http://vn.spoj.com/problems/MTWALK/

http://vn.spoj.com/problems/STABLE/

http://vn.spoj.com/problems/FLOYD/

http://vn.spoj.com/problems/QBSCHOOL/

http://vn.spoj.com/problems/BESTSPOT/

http://vn.spoj.com/problems/VDANGER/

http://vn.spoj.com/problems/GONDOR/

http://vn.spoj.com/problems/TTRIP/

http://vn.spoj.com/problems/QBBUILD/

http://vn.spoj.com/problems/CENTRE28/

http://vn.spoj.com/problems/NETACCEL/

http://vn.spoj.com/problems/TRAFFICN/

http://vn.spoj.com/problems/REVAMP/

http://vn.spoj.com/problems/QBROBOT/

http://vn.spoj.com/problems/ROADS/

http://vn.spoj.com/problems/PWRFAIL/

http://vn.spoj.com/problems/XUCXAC/

http://vn.spoj.com/problems/BINLADEN/

http://vn.spoj.com/problems/DHFRBUS/

http://vn.spoj.com/problems/V8SORT/

http://vn.spoj.com/problems/PBCWATER/

Tài liệu tham khảo

[1] Lê Minh Hoàng (2006)*, Giải thuật và lập trình*, Sách ebook.

[2] Website: *http://vn.spoj.com/* và *http://vnoi.info/*