

Bài 1: Thiếu hụt

Trong quỹ của một Công ty còn s đồng. Trong m ngày tới sẽ có n giao dịch liên quan tới thu chi: một số công việc đã hoàn thành và đối tác sẽ chuyển khoản vào quỹ của công ty một số tiền trong khoảng các ngày từ a đến b (kể cả b) hoặc phải trả một khoản tiền thanh toán cho vật tư đã mua, yêu cầu chi trả sẽ đến trong khoảng các ngày từ x đến y (kể cả y).

Khi có yêu cầu chi trả Công ty phải trả ngay trong ngày nhận được thông báo. Việc chuyển tiền vào tài khoản của Công ty có thể được thực hiện ở ngày bất kỳ trong phạm vi từ a đến b .

Nếu trong một ngày xuất hiện nhiều giao dịch – có thể thực hiện chúng theo trình tự tùy chọn.

Hãy xác định có thể xảy ra tình trạng ở một ngày nào đó Công ty không có đủ tiền đáp ứng yêu cầu chi trả hay không.

Dữ liệu: Vào từ file DEFICIENCY.INP:

- ✚ Dòng đầu tiên chứa 3 số nguyên n, m và s ($1 \leq n, m \leq 1000, 1 \leq s \leq 10^6$),
- ✚ Mỗi dòng trong n dòng sau chứa 3 số nguyên $t, d1, d2$, trong đó nếu $t > 0$ là số tiền sẽ được chuyển tới, $t < 0$ – cần chi trả, $|t| \leq 10^6, 1 \leq d1 \leq d2 \leq m$ – phạm vi ngày có thể xuất hiện giao dịch.

Kết quả: Đưa ra file văn bản DEFICIENCY.OUT thông báo **YES** nếu có khả năng xảy ra thiếu tiền hoặc **NO** trong trường hợp ngược lại.

Ví dụ:

DEFICIENCY.INP
4 3 100
100 1 2
-100 1 2
1 2 3
1 2 2

DEFICIENCY.OUT
YES

Hướng dẫn code Pascal

```
#include <bits/stdc++.h>

using namespace std;
const string filename = "DEFICIENCY";

#define int long long

int n,m,s;

int inc1[1001];
int inc2[1001];

int32_t main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    freopen( (filename + ".inp").c_str(), "r", stdin);
    freopen( (filename + ".out").c_str(), "w", stdout);

    cin >> n >> m >> s;
```

```

while(n--)
{
    int t,d1,d2;
    cin >> t >> d1 >> d2;
    if (t < 0) inc2[d1] += t;
    else inc1[d2] += t;
}
int cur = s;
for (int i=1;i<=m;i++)
{
    cur += inc2[i];
    if (cur < 0)
    {
        cout << "YES";
        return 0;
    }
    cur += inc1[i];
}
cout << "NO";
}

```

Bài 2:

Ước số chung lớn nhất của dãy số nguyên dương **A** không rỗng là số nguyên dương **d** lớn nhất đồng thời là ước của mọi số trong dãy **A**.

Cho mảng số nguyên dương a_1, a_2, \dots, a_n và số nguyên **k**.

Hãy tìm đoạn $a_i, a_{i+1}, \dots, a_{i+k-1}$ có ước số chung lớn nhất và đưa ra ước số chung đó.

Dữ liệu: Vào từ file văn bản GCD.INP:

- ✚ Dòng đầu tiên chứa số nguyên **n** và **k** ($2 \leq n \leq 5 \times 10^5, 2 \leq k \leq n$),
- ✚ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^{18}, i = 1 \div n$).

Kết quả: Đưa ra file văn bản GCD.OUT một số nguyên – ước số chung lớn nhất tìm được.

Ví dụ:

GCD.INP	GCD.OUT
10 4 2 3 4 8 12 6 12 18 4 3	6

```
const oo=1000000000000000000;
var i,n,k:longint;
    a:array[0..5000006] of int64;
function max(a,b:int64):int64;
begin
    if (a > b) then exit(a);
    exit(b);
end;
function GCD(a,b:int64):int64;
var tmp:int64;
begin
    if (a = 0) then exit(b);
    if (b = 0) then exit(a);
    while (b > 0) do
    begin
        tmp:=b;
        b:=a mod b;
        a:=tmp;
    end;
    exit(a);
end;
procedure sub;
var i:longint;
    res:int64;
begin
    res:=a[1];
    for i:=2 to n do res:=GCD(res,a[i]);
    write(res);
    halt;
end;
procedure sub1;
var i,j:longint;
    res,tmp:int64;
begin
    res:=0;
```

```

        for i:=k to n do
        begin
            tmp:=a[i];
            for j:=i - k + 1 to i - 1 do
            begin
                tmp:=GCD(tmp,a[j]);
            end;
            res:=max(res,tmp);
        end;
        write(res);
        halt;
    end;
    procedure build(id,l,r:longint);
    var
        mid:longint;
    begin
        if (l > r) then exit;
        if (l = r) then
        begin
            tree[id]:=a[l];
            exit;
        end;
        mid:=(l + r) >> 1;
        build(id * 2,l,mid);
        build(id * 2 + 1,mid + 1,r);
        tree[id]:=GCD(tree[id * 2],tree[id * 2 + 1]);
    end;
    function get(id,l,r,u,v:longint):int64;
    var
        mid:longint;
    begin
        if (l > v) or (r < u) then exit(0);
        if (u <= l) and (v >= r) then exit(tree[id]);
        mid:=(l + r) >> 1;
        exit(GCD(get(id * 2, l, mid, u, v),get(id * 2 + 1, mid + 1, r, u,
v)));
    end;
    procedure sub2;
    var
        i:longint;
        res:int64;
    begin
        build(1,1,n);
        res:=get(1,1,n,1,k);
        for i:=k + 1 to n do
            res:=max(res,get(1,1,n,i - k + 1,i));
        write(res);
        halt;
    end;
    begin
        assign(input,'GCD.INP'); reset(input);
        assign(output,'GCD.OUT'); rewrite(output);
        readln(n,k);
        for i:=1 to n do read(a[i]);
        if (n = k) then sub;
        if (n <= 5000) then sub1;
        sub2;
        close(input); close(output);
    end.

```

Bài 3:

Trong khu vực có n tỉnh, mỗi tỉnh có một trường chuyên. Giữa một số cặp 2 tỉnh (a, b) có tuyến xe tốc hành nối a với b và ngược lại. Ban Giám hiệu của một số trường chuyên có ký thỏa thuận hợp tác trao đổi kinh nghiệm với nhau. Hiện nay đã có m thỏa thuận được ký và có k tuyến tốc hành khác nhau. Giữa 2 cặp tỉnh có không quá một tuyến tốc hành.

Hàng năm một trường sẽ đăng cai tổ chức trại hè. Những trường có quan hệ hợp tác với trường đăng cai sẽ cử giáo viên và học sinh của mình tới dự nếu từ đó có thể tới tỉnh có trường đăng cai, trực tiếp hoặc qua một số tỉnh trung gian bằng xe tốc hành.

Theo thời gian, một số tuyến tốc hành mới được xác lập hòa vào mạng lưới tốc hành hiện có, một số cặp trường có thể ký thỏa thuận hợp tác, các quan hệ hợp tác cũ vẫn được giữ nguyên.

Thông tin về cặp quan hệ mới được đưa dưới dạng thông báo $F \ a \ b$ – hai trường a và b ký thỏa thuận hợp tác. Thông tin về tuyến tốc hành mới được đưa dưới dạng $T \ a \ b$ – có tuyến nối tỉnh a với tỉnh b và ngược lại.

Nếu trường đăng cai ở tỉnh v thì người ta cần biết trước sẽ có bao nhiêu trường từ các tỉnh bạn có thể tới tham dự và truy vấn sẽ có dạng $? \ v$.

Với nơi đăng cai cho trước hãy xác định số trường bạn tới dự.

Dữ liệu: Vào từ file SUM_CAMP.INP:

- ✚ Dòng đầu tiên chứa số nguyên n, m và k ($1 \leq n \leq 10^5, 0 \leq m, k \leq 10^5$),
- ✚ Mỗi dòng trong m dòng tiếp theo chứa 2 số nguyên a và b xác định 2 trường a và b có quan hệ hợp tác ($1 \leq a, b \leq n, a \neq b$), không có 2 dòng nào giống nhau,
- ✚ Mỗi dòng trong k dòng tiếp theo chứa 2 số nguyên a và b xác định giữa 2 tỉnh a và b có tuyến tốc hành ($1 \leq a, b \leq n, a \neq b$), không có 2 dòng nào giống nhau,
- ✚ Dòng tiếp theo chứa số nguyên q – số truy vấn cần xử lý ($1 \leq q \leq 10^5$),
- ✚ Mỗi dòng trong q dòng sau chứa một truy vấn thuộc một trong các dạng đã nêu.

Kết quả: Đưa ra file văn bản SUM_CAMP.OUT, với mỗi truy vấn dạng $? \ v$ – đưa ra số trường bạn tới dự.

Ví dụ:

SUM_CAMP. INP
4 2 2
1 2
1 3
1 2
1 4
5
? 1
F 4 1
? 1
T 4 3
? 1

SUM_CAMP.OUT
1
2
3



WA29 SIP 20192710 C
XVI

```
var      i,n,q,u,v,oo,m,k:longint;  
         c:char;  
         link,head,pair,dad:array[0..1000006] of longint;
```

```

procedure swap(var a,b:longint);
var    tmp:longint;
begin
    tmp:=a; a:=b; b:=tmp;
end;
procedure themcanh(u,v:longint);
begin
    inc(oo);
    link[oo]:=head[u];
    head[u]:=oo;
    pair[oo]:=v;
end;
function root(u:longint):longint;
begin
    if (dad[u] < 0) then exit(u);
    dad[u]:=root(dad[u]);
    exit(dad[u]);
end;
procedure union(u,v:longint);
begin
    if (dad[v] > dad[u]) then swap(u,v);
    dad[u]:=dad[u] + dad[v];
    dad[v]:=u;
end;
procedure ques(u:longint);
var    j,tmp:longint;
begin
    tmp:=0;
    j:=head[u];
    u:=root(u);
    while (j > 0) do
    begin
        v:=pair[j];
        j:=link[j];
        v:=root(v);
        if (u = v) then inc(tmp);
    end;
    writeln(tmp);
end;
begin
    assign(input,'SUM_CAMP.INP'); reset(input);
    assign(output,'SUM_CAMP.OUT'); rewrite(output);
    readln(n,m,k);
    for i:=1 to n do dad[i]:=-1;
    for i:=1 to m do
    begin
        readln(u,v);
        themcanh(u,v);
        themcanh(v,u);
    end;
    for i:=1 to k do
    begin
        readln(u,v);
        u:=root(u);
        v:=root(v);
        if (u = v) then continue;
        union(u,v);
    end;
end;

```

```

        readln(q);
        for q:=1 to q do
        begin
            read(c);
            if (c = '?') then
            begin
                readln(u);
                ques(u);
            end else
            if (c = 'F' ) then
            begin
                readln(u,v);
                themcanh(u,v);
                themcanh(v,u);
            end else
            begin
                readln(u,v);
                u:=root(u);
                v:=root(v);
                if (u = v) then continue;
                union(u,v);
            end;
        end;
    end;
    close(input); close(output);
end.

```

code c++

```

#include <bits/stdc++.h>

using namespace std;
const string filename = "SUM_CAMP";

typedef pair<int,int> ii;

int val[100001];
int par[100001];
int sz[100001];
set<int> s[100001];
vector<int> AdjList[100001];
int n,m,k;
int q;

int anc(int x)
{
    if (par[x] == x) return x;
    return par[x] = anc(par[x]);
}

void join(int x,int y)
{
    int ax = anc(x);
    int ay = anc(y);
    if (sz[ax] < sz[ay])
        swap(ax,ay);
    sz[ax] += sz[ay];
    par[ay] = ax;
    for (set<int>::iterator it2 = s[ay].begin(); it2 != s[ay].end(); it2++)
    {
        int v = *it2;
    }
}

```

```

        for (int i=0;i<AdjList[v].size();i++)
        {
            int u = AdjList[v][i];
            set<int>::iterator it1 = s[ax].find(u);
            if (it1 != s[ax].end())
            {
                u = *it1;
                val[u]++;
                val[v]++;
            }
        }
    }
    for (set<int>::iterator it = s[ay].begin();it != s[ay].end();it++)
        s[ax].insert(*it);
    return;
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    freopen( (filename + ".inp").c_str(), "r", stdin);
    freopen( (filename + ".out").c_str(), "w", stdout);

    cin >> n >> m >> k;
    for (int i=1;i<=n;i++)
    {
        s[i].insert(i);
        par[i] = i;
        sz[i] = 1;
    }
    for (int i=1;i<=m;i++)
    {
        int u,v;
        cin >> u >> v;
        AdjList[u].push_back(v);
        AdjList[v].push_back(u);
    }
    for (int i=1;i<=k;i++)
    {
        int u,v;
        cin >> u >> v;
        if (anc(u) != anc(v))
            join(u,v);
    }
    cin >> q;
    while(q--)
    {
        char type;
        cin >> type;
        if (type == '?')
        {
            int u;
            cin >> u;
            cout << val[u] << '\n';
        }
        else if (type == 'F')

```



```

{
    int u,v;
    cin >> u >> v;
    AdjList[u].push_back(v);
    AdjList[v].push_back(u);
    if (anc(u) == anc(v))
    {
        val[u]++;
        val[v]++;
    }
}
else if (type == 'T')
{
    int u,v;
    cin >> u >> v;
    if (anc(u) != anc(v))
        join(u,v);
}
}
}

```

Bài 4: Vác tre

Để chuẩn bị làm một cây đàn T'rung cần tìm được cây tre hoặc vầu cái, có đốt dài và đều, dáng thuần, không có lỗi như nứt, lõm sâu, . . . Tóm lại, tìm được một cây ưng ý là rất khó. Một nghệ nhân, khi vào rừng hái lá thuốc tìm một cây tre rất ưng ý, nhưng đường đi ra phải qua một hẻm núi độ rộng n và dài m bước chân.



Có thể coi hẻm núi như lưới ô vuông $n \times m$, một số ô có đá cao. Cây tre vác trên vai nên khi đi nó sẽ song song với một cạnh của hẻm núi. Cần phải đi từ trái sang phải. Có thể vào hẻm từ ô bất kỳ bên trái với cây tre song song cạnh trái. Khi cây tre đang ở hướng dọc, chỉ có thể di chuyển dọc lên trên hoặc xuống dưới, nếu cây tre đang ở hướng ngang – chỉ có thể di chuyển sang trái hoặc phải. Trong trường hợp cần chuyển hướng có thể chống một đầu cây tre xuống đất, dựng thẳng đứng và cho đổ sang hướng song song với cạnh kia. Dĩ nhiên tre phải được nằm trên các ô không có đá cao. Để ra khỏi hẻm núi phải tới được ô ở cạnh phải với cây tre nằm song song cạnh này. Cần chặt cây tre thành các đoạn ngắn để mang về. Các đoạn đó phải càng dài càng tốt để có nhiều lựa chọn cắt khúc khi làm đàn.

Trạng thái hẻm núi được thể hiện bằng bảng ký tự kích thước $n \times m$, ký tự '.' chỉ vị trí trống, '#' chỉ ô có đá.

Hãy xác định độ dài lớn nhất của đoạn tre có thể mang qua (đơn vị độ dài là cạnh của ô).

Dữ liệu: Vào từ file INPUT chuẩn của hệ thống:

- ➦ Dòng đầu tiên chứa 2 số nguyên n và m ($1 \leq n, m \leq 300$),
- ➦ Mỗi dòng trong n dòng sau chứa xâu s độ dài m mô tả trạng thái một dòng của hẻm núi.

Kết quả: Đưa ra file OUTPUT chuẩn của hệ thống một số nguyên – độ dài lớn nhất của đoạn tre có thể mang qua.

Ví dụ:

INPUT	OUTPUT
3 5 ...## .#.#. ##...	2

Tham khảo giải thuật

```
#include <bits/stdc++.h>

using namespace std;
const string filename = "BAMBOO";

typedef pair<int,int> ii;
typedef pair<int,ii> iii;

int n,m;
char a[311][311];
int go[311][311][5];
bool visited[311][311][5];
int dx[5] = {0,-1,0,1,0};
int dy[5] = {0,0,1,0,-1};

void init()
{
```

```

for (int i=1;i<=n;i++)
    for (int j=1;j<=m;j++)
        if (a[i][j] == '.')
            for (int k=1;k<=4;k++)
            {
                go[i][j][k] = 1;
                int xx = i + dx[k];
                int yy = j + dy[k];
                while(1 <= xx && xx <= n && 1 <= yy && yy <= m &&
a[xx][yy] == '.')
                {
                    xx += dx[k];
                    yy += dy[k];
                    go[i][j][k]++;
                }
            }
}

bool check(int z)
{
    //cerr << z << '\n';
    for (int i=1;i<=n;i++)
        for (int j=1;j<=m;j++)
            for (int k=1;k<=4;k++)
                visited[i][j][k] = false;
    queue<iii> CurrList;
    for (int i=1;i<=n;i++)
    {
        //cerr << i << ' ' << go[i][1][3] << ' ' << go[i][1][1] << '\n';
        if (go[i][1][3] >= z)
        {
            CurrList.push(iii(3,ii(i,1)));
            visited[i][1][3] = true;
        }
        if (go[i][1][1] >= z)
        {
            CurrList.push(iii(1,ii(i,1)));
            visited[i][1][1] = true;
        }
    }
    while(!CurrList.empty())
    {
        int t = CurrList.front().first;
        int x = CurrList.front().second.first;
        int y = CurrList.front().second.second;
        //cerr << x << ' ' << y << ' ' << t << '\n';
        CurrList.pop();
        if (y == m && (t == 1 || t == 3)) return true;
        // Here comes the dumb shit
        if (t == 1)
        {
            if (x - z >= 1 && a[x - z][y] == '.' && !visited[x-1][y][t])
            {
                visited[x-1][y][t] = true;
                CurrList.push(iii(t,ii(x-1,y)));
            }
            if (x + 1 <= n && a[x + 1][y] == '.' && !visited[x+1][y][t])
            {
                visited[x+1][y][t] = true;
            }
        }
    }
}

```

```

        CurrList.push(III(t, II(x+1, y)));
    }
    if (go[x-z+1][y][2] >= z && !visited[x-z+1][y][2])
    {
        visited[x-z+1][y][2] = true;
        CurrList.push(III(2, II(x-z+1, y)));
    }
    if (go[x-z+1][y][4] >= z && !visited[x-z+1][y][4])
    {
        visited[x-z+1][y][4] = true;
        CurrList.push(III(4, II(x-z+1, y)));
    }
    if (go[x][y][2] >= z && !visited[x][y][2])
    {
        visited[x][y][2] = true;
        CurrList.push(III(2, II(x, y)));
    }
    if (go[x][y][4] >= z && !visited[x][y][4])
    {
        visited[x][y][4] = true;
        CurrList.push(III(4, II(x, y)));
    }
}
else if (t == 2)
{
    if (y + z <= m && a[x][y + z] == '.' && !visited[x][y+1][t])
    {
        visited[x][y+1][t] = true;
        CurrList.push(III(t, II(x, y+1)));
    }
    if (y - 1 >= 1 && a[x][y - 1] == '.' && !visited[x][y-1][t])
    {
        visited[x][y-1][t] = true;
        CurrList.push(III(t, II(x, y-1)));
    }
    if (go[x][y+z-1][1] >= z && !visited[x][y+z-1][1])
    {
        visited[x][y+z-1][1] = true;
        CurrList.push(III(1, II(x, y+z-1)));
    }
    if (go[x][y+z-1][3] >= z && !visited[x][y+z-1][3])
    {
        visited[x][y+z-1][3] = true;
        CurrList.push(III(3, II(x, y+z-1)));
    }
    if (go[x][y][1] >= z && !visited[x][y][1])
    {
        visited[x][y][1] = true;
        CurrList.push(III(1, II(x, y)));
    }
    if (go[x][y][3] >= z && !visited[x][y][3])
    {
        visited[x][y][3] = true;
        CurrList.push(III(3, II(x, y)));
    }
}
else if (t == 3)
{
    if (x + z <= n && a[x + z][y] == '.' && !visited[x+1][y][t])

```

```

{
    visited[x+1][y][t] = true;
    CurrList.push(III(t,ii(x+1,y)));
}
if (x - 1 >= 1 && a[x - 1][y] == '.' && !visited[x-1][y][t])
{
    visited[x-1][y][t] = true;
    CurrList.push(III(t,ii(x-1,y)));
}
if (go[x+z-1][y][2] >= z && !visited[x+z-1][y][2])
{
    visited[x+z-1][y][2] = true;
    CurrList.push(III(2,ii(x+z-1,y)));
}
if (go[x+z-1][y][4] >= z && !visited[x+z-1][y][4])
{
    visited[x+z-1][y][4] = true;
    CurrList.push(III(4,ii(x+z-1,y)));
}
if (go[x][y][2] >= z && !visited[x][y][2])
{
    visited[x][y][2] = true;
    CurrList.push(III(2,ii(x,y)));
}
if (go[x][y][4] >= z && !visited[x][y][4])
{
    visited[x][y][4] = true;
    CurrList.push(III(4,ii(x,y)));
}
}
else if (t == 4)
{
    if (y - z >= 1 && a[x][y - z] == '.' && !visited[x][y-1][t])
    {
        visited[x][y-1][t] = true;
        CurrList.push(III(t,ii(x,y-1)));
    }
    if (y + 1 <= m && a[x][y + 1] == '.' && !visited[x][y+1][t])
    {
        visited[x][y+1][t] = true;
        CurrList.push(III(t,ii(x,y+1)));
    }
    if (go[x][y-z+1][1] >= z && !visited[x][y-z+1][1])
    {
        visited[x][y-z+1][1] = true;
        CurrList.push(III(1,ii(x,y-z+1)));
    }
    if (go[x][y-z+1][3] >= z && !visited[x][y-z+1][3])
    {
        visited[x][y-z+1][3] = true;
        CurrList.push(III(3,ii(x,y-z+1)));
    }
    if (go[x][y][1] >= z && !visited[x][y][1])
    {
        visited[x][y][1] = true;
        CurrList.push(III(1,ii(x,y)));
    }
    if (go[x][y][3] >= z && !visited[x][y][3])
    {

```

```

        visited[x][y][3] = true;
        CurrList.push(III(3,II(x,y)));
    }
}
return false;
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    freopen( (filename + ".inp").c_str(), "r", stdin);
    freopen( (filename + ".out").c_str(), "w", stdout);

    cin >> n >> m;
    for (int i=1;i<=n;i++)
    {
        string s;
        cin >> s;
        for (int j=0;j<s.size();j++)
            a[i][j+1] = s[j];
    }
    init();
    if (m == 1) // special cases
    {
        int res = 0;
        for (int i=1;i<=n;i++)
            res = max(res,go[i][1][3]);
        cout << res;
        return 0;
    }
    //cerr << "DONE INIT\n";
    int l = 0, r = min(n,m);
    int ans = 0;
    while (l <= r) // Binary Search to optimize time i guess :P
    {
        int mi = (l+r)/2;
        if (check(mi))
        {
            ans = mi;
            l = mi + 1;
        }
        else
            r = mi - 1;
    }
    cout << ans;
}



```

Bài 5: Dãy chứa max

Xét dãy số nguyên $\mathbf{A} = (a_1, a_2, \dots, a_n)$. Dãy chứa các phần tử ở các vị trí liên tiếp của \mathbf{A} được gọi là dãy con. Hai dãy con được gọi là khác nhau nếu tồn tại ít nhất một vị trí mà phần tử của \mathbf{A} ở vị trí đó tham gia vào dãy con này và không tham gia vào dãy con kia.

Cho số nguyên b . Hãy xác định số lượng dãy con có giá trị lớn nhất của các phần tử trong dãy con bằng b .

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

-  Dòng đầu tiên chứa số nguyên n và b ($2 \leq n \leq 10^5$, $1 \leq b \leq 10^9$),
-  Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^9$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản NUMMAX.OUT: một số nguyên – số lượng dãy con tìm được.

Ví dụ:

NUMMAX.INP	NUMMAX.OUT
4 5 1 3 5 2	6

```
#include <bits/stdc++.h>

using namespace std;
const string filename = "NUMMAX";

#define int long long

int n,b;
int a[100011];
int l[100011];
int r[100011];

int32_t main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    freopen( (filename + ".inp").c_str(), "r", stdin);
    freopen( (filename + ".out").c_str(), "w", stdout);

    cin >> n >> b;
    for (int i=1;i<=n;i++)
        cin >> a[i];
    a[0] = 2e9;
    a[n+1] = 2e9;
    deque<int> dq;
    dq.push_back(0);
    for (int i=1;i<=n;i++)
    {
        while(!dq.empty() && a[dq.back()] <= a[i])
            dq.pop_back();
        l[i] = dq.back();
        dq.push_back(i);
    }
    dq.clear();
}
```

```

dq.push_back(n+1);
for (int i=n;i>=1;i--)
{
    while(!dq.empty() && a[dq.back()] < a[i])
        dq.pop_back();
    r[i] = dq.back();
    dq.push_back(i);
}
int res = 0;
for (int i=1;i<=n;i++)
    if (a[i] == b)
        res += (i - l[i])*(r[i] - i);
cout << res;
}

```


Bài 6: Đội hình thi đấu

Các môn thể thao phối hợp đòi hỏi vận động viên phải có sức mạnh và sự dẻo dai. Câu lạc bộ thể thao của nhà trường có n bạn tham gia, người thứ i có sức mạnh a_i và độ dẻo dai là b_i , $i = 1 \div n$.

Đội thi đấu có k người. Trong đội sẽ có một đội trưởng, những người còn lại là thành viên. Tiềm năng của đội được đánh giá bằng tổng sức mạnh của đội trưởng với độ dẻo dai của các thành viên.

Để chuẩn bị đấu giao hữu với trường bạn, huấn luyện viên quyết định sẽ đưa ra đội hình có tiềm năng thấp nhất, chủ yếu là tạo điều kiện cho mọi người có dịp cọ xát với thực tế, đồng thời cũng thử nghiệm các chiến thuật thi đấu.

Ví dụ, với $n = 4$, sức mạnh và độ dẻo dai của mọi người tương ứng là $A = (3, 7, 1, 6)$ và $B = (6, 3, 8, 5)$. Nếu $k = 3$ thì để có đội hình tiềm năng thấp nhất cần chọn các người 2, 3, 4 và chỉ định người 3 làm đội trưởng. Khi đó tiềm năng của đội sẽ là $1+3+5 = 9$ – thấp nhất có thể!

Do không biết trước lần này cần phải cử bao nhiêu người đi nên huấn luyện viên phải lên phương án cho mọi khả năng với k từ 1 đến n .

Hãy đưa ra tiềm năng thấp nhất của đội được cử đi với k lần lượt nhận giá trị từ 1 đến n .

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

- Đòng đầu tiên chứa số nguyên n ($1 \leq n \leq 2 \times 10^5$),
- Đòng thứ i trong n dòng sau chứa 2 số nguyên a_i và b_i ($1 \leq a_i, b_i \leq 10^9$).

Kết quả: Đưa ra file văn bản TEAM.OUT n số nguyên – các tiềm năng thấp nhất tính được, mỗi số trên một dòng, số thứ i ứng với trường hợp $k = i$.

Ví dụ:

TEAM.INP	TEAM.OUT
4	1
3 6	4
7 3	9
1 8	15
6 5	

Hướng dẫn giải thuật:

```
#include <bits/stdc++.h>

using namespace std;
const string filename = "TEAM";

#define int long long

typedef pair<int,int> ii;

bool cmp (ii a, ii b)
{
    if (a.second != b.second)
```

```

        return a.second < b.second;
    return a.first < b.first;
}
int n;
ii a[200011];
int psum[200011];
int mn_a_suf[200011];
int mn_c_pre[200011]; // a - b

int32_t main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    freopen( (filename + ".inp").c_str(), "r", stdin);
    freopen( (filename + ".out").c_str(), "w", stdout);

    cin >> n;
    for (int i=1;i<=n;i++)
        cin >> a[i].first >> a[i].second;
    sort(a+1,a+1+n,cmp);
    mn_c_pre[0] = 1e18;
    for (int i=1;i<=n;i++)
    {
        psum[i] = psum[i-1] + a[i].second;
        mn_c_pre[i] = min(mn_c_pre[i-1],a[i].first - a[i].second);
    }
    mn_a_suf[n+1] = 1e18;
    for (int i=n;i>=1;i--)
        mn_a_suf[i] = min(mn_a_suf[i+1],a[i].first);
    for (int k=1;k<=n;k++)
    {
        if (k == n)
        {
            cout << psum[n] + mn_c_pre[n];
            break;
        }
        int x = psum[k-1] + mn_a_suf[k];
        int y = psum[k] + mn_c_pre[k-1];
        cout << min(x,y) << '\n';
    }
}

```

Bài 7: Cặp điểm

Với nhiều học sinh hình học thường mang lại nỗi khiếp sợ vô hình. Để chứng minh rằng cái đáng sợ là cấu trúc dữ liệu và giải thuật chứ không phải hình học thầy giáo ra một bài có nội dung hình học: Cho n điểm trên trục hoành, điểm thứ i có tọa độ $(x_i, 0)$ và n điểm trên trục tung, điểm thứ i có tọa độ $(0, y_i)$, $i = 1 \div n$. Tất cả các điểm đều có tọa độ nguyên và không có điểm nào trùng với gốc tọa độ. Khi nối một điểm trên trục hoành với một điểm trên trục tung ta có một đoạn thẳng.

Hãy xác định có bao nhiêu cách nối mỗi điểm trên trục hoành với một điểm trên trục tung sao cho không có hai đoạn thẳng nào cắt nhau và đưa ra theo mô đun 998244353.

Dữ liệu vào từ file văn bản POINTS.INP:

- 🚩 Dòng đầu tiên chứa số nguyên n ($1 \leq n \leq 10^5$),
- 🚩 Dòng thứ 2 chứa n số nguyên x_1, x_2, \dots, x_n ($-10^9 \leq x_1 < x_2 < \dots < x_n \leq 10^9$),
- 🚩 Dòng thứ 3 chứa n số nguyên y_1, y_2, \dots, y_n ($-10^9 \leq y_1 < y_2 < \dots < y_n \leq 10^9$).

Kết quả: Đưa ra file văn bản POINTS.OUT số nguyên tính được.

Ví dụ:

POINTS.INP	POINTS.OUT
2 -1 1 -1 2	2

Hướng dẫn cài đặt:

```
#include <bits/stdc++.h>

using namespace std;
const string filename = "POINTS";

#define int long long

const int mod = 998244353;

int fact[100001];
int re_fact[100001];

int p(int a, int b)
{
    if (b == 0)
        return 1;
    int tmp = p(a, b/2);
    if (b%2 == 0) return (tmp * tmp)%mod;
    return ((tmp * tmp)%mod * a)%mod;
}

int inv(int x)
{
    return p(x, mod - 2);
}

int comb(int n, int k)
{
    }
```

```

        if (k > n)
            return 0;
        return ((fact[n] * re_fact[k])%mod * re_fact[n-k])%mod;
    }

int n;
int cnt_x_pos = 0;
int cnt_y_pos = 0;
int cnt_x_neg = 0;
int cnt_y_neg = 0;

int32_t main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    freopen( (filename + ".inp").c_str(), "r", stdin);
    freopen( (filename + ".out").c_str(), "w", stdout);

    fact[0] = 1;
    for (int i=1;i<=100000;i++)
        fact[i] = (fact[i-1] * i)%mod;
    re_fact[100000] = inv(fact[100000]);
    for (int i=100000;i>=1;i--)
        re_fact[i-1] = (re_fact[i] * i)%mod;
    cin >> n;
    for (int i=1;i<=n;i++)
    {
        int x; cin >> x;
        if (x > 0) cnt_x_pos++;
        else cnt_x_neg++;
    }
    for (int i=1;i<=n;i++)
    {
        int y; cin >> y;
        if (y > 0) cnt_y_pos++;
        else cnt_y_neg++;
    }
    int res = 0;
    for (int i=0;i<=cnt_x_pos;i++)
    {
        int cur = 1;
        cur = (cur * comb(cnt_x_pos,i))%mod;
        cur = (cur * comb(cnt_y_pos,i))%mod;
        if (cnt_y_pos - i < 0) continue;
        if (cnt_x_pos - i < 0) continue;
        cur = (cur * comb(cnt_x_neg,cnt_y_pos - i))%mod;
        cur = (cur * comb(cnt_y_neg,cnt_x_pos - i))%mod;
        res = (res + cur)%mod;
    }
    cout << res;
}

```

Bài 8: Tam giác

Alice có 3 thanh nhựa độ dài tương ứng là a , b và c . Alice định lắp một hình tam giác diện tích khác 0 có cạnh là các thanh nói trên, mỗi thanh là một cạnh.

Nếu không thể làm điều đó từ các thanh ban đầu thì Alice có thể hơi nóng và kéo dài một hoặc vài thanh để lắp. Cứ mỗi phút Alice kéo dài thanh được chọn thêm 1cm.

Hãy xác định thời gian tối thiểu cần thiết cho việc kéo dài thanh để lắp được tam giác.

Dữ liệu: Vào từ file văn bản TRIANGLE.INP gồm một dòng chứa 3 số nguyên a, b, c ($1 \leq a, b, c \leq 10^9$).

Kết quả: Đưa ra file văn bản TRIANGLE.OUT một số nguyên – thời gian tối thiểu (tính theo số phút) cần thiết cho việc kéo dài thanh để lắp được tam giác.

Ví dụ:

TRIANGLE.INP
100 10 10

TRIANGLE.OUT
81

```
#include <bits/stdc++.h>
using namespace std;
const string filename = "TRIANGLE";
#define int long long

int a[4];

int32_t main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    freopen( (filename + ".inp").c_str(), "r", stdin);
    freopen( (filename + ".out").c_str(), "w", stdout);

    cin >> a[1] >> a[2] >> a[3];
    sort(a+1, a+1+3);
    int res = 0;
    if (a[1] == 0) a[1] = 1, res++;
    if (a[2] == 0) a[2] = 1, res++;
    if (a[3] == 0) a[3] = 1, res++;
    if (a[1] + a[2] <= a[3])
        res += (a[3] + 1) - (a[1] + a[2]);
    cout << res;
}
```

Bài 9: Mex

Trong lý thuyết trò chơi hàm **mex** đóng vai trò quan trọng. Hàm **mex** được định nghĩa như sau: Cho tập số nguyên dương **A**. **mex(A)** là số nguyên dương nhỏ nhất không có trong tập **A**. Ví dụ, với **A** = {2, 1, 3, 5, 100}, **mex(A)** = 4, với **A** = {2, 3, 4, 5}, **mex(A)** = 1.

Alice rất thích thú với vai trò và ứng dụng của hàm **mex**. Sẵn có trong tay dãy số nguyên dương **A** = (**a**₁, **a**₂, ..., **a**_{*n*}), trong đó các số khác nhau từng đôi một, Alice quyết định thực hiện **k** lần phép bổ sung **mex** vào dãy, mỗi lần đưa thêm vào **A** số **mex** tìm được và làm tăng số phần tử của dãy lên 1.

Hãy xác định giá trị của phần tử cuối cùng được bổ sung vào dãy.

Dữ liệu: Vào từ file văn bản MEX.INP:

- 🚩 Dòng đầu tiên chứa 2 số nguyên **n** và **k** ($1 \leq n \leq 10^5$, $1 \leq k \leq 10^9$),
- 🚩 Dòng thứ 2 chứa **n** số nguyên khác nhau **a**₁, **a**₂, ..., **a**_{*n*} ($1 \leq a_i \leq 10^5$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản MEX.OUT một số nguyên – giá trị số cuối cùng được bổ sung vào dãy.

Ví dụ:

MEX.INP
7 10
1 3 20 2 7 45 5

MEX.OUT
15

```
#include <bits/stdc++.h>

using namespace std;
const string filename = "MEX";

int n, k;
int a[100001];

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    freopen( (filename + ".inp").c_str(), "r", stdin);
    freopen( (filename + ".out").c_str(), "w", stdout);

    cin >> n >> k;
    for (int i=1; i<=n; i++)
        cin >> a[i];
    sort(a+1, a+1+n);
    for (int i=1; i<=n; i++)
    {
        if (k > (a[i] - a[i-1] - 1))
            k -= (a[i] - a[i-1] - 1);
        else
        {
            cout << a[i-1] + k;
            return 0;
        }
    }
```

```
    }  
    cout << a[n] + k;  
    return 0;  
}
```




Bài 10: Phương trình

Việc giải phương trình đại số Boolean với sự tham gia của các phép tính lô gic và xử lý bit khác khá xa với việc giải các phương trình đại số thông thường.



Để chứng minh cho điều đó, thầy giáo yêu cầu cả lớp về nhà xác định số lượng nghiệm không âm của từng phương trình trong số t phương trình, mỗi phương trình có dạng:

$$a - (a^x) - x = 0$$

trong đó:

-  x là ẩn số,
-  $0 \leq a < 2^{30} - 1$,
-  \wedge là phép tính XOR.

Dữ liệu: Vào từ file văn bản EQUATION.INP

-  Dòng đầu tiên chứa số nguyên t ($1 \leq t \leq 1\,000$),
-  Mỗi dòng trong t dòng sau chứa số nguyên a ($0 \leq a < 2^{30}$).

Kết quả: Đưa ra file văn bản EQUATION.OUT t số nguyên, mỗi số trên một dòng, số thứ i xác định số lượng nghiệm không âm của phương trình thứ i , $i = 1 \div t$.

Ví dụ:

EQUATION.INP
3
0
2
1073741823

EQUATION.OUT
1
2
1073741824

```
#include <bits/stdc++.h>

using namespace std;
const string filename = "EQUATION";

int t;
int a;

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    freopen( (filename + ".inp").c_str(), "r", stdin);
    freopen( (filename + ".out").c_str(), "w", stdout);

    cin >> t;
    while(t-->0)
    {
        cin >> a;
        cout << (1<<__builtin_popcount(a)) << '\n';
    }
}
```


Bài 11: Coca

Giữa giờ nghỉ chuyển tiết học Alice và Tom tới ô tô mát bán nước giải khát để mua Cô ca. Không may trong máy không còn một lon Cô ca nào. Hai bạn quyết định chạy ra phố mua và dĩ nhiên chỉ cần một người đi là đủ. Trời nắng gắt và ai cũng ngại đi. Hai bạn quyết định chơi một trò chơi nhỏ và ai thua sẽ phải đi.

Trong tay Alice đang có một băng giấy gồm các ô vuông, mỗi ô được tô một trong 2 màu Đỏ (**R**) hoặc Xanh (**B**). Độ rộng băng giấy bằng độ rộng ô vuông. Hai người lần lượt cắt băng thành các đoạn độ dài (*số ô trên đoạn*) lớn hơn 0

Quy tắc chơi là hai người lần lượt đi, ai đến lượt mình đi chọn một đoạn có ô ở hai đầu khác màu nhau và cắt đoạn đó ở vị trí tùy chọn để nhận được 2 đoạn mỗi đoạn có độ dài lớn hơn 0.

Ai đến lượt đi nhưng không thể chọn được đoạn để cắt là thua và phải đi mua Cô ca.

Alice đi trước.

Cho trạng thái băng giấy. Hãy xác định Alice sẽ thắng hay thua và đưa ra thông báo tương ứng là Win hoặc Lose với giả thiết cả hai đều biết cách đi tối ưu.

Dữ liệu: vào từ file văn bản COCA.INP gồm một dòng chứa xâu s mô tả trạng thái ban đầu của băng giấy, s chỉ chứa các ký tự trong tập {**R**, **B**}, độ dài không vượt quá 10^5 .

Kết quả: Đưa ra file văn bản COCA.OUT thông báo tương ứng tìm được.

Ví dụ:

COCA.INP	COCA.OUT
RBRB	Win

```
#include <bits/stdc++.h>

using namespace std;
const string filename = "COCA";

string s;

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

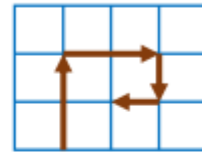
    freopen( (filename + ".inp").c_str(), "r", stdin);
    freopen( (filename + ".out").c_str(), "w", stdout);

    cin >> s;
    if (s[0] == s.back()) cout << "LOSE";
    else cout << "WIN";
}
```

Bài 12: Kéo cắt giấy

Để trang trí phòng phục vụ tổ chức sinh nhật cho một người bạn Alice lấy một tờ giấy màu thủ công kẻ ô vuông kích thước $n \times m$ (n hàng và m cột), cắt thành hình lò xo xoắn theo hướng phải sang trái và có độ rộng của đường bằng 1:

- ✎ Bắt đầu từ biên phải cột 0 cắt lên trên cho đến khi cách lề trên một ô,
- ✎ Cắt sang phải theo đường biên dưới cho đến khi cách lề phải một ô,
- ✎ Cắt xuống dưới, rồi sang trái, sau đó lên trên, . . . để có băng giấy độ rộng 1 ô,
- ✎ Quá trình cắt sẽ dừng khi không cách cắt tiếp mà không làm đứt băng giấy.



Hãy tính tổng độ dài đường cắt theo đơn vị ô.

Dữ liệu: Vào từ file SCISSOR.INP gồm một dòng chứa 2 số nguyên n và m ($2 \leq n, m \leq 10^9$).

Kết quả: Đưa ra file văn bản SCISSOR.OUT một số nguyên – độ dài đường cắt.

Ví dụ:

SCISSOR.INP
3 4

SCISSOR.OUT
6

```
#include <bits/stdc++.h>

using namespace std;
const string filename = "SCISSOR";

#define int long long

int n,m;

int32_t main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    freopen( (filename + ".inp").c_str(), "r", stdin);
    freopen( (filename + ".out").c_str(), "w", stdout);

    cin >> n >> m;
    int x,y;
    int cnt_x = m-1;
    int tmp = n-1;
    x = tmp * (tmp + 1) / 2;
    if (tmp > cnt_x)
    {
        int z = (tmp - cnt_x);
        x -= z * (z + 1) / 2;
    }
    int cnt_y = n-1;
    tmp = m-2;
    y = tmp * (tmp + 1) / 2;
    if (tmp > cnt_y)
```

```
{
    int z = (tmp - cnt_y);
    y -= z * (z + 1) / 2;
}
cout << x + y;
```

```
}
```

Bài 12: Tập lớn nhất

Cho dãy số nguyên dương $\mathbf{A} = (a_1, a_2, \dots, a_n)$. Hãy tìm tập nhiều phần tử nhất có cùng ít nhất một ước số chung lớn hơn 1 và đưa ra số phần tử trong tập tìm được.

Ví dụ, với $\mathbf{A} = (6, 15, 10, 42)$, tập $\{6, 10, 42\}$ chứa các số cùng chia hết cho 2 và là tập nhiều phần tử nhất có cùng ít nhất một ước số chung lớn hơn 1. Số lượng các phần tử trong tập là 3.

Dữ liệu: Vào từ file MAX_SET.INP:

- 🚩 Dòng đầu tiên chứa số nguyên n ($1 \leq n \leq 1000$),
- 🚩 Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($2 \leq a_i \leq 10^{18}$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản MAX_SET.OUT một số nguyên – số lượng phần tử của tập tìm được.

Ví dụ:

MAX_SET.INP
4
6 15 10 42

MAX_SET.OUT
3

```
#include <bits/stdc++.h>

using namespace std;
const string filename = "MAX_SET";

typedef long long ll;
#define MAXL (50000>>5)+1
#define GET(x) (mark[x>>5]>>(x&31)&1)
#define SET(x) (mark[x>>5] |= 1<<(x&31))
int mark[MAXL];
int P[50000], Pt = 0;
void sieve() {
    register int i, j, k;
    SET(1);
    int n = 46340;
    for (i = 2; i <= n; i++) {
        if (!GET(i)) {
            for (k = n/i, j = i*k; k >= i; k--, j -= i)
                SET(j);
            P[Pt++] = i;
        }
    }
}

long long mul(unsigned long long a, unsigned long long b, unsigned long long mod)
{
    long long ret = 0;
    for (a %= mod, b %= mod; b != 0; b >>= 1, a <= 1, a = a >= mod ? a - mod :
a) {
        if (b&1) {
            ret += a;
            if (ret >= mod) ret -= mod;
        }
    }
    return ret;
}
```

```

void exgcd(long long x, long long y, long long &g, long long &a, long long &b) {
    if (y == 0)
        g = x, a = 1, b = 0;
    else
        exgcd(y, x%y, g, b, a), b -= (x/y) * a;
}

long long llgcd(long long x, long long y) {
    if (x < 0) x = -x;
    if (y < 0) y = -y;
    if (!x || !y) return x + y;
    long long t;
    while (x%y)
        t = x, x = y, y = t%y;
    return y;
}

long long inverse(long long x, long long p) {
    long long g, b, r;
    exgcd(x, p, g, r, b);
    if (g < 0) r = -r;
    return (r%p + p)%p;
}

long long mpow(long long x, long long y, long long mod) { // mod < 2^32
    long long ret = 1;
    while (y) {
        if (y&1)
            ret = (ret * x)%mod;
        y >>= 1, x = (x * x)%mod;
    }
    return ret % mod;
}

long long mpow2(long long x, long long y, long long mod) {
    long long ret = 1;
    while (y) {
        if (y&1)
            ret = mul(ret, x, mod);
        y >>= 1, x = mul(x, x, mod);
    }
    return ret % mod;
}

int isPrime(long long p) { // implements by miller-babin
    if (p < 2 || !(p&1)) return 0;
    if (p == 2) return 1;
    long long q = p-1, a, t;
    int k = 0, b = 0;
    while (!(q&1)) q >>= 1, k++;
    for (int it = 0; it < 2; it++) {
        a = rand()%(p-4) + 2;
        t = mpow2(a, q, p);
        b = (t == 1) || (t == p-1);
        for (int i = 1; i < k && !b; i++) {
            t = mul(t, t, p);
            if (t == p-1)
                b = 1;
        }
        if (b == 0)
            return 0;
    }
    return 1;
}

```

```

long long pollard_rho(long long n, long long c) {
    long long x = 2, y = 2, i = 1, k = 2, d;
    while (true) {
        x = (mul(x, x, n) + c);
        if (x >= n) x -= n;
        d = llgcd(x - y, n);
        if (d > 1) return d;
        if (++i == k) y = x, k <= 1;
    }
    return n;
}

void factorize(int n, vector<long long> &f) {
    for (int i = 0; i < Pt && P[i]*P[i] <= n; i++) {
        if (n%P[i] == 0) {
            while (n%P[i] == 0)
                f.push_back(P[i]), n /= P[i];
        }
    }
    if (n != 1) f.push_back(n);
}

int T;

void llfactorize(long long n, vector<long long> &f) {
    if (T == 50)
        return ;
    T++;
    if (n == 1)
        return ;
    if (n < 1e+9) {
        factorize(n, f);
        return ;
    }
    if (isPrime(n)) {
        f.push_back(n);
        return ;
    }
    long long d = n;
    for (int i = 2; d == n; i++)
        d = pollard_rho(n, i);
    llfactorize(d, f);
    llfactorize(n/d, f);
}

map<ll,int> cnt;

int32_t main()
{
    srand(time(NULL));

    freopen( (filename + ".inp").c_str(), "r", stdin);
    freopen( (filename + ".out").c_str(), "w", stdout);

    sieve();
    int n;
    cin >> n;
    for (int i=1;i<=n;i++)
    {

```

```

        T = 0;
        ll x;
        cin >> x;
        vector<ll> tmp;
        llfactorize(x,tmp);
        sort(tmp.begin(),tmp.end());
        vector<ll>::iterator it = unique(tmp.begin(),tmp.end());
        tmp.resize(distance(tmp.begin(),it));
        for (int j=0;j<tmp.size();j++)
        {
            //cerr << tmp[j] << ' ';
            cnt[tmp[j]]++;
        }
        //cerr << '\n';
    }
    int res = 0;
    for (auto&p : cnt)
        res = max(res,p.second);
    cout << res;
    return 0;
}

```

Bài 13: Mắt xích yêu nhất