

VP24. ĐỐI ĐẦU

Tên chương trình: OPPOSITION.???

Quan hệ giữa 2 bộ lạc Volantis và Pentos ngày càng căng thẳng và chiến tranh là điều không thể tránh khỏi. Hướng mà người Pentos có thể tấn công là một dải đất một bên giới hạn bởi sông và bên kia – bởi vách núi dựng đứng.

Để tăng cường khả năng chống đỡ người Volantis tạo một tuyến phòng thủ là một đường thẳng từ vách núi tới sông và cắm chông dày đặc ở n đoạn trên đó, đoạn thứ i trong khoảng từ a_i đến b_i . Con sông cắt tuyến phòng thủ ở điểm l và vách núi - ở điểm r . Như vậy $l \leq a_i \leq b_i \leq r$. Các đoạn cắm chông không giao nhau.

Theo kế hoạch ban đầu đội quân của Pentos chia thành m nhóm không giao nhau, dàn hàng ngang tấn công, nhóm thứ j bắt đầu từ vị trí c_j đến vị trí d_j ($l \leq c_j \leq d_j \leq r$). Ở phút chót, khi cuộc tấn công sắp bắt đầu, trinh sát của Pentos mới phát hiện ra các bãi cắm chông của người Volantis. Không kịp bố trí lại đội hình thủ lĩnh của Pentos chỉ kịp ra lệnh cho tất cả các binh sĩ của mình dịch sang phải (hoặc sang trái) k đơn vị để số lượng người phải đi vào các bãi chông là ít nhất Giá trị k đảm bảo vị trí mọi binh sĩ đều nằm trong khoảng từ l đến r .

Tổn thất do đâm phái chông được tính bằng tổng độ dài phần giao nhau giữa các bãi chông với các nhóm tấn công.

Hãy tính tổn thất tối thiểu phải chịu của Pentos do đâm chông phòng thủ.

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

- ✚ Dòng đầu tiên chứa 2 số nguyên n và m ($1 \leq n, m \leq 300$),
- ✚ Dòng thứ 2 chứa 2 số nguyên l và r ($-10^9 \leq l \leq r \leq 10^9$),
- ✚ Dòng thứ i trong n dòng tiếp theo chứa 2 số nguyên a_i và b_i ,
- ✚ Dòng thứ j trong m dòng tiếp theo chứa 2 số nguyên c_j và d_j .

Kết quả: Đưa ra file văn bản OPPOSITION.OUT tổn thất tối thiểu tính được.

Ví dụ:

OPPOSITION.INP
3 2
0 5
0 1
2 3
4 5
0 1
2 3

OPPOSITION.OUT
0

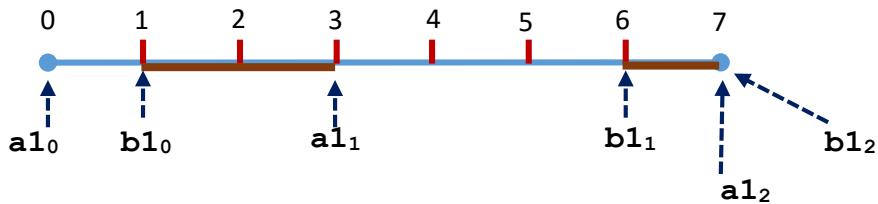
OPPOSITION.INP
2 3
0 7
1 3
6 7
1 2
3 5
6 7

OPPOSITION.OUT
1

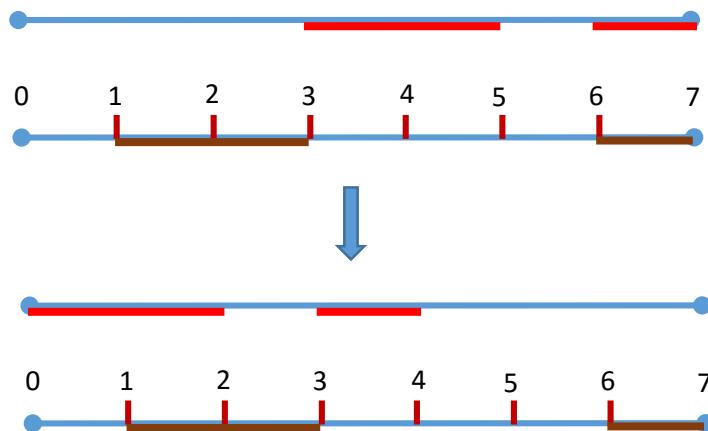


Giải thuật: (xem ví dụ 2)

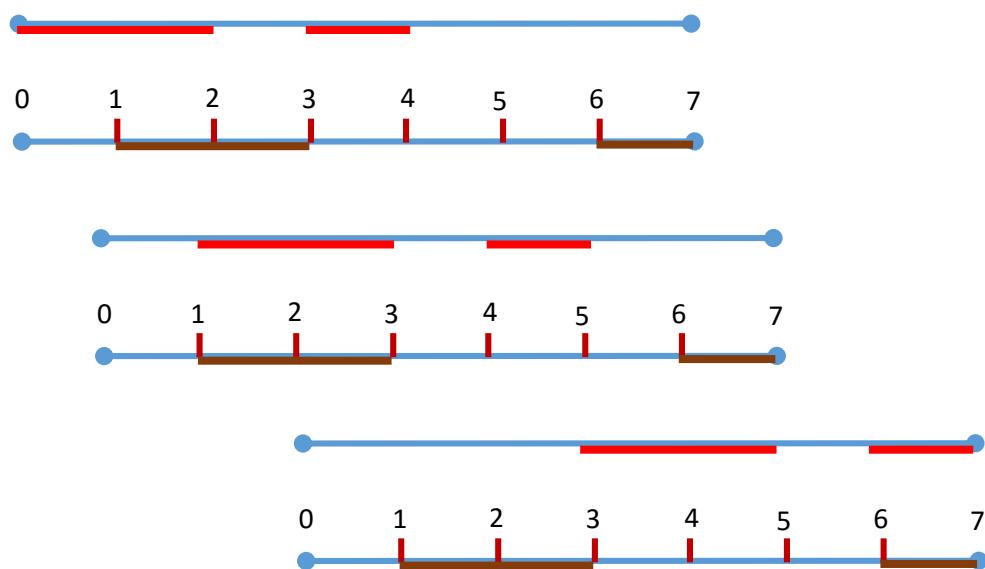
Mảng `int a1[302], b1[302]` – quản lý điểm đầu và cuối đoạn không cắm chông:



Mảng `int a2[302], b2[302]` – quản lý điểm đầu và cuối đoạn tấn công, tịnh tiến các đoạn để $a2_0$ trùng với mép trái ($\equiv L$):



Đây tịnh tiến hàng tấn công sang phải để lần lượt điểm đầu, điểm cuối khoảng tấn công ($a2_j, b2_j$) trùng với điểm đầu, điểm cuối khoảng trống không có chông ($a1_i, b1_i$), tính và lưu *min* tồn thât.



Mảng `int s[302]` quản lý tổng độ dài khoảng trống, s_i – tổng độ dài khoảng trống trước khoảng $[a1_i, b1_i]$, $s_0 = 0$.

Các mảng `int a3[302], b3[302]` – quản lý $a2_i, b2_i$ sau khi dịch chuyển sang phải ở mỗi bước xử lý.

Hàng đợi `priority_queue<int, vector<int>, greater<int> > q` – quản lý các đại lượng dịch cần xét.

```
void shift_v()
{ int ta, tb;
    q.push(0);
    for(int i=0;i<m;++i)
        for(int j=0;j<n;++j)
            {ta=a1[j]-a2[i]; tb=b1[j]-b2[i];
             if(ta>0 && ta<=d)q.push(ta);
             if(tb>0 && tb<=d)q.push(tb);
             if(ta>d && tb> d)break;
            }
}
```

Các giá trị dịch chuyển được lưu trữ theo thứ tự không giảm và có thể có nhiều giá trị giống nhau. Khi xử lý cần lọc và bỏ qua những giá trị đã xử lý.

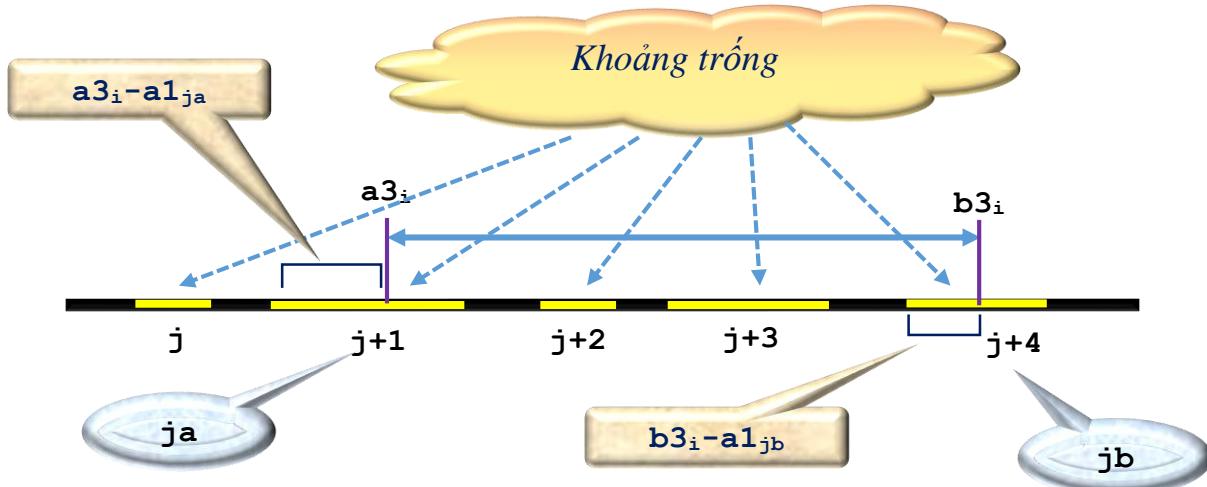
Với mỗi giá trị chuyển dịch **dq** cần xử lý:

- ✚ Từ a_{2i} và b_{2i} tính vị trí mới của đoạn này: $a_{3i} = a_{2i} + dq$, $b_{3i} = b_{2i} + dq$.
- ✚ Tìm tổng khoảng trống trong đoạn $[a_{3i}, b_{3i}]$:
- ✚ Tìm các chỉ số ja và jb thỏa mãn:

$$ja = \min\{j \mid a_{3i} \leq b_{1j}\}$$

$$jb = \min\{j \mid b_{3i} \leq b_{1j}\}$$

- ✚ Tính tổng khoảng trống t trong đoạn này.



Chương trình

```
#include <fstream>
#include <algorithm>
#include <queue>
#include <vector>
#include <ctime>

#define NAME "opposition."

using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n, m, l, r, d, dq, sum, ans=2000000009;
int a1[302], a2[302], b1[302], b2[302], a3[302], b3[302], s[302];
priority_queue<int, vector<int>, greater<int> > q;

void init()
{
    fi>>n>>m>>l>>r;
    int x,y, last = 1;
    for (int i = 0; i < n; ++i)
    { fi>>x>>y;
        a1[i] = last;
        b1[i] = x;
        last = y;
    }
    a1[n] = last;
    b1[n] = r;
    ++n;s[0]=0;
    for(int i=1;i<n;++i)s[i]=s[i-1]+b1[i-1]-a1[i-1];
    for (int i = 0; i < m; ++i) fi>>a2[i]>>b2[i];
    int tg = l - a2[0];
    for (int i = 0; i < m; ++i) {
        a2[i] += tg;
        b2[i] += tg;
    }
    d=r-b2[m-1];
}

void intersec()
{
    int ja=0,jb=0,t;
    for(int i=0;i<m;++i){a3[i]=a2[i]+dq; b3[i]=b2[i]+dq;}a3[m]=0;
    sum=0;
    ja=lower_bound(b1, b1+n,a3[0]-1)-b1;
    for(int i=0;i<m;++i)
    {
        jb=ja; while(b3[i]>b1[jb])++jb;
        t=s[jb]-s[ja];
        if(a3[i]>a1[ja])t-=a3[i]-a1[ja];
        if(b3[i]>a1[jb])t+=b3[i]-a1[jb];
        sum+=b3[i]-a3[i]-t;
        ja=jb; while(a3[i+1]>b1[ja])++ja;
    }
}

void shift_v()
{ int ta, tb;
```

```

q.push(0);
for(int i=0;i<m;++i)
    for(int j=0;j<n;++j)
        {ta=a1[j]-a2[i]; tb=b1[j]-b2[i];
         if(ta>0 && ta<=d)q.push(ta);
         if(tb>0 && tb<=d)q.push(tb);
         if(ta>d && tb> d)break;
        }
}

int main()
{clock_t aa=clock();
 init();
 shift_v();
 dq=-1;
 while(!q.empty())
 {
     while(dq==q.top() && !q.empty())q.pop();
     if(!q.empty())
     {
         dq=q.top(); q.pop();
         intersec();
         if(ans>sum)ans=sum;
         if(ans==0)break;
     }
 }
 fo<<ans;

clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}

```

VQ07. MERLIN

Tên chương trình: MERLIN.???

Một hôm Merlin qua về tòa tháp của mình thì thấy tất cả n hũ rượu thuốc quý đều bị Morgana yểm bùa.

Merlin biết cách gỡ bỏ bùa chú, nhưng điều này đòi hỏi các bình cần khử bùa phải có rượu và chứa một lượng rượu như nhau.

Sau một lúc suy nghĩ Merlin quyết định chọn một số bình, rót hết rượu từ những bình được chọn sang các bình còn lại sao cho chúng có cùng một lượng rượu. Những bình rỗng không thể gỡ bỏ bùa chú bị đập vỡ. Với những bình còn lại Merlin tiến hành xử lý gỡ bỏ bùa chú. Bản thân các bình đựng rượu đều rất đẹp và quý, vì vậy Merlin có gắng chọn cách làm sao cho số bình phải đập bỏ là ít nhất.

Hãy xác định số lượng bình tối thiểu phải đập bỏ.

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

- ✚ Dòng đầu tiên chứa số nguyên n ($2 \leq n \leq 10^5$),
- ✚ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n – số lượng rượu trong các bình ($1 \leq a_i \leq 10^9$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản MERLIN.OUT một số nguyên – số lượng bình tối thiểu phải đập bỏ.

Ví dụ:

MERLIN.INP
5
1 2 3 4 5

MERLIN.OUT
2



Vq07 KStP20141109 H

Giải thuật:

Nhận xét: với cùng một số lượng bình phải đập, việc đập các bình dung lượng lớn sẽ có khả năng làm cho nhiều bình còn lại có dung lượng bằng nhau hơn việc đập các bình dung lượng nhỏ.

Các bước xử lý:

- ✚ B1. Nhập dữ liệu,
- ✚ B2. Sắp xếp ai theo thứ tự tăng dần,
- ✚ B3. Tạo mảng **int64_t b[100001]**: $b_0=0$; $b_i=b_{i-1}+a_i$, $i = 1 \div n$,
- ✚ B4. Với mọi $i = n \div 1$ kiểm tra có thể giữ lại các bình từ 1 đến i hay không, nếu được – ghi nhận kết quả **ans=n-i** và chuyển sang B5,
- ✚ B5. Đưa ra kết quả ans.

Độ phức tạp: O(nlogn).

Chương trình

```
#include <fstream>
#include <ctime>
using namespace std;
ifstream fi ("merlin.inp");
ofstream fo ("merlin.out");
int64_t a[100001],b[100001],d,r;
int n,ans;

int main()
{clock_t aa=clock();
 fi>>n;
 for(int i=1;i<=n;++i) fi>>a[i];
 sort(a+1,a+n+1);
 b[0]=0; for(int i=1;i<=n;++i)b[i]=b[i-1]+a[i];
 for(int i=n;i>0;--i)
 {
     d=a[i]*i-b[i]; r=b[n]-b[i];
     if(r>=d) {ans=n-i; break;}
 }
 fo<<ans;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ18. KHỞI TẠO MẬT KHẨU

Tên chương trình: PWGEN.???

Steve xây dựng một diễn đàn trên mạng cho Câu lạc bộ Tin học của nhà trường. Mỗi thành viên, khi đăng ký vào diễn đàn sẽ được hệ thống cấp một mật khẩu. Mật khẩu tạo ra phải không quá đơn giản nhưng cũng phải dễ nhớ. Với mỗi thành viên của diễn đàn hệ thống sẽ cung cấp 4 số nguyên **n**, **a**, **b** và **c**. Muốn vào diễn đàn người sử dụng phải nhập vào một xâu độ dài **n** thỏa mãn các yêu cầu:

- ✚ Có ít nhất **a** ký tự chữ cái hoa,
- ✚ Có ít nhất **b** ký tự chữ cái thường,
- ✚ Có ít nhất **c** ký tự chữ số,
- ✚ Không có 2 ký tự liên tiếp giống nhau.

Với **n**, **a**, **b**, **c** cho trước, hãy đưa ra mật khẩu phù hợp có thứ tự từ diễn nhỏ nhất và mật khẩu phù hợp có thứ tự từ diễn lớn nhất.

Dữ liệu: Vào từ file văn bản PWGEN.INP gồm một dòng chứa 4 số nguyên **n**, **a**, **b** và **c** ($a+b+c \leq n$, $1 \leq n \leq 100$).

Kết quả: Đưa ra file văn bản PWGEN.OUT các mật khẩu tìm được, mỗi mật khẩu trên một dòng. Dòng đầu tiên chứa mật khẩu có thứ tự từ diễn nhỏ nhất

Ví dụ:

PWGEN.INP
8 2 5 1

PWGEN.OUT
0Abababa
zyzyzzY9



Vq18 VKO20141130 G

Giải thuật:

Nhận xét: theo thứ tự từ điển ký tự số < ký tự chữ cái hoa < ký tự chữ cái thường.

Mật khẩu cần tìm bao gồm 3 thành phần **sa**, **sb** và **sc**, trong đó **sa** – xâu các ký tự chữ cái hoa, **sb** – xâu các ký tự chữ cái thường, **sc** – xâu các ký tự số.

Gọi **smn** – mật khẩu phù hợp có thứ tự từ điển nhỏ nhất, **smx** – mật khẩu phù hợp có thứ tự từ điển lớn nhất.

Dễ dàng nhận thấy:

$$smn = sc + sa + sb$$

$$smx = sb + sa + sc$$

Ở **smn**, các ký tự tự do được nạp vào **sc**, còn ở **smx** – các ký tự tự do được nạp vào **sb**.

$smn = \underline{010101\dots} \quad \underline{ABABAB\dots} \quad \underline{ababab\dots}$

$c+d$ ký tự a ký tự b ký tự

$$smx = \underbrace{zyzyzy\dots}_{b+d \text{ ký tř}} \underbrace{ZYZYZY\dots}_a \underbrace{989898\dots}_c$$

Chương trình

```

#include <fstream>
#include <string>
#define NAME "pwgen."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,a,b,c,b1,c1,d,t;
string smn,smx,sa,sb,sc;

int main()
{
    fi>>n>>a>>b>>c;
    d=n-a-b-c;
    c1=c+d; t=c1/2;
    sc=""; for(int i=1;i<=t;++i) sc+="01"; if(c1&1) sc+='0';
    t=a/2;sa="";for(int i=1;i<=t;++i) sa+="AB"; if(a&1) sa+='A';
    t=b/2;sb="";for(int i=1;i<=t;++i) sb+="ab"; if(b&1) sb+='a';
    smn=sc+sa+sb;
    b1=b+d;
    t=c/2;sc=""; for(int i=1;i<=t;++i) sc+="98"; if(c&1) sc+='9';
    t=a/2;sa="";for(int i=1;i<=t;++i) sa+="ZY"; if(a&1) sa+='Z';
    t=b1/2;sb="";for(int i=1;i<=t;++i) sb+="zy"; if(b1&1) sb+='z';
    smx=sb+sa+sc;
    fo<<smn<<' \n '<<smx;
}

```

VQ10. BIẾN ĐỔI DÃY SỐ

Tên chương trình: SEQ.???

Cô giáo dạy toán rất không ưa Steve và thường gọi Steve lên bảng giải những bài rất khó. Hôm nay cô giáo viết lên bảng dãy số nguyên không âm a_1, a_2, \dots, a_n và yêu cầu dùng ít phép biến đổi nhất để đưa về dãy số mới sao cho trong đó có h số liên tiếp nhau tạo thành một dãy tăng dần từ 1 đến h , tức là tồn tại i sao cho $a_i = 1, a_{i+1} = 2, \dots, a_{i+h-1} = h$. Nội dung mỗi phép biến đổi là xóa một số tùy chọn và thay nó bằng số mới lớn hơn số cũ một đơn vị.

Đĩ nhiên, anh bạn tội nghiệp Steve của chúng ta lại bị gọi lên bảng.

Hãy xác định số phép biến đổi tối thiểu Steve cần thực hiện hoặc đưa ra số -1 nếu không tồn tại cách nhận được dãy số mới theo yêu cầu.

Dữ liệu: Vào từ file văn bản SEQ.INP:

- ✚ Dòng đầu tiên chứa 2 số nguyên n và h ($1 \leq h \leq n \leq 200\,000$),
- ✚ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($0 \leq a_i \leq n$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản SEQ.OUT một số nguyên – kết quả tìm được.

Ví dụ:

SEQ.INP
4 3
1 1 0 2

SEQ.OUT
3



Gai thuật:

Nhân xét: Nếu đoạn từ i đến $i+h-1$ của dãy a có thể biến đổi được để có dãy số liên tục từ 1 đến h thì giá trị mới của a_{i+j} sau khi biến đổi sẽ là $j+1$.

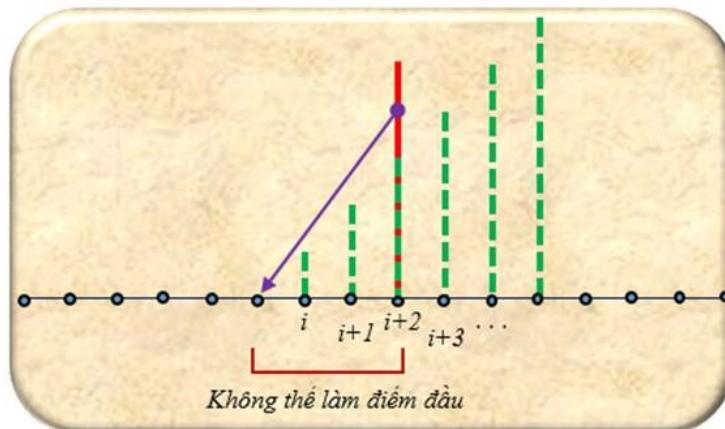
Vì vậy trong khoảng $[i, i+h-1]$ biến đổi được $a_{i+j} \leq j+1$, $j=0 \dots h-1$.

Chi phí biến đổi v sẽ là:

$$v = \frac{h * (h + 1)}{2} - \sum_{j=i}^{i+h-1} a_j$$

Nếu $a_i > 1$ thì các điểm trong khoảng $[i-g+1, i]$ không thể làm điểm đầu của một khoảng biến đổi được, trong đó:

$$g = \min\{a_i-1, h\}$$



Như vậy, với mỗi i cần biết xem nó có thể là điểm đầu của một khoảng biến đổi được hay không, nếu được – tính chi phí biến đổi và cập nhật giá trị sẽ đưa ra.

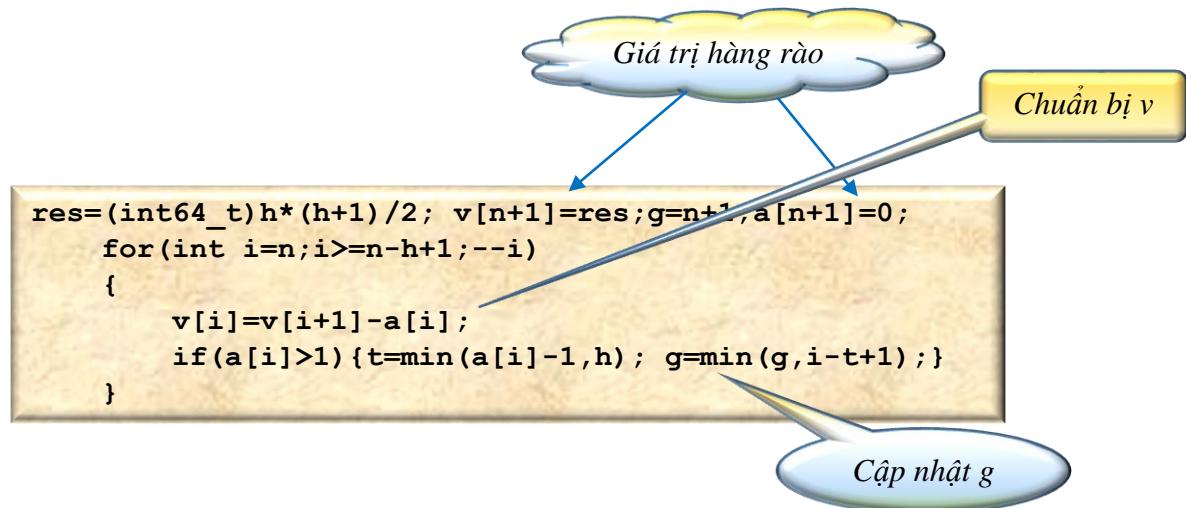
Dữ liệu:

- ✚ Mảng `int a[200002]` lưu dữ liệu ban đầu,
- ✚ Mảng `int64_t v[200002]` lưu chi phí biến đổi (có thể vòng tránh mảng này bằng cách dùng 2 giá trị `void` và `vnew`).

Các bước xử lý:

B1. Nhập dữ liệu,

B2. Chuẩn bị:



B3. Duyệt và cập nhật kết quả với $i = n-h+1 \div 1$.

```
for(int i=n-h+1;i>0;--i)
{
    v[i]=v[i+1]-a[i]+a[i+h];
    if(i<g && a[i]<2)res=min(res,v[i]);
    if(a[i]>1){t=min(a[i]-1,h); g=min(g,i-t+1);}
}
```

B4. Đưa ra kết quả.

Độ phức tạp của giải thuật: $O(n)$.

Chương trình

```
#include <fstream>
#include <cmath>
#include <ctime>
#include <iomanip>
#define NAME "seq."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,h,g,a[200002],t;
int64_t v[200002],res;

int main()
{
    clock_t aa=clock();
    fi>>n>>h;
    for(int i=1;i<=n;++i)fi>>a[i];
    res=(int64_t)h*(h+1)/2; v[n+1]=res;g=n+1;a[n+1]=0;
    for(int i=n;i>=n-h+1;--i)
    {
        v[i]=v[i+1]-a[i];
        if(a[i]>1){t=min(a[i]-1,h); g=min(g,i-t+1);}
    }
    for(int i=n-h+1;i>0;--i)
    {
        v[i]=v[i+1]-a[i]+a[i+h];
        if(i<g && a[i]<2)res=min(res,v[i]);
        if(a[i]>1){t=min(a[i]-1,h); g=min(g,i-t+1);}
    }
    fo<<res;
    clock_t bb=clock();
    fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ12. GIÁO SƯ CHAOS

Tên chương trình: CHAOS.???

Phòng thí nghiệm tuyệt mật của giáo sư Chaos nghiên cứu về một loại vi khuẩn cực độc. Đầu giờ ngày đầu tiên phòng thí nghiệm có **a** cá thể vi khuẩn nguy hiểm này.

Mỗi ngày thí nghiệm được tiến hành theo quy trình sau. Buổi sáng giáo sư lấy hộp chứa vi khuẩn, chuyển tất cả chúng sang tủ âm nhân giống. Ở đó chúng sẽ sinh sản, mỗi vi khuẩn sẽ tạo thành **b** vi khuẩn mới. Tiếp theo, người ta sẽ lấy ra khỏi tủ nhân giống **c** vi khuẩn để tiến hành các nghiên cứu khác nhau, sau đó khử trùng, diệt hết chúng. Nếu số lượng lấy ra không đủ **c** thì tất cả vi khuẩn sẽ được lấy ra thí nghiệm và nghiên cứu kết thúc. Số vi khuẩn còn lại cuối ngày sẽ được đưa vào hộp chứa để tiếp tục nghiên cứu trong những ngày tiếp theo. Nhưng hộp chứa chỉ có thể lưu giữ tối đa **d** vi khuẩn. Nếu còn thừa nhiều người ta cũng chỉ lưu lại **d** vi khuẩn, số vi khuẩn còn thừa, nếu có, sẽ bị diệt hết.

Để lên kế hoạch nghiên cứu giáo sư Chaos cần biết sau ngày thứ **k** sẽ có bao nhiêu vi khuẩn còn lại.

Hãy xác định số lượng vi khuẩn còn lại sau ngày thứ **k**.

Dữ liệu: Vào từ file văn bản CHAOS.INP gồm một dòng chứa 5 số nguyên **a**, **b**, **c**, **d** và **k** ($1 \leq a, b \leq 1000$, $0 \leq c \leq 1000$, $a \leq d \leq 1000$, $1 \leq k \leq 10^{18}$).

Kết quả: Đưa ra file văn bản CHAOS.OUT một số nguyên – số lượng vi khuẩn còn lại sau ngày thứ **k**.

Ví dụ:

CHAOS.INP
1 3 1 5 2

CHAOS.OUT
5



Giải thuật:

Nhận xét:

- ✚ Từ số vi rút ban đầu a , qua một ngày lượng a mới sẽ là
$$a = \max\{0, \min\{a^*b - c, d\}\}$$
- ✚ Giá trị a sẽ tăng dần tới d hoặc giảm dần về 0,
- ✚ Nếu khi chuyển sang ngày mới a không thay đổi thì từ đó trở đi a không thay đổi.
- ✚ Như vậy ta chỉ cần tính và lưu giá trị của a trong d ngày đầu tiên. Nếu $k \leq d$ thì tra cứu kết quả từ bảng đã lưu. Nếu $k > d$ – kết quả giống ngày d .

Chương trình

```
#include <iostream>
#include <cmath>
#include <ctime>
#include <iomanip>
#define NAME "chaos."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int a,b,c,d,r[1001],ans;
int64_t k;

int main()
{
    fi>>a>>b>>c>>d>>k;
    r[0]=a;
    for(int i=1;i<=d;++i) r[i]=max(0,min(r[i-1]*b-c,d));
    if (k<=d) ans=r[k]; else ans=r[d];
    fo<<ans;
}
```

VQ15. CỔ BÀI

Tên chương trình: CARDS.???

Steve có nhiệm vụ lập trình minh họa cho một trò chơi mới với cỗ bài n quân. Đây là trò chơi một người. Trên mỗi quân bài có ghi một số nguyên trong phạm vi từ 1 đến m . Người chơi được chia k quân bài trên cùng của cỗ bài. Phần còn lại của cỗ bài được đặt trên bàn. Trong suốt quá trình chơi, không lúc nào người chơi không được cầm trên tay quá k quân bài. Ở mỗi thời điểm có thể thực hiện một trong 3 hành động sau:

- ✚ Bỏ một quân bài đang cầm trên tay xuống tập bài dập. Quân bài này sẽ không được dùng lại trong quá trình chơi tiếp theo,
- ✚ Nếu số bài cầm trên tay ít hơn k , người chơi có thể lấy một quân bài trên cùng ở cỗ bài trên bàn, bổ sung vào số bài mình đang cầm,
- ✚ Hạ quân bài có số x xuống bàn nếu trước đó đã hạ các quân bài với các số $1, 2, \dots, x-1$ và quân bài số x chưa hạ.

Trò chơi kết thúc khi không thể thực hiện được hành động nào trong số kể trên. Mục tiêu của trò chơi là làm thế nào để hạ được nhiều quân bài nhất có thể.

Steve lập trình minh họa nên biết trước số ghi trên các quân bài ở cỗ bài trên bàn và biết các quân được chia cho người chơi. Hãy xác định số tối đa các quân bài có thể hạ xuống bàn.

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

- ✚ Dòng đầu tiên chứa số nguyên t – số lượng tests trong file ($1 \leq t \leq 10^5$),
- ✚ Dữ liệu mỗi test ghi trên 2 dòng:
 - Dòng đầu tiên chứa 3 số nguyên n, m và k ($1 \leq n, m \leq 10^5, 1 \leq k \leq n$),
 - Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($1 \leq a_i \leq m, i = 1 \div n$).

Tổng các n trong file không vượt quá 10^5 .

Kết quả: Đưa ra file văn bản CARDS.OUT, kết quả mỗi test đưa ra trên một dòng dưới dạng số nguyên, xác định số quân bài tối đa có thể hạ.

Ví dụ:

CARDS.INP
3
4 3 1
3 2 1 2
1 2 1
2
5 5 2
4 2 1 4 3

CARDS.OUT
2
0
4



Giải thuật:

Nhận xét:

- ⊕ Các thao tác được thực hiện theo từng bước và không quay lui, vì vậy đây là bài toán mô hình quy hoạch động,
- ⊕ Mô hình quy hoạch động đòi hỏi duyệt dữ liệu, xử lý từ cuối về đầu,
- ⊕ Nếu bài toán có lời giải với yêu cầu hạ các quân bài tạo thành dãy $1, 2, \dots, \mathbf{x}$ thì cũng có lời giải khi yêu cầu hạ các quân bài tạo thành dãy $1, 2, \dots, \mathbf{y}$ với $\mathbf{y} < \mathbf{x}$ vì vậy có thể tìm $\max\{\mathbf{x}\}$ theo giải thuật tìm kiếm nhị phân trong khoảng $[0, \mathbf{m}]$.

Từ các nhận xét trên ta có giải thuật xử lý cho mỗi bộ dữ liệu như sau:

- ❖ B1. Nhập dữ liệu,
- ❖ B2. Tìm $\max\{\mathbf{x}\}$ theo giải thuật tìm kiếm nhị phân trong khoảng $[0, \mathbf{m}]$,
- ❖ B3. Đưa ra \max tìm được.

Trọng tâm của giải thuật là hàm $\text{check}(\mathbf{u})$ kiểm tra xem có thể hạ bài tạo thành dãy $1, 2, \dots, \mathbf{u}$. Để tiện xử lý ta sẽ ghi số các số trên quân bài bắt đầu từ 0. Như vậy khoảng cần xét sẽ là $[0, \mathbf{m}-1]$, các giá trị a_i sẽ giảm 1 trong lưu trữ.

Giải thuật kiểm tra với giá trị \mathbf{u} :

- ⊕ Quân bài có số ghi \mathbf{x} ($\mathbf{x} \leq \mathbf{u}$) chưa có thể hạ xuống được gọi là quân bài chờ,
- ⊕ Ban đầu số ở các quân bài chờ là $\mathbf{u}, \mathbf{u}-1, \dots, \mathbf{u} - \max\{k-1, 0\}$,
- ⊕ Duyệt các quân bài từ \mathbf{n} lùi về 1, nếu gặp quân bài chờ thì xóa dấu hiệu chờ của quân bài đó và bổ sung thêm quân bài chờ mới là $\mathbf{p}-1$, trong đó \mathbf{p} – quân bài chờ hiện tại nhỏ nhất.
- ⊕ Số lượng bài chờ bằng 0 nghĩa là có thể hạ hết các quân bài theo yêu cầu, giá trị \mathbf{u} có thể đạt được,
- ⊕ Trong trường hợp ngược lại, khi duyệt hết cỗ bài mà vẫn còn quân chờ - không thể đạt tới giá trị \mathbf{u} .

Tổ chức dữ liệu cho hàm kiểm tra $\text{check}(\mathbf{u})$: xin cấp phát mới vec tơ used với kích thước \mathbf{u} phần tử, bộ nhớ sẽ được xóa 0 khi cấp phát.

Độ phức tạp của giải thuật: $O((n+m)\log m)$.

Chương trình

```
#include <fstream>
#include <ctime>
#include <vector>
#define NAME "cards."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int t,n,m,k,a[100000];

bool check(int u)
{
    vector<bool> used(u);
    int d = 0;
    for (int i = n-1; i >=0; i--) {
        if (a[i] >= u) continue;
        if (used[a[i]]) continue;
        if (a[i] >= u - k - d)
            used[a[i]] = true, d++;
    }
    return d == u;
}

void xly()
{
    fi>>n>>m>>k;
    for(int i=0;i<n;++i)fi>>a[i],--a[i];
    int l = 0;
    int r = m+1;
    while (r - l > 1) {
        int mid = (l + r) / 2;
        if (check(mid))
            l = mid;
        else
            r = mid;
    }
    fo<<l<<'\'n';
}
int main()
{clock_t aa=clock();
 fi>>t;
 for(int i=0;i<t;++i) xly();
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ21. SỬA HÀNG RÀO

Tên chương trình: WALL.???

Sau khi dựng xong nhà kho chúa cỏ, dì Poly quyết định dùng m tấm gỗ còn thừa gia cố hàng rào của vườn rau ngăn không cho gà vào phá và giao công việc này cho Tôm và Hắc Phin làm. Nhiệm vụ của hai cậu bé tội nghiệp là đóng thêm vào các tấm ván hàng rào để có hàng rào mới càng cao càng tốt. Nhìn vẽ mặt buồn thiu và lóng ngóng của 2 đứa Jim quyết định sẽ làm giúp. Hàng rào được ghép từ n tấm gỗ cùng độ rộng như nhau và bằng độ rộng của các tấm gỗ còn thừa, tấm thứ i có độ cao a_i , $i = 1 \div n$. Tôm và Hắc Phin chỉ phải xếp các tấm còn thừa lên xe ba gác để Jim kéo đi. Các tấm gỗ được xếp thành một chồng, tính từ trên xuống tấm thứ j có độ dài b_j , $j = 1 \div m$. Jim kéo xe ba gác đi dọc theo hàng rào. Đến một tấm nào đó muôn gia cố Jim sẽ lấy một tấm gỗ từ xe đóng tiếp lên tấm gỗ trên hàng rào và độ cao mới của tấm này trên hàng rào sẽ là tổng độ cao của tấm cũ và tấm mới đóng thêm. Jim chỉ đóng thêm một tấm mới vào tấm cũ vì muôn đảm bảo độ chắc chắn của hàng rào. Jim có thể lấy tấm trên cùng ở xe hoặc vất ra khỏi xe một số tấm cho đến khi gặp tấm vừa ý. Người ta vẫn nói “*Không đâu tới trέ, khỏe đâu tới già!*” Jim đã đứng tuổi và không còn sức để xếp lại các tấm gỗ bị bỏ ra vào xe. Ngoài ra, Jim cũng khá mê tín nên không quay lại lấy những tấm đã loại.

Hãy xác định độ cao lớn nhất có thể đạt được của hàng rào sau khi gia cố. Độ cao của hàng rào được tính bằng độ cao tấm gỗ thấp nhất trên hàng rào.

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

- ✚ Dòng đầu tiên chứa số nguyên n ($1 \leq n \leq 10^5$),
- ✚ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^8$, $i = 1 \div n$),
- ✚ Dòng thứ 3 chứa số nguyên m ($1 \leq m \leq 10^5$),
- ✚ Dòng cuối cùng chứa m số nguyên b_1, b_2, \dots, b_m ($1 \leq b_j \leq 10^8$, $j = 1 \div m$).

Kết quả: Đưa ra file văn bản WALL.OUT, dòng đầu tiên chứa 2 số nguyên h và k – độ cao lớn nhất có thể của hàng rào và số tấm gỗ đã được đóng thêm, mỗi dòng trong k dòng tiếp theo chứa 2 số nguyên x và y , trong đó x – tấm gỗ trên hàng rào được đóng cao hơn, y – tấm gỗ được dùng để đóng. Đưa ra phương án tùy chọn nếu tồn tại nhiều cách đóng khác nhau.

Ví dụ:

WALL.INP
6
2 5 4 1 7 5
7
2 3 1 3 2 4 6

WALL.OUT
5 3
1 2
3 4
4 7



Giải thuật:

Nhận xét: Bằng kỹ thuật tìm kiếm nhị phân ta có thể xác định độ cao lớn nhất có thể đạt được.

Giải thuật kiểm tra có thể đạt độ cao h hay không:

- ✚ Duyệt từng a_i ($i = 0 \div n-1$):
 - ✚ Nếu mọi $a_i \geq h$ – có thể đạt độ cao h ,
 - ✚ Nếu gặp $a_i < h$ – tìm b_j đầu tiên thỏa mãn $a_i + b_j \geq h$ và ghi nhận cặp (i, j) . Lưu ý: j tăng dần trong quá trình tìm kiếm với các a_i), nếu không tìm thấy b_j thỏa mãn – không thể đạt độ cao h .

Lưu ý:

- Sau khi xác định được độ cao h lớn nhất phương án ghép các tám gỗ có thể đã bị xóa. Việc lưu các kết quả trong quá trình tìm kiếm sẽ mất nhiều thời gian, vì vậy sau khi đã tìm được độ cao lớn nhất, để ghi nhận phương án lắp ghép cần gọi lại hàm kiểm tra một lần nữa với độ cao tìm được!
- Có thể giảm thời gian thực hiện chương trình bằng cách tách việc kiểm tra (không ghi nhận kết quả ghép) với việc dẫn xuất kết quả thành 2 hàm riêng biệt.

Độ phức tạp của giải thuật: $O(n \log k)$, trong chương trình: $k = 2 \times 10^9$.

Chương trình

```
#include <fstream>
#include <vector>
#include <ctime>
#define NAME "wall."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,m,a[100000],b[100000];
vector< pair<int, int> > res;

bool check(int h)
{
    int ptr = 0;
    res.clear();
    for (int i = 0; i < n; i++) {
        while (ptr < m && a[i] + b[ptr] < h) ptr++;
        if (a[i] < h && ptr == m) return false;
        if (a[i] < h) {
            res.push_back(make_pair(i, ptr));
            ptr++;
        }
    }
    return true;
}

int main()
{
    clock_t aa=clock();
    fi>>n;
    for (int i = 0; i < n; i++) fi>>a[i];
    fi>>m;
    for (int i = 0; i < m; i++) fi>>b[i];
    int l = 0;
    int r = (int)2e9+1;
    while (r - l > 1)
    {
        int mid = l + (r - l) / 2;
        if (check(mid))
            l = mid;
        else
            r = mid;
    }
    fo<<l<<' ';
    check(l);
    fo<<res.size()<<'\n';
    for (int i = 0; i < (int)res.size(); i++)
        fo<<res[i].first+1<<' '<<res[i].second+1<<'\n';
    clock_t bb=clock();
    fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ11. TRỢ GIÚP

Tên chương trình: F1.???

Petya và Vasya chơi cờ ca rô trong giờ nghỉ giải lao. Lưới ca rô có thể được coi là vô hạn. Petya đánh dấu **X** còn Vasya – chữ số **0**. Ai đặt được liên tiếp 5 dấu của mình theo đường ngang hoặc dọc hay theo đường chéo là thắng. Giờ nghỉ giải lao sắp hết mà ván đấu vẫn chưa kết thúc. Đến lượt Petya đi. Steve ngồi cạnh Petya sót ruột và quyết định trợ giúp, chỉ cách đi để Petya thắng ngay sau nước đi này hoặc cùng lâm – sau nước đi tiếp theo của mình không phụ thuộc vào cách đi của Vasya.

Toàn bộ các ô có nước đi hiện nay nằm gọn trong hình chữ nhật kích thước $n \times m$ ô.

Hãy đưa ra số lượng vị trí khác nhau để đi nếu Petya có cách thắng ngay sau khi thực hiện một hoặc 2 nước đi và đưa ra số 0 trong các trường hợp còn lại.

Dữ liệu: Vào từ file văn bản F1.INP:

- ➡ Dòng đầu tiên chứa 2 số nguyên **n** và **m** ($1 \leq n, m \leq 200$),
- ➡ Mỗi dòng trong **n** dòng sau chứa xâu độ dài **m** chứa các ký tự từ tập **{X, 0, .}**, ký tự ‘.’ chỉ ô trống.

Kết quả: Đưa ra file văn bản F1.OUT một số nguyên – kết quả xác định được.

Ví dụ:

F1.INP
5 3
...
000
XXX
...
...

F1.OUT
2



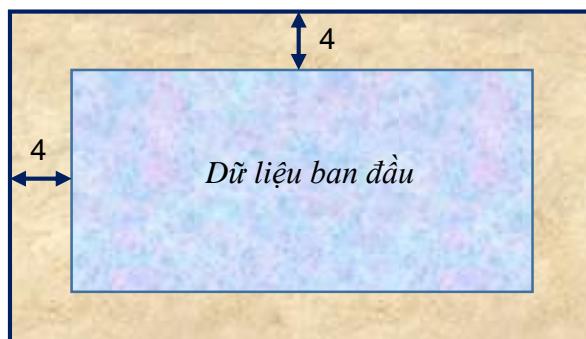
Giải thuật:

Nhận xét: Đây là một bài toán thuộc lĩnh vực lý thuyết Trò chơi. Các bài toán trong lĩnh vực này đều có đặc trưng:

- ✚ Tính toán đơn giản,
- ✚ Lô gic phức tạp: có nhiều tình huống và phải nhận dạng được các tình huống đó,
- ✚ Cần có hàm đánh giá “độ tốt” của một nước đi,
- ✚ Công cụ để nhận dạng tình huống và đánh giá nước đi: dựa trên cơ sở các bảng phương án (Decide Tables).

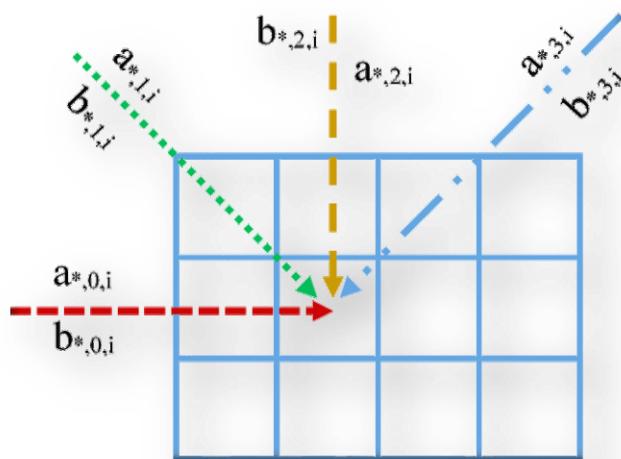
Tổ chức dữ liệu:

Lưu trữ bảng thông tin ban đầu: Mặc dù kích thước bàn cờ là vô hạn nhưng các nước đi hiện có chỉ nằm gọn trong hình chữ nhật kích thước $n \times m$, các nước đi mới được đặt ở những ô cách những ô đã đi cùng lăm là 2. Để tiện xử lý ta cần một hàng rào độ rộng 4 bao các ô cần xét.



Như vậy cần khai báo mảng `char c[208][208]`, các ô trên đường viền nhận giá trị ‘.’.

Mỗi nước đi có giá trị phụ thuộc vào 4 hướng xét: ngang, dọc và hai đường chéo qua ô đó. Để quản lý và đánh giá nước đi của mỗi người chơi cần 2 mảng `int a[4][208][208], b[4][208][208]`; mảng **a** để quản lý và đánh giá các nước đi của người thứ nhất, mảng **b** – cho người thứ 2.



Độ dài một chuỗi các ký tự ‘x’ hoặc ‘0’ phải được lưu trữ đối xứng trong mỗi mảng **a** và **b**: giá trị tương ứng với điểm đầu và cuối của đều phải lưu độ dài của chuỗi theo hướng đang xét. Ví dụ, khi duyệt từ trên xuống dưới và từ trái qua phải để tính **a**, ta có:

.	X	X	X	X	.
---	---	---	---	---	---

Các giá trị tương ứng trong mảng a : 0 4 2 3 4 0

Các công việc chính:

- ⊕ Tính giá trị các mảng a, b ,
- ⊕ Đếm số nước đi người thứ nhất có thể thắng ngay sau một nước đi,
- ⊕ Đếm số nước đi người thứ hai có thể thắng ngay sau một nước đi của mình nếu người thứ nhất chưa thắng,
- ⊕ Đếm số nước đi người thứ nhất có thể thắng được sau 2 nước đi.

a) Tính mảng a, b :

Việc tính a và b được thực hiện tương tự như nhau, vì vậy dưới đây ta chỉ xét cách tính a .

- ⊕ Giả thiết đang xét hướng k ($k = 0, 1, 2, 3$), tại vị trí (i, j) có $c_{i,j} = 'X'$:
- ⊕ Gán $a_{k,i,j}$ bằng giá trị trước đó của a theo hướng k và cộng thêm 1,
- ⊕ Kiểm tra, theo hướng k nếu ký tự tiếp theo khác ' X ' thì cập nhật giá trị của phần tử $t - 1$ trước đó của a theo hướng k , trong đó $t = a_{k,i,j}$.

Ví dụ, với $k = 1$ thì ký tự tiếp theo ở c sẽ là $c_{i+1,j+1}$, phần tử của a có thể được cập nhật là $a_{1,i-t+1,j-t+1}$.

b) Đếm số nước đi người thứ nhất có thể thắng ngay sau một nước đi:

- ⊕ Kiểm tra các ô (i, j) có $c_{i,j} = '.'$,
- ⊕ Gọi u, v là tọa độ ô trước ô (i, j) theo hướng k và p, q – tọa độ ô sau ô (i, j) theo hướng k , điều kiện tăng số lượng ô thắng được là $a_{k,u,v} + a_{k,p,q} \geq 4$, $k = 0, 1, 2, 3$.

Ví dụ, $k = 2$ và $c_{i,j} = 'X'$ thì cần kiểm tra $a_{2,i-1,j} + a_{2,i+1,j} \geq 4$.

c) Đếm số nước đi người thứ hai có thể thắng ngay sau một nước đi của mình nếu người thứ nhất chưa thắng:

Thực hiện tương tự như cách đếm số nước đi người thứ nhất có thể thắng ngay sau một nước đi, nhưng có lưu tọa độ của **một ô**, nơi có thể đi để thắng,

Nếu số lượng đếm được **lớn hơn 1** – người thứ nhất không có cách đi thắng (só nước có thể đi để thắng bằng 0).

d) Đếm số nước đi người thứ nhất có thể thắng được sau 2 nước đi:

Ô (i, j) được gọi là ô tiềm năng nếu:

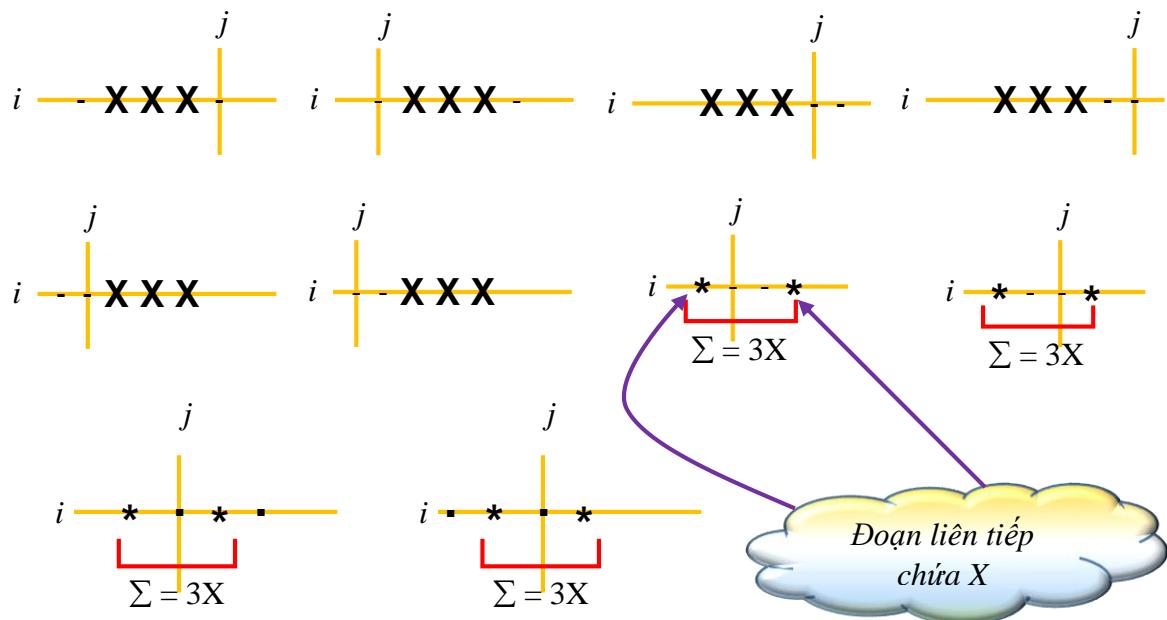
- ⊕ $c_{i,j} = '.'$
- ⊕ Nếu có cách đi 2 nước liên tiếp, trong đó có một nước ở ô (i, j) thì sẽ có một chuỗi 5 ký tự ' X ' liên tiếp.

Lượng tiềm năng là số lượng chuỗi 5 ký tự ' X ' liên tiếp khác nhau có thể nhận được với sự tham gia của ô (i, j) . Nếu lượng tiềm năng lớn hơn 1 thì đó là một ô có thể đánh để thắng.

Cần kiểm tra xem ô có lượng tiềm năng lớn hơn 1 có trùng với ô mà người thứ 2 cần đi để thắng hay không, nếu trùng thì cho kết quả số nước đi bằng 1 và thoát khỏi hàm đếm.

Lần lượt kiểm tra theo *tùng hướng* và *tích lũy* lượng tiềm năng. Ở mỗi hướng – kiểm tra xem câu hình thực tế có thuộc một trong số các câu hình đã liệt kê hay không.

Ví dụ ở hướng $k=0$, các câu hình đê ô (i, j) thành tiềm năng là:



Độ phức tạp của giải thuật: $O(n \times m)$.

Chương trình

```
#include <fstream>
#include <cmath>
#include <ctime>
#include <iomanip>
#define NAME "F1."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,m,a[4][209][209]={0},b[4][209][209]={0},ib=0,jb=0;
int ans1=0,ans2=0;
char c[209][209];
string s;

void xlB(int x, int y)
{int t;
 b[0][x][y]=b[0][x][y-1]+1;
if(c[x][y+1]!='0') {t=b[0][x][y];b[0][x][y-t+1]=t;}
 b[1][x][y]=b[1][x-1][y-1]+1;
if(c[x+1][y+1]!='0') {t=b[1][x][y];b[1][x-t+1][y-t+1]=t;}
 b[2][x][y]=b[2][x-1][y]+1;
if(c[x+1][y]!='0') {t=b[2][x][y];b[2][x-t+1][y]=t;}
 b[3][x][y]=b[3][x-1][y+1]+1; if(c[x+1][y-1]!='0') {t=b[3][x][y];b[3][x-t+1][y+t-1]=t;}
}

void xlA(int x, int y)
{int t;
 a[0][x][y]=a[0][x][y-1]+1;
if(c[x][y+1]!='X') {t=a[0][x][y];a[0][x][y-t+1]=t;}
 a[1][x][y]=a[1][x-1][y-1]+1;
if(c[x+1][y+1]!='X') {t=a[1][x][y];a[1][x-t+1][y-t+1]=t;}
 a[2][x][y]=a[2][x-1][y]+1;
if(c[x+1][y]!='X') {t=a[2][x][y];a[2][x-t+1][y]=t;}
 a[3][x][y]=a[3][x-1][y+1]+1; if(c[x+1][y-1]!='X') {t=a[3][x][y];a[3][x-t+1][y+t-1]=t;}
}

void upd(int x, int y)
{int t;
 t=a[0][x][y-1];if(t)a[0][x][y-t]=t;
 t=a[1][x-1][y-1];if(t)a[1][x-t][y-t]=t;
 t=a[2][x-1][y];if(t)a[2][x-t][y]=t;
 t=a[3][x-1][y+1];if(t)a[1][x+t][y+t]=t;
 t=b[0][x][y];if(t)b[0][x][y-t]=t;
 t=b[1][x-1][y-1];if(t)b[1][x-t][y-t]=t;
 t=b[2][x-1][y];if(t)b[2][x-t][y]=t;
 t=b[3][x-1][y+1];if(t)b[1][x-t][y+t]=t;
}

int check_0(int x, int y)
{int t,tg=0;
 if(a[0][x][y-1]==3 && c[x][y-4]=='.')++tg; //0 X *
 if(a[0][x][y+1]==3 && c[x][y+4]=='.')++tg; ///* X 0
 if(a[0][x][y-1]==3 && c[x][y+1]=='.')++tg; //X * 0
 if(a[0][x][y-2]==3 && c[x][y-1]=='.')++tg; //X 0 *
 if(a[0][x][y+2]==3 && c[x][y+1]=='.')++tg; ///* 0 X
```

```

if(a[0][x][y+1]==3 && c[x][y-1]=='.')++tg;      //0 * X
if(a[0][x][y-1]>0 && a[0][x][y+2]>0 && a[0][x][y-1]+a[0][x][y+2]==3
&& c[x][y+1]=='.')++tg;      //X * 0 X
if(a[0][x][y-2]>0 && a[0][x][y+1]>0 && a[0][x][y-2]+a[0][x][y+1]==3
&& c[x][y-1]=='.')++tg;      //X 0 * X
t=a[0][x][y-1]; if(t>0 && a[0][x][y+1]>0 && t+a[0][x][y+1]==3 &&
c[x][y-t-1]=='.')++tg;      //0 X * X
t=a[0][x][y+1]; if(a[0][x][y-1]>0 && t>0 && a[0][x][y-1]+t==3 &&
c[x][y+t+1]=='.')++tg;      //X * X 0
return tg;
}

int check_1(int x, int y)
{int t,tg=0;
if(a[1][x-1][y-1]==3 && c[x-4][y-4]=='.')++tg;      //0 X *
if(a[1][x+1][y+1]==3 && c[x+4][y+4]=='.')++tg;      //* X 0
if(a[1][x-1][y-1]==3 && c[x+1][y+1]=='.')++tg;      //X * 0
if(a[1][x-2][y-2]==3 && c[x-1][y-1]=='.')++tg;      //X 0 *
if(a[1][x+2][y+2]==3 && c[x+1][y+1]=='.')++tg;      //* 0 X
if(a[1][x+1][y+1]==3 && c[x-1][y-1]=='.')++tg;      //0 * X
if(a[1][x-1][y-1]>0 && a[1][x+2][y+2]>0 && a[1][x-1][y-
1]+a[1][x+2][y+2]==3 && c[x+1][y+1]=='.')++tg;      //X * 0 X
if(a[1][x-2][y-2]>0 && a[1][x+1][y+1]>0 && a[1][x-2][y-
2]+a[1][x+1][y+1]==3 && c[x-1][y-1]=='.')++tg;      //X 0 * X
t=a[1][x-1][y-1]; if(t>0 && a[1][x+1][y+1]>0 && t+a[1][x+1][y+1]==3
&& c[x-t-1][y-t-1]=='.')++tg;      //0 X * X
t=a[1][x+1][y+1]; if(a[1][x-1][y-1]>0 && t>0 && a[1][x-1][y-1]+t==3
&& c[x+t+1][y+t+1]=='.')++tg;      //X * X 0
return tg;
}

int check_2(int x, int y)
{int t,tg=0;
if(a[2][x-1][y]==3 && c[x-4][y]=='.')++tg;      //0 X *
if(a[2][x+1][y]==3 && c[x+4][y]=='.')++tg;      //* X 0
if(a[2][x-1][y]==3 && c[x+1][y]=='.')++tg;      //X * 0
if(a[2][x-2][y]==3 && c[x-1][y]=='.')++tg;      //X 0 *
if(a[2][x+2][y]==3 && c[x+1][y]=='.')++tg;      //* 0 X
if(a[2][x+1][y]==3 && c[x-1][y]=='.')++tg;      //0 * X
if(a[2][x-1][y]>0 && a[2][x+2][y]>0 && a[2][x-1][y]+a[2][x+2][y]==3
&& c[x+1][y]=='.')++tg;      //X * 0 X
if(a[2][x-2][y]>0 && a[2][x+1][y]>0 && a[2][x-2][y]+a[2][x+1][y]==3
&& c[x-1][y]=='.')++tg;      // X 0 * X
t=a[2][x-1][y]; if(t>0 && a[2][x+1][y]>0 && t+a[2][x+1][y]==3 &&
c[x-t-1][y]=='.')++tg;      // 0 X * X
t=a[2][x+1][y]; if(a[2][x-1][y]>0 && t>0 && a[2][x-1][y]+t==3 &&
c[x+t+1][y]=='.')++tg;      // X * X 0
return tg;
}

int check_3(int x, int y)
{int t,tg=0;
if(a[3][x-1][y+1]==3 && c[x-4][y+4]=='.')++tg;      //0 X *
if(a[3][x+1][y-1]==3 && c[x+4][y-4]=='.')++tg;      //* X 0
if(a[3][x-1][y+1]==3 && c[x+1][y-1]=='.')++tg;      //X * 0
if(a[3][x+1][y-1]==3 && c[x-1][y+1]=='.')++tg;      //0 * X
if(a[3][x-2][y+2]==3 && c[x-1][y+1]=='.')++tg;      //X 0 *
if(a[3][x+2][y-2]==3 && c[x+1][y-1]=='.')++tg;      //* 0 X

```

```

    if(a[3][x-1][y+1]>0 && a[3][x+2][y-2]>0 && a[3][x-
1][y+1]+a[3][x+2][y-2]==3 && c[x+1][y-1]=='.')++tg; // X * 0 X
    if(a[3][x-2][y+2]>0 && a[3][x+1][y-1]>0 && a[3][x-
2][y+2]+a[3][x+1][y-1]==3 && c[x-1][y+1]=='.')++tg; // X 0 * X
    t=a[3][x-1][y+1]; if(t>0 && a[3][x+1][y-1]>0 && t+a[3][x+1][y-1]==3
&& c[x-t-1][y+t+1]=='.')++tg; // 0 X * X
    t=a[3][x+1][y-1]; if(a[3][x-1][y+1]>0 && t>0 && a[3][x-1][y+1]+t==3
&& c[x+t+1][y-t-1]=='.')++tg; // X * X 0
    return tg;
}

void get_a1()
{for(int i=3;i<=n+4;++i)
    for(int j=3;j<=m+4;++j)
        if(c[i][j]=='.')
            if(a[0][i][j-1]+a[0][i][j+1]>=4 ||
               a[1][i-1][j-1]+a[1][i+1][j+1]>=4 ||
               a[2][i-1][j]+a[2][i+1][j]>=4 ||
               a[3][i-1][j+1]+a[3][i+1][j-1]>=4) ++ans1;

}

void get_a2()
{int t=0,b4=0,tg1,tg2,t2;
    for(int i=2;i<=n+6;++i)
        for(int j=2;j<=m+6;++j)
            if(c[i][j]=='.')
                if(b[0][i][j-1]+b[0][i][j+1]>=4 ||
                   b[1][i-1][j-1]+b[1][i+1][j+1]>=4 ||
                   b[2][i-1][j]+b[2][i+1][j]>=4 ||
                   b[3][i-1][j+1]+b[3][i+1][j-1]>=4)
                    {++b4;ib=i;jb=j; if(b4>1) return;}

            for(int i=1;i<=n+6;++i)
                for(int j=1;j<=m+6;++j)
                    if(c[i][j]=='.')
                        { t2=check_0(i,j);
                            if(i==ib && j==jb && t2>1){ib=0;ans2=1;return;}
                            if(t2>1){++ans2;continue;}
                            t2+=check_1(i,j);
                            if(i==ib && j==jb && t2>1){ib=0;ans2=1;return;}
                        if(t2>1){++ans2;continue;}
                            t2+=check_2(i,j);
                            if(i==ib && j==jb && t2>1){ib=0;ans2=1;return;}
                        if(t2>1){++ans2;continue;}
                            t2+=check_3(i,j);
                            if(i==ib && j==jb && t2>1){ib=0;ans2=1;return;}
                        if(t2>1){++ans2;continue;}
                        }
                    if(ib) ans2=0;
    }

int main()
{ clock_t aa=clock();
    fi>>n>>m;
    for(int i=0;i<=n+8;++i)
        for(int j=0;j<=m+8;++j)c[i][j]='.';
    for(int i=4;i<=n+3;++i)

```

```

{
    fi>>s;
    for(int j=4;j<=m+3;++j)c[i][j]=s[j-4];
}

for(int i=4;i<=n+3;++i)
for(int j=4;j<=m+3;++j)
switch(c[i][j])
{case 'X':xla(i,j); break;
 case '0':xlb(i,j); break;
}

get_a1();
if(ans1==0) get_a2();
if(ans1) fo<<ans1;else {if(ans2) fo<<ans2;else fo<<0; }

clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}

}

```

VQ22. TÌM KIẾM

Tên chương trình: SOUGHT.???

Máy bay tuần tiễu quân sự chiến lược trong một chuyến bay tuần tra đã gặp sự cố kỹ thuật. Tổ lái đã xử lý hết sức chuẩn mực, đưa được máy bay trở về căn cứ an toàn, nhưng một đầu đạn hạt nhân đã bị rơi xuống biển. Một lực lượng hùng hậu các tàu tìm kiếm được điều động tới khu vực dự kiến có đầu đạn thất lạc. Toàn bộ vùng biển được chia thành các băng, mỗi băng có độ rộng đơn vị. Để tránh các va chạm có thể xảy ra giữa các tàu cũng như tránh nhiễu loạn tín hiệu dò tìm các tàu được chia thành n nhóm, nhóm thứ i có c_i tàu ($i = 1 \dots n$). Nhóm thứ i bắt đầu tìm kiếm ở băng a_i và cứ sau một đơn vị thời gian tìm kiếm thì chuyển lên phía trước b_i đơn vị độ dài. Như vậy nhóm thứ i thực hiện việc tìm kiếm ở các băng $a_i, a_i + b_i, a_i + 2b_i, a_i + 3b_i, \dots$.

Có nhiều dấu hiệu cho thấy đầu đạn cần tìm có thể ở băng k .

Hãy xác định có bao nhiêu tàu đã thực hiện công việc tìm kiếm ở băng k và số tàu nhiều nhất tại một thời điểm cùng tìm kiếm ở băng k .

Dữ liệu: Vào từ file văn bản SOUGHT.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 i trong n dòng sau chứa 3 số nguyên a_i, b_i và c_i ($1 \leq a_i, b_i, c_i \leq 10^9$).

Kết quả: Dưa ra file văn bản SOUGHT.OUT trên dòng hai số nguyên – số lượng tàu đã tham gia tìm kiếm ở băng k và số tàu nhiều nhất đồng thời tìm kiếm ở băng k .

Ví dụ:

SOUGHT.INP
2 4
1 1 2
1 2 3

SOUGHT.OUT
2 2



Giải thuật:

Tổ chức dữ liệu:

- ✚ Mảng `pair<int, int> d[100000]` để lưu thời điểm và số tàu trong nhóm khi tới khảo sát bằng k :
- ✚ $d[i].first$ – thời điểm tới khảo sát bằng k ,
- ✚ $d[i].second$ – số tàu trong nhóm,
- ✚ Mảng `int64_t e[100000]` – lưu tổng số tàu cùng khảo sát bằng k ở từng thời điểm khi có tàu khảo sát bằng này,
- ✚ Biến lưu kết quả: `int64_t ans1=0, ans2=0;`

Mỗi nhóm được đặc trưng bởi 3 tham số a , b và c .

Điều kiện để nhóm khảo sát bằng k :

- ✚ $a \leq k$,
- ✚ $(k-a) \% b == 0$.

Duyệt tất cả các nhóm, với mỗi nhóm thỏa mãn các điều kiện trên:

- ✚ Tích lũy số tàu trong nhóm vào $ans1$ (`ans1+=c;`)
- ✚ Nạp cặp giá trị $(k-a)/b, c$ vào d .

Sắp xếp lại mảng d theo thứ tự tăng dần của $d[i].first$.

Nhóm số lượng tàu khảo sát bằng k ở cùng một thời điểm vào mảng e ,

Tính $ans2 = \max\{ei\}$, $i=0, 1, 2, \dots$

Đưa ra các kết quả $ans1$ và $ans2$.

Độ phức tạp của giải thuật: $O(n \log n)$.

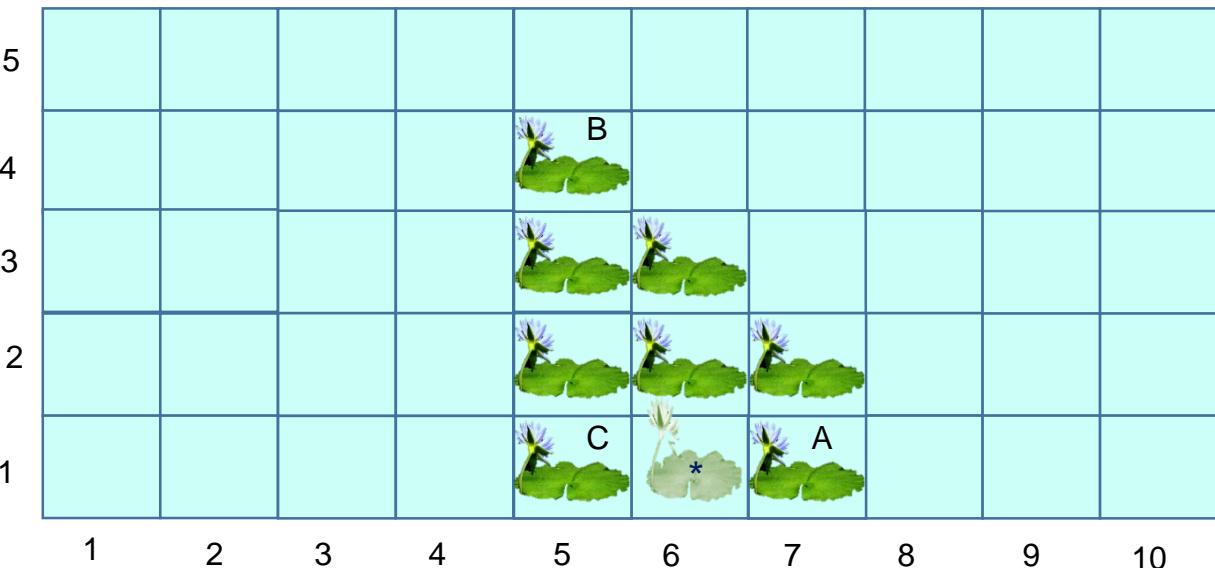
Chương trình

```
#include <fstream>
#include <iostream>
#include <cmath>
#include <ctime>
#include <iomanip>
#define NAME "sought."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,m=0,k,l=0,a,b,c,t,e[100000];
pair<int,int> d[100001];
int64_t ans1=0,ans2=0;
int main()
{
    clock_t aa=clock();
    fi>>n>>k;
    for(int i=0;i<n;++i)
    {
        fi>>a>>b>>c;
        if(a<=k && (k-a)%b==0){ans1+=c;
                                t=(k-a)/b;d[m++]=make_pair(t,c);}
    }
    if(m>0)
    {
        sort(d,d+m);
        e[1]=d[0].second;
        for(int i=1;i<m;++i)
            if(d[i].first==d[i-1].first)e[1]+=d[i].second; else
        e[++1]=d[i].second;
        ans2=e[0];
        for(int i=1;i<=l;++i) if(ans2<e[i])ans2=e[i];
    }
    fo<<ans1<<' ' <<ans2;
    clock_t bb=clock();
    fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ20. CÁC CHÚ RÙA

Tên chương trình: TURTLES.???

Hồ của chú rùa Tortilla đã thay đổi nhiều kể từ khi lần cuối cùng Buratino tới chơi. Bây giờ hồ có hình chữ nhật kích thước $w \times h$ ô. Ở một số ô có hoa súng mọc. Các con cái cháu chắt của Tortilla thường leo lên lá hoa súng sưởi nắng và trò chuyện với nhau. Có tất cả n cụm hoa súng tạo thành một miền *liên thông*, tức là mọc trong trap các kè cạnh, cụm thứ i mọc ở ô (r_i, c_i). Thỉnh thoảng các chú rùa cũng tới thăm nhau bằng cách bò từ cụm hoa súng này sang cụm hoa



ở ô kè cạnh. Các chú rùa sống rất lâu và nỗi tiếng là chậm chạp. Cá trong suy nghĩ cũng vậy! Mỗi chú rùa, khi định đi thăm người khác, sẽ chọn trong đầu 2 trong số 4 hướng chuyển động: lên trên, xuống dưới, sang phải, sang trái và sau đó chỉ di chuyển theo 2 hướng đã chọn đó. Ở hình trên từ A có thể tới được B (bằng cách chọn 2 hướng lên trên và sang trái), nhưng từ A không thể tới C. Nếu trồng thêm một cụm hoa súng ở ô đánh dấu "*" thì từ một cụm hoa có thể đến được cụm hoa khác bất kỳ. Khi đó tập các ô có hoa được gọi là *thuận lợi*.

Với quyền hạn, uy tín và tầm nhìn xa trông rộng của mình Tortilla quyết định huy động nhân lực trồng thêm q cụm hoa nữa ở q ô khác nhau còn trồng, lần lượt ở các ô (nr_j, nc_j), $j = 1 \dots q$. Mỗi cụm hoa mới, khi xuất hiện vẫn đảm bảo tính liên thông của các ô có hoa. Tortilla cũng thành lập một nhóm giám sát thi công, thường xuyên kiểm tra và báo cáo tình hình *hiện tại* cũng như sau *mỗi lần trồng thêm* ở một ô mới các khóm hoa có tạo thành một tập thuận lợi hay không, đưa ra báo cáo ngắn gọn YES hoặc NO.

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

- ✚ Dòng đầu tiên chứa 2 số nguyên h và w ($1 \leq h, w \leq 10^5$),
- ✚ Dòng thứ 2 chứa số nguyên n ($1 \leq n \leq 10^5$),
- ✚ Dòng thứ i trong n dòng sau chứa 2 số nguyên r_i và c_i ($1 \leq r_i \leq h, 1 \leq c_i \leq w$),
- ✚ Dòng tiếp theo chứa số nguyên q ($0 \leq q \leq 10^5$),
- ✚ Dòng thứ j trong q dòng tiếp theo chứa 2 số nguyên nr_j và nc_j ($1 \leq nr_j \leq h, 1 \leq nc_j \leq w$).

Kết quả: Đưa ra file văn bản TURTLES.OUT các báo cáo nhận được, mỗi báo cáo trên một dòng.

Ví dụ 1:

TURTLES.INP	TURTLES.OUT
5 10	NO
8	YES
1 4	YES
2 4	NO
2 5	YES
2 6	
1 6	
3 5	
3 4	
4 4	
4	
1 5	
2 7	
3 7	
3 6	

Ví dụ 2:

TURTLES.INP	TURTLES.OUT
3 3	YES
5	NO
1 1	NO
1 2	NO
1 3	YES
2 3	
3 3	
4	
2 1	
3 2	
3 1	
2 2	

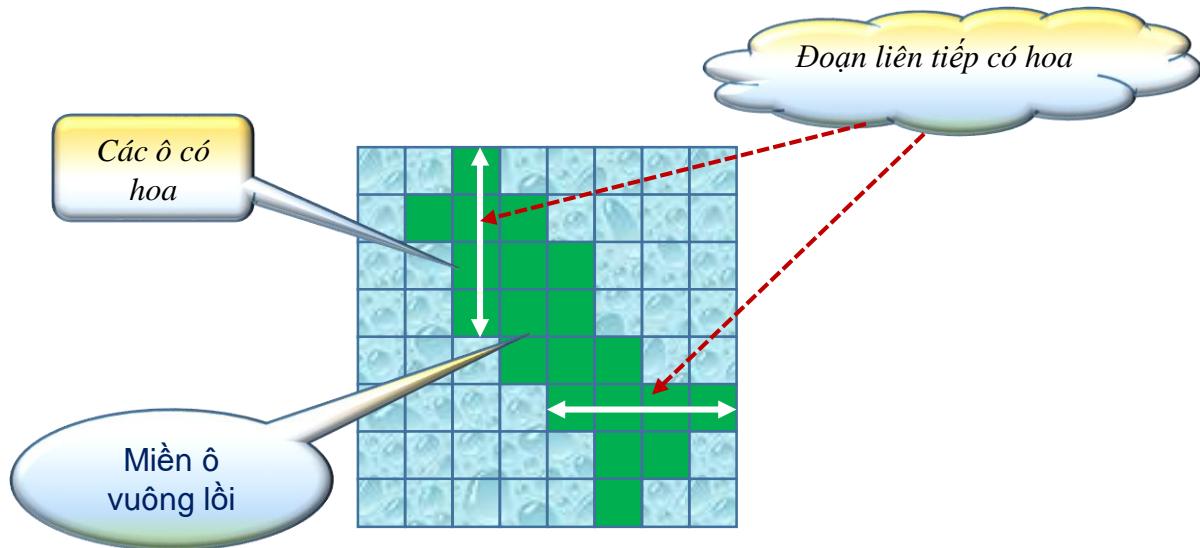


VnQ20 VKO201411301

Giải thuật:

Nhận xét: Nếu một ô có hoa là **thuận lợi** thì:

- ✚ Tất cả các ô giữa 2 ô có hoa trong cùng một hàng đều phải có hoa, như vậy các ô có hoa trong mỗi hàng phải tạo thành một **dãy ô kề nhau liên tiếp**,
- ✚ Tương tự như vậy đối với cột.



Tập cá ô có tính chất như vậy được gọi là **miền ô vuông lồi**.

Như vậy tập các ô có hoa là thuận tiện khi và chỉ khi nó tạo thành một miền ô vuông lồi.

Để kiểm tra tính lồi của một tập các ô, mỗi hàng thứ i ta phải quản lý:

- ✚ rt_i – tổng số lượng ô có hoa trong hàng i ,
- ✚ rl_i – vị trí trái nhất có hoa trong hàng,
- ✚ rr_i – vị trí phải nhất có hoa trong hàng,

Tương tự như vậy đối với cột j cần có 3 đại lượng ct_j , cl_j và cr_j .

Một tập ô là lồi khi và chỉ khi:

- ❖ Nếu $rt_i > 0$ thì $rt_i = rr_i - rl_i + 1$ với mọi i ,
- ❖ Nếu $ct_j > 0$ thì $ct_j = cr_j - cl_j + 1$ với mọi j .

Để tiện việc kiểm tra khi bổ sung thêm ô mới ta cần đánh dấu các dòng và các cột có ô trống đan xen giữa các ô có hoa bằng các mảng `int rtemp[100001], cemp[100001];`

$$rtemp_i = \begin{cases} 1 & \text{nếu có ô trống đan xen giữa các ô có hoa trong dòng } i \\ 0 & \text{trong trường hợp ngược lại.} \end{cases}$$

$$cemp_j = \begin{cases} 1 & \text{nếu có ô trống đan xen giữa các ô có hoa trong cột } j \\ 0 & \text{trong trường hợp ngược lại.} \end{cases}$$

Ngoài ra ta cần lưu $vtemp$ – tổng số dòng bị đánh dấu và $vcemp$ – tổng số cột bị đánh dấu.

Các mảng rt, rl, rc, ct, cl, cr được tạo ra khi ghi nhận tập ban đầu.

Lần kiểm tra đầu tiên (hàm ***check_0()***) cũng phục vụ cho việc xác định các mảng ***xemp*** và ***cemp*** cũng như tính các giá trị ***vramp*** và ***vcemp***.

Các lần kiểm tra tiếp theo được thực hiện bởi hàm ***check_q(u, v)***, trong đó chỉ xét và điều chỉnh các thông tin liên quan tới dòng ***u*** và các thông tin liên quan tới cột ***v***.

Ở cả 2 hàm, điều kiện để có tập lồi là ***vramp+vcemp==0***.

Như vậy, độ phức tạp của phần kiểm tra các ô bỗ sung là O(*q*).

Chương trình

```
#include <fstream>
#include <ctime>
#define NAME "turtles."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,h,w,q,x,y,vremp=0,vcemp=0;
int rt[100001]={0},rl[100001]={0},rr[100001]={0},ct[100001]={0},
    cl[100001]={0},cr[100001]={0};
int remp[100001]={0},cemp[100001]={0};
bool check_0()
{
    for(int i=1;i<=h;++i) if(rt[i]>0 && rt[i]<rr[i]-rl[i]+1)
        ++vremp, remp[i]=1;
    for(int j=1;j<=w;++j) if(ct[j]>0 && ct[j]<cr[j]-cl[j]+1)
        ++vcemp, cemp[j]=1;
    return vremp+vcemp==0;
}
bool check_q(int u, int v)
{
    if(rt[u]==rr[u]-rl[u]+1) {if(remp[u])--vremp, remp[u]=0;}
    else if(remp[u]==0)++vremp, remp[u]=1;
    if(ct[v]==cr[v]-cl[v]+1) {if(cemp[v])--vcemp, cemp[v]=0;}
    else if(cemp[v]==0)++vcemp, cemp[v]=1;
    return vremp+vcemp==0;
}

int main()
{ clock_t aa=clock();
    fi>>h>>w;
    fi>>n;
    for(int i=0;i<n;++i)
    {
        fi>>x>>y;
        ++rt[x];++ct[y];
        if(rl[x]==0) rl[x]=y;else if(rl[x]>y) rl[x]=y;
        if(rr[x]<y) rr[x]=y;
        if(cl[y]==0) cl[y]=x;else if(cl[y]>x) cl[y]=x;
        if(cr[y]<x) cr[y]=x;
    }
    if(check_0()) fo<<"YES\n";else fo<<"NO\n";
    fi>>q;
    for(int i=0; i<q;++i)
    {
        fi>>x>>y;
        ++rt[x];++ct[y];
        if(rl[x]==0) rl[x]=y;else if(rl[x]>y) rl[x]=y;
        if(rr[x]<y) rr[x]=y;
        if(cl[y]==0) cl[y]=x;else if(cl[y]>x) cl[y]=x;
        if(cr[y]<x) cr[y]=x;
        if(check_q(x,y)) fo<<"YES\n";else fo<<"NO\n";
    }

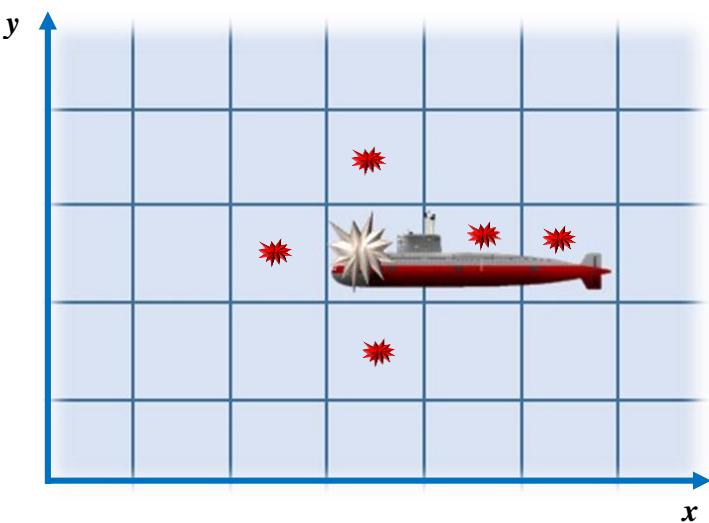
    clock_t bb=clock();
    fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ23. TÀU NGẦM

Tên chương trình: SUBMARINE.???

Trong một cuộc tập trận bảo vệ vùng biển tình huống giả định là cần tiêu diệt một tàu ngầm của đối phương thâm nhập vào vùng biển cần bảo vệ. Vùng biển này được chia thành các ô và tạo thành lưới ô vuông kích thước $n \times m$ ô. Tàu ngầm có k khoang và có thể coi như một hình chữ nhật kích thước $1 \times k$.

Sau khi bay khảo sát, một quả bom chống tàu ngầm (*depth charge*) được thả xuống ô có tọa độ (x, y) . Thật may mắn, có dấu hiệu cho thấy bom đã phá vỡ một khoang của tàu ngầm. Muốn đánh đắm tàu ngầm phải phá được tất cả các khoang của nó. Bom chống tàu ngầm rất đắt tiền và số lượng bom mà máy bay có thể mang cũng bị hạn chế, vì vậy phải xác định số lượng bom ít nhất cần thả thêm để đánh đắm tàu địch.



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

- ✚ Dòng đầu tiên chứa 2 số nguyên n và m ($1 \leq n, m \leq 20$),
- ✚ Dòng thứ 2 chứa 2 số nguyên x và y ($1 \leq x \leq n, 1 \leq y \leq m$),
- ✚ Dòng thứ 3 chứa số nguyên k ($1 \leq k \leq \max\{n, m\}$).

Kết quả: Đưa ra file văn bản SUBMARINE.OUT một số nguyên – số lượng bom ít nhất cần thả thêm.

Ví dụ:

SUBMARINE.INP
7 5
4 3
3

SUBMARINE.OUT
5



Giải thuật:

Nhận xét:

- ✚ Đây là bài toán nhận dạng,
- ✚ Nguyên tắc giải các bài toán nhận dạng:
 - ▣ Chuẩn hóa cấu hình: đưa về một hoặc một vài cấu hình chuẩn để giảm bớt các trường hợp cần phân tích,
 - ▣ Dùng kỹ thuật bảng phương án để phục vụ phân tích và lập trình,
 - ▣ Liệt kê và xử lý các trường hợp riêng,
 - ▣ Xét và xử lý trường hợp chung (các tình huống còn lại).

Xử lý:

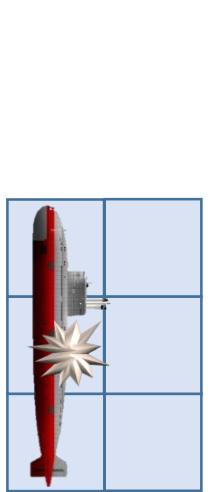
Chuẩn hóa cấu hình: đưa về trường hợp luôn có $n \leq m$:

```
if (n > m)
    swap(n, m), swap(x, y);
```

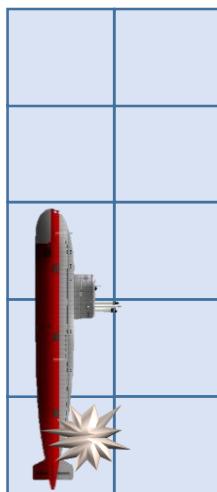
Các trường hợp riêng:

Trường hợp $k = 1$,

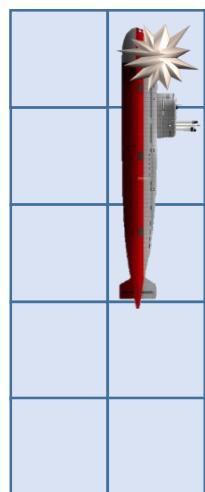
Trường hợp $n < k$: tàu chỉ có thể ở trong cột



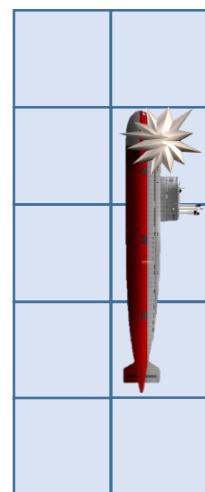
$ans = k-1$



$ans = k-1$

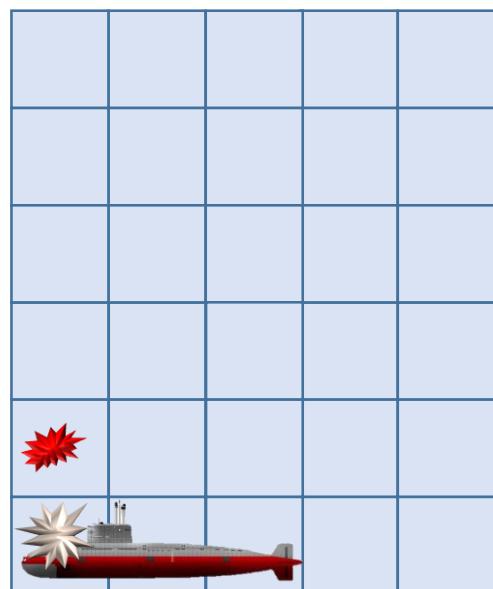
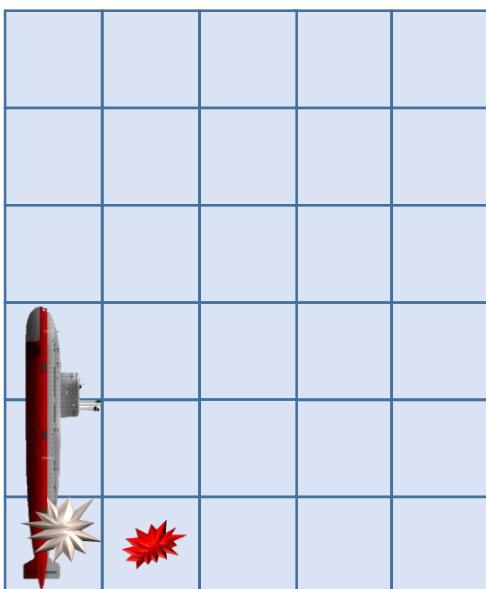
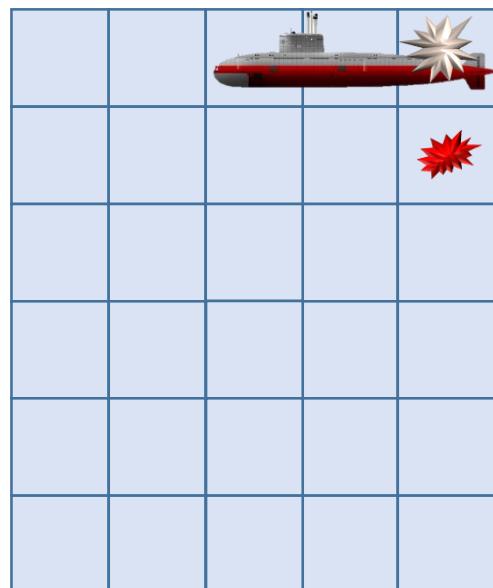
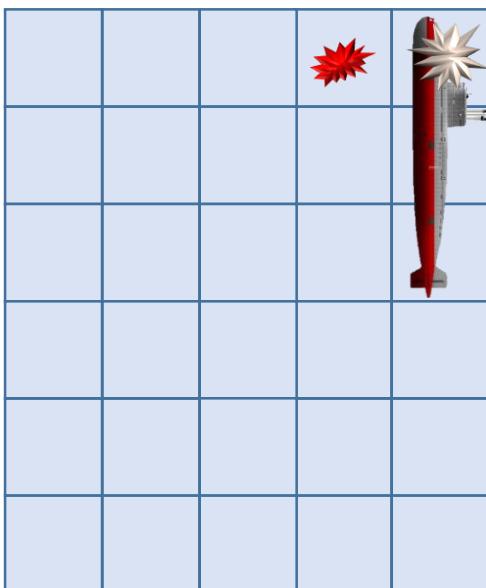
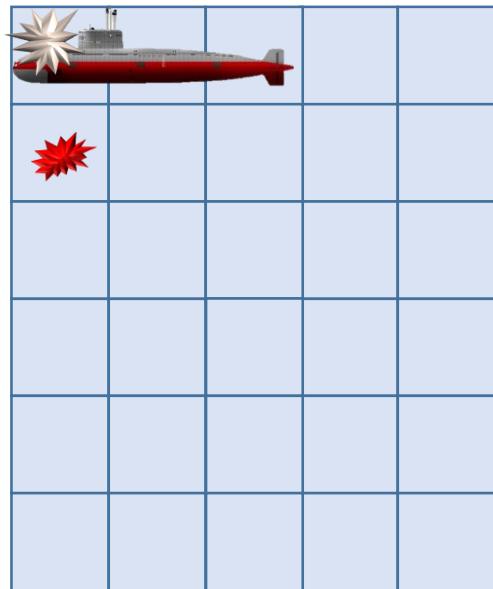
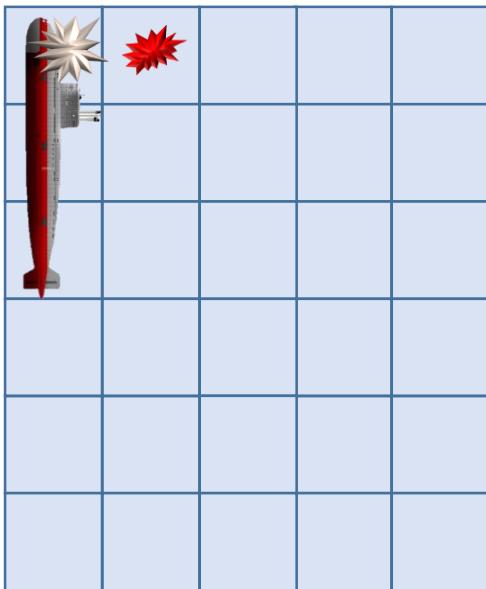


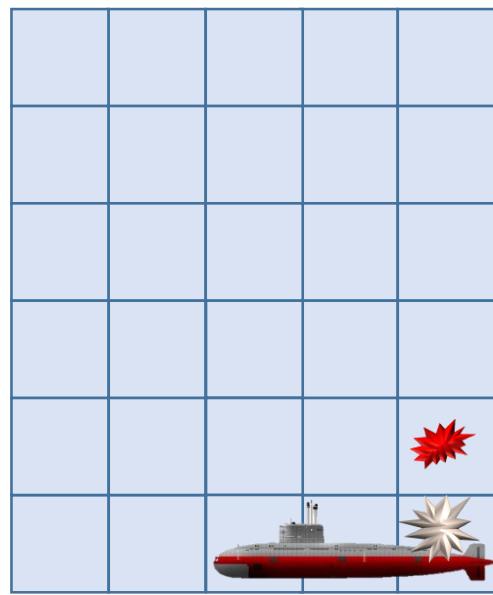
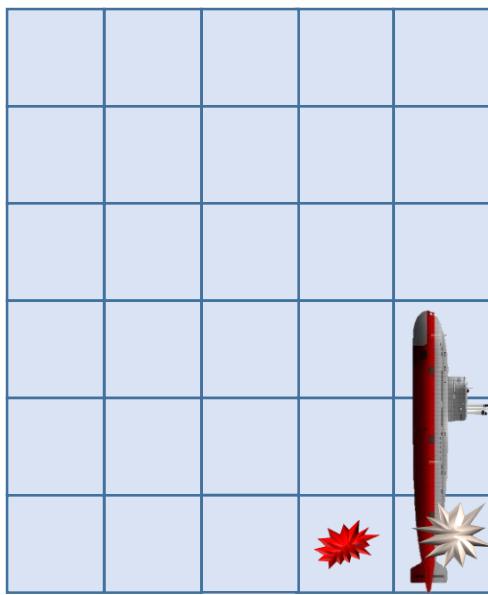
$ans = k-1$



$ans = k$

Trường hợp ô ban đầu ở góc:

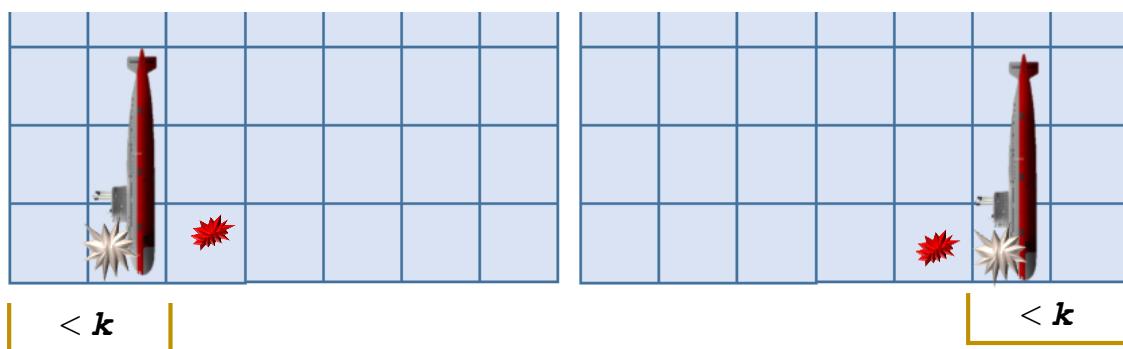
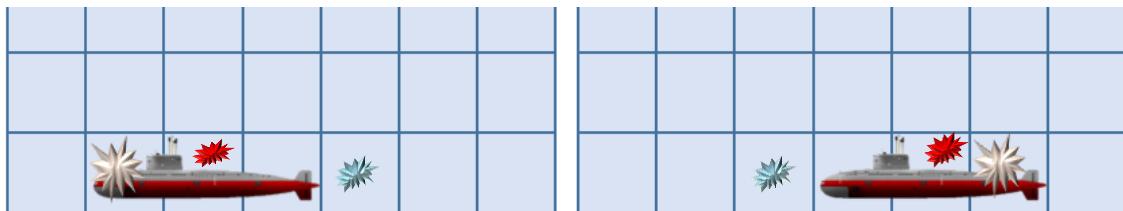
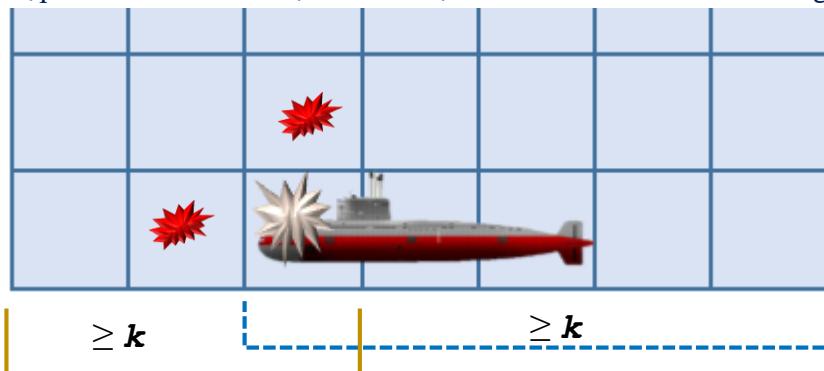




Cần một quả bom xác định tàu ở dòng hay cột, $\text{ans} = k$.

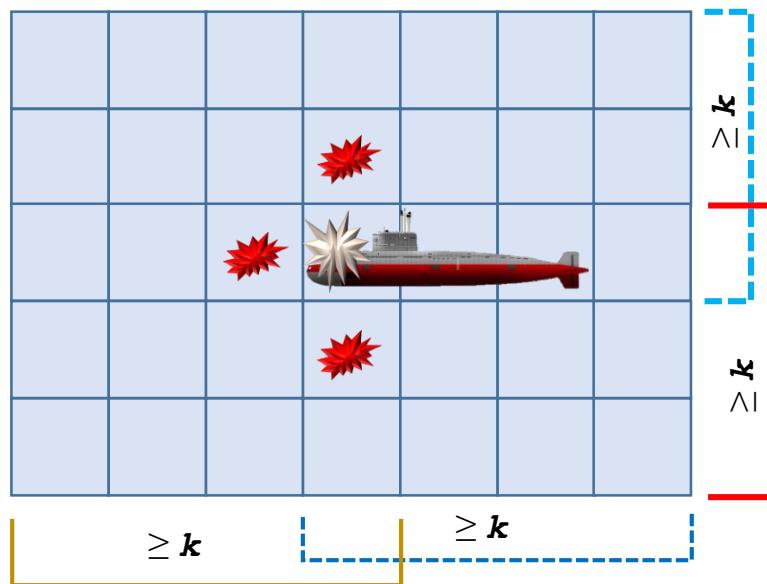
Trường hợp ô ban đầu trên cạnh (nhưng không ở góc):

Xét trường hợp ô ban đầu ở trên cạnh trên hoặc dưới của miền: Có 2 trường hợp con:

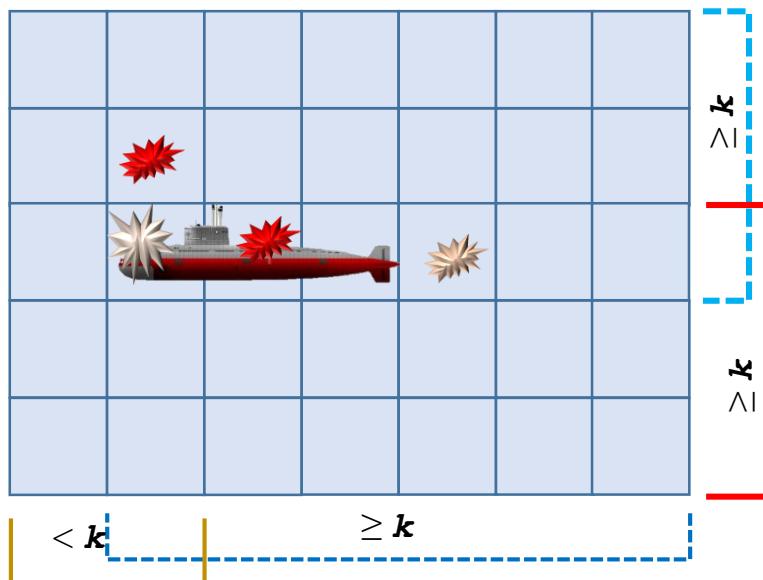


Trường hợp ô ban đầu ở trên cạnh trái hoặc phải của miền: xoay hình để đưa về trường hợp trên.

Trường hợp ô không nằm trên cạnh (nằm ở trong miền): hai trường hợp con:



Cần thêm 2 quả bom xác định hướng và một quả - xác định điểm cuối



Trường hợp có ít nhất một đoạn $< k$

Cần thêm một quả bom xác định hướng và một quả - xác định điểm cuối

Kiểm tra số bom cần thiết để nhận dạng tàu nằm theo chiều dọc hay theo chiều ngang, chọn hướng cần ít bom hơn.

Chương trình

```
#include <fstream>
#include <algorithm>
#define NAME "submarine."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,m,x,y,k,ans,dx1,dxr,dyl,dyr,drx0,drx1,dry0,dry1;

int analysis()
{
    if (k == 1) return 0;
    if (n < k)
        {if (m == k) return k-1;
        if(y==1 || y==m) return k-1; else return k;
        }
    // **** n>= k ****
    if ((x==1 || x==n) && (y==1 || y==m)) return k;
    if (x==1 || x==n || y==1 || y==m)
    {
        if (y == 1 || y == m)
            swap(n, m), swap(x, y);
        return k + (y >= k && (m - y + 1) >= k);
    }
    //inside
    drx0 = n > k;
    drx1 = n >= k;
    dx1 = (n - x + 1) >= k;
    dxr = x >= k;
    dry0 = m > k;
    dry1 = m >= k;
    dyl = (m - y + 1) >= k;
    dyr = y >= k;
    return k - 1 + min(drx0+dry1+dyl*dyr, dry0+drx1+dx1* dxr);
}

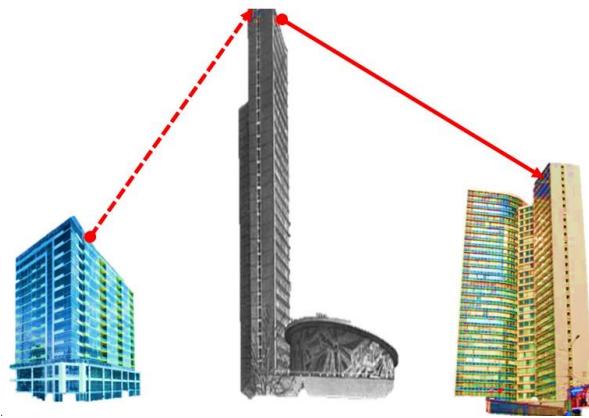
int main()
{
    fi>>n>>m >>x>>y>>k;
    if (n > m)
        swap(n, m), swap(x, y);
    ans=analysis();
    fo<<ans;

    return 0;
}
```

VQ24. ĐỘT KÍCH

Tên chương trình: SWOOP.???

Đội đặc nhiệm phải thường xuyên rèn luyện kỹ năng để thực hiện các nhiệm vụ đặc biệt khó khăn trong các tình huống khẩn trương và hết sức nguy hiểm. Nhiệm vụ của bài tập lần này là bí mật đột nhập vào một tòa nhà cao tầng từ trên nóc, trán áp bọn khủng bố đang bắt giữ con tin ở trong đó. Từ trụ sở của đội tới tòa nhà cần đột nhập nếu vạch một đường thẳng tới tòa nhà cần đột nhập thì có n tòa nhà cao tầng, nhà thứ i ở tọa độ x_i trên đường thẳng và có độ cao y_i , $i = 1 \dots n$, trụ ở ở tòa nhà thứ 1 và ngôi nhà cần đột nhập – thứ n . Các thành viên của đội leo lên trần của trụ sở, dùng súng ép hơi bắn dây móc lên tòa nhà cần tới, treo người theo dây móc leo tới tòa nhà này. Từ một tòa nhà có thể tới tòa nhà khác bất kỳ cao hơn hoặc thấp hơn nếu dây móc không chạm vào tòa nhà khác. Dây móc là loại dây kép có thể thu hồi khi tới nơi mới.



Hãy xác định độ dài dây móc ngắn nhất cần có để có thể tới được tòa nhà cần đột nhập.

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

- ✚ Dòng đầu tiên chứa số nguyên n ($2 \leq n \leq 10^5$),
- ✚ Dòng thứ i trong n dòng sau chứa 2 số nguyên x_i và y_i ($0 \leq x_i < x_{i+1} \leq 10^9$, $0 \leq y_i \leq 10^9$).

Kết quả: Đưa ra file văn bản SWOOP.OUT độ dài tìm được với độ chính xác 10^{-10} .

Ví dụ:

SWOOP.INP
3
0 10
5 15
10 10

SWOOP.OUT
7.071068



Giải thuật:

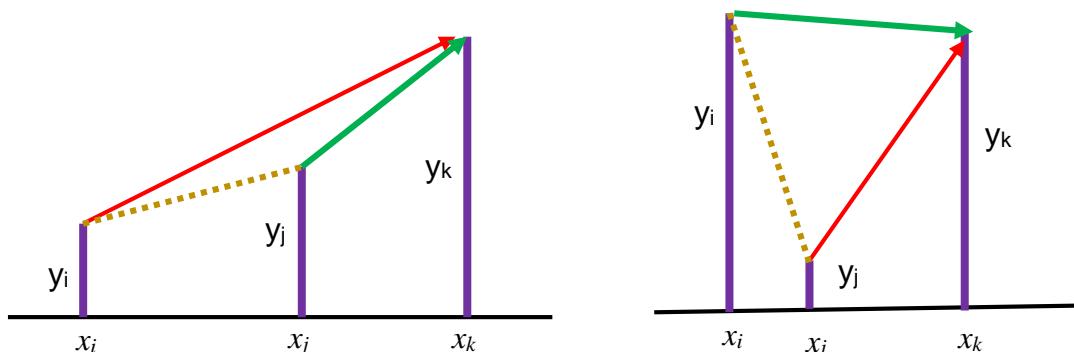
Nhận xét:

- ✚ Xét dây có độ dài u :
 - ❖ Nếu với dây có độ dài u có thể đi từ tòa nhà 1 đến n thì với các dây có độ dài lớn hơn u cũng có thể đi được,
 - ❖ Nếu với dây có độ dài u không thể đi từ tòa nhà 1 đến n thì với các dây có độ dài nhỏ hơn u cũng càng không thể đi được.
- ✚ Như vậy có thể dùng giải thuật tìm kiếm nhị phân để xác định độ dài ngắn nhất của đoạn dây cần thiết và số lần kiểm tra không vượt quá 32,
- ✚ Việc di chuyển được thực hiện theo từng bước và có thể chứng minh được là không cần quay lui vì vậy có thể dùng phương pháp quy hoạch động để giải, độ phức tạp của giải thuật không thay đổi, nhưng độ phức tạp lập trình sẽ lớn hơn so với việc sử dụng giải thuật tìm kiếm nhị phân.

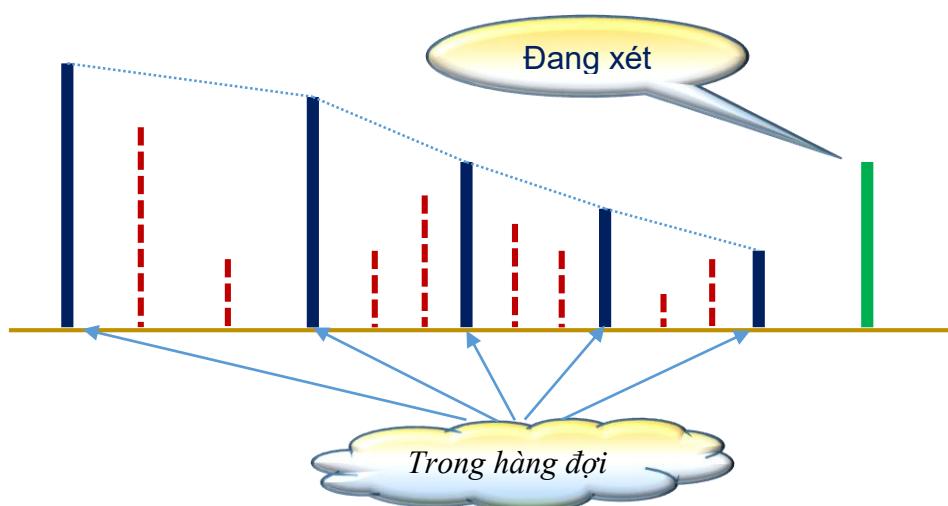
Kiểm tra dây độ dài u có thể được sử dụng hay không:

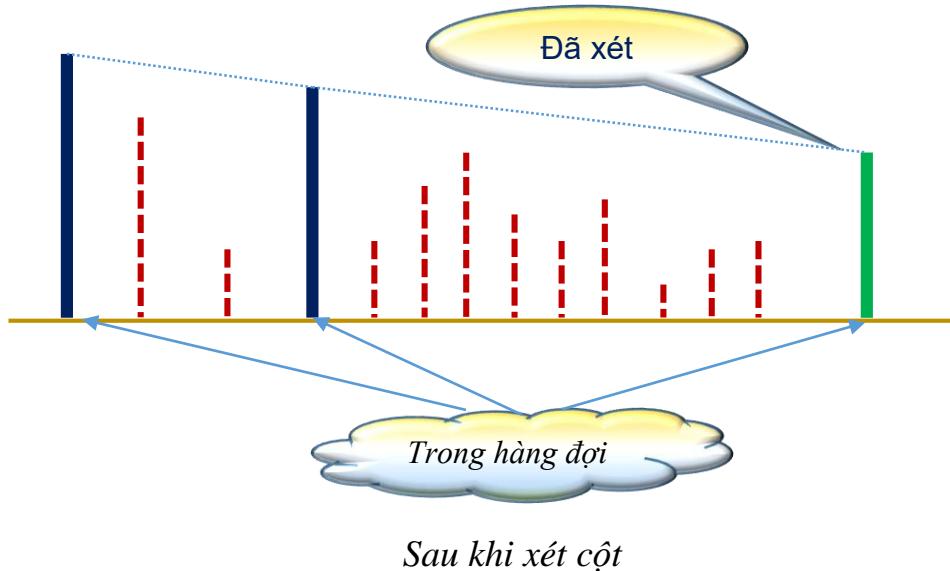
Xét 3 tòa nhà i, j, k , trong đó $i < j < k$.

Nếu từ i và từ j đều có thể trực tiếp tới k và $y_i < y_j$ thì đường đi từ j tới k sẽ ngắn hơn đường đi từ i tới k (tính chất độ dài cạnh trong tam giác), nhưng nếu $y_i > y_j$ thì *có thể* đường đi từ i tới k ngắn hơn đường đi từ j tới k .



Như vậy ta phải tổ chức một hàng đợi chỉ lưu trữ độ cao các tòa nhà đã duyệt theo thứ tự *giảm dần* của độ cao.





Trong quá trình xử lý ta phải truy nhập vào tất cả các phần tử trong đó vì vậy hàng đợi cần được tổ chức dưới dạng vector hoặc, để xử lý nhanh hơn – dạng mảng tĩnh.

Với mỗi tòa nhà cần một biến đánh dấu có thể tới được hay không. Kết quả kiểm tra sẽ là trạng thái tòa nhà cuối cùng: đến được hay không đến được.

Lưu ý:

- Để tránh ảnh hưởng của sai số làm tròn nên làm việc với bình phương của khoảng cách,
 - Cách đưa ra 10 chữ số sau dấu chấm thập phân trong chế độ vào ra fstream của C++.
- Độ phức tạp của giải thuật: $O(n)$.

Chương trình

```
#include <fstream>
#include <iomanip>
#include <cmath>
#include <ctime>
#define NAME "swoop."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,k,x[100000], y[100000],can[100000],st[100000];

int64_t dist(int a, int b)
{
    return ((int64_t)x[a] - x[b]) * (x[a] - x[b]) +
           ((int64_t)y[a] - y[b]) * (y[a] - y[b]);
}

int check(int64_t u) {
    for (int i = 0; i < n; ++i)
        can[i] = 0;
    can[0] = 1;k=0; st[k]=0;
    for(int i=1;i<n;++i)
    {
        while(k>=0 && y[st[k]]<=y[i])
        {
            can[i]|=(can[st[k]]>0 && (dist(st[k],i)<=u));
            --k;
        }
        if(k>=0) can[i]|=(can[st[k]] && (dist(st[k],i)<=u));
        st[++k]=i;
    }
    return can[n - 1];
}

int main()
{
    clock_t aa=clock();
    fi>>n;
    for (int i = 0; i < n; ++i)
        fi>>x[i]>>y[i];
    int64_t l = 0, r = (int64_t)1e18;
    while (r - l > 1) {
        int64_t m = (r + l) / 2;
        if (check(m))
            r = m;
        else
            l = m;
    }
    fo<<fixed<<setprecision(10)<<sqrt(r);
    clock_t bb=clock();
    fo<<"\nTime: "<<setprecision(3)<<(double) (bb-aa)/1000<<" sec";
}
```

VQ17. PHẢN VẬT CHẤT

Tên chương trình: ANTIMATTER.???

Các nhà vật lý đã chứng minh được sự tồn tại của phản vật chất. Đã có nhiều tác phẩm khoa học viễn tưởng khai thác về đề tài này như tiểu thuyết “*Pháo dài số*” của Dan Brown (tác giả của một loạt các tiểu thuyết nổi tiếng như *Mật mã Da Vinci*, *Điểm dối lừa*, *Biểu tượng thát truyền*, . . .) *Pháo dài số* và *Mật mã Da Vinci* đã được Hollywood chuyển thể thành phim rất ăn khách.

Các nhà tin học cũng không muốn chộm chân trong lĩnh vực này. Họ cố hình dung trong phản thế giới máy tính được xây dựng và hoạt động như thế nào. Mô hình lý thuyết phù hợp nhất là máy tính ở đó lưu trữ và xử lý thông tin theo cơ số -2. Số nguyên x sẽ được biểu diễn dưới

dạng $\mathbf{x} = \sum_{i=0}^{n-1} a_i (-2)^i$, trong đó a_i bằng 0 hoặc 1 và $a_{n-1} \neq 0$. Ví dụ $3 = 111_2$.

Cho số nguyên \mathbf{x} . Hãy xác định độ dài tối thiểu của dãy bít biểu diễn \mathbf{x} ở cơ số -2 và bắn thân các bít đó.

Dữ liệu: Vào từ file văn bản ANTIMATTER.INP gồm một dòng chứa số nguyên \mathbf{x} ($-10^{18} \leq \mathbf{x} \leq 10^{18}$).

Kết quả: Đưa ra file văn bản ANTIMATTER.OUT, dòng đầu tiên chứa số nguyên n , dòng thứ 2 chứa n số nguyên a_0, a_1, \dots, a_{n-1} - các bít tương ứng trong dạng biểu diễn \mathbf{x} ở cơ số -2.

Ví dụ:

ANTIMATTER.INP	ANTIMATTER.OUT
3	3 1 1 1



Giải thuật:

$$a_0 = \begin{cases} 0 & \text{nếu } n \text{ chẵn,} \\ 1 & \text{nếu } n \text{ lẻ.} \end{cases}$$

Sau khi đã tính được a_i , thay n bằng $(n - a_i) / (-2)$, tính a_{i+1} và lập lại khâu xử lý này cho đến khi $n = 0$.

Công thức tính a_i : $a_i = (n \% b + b2) \% b$, trong đó $b = -2$, $b2 = 2$.

Chương trình:

```
#include <iostream>
#include <vector>
#define NAME "antimatter."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
const int b = -2,b2 = 2;

int main()
{
    int64_t n;
    fi>> n;
    vector <int> ans;
    while (n != 0)
    {
        int r = (n % b + b2) % b;
        ans.push_back(r);
        n = (n - r) / b;
    }
    if (ans.size() == 0)
        ans.push_back(0);
    int m=(int)ans.size();
    fo<<m<<'\
n';
    for(int i=0;i<m;++i) fo<<ans[i]<<' ';
}
```

VQ26. PHÒNG THI

Tên chương trình: HALL.???

Để chuẩn bị cho lễ bế mạc và trao giải một cuộc thi tin học người ta quyết định dùng các tấm vách lắp thành một căn phòng hình chữ nhật có kích thước các cạnh là nguyên trên sân bóng của nhà trường. Để khi mọi người vào ngồi trông không loãng căn phòng cần có diện tích trong đoạn từ **a** đến **b** (mét vuông).

Trên tường của phòng người ta dự tính treo các pano giới thiệu thành tích của các thí sinh và đơn vị dự thi. Để các pano trông không quá thưa chu vi căn phòng phải nằm trong đoạn từ **c** tới **d** mét.

Với **a**, **b**, **c**, **d** cho trước hãy xác định xem có bao nhiêu loại phòng khác nhau có thể lắp ráp. Hai phòng kích thước **x**×**y** và **y**×**x** được coi là cùng một loại.

Dữ liệu: Vào từ file văn bản HALL.INP gồm một dòng chứa 4 số nguyên **a**, **b**, **c** và **d** ($1 \leq a \leq b \leq 10^9$, $4 \leq c \leq d \leq 10^9$).

Kết quả: Đưa ra file văn bản HALL.OUT một số nguyên – số loại phòng khác nhau có thể lắp ráp.

Ví dụ:

HALL.INP
2 10 4 8

HALL.OUT
3



Giải thuật:

Gọi x và y – hai cạnh của phòng.

Xây dựng hàm $f(u, v)$ tính số lượng hình chữ nhật có các cạnh là x, y khác nhau thỏa mãn:

- ❖ $x \leq y$,
- ❖ $xy \leq u$,
- ❖ $2x(x+y) \leq v$.

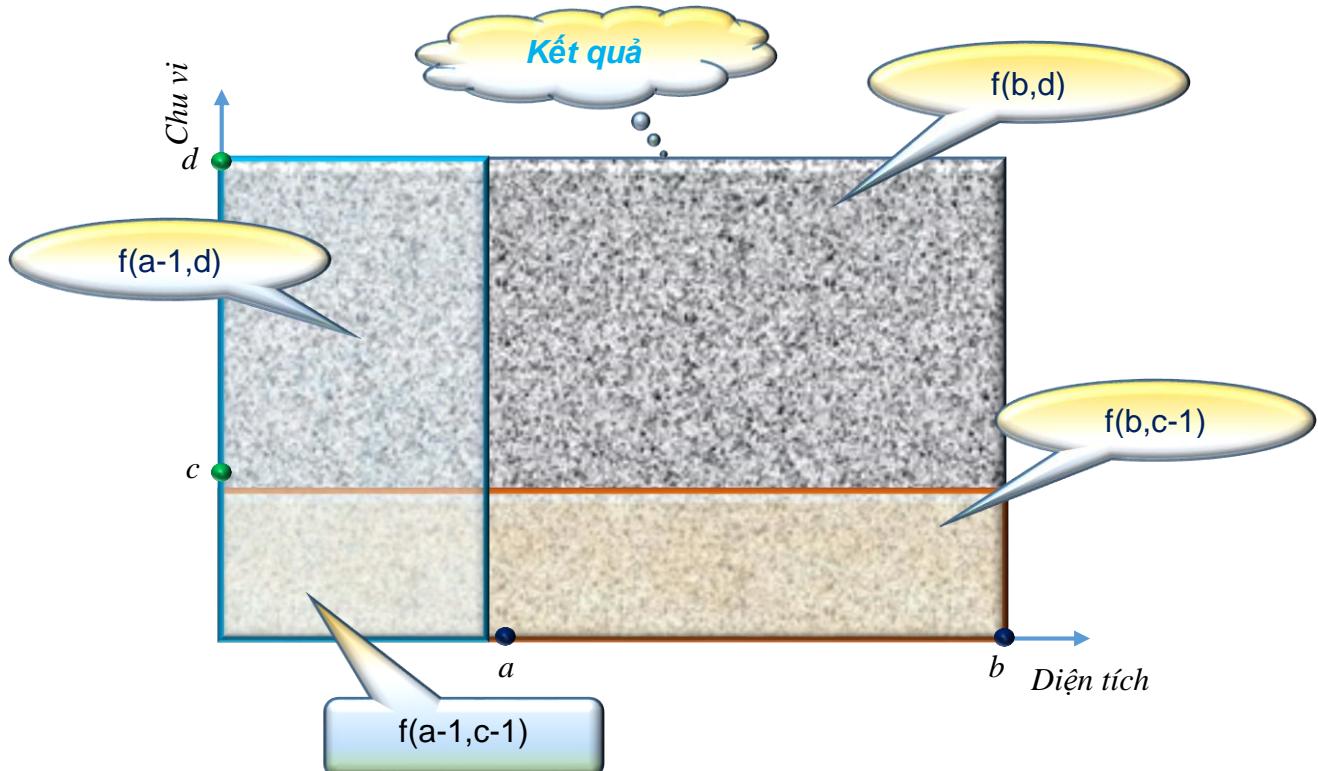
Ta chỉ cần duyệt các giá trị x thỏa mãn $x^2 \leq u$.

Với mỗi x trong khoảng cần duyệt cần có:

- ❖ $y \leq t1 = u/x$,
- ❖ $y \leq t2 = v/2 - x$.

Nếu $sl = \min(t1, t2) \geq x$ thì số lượng hình chữ nhật thỏa mãn điều kiện tìm kiếm sẽ là $sl = x+1$.

Lời giải của bài toán: $ans = f(b, d) - f(b, c-1) - f(a-1, d) + f(a-1, c-1)$.



Chương trình:

```
#include <fstream>
#include <ctime>
#define NAME "hall."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int a, b,c,d;
int64_t ans;

int64_t f(int u,int v)
{int y,t,t1,t2;
 int64_t sq,rf=0;
 t=v/2;
 for(int x=1;x<=t;++x)
 {
    sq=x*x;
    if(sq>u)break;
    t1=u/x; t2=t-x;
    t1=min(t1,t2);
    if(t1>=x)rf+=(t1-x+1);
 }
 return rf;
}

int main()
{fi>>a>>b>>c>>d;
 ans=f(b,d)-f(b,c-1)-f(a-1,d)+f(a-1,c-1);
 fo<<ans;
}
```

VQ27. PHẦN THƯỞNG

Tên chương trình: PRIZES.???

Harry và Hermione thắng cuộc trong một trò chơi truyền hình và bây giờ là giai đoạn trao thưởng. Quy tắc trao thưởng là như sau: có n phần thưởng xếp thành một hàng dài, đánh số từ 1 đến n . Tùy thuộc vào số điểm đã đạt được của cặp chơi, người dẫn chương trình sẽ nói một số k ($1 \leq k \leq n/3$). Một người chơi sẽ chọn cho mình k phần thưởng xếp liên tiếp nhau, người thứ hai cũng sẽ chọn cho mình k phần thưởng xếp liên tiếp nhau *trong số còn lại*. Hermione là nữ nên được ưu tiên chọn trước.

Trò chơi đã kết thúc. Bây giờ không cần phải đồng tâm hiệp lực. Hermione vẫn còn rất giận Harry về một câu nói vô tâm mà chắc bây giờ Harry cũng không nhớ là mình nói cái gì và khi nào. Hermione hiểu rất rõ giá trị mỗi phần thưởng đối với Harry, cụ thể là phần thưởng thứ i sẽ có giá trị a_i , $i = 1 \div n$ và quyết định cách chọn của mình sao cho tổng giá trị phần thưởng mà Harry có thể đạt được càng nhỏ càng tốt. Về tổng giá trị phần thưởng của mình, Hermione không mấy may quan tâm!

Hãy xác định x – tổng nhỏ nhất giá trị phần thưởng mà Hermione có thể chọn để Harry không có cách chọn phần thưởng với tổng giá trị lớn hơn x .

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

- ✚ Dòng đầu tiên chứa 2 số nguyên n và k ($3 \leq n \leq 10^5$, $1 \leq k \leq n/3$),
- ✚ 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 PRIZES.OUT số nguyên x .

Ví dụ:

PRIZES.INP
10 2
1 2 4 5 2 4 2 2 1 6

PRIZES.OUT
7



Vq27 ROI_reg20150124 B

Giải thuật:

Nhận xét:

- ✚ Đây là bài toán sử dụng các hàm tổng tiền tố và tổng hậu tố,
- ✚ Khoảng cần tính tổng không thay đổi, vì vậy không cần quản lý các tổng này bằng các cây.

Nếu Hermione chọn các phần thưởng từ p đến $p+k-1$ thì tổng giá trị t có được sẽ là:

$$\begin{aligned} t &= a_p + a_{p+1} + \dots + a_{p+k-1} \\ &= s_{p+k-1} - s_{p-1}, \end{aligned}$$

trong đó $s_0 = 0$, $s_i = s_{i-1} + a_i$, $i = 1 \div n$.

Harry có thể chọn phần thưởng cho mình ở bên trái hay bên phải dây Hermione đã chọn.

Nếu chọn bên trái thì tổng giá trị nhận được sẽ là

$$vleft_p = \max\{vleft_{p-1}, s_{p-1} - s_{p-k}\}$$

Nếu chọn bên phải thì tổng giá trị nhận được sẽ là

$$vright_{p+k} = vright_{p+k}$$

Giữa 2 giá trị này cần chọn giá trị lớn nhất.

Trong các công thức trên:

$$vleft_i = \begin{cases} 0, i = 0 \div k-1 \\ \max\{vleft_{i-1}, s_i - s_{i-k}\}, i = k \div n-k+1 \end{cases}$$

$$vright_i = \begin{cases} 0, i = n \div n-k+1 \\ \max\{vright_{i+1}, s_{i+k-1} - s_{i-1}\}, i = n-k+1 \div k+1 \end{cases}$$

Để tìm x cần duyệt với mọi i từ 1 đến $n-k+1$, tính

$$x = \min\{x, \max\{vleft_{i-1}, vright_{i+k}\}\}$$

Độ phức tạp của giải thuật: $O(n)$.

Chương trình:

```
#include <fstream>
#include <ctime>
#define NAME "prizes."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,k,t;
int64_t
s[100001],vleft[100001]={0},vright[100001]={0},x=(int64_t)2e18;;
double a1,a2,b1,b2,t1,t2;

int main()
{clock_t aa=clock();
 fi>>n>>k; s[0]=0;
 for(int i=1;i<=n;++i){fi>>t;s[i]=s[i-1]+t;}
 for (int i = k; i <= n; ++i)
    vleft[i] = max(vleft[i - 1], s[i] - s[i - k]);
 for (int i = n - k + 1; i >= 1;--i)
    vright[i] = max(vright[i + 1], s[i + k - 1] - s[i - 1]);
 for (int i = 1; i <= n - k + 1; i++)
    x = min(x, max(vleft[i - 1], vright[i + k]));
 fo<<x;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ29. ĐƯỜNG VÀNH ĐAI

Tên chương trình: CIRCLE.???

Đường vành đai thành phố có hệ thống xe lửa trên cao, chạy vòng tròn và cả về 2 phía, bao thành phố dưới dạng một vòng tròn. Có n ga nằm cách đều nhau. Khoảng cách giữa 2 ga được gọi là một cung đoạn. Hành khách thường chọn cho mình hướng đi để số cung đoạn tới ga cần đến là ít nhất. Số cung đoạn ít nhất nối 2 ga được gọi là khoảng cách của 2 ga đó.

Alisa và Bob sống ở gần đường vành đai và thường sử dụng phương tiện giao thông này. Một hôm Bob nhận thấy rằng nếu lên tàu ở một ga x nào đó thì dễ dàng tính được khoảng cách d_a từ ga đó tới nhà Alisa và khoảng cách d_b – tới nhà Bob, ngược lại nếu biết các khoảng d_a và d_b mà mỗi người đã đi thì cũng xác định được ga lên x .

Lúc đầu 2 bạn rất thích thú với tính chất đã phát hiện ra, coi cặp ga của mình là độc đáo và kể về “phát kiến” của mình với bố mẹ. Bố của Bob khen con chịu khó quan sát nhưng lưu ý rằng còn nhiều cặp ga khác có tính chất như vậy và đề xuất Bob lúc rảnh tính số lượng các cặp ga có cùng tính chất.

Hãy xác định kết quả mà, có thể Bob sẽ tính được.

Dữ liệu: Vào từ file văn bản CIRCLE.INP gồm một dòng chứa số nguyên n ($3 \leq n \leq 40\,000$).

Kết quả: Đưa ra file văn bản CIRCLE.OUT một số nguyên – số các cặp ga cùng tính chất đã nêu,

Ví dụ:

CIRCLE.INP
4

CIRCLE.OUT
8



VQ269 ROI_ROI_reg20150126 5

Giải thuật:

Nhận xét: Bài toán đơn giản, rèn luyện kỹ năng phân tích và đoán nhận giải thuật.

Các tình huống:

- ✚ Alisa và Bob có nhà ở cùng một ga, ví dụ, ga 1. Cặp giá trị (d_a, d_b) của các ga trái và phải bên cạnh đều là (1,1) – không thể xác định được ga lén,
- ✚ Nếu n là chẵn và nhà của Alisa ở ga 1, còn Bob - ở ga $n/2$. Cặp giá trị (d_a, d_b) của các ga trái và phải của ga 1 đều là (1, $n/2-1$) – không thể xác định được ga lén,
- ✚ Các trường hợp còn lại – có thể xác định đơn trị ga lén.

Như vậy kết quả ans sẽ là:

$$\text{ans} = \begin{cases} n(n-2) & \text{nếu } n \text{ chẵn,} \\ n(n-1) & \text{nếu } n \text{ lẻ.} \end{cases}$$

Chương trình:

```
#include <iostream>
#define NAME "circle."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n, ans;

int main()
{fi>>n;
 ans=n*(n-2);
 if(n&1) ans+=n;
 fo<<ans;
}
```

Thành phố quyết định chặt hạ hết n cây xanh hiện có trong thành phố để trồng một loại cây mới duy nhất. Nhiệm vụ được giao cho Công ty Cây xanh thành phố. Do hạn chế về thiết bị Công ty chỉ tổ chức được 2 đội đốn hạ cây. Đội I hạ được a cây mỗi ngày, nhưng cứ mỗi ngày thứ k thì phải nghỉ để bảo dưỡng kỹ thuật, tức là đội I sẽ nghỉ vào các ngày $k, 2k, 3k, \dots$. Đội II hạ được b cây mỗi ngày, nhưng cứ mỗi ngày thứ m thì phải nghỉ để bảo dưỡng kỹ thuật, tức là đội II sẽ nghỉ vào các ngày $m, 2m, 3m, \dots$. Ở ngày nghỉ, số cây chặt hạ của đội sẽ là 0. Cả hai đội bắt đầu công việc vào cùng một ngày và làm việc song song với nhau.

Công việc trồng cây mới sẽ bắt đầu sau khi toàn bộ cây cũ đã bị đốn hạ.

Hãy xác định sau bao nhiêu ngày thì có thể bắt đầu việc trồng mới cây.

Dữ liệu: Vào từ file văn bản RENEWED.INP gồm một dòng chứa 5 số nguyên a, k, b, m và n ($1 \leq a, b \leq 10^9, 2 \leq k, m \leq 10^{18}, 1 \leq n \leq 10^{18}$).

Kết quả: Đưa ra file văn bản RENEWED.OUT một số nguyên – số ngày tính được.

Ví dụ:

RENEWED.INP
2 4 3 3 25

RENEWED.OUT
7



Giải thuật:

Nhận xét:

- ✚ Số cây chặt được tăng dần theo thời gian, vì vậy số ngày hoàn thành xong công việc có thể tính bằng phương pháp tìm kiếm nhị phân,
- ✚ Sau d ngày số cây mỗi đội chặt được là như sau:
 - ✚ Đội I: $c_1 = a * (k-1) * (d/k) + a * (d \% k)$
 - ✚ Đội II: $c_2 = b * (m-1) * (d/m) + b * (d \% m)$
- ✚ Cận trái và phải để tìm kiếm: 0 và $2 * n / (a+b) + 1$

Chương trình:

```
#include <fstream>
#define NAME "renewed."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int64_t n,a,b,k,m,ans;

int main()
{fi>>a>>k>>b>>m>>n;
 int64_t l,r,mid,c1,c2;
 l=0; r=2*n/(a+b)+1;
 while (l+1<r)
 {
     mid=(l+r)/2;
     c1 =a*(k-1)*(mid/k)+a*(mid%k);
     c2 =b*(m-1)*(mid/m)+b*(mid%m);
     if (c1+c2>=n) r=mid; else l=mid;
 }
 ans=r;
 fo<<ans;
}
```

VQ31. LÁT NỀN

Tên chương trình: TILING.???

Viện Công nghệ tính toán hiệu năng cao được tu sửa và nâng cấp. Một trong những hạng mục công việc là lát lại hành lang nối từ phòng làm việc sang phòng đặt server. Hành lang có độ dài n và độ rộng 2. Để lát người ta dùng các viên gạch men loại kích thước 1×1 và kích thước 1×2 với số lượng dự trữ không hạn chế. Các viên gạch 1×2 có thể lát dọc hoặc xoay ngang. Trước đây hành lang được lát bằng các viên gạch kích thước 1×1 và dưới một số viên gạch có lắp các thiết bị điện tử khác nhau. Ban Giám đốc Viện không muốn lắp lại hệ thống điện tử vốn đang hoạt động rất hữu hiệu nên yêu cầu đánh dấu những viên này và không được bóc chúng lên trong quá trình lát nền. Có tất cả k viên như vậy, viên thứ i ở vị trí (x_i, y_i) , $1 \leq x_i \leq n$, $1 \leq y_i \leq 2$, $i = 1 \dots k$.

Bộ phận thi công phàn nàn về yêu cầu trên vì như thế sẽ hạn chế khả năng lát. Điều này làm Trưởng phòng vật tư tức giận và đề nghị bộ phận lập trình tính số lượng phương án khác nhau lát nền mà vẫn đảm bảo yêu cầu đã nêu để bên thi công thấy họ vẫn còn vô số cách làm khác nhau!

Hãy tính và đưa ra số cách lát theo mô đun $10^9 + 7$. Hai phương án gọi là khác nhau ở một chỗ nào đó theo một phương án được phủ bằng gạch 1×1 , còn theo phương án khác – được phủ bằng gạch 1×2 .

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

- ➡ Dòng đầu tiên chứa 2 số nguyên n và k ($1 \leq n \leq 10^5$, $0 \leq k < 2n$),
- ➡ Dòng thứ i trong k dòng sau chứa 2 số nguyên x_i và y_i .

Kết quả: Đưa ra file văn bản TILING.OUT một số nguyên – số cách lát theo mô đun $10^9 + 7$.

Ví dụ:

TILING.INP
3 1
2 1

TILING.OUT
8



Giải thuật:

Nhận xét: Hành lang có thể được chia thành các cột độ rộng 1. Số cách lát nền từ đầu tới cột thứ i có thể tính được dựa vào số cách lát từ đầu tới 2 cột trước vì vậy đây là bài toán sử dụng công thức lặp (còn gọi là quy hoạch động đơn giản).

Tổ chức dữ liệu:

- ✚ Mảng **int f[100001]**: f_i – số cách lát từ đầu đến cột thứ i , $i=0, 1, 2, \dots, n$, $f_0 = 1$,
- ✚ Mảng **int down[100001]**: $down_i$ – số cách lát từ đầu đến ô dưới của cột thứ i , $i=0, 1, 2, \dots, n$, $down_0 = 0$,
- ✚ Mảng **int up[100001]**: up_i – số cách lát từ đầu đến ô trên của cột thứ i , $i=0, 1, 2, \dots, n$, $up_0 = 0$,
- ✚ Mảng **bool blocked[2][100000]**: $blocked_{*,i}$ – đánh dấu các ô cấm thay đổi ở cột $i+1$, $i=0, 1, 2, \dots, n-1$.

Xét các tính huống ở cột i :

Trường hợp 3: cả 2 ô của cột đều bị cấm ($blocked_{0,i}=\text{true}$, $blocked_{1,i}=\text{true}$),

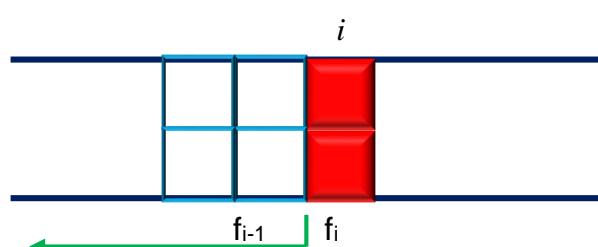
Trường hợp 2: chỉ có ô dưới bị cấm ($blocked_{0,i}=\text{true}$, $blocked_{1,i}=\text{false}$),

Trường hợp 1: chỉ có ô trên bị cấm ($blocked_{0,i}=\text{false}$, $blocked_{1,i}=\text{true}$),

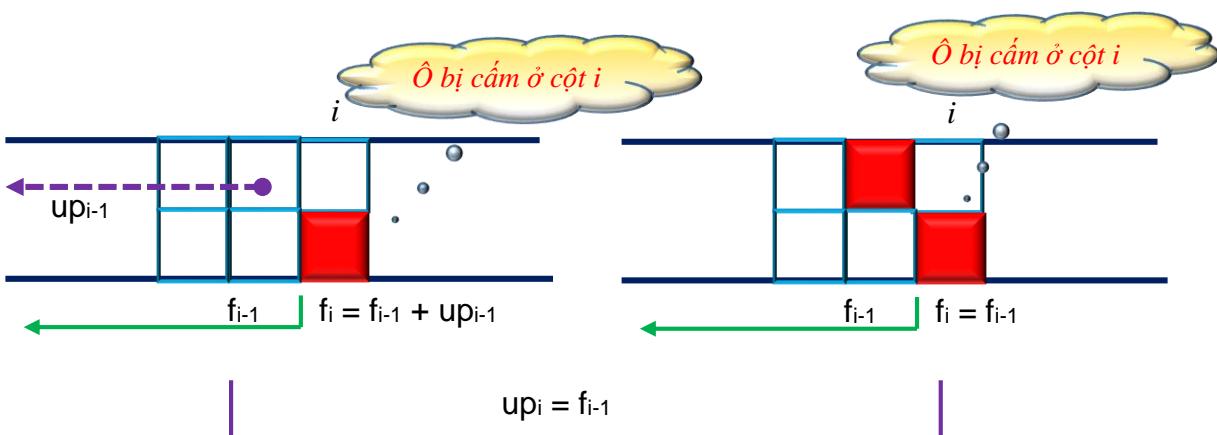
Trường hợp 0: không có ô nào bị cấm ($blocked_{0,i}=\text{false}$, $blocked_{1,i}=\text{false}$).

Xử lý tình huống:

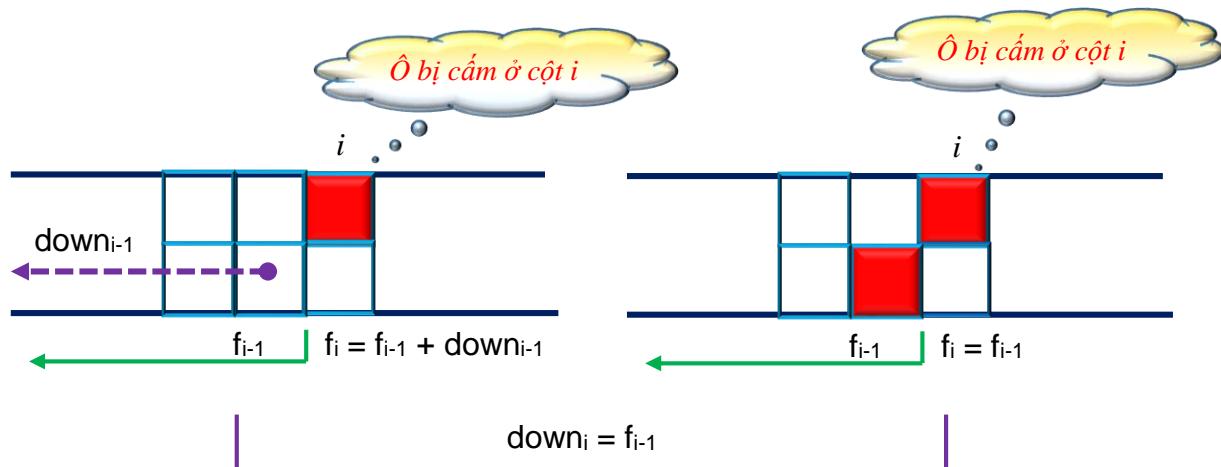
Trường hợp 3: số cách lát không thay đổi, $f_i = f_{i-1}$.



Trường hợp 2: chỉ có ô dưới bị cấm

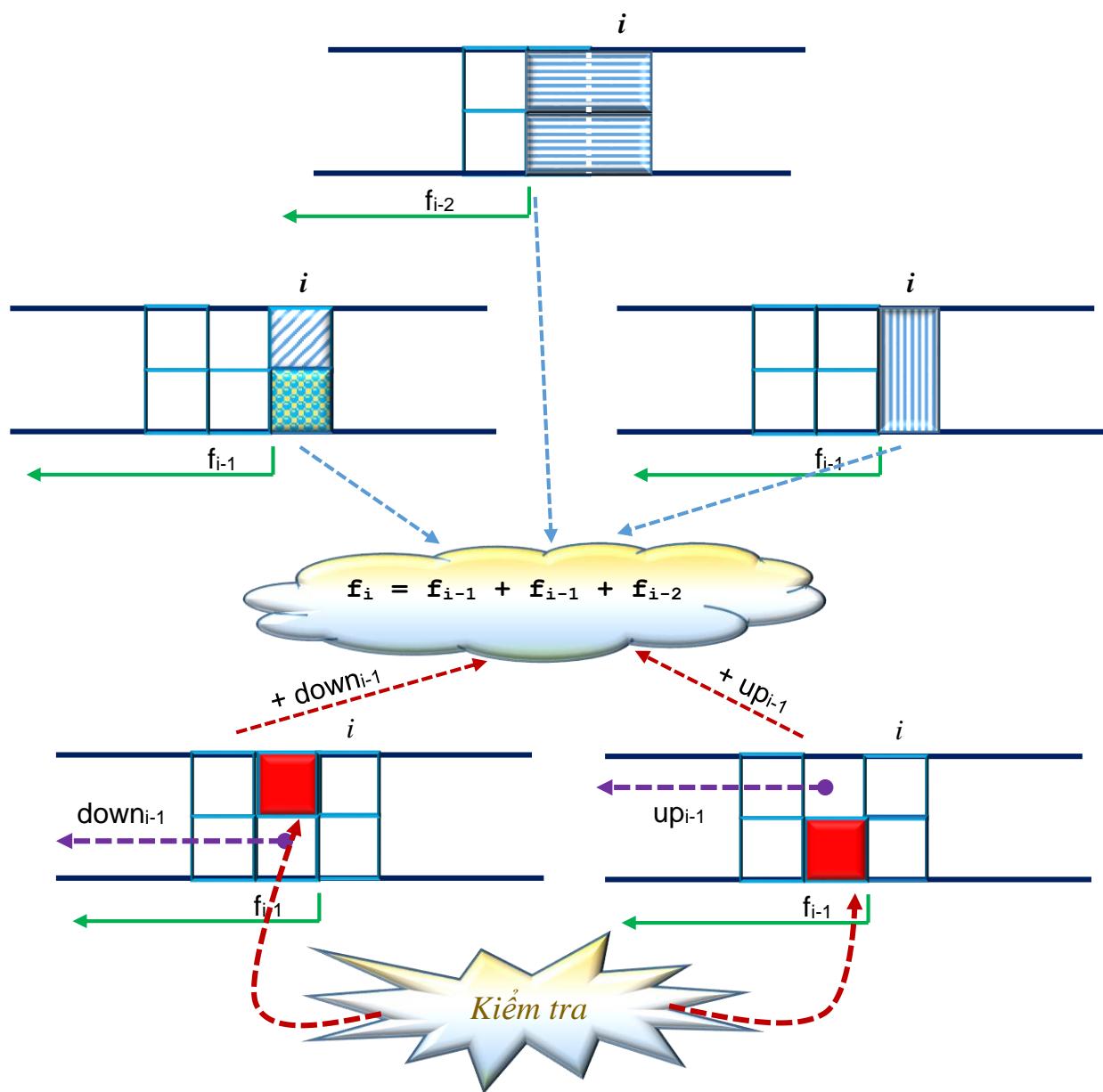


Trường hợp 1: chỉ có ô trên bị cấm

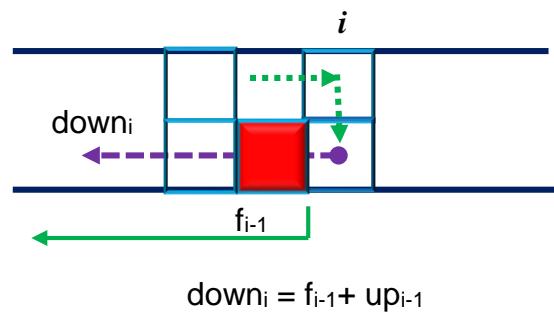
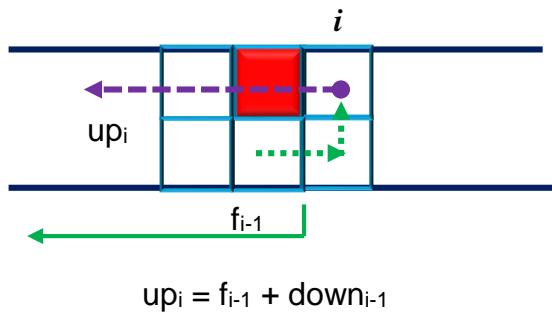


Trường hợp 0: không có ô nào bị cấm

❖ Tính f_i :



❖ Cập nhật up_i , down_i :



Lưu ý: phân biệt và xử lý trường hợp $i = 1$.

Độ phức tạp của giải thuật: $O(n)$.

Chương trình:

```
#include <fstream>
#include <ctime>
#define NAME "tiling."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int const N = 100001;
int const MOD = 1000000007;

int n,k,t;
bool blocked[2][N];
int f[N], up[N], down[N];

template <typename tt>
void upd(tt & a) {
    while (a >= MOD) {
        a -= MOD;
    }
}

int main()
{clock_t aa=clock();
 fi>>n>>k;
 for (int x, y, i = 0; i < k; ++i)
 {
    fi>>x>>y;
    blocked[y - 1][x - 1] = true;
 }
 f[0] = 1; up[0]=0; down[0]=0;
 for (int i = 1; i <= n; ++i)
 {   t=blocked[0][i - 1]<<1 | blocked[1][i - 1];
    switch(t)
    {case 3:f[i] = f[i - 1]; break;
     case 2:{ f[i] = f[i - 1];
               if (i == 1 || !blocked[1][i - 2])
               {
                  f[i] += up[i - 1];
                  upd(f[i]);
               }
               up[i] = f[i - 1]; break;
            }
     case 1:{ f[i] = f[i - 1];
               if (i == 1 || !blocked[0][i - 2])
               {
                  f[i] += down[i - 1];
                  upd(f[i]);
               }
               down[i] = f[i - 1]; break;
            }
     case 0:{int64_t res = 0;
              res += 2 * f[i - 1];
              if (i == 1 || !blocked[1][i - 2]) res += up[i - 1];
              if (i == 1 || !blocked[0][i - 2]) res += down[i - 1];
              if (i > 1 && !blocked[0][i - 2] && !blocked[1][i - 2])
                  res += f[i - 2];
              upd(res);
              f[i] = res;
              res = f[i - 1];
              if (i == 1 || !blocked[1][i - 2])res += up[i - 1];
              upd(res);
              down[i] = res;
            }
    }
}
```

```
    res = f[i - 1];
    if (i == 1 || !blocked[0][i - 2]) res += down[i - 1];
    upd(res);
    up[i] = res; break;
}
}
fo<<f[n];
clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ32. VUI HAY BUỒN

Tên chương trình: HAPPY.???

Các nhà khảo cổ học đang tiến hành khai quật một di chỉ chứa dấu vết của một nền văn minh cổ đại. Người ta tìm thấy một tấm bia với các văn tự cổ. Kết quả khảo sát cho thấy nội dung của nó, theo cách hiểu của chúng ta, là một dãy các số a_1, a_2, \dots, a_n mà hóa một thông tin nào đó. Đó là thói quen của nền văn minh này. Người ta thường mã hóa thông tin để chỉ có người của bộ tộc mình mới có thể hiểu được. Công cuộc giải mã vẫn đang được tiến hành, nhưng người ta biết được rằng nếu nội dung thông tin liên quan tới một sự kiện vui mừng như thắng trận hay một hoàng tử ra đời thì trong thông báo sẽ có nhiều số chẵn hơn số lẻ, ngược lại, nếu đó là một sự kiện đau buồn thì số chẵn sẽ không nhiều hơn số lẻ.

Hãy xác định xem thông tin tìm được liên quan tới loại sự kiện nào và đưa ra thông báo tương ứng là **Happy** hoặc **Sad**.

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

- ✚ Dòng đầu tiên chứa số nguyên n ($1 \leq n \leq 10^3$),
- ✚ 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 HAPPY.OUT thông báo tương ứng xác định được.

Ví dụ:

HAPPY.INP
3
1 2 1

HAPPY.OUT
Sad



Giải thuật: Bài tập khởi động.

```
#include <fstream>
#define NAME "happy."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,a,r=0;

int main()
{fi>>n;
 for(int i=0;i<n;++i){fi>>a; r+=(a&1);}
 if(r>=n-r) fo<<"Sad"; else fo<<"Happy";
}
```

VQ33. KIỂM TRA

Tên chương trình: AUDIT.???

Hệ thống thiết bị điện tử tự động đã giải phóng cán bộ khí tượng thủy văn khỏi phải ngày ngày trèo lên các trạm đo đạc ở sườn núi chênh vênh hay chèo thuyền ra các đảo lấy số liệu quan trắc. Tuy vậy cũng phải định kỳ kiểm tra xem chúng có hoạt động tốt hay không để bảo dưỡng, thay thế kịp thời.

Steve có nhiệm vụ kiểm tra các máy đo lượng mưa lắp ở n trạm. Mỗi trạm có một bình thu nước. Hiện tại ở trạm thứ i máy báo về dung lượng nước trong bình là a_i , $i = 1 \dots n$, bình thứ i có van xả nước cho phép xả b_i đơn vị thể tích nước trong một giây. Khi Steve bấm nút bảo dưỡng, tất cả các van động thời điểm mở. Có nhiều tham số được ghi lại, nhưng hiện tại Steve chỉ quan tâm tới số liệu về tổng số nước ở các bình tại thời điểm 0 (lúc bắt đầu mở van và nước chưa kịp chảy ra), tổng số lượng nước trong các bình sau khi mở van 1 giây, 2 giây, ..., tổng số lượng nước trong các bình sau khi mở van t giây. Máy móc vẫn hoạt động bình thường.

Hãy xác định các số liệu mà Steve thu được.

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

- ✚ Dòng đầu tiên chứa 2 số nguyên n và t ($1 \leq n \leq 10^5$, $1 \leq t \leq 10^6$),
- ✚ Dòng thứ i trong n dòng sau chứa 2 số nguyên a_i và b_i ($0 \leq a_i \leq 10^9$, $0 \leq b_i \leq 10^9$).

Kết quả: Đưa ra file văn bản AUDIT.OUT $t+1$ số nguyên, mỗi số trên một dòng – các giá trị thiết bị báo về theo trình tự thời gian.

Ví dụ:

AUDIT.INP	AUDIT.OUT
6 6	46
12 2	33
10 3	23
3 1	14
5 4	9
7 2	6
9 1	3



Vq33_OI20150201_B

Giải thuật:

Nhận xét: Các kết quả theo thời gian được tính dựa vào mốc thời điểm 0.

Tổ chức dữ liệu:

- ⊕ **int $a[100000], b[100000]$** – lưu dữ liệu ban đầu,
- ⊕ **pair<int, int> $tm0[100001]$:**
 - ❖ $tm0[j].first$ – thời điểm bình k hết nước,
 - ❖ $tm0[j].second$ – chỉ số k : bình bị hết nước,
- ⊕ **suma** – tổng số nước trong các bình đang xét,
- ⊕ **perdu** – tổng số nước chảy ra trong một đơn vị thời gian ở các bình đang xét.

Xử lý:

- ⊕ Tính $tm0$ và chỉ lưu các giá trị thời điểm không vượt quá t ,
- ⊕ Sắp xếp $tm0$ theo thời điểm bình hết nước,
- ⊕ Đặt giá trị **max** $> t$ ở cuối để chặn (hàng rào).
- ⊕ Duyệt với mọi i từ 0 đến t : nếu thời điểm i có bình k hết nước thì loại a_k khỏi **suma** và loại b_k khỏi **perdu**.
- ⊕ Kết quả cần đưa ra là **suma** – $i * perdu$.

Độ phức tạp của giải thuật: O($n \log n$).

Chương trình:

```
#include <fstream>
#define NAME "audit."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
const int MXX=1000010;
int n,t,a[100001],b[100001],tmp,m=-1;
int64_t ans,suma=0,perdu=0;
pair<int,int>tm0[100001];

int main()
{ clock_t aa=clock();
fi>>n>>t;
int ta,tb;
for(int i=0;i<n;++i)
{fi>>ta>>tb; suma+=ta; perdu+=tb; a[i]=ta; b[i]=tb;
if(tb){tmp=(ta+tb-1)/tb;
if(tmp<=t) tm0[++m]=make_pair(tmp,i);}
}
sort(tm0,tm0+m+1);tm0[m+1]=make_pair(MXX,MXX);
int k=0,p;
for(int i=0;i<=t;++i)
{
    while(i==tm0[k].first)
    {
        p=tm0[k++].second;suma-=a[p];perdu-=b[p];
    }
    ans=suma-i*perdu;
    fo<<ans<<'\n';
}
clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ28. THU THUẾ

Tên chương trình: TAX.???

Sông Big Flat của xứ Flatland rất giàu tôm cá và toàn bộ con sông, từ đầu nguồn cho đến cửa biển đều nằm gọn trong Flatland. Dọc con sông có n xí nghiệp đánh bắt và chế biến thủy sản. Xí nghiệp thứ i (tính từ đầu nguồn xuống hạ lưu) sở hữu quyền đánh bắt trên đoạn sông dài a_i , $i = 1 \dots n$. Không có đoạn sông nào không có xí nghiệp sở hữu và mỗi đoạn sông chỉ thuộc quyền sở hữu của một xí nghiệp.

Cơ chế thị trường đã có tiếng nói mạnh mẽ trong việc điều tiết sản xuất. Đã có k lần xuất hiện những biến động trên bản đồ sản xuất. Các biến động đó có thể là một xí nghiệp phá sản và giải thể (sự kiện loại 1) hoặc một xí nghiệp phát triển và tách thành 2 xí nghiệp độc lập (sự kiện loại 2). Mỗi sự kiện được đặc trưng bằng cặp dữ liệu (e, c), trong đó e – loại sự kiện ($e = 1, 2$), c – số thứ tự tính từ thượng nguồn của xí nghiệp có sự kiện xảy ra.

Khi xuất hiện sự kiện loại 2 khúc sông mà xí nghiệp trước khi tách sở hữu sẽ được chia thành 2 phần bằng nhau (nếu độ dài chẵn), mỗi xí nghiệp mới sẽ sở hữu một phần, nếu độ dài khúc sông cần chia là lẻ thì 2 phần được tách ra chênh nhau 1 và xí nghiệp gần thượng lưu hơn sẽ sở hữu phần ngắn hơn.

Khi xí nghiệp bị phá sản khúc sông tương ứng sẽ được chia cho các xí nghiệp liền kề với xí nghiệp bị phá sản. Nếu chỉ có một xí nghiệp liền kề thì toàn bộ khúc sông tương ứng sẽ thuộc xí nghiệp liền kề. Nếu có 2 xí nghiệp liền kề thì khúc sông thuộc sở hữu của xí nghiệp bị phá sản sẽ bị chia thành 2 phần theo quy tắc tương tự như trường hợp tách xí nghiệp và xí nghiệp liền kề gần thượng lưu hơn sẽ sở hữu phần ngắn hơn.

Thuế tài nguyên mà xí nghiệp phải đóng là bình phương độ dài đoạn sông thuộc sở hữu của xí nghiệp.

Kiểm toán nhà nước muốn kiểm tra lại việc thu thuế trong thời gian qua có được tiến hành đúng luật hay không và vì vậy cần tính lại số thuế thu lúc ban đầu cũng như sau mỗi lần có biến động.

Hãy tính và đưa ra tổng số thuế cần thu theo luật mà kiểm toán cần.

Dữ liệu: Vào từ file văn bản TAX.INP:

- ✚ Dòng đầu tiên chứa 2 số nguyên n và p , trong đó p – loại nhóm tests để kiểm tra và cho điểm ($2 \leq n \leq 10^5$, $0 \leq p \leq 4$),
- ✚ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^4$, $i = 1 \dots n$),
- ✚ Dòng thứ 3 chứa số nguyên k ($1 \leq k \leq 10^5$),
- ✚ Mỗi dòng trong k dòng sau chứa 2 số nguyên e và c .

Dữ liệu đảm bảo xí nghiệp bị phân chia sở hữu khúc sông độ dài lớn hơn 1 và nếu chỉ còn một xí nghiệp thì xí nghiệp đó không phá sản, giá trị c luôn tồn tại với n của thời điểm xảy ra sự kiện.

Kết quả: Đưa ra file văn bản TAX.OUT $k+1$ giá trị tính được, mỗi giá trị trên một dòng.

Ví dụ:

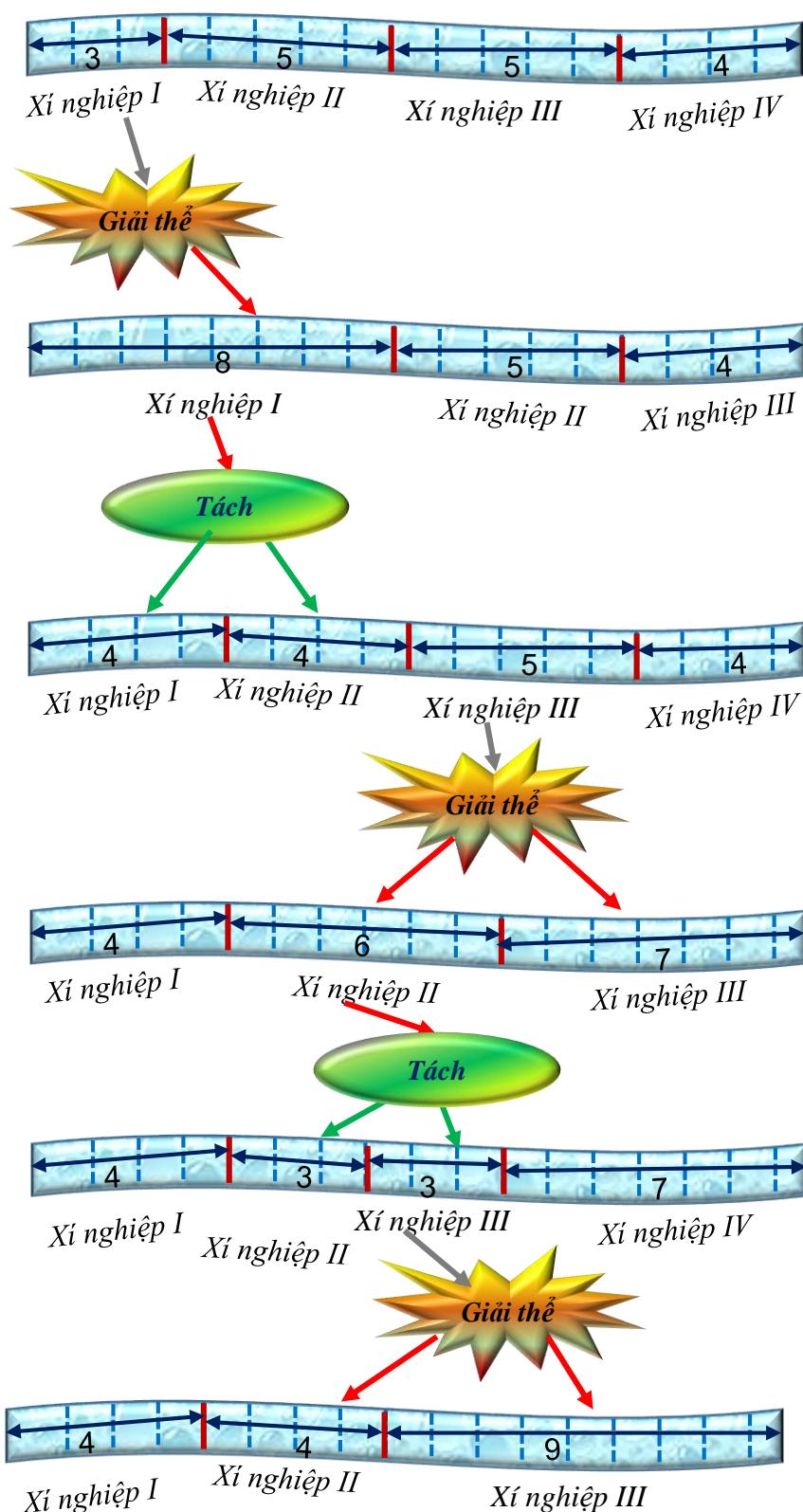
TAX.INP
4 0
3 5 5 4
5
1 1
2 1
1 3
2 2
1 3

TAX.OUT
75
105
73
101
83
113



Vq28 ROI_reg20150124 C

Giải thích ví dụ:



Tổng số thuế:

$$3^2 + 5^2 + 5^2 + 4^2 = 75$$

Tổng số thuế:

$$8^2 + 5^2 + 4^2 = 105$$

Tổng số thuế:

$$4^2 + 4^2 + 5^2 + 4^2 = 73$$

Tổng số thuế:

$$4^2 + 6^2 + 7^2 = 101$$

Tổng số thuế:

$$4^2 + 3^2 + 3^2 + 7^2 = 83$$

Tổng số thuế:

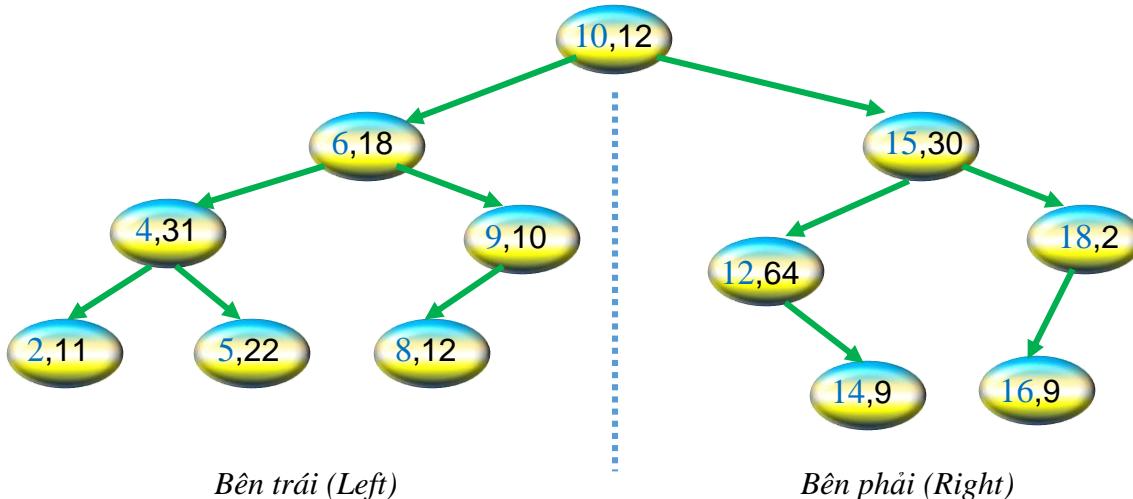
$$4^2 + 4^2 + 9^2 = 113$$

Như vậy khó khăn lớn nhất trong việc giải quyết bài toán này là các đối tượng (xí nghiệp) được đánh số tuyệt đối theo vị trí tính từ đầu dãy. Trong quá trình tồn tại, số thứ tự của một xí nghiệp có thể bị thay đổi theo thời gian, phụ thuộc vào dãy sự kiện đã xảy ra. Trên thực tế, đối tượng xử lý là các đoạn sông và số thứ tự của nó tính từ đầu nguồn, mỗi đơn vị dữ liệu xử lý là một cặp giá trị:

(Số thứ tự, Độ dài đoạn sông)

Cấu trúc dữ liệu cho phép đánh số động các đoạn sông là Cây Đè Các (Cartesian Tree) hay còn thường được gọi với tên khác là Cây vun đống Treap (Tree + Heap).

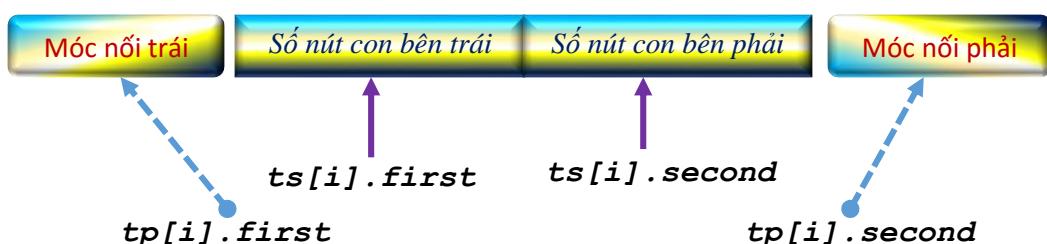
Trong cặp giá trị nêu trên *Số thứ tự* là khóa phục vụ tổ chức lưu trữ và tìm kiếm dữ liệu. Trong trường hợp tổng quát, mỗi nút của Treap cần lưu trữ bản ghi dạng (*Khóa, Giá trị*).



Khóa của nút con bên trái phải nhỏ hơn khóa của nút cha, khóa các nút con bên phải – lớn hơn khóa của nút cha.

Đối với các bài toán có khóa là số nguyên (hoặc tồn tại ánh xạ đơn giản từ khóa sang số nguyên) thì có thể không cần thiết lưu trữ tường minh khóa ở mỗi nút.

Bài toán đang xét có khóa tạo thành tập các số nguyên liên tục trong đoạn $[1, n]$, vì vậy có thể tổ chức một Treap, trong đó mỗi nút chứa 4 đơn vị thông tin:

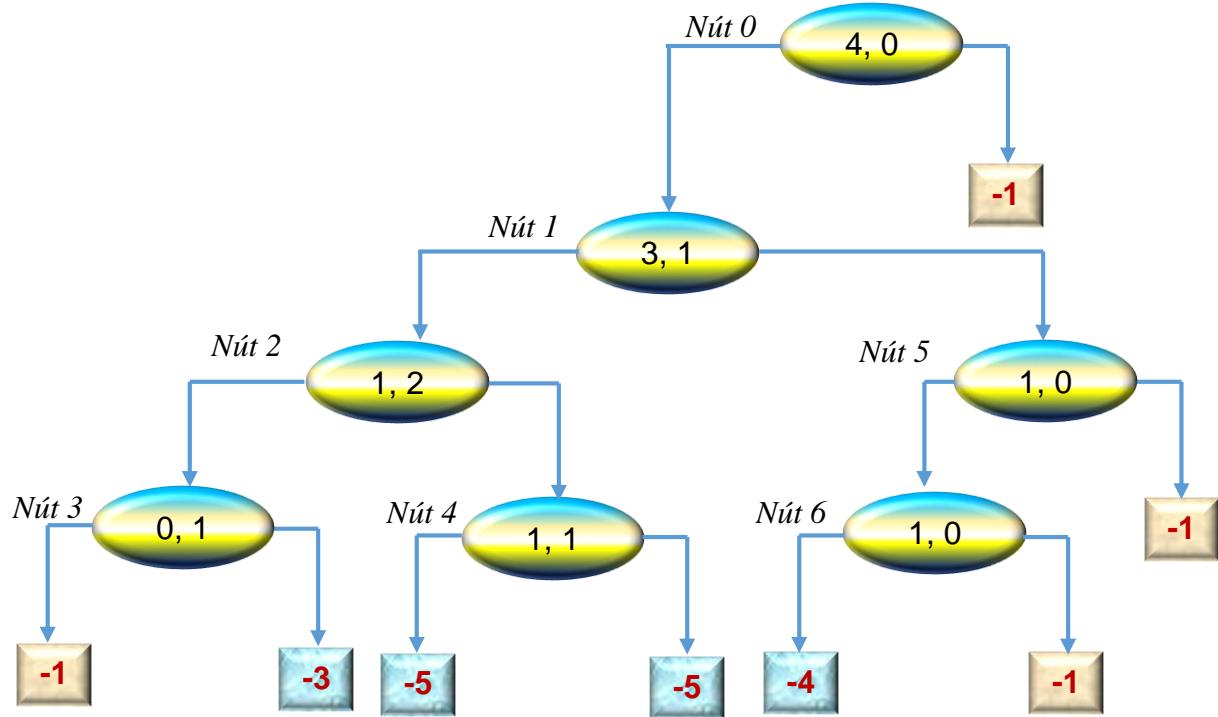


Thời gian thực hiện chương trình sẽ *giảm một cách đáng kể* nếu các mảng *ts* và *tp* được khai báo tĩnh.

Giá trị của móc nối sẽ là âm khi không có nhánh đi tiếp. Độ dài đoạn sông được lưu trữ với dấu âm ở phần móc nối của nút tương ứng.

Cây ban đầu được khởi tạo tương tự cây tiền tố quản lý n giá trị từ 1 tới n.

Với ví dụ ở đầu bài, cây ban đầu có dạng:



Các phép xử lý:

split_c(int x) – tách xí nghiệp x thành 2 xí nghiệp,
bankrupt_c(int x) – giải thể xí nghiệp **x**.

a) Tách xí nghiệp:

- + Duyệt theo danh sách mốc nối, tìm tới nút tương ứng với xí nghiệp **x**,
- + Trong quá trình duyệt: tăng số lượng nút gấp trên đường đi lên 1
- + Lưu độ dài **tg** của đoạn sòng tìm được,
- + Tạo nút mới, gắn vào bên trái (hoặc bên phải) nút tìm được,
- + Gán giá trị các mốc nối của nút mới là $-tg/2$ và $-(tg+1)/2$,
- + Cập nhật lại tổng số thuế và **n**.

b) Xử lý giải thể:

- + Duyệt theo danh sách mốc nối, tìm tới nút tương ứng với xí nghiệp **x**,
- + Trong quá trình duyệt: giảm số lượng nút gấp trên đường đi 1,
- + Lưu độ dài **tg** của đoạn sòng tìm được,
- + Phân biệt 3 trường hợp:
 - ❖ Xí nghiệp giải thể ở đầu:cập nhật lại dữ liệu của xí nghiệp 1,
 - ❖ Xí nghiệp giải thể ở cuối:cập nhật lại dữ liệu của xí nghiệp **x-1**,
 - ❖ Xí nghiệp giải thể ở giữa:cập nhật lại dữ liệu của các xí nghiệp **x-1** và **x**,

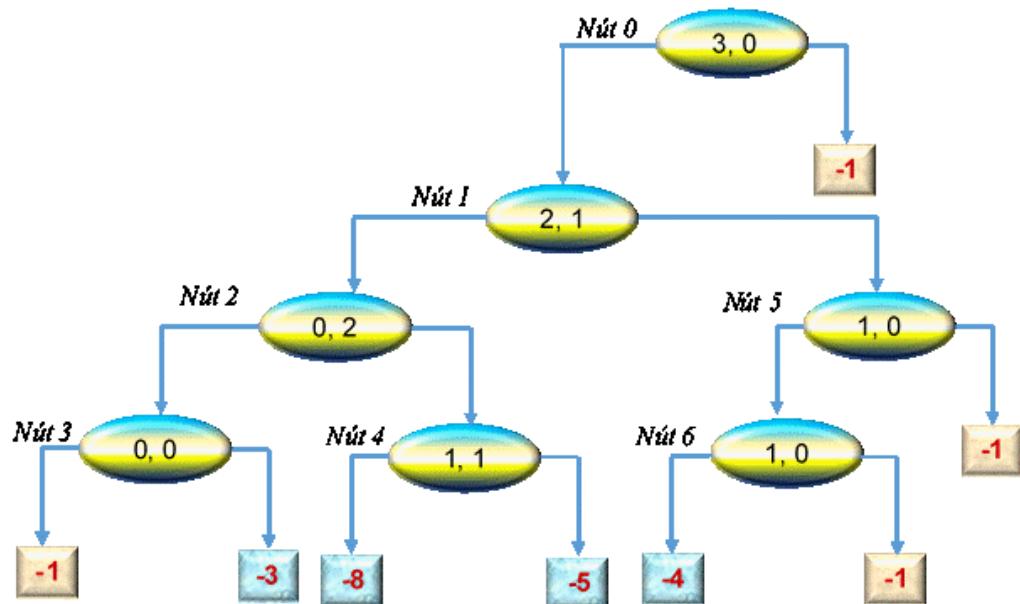
- Trong mỗi trường hợp: Cập nhật lại tổng số thuế và n .

Nhận xét:

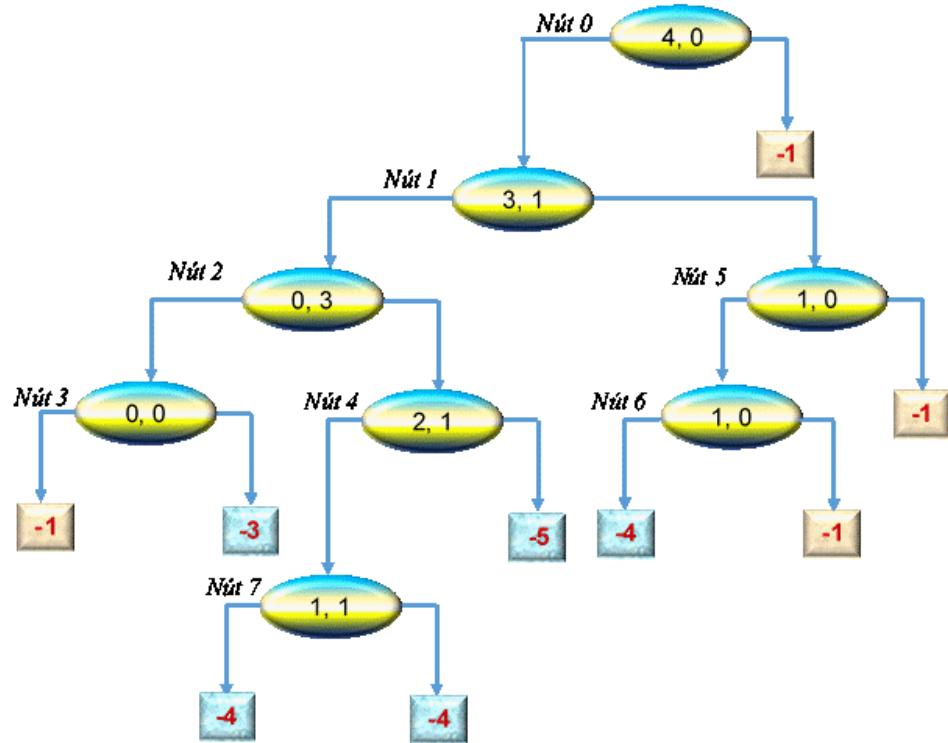
- Mỗi xí nghiệp không thể bị tách quá 17 lần, vì vậy không xảy ra hiện tượng cây bị mất cân đối và làm tăng thời gian tìm kiếm,
- Không cần thiết phải xóa khỏi danh sách các nút tương ứng với xí nghiệp bị giải thể.

Độ phức tạp của giải thuật: $O(k \log n)$.

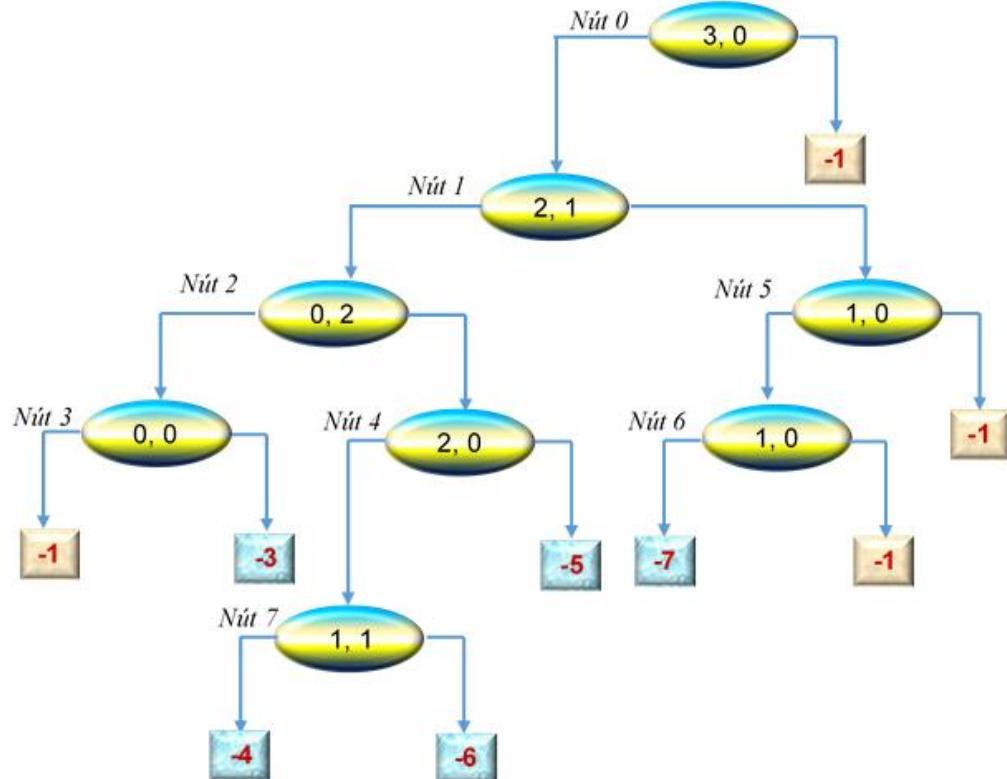
Treap sau phép xử lý giải thể xí nghiệp 1:



Treap sau bước xử lý tiếp theo: tách xí nghiệp 1:



Treap sau bước xử lý tiếp theo: giải thê xí nghiệp 3:



Chương trình:

```
#include <fstream>
#include <ctime>
#define NAME "tax."
using namespace std;

ifstream fi(NAME"inp");
ofstream fo(NAME"out");
int n,m=0,k,e,c,t,tq;
int64_t ans=0;
pair<int,int>tp[200000],ts[200000];
pair<int,int> ep,es;

void new_el()
{tp[++m]=ep; ts[m]=es; }

void push_qb(int x,int y)
{int64_t tg,p=0,q;
 for(int j=k;j>=0;--j)
 {tg=x&(1<<j);
 if(tg){q=tp[p].second;++ts[p].second;
 if(q<0 && (j!=0)){new_el();tp[p].second=m;q=m;} p=q;}
 else{q=tp[p].first;++ts[p].first;
 if(q<0 && (j!=0)){new_el();tp[p].first=m; q=m;}p=q;}
 if(j==0) {if(tg)tp[m].second=-y;else tp[m].first=-y;}
 }
}

void chbi()
{ m=0;ep=make_pair(-1,-1);es=make_pair(0,0);
 tp[0]=ep;ts[0]=es;
}

void split_c(int x)
{int p=0,r,q,tg;
 tg=x;
 while(p>=0)
 {if(tg>ts[p].first)
 {tg-=ts[p].first;++ts[p].second;r=1;q=p;p=tp[p].second;}
 else {r=0;q=p;ts[p].first++;p=tp[p].first;}
 tg=-p;
 new_el(); if(r) tp[q].second=m; else tp[q].first=m;
 tp[m].first=-tg/2;ts[m].first=1;
 tp[m].second=- (tg+1)/2;ts[m].second=1;
 ans-=(int64_t)((tg/2)*((tg+1)/2))*2; ++n;
}

void bankrupt_c(int x)
{int tx,tg,t2,q,r,p=0;
 tg=x;
 while(p>=0)
 if(tg>ts[p].first)
 {tg-=ts[p].first;--ts[p].second;q=p;r=1;p=tp[p].second;}
 else {q=p;r=0;--ts[p].first;p=tp[p].first;}
 tx=-p; ans-=(int64_t)(tx)*tx;
 if(x==1)
 {
```

```

        tg=x;p=0;
        while(p>=0)
            if(tg>ts[p].first){tg-
=ts[p].first;q=p;r=1;p=tp[p].second;}
            else {q=p;r=0;p=tp[p].first;}
            tg=-p; ans+=(int64_t)(tg)*tg;
            tg+=tx; if(r)tp[q].second=-tg;
            else tp[q].first=-tg; ans+=(int64_t)(tg)*tg;
--n;return;
    }
    if(x==n)
    {
        tg=x-1;p=0;
        while(p>=0)
            if(tg>ts[p].first){tg-
=ts[p].first;q=p;r=1;p=tp[p].second;}
            else {q=p;r=0;p=tp[p].first;}
            tg=-p; ans+=(int64_t)(tg)*tg;
            tg+=tx; if(r)tp[q].second=-tg;
            else tp[q].first=-tg; ans+=(int64_t)(tg)*tg;
--n;return;
    }
    t2=(tx+1)/2; tx/=2;
    tg=x-1;p=0;
    while(p>=0)
        if(tg>ts[p].first){tg-
=ts[p].first;r=1;q=p;p=tp[p].second;}
        else {r=0;q=p;p=tp[p].first;}
        tg=-p; ans+=(int64_t)(tg)*tg;
        tg+=tx; if(r)tp[q].second=-tg;
        else tp[q].first=-tg; ans+=(int64_t)(tg)*tg;
    tg=x;p=0;
    while(p>=0)
        if(tg>ts[p].first){tg-
=ts[p].first;r=1;q=p;p=tp[p].second;}
        else {r=0;q=p;p=tp[p].first;}
        tg=-p; ans+=(int64_t)(tg)*tg;
        tg+=t2; if(r)tp[q].second=-tg;
        else tp[q].first=-tg; ans+=(int64_t)(tg)*tg;
--n;return;
}

int main()
{clock_t aa=clock();
 fi>>n>>t;
 for(int i=17;i>=0;--i)if(n&(1<<i)){k=i+1;break;}
 chbi(); tq=0;
 for(int i=1;i<=n;++i)
 {
     fi>>t;ans+=t*t; push_qb(i,t);
 }

 fo<<ans<<'\n';
 fi>>k;
 for(int i=1; i<=k;++i)
 {
     fi>>e>>c;
}

```

```
    if(e==1)bankrupt_c(c); else split_c(c);
fo<<ans<<'\n';
}

clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ35. QUÁ TẢI

Tên chương trình: OVERLOAD.???

Trên đường cao tốc dẫn đến bến cảng có rất nhiều xe lưu thông. Các xe khi rời cảng được kiểm soát chặt không để xảy ra hiện tượng chờ quá tải, nhưng các xe chờ hàng tới có thể vi phạm các quy định về tải trọng. Để phát hiện và bắt giữ các xe vi phạm người ta bố trí trên đường m thiết bị cân tự động, thiết bị thứ i đặt ở kí lô mét b_i ($0 \leq b_i \leq b_{i+1}$, $i = 1, 2, \dots, m-1$). Nếu xe có tải trọng lớn hơn mức được phép đi qua cân, các cảm biến sẽ được kích hoạt, các má phanh bật lên ôm sát bánh buộc xe phải dừng lại. Mỗi cân chỉ được kích hoạt tự động một lần, muốn tháo cân trả về trạng thái ban đầu phải có sự can thiệp trực tiếp của cán bộ vận hành.

Có n xe quá tải lưu thông trên đường, xe thứ j vào đường cao tốc ở km a_j ($0 \leq a_j \leq a_{j+1}$, $j = 1, 2, \dots, n-1$). Nếu một xe đang ở vị trí km x và vẫn đi được thì sau một đơn vị thời gian xe sẽ ở vị trí $x+1$.

Với mỗi xe hãy xác định nó sẽ bị cân tự động nào phát hiện. Nếu xe không bị phát hiện thì đưa ra số -1.

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

- ✚ Dòng đầu tiên chứa 2 số nguyên n và m ($1 \leq n, m \leq 10^5$),
- ✚ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($0 \leq a_j \leq a_{j+1} \leq 10^9$, $j = 1, 2, \dots, n-1$),
- ✚ Dòng thứ 3 chứa m số nguyên b_1, b_2, \dots, b_m ($0 \leq b_i \leq b_{i+1} \leq 10^9$, $i = 1, 2, \dots, m-1$).

Kết quả: Đưa ra file văn bản OVERLOAD.OUT n số nguyên, mỗi số trên một dòng, số thứ j xác định số thứ tự của cân phát hiện xe j , $j=1, 2, \dots, n$.

Ví dụ:

OVERLOAD.INP
8 6
0 2 3 4 5 6 8 13
1 3 5 6 9 12

OVERLOAD.OUT
1
-1
2
6
3
4
5
-1



Giải thuật: Tổ chức và xử lý stacks.

- ⊕ Tổ chức stack c lưu các cân chưa bị kích hoạt,
- ⊕ Duyệt các xe và cân từ cuối về đầu,
- ⊕ Với mỗi xe j ($j = n \div 1$):
 - ❖ Nạp vào hàng đợi c các số thứ tự i thỏa mãn $a_j \geq b_i$,
 - ❖ Nếu hàng đợi c rỗng – gán $a_j = -1$, trong trường hợp ngược lại – gán cho a_j phần tử đầu hàng đợi và xóa phần tử này khỏi hàng đợi,
- ⊕ Đưa ra các giá trị a_j tìm được, $j = 1 \div n$.

Lưu ý:

- Chương trình sẽ hoạt động nhanh hơn nếu hàng đợi c được tổ chức dưới dạng mảng,
- Có thể xin cấp phát động để tiết kiệm bộ nhớ khi lưu trữ các mảng a , b và c , nhưng trong bài toán giá trị kích thước n, m không lớn (không vượt quá 10^5) và cố định, biết trước, vì vậy việc tổ chức cấp phát tĩnh bộ nhớ sẽ giảm thời gian thực hiện chương trình.

Độ phức tạp của giải thuật: $O(n)$.

Chương trình:

```
#include <fstream>
#include <iostream>
#include <cmath>
#include <ctime>
#include <iomanip>
#define NAME "overload."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int a[100001],b[100001],c[100001],m,n,p,q;

int main()
{clock_t aa=clock();
 fi>>n>>m;
 for(int i=1;i<=n;++i)fi>>a[i];
 for(int j=1;j<=m;++j)fi>>b[j]; b[0]=-1;
 q=m;p=0;
 for(int i=n;i>0;--i)
 {
     while(b[q]>=a[i])c[++p]=q--;
     if(p>0)a[i]=c[p--];else a[i]=-1;
 }
 for(int i=1;i<=n;++i)fo<<a[i]<<'\'n';
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ36. ĐƯỜNG ĐUA

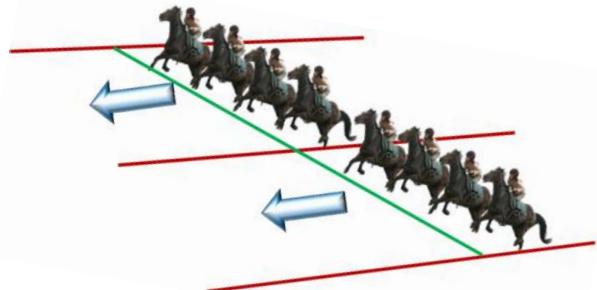
Tên chương trình: RACETRACK.???

Trường đua ngựa có k đường đua. Lần đua này có n chú ngựa tham gia. Ban tổ chức tiến hành phân nhóm ngựa vào các đường đua. Đường đua thứ i sẽ có a_i ngựa. Như vậy $a_1 + a_2 + \dots + a_k = n$. Đường đua nào cũng phải có ngựa, tức là $a_i > 0 \ \forall i$.

Với tinh thần chỉ đạo các lễ hội phải đậm đà bản sắc dân tộc nhưng cũng phải hiện đại, thể hiện được các yếu tố của thời đại @, Trưởng ban tổ chức yêu cầu phải cải tiến cách phân nhóm, cụ thể là phải thỏa mãn yêu cầu:

$$a_1 \oplus a_2 \oplus \dots \oplus a_k = 0,$$

trong đó \oplus là phép tính **xor** (phép \wedge trong C/C++).



Hãy đưa ra một cách phân nhóm thỏa mãn các yêu cầu đã nêu hoặc đưa ra số -1 nếu không tồn tại cách phân nhóm phù hợp.

Dữ liệu: Vào từ file văn bản RACETRACK.INP gồm một dòng chứa 2 số nguyên n và k ($1 \leq n, k \leq 10^5$).

Kết quả: Đưa ra file văn bản RACETRACK.OUT trên một dòng k số nguyên a_1, a_2, \dots, a_k xác định một cách phân nhóm thỏa mãn mọi yêu cầu đề ra.

Ví dụ:

RACETRACK.INP
8 2

RACETRACK.OUT
4 4



Giải thuật: Liệt kê tình huống, xây dựng bảng phương án.

Các trường hợp vô nghiệm:

- ⊕ $k = 1$ – chỉ có a_1 , vì vậy không thể có tổng $xor = 0$,
- ⊕ $n < k$ – sẽ có một số đường đua không có ngựa ($a_i = 0$),
- ⊕ $n = k+1$ – để mọi $a_i > 0$ thì chỉ tồn tại cách phân chia duy nhất $a_i=1$, $i = 1 \div k-1$ và $a_k = 2$ và xor khác 0,
- ⊕ n lẻ – không tồn tại cách phân chia để xóa bít 1 hàng đơn vị trong tổng xor ,
- ⊕ $k = 3$ và $n = 2^m$ – không tồn tại cách phân chia để xóa bít 1 trái nhất của a_i trong tổng xor .

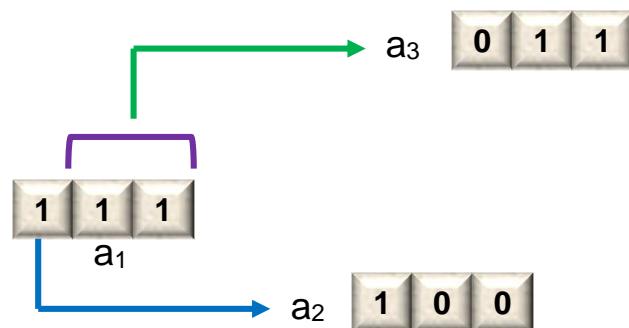
Xử lý các trường hợp có nghiệm:

- ⊕ Trường hợp k chẵn: cho $a_i = 1$, $i = 1 \div k-2$, $a_{k-1} = a_k = (n-k+2)/2$,
- ⊕ Trường hợp $k = 3$:
 - ❖ $a_1 = n/2$,
 - ❖ a_2 = giá trị bít cao nhất của a_1 ,
 - ❖ a_3 = giá trị các bít còn lại của a_1 .

Ví dụ $n=14$, $k=3$.

$$a_1 = n/2 = 7.$$

Việc tách này bao giờ cũng thực hiện được vì trường hợp $n = 2^m$ đã bị loại trừ!



- ⊕ Trường hợp còn lại: $k > 3$ và lẻ:

- Xác định 3 giá trị đầu tiên: $a_1 = 1$, $a_2 = 2$, $a_3 = 3$.
- Dễ dàng nhận thấy rằng $a_1 \oplus a_2 \oplus a_3 = 0$.
- Phần còn lại có $n = n - 6$ và $k = k - 3$. k mới sẽ là số chẵn và việc phân chia phần còn lại thực hiện theo trường hợp k chẵn đã xét.
- Tổng $a_4 \oplus a_5 \oplus \dots \oplus a_k = 0$, vì vậy $(a_1 \oplus a_2 \oplus a_3) \oplus (a_4 \oplus a_5 \oplus \dots \oplus a_k) = 0$.

Độ phức tạp của giải thuật: O(1).

Chương trình:

```
#include <fstream>
#include <iostream>
#include <cmath>
#include <ctime>
#include <iomanip>
#define NAME "racetrack."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int k,n,t;
void xly_3()
{
    if(n&(-n)==n){fo<<-1;return;}
    for(int i=18;i>=0;--i) if(n&(1<<i)){t=i;break;}
    t=1<<(t-1);
    fo<<n/2<<' '<<n/2-t<<' '<<t;
    return;
}

int main()
{//clock_t aa=clock();
fi>>n>>k;
if(k==1 || n<k || n==k+1 || n&1) {fo<<-1; return 0;}
if((k&1)==0)
    {for(int i = 1;i<=k-2;++i)
     fo<<1<<' ';t=(n-k+2)/2;fo<<t<<' '<<t; return 0;
    }
if(k==3){xly_3(); return 0;}
fo<<"1 2 3 "; n-=6; k-=3; t=(n-k+2)/2;
for(int i=1;i<=k-2;++i) fo<<"1 ";
fo<<t<<' '<<t;
//clock_t bb=clock();
//fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ39. TRÌNH TỰ

Tên chương trình: ORDER.???

Viện bảo tàng Thời trang và May mặc có một thư viện lớn gồm $2n$ cuốn sách về thời trang qua các thời đại, trong đó n cuốn giới thiệu về lịch sử thời trang, n cuốn khác – về thời trang đương đại. Các nhà tạo mốt thường đến tham khảo để tìm những ý tưởng sáng tạo mới. Các cuốn sách được đánh mã số từ 1 đến $2n$ và sắp xếp trên giá theo thứ tự tăng dần của mã số từ trái sang phải. Mỗi người, khi vào đọc được lấy 2 cuốn sách tùy chọn, mang về bàn đọc, sau đó trả lại sách về chỗ cũ trên giá. Các sách đều được gắn chíp điện tử và khi đặt lại không đúng chỗ hệ thống quản lý sẽ phát tín hiệu nhắc nhở. Trong phòng chỉ có một bàn vì vậy mỗi lần chỉ có không quá một người vào làm việc và phải trả lại hai sách cũ mới được lấy 2 quyển khác.

Hệ thống điện tử quản lý sách đang được sử dụng, nâng cấp vì vậy tín hiệu cảnh báo đặt sách sai chỗ tạm thời không hoạt động. Người đọc có thói quen đọc quyển có mã số thấp rồi đọc quyển thứ 2 đã lấy và sau khi đọc trình tự 2 cuốn sách trên bàn bị đảo ngược và cứ theo trình tự đó người ta nhét trở lại vào 2 chỗ trống trên giá sách. Nói một cách khác, nếu 2 cuốn sách được lấy ra từ các vị trí i và j thì sau đó cuốn ở vị trí i được đặt vào vị trí j , còn cuốn lấy ở vị trí j – đặt vào vị trí i .

Có m lần đọc giả đã đến lấy sách đọc, lần thứ k 2 cuốn được lấy ở các vị trí i_k, j_k và bị đặt ngược vị trí khi trả về ($k = 1 \div m$).

Hãy xác định sau mỗi lần đọc giả trả lại sách có bao nhiêu cuốn về lịch sử thời trang vẫn nằm trong các vị trí từ 1 đến n .

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

- ✚ Dòng đầu tiên chứa số nguyên n ($1 \leq n \leq 10^5$),
- ✚ Dòng thứ 2 chứa số nguyên m ($1 \leq m \leq 10^5$),
- ✚ Dòng thứ k trong m dòng sau chứa 2 số nguyên i_k và j_k ($1 \leq i_k, j_k \leq 2n$).

Kết quả: Đưa ra file văn bản ORDER.OUT m số nguyên, mỗi số trên một dòng – số lượng cuốn về lịch sử thời trang vẫn nằm trong các vị trí từ 1 đến n sau mỗi lần đọc giả trả sách.

Ví dụ:

ORDER.INP
3
3
1 4
2 5
3 6

ORDER.OUT
2
1
0



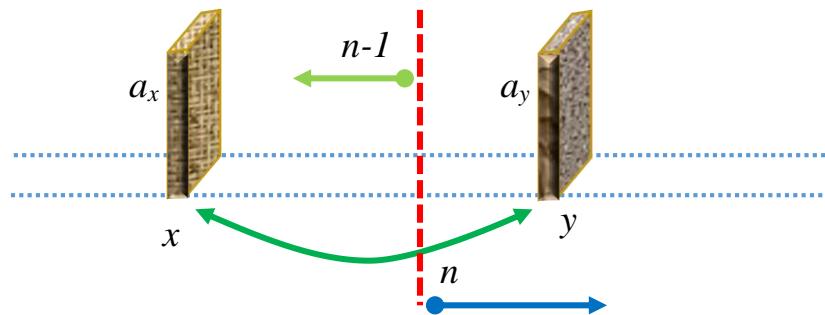
Giải thuật: Phân tích các khả năng lô gic ở mỗi bước.

Tổ chức dữ liệu:

Mảng \mathbf{a} kích thước $2 \times n$ chứa mã các cuốn sách, ban đầu: $\mathbf{a}[i] = i$, $i = 0 \div 2 \times n - 1$,

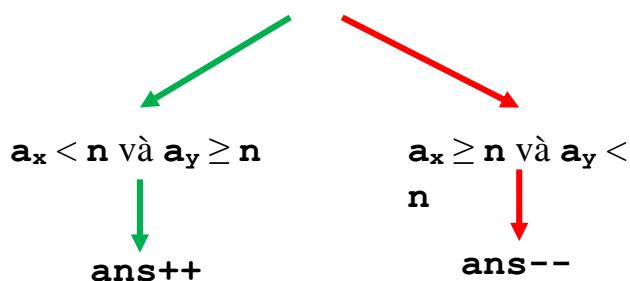
Biến \mathbf{ans} – chứa kết quả, ban đầu $\mathbf{ans} = n$.

Xử lý các phép đổi chỗ \mathbf{x} và \mathbf{y} .



Chuẩn hóa để $\mathbf{x} < \mathbf{y}$,

Tính huống làm thay đổi kết quả: $\mathbf{x} < \mathbf{n}$ và $\mathbf{y} > \mathbf{n}$



Chương trình:

```
#include <fstream>
#include <vector>
#include <algorithm>
#include <ctime>
#define NAME "order."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int main()
{clock_t aa=clock();
 int n, m;
 fi>>n>>m;
 vector<int> a(2 * n);
 for (int i = 0; i < 2 * n; i++) a[i] = i;
 int ans = n;
 for (int i = 0; i < m; i++)
 {
     int x, y;
     fi>>x>>y;
     x--;
     y--;
     swap(a[x], a[y]);
     if (x > y)
         swap(x, y);
     if (x < n && y >= n)
     {
         if (a[x] < n && a[y] >= n)
             ans++;
         if (a[x] >= n && a[y] < n)
             ans--;
     }
     fo<<ans<<' \n';
 }
 clock_t bb=clock();
 fo<<"\nTime: "<<(double) (bb-aa)/1000<<" sec";
}
```

VQ40. TRUY ĐUỒI

Tên chương trình: CHASE.???

Nhận được thông báo có xe chở hàng buôn lậu quốc cấm hiện đang km s của quốc lộ đồn biên phòng đóng ở km 0 lập tức cho xe truy đuổi. Bọn buôn lậu cũng đã phát hiện ra là bị truy đuổi và không từ một thủ đoạn nào để tìm cách trốn thoát. Trên xe của bọn buôn lậu có k thùng phuy dầu máy. Chúng quyết định khi cần thiết, tại các đoạn đường dốc hiểm trở sẽ đổ dầu ra đường làm xe truy đuổi buộc phải giảm tốc độ, mỗi lần sẽ phải đổ hết cả một thùng phuy. Có n điểm có thể đổ dầu cản trở xe của lực lượng truy đuổi, điểm thứ i ở km x_i và sẽ làm cho xe truy đuổi phải mất thêm a_i thời gian để vượt qua đoạn đường bị đổ dầu ($i = 1 \dots n$).

Tốc độ xe của bọn buôn lậu là $v1$, tốc độ xe của đồn biên phòng là $v2$.

Hãy xác định thời gian tối đa bọn buôn lậu có thể trì hoãn trước khi bị bắt. Thời điểm bọn buôn lậu bị bắt là khi 2 xe ở cùng một địa điểm, thậm chí nếu đó là thời điểm xe bỏ chạy đang đổ dầu ra đường! Nếu không thể đuổi kịp bọn buôn lậu thì đưa ra thông báo “**inf**”.

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

- + Dòng đầu tiên chứa 2 số nguyên n và k ($1 \leq n, k \leq 10^5$),
- + Dòng thứ 2 chứa 2 số nguyên $v1$ và $v2$ ($1 \leq v1, v2 \leq 1000$),
- + Dòng thứ 3 chứa số nguyên s ($0 \leq s \leq 10^8$),
- + Dòng thứ i trong n dòng sau chứa 2 số nguyên x_i và a_i ($0 \leq x_i \leq 10^8$, $0 \leq a_i \leq 1000$, $x_i < x_{i+1}$, $i = 1 \dots n-1$).

Kết quả: Đưa ra file văn bản CHASE.OUT một số thực với độ chính xác 10^{-6} – thời gian tính được hoặc thông báo “**inf**” nếu không thể đuổi kịp xe buôn lậu.

Ví dụ:

CHASE.INP
6 2
1 2
3
0 1
5 2
7 3
10 4
11 5
12 6

CHASE.OUT
13.000000



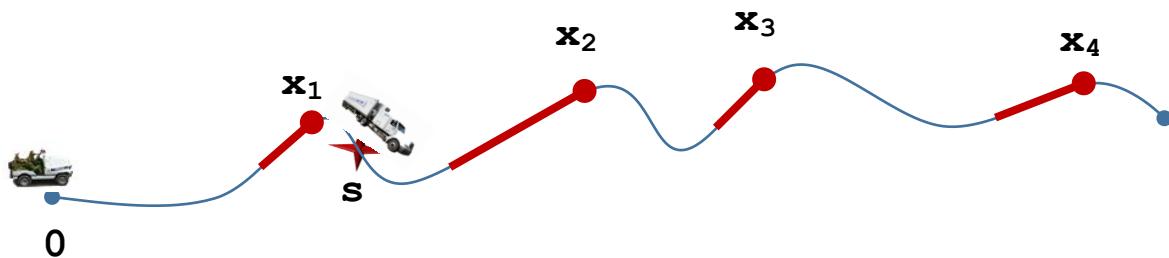
Giải thuật: Rèn luyện kỹ năng sử dụng hàng đợi ưu tiên.

Nhận xét:

Cần phân lập, xử lý các trường hợp riêng:

- ⊕ Trường hợp $s = 0$, thời gian truy đuổi là 0,
- ⊕ Trường hợp $s > 0$ và $v_1 \geq v_2$ - không thể đuổi kịp,
- ⊕ Trường hợp còn lại: tính thời gian truy đuổi.

Xử lý trường hợp 3 (đuối kịp): Nếu khoảng cách giữa 2 xe là $dist$ thì thời gian 2 xe gặp nhau sẽ là $\frac{dist}{v_2 - v_1}$.



Lọc và loại bỏ các $x_i < s$: $m = \min\{i \mid x_i \leq s\}$,

Ban đầu, khoảng cách 2 xe $dist = s$,

Nếu xe bỏ chạy đỗ đầu ở điểm x_i thì thời gian truy đuổi sẽ tăng thêm một lượng là a_i , điều này tương đương như không có hiện tượng đỗ đầu, nhưng khoảng cách ban đầu của 2 xe tăng thêm một khoảng là $a_i \times v_2$,

Vẫn đề phải giải quyết là khi nào xe trốn chạy đỗ đầu và đỗ ở đâu để kéo dài nhất thời gian trốn chạy.

Gọi d là khoảng cách gia tăng do hiệu ứng đỗ đầu, d ban đầu bằng 0.

Lần lượt xét các điểm x_i , $i = m \div k$, nếu xe trốn chạy tới điểm này trước xe truy đuổi thì nạp a_i vào hàng đợi q , trong trường hợp ngược lại – lấy a_i từ đầu hàng đợi ra khỏi hàng đợi và chỉnh lý lại d .

Hàng đợi q được tổ chức theo kiểu **priority_queue<int>**, vì vậy giá trị lấy ra là lớn nhất trong các khả năng lựa chọn, thời gian truy đuổi sẽ kéo dài nhiều nhất.

Độ phức tạp của giải thuật: $O(n \log(n))$ (sử dụng hàng đợi ưu tiên).

Chương trình:

```
#include <fstream>
#include <iomanip>
#include <queue>
#include <ctime>
#define NAME "chase."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,k,s,v1,v2,x[100001],a[100001],m;
priority_queue<int> q;
int64_t t,d=0;
double ans;

int main()
{clock_t aa=clock();
 fi>>n>>k>>v1>>v2>>s;
 for(int i=0;i<n;++i)fi>>x[i]>>a[i];
 if(s==0){fo<<0; return 0;}
 if(v1>=v2){fo<<"inf"; return 0;}
 m=n; for(int i=0;i<n;++i)if(x[i]>=s){m=i; break;}
 while(k>0)
 {
     for(int &i=m;i<n;++i)
         {if((double)(x[i]-s)/v1>=(double)(x[i]+d)/v2)break;
          q.push(a[i]);
         }
     --k;
     if(!q.empty())t=q.top(); else break;
     d+=t*v2; q.pop();
 }
 ans=(double)(s+d)/(v2-v1);
 fo<<fixed<<setprecision(6)<<ans;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ37. KHỞI TẠO KHÓA

Tên chương trình: GENKEY.???

Trong một trò chơi online, muốn qua được mức mới bạn phải tạo khóa dựa trên các công cụ được cung cấp trong trò chơi. Khóa là một xâu **s** độ dài **n** chỉ chứa các ký tự la tinh thường. Người chơi phải tạo lại một khóa giống hệt như vậy dựa trên 3 công cụ được cung cấp:

- Gắn thêm một ký tự vào cuối xâu đang xây dựng với chi phí thời gian thực hiện là **a**,
- Gắn vào cuối xâu đang xây dựng xâu đã có, tức là “gấp đôi” xâu với chi phí thời gian thực hiện là **b**,
- Xóa ký tự cuối xâu đang xây dựng với chi phí thời gian thực hiện là **c**.

Bạn chỉ có thể qua được mức mới khi thời gian tạo khóa là nhỏ nhất.

Hãy xác định thời gian tạo khóa cần thiết để có thể sang mức mới.

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

- ✚ Dòng đầu tiên chứa số nguyên **n** ($1 \leq n \leq 10^5$),
- ✚ Dòng thứ 2 chứa xâu **s**,
- ✚ Dòng thứ 3 chứa 3 số nguyên **a**, **b** và **c** ($0 \leq a, b, c \leq 10^9$).

Kết quả: Đưa ra file văn bản GENKEY.OUT một số nguyên – thời gian tạo khóa cần thiết để có thể sang mức mới.

Ví dụ:

GENKEY.INP
7
abcdabc
1 2 0

GENKEY.OUT
6



Chương trình:

```
#include <fstream>
#include <set>
#include <vector>
#include <ctime>
#include <string>
#include <algorithm>
#define NAME "genkey."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
string s;
int64_t a, b, c;
int n;
const int64_t INF = (int64_t)1e18;

vector<int> z_function(string s)
{
    vector<int> z(n, 0);
    int l = 0, r = 0;
    for (int i = 1; i < n; i++)
    {
        if (i <= r)
            z[i] = min(r - i + 1, z[i - 1]);
        while (z[i] + i < n && s[z[i]] == s[z[i] + i])
            z[i]++;
        if (z[i] + i - 1 > r)
            l = i, r = z[i] + i - 1;
    }
    return z;
}

int main()
{clock_t aa=clock();
    fi >> n >> s >> a >> b >> c;
    vector<int> z = z_function(s);
    vector<int64_t> d(n, INF);
    vector<pair<int, int> > p(n);
    d[0] = a;
    p[0] = make_pair(-1, 0);
    set<pair<int64_t, int> > q;
    q.insert(make_pair(d[0], 0));
    while (!q.empty())
    {
        int v = q.begin()>>second;
        q.erase(q.begin());
        int to;
        int64_t cost;
        //1)double string
        if (v + 1 < n)
        {
            int per = min(v + 1, z[v + 1]);
            to = v + per;
            cost = b + (v + 1 - per) * c;
            if (d[v] + cost < d[to])
            {
```

```

        q.erase(make_pair(d[to], to));
        d[to] = d[v] + cost;
        p[to] = make_pair(v, 2);
        q.insert(make_pair(d[to], to));
    }
}

//2)insert letter
if (v + 1 < n)
{
    to = v + 1;
    cost = a;
    if (d[v] + cost < d[to])
    {
        q.erase(make_pair(d[to], to));
        d[to] = d[v] + cost;
        p[to] = make_pair(v, 1);
        q.insert(make_pair(d[to], to));
    }
}
//3)delete letter
if (v - 1 >= 0)
{
    to = v - 1;
    cost = c;
    if (d[v] + cost < d[to])
    {
        q.erase(make_pair(d[to], to));
        d[to] = d[v] + cost;
        p[to] = make_pair(v, 3);
        q.insert(make_pair(d[to], to));
    }
}
fo << d[n - 1];
clock_t bb=clock();
fo<<"\nTime: "<<(double) (bb-aa)/1000<<" sec";
}

```

VQ41. CƠ HỘI THẮNG

Tên chương trình: WINCHANCE.???

Alice và Bob có nhiệm vụ trực đường dây nóng của Công ty. Không có gì buồn chán và căng thẳng hơn chờ đợi. Để thư giãn Bob đề xuất tiêu khiển bằng một trò chơi đơn giản. Với một số nguyên dương n chọn trước, xuất phát từ 1 hai người lần lượt xóa số cũ, viết một số mới lớn hơn số cũ 1 hoặc lớn gấp đôi số cũ. Các số viết ra không được lớn hơn n . Ai đến lượt mình không thể viết được số mới là thua. Alice đi trước. Ví dụ, với $n = 8$ Alice viết số 2, Bob chọn số mới là 3, Alice – số 6. Bob chỉ có một cách chọn duy nhất tiếp theo là 7, Alice chọn số 8 và thắng.

Sau một số lần chơi Alice nhận thấy kết quả trò chơi phụ thuộc vào n và số lượng giá trị n đảm bảo mình thắng không nhiều.

Hãy xác định xem trong đoạn $[a, b]$ có bao nhiêu giá trị khác nhau để nếu Alice chọn thì sẽ thắng nếu cả Alice và Bob đều biết cách đi tối ưu.

Dữ liệu: Vào từ file văn bản WINCHANCE.INP gồm một dòng chứa 2 số nguyên a và b ($1 \leq a \leq b \leq 10^{18}$).

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

Ví dụ:

WINCHANCE.INP
8 10

WINCHANCE.OUT
2



Vq41_OI20150228_C

Chương trình:

```
#include <fstream>
#define NAME "winchance."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
const int INF = (int)1e9;

int64_t chance(int64_t x, int lastPower)
{int64_t t;
 if (x == 0)
    return 0;
 int power = 1;
 while (((int64_t)1 << power) <= x && power < lastPower)
    power += 2;
 power -= 2;
 if (power < 0)
    return 0;
 return ((int64_t)1 << (power >> 1)) + chance(x - ((int64_t)1 << power), power);
}

int main()
{int64_t a,b;
 fi>>a>>b;
 fo<<chance(b,INF)-chance(a-1,INF);
}
```

VQ44. ĐÈN HOA

Tên chương trình: FLOWERS.???

Đèn chuẩn bị một ngày lễ lớn lần thứ k thành phố quyết định dùng đèn LED kết thành k bông hoa trang trí dọc đường phố chính, mỗi bông hoa được kết từ một loại bóng LED cùng màu. Để có hiệu ứng ánh sáng tốt nhất Công ty Chiếu sáng được yêu cầu tạo ra được càng nhiều hoa khác màu càng tốt. Trong kho của Công ty có n bộ đèn LED mỗi bộ lắp được một đèn và bộ thứ i có màu a_i .

Hãy chỉ ra k màu của các bộ đèn cần chọn sao cho số lượng hoa khác màu lắp được là nhiều nhất. Nếu có nhiều cách khác nhau thì đưa ra cách tùy chọn.

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

- ✚ Dòng đầu tiên chứa 2 số nguyên n và k ($1 \leq k \leq n \leq 10^5$),
- ✚ 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 FLOWERS.OUT trên một dòng k số nguyên – màu của các đèn được chọn.

Ví dụ:

FLOWERS.INP
10 4
8 8 8 8 8 8 8 2 1

FLOWERS.OUT
1 2 8 8



Vq44_OI20150315_A

Giải thuật: Bài toán thống kê đơn giản.

- ⊕ Sắp xếp lại mảng a theo thứ tự tăng dần,
- ⊕ Giá trị hàng rào: $a_0 = 0$,
- ⊕ Đếm số bộ đèn khác màu ($a_i \neq a_{i-1}$), đánh dấu bộ đèn đã sử dụng và giảm k ,
- ⊕ Công việc trên được thực hiện cho đến khi $k = 0$ hoặc xét hết các bộ đèn,
- ⊕ Nếu $k \neq 0$ – chọn tiếp k bộ đèn chưa được đánh dấu (duyệt mảng đánh dấu từ cuối về đầu hoặc từ đầu về cuối),
- ⊕ Dưa ra các bộ đèn đã được đánh dấu chọn.

Độ phức tạp của giải thuật: $O(n\log n)$.

Lưu ý: Có thể dùng cấu trúc dữ liệu **set** hoặc **multiset** để tránh sắp xếp, *độ phức tạp* của giải thuật *không thay đổi*.

Chương trình:

```
#include<fstream>
#include<ctime>
#define NAME "flowers."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,k,a[100001],r[100001]={0};

int main()
{clock_t aa=clock();
 fi>>n>>k;
 for(int i=1;i<=n;++i)fi>>a[i];
 sort(a+1,a+n+1);a[0]=0;
 for(int i=1;i<=n;++i)
 {
     if(a[i]!=a[i-1])r[i]=1,--k;
     if(k==0)break;
 }
 if(k)
     for(int i=n;i>0;--i)
     {
         if(r[i]==0)r[i]=1, --k;
         if(k==0)break;
     }
     for(int i=1;i<=n;++i) if(r[i])fo<<a[i]<<' ';
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ45. MUA HÀNG TRỰC TUYẾN

Tên chương trình: BESTBUY.???

Khi được nuôi nhiều để phục vụ các thí nghiệm khoa học và điều chế vắc xin. Ban quản lý đảo khi được cung cấp ngân sách để mua chuỗi làm thức ăn bổ sung cho khỉ. Ngân sách được rót về tài khoản của Ban quản lý từng phần và được chia thành n lần, lần thứ i số tiền a_i được chuyển về tại thời điểm t_i . Ban đầu, số dư tài khoản là 0.

Thông tin trên mạng cho thấy có m công ty có thể cung cấp hàng. Công ty thứ j có hàng và nhận đặt ở thời điểm u_j . Nếu đặt hàng thì lô hàng sẽ được mang tới vào thời điểm v_j , $j = 1 \dots m$. Với mỗi lô hàng bên mua có thể thanh toán trực tuyến vào thời điểm đặt hàng với giá $c1$, khách hàng cũng có thể trả sau vào thời điểm hàng được mang tới với giá $c2$ ($c1 \leq c2$). Hàng chỉ có thể mua được khi có đủ tiền thanh toán ở một trong 2 thời điểm nói trên. Do không biết trước thời điểm tiền được chuyển vào tài khoản nên mọi yêu cầu chào hàng đều được đặt. Nếu đến thời điểm nhận mà số dư tài khoản vẫn không đủ thanh toán thì lô hàng sẽ bị trả về.

Hãy xác định số lượng lô hàng nhiều nhất mà Ban quản lý đảo có thể mua, biết rằng không có thời điểm nào xuất hiện nhiều hơn một nơi có thể đặt hàng, không có 2 đơn hàng nào được mang tới cùng một lúc và không có có thời điểm đặt hàng nào trùng với thời điểm mang hàng tới của công ty khác. Nói một cách hình thức, với $p \neq q$ có $u_p \neq u_q$, $v_p \neq v_q$ và $u_p \neq v_q$.

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

- ✚ Dòng đầu tiên chứa 2 số nguyên $c1$ và $c2$ ($1 \leq c1 \leq c2 \leq 1000$),
- ✚ Dòng thứ 2 chứa số nguyên n ($1 \leq n \leq 10^5$),
- ✚ Dòng thứ i trong n dòng sau chứa 2 số nguyên a_i và t_i ($1 \leq a_i \leq 1000$, $1 \leq t_i \leq 10^9$),
- ✚ Dòng tiếp theo chứa số nguyên m ($1 \leq m \leq 10^5$),
- ✚ Dòng thứ j trong m dòng sau chứa 2 số nguyên u_j và v_j ($1 \leq u_j \leq v_j \leq 10^9$).

Kết quả: Đưa ra file văn bản BESTBUY.OUT một số nguyên – số lượng lô hàng nhiều nhất mà Ban quản lý đảo có thể mua.

Ví dụ:

BESTBUY.INP
100 200
3
100 1
200 10
400 21
4
12 22
2 4
5 23
8 19

BESTBUY.OUT
3



Giải thuật:

Tổ chức dữ liệu:

- ✚ Mảng `pair<int, int> a[100001]` quản lý thời điểm nhận tiền và số tiền:

- **a_i.first** – thời điểm nhận tiền,
- **a_i.second** – số tiền chuyển vào tài khoản,
- + Mảng **pair<int, int> p[200002]** quản lý các thời điểm nhận đặt hàng và giao hàng:
 - ❖ Nhận đặt hàng: **pi** = (*Thời điểm*, *-i*),
 - ❖ Thời điểm giao hàng: **pi** = (*Thời điểm*, *i*),
- + Mảng **int d[100001={0}]** – đánh dấu hàng đã thanh toán,
- + **int64_t sum=0** – số dư tài khoản ban đầu,
- + **int ans = 0** – số lô hàng đã mua được.

Xử lý:

- + Nhập dữ liệu, ghi nhận vào **a** và **p**,
- + Sắp xếp tăng dần **a** và **p**,
- + Tạo giá trị hàng rào: gán thêm vào cuối **a** phần tử có thời điểm muộn hơn thời điểm giao hàng cuối cùng và số tiền là 0,
- + Duyệt mọi thời điểm đặt/giao hàng (**i** = 1 ÷ 2×**m**) nếu hàng chưa được mua:
 - ❖ Cập nhật lại số dư tài khoản cho tới thời điểm đặt/giao hàng đang xét,
 - ❖ Nếu đủ tiền – thanh toán, đánh dấu mua và cập nhật **ans**.

Lưu ý: Dùng mảng tĩnh sẽ cho chương trình có thời gian thực hiện nhỏ hơn so với chương trình dùng mảng động kiểu **vector**.

Độ phức tạp của giải thuật: thuộc lớp O(nlogn)

Chương trình:

```
#include <fstream>
#include <cmath>
#include <ctime>
#define NAME "bestbuy."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,m,d[100001]={0},tg,tp,k=0,c1,c2,ans,u,v;
pair<int,int> a[100001],p[200002];
int64_t sum=0;

int main()
{clock_t aa=clock();
 fi>>c1>>c2>>n;
 for(int i=0;i<n;++i)fi>>a[i].second>>a[i].first;
 fi>>m;
 for(int i=1;i<=m;++i)
 {
     fi>>u>>v; p[2*i]=make_pair(u,-i),p[2*i+1]=make_pair(v,i);
 }
 sort(a,a+n); sort(p,p+2*m+1);tg=p[2*m].first+1; a[n]=make_pair(tg,0);
 for(int i=1;i<=2*m;++i)
 {
     tg=p[i].first; tp=abs(p[i].second);
     if(d[tp]==0)
     {
         while(a[k].first<=tg)sum+=a[k++].second;
         if(p[i].second<0 && sum>=c1){d[tp]=1;sum-=c1;++ans;}
         else if(p[i].second>0 && sum>=c2){d[tp]=1; sum-=c2;++ans;}
     }
 }
 fo<<ans;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ46. LỄ HỘI ĐƯỜNG PHỐ

Tên chương trình: CARNIVAL.???

Cuộc sống thay đổi, một số lễ hội bị rơi vào lãng quên và thế chỗ vào đó là một số lễ hội mới. Tin học phát triển đã làm xuất hiện lễ hội đường phố Tin học. Dẫn đầu đoàn diễu hành là dàn nhạc đánh trống và thổi kèn. Khối có n hàng, mỗi hàng có n người, mỗi người mặc một áo phông có in số nguyên tố không vượt quá 10^7 sau lưng. Ban Tổ chức yêu cầu phải phát áo cho mọi người sao cho thỏa mãn được các điều kiện:

- Trong toàn khối có đúng k số nguyên tố khác nhau,
- Tích các số in trên áo ở mỗi hàng và mỗi cột đều có số lượng ước số như nhau.

Hãy đưa ra một cách bố trí số các áo cần mặc trong khối để thỏa mãn các yêu cầu đề ra. Nếu không có cách bố trí thì đưa ra số -1.

Dữ liệu: Vào từ file văn bản CARNIVAL.INP gồm một dòng chứa 2 số nguyên k và n ($1 \leq k \leq 10^9$, $1 \leq n \leq 1000$).

Kết quả: Đưa ra file văn bản CARNIVAL.OUT trên n dòng, mỗi dòng – n số nguyên tố xác định cách bố trí các áo trong khối diễu hành. Nếu không có cách bố trí thì đưa ra một số -1.

Ví dụ:

CARNIVAL.INP
2 3

CARNIVAL.OUT
2 3 2
3 2 3
2 3 2



Vq46 OI20150315 C

Giải thuật:

Tạo bảng số nguyên tố có giá trị không vượt quá 10^7 , lưu vào vector **primes**:

- ⊕ Dùng mảng **bool p[10000010]** đánh dấu các hợp số,
- ⊕ Nếu i là số nguyên tố thì các số $i^2, i^2+i, i^2+2i, i^2+3i, \dots$ sẽ bị đánh dấu là hợp số,
- ⊕ Sau khi đánh dấu hợp số xong: nạp các số nguyên tố vào **primes**.

```

for (int i = 2; i * i <= N; ++i)
    if (!p[i])
        for (int j = i * i; j <= N; j += i)
            p[j] = true;
    for (int i = 2; i <= N; ++i)
        if (!p[i])
            primes.push_back(i);
    
```

Bố trí bảng số nguyên tố:

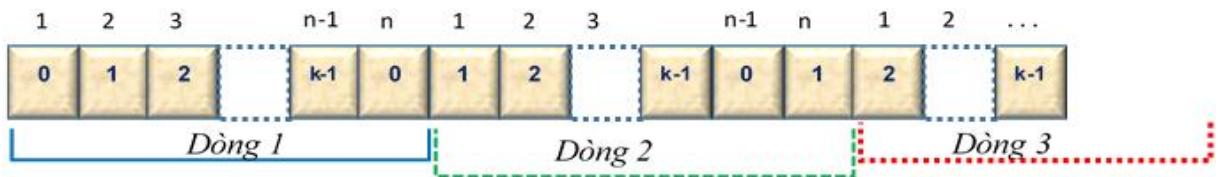
- ⊕ Phân lập các trường hợp vô nghiệm:
 - ❖ $k > n^2$,
 - ❖ k lớn hơn số lượng số nguyên tố có thể sử dụng.

Xử lý trường hợp có nghiệm:

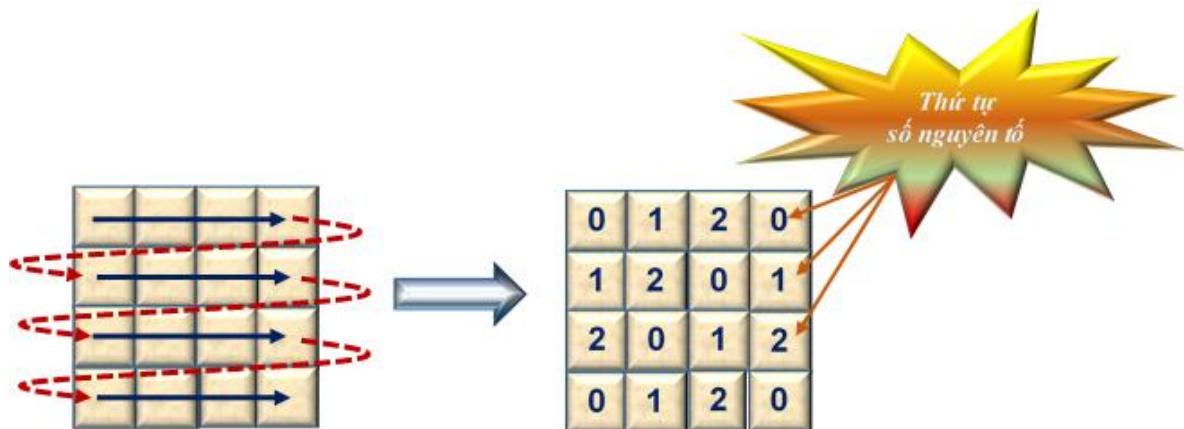
- ⊕ Tích các số trong mỗi hàng và mỗi cột đều có 2 ước số hiển nhiên: 1 và chính nó, như vậy chỉ cần xét số lượng ước không hiển nhiên,
- ⊕ Số lượng ước không hiển nhiên phụ thuộc vào:
 - ❖ Số các số nguyên tố khác nhau trong hàng/cột,
 - ❖ Số mũ của các thừa số.

Phân biệt 2 trường hợp: $k \leq n$ và $k > n$,

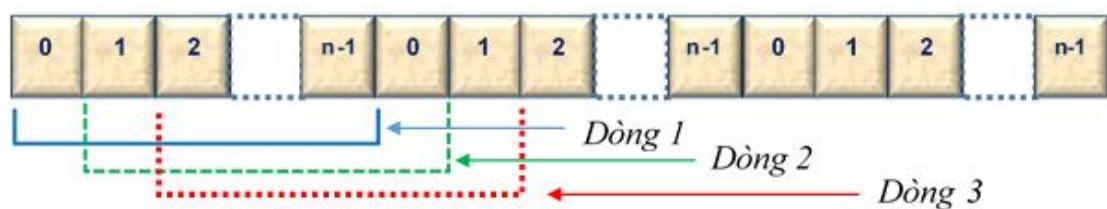
- ⊕ Trường hợp $k \leq n$: đầy vòng tròn k số nguyên tố đầu tiên, ghi kết quả đầy vào bảng theo dòng.
- ⊕ Trường hợp $k > n$:
 - ❖ Diền số sao cho mọi số trong cột là khác nhau và mọi số trong hàng là khác nhau,
 - ❖ Nếu đánh số các ô tuyển tính từ trái sang phải và từ trên xuống dưới:
 - k ô đầu tiên – đánh số theo kiểu vòng tròn các số nguyên tố với thứ tự từ 0 đến $n-1$, sau mỗi hàng – bỏ qua một số,
 - $n^2 - k$ ô còn lại – gán các số nguyên tố mới (với số từ $k-1$ lùi về n).



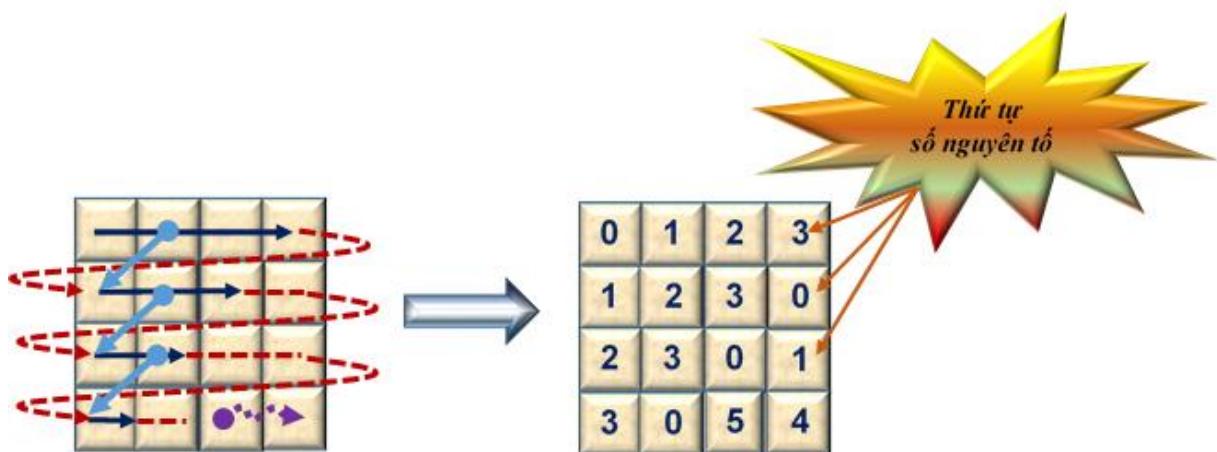
Cách đánh số vòng tròn khi $k \leq n$



Ví dụ: $k = 3, n = 4$



Cách đánh số vòng tròn khi $k > n$



Ví dụ: $k = 6, n = 4$

Chương trình:

```
#include <fstream>
```

```

#include <vector>
#include <ctime>
using namespace std;
#define NAME "carnival."
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
const int N = 10000000;
int k, n;
bool p[N + 10];
vector <int> primes;

void dispose()
{if (primes.size() < k || k > n * n)
    fo << -1 << endl;
else if (k <= n) {
    for (int i = 0; i < n; ++i, fo<<'\n')
        for (int j = 0; j < n; ++j)
            fo<<primes.at((i + j) % k)<< ' ';
} else {
    int i = 0;
    while (i < n * n) {
        if (n * n - i > k - n)
            fo<<primes.at((i + i / n) % n)<< ' ';
        else
            fo<<primes.at(n * n - i - 1 + n)<< ' ';
        ++i;
        if (i % n == 0)
            fo<<'\n';
    }
}
}

int main()
{clock_t aa=clock();
fi >> k >> n;
for (int i = 2; i * i <= N; ++i)
    if (!p[i])
        for (int j = i * i; j <= N; j += i)
            p[j] = true;
for (int i = 2; i <= N; ++i)
    if (!p[i])
        primes.push_back(i);
dispose();

clock_t bb=clock();
fo<<"\nTime: "<<(double) (bb-aa)/1000<<" sec";
}

```

VQ42. BÁNH KẸP

Tên chương trình: BURGER.???

Cửa hàng *Snow Crab* có món bánh mì kẹp hải sản cực kỳ nổi tiếng. Tiêu chí phục vụ của cửa hàng là trong ngày không có hai khách hàng nào nhận được bánh giống nhau. Các món đặt vào bánh kẹp được đánh giá phân loại bởi chỉ số nguyên không âm a_1, a_2, a_3, \dots . Nếu bánh được kẹp các món với chỉ số a_1, a_2, \dots, a_m thì chỉ số của bánh thành phẩm sẽ là $a_1 \oplus a_2 \oplus \dots \oplus a_m$, trong đó \oplus là phép tính **xor** (phép \wedge trong C++). Các món được chuẩn bị cho ngày nào thì chỉ dùng trong ngày đó và phải *dùng hết trong ngày*.

Nhân viên cơ quan, công sở không còn đi tới đón các cửa hàng Cơm trưa văn phòng, thay vào đó họ gửi yêu cầu tới *Snow Crab* và được mang bánh tới tận tay.

Hôm nay cửa hàng nhận được k yêu cầu, yêu cầu thứ i muốn có bánh với chỉ số không nhỏ hơn u_i và không lớn hơn v_i , $i = 1 \div k$. Bộ phận chế biến cung cấp n món. Các món được băng chuyên chuyển sang bộ phận kẹp bánh đóng gói thành phẩm. Bộ phận đóng gói sẽ lấy ra dây các món liên tục (tính từ đầu theo trình tự xuất hiện), đóng thành một chiếc bánh kẹp, cho vào hộp và chuyển tới địa chỉ đặt hàng. Tất cả đều được tự động hóa, nhưng bếp trưởng muốn biết với các món đã chuẩn bị thì có nhiều cách khác nhau nhau đáp ứng yêu cầu của khách hàng hay không để không phải chuẩn bị thêm các món loại khác. Các yêu cầu được đáp ứng theo trình tự gửi tới nhà hàng.

Hãy đưa ra số lượng cách đóng gói khác nhau có thể thực hiện. Hai cách đóng gói gọi là khác nhau nếu tồn tại ít nhất một món thuộc một bánh trong cách đóng gói thứ nhất và thuộc bánh khác trong cách đóng gói thứ hai.

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

- ✚ Dòng đầu tiên chứa 2 số nguyên n và k ($1 \leq n \times k \leq 10^5$, $k \leq n$),
- ✚ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($0 \leq a_i \leq 10^9$, $i = 1 \div n$),
- ✚ Dòng thứ j trong k dòng sau chứa 2 số nguyên u_j và v_j ($0 \leq u_j \leq v_j \leq 10^9$).

Kết quả: Đưa ra file văn bản BURGER.OUT số lượng cách đóng gói khác nhau có thể thực hiện theo mô đun $10^9 + 7$.

Ví dụ:

BURGER.INP
7 3
1 0 1 0 1 0 1
1 1
0 0
1 1

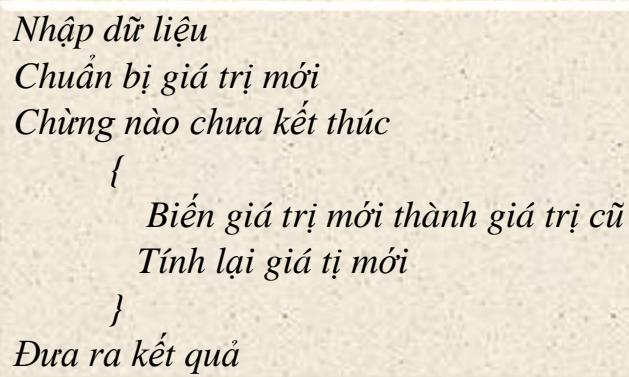
BURGER.OUT
6



Giải thuật: Sơ đồ lặp (Quy hoạch động đơn giản)

Nhận xét:

- ✚ Bài toán có mô hình toán học tương tự bài toán lát nền,
- ✚ Giải thuật áp dụng: sơ đồ lặp (*recurrent*) hay còn gọi là quy hoạch động đơn giản,
- ✚ Kích thước không lớn: $n \times k \leq 10^5$, vì vậy không cần áp dụng các tiêu xảo đặc biệt trong tổ chức dữ liệu và xử lý.
- ✚ Nguyên tắc tổ chức chương trình theo sơ đồ lặp:



Lưu ý: Phụ thuộc vào cách chuẩn bị giá trị đầu để lặp, việc hoán đổi giá trị mới và cũ có thể thực hiện trước hoặc sau khi tính giá trị mới.

Tổ chức dữ liệu:

- ✚ Mảng **int a[100001]** – lưu thông tin về các món,
- ✚ Mảng **int f[2][100001]** – chứa kết quả lặp cũ và mới,
- ✚ Các biến **p** và **q** – quản lý dòng cũ và mới.

Xử lý:

- ✚ Nhập dữ liệu: nhập **n**, **k** và **a_i**, **i = 1 ÷ n**,
- ✚ Chuẩn bị lặp:
 - ❖ Có thể chỉ cần **p=0; q=1; f[0][0]=1;**
 - ❖ Thông thường sơ đồ tính lặp không phức tạp nên người ta thường tính ngay bước lặp đầu tiên và dùng nó làm giá trị đầu.

```
p=0; q=1;
fi>>u>>v; t=0;
for(int i=1;i<=n-k+1;++i)
{
    t^=a[i]; if(u<=t && t<=v) f[p][i]=1;
}
```

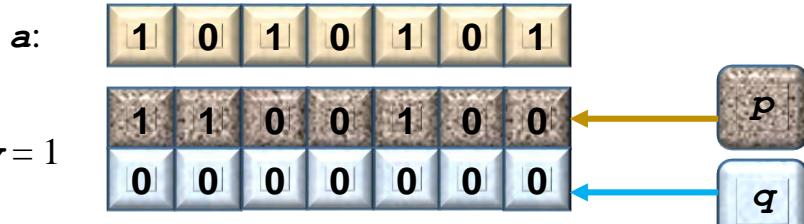
- ✚ Chu trình lặp:
 - ❖ Nhập **u**, **v** mới,
 - ❖ Xóa 0 mảng chứa kết quả mới,
 - ❖ Tìm các điểm được đánh dấu ở bước cũ,

- ❖ Với mỗi điểm tìm được: ghi nhận tích lũy khả năng tới vị trí mới tiếp sau,
- ❖ Hoán đổi giá trị của p và q .

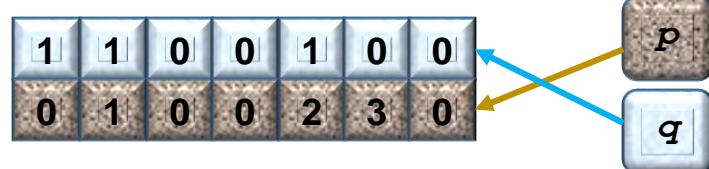
➡ Kết quả: Đưa ra $f[p] [n]$.

Ví dụ: $n = 7$, $k = 3$, $a = (1, 0, 1, 0, 1, 0, 1)$

Chuẩn bị: $u = 1$, $v = 1$



Bước lặp 1: $u = 0$, $v = 0$



Bước lặp 2: $u = 1$, $v = 1$



Chương trình:

```
#include <fstream>
#include <ctime>
#define NAME "burger."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
const int p7=1000000007;
int n,k,a[100001],p,q,u,v,t;
int f[2][100001]={0};

void ext_f(int x)
{int ib;
 fi>>u>>v;
 for(int i=1;i<=n;++i)f[q][i]=0;
 for(int i=1;i<=n+x-k;++i)
 {
 if(f[p][i])
 {ib=i;t=0;
 for(int j=ib+1;j<=n+x-k;++j)
 {t^=a[j]; if(u<=t && t<=v)f[q][j]=(f[p][ib]+f[q][j])%p7;}
 }
 p^=1;q^=1;
 }
 int main()
 {clock_t aa=clock();
 fi>>n>>k;
 for(int i=1;i<=n;++i)fi>>a[i];
 p=0; q=1;
 fi>>u>>v; t=0;
 for(int i=1;i<=n-k+1;++i)
 {
 t^=a[i]; if(u<=t && t<=v)f[p][i]=1;
 }
 for(int i=2;i<=k;++i) ext_f(i);
 fo<<f[p][n];
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
 }
```

VQ43. TÂM THẢM

Tên chương trình: CARPET.???

Cần hoàn thiện gấp sân bay quốc tế để kịp đón một nguyên thủ quốc gia tới thăm. Theo thiết kế có 2 phòng VIP – Phòng trái và Phòng phải. Hai phòng này phải được trang trí nội thất giống nhau để tránh mọi rắc rối ngoại giao có thể xảy ra. Thảm trải sàn được chở tới là một cuộn được khâu từ n tấm thảm cẩm độ dài giống nhau. Mỗi tấm thảm cẩm thuộc một trong số 26 loại khác nhau và được ký hiệu bằng một chữ cái la tinh thường. Như vậy cuốn thảm được xác định bởi xâu s độ dài n chỉ chứa các ký tự la tinh thường. Thời gian còn lại rất ít vì vậy người ta quyết định tháo chỉ khâu ở 2 chỗ, lấy đoạn giữa đưa vào lót sàn Phòng trái, trong thời gian đó những người còn lại sẽ dùng toàn bộ phần bên trái còn lại hoặc tháo chỉ ở một hay 2 nơi của phần này, bỏ đi các phần đầu và cuối nếu cần để được một tấm thảm con, khâu với tấm thảm con nhận được từ phần bên phải còn lại từ cuộn thảm ban đầu. Hai tấm thảm con được khâu lại theo trình tự trái phải ban đầu. Tấm thảm mới phải giống hệt tấm thảm đã mang vào Phòng trái.

Nói một cách hình thức, phải chọn cặp số i, j ($1 < i \leq j < n$) sao cho tồn tại các số $i1, j1, i2, j2$ thỏa mãn các điều kiện:

- $1 \leq i1 \leq j1 < i, j < i2 \leq j2 \leq n$,
- $s[i..j] = s[i1..j1] + s[i2..j2]$, trong đó $s[u..v]$ là xâu con các ký tự liên tiếp từ u tới v .

Hãy xác định có bao nhiêu cách chọn tấm thảm ban đầu lót ở sàn Phòng trái.

Dữ liệu: Vào từ file văn bản CARPET.INP gồm một dòng chứa xâu s độ dài không quá 10^5 và chỉ chứa các ký tự chữ cái la tinh thường.

Kết quả: Đưa ra file văn bản CARPET.OUT một số nguyên – số cách chọn có thể thực hiện.

Ví dụ:

CARPET.INP
abababb

CARPET.OUT
3



Vq43 OI20150228 E

Giải thuật:

Chương trình:

```
#include <fstream>
#include <cmath>
#include <string>
#include <ctime>
#define NAME "carpet."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,p,q,n2,nf,tf=0,m,f[150010]={0},a[100010],ans=0;;
string s;

void insert_t(int ii)
{
    while(ii<=nf){++f[ii];ii+=(ii&(-ii));}
}

void upd_t(int ii)
{
    while(ii<=nf){--f[ii];ii+=(ii&(-ii));}
}

int sum_t(int ii)
{int r=0;
while(ii>0){r+=f[ii]; ii&=(ii-1);}
return r;
}

int maxsuff(int x)
{char ch; int k=0,t;
m=min(n2,x);
ch=s[x];
for(int i=n-1;i>x;--i)
{ if(s[i]==ch)
    {p=min(m,i-x);t=0;
     for(int j=0;j<p;++j)if(s[i-j]==s[x-j])++t;else break;
     k=max(t,k);
    }
}
return k;
}

void maxprefix(int x)
{char ch; int k=0,t;
ch=s[x]; m=min(n2,n-x-1);
for(int i=0;i<x;++i)
    if(s[i]==ch)
    {p=min(m,x-i);t=0;
     for(int j=0;j<p;++j)if(s[i+j]==s[x+j])++t;else break;
     k=max(t,k);
    }
a[x]=(k>0)? k+x:0;
if(a[x]>0){insert_t(a[x]); ++tf;}
}

int main()
{clock_t aa=clock();
fi>>s; n=s.size(); n2=n>>1;nf=n+n2;
for(int i=1;i<n-2;++i) maxprefix(i);
int tma,tmb;
```

```
for(int i=n-2;i>1;--i)
{
    if(a[i]>0){upd_t(a[i]); --tf;}
    tmb=maxsuff(i);
    if(tmb>0)
    {
        tmb=i-tmb;
        tma=tf-sum_t(tmb);
        ans+=tma;
    }
    fo<<ans;
clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ49. SAN BẰNG

Tên chương trình: ALIGN.???

Để rèn luyện sự khéo tay và kiên nhẫn cô giáo đề xuất các bé trong lớp chơi trò dựng cột nhà. Mỗi bé phải xếp một cột từ các khối hình vuông đơn vị đặt chồng lên nhau. Sau một thời gian các bé bắt đầu thấy nhảm chán và chuyển sang chơi trò khác. Trên sàn còn lại n cột, cột thứ i có độ cao a_i . Jimmy vẫn muốn chơi tiếp với các khối lập phương nên ở lại. Jimmy không thích việc các cột nhấp nhô như răng cưa và cho rằng muốn xây nhà đẹp thì phải có các cột cao như nhau. Vốn được thường xuyên giáo dục ở nhà là muốn có kết quả tốt thì không những phải có ý tưởng, biết ước mơ mà còn phải biết hành động để biến các ước mơ, ý tưởng đó thành hiện thực, Jimmy quyết tâm làm bằng các cột.

Trong một đơn vị thời gian Jimmy có thể thực hiện một trong 3 thao tác:

- Chuyển một khối vuông từ đỉnh cột này sang đỉnh cột khác,
- Bỏ một khối vuông từ đỉnh một cột ra ngoài,
- Lấy thêm một khối vuông ở ngoài đặt vào đỉnh một cột. Số khối vuông còn thừa ở ngoài rất nhiều.

Hãy xác định thời gian ít nhất cần thiết để Jimmy có thể làm các cột có độ cao giống nhau.

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

- ✚ Dòng đầu tiên chứa số nguyên n ($1 \leq n \leq 1\,000$),
- ✚ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($1 \leq a_i \leq 1\,000$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản ALIGN.OUT một số nguyên – thời gian tối thiểu tìm được.

Ví dụ:

ALIGN.INP
5
3 2 2 5 4

ALIGN.OUT
3



Giải thuật: Kỹ năng phân tích mô hình toán học

Gọi \mathbf{h} là độ cao mỗi cột sau khi làm bằng,

Có 2 khả năng:

 $\mathbf{h} = (\sum \mathbf{a}_i) / \mathbf{n}$ *(Làm tròn xuống)*,

 $\mathbf{h} = (\sum \mathbf{a}_i + \mathbf{n} - 1) / \mathbf{n}$ *(Làm tròn lên)*.

Xét cả 2 trường hợp (nếu các độ cao này khác nhau) và chọn kết quả nhỏ hơn.

Tính số lượng thao tác:

- Tính $\mathbf{t} = \mathbf{a}_i - \mathbf{h}$, $i = 1 \div \mathbf{n}$,
- Nếu $\mathbf{t} < 0$ – phải lấp thêm các khối lập phương,
- Nếu $\mathbf{t} > 0$ – bỏ bớt các khối lập phương,
- Một cặp thao tác (*Lấp thêm, Bỏ bớt*) tương ứng với một thao tác loại 1.

Độ phức tạp của giải thuật: $O(n)$.

Chương trình:

```
#include <fstream>
#include <iostream>
#include <cmath>
#include <ctime>
#include <iomanip>
#define NAME "align."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,a[1001],t,tn,tp,res1,res2,ans,h1,h2,m=0;

int main()
{clock_t aa=clock();
 fi>>n;
 for(int i=0;i<n;++i){fi>>a[i]; m+=a[i];}
 h1=m/n; h2=(m+n-1)/n;
 t=0;tn=0;tp=0;
 for(int i=0;i<n;++i){t=a[i]-h1;if(t<=0)tn+=t;else tp+=t;}
 res1=max(-tn,tp);
 if(h1!=h2)
 {   t=0;tn=0;tp=0;
     for(int i=0;i<n;++i){t=a[i]-h2;if(t<=0)tn+=t;else tp+=t;}
     res2=max(-tn,tp);
 }
 if(h1==h2)ans=res1; else ans=min(res1,res2);
 fo<<ans;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VQ50. XẾP HÀNG

Tên chương trình: LONGQUEUE.???

McLaine là nhân viên bảo vệ ở một ngân hàng. Công việc khá buồn tẻ nhưng căng thẳng. Hiện tại đang có n người đứng xếp hàng thực hiện các giao dịch. McLaine đánh số trong đầu một cách tự động các người đang xếp hàng từ 0 đến $n-1$, bắt đầu từ người đang thực hiện giao dịch. Như vậy người thứ i là người mà trong hàng có i người đứng trước. Như vậy McLaine đánh số mọi người theo vị trí đứng của họ trong dòng xếp hàng. Phản xạ nghề nghiệp và kinh nghiệm lâu năm cho phép McLaine xác định được ngay trạng thái tâm lý của người thứ i là a_i . Nếu $a_i \geq x$ thì đó là người có trạng thái tâm lý tốt và nói chung là đối tượng không nguy hại, không cần chú ý nhiều. Ngược lại, nếu $a_i < x$ thì đó là người có trạng thái tâm lý không tốt và cần phải để ý tới họ nhiều hơn.

Theo thời gian, một số người thực hiện giao dịch xong và rời đi, một số người tới đứng vào cuối hàng. Khi có người mới tới McLaine tự động đánh giá ngay trạng thái tâm lý của họ. Trạng thái này không thay đổi trong suốt quá trình xếp hàng.

Khi trong hàng một người nào đó ở vị trí thứ i có một hành động gì như cho tay vào túi hay mở cắp lập tức McLaine để ý đến họ ngay và tự động tính trong đầu xem trước người đó có bao nhiêu người có trạng thái tâm lý tốt – chở dựa cho McLaine nếu có tình huống xấu xảy ra. Trong suốt ca làm việc hôm nay của McLaine có m sự kiện thuộc một trong 3 loại:

- **1** b – một người mới tới đứng vào hàng với trạng thái tâm lý b ($0 \leq b \leq 10^9$),
- **2** – một khách hàng đã giao dịch xong và rời đi,
- **3** i – cần lưu ý hành động người ở vị trí i trong dòng xếp hàng.

Với mỗi sự kiện loại **3** hãy cho biết có bao nhiêu người có trạng thái tâm lý tốt đứng trước vị trí i trong hàng. Thông tin về các sự kiện là hợp lý, tức là không có sự kiện loại **2** khi dòng xếp hàng rỗng hay giá trị i trong sự kiện loại **3** lớn hơn số người xếp hàng.

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

- ✚ Dòng đầu tiên chứa 2 số nguyên n và x ($1 \leq n = 10^5$, $0 \leq x \leq 10^9$),
- ✚ Dòng thứ i trong n dòng tiếp theo chứa số nguyên a_i ($0 \leq a_i \leq 10^9$),
- ✚ Dòng tiếp theo chứa số nguyên m ($1 \leq m \leq 10^5$),
- ✚ Mỗi dòng trong m dòng sau đó chứa thông tin về một sự kiện.

Kết quả: Đưa ra file văn bản LONGQUEUE.OUT, với mỗi sự kiện loại **3** đưa ra số người tính được, mỗi số trên một dòng.

Ví dụ:

LONGQUEUE.INP
1 2
3
5
1 2
1 1
3 0
3 1
3 2

LONGQUEUE.OUT
0
1
2



Vq50 R OI20150329 B

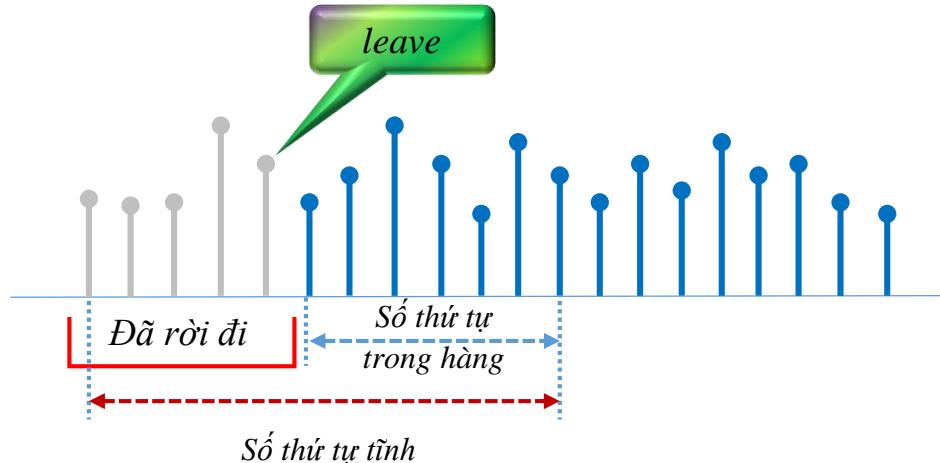
Giải thuật: Tổng tiền tố

Không thay đổi số thứ tự của người xếp hàng (*số thứ tự tĩnh*),

Quản lý điểm đầu dòng xếp hàng,

Tạo tổng tiền tố: f_i – lưu số người có trạng thái tâm lý tốt trong đoạn $[0, i]$,

Để thuận tiện xử lý: chuyển sang đánh số bắt đầu từ 1, $f_0 = 0$.



Chương trình:

```
#include <fstream>
#include <iostream>
#include <cmath>
#include <ctime>
#include <iomanip>
#define NAME "longqueue."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,k,x,a,m,t,c,r,f[200001]={0},leave=0;

int main()
{clock_t aa=clock();
 fi>>n>>x; k=n;
 for(int i=1;i<=n;++i){fi>>a; f[i]=f[i-1]+(a>=x) ;}
 fi>>m;
 for(int i=0;i<m;++i)
 {
     fi>>c;
     switch(c)
     {
         case 1:fi>>a; ++k; f[k]=f[k-1]+(a>=x) ;break;
         case 2:++leave; break;
         case 3:fi>>t; t+=leave+1; r=f[t-1]-f[leave];
                  fo<<r<<'\'n'; break;
     }
 }

clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VR01. SÁT NHẬP

Tên chương trình: COMPANIES.???

Hai công ty sát nhập với nhau. Hội đồng quản trị có n người và có nhiệm vụ bầu một thành viên Hội đồng quản trị làm Tổng giám đốc mới. Thể lệ bầu cử là mỗi người chỉ được bỏ một phiếu và người được lựa chọn trong lá phiếu phải khác công ty ban đầu với người bỏ lá phiếu đó.

Việc bỏ phiếu đã hoàn tất và kết quả kiểm phiếu sơ bộ được trình lên Chủ tịch Hội đồng quản trị. Đó là một danh sách n số nguyên, số nguyên a_i cho biết số phiếu mà người thứ i nhận được.

Thông tin bị dò rỉ ra ngoài và mọi người muốn dựa trên kết quả bỏ phiếu xác định người thứ i trong Hội đồng quản trị thuộc công ty nào trong số các công ty 1 và 2 trước khi sát nhập, $i = 1 \div n$. Thông tin này cũng có thể chứa sai sót, vì vậy cần kiểm tra tính đúng đắn về mặt lô gic của nó.

Hãy kiểm tra tính đúng đắn, đưa ra thông báo “**YES**” nếu số liệu hợp lý về mặt lô gic hoặc “**NO**” trong trường hợp ngược lại. Nếu dữ liệu hợp lý thì đưa ra một môt phán đoán ai ở Công ty ban đầu nào trong số các thành viên của Hội đồng quản trị.

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

- ✚ Dòng đầu tiên chứa số nguyên n ($2 \leq n \leq 10^5$),
- ✚ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($0 \leq a_i < n$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản COMPANIES.OUT, dòng đầu tiên chứa thông báo “**YES**” hoặc “**NO**”. Nếu kết quả hợp lý thì đưa ra trên dòng thứ 2 n số thuộc tập {1, 2} cho biết người thứ i trong Hội đồng quản trị thuộc Công ty ban đầu nào, $i = 1 \div n$.

Ví dụ:

COMPANIES.INP
5
1 2 0 2 0

COMPANIES.OUT
YES
1 2 2 1 2



Vq43 OI20150228 E

Giải thuật: Tư duy toán học

Nhận xét:

- ⊕ Gọi $b_i = a_i + 1$,
- ⊕ Dễ dàng nhận thấy $\sum b_j = n$ với các j thuộc một công ty.
- ⊕ Như vậy bài toán đưa về việc tìm tập chỉ số j thỏa mãn điều kiện trên.
- ⊕ Bài toán vô nghiệm khi và chỉ khi thỏa mãn 2 điều kiện:
 - ❖ n là lẻ,
 - ❖ $a_i = 1$ với mọi $i = 1, 2, \dots, n$.

(các b_i đều chẵn, nhưng cần có một trong 2 tổng $\sum b_j$ phải lẻ!)

Sơ đồ xử lý:

- ⊕ Nhập dữ liệu và tạo mảng cặp giá trị $B = \{(b_i, i)\}$
- ⊕ Sắp xếp mảng B theo thứ tự giảm dần,
- ⊕ Dùng mảng d đánh dấu phân loại công ty,
- ⊕ Tìm phần đầu của mảng B có tổng $b_i \leq n$, đánh dấu những người tương ứng thuộc Công ty 1,
- ⊕ Nếu có tổng bằng n – kết thúc xử lý, trong trường hợp ngược lại – bổ sung vào tổng các giá trị b_j với $j = n, n-1, n-2, \dots$ cho đến khi nhận được tổng bằng n , đánh dấu những người tương ứng thuộc Công ty 1.

Độ phức tạp của giải thuật: $O(n \log n)$.

Chương trình:

```
#include <fstream>
#include <iostream>
#include <cmath>
#include <ctime>
#include <iomanip>
#define NAME "companies."

using namespace std;
typedef pair<int,int> pi;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
pi a[100001];
int n,t,d[100001]={0},vote=0;
/*
void check_res()
{int sum1=0,sum2=0;
for(int i=0;i<n;++i) if(d[a[i].second])sum1+=a[i].first; else
sum2+=a[i].first;
if(sum1==n && sum2==n) fo<<"\nYES";else fo<<"\nNO";
}
*/
int main()
{clock_t aa=clock();
fi>>n;
for(int i=0;i<n;++i){fi>>t; a[i].first=t+1;a[i].second=i;}
sort(a,a+n,greater<pi>());
if((n&1) && (a[0].first==2) && a[n-1].first==2){fo<<"NO"; return
0;}
fo<<"YES\n";
for(int i=0;i,n;++i)
if(vote+a[i].first<=n)
{vote+=a[i].first; d[a[i].second]=1; if(vote==n)break;}
if(vote<n)
for(int i=n-1;i>0;--i)
{
    vote+=a[i].first;d[a[i].second]=1; if(vote==n)break;
}
for(int i=0;i<n;++i) fo<<((d[i])? 1:2)<<' ';
//check_res();
clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VR02. BÁNH NGỌT

Tên chương trình: CAKES.???

Cocoros là một tỷ phú và là Giám đốc công ty phần mềm “OSModern”. Nhân ngày sinh nhật của Giám đốc các nhân viên tổ chức một bữa tiệc chúc mừng. Bánh trái được bày trên một bàn dài chạy thẳng từ chân cầu thang ra cổng chính của Công ty, đặc biệt là có n bánh ngọt, chiếc bánh thứ i được đặt ở vị trí \mathbf{x}_i tính từ đầu bàn ở chân cầu thang. Mọi người đều biết là Giám đốc rất thích đồ ngọt! Bản thân Cocoros cũng rất ngạc nhiên và cảm động trước thịnh tình của các nhân viên. Ông tuyên bố sẽ cố gắng ăn càng nhiều bánh ngọt càng tốt để không phụ công mọi người trong việc chuẩn bị tiệc cũng như trong suốt quá trình xây dựng và phát triển công ty.

Tỷ phú thì không bao giờ có nhiều thời gian, ông chỉ có thể dành T đơn vị thời gian ăn uống với mọi người. Liếc mắt nhìn qua bàn tiệc ông biết rằng để ăn chiếc bánh thứ i (ở vị trí \mathbf{x}_i) sẽ cần t_i thời gian. Để đi từ vị trí i đến vị trí j ông cần $|\mathbf{x}_i - \mathbf{x}_j|$ thời gian. Ở cùng một vị trí có thể có nhiều bánh, việc di chuyển là không cần thiết nhưng bánh thì phải ăn lần lượt từng chiếc.

Xuất phát từ vị trí 0, hãy xác định số bánh nhiều nhất Cocoros có thể ăn trong khoảng thời gian T .

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

- ✚ Dòng đầu tiên chứa 2 số nguyên n và T ($1 \leq n \leq 10^5$, $1 \leq T \leq 10^9$),
- ✚ Dòng thứ i trong n dòng sau chứa 2 số nguyên \mathbf{x}_i và t_i ($1 \leq \mathbf{x}_i, t_i \leq 10^9$, với $i < j$ có $\mathbf{x}_i \leq \mathbf{x}_j$).

Kết quả: Đưa ra file văn bản CAKES.OUT một số nguyên – số bánh tối đa có thể được ăn.

Ví dụ:

CAKES.INP
8 100
1 21
3 10
4 3
5 19
8 8
9 32
50 1
100 1

CAKES.OUT
5



Giải thuật: Quy hoạch động

Nhận xét:

- ⊕ Một chiếc bánh, nếu nằm trong danh sách ăn thì cần ăn ở thời điểm gấp lần đầu tiên, trong trường hợp ngược – sẽ mất thêm chi phí thời gian quay lui,
- ⊕ Như vậy đây là bài toán mô hình quy hoạch động,
- ⊕ Trong phần lớn các bài toán quy hoạch động việc sử dụng cây nhị phân tìm kiém cân bằng (các cấu trúc dữ liệu heap, priority_queue, set, multiset) sẽ giúp giảm độ phức tạp của giải thuật.

Sơ đồ xử lý:

- Gọi **C** là tập chỉ số các bánh đã ăn cho đến thời điểm hiện tại,
- Khi đó chi phí thời gian để ăn thêm được một bánh ở vị trí thứ **i** xa hơn sẽ là:

$$tm = \sum_{j \in C} t_j + t_i + x_i$$

Thời gian ăn *Thời gian
di chuyển*

- Thời gian di chuyển chỉ phụ thuộc vào chiếc bánh xa nhất cần ăn.
- Gọi thời gian ăn là **sumt**,
- Nếu $tm \leq T$ thì cập nhật lại số lượng bánh đã ăn và giá trị **sumt**,
- Nếu $tm > T$ thì để có thể ăn được bánh ở vị trí **x_i** cần loại bỏ khỏi **C** một số bánh đã ăn theo phương án trước đó,
- Những bánh cần loại bỏ là bánh có t_j lớn nhất, $j \in C$.
- Như vậy cần quản lý các t_j bằng hàng đợi ưu tiên (theo thứ tự giảm dần) **q**.
- Quá trình xử lý kết thúc khi duyệt xong phần tử ở vị trí **x_n** hoặc khi **q** rỗng và không thể ăn chiếc bánh ở vị trí **x_i** nào đó.

Độ phức tạp của giải thuật: $O(n \log n)$.

Lưu ý: Cần cẩn nhắc khi chọn kiểu dữ liệu cho các biến.

Chương trình:

```
#include <fstream>
#include <ctime>
#include <queue>
#define NAME "cakes."
using namespace std;
//typedef pair<int,int> pi;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,t,ans=0,r=0,a[100001],b[100001],tq;
priority_queue<int> q;
int64_t sumt=0;

int main()
{clock_t aa=clock();
 fi>>n>>t;
 for(int i=0;i<n;++i)fi>>a[i]>>b[i];

 for(int i=0;i<n;++i)
 {
     q.push(b[i]);
     sumt+=b[i];++r;
     while(!q.empty() && sumt+a[i]>t)
     {
         tq=q.top(); sumt-=tq;--r;q.pop();
     }
     if(sumt+a[i]<=t)if(ans<r)ans=r;
 }
 fo<<ans;

 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VR06. BOT

Tên chương trình: BOT.???

BOT (*Built-Operation-Transfer*, có nghĩa: Xây dựng-Vận hành-Chuyển giao) là hình thức Chính phủ kêu gọi các công ty bỏ vốn xây dựng trước (*Built*) thông qua đấu thầu, sau đó khai thác vận hành một thời gian (*Operation*) và sau cùng là chuyển giao (*Transfer*) lại cho nhà nước sở tại.

Đường cao tốc xuyên quốc gia được xây dựng theo hình thức BOT. Công ty Đa quốc gia *Modern Highway* trúng thầu, chia toàn bộ con đường thành n đoạn. Theo tính toán của Công ty sau khi chuyên giao con đường cho chính phủ sở tại quản lý thì lãi thu được ở đoạn đường thứ i là a_i , a_i có thể dương, âm hoặc bằng 0, tức là với từng đoạn con có thể lãi, lỗ hoặc hòa vốn. Từng nhóm các đoạn đường liên tiếp nhau (gọi tắt là khoảng) được chia cho các công ty con thực hiện. Công ty con *ASEAM Highway* hiện đang có trụ sở ở nước sở tại được quyền chọn trước khoảng tùy ý (có thể là cả con đường).

Đĩa nhiên Ban Giám đốc *ASEAM Highway* muốn chọn khoảng bắt đầu từ đoạn p đến hết đoạn q mang lại lợi nhuận cao nhất hoặc lỗ ít nhất nếu không có khoảng nào cho lãi.

Hãy chỉ ra khoảng cần chọn và lãi thu được. Nếu có nhiều cách chọn thì chỉ ra cách chọn có p nhỏ nhất.

Dữ liệu: Vào từ file văn bản BOT.INP:

- ✚ Dòng đầu tiên chứa số nguyên n ($1 \leq n \leq 10^6$),
- ✚ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($0 \leq |a_i| \leq 10^9$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản BOT.OUT trên một dòng 2 số nguyên p, q và lãi thu được.

Ví dụ:

BOT.INP	BOT.OUT
16 2 -4 5 -8 4 -1 -1 1 1 1 -2 2 4 -6 9 -4	5 15 12



Giải thuật:

- ⊕ Nếu $a_i < 0$ với $\forall i = 1 \div n$: kết quả lõi $ans = \max\{a_i, i=1 \div n\} = a_j$, khoảng cần tìm chỉ chứa một phần tử,
- ⊕ Nếu $a_i \leq 0$ với $\forall i = 1 \div n$: kết quả lõi $ans = a_j = 0$, khoảng cần tìm chỉ chứa một phần tử,
- ⊕ Trường hợp tồn tại $a_i > 0$:
 - Tính tổng sum các phần tử liên tiếp ở các vị trí từ u đến v , bắt đầu từ $u = 1$,
 - Nếu $sum > 0$: lưu giá trị tổng lớn nhất tìm được và khoảng chứa giá trị này,
 - Nếu $sum < 0$: bỏ qua khoảng đã duyệt, gán $sum = 0$.

Không cần phải lưu mảng số ban đầu,

Dộ phức tạp của giải thuật: O(n).

Chương trình:

```
#include <fstream>
#include <ctime>
#include <iomanip>
#define NAME "bot."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,t,p,q,it,r=-1000000010,ir,ir0=-1;
int64_t ans,sum;

int main()
{clock_t aa=clock();
 fi>>n; sum=0; ans=-1;
 for(int i=0;i<n;++i)
 {fi>>t;
 if(t==0 && ir0<0)ir0=i;
 if(t<0 && t>r){r=t;ir=i;}
 sum+=t;if(sum<0) {it=i;sum=0;}
 else if(sum>ans){ans=sum;q=i;p=it;}
 }
 if(ans<0)fo<<ir+1<<' '<<ir+1<<' '<<r;
 else if(ans==0)fo<<ir0+1<<' '<<ir0+1<<" 0";
 else fo<<p+2<<' '<<q+1<<' '<<ans;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VR07. KẾT BẠN

Tên chương trình: ONLYONE.???

Nhà trường quyết định xây dựng riêng cho mình một mạng xã hội để các bạn trẻ có điều kiện giao lưu một cách tốt nhất. Hệ thống sẽ tự động chọn và giới thiệu cho mỗi người những người bạn tiềm năng trong trường. Khi đăng ký, người tham gia sẽ phải trải qua thủ tục trắc nghiệm tâm lý. Kết quả trắc nghiệm cho biết giá trị tâm lý theo ba chỉ số, mỗi giá trị là một số nguyên dương.

Thực tế cuộc sống cho thấy, nếu 2 người có giá trị khác nhau ở cả 3 chỉ số thì họ sẽ thường xuyên rơi vào tranh luận, cãi nhau bất tận, còn nếu có giá trị ở 2 hay 3 chỉ số trùng nhau thì mối quan hệ, nếu có – sẽ rất buồn chán. Như vậy, để có một mối quan hệ có lợi và duy trì được lâu dài thì hai người phải có cùng giá trị ở một chỉ số nào đó, còn giá trị ở các chỉ số còn lại phải khác nhau.

Với n nhóm ba (a_i, b_i, c_i) hãy cho biết có bao nhiêu cặp $i < j$ mà số lượng đăng thức $a_i = a_j, b_i = b_j, c_i = c_j$ chỉ có đúng một.

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

- ✚ Dòng đầu tiên chứa số nguyên n ($1 \leq n \leq 10^5$),
- ✚ Dòng thứ i trong n dòng sau chứa 3 số nguyên a_i, b_i và c_i ($1 \leq a_i, b_i, c_i \leq 100$).

Kết quả: Đưa ra file văn bản ONLYONE.OUT một số nguyên – số lượng cặp tìm được.

Ví dụ:

ONLYONE.INP
4
100 100 100
100 100 100
100 99 99
99 99 100

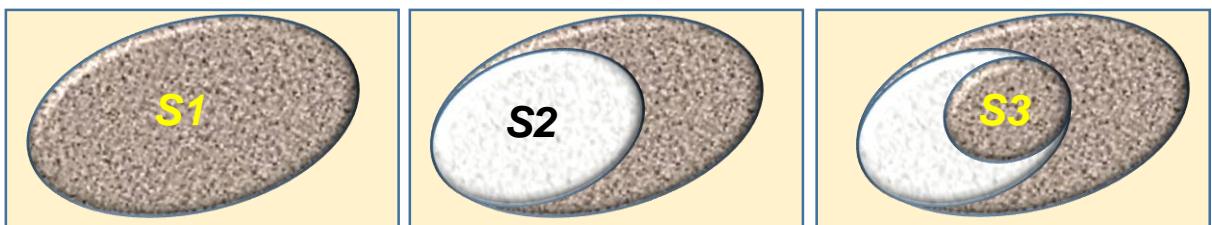
ONLYONE.OUT
5



Giải thuật: Tính lực lượng tập hợp.

Nguyên lý giải:

- ⊕ Tạm thời không xét một số điều kiện, tìm lực lượng của tập S1 rộng hơn, bao tập cần tìm S0 (*xử lý điều kiện cần*), thông thường chỉ giữ lại một điều kiện,
- ⊕ Xét tiếp một kiện mới loại bỏ khỏi S1 các phần tử không thuộc S0, tìm lực lượng tập $S_2 \subset S_1$ (thông thường – S_2 chứa các phần tử thỏa mãn điều kiện đủ),
- ⊕ Nếu S_2 không trùng với S_0 : Bổ sung thêm điều kiện tìm lực lượng tập S_3 – mở rộng S_2 ,
- ⊕ Hai bước cuối cùng được lặp lại nhiều lần cho đến khi nhận được S_0 .



Tổ chức dữ liệu:

- ❖ `int va[101]={0}, vb[101]={0}, vc[101]={0}`: va_i lưu trữ số người có giá trị chỉ số thứ nhất bằng i , tương tự như vậy với vbi và vc_i ,
- ❖ `int vab[101][101]={0}, vac[101][101]={0}, vbc[101][101]={0}`: $vab_{i,j}$ lưu trữ số người có giá trị chỉ số thứ nhất bằng i và giá trị chỉ số thứ 2 bằng j ,
- ❖ `int vabc[101][101][101]={0}`: $vabc_{i,j,k}$ lưu trữ số người có giá trị chỉ số thứ nhất bằng i , giá trị chỉ số thứ 2 bằng j và giá trị chỉ số thứ 3 bằng k .

Xử lý:

- Tính tổng các cặp có giá trị một chỉ số giống nhau, không phụ thuộc vào các chỉ số còn lại, với $t = va_i$, tổng số các cặp có giá trị chỉ số thứ nhất bằng i sẽ là $t \times (t-1)/2$, tương tự như vậy với các chỉ số thứ 2 và thứ 3,
- Mỗi cặp giống nhau ở 2 chỉ số bị tính lặp 2 lần, vì vậy cần trừ 2 lần số lượng các cặp giống nhau ở 2 chỉ số,
- Mỗi cặp giống nhau ở cả 3 chỉ số bị tính lặp 3 lần, nhưng ở bước trên chúng bị trừ 6 lần (2 lần với vab , 2 lần với vac và 2 lần với vbc), vì vậy phải cộng thêm 3 lần số lượng các cặp giống nhau ở cả 3 chỉ số.

Độ phức tạp của giải thuật: O(1).

Chương trình:

```
#include <fstream>
#include <iostream>
#include <cmath>
#include <ctime>
#define NAME "onlyone."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,a,b,c,va[101]={0},vb[101]={0},vc[101]={0},
    vab[101][101]={0},vac[101][101]={0},vbc[101][101]={0},
    vabc[101][101][101]={0};
int64_t ans=0,t;

int main()
{clock_t aa=clock();
 fi>>n;
 for(int i=1;i<=n;++i)
 {
     fi>>a>>b>>c;
     ++va[a]; ++vb[b];++vc[c];
     ++vab[a][b];++vac[a][c];++vbc[b][c];
     ++vabc[a][b][c];
 }

 for(int i=1;i<=100;++i)
 {
     t=va[i]; if(t>1)ans+=t*(t-1)/2;
     t=vb[i]; if(t>1)ans+=t*(t-1)/2;
     t=vc[i]; if(t>1)ans+=t*(t-1)/2;
 }

 for(int i=1;i<=100;++i)
     for(int j=1;j<=100;++j)
     {
         t=vab[i][j]; if(t>1)ans-=t*(t-1);
         t=vac[i][j]; if(t>1)ans-=t*(t-1);
         t=vbc[i][j]; if(t>1)ans-=t*(t-1);
     }

 for(int i=1;i<=100;++i)
     for(int j=1;j<=100;++j)
         for(int k=1;k<=100;++k)
     {
         t=vabc[i][j][k];
         if(t>1)ans+=3*t*(t-1)/2;
     }
 fo<<ans;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VR08. TƯỢNG ĐÀI

Quảng trường trung tâm của thành phố được lát bằng các viên gạch hình chữ nhật kích thước $1 \times k$ và được đặt dọc cạnh song song với trục tọa độ. Một trong số các viên gạch có đỉnh dưới trái ở điểm tọa độ $(0, 0)$. Như vậy điểm dưới trái của các viên gạch sẽ có tọa độ $(i \times k + j, j)$ với mọi i, j .

Người ta quyết định dựng tượng đài tôn vinh người đã đặt nền móng xây dựng thành phố tại quảng trường. Bệ của tượng đài có hình đa giác n đỉnh, các cạnh của đa giác song song với trục tọa độ, đỉnh của đa giác có tọa độ nguyên, mọi đường thẳng song song với trục tọa độ, cắt bệ của tượng đài và có điểm trong thì tập các điểm trong tạo thành một đoạn thẳng.

Để xây bệ của tượng đài người ta phải bóc các viên gạch bị bệ đè lên (một phần hoặc toàn bộ). Để giảm thiểu số gạch cần bóc thành phố cho phép chuyển dịch bệ song song với các trục tọa độ.

Hãy xác định số lượng tối thiểu các viên gạch cần bóc.

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

- ✚ Dòng đầu tiên chứa 2 số nguyên n và k ($1 \leq n, k \leq 10^5$),
- ✚ Dòng thứ i trong n dòng sau chứa 2 số nguyên x_i và y_i – tọa độ đỉnh i của bệ, đỉnh được liệt kê theo chiều ngược kim đồng hồ ($0 \leq x_i, y_i \leq 10^6$).

Kết quả: Đưa ra file văn bản MONUMENT.OUT một số nguyên – số lượng tối thiểu các viên gạch cần bóc.

Ví dụ:

MONUMENT.INP
12 3
2 3
1 3
1 2
3 2
3 1
8 1
8 2
10 2
10 3
8 3
8 4
2 4

MONUMENT.OUT
7



Giải thuật: Tổng tiền tố.

Nhận xét:

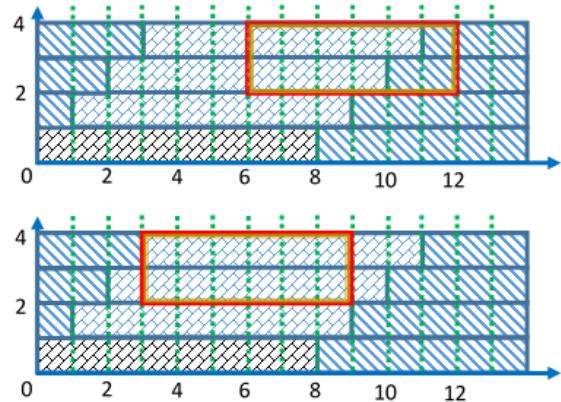
- ⊕ Với mỗi giá trị y ta có thể xác định $\min\{x\}$ và $\max\{x\}$ tương ứng,
- ⊕ Có không quá 10^6 cặp vì vậy các giá trị này có thể lưu được dưới dạng các giá trị biên $\min_{x,y}$ và $\max_{x,y}$,
- ⊕ Từ đó có thể dễ dàng tính được số viên gạch cần bóc (**bricks**) ở vị trí ban đầu,
- ⊕ Số gạch cần bóc thay đổi một cách tương đương khi di chuyển theo trục x hoặc theo trục y , vì vậy chỉ cần xét phép tính tiền theo trục x ,
- ⊕ Sau $k-1$ lần dịch chuyển các giá trị sẽ lặp lại,
- ⊕ Cần phải giảm thiểu số viên gạch bị đè lên một phần,
- ⊕ Ở mỗi dòng tối đa chỉ có 2 viên bị đè một phần,
- ⊕ Với mỗi dòng cần tính phải di chuyển bao nhiêu để biên trái trùng với đầu một viên gạch, số lượng gạch cần bóc sẽ bớt đi 1,
- ⊕ Xác định bắt đầu từ vị trí nào thì biên phải không trùng với đầu viên gạch, số lượng gạch cần bóc sẽ tăng thêm 1,
- ⊕ Ghi nhận các vị trí dẫn đến thay đổi số lượng gạch cần bóc vào mảng **sum** kích thước k ,
- ⊕ Tính tổng tích lũy mảng **sum** ta có sum_i cho biết nếu đẩy hình sang phải i vị trí thì bớt được bao nhiêu viên gạch cần bóc (*ở hình trên nếu trường hợp đẩy tương đương sang trái $k-i$ vị trí*),
- ⊕ Vấn đề còn lại là tính số viên gạch cần bóc ứng với vị trí giảm được nhiều viên gạch bị đè một phần nhất).

Hình minh họa tương ứng với $k = 8$.

Mảng **sum** sẽ có giá trị $(0, 0, 0, 0, -1, -2, -2, -1)$,

Số viên gạch cần bóc ở vị trí ban đầu: **bricks** = 4,

Số viên gạch cần bóc ở vị trí tối ưu: **bricks** = $4 - 2 = 2$.



Độ phức tạp của giải thuật: $O(n)$.

Chương trình:

```
#include <fstream>
#include <algorithm>
#include <ctime>
#define NAME "monument."

using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
const int MAXC = 1001000;
int minx[MAXC];
int maxx[MAXC];
int sum[MAXC + 1],mxy=-1;

void edge(int y1, int y2, int x2) {
    for (int j = min(y1,y2); j < max(y1,y2); j++){
        minx[j] = min(minx[j], x2);
        maxx[j] = max(maxx[j], x2);
    }
    if(y2>mxy)mxy=y2;
}

int main()
{clock_t aa=clock();
    int n, k;
    fi>>n>>k;
    int x1, y1;
    int x0, y0;
    fi>>x1>>y1;
    y0 = y1, x0 = x1;
    for (int i = 0; i < MAXC; i++)
        maxx[i] = -1, minx[i] = MAXC;
    for (int i = 1; i < n; i++)
    {
        int x2, y2;
        fi>>x2>>y2;
        edge(y1, y2, x2);
        y1 = y2, x1 = x2;
    }
    edge(y1, y0, x0);

    int64_t bricks = 0;
    int t,t1,t2,m,lpos,rpos;
    for (int j = 0; j < mxy; j++)
    {
        if (minx[j] > maxx[j]) continue;
        int lpos = minx[j] + k - j % k;
        int rpos = maxx[j] + k - j % k;
        sum[k - lpos % k]--;
        sum[k - (rpos + k - 1) % k]++;
        lpos = (lpos / k);
        rpos = (rpos + k - 1) / k;
        bricks += max(0, rpos - lpos);
    }
    for (int i = 0; i < k; i++)sum[i+1] += sum[i];
    fo << bricks + *min_element(sum, sum + k) << endl;
}
```

```
clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
return 0;
}
```

HÀM Z

Hàm Z của xâu **S** là mảng **Z**, trong đó Z_i là độ dài lớn nhất của tiền tố xâu con **U** các ký tự liên tiếp nhau của **S** bắt đầu từ vị trí i đồng thời cũng là tiền tố của xâu **S**.

Z_0 thường được gán giá trị bằng 0 hoặc bằng độ dài của xâu **S**.

Ví dụ: **S** = '**abcdabscabcdabia**'

$$Z(S)=[16,0,0,0,2,0,0,0,6,0,0,0,0,2,0,0,1].$$

Hàm Z được sử dụng trong nhiều bài toán xử lý xâu, đặc biệt có hiệu quả khi tìm kiếm theo mẫu.

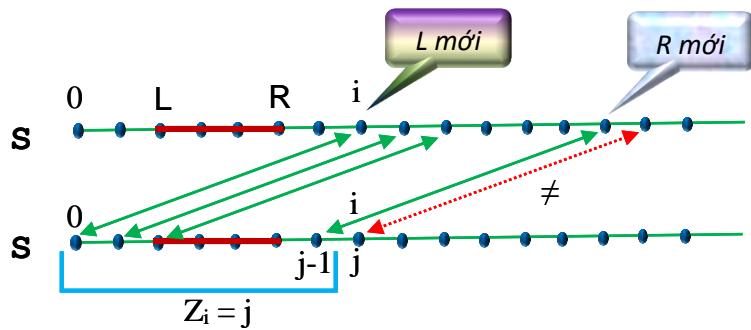
Giải thuật xác định Z:

- ✚ Các ký tự của xâu **S** được đánh số bắt đầu từ 0,
- ✚ Ký hiệu **L** và **R** là điểm đầu và cuối của tiền tố ứng với giá trị **R** lớn nhất đã tính được, ban đầu **L** = **R** = 0,
- ✚ $Z_0 = S.size()$,
- ✚ Giá thiết đã tính được Z_1, Z_2, \dots, Z_{i-1} ,
- ✚ Tính Z_i :
 - ❖ Nếu $i \in [L, R]$:
 - Đặt $j = i - L$,
 - Nếu $i + Z_j \leq R \rightarrow Z_i = Z_j$,
 - Nếu $i + Z_j > R$: Duyệt tiếp các ký tự sau **R**, kiểm tra sự trùng nhau các tiền tố (của **S** và của xâu con bắt đầu từ i), gán độ dài tìm được cho Z_i , điểm cuối mới – cho **R**, gán **L** = **i**.
 - ❖ Nếu $i \notin [L, R]$: So sánh trực tiếp các ký tự của xâu con (bắt đầu từ i) và của xâu **S**, tìm tiền tố dài nhất để xác định Z_i , cập nhật lại **L** và **R** (đầu và cuối tiền tố ở xâu con).

Độ phức tạp của giải thuật: Mỗi ký tự của xâu **S** được xét không quá 2 lần, vì vậy ta có độ phức tạp của giải thuật là $O(n)$, trong đó **n** – độ dài xâu **S**.

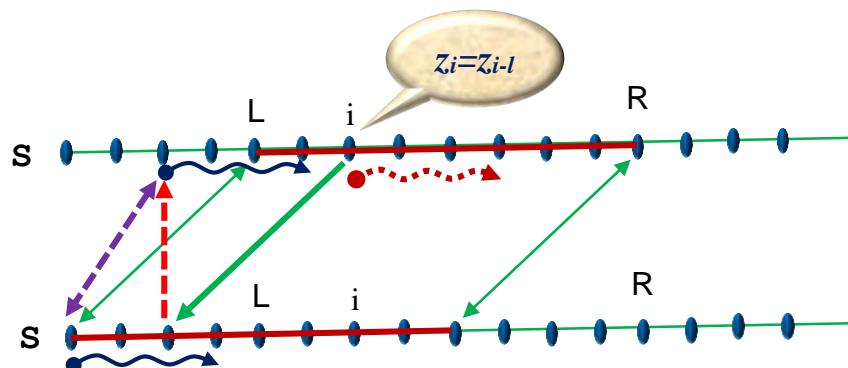
Sơ đồ hoạt động của giải thuật:

Khi i nằm ngoài khoảng $[L, R]$: ta có $i > R$,

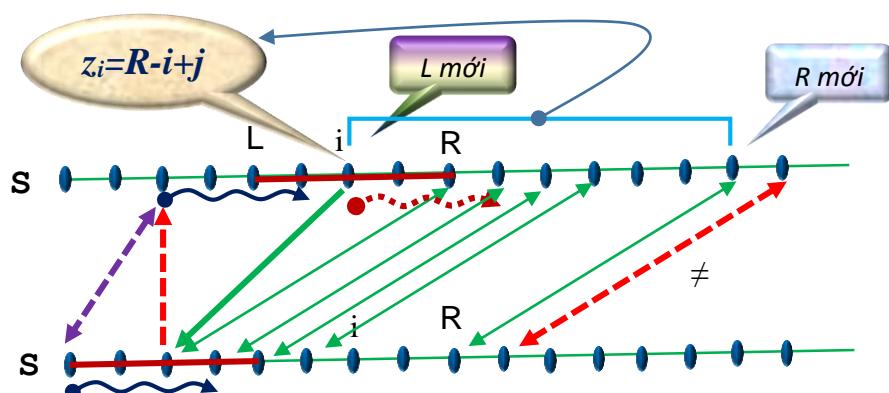


Khi i thuộc khoảng $[L, R]$: Hai trường hợp:

Trường hợp $Z_{i-L} < R-i+1$



Trường hợp $Z_{i-L} \geq R-i+1$



Hàm C++ tính Z:

```
vector<int> calc_z (string s){  
    vector<int> z;  
    int len = s.size();  
    z.resize(len);  
    z[0] = len;  
    int l = 0, r = 0;  
    int j;  
    for (int i = 1; i < len; i++)  
        if (i > r){  
            for (j = 0; ((j + i) < len) && (s[i + j] == s[j]) ; j++)  
                z[i] = j;  
            l = i;  
            r = i + j - 1;  
        }  
        else  
            if (z[i - 1] < r - i + 1)  
                z[i] = z[i - 1];  
            else{  
                for (j = 1; ((j + r) < len) && (s[r + j] == s[r - i + j]);  
j++)  
                    z[i] = r - i + j;  
                l = i;  
                r = r + j - 1;  
            }  
    return z;  
}
```

Tương tự như vậy, ta có thể xây dựng hàm *tính trực tiếp* hậu tố (suffix):

```
void calc_zr(string s)

{int j,l,r;
 zr[n-1]=n;l=n;r=n;
 for(int i=n-2;i>=0;--i)
 {
    if(i<l)
    {   for(j=0;((i-j)>=0)&&(s[i-j]==s[n-j-1]);++j);
        zr[i]=j; r=i; l=i-j+1;
    }
    else
        if(zr[n-r+i-1]<i-1+1)zr[i]=zr[n-r+i-1];
        else
        {
            for(j=0;((l-j)>=0)&&(s[l-j]==s[i-1-j]);++j);
            zr[i]=i-1+j; r=i; l=(j-1);
        }
    }
}
```



Giải thuật 2 tính hàm Z

Sơ đồ trên cho phép tính hàm Z một cách nhanh nhất nhưng *độ phức tạp chương trình* hơi cao. Tồn tại sơ đồ tính toán có *độ phức tạp của giải thuật* tăng thêm đôi chút nhưng độ phức tạp của chương trình giảm đáng kể.

Ta nhận thấy ở vị trí i bất kỳ việc tính xâu tiền tố có thể xuất phát từ độ dài xâu tiền tố đã tính ở vị trí $i-L$. Ban đầu ta có z_i không vượt quá z_{i-L} và phần còn lại $R-i+1$ của tiền tố đã tính.

Để xác định giá trị thực của z_i ta chỉ cần kiểm tra xem có thể tiếp tục kéo dài tiền tố hay không bằng cách so sánh trực tiếp các ký tự tiếp theo.

Hàm sau cho phép tính Z bắt đầu từ một vị trí x của xâu s và ghi kết quả vào mảng z (để thuận tiện cho các xử lý tiếp theo nên gán giá trị đầu $z_x = 0$).

```
void calc_z(string s, int x, int * z)
{
    int l, r;
    z[x] = 0;
    for (int i = x+1, l = 0, r = 0; i < n; ++i) {
        z[i] = min(z[i - 1], max(0, r - i + 1));
        while (s[z[i]] == s[i + z[i]])
            z[i]++, l = i, r = i + z[i] - 1;
    }
}
```

VR16. THUẦN CHỦNG

Tên chương trình: PURE.???

Gene là một đoạn kết nối các cặp AND, mỗi cặp AND được đặc trưng bằng một chữ cái trong tập $\{A, C, G, T\}$. Gene thuần chủng là gene hình thành từ một đoạn AND cơ sở độ dài không quá m , được kết nối lặp đi lặp lại nhiều lần và ở lần lặp cuối cùng có thể chỉ chứa phần đầu của đoạn cơ sở. Gene được mô tả dưới dạng xâu S chỉ chứa các ký tự trong tập nêu trên. Như vậy gene thuần chủng là xâu có thể biểu diễn như tổng của k đoạn cơ sở ($k \geq 0$) và có thể có thêm một đoạn đầu của cơ sở.

Ví dụ, với $m = 10$, $S = "ACATAGACATAGACATAGACA"$ là một gene thuần chủng vì có đoạn cơ sở là "**ACATAG**" và $S = "ACATAG" + "ACATAG" + "ACATAG" + "ACA"$, nhưng với $m = 5$ thì S không phải là gene thuần chủng.

Cho gene S độ dài n và giá trị m . Hãy xác định S có phải là gene thuần chủng hay không và đưa ra *đoạn cơ sở ngắn nhất* nếu S là gene thuần chủng hoặc đưa ra thông báo "**NO**" trong trường hợp ngược lại.

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

- ✚ Dòng đầu tiên chứa số nguyên m ($1 \leq m \leq 10^6$),
- ✚ Dòng thứ 2 chứa xâu S độ dài n chỉ chứa các ký tự trong tập đã nêu.

Kết quả: Đưa ra file văn bản PURE.OUT đoạn cơ sở ngắn nhất tìm được hoặc thông báo **NO**.

Ví dụ:

PURE.INP	PURE.OUT
10	
ACATAGACATAGACATAGACA	ACATAG



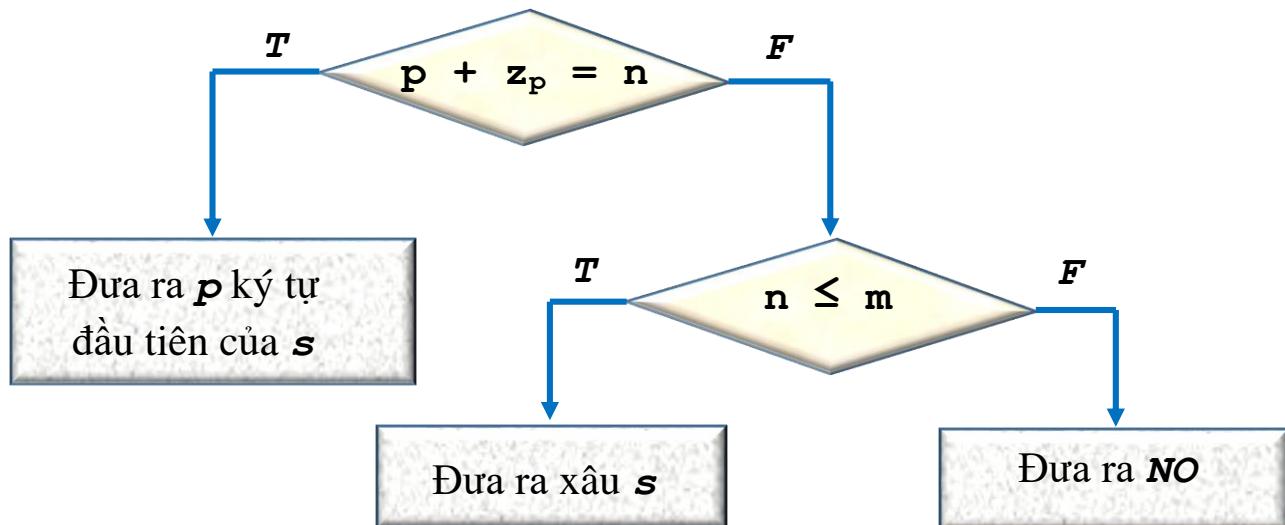
Giải thuật: Ứng dụng hàm Z.

Tổ chức dữ liệu:

Mảng int z[1000010] lưu trữ độ dài tiền tố, z_i – độ dài lớn nhất của xâu ký tự bắt đầu từ i là tiền tố của xâu s .

Các bước xử lý:

- + Nhập dữ liệu,
- + Tính z ,
- + Tính $k = \min\{n, m\}$,
- + Xác định $zx = \max\{z_i, i = 1 \div k\} = z_p$,
- + Kiểm tra:



Độ phức tạp của giải thuật: $O(n)$.

Chương trình:

```
#include <fstream>
#include <string>
#include <ctime>
#include <iomanip>
#define NAME "pure."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
string s,sp;
int n,m,z[1000010];

void calc_z()
{int j,l=0,r=0;
 z[0]=n;
 for(int i=1;i<n;++i)
 {
     if(i>r)
     {
         for(j=0;((j+i<n)&&(s[i+j]==s[j]));++j);
         z[i]=j; l=i; r=i+j-1;
     }
     else
     {
         if(z[i-1]<r-i+1)z[i]=z[i-1];
         else
         {
             for(j=1;((j+r<n)&&(s[r+j]==s[r-i+j]));++j);
             z[i]=r-i+j; l=i; r+=(j-1);
         }
     }
 }
}

int main()
{clock_t aa=clock();
 fi>>m;
 fi>>s; n=s.size();
 calc_z();
 int k,p,zx;
 k=min(n,m);
 zx=0;
 for(int i=1;i<=k;++i)if(zx<z[i])zx=z[i],p=i;
 if(p+z[p]==n) for(int i=0;i<p;++i)fo<<s[i];
 else if(n<=m)fo<<s; else fo<<"NO";
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

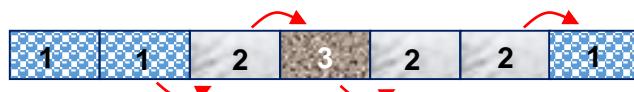
VR19. SỬA ĐƯỜNG

Tên chương trình: REPAIRS.???

Đường đại lộ xuyên qua khu đô thị mới với hàng phượng tím hai bên cực đẹp, rất nổi tiếng nhưng bởi một lý do đáng buồn: chất lượng quá kém. Lý do là thành phố chia con đường thành n đoạn bằng nhau đánh số từ 1 đến n tính từ đầu con đường, mỗi đoạn giao cho một công ty duy tu bảo dưỡng. Mỗi công ty, khi bảo dưỡng, bóc lớp nhựa cũ và thay nó bằng lớp nhựa trải đường mà mình đang có. Kết quả là đường có nhiều đoạn chất lượng khác nhau ảnh hưởng đến độ an toàn của các phương tiện tham gia lưu thông.



Người dân trong khu đô thị đã cùng nhau xây dựng một trang WEB thông báo về tình trạng của đường, trong đó có nêu *Chỉ số mất an toàn*. Chỉ số này được xác định như sau: Nếu toàn bộ con đường được lát cùng một loại nhựa trải đường thì chỉ số là 1, trong trường hợp ngược lại, khi đi từ đầu đến cuối đường cứ mỗi lần phải chuyển sang đoạn có lớp phủ khác loại, chỉ số tăng thêm 1. Mỗi loại lớp phủ được đặc trưng bằng một số nguyên c theo danh mục quản lý chất lượng quốc gia. Ví dụ, $n = 7$ và các đoạn có loại lớp phủ là (1, 1, 2, 3, 2, 2, 1) thì Chỉ số mất an toàn sẽ là 5.



Ban đầu, lớp phủ trên các đoạn là t_1, t_2, \dots, t_n . Trong quá trình trang WEB hoạt động có q lần đường bị sửa, lần thứ j đoạn p_j được phủ lại bằng nhựa đƣợng loại c_j , $j = 1 \div q$.

Hãy xác định Chỉ số mất an toàn sau mỗi lần đường được sửa.

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

- ✚ Dòng đầu tiên chứa số nguyên n ($1 \leq n \leq 10^5$),
- ✚ Dòng thứ 2 chứa n số nguyên t_1, t_2, \dots, t_n ($|t_i| \leq 10^9$, $i = 1 \div n$),
- ✚ Dòng thứ 3 chứa số nguyên q ($1 \leq q \leq 10^5$),
- ✚ Dòng thứ j trong q dòng tiếp theo chứa 2 số nguyên p_j và c_j ($1 \leq p_j \leq n$, $|c_j| \leq 10^9$).

Kết quả: Đưa ra file văn bản REPAIRS.OUT q số nguyên – các chỉ số tính được, mỗi số trên một dòng.

Ví dụ:

REPAIRS.INP	REPAIRS.OUT
7	5
1 1 2 3 2 2 1	3
3	4
2 2	
4 2	
6 9	



Vr19 Mosk20150307 D Yellow

Giải thuật: Thống kê đơn giản.

Các bước xử lý:

- ✚ Nhập n và mảng t ban đầu,
- ✚ Chuẩn bị $ans = 1$,
- ✚ Với $i = 2 \div n$: nếu $t_i \neq t_{i-1} \rightarrow ++ans$,
- ✚ Nhập q và xử lý các truy vấn:
 - ❖ Nhập p, c ,
 - ❖ Khi $c \neq t_p$:
 - ❖ Phân biệt 3 trường hợp:
 - $p = 1$: giảm ans nếu $t_1 \neq t_2$, tăng ans nếu $c \neq t_2$,
 - $p = n$: giảm ans nếu $t_n \neq t_{n-1}$, tăng ans nếu $c \neq t_{n-1}$,
 - $1 < p < n$:
 - So sánh t_p với t_{p-1} và với t_{p+1} , chỉnh lý ans ,
 - So sánh c với t_{p-1} và với t_{p+1} , chỉnh lý ans ,
 - ❖ Gán $t_p=c$;
 - ❖ Đưa ra ans .

Độ phức tạp của giải thuật: $O(n)$.

Chương trình:

```
#include <fstream>
#include <ctime>
#define NAME "repairs."

using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,q,p,c,t[100001],ans=0;

int main()
{clock_t aa=clock();
 fi>>n>>t[1]; ans=1;
 for(int i=2;i<=n;++i){fi>>t[i]; ans+=(t[i]!=t[i-1]);}
 fi>>q;
 for(int i=1;i<=q;++i)
 {
    fi>>p>>c;
    if(c!=t[p])
    {   if(p==1)ans=ans-(t[1]!=t[2])+(c!=t[2]);
        else if(p==n)ans=ans-(t[n]!=t[n-1])+(c!=t[n-1]);
        else
            ans=ans-(t[p]!=t[p+1])-(t[p]!=t[p-1])+(c!=t[p-1])+(c!=t[p+1]);
        t[p]=c;
    }
    fo<<ans<<'\'n';
 }
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VR14. TỪ ĐIỂN

Tên chương trình: DICTIONARY.???

Các hệ thống soạn thảo có nhiều tính năng nhận dạng và xử lý tự động theo ngữ cảnh. Ví dụ WINWORD tự động chuyển chữ cái đầu từ sang thành chữ hoa ở từ đầu tiên sau dấu chấm câu, chương trình soạn thảo trong CODE BLOCKS tự động thêm dấu đóng ngoặc kiểu tương ứng khi người dùng gõ dấu mở ngoặc, . . .

Steve tham gia vào nhóm xây dựng hệ thống soạn thảo mới cho phép người dùng không cần phải gõ dấu cách, hệ thống sẽ nhận dạng từ và tự động chèn dấu cách vào vị trí cần thiết. Với mỗi loại ngôn ngữ hệ thống cần có một từ điển phục vụ việc nhận dạng từ. Từ điển được xây dựng tự động và mở rộng dần trong quá trình khai thác để đảm bảo thích nghi và hiệu quả tối đa với từng người sử dụng riêng biệt!

Để kiểm tra khả năng nhận dạng Steve gõ vào một xâu **s** chỉ chứa các ký tự trong tập {**a**, **b**, **c**}. Hệ thống phải số lượng từ khác nhau tối thiểu xuất hiện trong xâu, mỗi từ là một dãy các ký tự liên tục độ dài không quá **L**.

Hãy đưa ra số lượng từ tối thiểu tìm được và các từ đó. Nếu có nhiều cách xác định từ thì đưa ra kết quả theo cách tùy chọn.

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

- ✚ Dòng đầu tiên chứa số nguyên **L** ($1 \leq L \leq 10^4$),
- ✚ Dòng thứ 2 chứa xâu **s** độ dài không quá 2×10^4 .

Kết quả: Đưa ra file văn bản DICTIONARY.OUT, dòng đầu tiên chứa số nguyên **m** xác định số lượng từ khác nhau tối thiểu tìm được, mỗi dòng trong **m** dòng sau chứa một từ trong số các từ khác nhau.

Ví dụ:

DICTIONARY.INP
3
ababaabab

DICTIONARY.OUT
2
aba
ab



Vr14 Mosk20150307 A Blue

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <iostream>
#include <ctime>
using namespace std;
const int N = (int) 2e5 + 10;
char s[N];
int z_a[N], z_b[N], bad_b[N], it, l, n;
bool bad_a[N];
```

```

void calc_z(char * s, int * z) {
    z[0] = 0;
    for (int i = 1, l = 0, r = 0; s[i]; ++i) {
        z[i] = min(z[i - 1], max(0, r - i + 1));
        while (s[z[i]] == s[i + z[i]])
            z[i]++, l = i, r = i + z[i] - 1;
    }
}

```

```

void print(char *s, int len) {
    for (int i = 0; i < len; ++i)
        printf("%c", s[i]);
    printf("\n");
}

```

```

int main() {clock_t aa=clock();
freopen("60","r",stdin);
freopen("dictionary.out","w",stdout);
scanf("%d%s", &l, s);
n = strlen(s);
calc_z(s, z_a);
z_a[0] = l;
for (int len_a = 1; len_a <= l; ++len_a) {
    if (bad_a[len_a]) continue;
    int start = len_a;
    while (z_a[start] >= len_a)
        start += len_a, bad_a[start] = true;
    if (start == n) {
        printf("1\n");
        print(s, len_a);
        clock_t bb=clock();
cout<<"\nTime: "<<(double)(bb-aa)/1000<<"sec";
        return 0;
    }
}
for (int len_a = 1; len_a <= l; ++len_a) {
    if (bad_a[len_a]) continue;
    int start = 0;
    while (z_a[start] >= len_a)
        start += len_a;
    calc_z(s + start, z_b + start);
    z_b[start] = 1, ++it;
}

```

```

for (int len_b = 1; len_b <= l; ++len_b) {
    if (bad_b[start + len_b] == it) continue;
    int pos = start;
    while (z_b[pos] >= len_b)
        pos += len_b, bad_b[pos] = it;
    while (true) {
        if (z_a[pos] >= len_a)
            pos += len_a;
        else if (z_b[pos] >= len_b)
            pos += len_b;
        else
            break;
    }
    if (pos == n) {
        printf("2\n");
        print(s, len_a);
        print(s + start, len_b);
        clock_t bb=clock();
        cout<<"\nTime: "<<(double)(bb-aa)/1000<<"sec";
        return 0;
    }
}
printf("3\na\nb\nc\n");
clock_t bb=clock();
cout<<"\nTime: "<<(double)(bb-aa)/1000<<"sec";
return 0;
}

```

VR20. PHÂN RÃ

Tên chương trình: DECOMP.???

Một số nguyên dương n ngoài dạng 1 nhân với chính nó còn có thể biểu diễn dưới dạng tích của hai hay nhiều số nguyên dương khác nhau, trong đó các thừa số khác 1 và n . Ví dụ với $n = 12$ ta có $n = 2 \times 6 = 3 \times 4 = 2 \times 2 \times 3$.

Việc phân tích số nguyên n thành tích các số nguyên được gọi là phân rã số nguyên (*decomposition*). Mỗi thừa số tham gia vào phép phân rã được gọi là thành phần phân rã hay gọi tắt là thành phần. Trong nhiều trường hợp người ta không cần xét tất cả mọi cách phân rã mà chỉ xét những phân rã có đúng k thành phần, các thành phần đều khác 1 và n . Ví dụ, với $n = 12$ và $k = 2$ ta có 2 cách phân rã: $n = 2 \times 6$ và $n = 3 \times 4$. Các thành phần khác nhau tham gia vào việc phân rã 12 thành hai thành phần là 2, 3, 4 và 6.

Yêu cầu: Cho n và k . Hãy xác định số lượng các thành phần khác nhau tham gia vào việc phân rã n thành k thành phần và chỉ ra các thành phần đó.

Dữ liệu: Vào từ file văn bản DECOMP.INP gồm một dòng chứa 2 số nguyên n và k ($1 < n \leq 10^{15}$, $1 < k \leq 20$).

Kết quả: Đưa ra file văn bản DECOMP.OUT dòng đầu tiên chứa số nguyên m – số các thành phần khác nhau, dòng thứ 2 chứa m số nguyên – các thành phần khác nhau tìm được. Nếu không có cách phân rã theo yêu cầu thì đưa ra một dòng chứa số -1.

Ví dụ:

DECOMP.INP
12 2

DECOMP.OUT
4 2 3 4 6



Giải thuật: Tính tổ hợp.

Nhận xét:

- ⊕ Phân rã n thành tích của các thừa số nguyên tố, gọi m là số lượng thừa số nguyên tố nhận được, có $n = \mathbf{x}_1 \times \mathbf{x}_2 \times \dots \times \mathbf{x}_m$,
- ⊕ Nếu $m < k$: bài toán vô nghiệm, đưa ra -1 và kết thúc xử lý,
- ⊕ Trường hợp $m \geq k$: các thừa số tham gia và phân rã n thành tích của k số nguyên dương khác 1 và n chứa ít nhất một giá trị \mathbf{x}_i và nhiều nhất là $m-k+1$ thừa số nguyên tố, vì vậy cần tìm tổ hợp chap 1, chap 2, . . . , chap $m-k+1$ các thừa số nguyên tố tìm được, lấy tích các số tham gia vào tổ hợp và lọc bỏ các số lặp lại.

Các bước xử lý:

- ⊕ Phân rã n thành tích của các thừa số nguyên tố,
- ⊕ Nếu $m < k$: bài toán vô nghiệm, đưa ra -1 và kết thúc xử lý,
- ⊕ Tính tích các tổ hợp chap 1, chap 2, . . . , chap $m-k+1$,
- ⊕ Loại bỏ các giá trị lặp,
- ⊕ Đưa ra kết quả nhận được.

Tổ chức dữ liệu:

- ⊕ Mảng `int64_t x[64]` – lưu các thừa số nguyên tố,
- ⊕ `vector<int64_t> b` hoặc mảng tĩnh `int64_t b[1000000]` – lưu các thừa số cần đưa ra.

Lưu ý trong tổ chức xử lý:

- ⊕ Để xác định các thừa số nguyên tố có thể áp dụng giải thuật phân rã Pollard. Tuy vậy, với $n \leq 10^{15}$ có thể sử dụng sàng Eratosthenes,
- ⊕ Khối lượng thông tin trung gian cần lưu trữ sẽ giảm một cách đáng kể nếu thông tin được lọc sơ bộ ngay trong quá trình tính tổ hợp,
- ⊕ Việc lọc dữ liệu có thể được thực hiện ẩn ngay trong quá trình lưu trữ nếu sử dụng tập hợp: thay `vector<int64_t> b` bằng `set<int64_t> b`, thời gian xử lý tăng thêm đôi chút, nhưng không đáng kể.

Độ phức tạp của giải thuật: không vượt quá $O(n^{1/2})$.

Chuong trinh:

```
#include <fstream>
#include <algorithm>
#include <ctime>
#define TASK "decomp."
using namespace std;
typedef int64_t ll;
ifstream fi (TASK"inp");
ofstream fo (TASK"out");
const int mxn=30001,mxe=10000001;
ll n,ans,x[64],b[1000000];
int k,m,p;

void factorize()
{int64_t i;
 i=2;m=0;
 while(i*i<=n)
 {
     while(n%i==0){x[++m]=i;n/=i;}
     ++i;if(n==1) return;
 }
 if(n>1)x[++m]=n;
}

void init_base()
{int km; int64_t t,tc,tb;
 p=0;x[0]=0;
 for(int i=1;i<=m;++i) if(x[i]!=x[i-1])b[++p]=x[i];
 for(int ik=2;ik<=m-k+1;++ik)
     for(int ib=1;ib<=m;++ib)
     {
         tc=0;t=1; km=min(ib+ik-1,m);
         for(int i=ib;i<km;++i)t*=x[i];
         if(t!=tc && ib+ik-1<=m)
         {
             for(int
i=km;i<=m;++i){tb=t*x[i];if(tb!=b[p])b[++p]=tb;}
             tc=t;
         }
     }
 }

void filter()
{int tp=0;
 sort(b+1,b+p+1);
 for(int i=1;i<=p;++i)if(b[i]!=b[i-1])b[++tp]=b[i];
 p=tp;
}

int main()
{clock_t aa=clock();
 fi>>n>>k;
 factorize();
 if(m<k)p=-1; else
 {
     init_base();
     filter();
 }
```

```
}

fo<<p<<'\n';
if(p>0)for(int i=1;i<=p;++i)fo<<b[i]<<' ';
clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VR23. CẶP TỐT

Tên chương trình: GOODPAIRS.CPP

Trong tủ đồ chơi của một lớp học mẫu giáo có bộ đồ sáp chữ gồm các tấm bìa, trên mỗi tấm có một ký tự la tinh trong số n ký tự đầu tiên của bảng chữ cái, có c_i tấm bìa ghi ký tự thứ i , $i = 1 \dots n$.

Để học sinh làm quen với bảng chữ cái cô giáo sắp các tấm bìa thành một dãy dài hàng ngang và yêu cầu đếm số cặp bìa đứng cạnh nhau tính từ trái sang phải ký tự ghi ở tấm bìa bên phải là ký tự tiếp theo trong bảng chữ cái của ký tự ghi trên tấm bìa bên trái. Ví dụ, nếu dãy bìa tạo thành xâu “**abdc**” thì số lượng cặp là 1, còn nếu xâu là “**abcdefghijklmнопqrstuvwxyz**” thì số lượng cặp là 25.

Hãy xác định số lượng cặp tốt tối đa có thể nhận được khi sắp tất cả các tấm bìa thành một dãy.

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

- ✚ Dòng đầu tiên chứa một số nguyên n ($1 \leq n \leq 26$),
- ✚ Dòng thứ i trong n dòng sau chứa số nguyên c_i ($1 \leq c_i \leq 10^9$).

Kết quả: Đưa ra file văn bản GOODPAIRS.OUT một số nguyên – số cặp tốt tối đa có thể nhận được.

Ví dụ:

GOODPAIRS.INP	GOODPAIRS.OUT
2	
3	
4	3



Vr23 Mosk2015 6-9 B

Giải thuật:

Nhận xét:

- ⊕ Một tấm bìa, trong trường hợp tốt nhất, tham gia vào hai cặp tốt: với tấm bìa trước và với tấm bìa sau nó, trạng thái này của tấm bìa được gọi là **tối ưu**,
- ⊕ Kết quả tối ưu đạt được khi có nhiều nhất các tấm bìa ở trạng thái tối ưu.

Các bước xử lý:

B1. Nhập dữ liệu,

B2. Xác định đoạn $[p, q]$ – khoảng liên tục của bảng chữ cái trong đó vẫn còn bìa ghi ký tự thuộc đoạn này, $c_{p-1} = c_{q+1} = 0$, $c_i \neq 0$, $i = p \div q$, nếu $p = 0$ – tới bước B3,

B3. Xử lý đoạn:

- ⊕ Tìm $t = \min\{c_i\}$, $i = p \div q$,
- ⊕ Cập nhật kết quả: bổ sung thêm số lượng cặp tốt $t \times (q-p)$ tìm được,
- ⊕ Cập nhật số lượng bìa còn lại: $c_i = c_i - t$, $i = p \div q$,
- ⊕ Chuyển tới bước B2,

B3. Đưa ra kết quả.

Nhận xét:

- Số lượng chữ cái khác nhau ít (không quá 26) vì vậy không cần tối ưu hóa quá trình tìm đoạn,
- Kết quả có thể vượt quá 10^9 , cần chú ý khi cập nhật kết quả.

Chương trình:

```
#include <fstream>
#include <ctime>
using namespace std;
#define NAME "goodpairs."
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,p,q,c[28]={0};
int64_t res=0;

void find_intv()
{
    p=0;
    for(int i=1;i<=n;++i) if(c[i]>0) {p=i;break;}
    if(p>0) for(int i=p;i<=n+1;++i) if(c[i]==0) {q=i-1;break;}
}

void upd_data()
{int t;
    t=1000000010;
    for(int i=p;i<=q;++i) if(t>c[i]) t=c[i];
    res+=(int64_t)(q-p)*t;
    for(int i=p;i<=q;++i)c[i]-=t;
}
int main()
{
    clock_t aa=clock();
    fi>>n;
    for(int i=1;i<=n;++i) fi>>c[i];
    p=1;
    while(p>0)
    {
        find_intv();
        if(p>0) upd_data();
    }
    fo<<res;
    clock_t bb=clock();
    fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VR24. ĐOÁN TỪ

Tên chương trình: GUESS.CPP

Đoán từ là trò chơi là một trò chơi phổ biến trên TV. Có n đội chơi đánh số từ 1 đến n , mỗi đội 2 người và có một số từ, mỗi từ chỉ chứa các ký tự la tinh thường, có không quá 10 ký tự và được viết trên một tờ giấy. Các từ được bỏ vào mũ. Khi đến lượt chơi, một người trong đội sẽ bốc từ mũ một tờ giấy và giải thích sao cho người thứ 2 trong đội hiểu và nói ra đúng từ viết trên giấy. Nếu từ được đoán đúng thì đội đó ghi thêm được 1 điểm, tờ giấy ghi từ này sẽ bị vứt bỏ ra ngoài. Nếu còn thời gian đội này có thể bốc và đoán từ tiếp theo. Nếu hết thời gian mà từ vẫn chưa được đoán đúng thì tờ giấy ghi từ này được bỏ trở lại vào trong mũ và lượt chơi được chuyển sang một đội nào đó (cũng có thể là chính đội này). Trò chơi kết thúc khi mọi từ đều được đoán ra.

Người dẫn trò chơi ghi nhận biên bản quá trình chơi trên máy tính. Trò chơi kết thúc sau m lần đoán từ. Nhưng không may, máy tính bị nhiễm vi rút và trong biên bản chỉ còn 2 cột ghi đội chơi và từ mà đội đó phải đoán. Trình tự các bản ghi vẫn được giữ nguyên.

Tù phần còn lại này của biên bản hãy xác định số điểm mỗi đội ghi được.

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

- ✚ Dòng đầu tiên chứa hai số nguyên n và m ($1 \leq n, m \leq 10^5$),
- ✚ Mỗi dòng trong m dòng tiếp theo chứa số nguyên xác định đội chơi và từ mà đội đó phải đoán.

Kết quả: Đưa ra file văn bản GUESS.OUT n số nguyên, số thứ i là điểm mà đội i ghi được.

Ví dụ:

GUESS.INP	GUESS.OUT
2 3 1 hat 1 shirt 2 hat	1 1



Giải thuật 1:

Nhận xét:

- ✚ Việc so sánh các từ phức tạp hơn so sánh số nguyên vì vậy nên lưu trữ giá trị hàm băm tương ứng với từ đó,
- ✚ Nếu ở một bước nào đó đội chơi giành được điểm thì từ tương ứng sẽ không xuất hiện ở các bước tiếp theo còn lại.

Tổ chức dữ liệu: số lượng đơn vị dữ liệu cần xử lý không lớn ($\leq 10^5$), vì vậy có thể khai báo cấp phát tĩnh cho các mảng cần dùng để giảm thời gian thực hiện.

- ❖ Mảng ***int a[100001]***: a_i lưu đội chơi ở lượt i ,
- ❖ Mảng ***int d[100001]***: d_i lưu điểm đội i đạt được,
- ❖ Mảng ***uint64_t b[100001]***: b_i lưu giá trị hàm băm của từ ở lần đoán thứ i ,
- ❖ Tập ***set<uint64_t> s*** lưu giá trị hàm băm của các từ đã đoán được.

Xử lý:

- Nhập dữ liệu và thay thế xâu bằng giá trị hàm băm tương ứng,
- Duyệt dữ liệu từ lần đoán cuối cùng lùi về đến lần đoán đầu tiên,
- Với mỗi dữ liệu b_i : nạp giá trị hàm băm vào s và kiểm tra sự kiện trùng dữ liệu, nếu trong tập s chưa có giá trị tương ứng thì thông tin sẽ được nạp vào tập và dấu hiệu nạp $s.insert(b[i]).second$ sẽ có giá trị ***true***, nếu giá trị được nạp vào tập – tăng số điểm của đội a_i .
- Sau khi duyệt hết các lần đoán – đưa ra điểm của các đội.

Độ phức tạp của giải thuật: $O(m \log m)$ do sử dụng tập hợp.

Chương trình I:

```
#include <fstream>
#include <ctime>
#include <set>
#include <string>
#define NAME "guess."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
set<int64_t> s;
const uint64_t h=307;
int n,m,d[100001]={0},t,l,a[100001];
uint64_t p,b[100001];
string st;

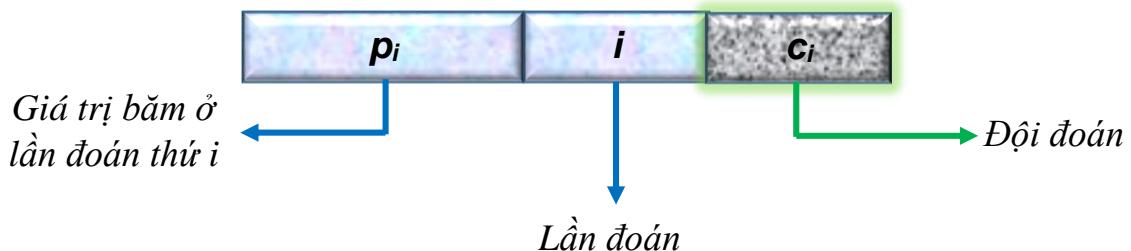
void hash_w()
{
    l=st.size();p=0;
    for(int i=0;i<l;++i)p=p*h+st[i]-96;
}

int main()
{clock_t aa=clock();
 fi>>n>>m;
 for(int i=1; i<=m; ++i)
 {
     fi>>t>>st;
     hash_w();
     a[i]=t; b[i]=p;
 }
 for(int i=m;i>0;--i)if(s.insert(b[i]).second)++d[a[i]];
 for(int i=1;i<= n; ++i)fo<<d[i]<<' ';
 clock_t bb=clock();
 fo<<"\nTime: "<<(double) (bb-aa)/1000<<" sec";
}
```

Giải thuật 2: Không sử dụng tập hợp.

Sử dụng mảng dữ liệu \mathbf{a} , các phần tử của mảng có 3 trường:

(giá trị băm, số thứ tự của lần đoán, đội đoán)



Xử lý:

- + Sắp xếp mảng \mathbf{a} theo thứ tự tăng dần,
- + Gán giá trị hàng rào: ở phần tử thứ $m+1$: cho $p_{m+1} = 0$,
- + Duyệt với mọi $i = 1 \div m$, kiểm tra nếu $p_i \neq p_{i+1}$ thì tăng số điểm của đội c_i .

Độ phức tạp của giải thuật: $O(m \log m)$.

Thời gian thực hiện: Tương tự như trường hợp dùng tập hợp.

Nhận xét: Có thể dễ dàng cài đặt trên các hệ thống lập trình không có thư viện hỗ trợ lưu trữ thông tin dưới dạng cây nhị phân cân bằng.

Chương trình II:

```
#include <fstream>
#include <ctime>
#include <algorithm>
#include <string>
#define NAME "guess."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
typedef pair<uint64_t,int> pii;
const uint64_t h=307;
int n,m,d[100001]={0},t,l;
uint64_t p
string st;
pair<pii,int> a[100002];
void hash_w()
{
    l=st.size();p=0;
    for(int i=0;i<l;++i)p=p*h+st[i]-96;
}

int main()
{clock_t aa=clock();
    fi>>n>>m;
    for(int i=1; i<=m; ++i)
    {
        fi>>t>>st;
        hash_w();
        a[i].first.first=p;a[i].first.second=i; a[i].second=t;
    }
    sort(a+1,a+m+1);a[m+1].first.first=0;
    for(int i=1;i<=m;++i)
        if(a[i].first.first!=a[i+1].first.first)++d[a[i].second];
    for(int i=1;i<= n;++i)fo<<d[i]<<' ';
    clock_t bb=clock();
    fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

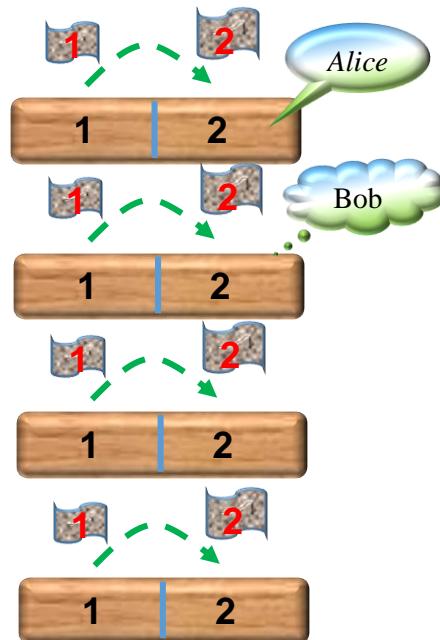
VR21. BÀI KIỂM TRA

Tên chương trình: TASKS.CPP

Alice và Bob là đôi bạn thân trong lớp và thường giúp nhau trong giờ học. Hôm nay sẽ có tiết kiểm tra toán và như mọi khi – sẽ có k đề khác nhau. Lớp học có n học sinh. Các bàn trong lớp xếp thành một hàng dọc, mỗi bàn 2 người ngồi. Vị trí bên trái của bàn đánh số là 1, vị trí bên phải – đánh số 2. Đề bài được phát bắt đầu từ vị trí 1 của bàn 1 rồi đến vị trí 2 của bàn 1, tiếp theo là vị trí 1 của bàn 2, rồi tới vị trí 2 của bàn 2, . . . lần lượt các đề 1, đề 2, đề 3, . . . cho tới đề k rồi lại quay lại đề 1, đề 2, . . . cho tới khi tất cả các học sinh đều nhận được đề.

Alice đến trước và ngồi vào vị trí quen thuộc của mình: hàng p và vị trí q . Bob đến sau và muốn ngồi vào nơi sẽ nhận được cùng đề với Alice và ở dãy bàn gần với dãy bàn của Alice nhất. Nếu có 2 cách ngồi thì Bob sẽ chọn bàn ở phía trước của Alice.

Hãy xác định bàn u và vị trí v trong bàn nơi Bob cần phải ngồi. Nếu không có cách ngồi thỏa mãn thì đưa ra một số -1.



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

- ✚ Dòng đầu tiên chứa một số nguyên n ($2 \leq n \leq 10^9$),
- ✚ Dòng thứ 2 chứa số nguyên k ($2 \leq k \leq n$),
- ✚ Dòng thứ 3 chứa số nguyên p ($1 \leq p \leq (n+1)/2$),
- ✚ Dòng thứ 4 chứa số nguyên q ($1 \leq q \leq 2$).

Kết quả: Đưa ra file văn bản TASKS.OUT trên một dòng 2 số nguyên u và v tìm được hoặc số -1 nếu không có cách ngồi thỏa mãn điều kiện đã nêu.

Ví dụ:

TASKS.INP	TASKS.OUT
25	
2	
1	
2	



Vr21 Mosk2015 6-9 A

Giải thuật: Địa chỉ tuyệt đối và địa chỉ vật lý.

Địa chỉ được cho ở dạng vật lý: $(p, q) - (bàn, vị trí trong bàn)$,

Để tiện xử lý: dùng địa chỉ tuyệt đối (*địa chỉ tuyến tính*), các vị trí đánh số từ 1 trở đi theo trình tự phát bài:

$$(p, q) \leftrightarrow x = (p-1) * 2 + q$$

Gọi $x1$ là vị trí trước x nhận cùng đê và $x2$ – vị trí sau x nhận cùng đê, có:

$$x1 = x - k,$$

$$x2 = x + k$$

Nếu $x1 > 0$ thì tồn tại vị trí nhận cùng đê ngoài trước với địa chỉ vật lý ($u1, v1$),

Nếu $x2 \leq n$ thì tồn tại vị trí nhận cùng đê ngoài sau với địa chỉ vật lý ($u2, v2$).

$$u1 = (x1+1)/2, v1 = x1 - (u1-1)*2,$$

$$u2 = (x2+1)/2, v2 = x2 - (u2-1)*2.$$

Đưa ra kết quả: phân biệt 3 trường hợp:

- Không tồn tại $u1$ và $v1$,
- Tồn tại một trong 2 giá trị $u1$ hoặc $u2$,
- Tồn tại cả 2 giá trị $u1$ và $u2$.

Độ phức tạp giải thuật: O(1).

Chương trình:

```
#include <fstream>
#include <ctime>
#define NAME "tasks."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,k,p,q,u,v,x,x1,x2,u1,v1,u2,v2;

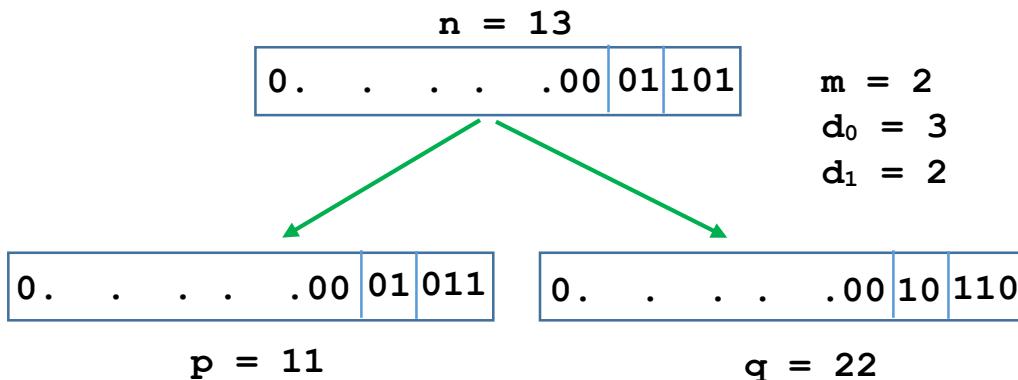
int main()
{clock_t aa=clock();
 fi>>n>>k>>p>>q;
 x=(p-1)*2+q; x1=x-k; x2=x+k;
 u1=0; if(x1>0)u1=(x1+1)/2,v1=x1-(u1-1)*2;
 u2=0; if(x2<=n)u2=(x2+1)/2,v2=x2-(u2-1)*2;
 u=-n;
 if(u1>0)u=u1,v=v1;
 if(u2>0 && u2-p<=p-u)u=u2,v=v2;
 if(u>0)fo<<u<<' '<<v; else fo<<-1;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VR22. CĂN LỀ

Tên chương trình: ALIGN.CPP

Xâu s độ dài không quá 64 chỉ chứa các ký tự 0 và 1 được cho dưới dạng số nguyên n , các bít của n được đánh số từ 0 đến 63, bít thứ i tương ứng với ký tự s_i , $i = 0 \div 63$. Có m đỏi tương ứng được đánh dấu bằng m trường trong xâu s , trường thứ j bao gồm d_j ký tự liên tiếp nhau, trường số 0 bao gồm các ký tự từ 0 đến d_0-1 , trường thứ 1 bao gồm các ký tự từ d_0 đến d_0+d_1-1 , trường thứ 2 bao gồm các ký tự từ d_0+d_1 đến $d_0+d_1+d_2-1$, ... Tổng độ dài các trường không vượt quá 64.

Để chuẩn hóa người ta chuyển các ký tự 1 trong mỗi trường về đứng cạnh nhau để nhận được số nguyên p tương ứng nhỏ nhất có thể (*chuẩn hóa min*) hoặc nhận được số nguyên q tương ứng lớn nhất có thể (*chuẩn hóa max*).



Với n , m và các d_j ($j = 0 \div m-1$) hãy xác định các số p và q .

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

- ✚ Dòng đầu tiên chứa hai số nguyên n và m ($0 < m \leq 10$),
- ✚ Dòng thứ 2 chứa m số nguyên d_0, d_1, \dots, d_{m-1} .

Kết quả: Đưa ra file văn bản ALIGN.OUT 2 số nguyên p và q , mỗi số trên một dòng.

Ví dụ:

ALIGN.INP
13 2 3 2

ALIGN.OUT
11 22



Giải thuật: Xử lý bit, tạo mặt nạ nhận dạng trường.

Tổ chức dữ liệu:

- ⊕ Mảng **int b[10]**: bộ lưu số lượng bít 1 trong trường **i**, $i = 0 \div m-1$,
- ⊕ Hằng **const uint64_t b1=1** – phục vụ tách bít.

Xử lý: Các phép xử lý cơ bản:

- Đếm số bít 1 ở mỗi trường: cần xác định giá trị bít **j** của **n**: **(n>>j) & b1**,
- Xóa các bít ở vùng cần căn lè: **(n>>t) << t**, trong đó $t = \sum_{i=0}^{m-1} d_i - 1$,
- Bật bít **j** của số nguyên **x**: **x |= (b1 << j)**;

Chương trình:

```
#include <fstream>
#include <ctime>
#define NAME "align."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
uint64_t n,p,q;
const uint64_t b1=1;
int m,d[11]={0},b[10]={0},t,t1,t2;

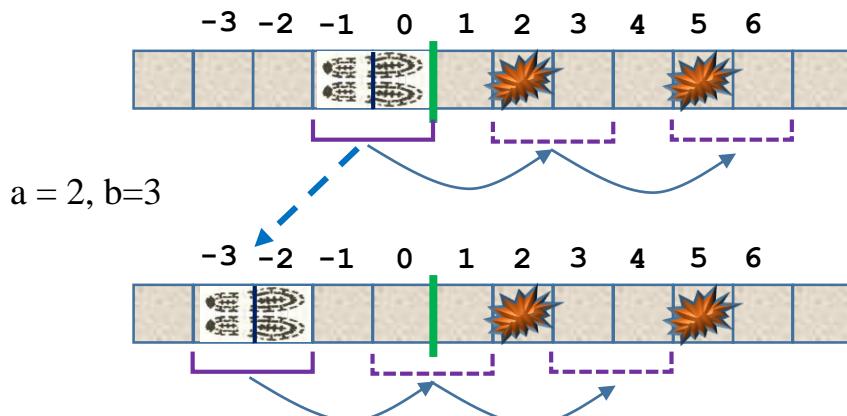
int main()
{clock_t aa=clock();
 fi>>n>>m;
 for(int i=0;i<m;++i)fi>>d[i];
 //Dem so bit1 trong moi truong
 t2=0;
 for(int i=0;i<m;++i)
 {
     t1=t2; t2+=d[i];
     for(int j=t1;j<t2;++j)b[i]+=(n>>j)&1;
 }
 //Xoa bit
 --t2;
 p=(n>>t2)<<t2; q=p;
 //Bat bit
 t=0;
 for(int i=0;i<m;++i)
 {
     for(int j=0;j<b[i];++j)p|=(b1<<(t+j));
     t+=d[i];
 }
 t=d[0]-1;
 for(int i=0;i<m;++i)
 {
     for(int j=0;j<b[i];++j)q|=(b1<<(t-j));
     t+=d[i+1];
 }
 fo<<p<<'n'<<q;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VR25. NHÃY CÓC

Tên chương trình: LEAPFROG.CPP

Để tăng cường thể lực chuẩn bị tham dự đại hội điền kinh sáng nào Steve cũng ra sân vận động tập nhảy cóc. Đường pit được chia thành các ô, đánh số 1, 2, 3, ... từ vạch xuất phát sang phải và 0, -1, -2, ... từ vạch xuất phát sang trái. Giày của Steve chiếm a ô từ $-a+1$ đến 0. Mỗi bước nhảy của Steve có độ dài là b ô, tức là mũi giày tiến lên phía trước b ô. Như vậy xuất phát từ vị trí nói trên, sau bước nhảy thứ nhất giày của Steve sẽ ở các ô từ $b-a+1$ đến b , ở bước nhảy tiếp theo giày sẽ ở các ô từ $2 \times b-a+1$ đến $2 \times b$, ...

Sáng nay ra sân tập Steve thấy trên đường pit còn vương lại một số vết bùn ở các ô p_1, p_2, \dots, p_n – kết quả của việc tu sửa mặt sân vận động đêm qua. Nếu xuất ở vị trí như bình thường có thể sẽ dẫm phải bùn, vừa bắn giày lại dễ bị ngã, điều mà Steve không hề muốn. Chính vì vậy Steve quyết định lùi điểm xuất phát lại một số ô sao cho ít dẫm phải ô có bùn nhất trong quá trình nhảy. Quãng đường mà Steve sẽ nhảy đủ dài, vượt qua tất cả các ô có bùn.



Hãy xác định số ô cần lùi lại và số lượng ô có bùn ít nhất mà Steve dẫm phải trong quá trình nhảy.

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

- ✚ Dòng đầu tiên chứa hai số nguyên a và b ($1 \leq a \leq b \leq 2 \times 10^5$),
- ✚ Dòng thứ 2 chứa số nguyên n ($1 \leq n \leq 2 \times 10^5$),
- ✚ Dòng thứ 2 chứa n số nguyên dương theo thứ tự tăng dần p_1, p_2, \dots, p_n ($1 \leq p_i \leq 10^9$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản LEAPFROG.OUT trên một dòng 2 số nguyên – số cần lùi và số lượng ô có bùn ít nhất mà Steve dẫm phải trong quá trình nhảy.

Ví dụ:

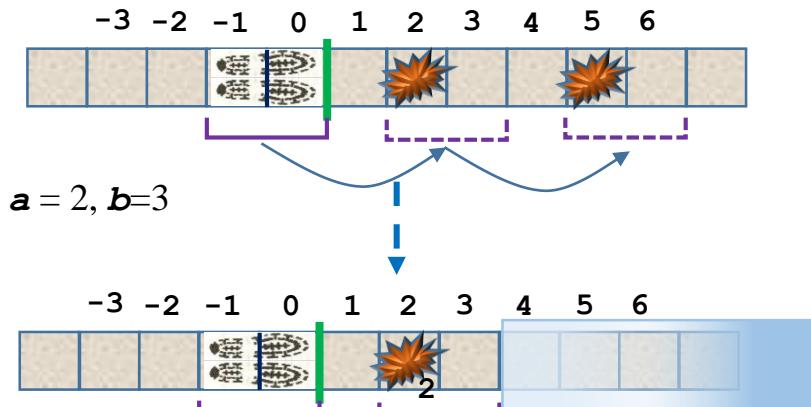
LEAPFROG.INP	LEAPFROG.OUT
2 3	
2	
2 5	2 0



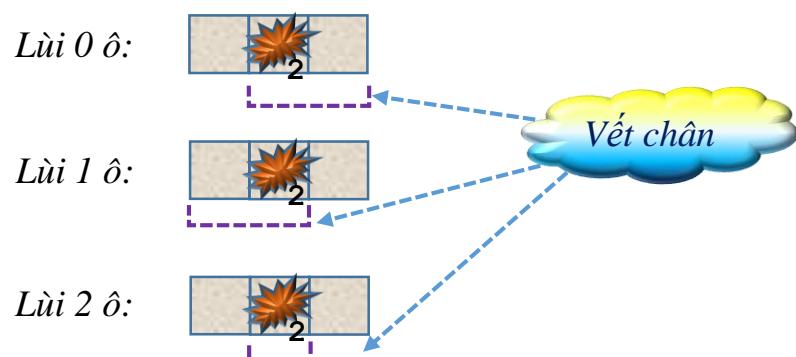
Giải thuật: Xử lý chu trình, tổ chức dữ liệu vòng tròn.

Nhận xét:

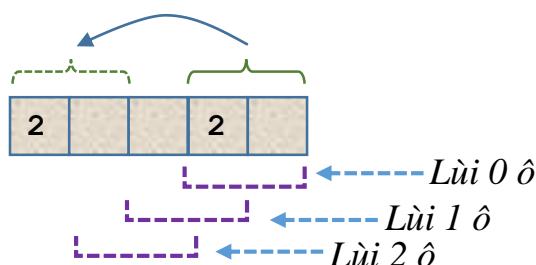
- ✚ Thông tin về bước nhảy được lặp lại theo chu kỳ b ,
- ✚ Thông tin về các ô dính bùn có thể ghi nhận trong đoạn độ dài b dưới dạng tích trọng số: số lần gặp ô có bùn ở mỗi ô trong chu kỳ:



- ✚ Việc thay đổi điểm đầu xuất phát sẽ làm thay đổi vị trí vết chân trong đoạn ghi nhận chu kỳ:



- ✚ Chỉ cần xét số ô lùi từ 0 đến $b-1$. Dấu vết chân thay đổi tương ứng với trường hợp khi dữ liệu được kết nối vòng tròn. Để thuận tiện xử lý dữ liệu ở các ô trên dấu vết chân được ghi lặp lại ở đầu, như vậy độ dài đoạn cần xử lý sẽ là $a+b$.



Để tính tổng trọng số các ô dính bùn trên vết chân: sử dụng tổng tiền tố.

Chương trình:

```
#include <fstream>
#include <ctime>
#define NAME "leapfrog."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,a,b,c,t,r,ib;
int64_t d[400002]={0};

int main()
{clock_t aa=clock();
 fi>>a>>b>>n;
 c=b+a;
 for(int i=1;i<= n;++i)
 {
     fi>>t;
     t=(t-1)%b+1;++d[t+a];
 }

 for(int i=1;i<=a;++i)d[i]=d[b+i];
 for(int i=1;i<=c;++i)d[i]+=d[i-1];

 ib=0;t=d[c]-d[b];
 for(int i=1;i<b;++i)
 if(t>d[c-i]-d[b-i]) {t=d[c-i]-d[b-i];ib=i;}

 fo<<ib<<' '<<t;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VR27. MÃ HÓA

Tên chương trình: CRYPT.???

Chèn gạch là một phương pháp mã hóa đơn giản thường được dùng trong các trò chơi. Một xâu **s** được sử dụng như *viên gạch* phục vụ mã hóa. Với thông báo **w** ban đầu, cứ sau mỗi ký tự nguyên âm người ta lại chèn thêm **s** vào để được xâu mã hóa **ws**. Các ký tự nguyên âm trong bảng chữ cái là tinh là **a, e, i, o, u** và **y**. Ví dụ, với **s = “fa”** và **w = “hello”**, ta có xâu đã mã hóa **ws** là **“hefallofa”**.

Cho xâu đã mã hóa **ws**. Hãy xác định xem có phải nó được mã hóa bằng phương pháp chèn gạch với **s = “fa”** hay không. Nếu đúng thì đưa ra xâu **w** ban đầu, trong trường hợp ngược lại – đưa ra thông báo **“impossible”**.

Dữ liệu: Vào từ file văn bản CRYPT.INP: gồm một dòng chứa xâu **ws** chỉ có các chữ cái là tinh thường và độ dài không quá 10^4 .

Kết quả: Đưa ra file văn bản CRYPT.OUT đưa ra xâu **w** ban đầu hoặc thông báo **impossible**.

Ví dụ:

CRYPT.INP
hefallofa

CRYPT.OUT
hello



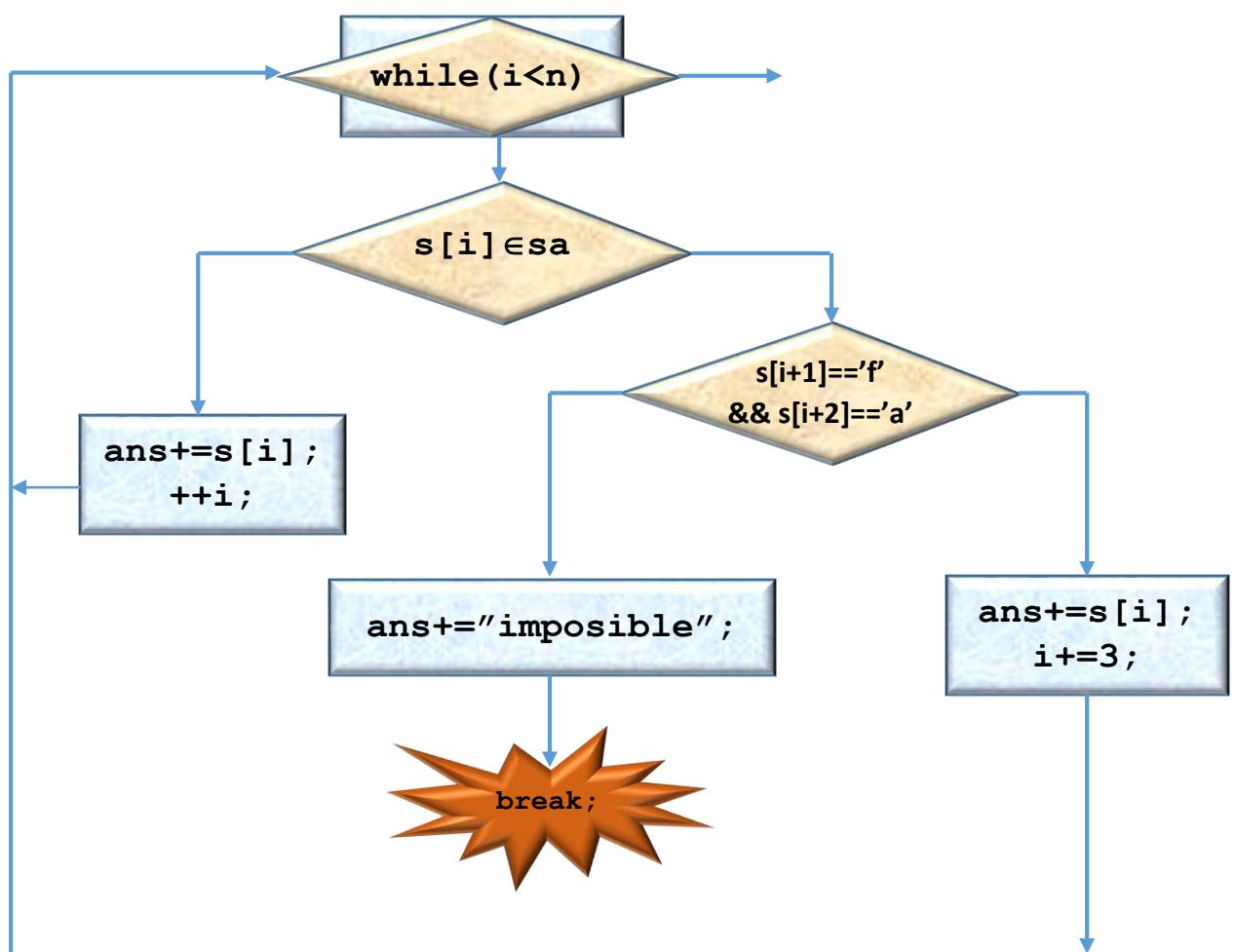
Giải thuật: Xử lý xâu + kỹ thuật hàng rào.

Tổ chức dữ liệu:

- ✚ Xâu **s** – chứa dữ liệu ban đầu,
- ✚ Xâu **sa** = “**a e i o u y**” – để nhận dạng nguyên âm,
- ✚ Xâu **ans** – chứa kết quả.

Xử lý:

- Nhập xâu s và xác định độ dài n của xâu,
- Bổ sung vào cuối s xâu “**” làm hàng rào xử lý, chuẩn bị **ans=""**;
- Xuất phát từ **i=0** xử lý ký tự **s[i]**:



Độ phức tạp của giải thuật: $O(n)$.

Chương trình:

```
#include <fstream>
#include <ctime>
#define NAME "crypt."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
string s,sa="aeiouy",ans;
int n,t,i;

int main()
{clock_t aa=clock();
 fi>>s;
n=s.size();s+="**"; ans="" ;i=0;
while(i<n)
{
    t=sa.find(s[i]);
    if(t>=0 && t<6)
    {
        if(s[i+1]=='f' &&s[i+2]=='a'){ans+=s[i];i+=3;}
        else {ans="impossible"; break;}
    }
    else {ans+=s[i];++i;}
}
fo<<ans;
clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

MỤC LỤC

VP24. ĐỒI ĐẦU	Tên chương trình: OPPOSITION.???	1
VQ07. MERLIN	Tên chương trình: MERLIN.???	6
VQ18. KHỞI TẠO MẬT KHẨU	Tên chương trình: PWGEN.???	8
VQ10. BIẾN ĐỒI DÃY SỐ	Tên chương trình: SEQ.???	10
VQ12. GIÁO SƯ CHAOS	Tên chương trình: CHAOS.???	14
VQ15. CÕI BÀI	Tên chương trình: CARDS.???	16
VQ21. SỬA HÀNG RÀO	Tên chương trình: WALL.???	19
VQ11. TRỢ GIÚP	Tên chương trình: F1.???	22
VQ22. TÌM KIÉM	Tên chương trình: SOUGHT.???	30
VQ20. CÁC CHÚ RÙA	Tên chương trình: TURTLES.???	33
VQ23. TÀU NGÀM	Tên chương trình: SUBMARINE.???	38
VQ24. ĐỘT KÍCH	Tên chương trình: SWOOP.???	44
VQ17. PHẢN VẬT CHẤT	Tên chương trình: ANTIMATTER.???	48
VQ26. PHÒNG THI	Tên chương trình: HALL.???	50
VQ27. PHẦN THƯỞNG	Tên chương trình: PRIZES.???	53
VQ29. ĐƯỜNG VÀNH ĐAI	Tên chương trình: CIRCLE.???	56
VQ30. ĐỒI MỚI	Tên chương trình: RENEWED.???	58
VQ31. LÁT NỀN	Tên chương trình: TILING.???	60
VQ32. VUI HAY BUỒN	Tên chương trình: HAPPY.???	66
VQ33. KIỂM TRA	Tên chương trình: AUDIT.???	68
VQ28. THU THUẾ	Tên chương trình: TAX.???	71
VQ35. QUÁ TẢI	Tên chương trình: OVERLOAD.???	80
VQ36. ĐƯỜNG ĐUA	Tên chương trình: RACETRACK.???	83
VQ39. TRÌNH TỰ	Tên chương trình: ORDER.???	86
VQ40. TRUY ĐUỔI	Tên chương trình: CHASE.???	89
VQ37. KHỞI TẠO KHÓA	Tên chương trình: GENKEY.???	92
VQ41. CƠ HỘI THẮNG	Tên chương trình: WINCHANCE.???	95
VQ44. ĐÈN HOA	Tên chương trình: FLOWERS.???	97
VQ45. MUA HÀNG TRỰC TUYẾN	Tên chương trình: BESTBUY.???	99
VQ46. LỄ HỘI ĐƯỜNG PHỐ	Tên chương trình: CARNIVAL.???	102
VQ42. BÁNH KẸP	Tên chương trình: BURGER.???	106
VQ43. TÁM THẨM	Tên chương trình: CARPET.???	110
VQ49. SAN BÀNG	Tên chương trình: ALIGN.???	113
VQ50. XÉP HÀNG	Tên chương trình: LONGQUEUE.???	116

VR01. SÁT NHẬP	<i>Tên chương trình: COMPANIES.???</i>	119
VR02. BÁNH NGỌT	<i>Tên chương trình: CAKES.???</i>	122
VR06. BOT	<i>Tên chương trình: BOT.???</i>	125
VR07. KẾT BẠN	<i>Tên chương trình: ONLYONE.???</i>	128
VR08. TƯỢNG ĐÀI	<i>Tên chương trình: MONUMENT.???</i>	131
HÀM Z		135
VR16. THUẦN CHỦNG	<i>Tên chương trình: PURE.???</i>	140
VR19. SỬA ĐƯỜNG	<i>Tên chương trình: REPAIRS.???</i>	143
VR14. TỪ ĐIỂN	<i>Tên chương trình: DICTIONARY.???</i>	146
VR20. PHÂN RÃ	<i>Tên chương trình: DECOMP.???</i>	149
VR23. CẤP TỐT	<i>Tên chương trình: GOODPAIRS.CPP</i>	153
VR24. ĐOÁN TỪ	<i>Tên chương trình: GUESS.CPP</i>	156
VR21. BÀI KIỂM TRA	<i>Tên chương trình: TASKS.CPP</i>	161
VR22. CĂN LỀ	<i>Tên chương trình: ALIGN.CPP</i>	164
VR25. NHảy CÓC	<i>Tên chương trình: LEAPFROG.CPP</i>	167
VR27. MÃ HÓA	<i>Tên chương trình: CRYPT.???</i>	170