

Dynamic Program and Data Structure

Dynamic Program - Quy hoạch động (DP) là một phương pháp hay giải quyết được nhiều lớp bài toán khó mà Duyệt khó có thể làm được với dữ liệu lớn. Tất nhiên DP cũng có nhiều dạng, mô hình như: Dãy con tổng S, Dãy con không giảm dài nhất, DP trạng thái, DP cấu hình, ...hay DP kết hợp với Duyệt, Sắp xếp, Tìm kiếm..

Nói chung khi gặp một bài toán việc khó nhất là nhìn nhận bài toán thuộc lớp bài toán nào, phương pháp giải nào, việc này các bạn tự đọc lại lý thuyết phần DP khi nào dùng DP. Khi gặp bài toán ở dạng này tốt nhất chúng ta nên đưa về các bài toán quen thuộc cơ bản đừng nghĩ chúng quá phức tạp.

Sau khi tìm được công thức DP việc quan trọng quyết định là cài đặt, điều này do kĩ năng lập trình của bạn quyết định. Có thể nói DP là một thuật toán tương đối khó cả về thuật toán và cài đặt vì vậy cần rèn luyện nhiều hơn để giải quyết các bài toán DP một cách có hiệu quả.

Cấu trúc dữ liệu (Data Structure) chiếm một vị trí hết sức quan trọng trong các thuật toán, nó làm giảm độ phức tạp của thuật toán đi rất nhiều ví dụ từ $O(n)$ xuống còn $O(\log n)$. Hầu hết các thuật toán DP hiện nay đều sử dụng cấu trúc dữ liệu để xử lý với dữ liệu lớn. Trong phạm vi chuyên đề này chúng ta sẽ đề cập tới: Interval Tree (IT), Binary Index Tree (BIT), Binary Search Tree (BST), Splay Tree, Treap, Heap...

Các bài tập dưới đây là một số bài tập trích từ đề thi học sinh giỏi quốc gia (**HSG QG**) một số năm, trên trang Web: [VNOL.INFO](#), Olympic tin học quốc tế (**IOI-International Olympiad in Informatics**) hoặc sưu tầm với mục đích chủ yếu là đưa ra một số kĩ thuật cài đặt, một số thuật toán DP hay cho các bạn tham khảo.

Shiningstar_193@yahoo.com

Selection 1: Skill of Dynamic Program basic



Problem 1: Cây khé 1

Mã bài :DTTUI1 VNOI.INFO

Trong truyện cổ tích "Cây Khé" ta đã biết rằng chim thần chở người anh với một cái túi ba gang đến hòn đảo đầy vàng bạc châu báu. Người em bắn khoăn không biết chọn đồ vật nào cho vào túi vì chỉ có một cái túi ba gang..

Giả sử rằng trên hòn đảo kia có N đồ vật khác nhau, đồ vật thứ i có giá trị là a_i và có thể tích là b_i . Cũng giả sử rằng cái túi mà người em mang đi chỉ có thể tích là M. Bạn hãy giúp người em chọn ra trong N đồ vật trên một số đồ vật sao cho tổng thể tích của các đồ vật được chọn không vượt quá M và tổng giá trị các đồ vật được chọn là lớn nhất.

Input: Cho trong file văn bản CAYKHE1.INP

- Dòng đầu tiên ghi hai số N, M ($N, M \leq 1000$)
- N dòng tiếp theo, dòng thứ i ghi hai số a_i và b_i lần lượt là giá trị và thể tích của đồ vật thứ i ($a_i, b_i \leq 1000$)

Output: Ghi ra file văn bản CAYKHE1.OUT 1 số duy nhất là tổng giá trị lớn nhất có thể cho vào trong túi

Ví dụ:

CAYKHE1.INP
5 10
20 3
19 1
30 7
24 3
15 6

CAYKHE1.OUT
63

Solution:

Gọi $L[j]$ là tổng giá trị của các đồ vật được chọn trong tập $(1, 2..i)$ lớn nhất thỏa mãn tổng trọng lượng nhỏ hơn hoặc bằng j.

Nếu $j \geq b[i]$ và $L[j] < L[j-b[i]]+a[i]$ thì $L[j]:=L[j-b[i]]+a[i]$ hay $L[j]:=Max(L[j], L[j-b[i]]+a[i])$

Có bạn sẽ thắc mắc nếu tính như trên thì có trường hợp $L[j]$ chọn đồ vật k và tồn tại $i > j$ mà $i-b[k]=j$ thì $L[i]=L[j-b[k]]+a[k]$ như vậy đồ vật k được chọn 2 lần dẫn đến vô lý. Ta thấy $L[i]$ chỉ phụ thuộc vào $L[j]$ với $j \leq i$ vì trọng lượng 1 vật luôn lớn hơn 0 và để không phải kiểm tra xem $j \geq b[i]$ ta chỉ duyệt j từ M trở về $b[i]$.

```
Procedure Solution;
Var
    i, j      : longint;
Begin
    For j:=0 to M do L[j]:=0; {Khởi tạo mang L[0..M]}
    For i:=1 to N do
        For j:=M downto b[i] do
            If L[j]<L[j-b[i]]+a[i] then
                L[j]:=L[j-b[i]]+a[i];
End;
```

Độ phức tạp: $O(n.m)$



Problem 2: Cây khé 2

Mã bài :DTTUI2 VNOI.INFO

Đã hết mùa khé. Trái khé cuối cùng đã rơi và giờ chỉ còn Khánh với cái cây toàn lá là lá. Khánh nhìn cây khé mà tiếc đứt ruột, nước mắt lấp lánh rơi. Vàng đâu nữa mà xài đây, ơi hỡi! Ngày nọ, con chim to to đó lại đến. Khé đâu ra mà cho nó ăn nữa bây giờ. Nhưng lạ lùng thay, chim to không đòi ăn khé. Số là vợ chim sai chim đi tìm dưa leo cho cô nàng đắp mặt. Chim to ngồi than với Khánh rằng nó đã đi một vòng Trái Đất rồi mà không tìm được trái dưa leo đủ to để đắp vừa khuôn mặt vợ y. Tưởng gì, dưa leo thì Khánh chẳng thiếu vì Khánh ngày nào cũng đắp mặt mà :) Khánh lôi trong tủ lạnh ra một trái dưa leo khổng lồ bự bằng cây dừa đưa cho chim to. Chim to cảm ơn rồi rít, rồi lại chờ Khánh ra đảo để... vơ vét.

Lần này, chim to muốn trả ơn Khánh hậu hĩnh hơn nên tặng Khánh một núi đá quý. Có N loại đá quý. Mỗi loại đá lại có trọng lượng, giá trị và số lượng riêng. Rút kinh nghiệm đợt 1, Khánh đã có may một cái túi bự gấp 10 lần cái túi lần trước mà vẫn không sao cho hết đồng đá quý đó vào được. Trái tim Khánh không thể chịu thêm nỗi đau nào quá lớn nữa. Các bạn hãy giúp anh ấy chọn các viên đá cần lấy sao cho anh ấy càng giàu càng tốt và dĩ nhiên là cái túi vẫn không được rách.

Input: Cho trong file văn bản CAYKHE2.INP

- Dòng 1: Hai số nguyên: Số loại đá quý N ($1 \leq N \leq 100$) và sức chứa của cái túi M ($1 \leq M \leq 10000$).
- N dòng tiếp theo: Mỗi dòng ghi 3 số nguyên: Khối lượng W_i , giá trị V_i và số lượng A_i của viên đá thứ i ($1 \leq W_i, V_i, A_i \leq 1000$).

Output: Ghi ra file văn bản CAYKHE2.OUT

- Ghi một số nguyên duy nhất là giá trị lớn nhất thu được.

Example

Input:

3 4
1 4 2
2 7 2
3 6 1

Output:

15

Solution:

Ta thấy bài này gần tương tự Problem 1 nhưng có điều khác là mỗi viên đá quý có số lượng là A_i viên và được chọn tối đa A_i viên thay vì chỉ được chọn 1.

Gọi $L[j]$ là tổng giá trị các đồ vật được chọn trong tập $(1, 2..i)$ lớn nhất thỏa mãn tổng trọng lượng nhỏ hơn hoặc bằng j .

Xét đến viên đá quý thứ i , trọng lượng túi j : $L[j] = \max(L[j], L[j-W[i]*k] + V[i]*k)$ với $0 \leq k \leq \min(A[i], j \text{ div } W[i])$.

Ta có thể hiểu là nếu $L[j] = L[j-W[i]*k] + V[i]*k$ là chọn k viên đá thứ i để cho vào túi khi đó trọng lượng túi có thể chứa được còn lại là $j-W[i]*k$ và giá trị khi chọn k viên này là $V[i]*k$.



Ta có thể mô tả thủ tục tính mảng L[0..M] như sau:

```
Procedure Solution;
Var
    i,j,k,imin :longint;
Begin
    For j:=0 to M do L[j]:=0;
    For i:=1 to N do
        For j:=M downto W[i] do
            Begin
                imin:=min(A[i],j div W[i]);
                For k:=0 to imin do
                    If L[j]<L[j-W[i]*k]+V[i]*k then
                        L[j]:=L[j-W[i]*k]+V[i]*k;
            End;
End;
```

Độ phức tạp: $O(n.m.k)$ với $k=m \text{ div } n$.

Nói chung 2 bài toán trên đều không khó đều là mô hình của bài toán cái túi quen thuộc nhưng chúng ta nên chọn các giải thuật tốt như hai giải thuật trên. Thông thường các bạn đều cài đặt [Problem 1](#) và [Problem 2](#) mảng L là mảng 2 chiều nhưng điều đó hoàn toàn có thể đưa về mảng 1 chiều giảm được rất nhiều không gian bộ nhớ tăng tốc độ chương trình.



Problem 3: SMARKET

Đề thi học sinh giỏi quốc gia năm 2007

An được mời tham gia trò chơi “Siêu thị may mắn” do đài truyền hình ZTV tổ chức.

Siêu thị được đặt trong trường quay truyền hình có n mặt hàng được đánh số từ 1 đến n và mặt hàng thứ i được niêm yết giá là c_i đồng, $i = 1, 2, \dots, n$.

Theo thể lệ của trò chơi, An được ban tổ chức tặng một thẻ mua hàng có giá trị là s đồng và phải dùng hết số tiền trong thẻ này để mua hàng trong siêu thị với điều kiện mặt hàng thứ i chỉ được mua với số lượng nhiều nhất là m_i , $i = 1, 2, \dots, n$.

An sẽ là người thắng cuộc nếu tìm được tổng số cách mua hàng thỏa mãn yêu cầu đặt ra và chỉ ra một cách mua hàng nếu có.

Yêu cầu: Hãy giúp An trở thành người thắng cuộc khi cho bạn biết trước các giá trị n , s , c_i và m_i ($1 \leq n \leq 500$; $1 \leq s \leq 10^5$; $1 \leq c_i \leq 10^4$; $1 \leq m_i \leq 100$) với $i = 1, 2, \dots, n$.

Input: Cho trong file văn bản SMARKET.INP

Dòng đầu tiên chứa hai số nguyên dương s và n .

Dòng thứ i trong n dòng tiếp theo chứa hai số nguyên dương c_i và m_i với $i = 1, 2, \dots, n$.

Output: Ghi ra file văn bản SMARKET.OUT Gồm 1 dòng duy nhất ghi số nguyên d là tổng số cách mua hàng tìm được.

Example

Input:

12 3
4 1
6 2
2 1

Output:

2

Solution:

Gọi $L[j]$ là số cách mua hàng khi có j đồng. Như vậy khi ta có khi ta mua các mặt hàng từ $(1..i-1)$ hết j đồng và số cách mua là $L[j]$ thì khi xét đến mặt hàng thứ i với k là số lượng mặt hàng i mà ta sẽ mua thì số cách mua hết $j+c[i]*k$ ($\leq S$) đồng sẽ tăng thêm một lượng $L[j]$. Ta có công thức quy hoạch động :

Nếu $j+c[i]*k \leq S$ thì $L[j+c[i]*k] := L[j+c[i]*k] + L[j]$ ($j=S..1, i=1..n, k=1..m_i$). Khởi tạo $L[0]=1$, $L[1..S]=0$.

Tại sao vì $j+c[i]*k \leq S$ vì nếu lớn hơn S rồi ta không cần quan tâm vì ta chỉ có tối đa S đồng thôi.

Trong qua trình cài đặt ta thấy nếu ta duyệt nếu ta duyệt j tăng dần từ 1 đến S , k tăng dần từ 1 đến m_i thì sẽ xảy ra trường hợp như sau: $j=0, L[0]=1, i=1, c[1]=1$. Đầu tiên $k=1$ thì $L[0+1*1]=L[0]+L[2]=1$, $k=2$ thì $L[0+1*2]=L[0]+L[4]=1$.

Tăng j và i giữ nguyên, $j=1$. Đầu tiên $k=1$ thì $L[1+1*1]=L[1]+L[1]=2$ đến đây ta đã thấy vô lí $L[2]$ với $i=1$ không thể nào bằng 2 được mà $L[2]=1$. Trường hợp này là do khi $j=j_1$ ta tính $L[j_1+x]$, j tăng lên j_1+x ta tính $L[j_1+x+y]$ ta có

$L[j_1+x+y]=L[j_1+x+y]+L[j_1+x]$ mà khi đó $L[j_1+x]$ không còn là số cách mua hết j_1+x đồng từ $i-1$ mặt hàng $(1..i-1)$ mà là

mua i mặt hàng (1..i) vì ta vừa tính $L[j_1+x]$ khi mua k mặt hàng i mà $c[i]*k=x$, tất nhiên $k \geq 1$. Nói ngắn gọn bạn phải hiểu

$L[j]$ trong công thức trên là số cách mua hết j đồng **từ i-1 mặt hàng (1..i-1)**

Để khắc phục tình trạng này ta duyệt j giảm từ S về 0.



Chương trình:

```
Const
    fi='SMARKET.INP';
    fo='SMARKET.OUT';
    maxn=500;
    maxs=100000;

Type
    arr1    =array[1..maxn] of longint;
    arr2    =array[0..maxs] of int64;

Var
    n,s      :longint;
    c,m      :arr1;
    L        :arr2;
    f        :text;

procedure nhap;
var
    i      :longint;
begin
    assign(f,fi);
    reset(f);
    readln(f,s,n);
    for i:=1 to n do readln(f,c[i],m[i]);
    close(f);
end;

procedure solution;
var
    i,j,k    :longint;
begin
    L[0]:=1; {Neu co 0 dong thi coi la co mot cach la khong mua gi ca}
    for j:=1 to s do L[j]:=0; { Khoi tao mang L }
    for i:=1 to n do
        for j:=s downto 0 do
            if L[j]>0 then
                for k:=1 to m[i] do
                    if j+k*c[i]<=s then
                        L[j+c[i]*k]:=L[j]+L[j+c[i]*k];
end;

procedure xuat;
begin
    assign(f,fo);
    rewrite(f);
    write(f,L[s]); {Ket qua nam o L[s]}
    close(f);
end;

begin
    nhap;
    solution;
    xuat;
end.
```

Độ phức tạp: $O(n \cdot s \cdot k)$ với $k = \text{Max}(m[1], m[2]..m[n])$.



Problem 4: FARMER

IOI 2004

Một người nông dân có một số các cánh đồng, mỗi một cánh đồng được bao quanh bởi các hàng cây bách. Ngoài ra ông ta còn có một tập các dải đất, mỗi một dải đất có một hàng cây bách. Trên các cánh đồng và dải đất, xen giữa hai cây bách liên tiếp là một cây ôliu. Tất cả các cây bách hoặc bao quanh cánh đồng hoặc nằm trên dải đất và tất cả các cây ôliu đều được trồng xen giữa hai cây bách liên tiếp.

Một ngày nọ người nông dân bị ốm rất nặng và ông ta cảm thấy mình sắp phải đi xa. Vài ngày trước khi qua đời ông đã gọi người con trai lớn nhất đến và nói với anh ta "Ta cho con chọn Q cây bách bất kỳ và tất cả các cây ôliu nằm giữa hai cây bách liên tiếp mà con đã chọn đều thuộc về con". Người con có thể chọn tổ hợp các cây bách bất kỳ từ các cánh đồng và dải đất. Vì người con rất thích ôliu nên anh ta muốn chọn Q cây bách sao cho anh ta thừa hưởng nhiều cây ôliu nhất có thể.

Cho trước thông tin về các cánh đồng và dải đất và số cây bách người con được chọn. Bạn hãy viết chương trình xác định số cây ôliu lớn nhất mà người con có thể được thừa hưởng.

INPUT

Dữ liệu được cho trong file FARMER.INP.

- Dòng đầu tiên bao gồm: đầu tiên là số nguyên Q ($0 \leq Q \leq 150000$): là số cây bách mà người con được chọn; sau đó là số nguyên M là số các cánh đồng; tiếp theo là số nguyên K là số dải đất.
- Dòng thứ hai chứa M số nguyên N_1, N_2, \dots, N_M ($3 \leq N_1, N_2, \dots, N_M \leq 150$): là số các cây bách trên các cánh đồng.
- Dòng thứ ba chứa K số nguyên R_1, R_2, \dots, R_K ($2 \leq R_1, R_2, \dots, R_K \leq 150$): là số cây bách trên dải đất.

Chú ý tổng số cây bách trên các cánh đồng và dải đất ít nhất cũng bằng Q

OUTPUT

Kết quả đưa ra tệp FARMER.OUT Gồm một dòng duy nhất chứa duy nhất một số nguyên: là số cây ôliu lớn nhất mà người con có thể thừa hưởng.

Example

Input:

```
17 3 3
13 4 8
4 8 6
```

Output:

```
17
```



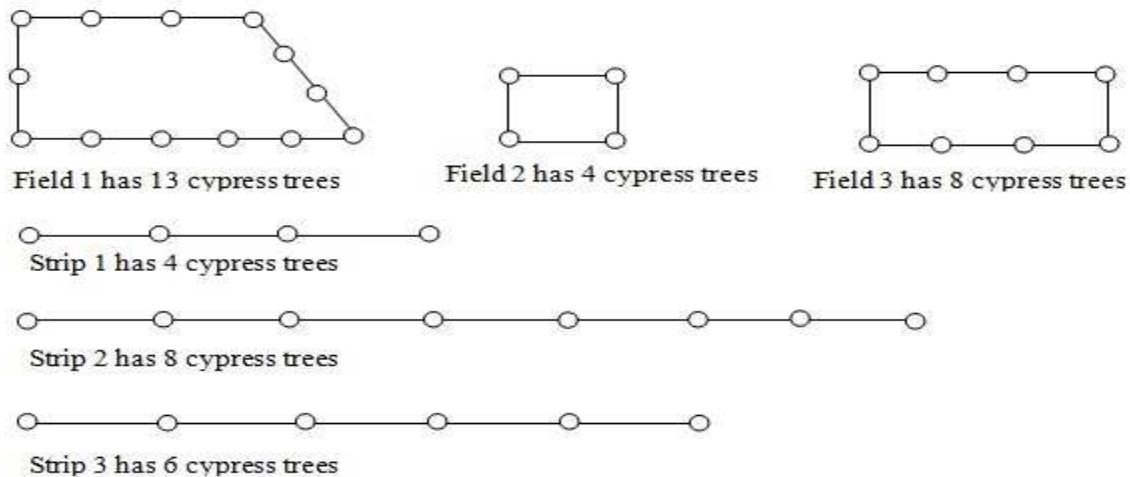


Figure 1 An example setting of cypress trees; olive trees are not shown.

Trong hình trên, giả thiết rằng người con được cho 17 cây bách ($Q=17$). Để có được số cây ôliu lớn nhất anh ta phải chọn tất cả các cây bách trong cánh đồng 1 và cánh đồng 2, với cách chọn này anh ta sẽ được thừa hưởng 17 cây ôliu.

Solution:

Bài toán này nghe thì có vẻ mới lạ nhưng thực chất ta có thể đưa về [Problem 1](#) như sau:

Ta coi Q là trọng lượng của túi. Số cây bách trên mỗi cánh đồng hoặc dài đất là trọng lượng của mỗi đồ vật còn số cây ôliu chính là giá trị của đồ vật như vậy $W[i]=N[i]$ ($W[i]$ là trọng lượng đồ vật thứ i) và $V[i]=N[i]$ ($V[i]$ là giá trị đồ vật thứ i) nếu là cánh đồng, $W[i]=R[i]$ và $V[i]=R[i]-1$ nếu là dài đất.

Vậy từ một bài thi IOI ta đưa về một bài toán rất quen thuộc đó là bài toán Knapsack 0/1. Khả năng nhìn nhận một bài toán là rất quan trọng vì vậy khi đọc một đề bài trước tiên các bạn thử suy nghĩ kĩ xem nó có thể đưa về một bài toán quen thuộc hay không.

Độ phức tạp: $O((n+m) \cdot Q)$



Problem 5: BONUS

Đề thi HSG QG ngày 1 năm 2011

Tuấn là người thắng cuộc trong một cuộc thi “Tìm hiểu kiến thức vũ trụ” và được nhận phần thưởng do công XYZ tài trợ. Các phần thưởng được bố trí trên một bảng hình vuông $n \times n$ có dạng một lưới ô vuông kích thước đơn vị. Các dòng của bảng được đánh số từ 1 đến n , từ trên xuống dưới và các cột của bảng được đánh số từ 1 đến n , từ trái sang phải. Ô nằm trên giao của dòng i và cột j gọi là ô (i,j) và trên đó có một món quà có giá trị là a_{ij} ($1 \leq i, j \leq n$).

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

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

Input: Vào từ file văn bản BONUS.INP

- Dòng thứ nhất chứa 2 số nguyên dương n, k ($n \leq 1000, \frac{n}{3} \leq k \leq n$).
- Dòng thứ i trong n dòng tiếp theo chứa n số nguyên dương, số thứ j là a_{ij} ($a_{ij} \leq 1000$).

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

Output: Ghi ra file văn bản BONUS.OUT một số nguyên duy nhất là tổng giá trị lớn nhất của các món quà mà Tuấn có thể nhận được.

Ví dụ:

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

Solution:

Đây là một bài toán khá đơn giản nếu ta tổ chức dữ liệu tốt.

Gọi $S[i,j]$ là tổng của hình chữ nhật có ô trên trái là $(1,1)$ và ô dưới phải là ô (i,j) .

Ta có thể tính $S[i,j]$ dựa vào $S[i-1,j]$, $S[i,j-1]$, $S[i-1,j-1]$, và $a[i,j]$ như sau: $S[i,j] := S[i-1,j] + S[i,j-1] - S[i-1,j-1] + a[i,j]$

Dễ dàng thấy công thức trên đúng khi biểu diễn trên hình vẽ:

	1	2	3	4
1	1	10	11	12
2	10	28	38	48
3	11	38	57	76
4	12	48	76	109

Giả sử $(i,j) = (4,4)$ là $S[4,4]$ khi đó $S[4,4] = S[4,3] + S[3,4] - S[3,3] + a[4,4]$.



Công thức trên đúng vì: $S[4,3]$ là tổng các ô có màu: $\textcolor{blue}{A}$ và $\textcolor{green}{A}$. $S[3,4]$ là tổng các ô có màu: $\textcolor{blue}{A}$ và $\textcolor{orange}{A}$. Khi đó $S[3,4]+S[4,3]$ bằng tổng các ô màu $\textcolor{green}{A}$ và $\textcolor{orange}{A}$ cộng 2 lần tổng các ô có màu $\textcolor{blue}{A}$. Mặt khác $S[3,3]$ là tổng các ô có màu $\textcolor{blue}{A}$. Như vậy $S[3,4]+S[4,3]-S[3,3]$ bằng tổng các ô đã tô màu (khác A) trên bảng nên $S[4,4]=S[3,4]+S[4,3]-S[3,3]+a[4,4]$.

Bây giờ ta xét tất cả các hình vuông cạnh k ô dưới phải là ô (i,j) trong lưới phần thưởng trên.

Tương tự như việc xây dựng mảng $S[1..N,1..N]$ ta có:

$$Value = S[i,j] - S[i,j-k] - S[i-k,j] + S[i-k,j-k].$$

Gọi Max là kết quả của bài toán khi đó nếu $\text{Max} < \text{Value}$ thì cập nhật lại $\text{Max} = \text{Value}$.

Chương trình:

```

Const
    fi='BONUS.INP';
    fo='BONUS.OUT';
    maxn=1000;
Type
    arr1  =array[1..maxn] of longint;
    arr2  =array[0..maxn,0..maxn] of int64;
Var
    n,k    :longint;
    a      :arr1;
    S      :arr2;
    Max   :int64;
    f      :text;

Procedure nhap;
Var
    i,j    :longint;
Begin
    Assign(f,fi);
    Reset(f);
    Readln(f,n,k);
    For i:=1 to n do
        For j:=1 to n do
            Read(f,a[i,j]);
    Close(f);
End;

Procedure Init;
Var
    i,j    :longint;
Begin
    Max:=0;
    For i:=1 to n do { Khoi tao mang S }
        Begin
            S[i,0]:=0;
            S[0,i]:=0;
        End;
    For i:=1 to n do
        For j:=1 to n do
            S[i,j]:=S[i-1,j]+S[i,j-1]-S[i-1,j-1]+a[i,j];

```



```

End;

Procedure Solution;
Var
    i,j      :longint;
    Value   :int64;
Begin
    For i:=k to n do
        For j:=k to n do
            Begin
                Value:=S[i,j]-S[i-k,j]-S[i,j-k]+S[i-k,j-k];
                If Max<Value then Max:=Value;
            End;
End;

Procedure Xuat;
Begin
    Assign(f,fo);
    Rewrite(f);
    Write(f,Max);
    Close(f);
End;

Begin
    Nhaph;
    Init;
    Solution;
    Xuat;
End.

```

Độ phức tạp: $O(n^2)$



Problem 6: BONUS 2

Hội thảo tập huấn GV chuyên Tin hè 2011

Tuấn là người thắng cuộc trong một cuộc thi “Tìm hiểu kiến thức vũ trụ” và được nhận phần thưởng do công XYZ tài trợ. Các phần thưởng được bố trí trên một bảng hình vuông $n \times n$ có dạng một lưới ô vuông kích thước đơn vị. Các dòng của bảng được đánh số từ 1 đến n , từ trên xuống dưới và các cột của bảng được đánh số từ 1 đến n , từ trái sang phải. Ô nằm trên giao của dòng i và cột j gọi là ô (i,j) và trên đó có một món quà có giá trị là a_{ij} ($1 \leq i, j \leq n$).

Để nhận phần thưởng, Tuấn được phép chọn hai hình vuông không giao nhau (có thể tiếp xúc) kích thước $k \times k$ chiếm trọn một số ô của bảng và nhận hết tất cả các phần quà nằm trong các ô của hai hình vuông đó.

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

Input: Vào từ file văn bản BONUS2.INP

- Dòng thứ nhất chứa 2 số nguyên dương n, k ($n \leq 1000, \frac{n}{3} \leq k \leq n$).
- Dòng thứ i trong n dòng tiếp theo chứa n số nguyên dương, số thứ j là a_{ij} ($a_{ij} \leq 1000$).

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

Output: Ghi ra file văn bản BONUS2.OUT một số nguyên duy nhất là tổng giá trị lớn nhất của các món quà mà Tuấn có thể nhận được.

Ví dụ:

BONUS2.INP	BONUS2.OUT
4 2 9 9 1 1 9 9 1 1 1 8 8 1 1 8 8 1	68

Solution:

Ta thấy chỉ thay đổi một câu “Tuấn được phép chọn một hình vuông” ở bài BONUS thành “Tuấn được phép chọn hai hình vuông không giao nhau (có thể tiếp xúc)” ta có ngay bài BONUS2 khó hơn rất nhiều bài BONUS.

Bài này đòi hỏi các bạn phải có sự nhận xét tốt, và tinh tế trong cài đặt thì mới có thể giải quyết được.

Ta hãy xem lại thuật toán ở bài trước ta đi xét tất cả các hình vuông ô dưới phải là (i,j) kích thước $k \times k$ rồi tìm hình vuông có tổng lớn nhất. Ở bài này ta xây dựng mảng $S[0..N, 0..N]$ tương tự bài BONUS.

Ta có thuật toán sau:

Gọi $C[i], D[j]$ lần lượt là tổng giá trị lớn nhất mà Tuấn nhận được khi chọn một hình vuông $k \times k$ từ bảng kích thước $i \times n$ và $n \times j$. Ban đầu khởi tạo cả 2 mảng D và C đều là 0.

Khi đó xét chọn hình vuông thứ 2 có ô dưới phải là (i,j) kích thước $k \times k$ thì tổng hai hình vuông được chọn lớn nhất là: $Value := (S[i,j] - S[i-k,j] - S[i,j-k] + S[i-k,j-k]) + Max(C[i-k], D[j-k])$



Trong đó: tổng $\text{Valuecd} := S[i,j] - S[i-k,j] - S[i,j-k] + S[i-k,j-k]$ giá trị nhận được của hình vuông thứ hai được chọn.

$\text{Max}(C[i-k], D[j-k])$ là ô vuông có giá trị lớn nhất không giao với hình vuông thứ hai được chọn.

Gọi MaxS là kết quả của bài toán khi đó mỗi lần tính giá trị Value ta so sánh Max với Value nếu Max < Value thì cập nhật lại Max. Đồng thời khi xét đến ô (i,j) sau khi tính xong Value ta tính C[i] và D[j] bằng công thức sau:

$$C[i] := \text{Max}(C[i], C[i-1], \text{Valuecd}).$$

$$D[j] := \text{Max}(D[j], D[j-1], \text{Valuecd}).$$

Chương trình:

```
Const
    fi='BONUS2.INP';
    fo='BONUS2.OUT';
    maxn=1000;
Type
    arr1 = array[0..maxn] of longint;
    arr2 = array[0..maxn,0..maxn] of int64;
    arr3 = array[0..maxn] of int64;
Var
    n, k : longint;
    a : arr2;
    S : arr2;
    d, c : arr3;
    MaxS : int64;
    f : text;

Procedure nhap;
Var
    i, j : longint;
Begin
    Assign(f,fi);
    Reset(f);
    Readln(f,n,k);
    For i:=1 to n do
        For j:=1 to n do
            Read(f,a[i,j]);
    Close(f);
End;

Procedure Init;
Var
    i, j : longint;
Begin
    MaxS:=0;
    For i:=1 to n do { Khoi tao mang S,D,C }
        Begin
            S[i,0]:=0;
            S[0,i]:=0;
            d[i]:=0;
            c[i]:=0;
        End;
    For i:=1 to n do
        For j:=1 to n do
            S[i,j]:=S[i-1,j]+S[i,j-1]-S[i-1,j-1]+a[i,j];
```



```

End;

Function Max(a,b:int64):int64;
Begin
    if a>b then exit(a);
    exit(b);
End;

Procedure Solution;
Var
    i,j      :longint;
    Value   :int64;
    Valuecd      :int64;
Begin
    For i:=k to n do
        For j:=k to n do
            Begin
                Valuecd:=S[i,j]-S[i-k,j]-S[i,j-k]+S[i-k,j-k];
                Value:=Valuecd+Max(c[i-k],d[j-k]); { Tinh Value theo cong thuc }
                If MaxS<Value then MaxS:=Value;
                If c[i]<Max(c[i-1],valuecd) then c[i]:=Max(c[i-1],Valuecd);
                If d[j]<Max(d[j-1],Valuecd) then d[j]:=Max(d[j-1],Valuecd);
            End;
End;

Procedure Xuat;
Begin
    Assign(f,fo);
    Rewrite(f);
    Write(f,MaxS);
    Close(f);
End;

Begin
    Nhap;
    Init;
    Solution;
    Xuat;
End.

```

Độ phức tạp: $O(n^2)$

Note: Ta thấy từ một bài toán có thuật giải tương đối đơn giản chỉ sửa một chút thôi đã làm bài toán trở nên phức tạp hơn rất nhiều. Solution trên là một thuật toán rất hay cài đặt không quá phức tạp có thể mở rộng với $n=5000$ thậm chí 10000 , thay vì phải dùng phương pháp khác cài đặt phức tạp hơn rất nhiều mà tốc độ chương trình lại rất chậm. Thực ra Solution của [Problem 5](#) và [Problem 6](#) là kết hợp DP và [duyệt vét cạn](#). Đôi khi ta không nên rạch rời từng phương pháp mà nên kết hợp lại để giải bài tập sẽ đạt hiệu quả cao hơn điều này ta thấy rất rõ ở các bài toán kết hợp giữa sắp xếp và tìm kiếm.

Khi có một thuật toán tốt chúng ta cũng phải có cách cài đặt linh hoạt, việc đó không những làm chương trình sáng sủa mà còn làm tăng tốc độ lập trình và chương trình lên rất nhiều- điều rất quan trọng trong các kì thi như: HSG QG, ACM...



Problem 7: Trò chơi chẵn lẻ

Đề thi HSG QG ngày 2 năm 2011

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

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

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

Dữ liệu:

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

o Dòng thứ nhất chứa số nguyên dương n ($n \leq 500$).

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

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

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

Ví dụ:

Dữ liệu	Kết quả
2	YES
3	NO
1 2 2	
1 2 3	
2 3 1	
4	
2 2 2 2	
2 2 2 2	
2 2 2 2	
2 2 2 2	

Ràng buộc: 50% số test ứng với 50% số điểm của bài có $n \leq 50$.

Solution:

Gọi $L[i,j]$ là trạng thái thắng (TRUE) hay thua (FALSE) khi bảng trò chơi có kích thước $i \times j$ của người đi trước.



Ta thấy:

- + $L[i,j] = \text{TRUE} \Leftrightarrow L[i-1,j] = \text{FALSE}$ nếu tổng các số trên hàng i chẵn, hoặc $L[i,j-1] = \text{FALSE}$ nếu tổng các số trên cột j chẵn.
- + $L[i,j] = \text{FALSE} \Leftrightarrow L[i-1,j] = \text{TRUE}$ hoặc tổng các số trên hàng i lẻ và $L[i,j-1] = \text{TRUE}$ hoặc tổng các số trên cột j lẻ.

Tóm lại ta có: $L[i,j] = x \text{ or } y$;

$x = \text{TRUE}$ nếu tổng các số trên hàng i chẵn và $L[i-1,j] = \text{FALSE}$; $x = \text{FALSE}$ trong trường hợp còn lại.

$y = \text{TRUE}$ nếu tổng các số trên cột j chẵn và $L[i,j-1] = \text{FALSE}$; $y = \text{FALSE}$ trong trường hợp còn lại.

Vấn đề còn lại là việc tính tổng các số trên hàng i và cột j :

Gọi $S[i,j]$ là tổng các số trên bảng trò chơi kích thước $i \times j$. Ta tính $S[i,j]$ tương tự bài BONUS:

$S[i,j] := S[i-1,j] + S[i,j-1] + a[i,j] - S[i-1,j-1]$;

Gọi h là tổng các số trên hàng i , ta có: $h = S[i,j] - S[i-1,j]$;

Gọi c là tổng các số trên cột j , ta có: $c = S[i,j] - S[i,j-1]$;

Từ việc tính trước mảng S ta đưa bài toán có độ phức tạp $O(n^2)$ tốt hơn rất nhiều nếu tính trực tiếp sẽ có độ phức tạp $O(n^3)$.

Chương trình:

```
Const
    tfi='PARIGAME.INP';
    tfo='PARIGAME.OUT';
    maxn=500;

Type
    arr1 = array[1..maxn,1..maxn] of longint;
    arr2 = array[0..maxn,0..maxn] of qword;
    arr3 = array[0..maxn,0..maxn] of boolean;

Var
    n      :longint;
    k      :longint;
    a      :arr1;
    s      :arr2;
    L      :arr3;
    fi,fo :text;

Procedure nhap;
var
    i,j      :longint;
begin
    readln(fi,n);
    for i:=1 to n do
        for j:=1 to n do
            read(fi,a[i,j]);
end;

Procedure Init;
var
    i,j      :longint;
begin
    for i:=0 to n do
        begin
```



```

        s[0,i]:=0;
        s[i,0]:=0;
    end;
for i:=1 to n do
    for j:=1 to n do
        s[i,j]:=s[i-1,j]+s[i,j-1]+a[i,j]-s[i-1,j-1];
for i:=1 to n do
begin
    if s[i,1] mod 2=0 then L[i,1]:=true
    else L[i,1]:=false;
    if s[1,i] mod 2=0 then L[1,i]:=true
    else L[1,i]:=false;
end;
end;

Procedure solution;
var
    i,j      :longint;
    x,y      :boolean;
    h,c      :qword;
begin
    for i:=2 to n do
        for j:=2 to n do
            begin
                h:=s[i,j]-s[i-1,j];
                c:=s[i,j]-s[i,j-1];
                if (h mod 2=0) and (L[i-1,j]=false) then x:=true
                    else x:=false;
                if (c mod 2=0) and (L[i,j-1]=false) then y:=true
                    else y:=false;
                L[i,j]:=x or y;
            end;
end;

Procedure xuat;
begin
    if L[n,n]=true then writeln(fo,'YES')
        else writeln(fo,'NO');
end;

Procedure Process;
var
    i      :longint;
begin
    assign(fi,tfi);
    reset(fi);
    assign(fo,tfo);
    rewrite(fo);
    readln(fi,k);
    for i:=1 to k do
        begin
            nhap;
            Init;
            solution;
            xuat;
        end;
    close(fi);
    close(fo);
end;

begin
    Process;
end.

```

Độ phức tạp: $O(k \cdot n^2)$



Problem 8: Phân trang

Đề thi HSG QG năm 1999

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

Input

- Dòng 1 chứa 2 số nguyên dương N, Le ($N \leq 6000, Le \leq 1000$)
- Dòng thứ i trong số N dòng tiếp theo chứa số nguyên dương wi ($w_i \leq Le$), $i=1, 2, \dots, N$

Output

- In ra hệ số phạt nhỏ nhất

Ví dụ

Input:

4 5
3
2
2
4

Output:

2

Solution:

Đây là một bài DP không mấy phức tạp chỉ cần suy nghĩ chút thôi là bạn có thể giải được.

Gọi L[i] là hệ số phạt nhỏ nhất khi phân trang các từ từ 1 đến i. Bây giờ ta sẽ xét từ i và các từ cùng dòng với từ i. Ta sẽ tiến hành phân cho các từ từ j đến i trên một dòng tất nhiên $\sum_{k=j}^i w[k] \leq Le$ khi đó $L[i] = \text{Max}(L[j], Le - \sum_{k=j}^i w[k])$ (1).

Công việc của ta là duyệt j từ i về 1 tìm j sao cho (1) đạt Min. Để tính $\sum_{k=j}^i w[k]$ ta dùng mảng S có ý nghĩa như sau:

$S[i] = \sum_{k=1}^i w[k]$ tất nhiên ta sẽ tính S[i] theo công thức $S[i] = S[i-1] + w[i] \Rightarrow \sum_{k=j}^i w[k] = S[i] - S[j-1]$.

Chương trình:

```
Const
    fi='PTRANG.INP';
    fo='PTRANG.OUT';
    maxn=6000;

Type
    arr1 = array[0..maxn] of longint;

Var
    n, le : longint;
    w : arr1;
    s : arr1;
```



```

l      :arr1;
f      :text;

Procedure Nhap;
var
    i      :longint;
begin
    s[0]:=0;
    assign(f,fi);
    reset(f);
    readln(f,n,le);
    for i:=1 to n do
        begin
            readln(f,w[i]);
            s[i]:=w[i]+s[i-1];
        end;
    close(f);
end;

function max(a,b:longint):longint;
begin
    if a>b then exit(a);
    exit(b);
end;

Procedure solution;
var
    i,j      :longint;
    k        :longint;
    u        :longint;
begin
    l[0]:=0;
    w[0]:=0;
    for i:=1 to n do
        begin
            l[i]:=maxlongint;
            for j:=i downto 1 do
                if s[i]-s[j-1]<=le then
                    begin
                        k:=le-(s[i]-s[j-1]);
                        u:=Max(L[j-1],k);
                        if l[i]>u then l[i]:=u;
                    end
                else break;{neu s[j]+s[j+1]+..+s[i]>le thi khong tim nua}
            end;
        end;
end;

Procedure xuat;
begin
    assign(f,fo);
    rewrite(f);
    write(f,l[n]);
    close(f);
end;

begin
    nhap;
    solution;
    xuat;
end.

```

Độ phức tạp: $O(n^2)$

Các bạn có thể TEST trên [VNOI.INFO](#) với mã bài là **PTRANG**



Problem 9: Lát gạch

Mã bài: LATGACH VNOL.INFO

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

Input

Gồm nhiều test, dòng đầu ghi số lượng test T ($T \leq 100$).
 T dòng sau mỗi dòng ghi một số N .

Output

Ghi ra T dòng là số cách lát tương ứng.

Example

Input:

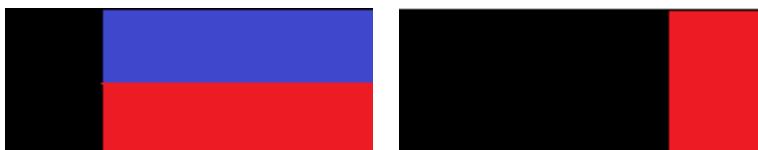
3
1
2
3

Output:

1
2
3

Solution:

Bài này tương đối đơn giản tôi sẽ đưa cho các bạn một hình vẽ sau các bạn tự suy ra nha:



Các bạn nghĩ ra rồi chứ tôi cung cấp Code cho các bạn tìm hiểu tại sao lại đưa về tính số Fibonacci thứ $n+1$ ☺.

Note: Nhớ xử lý số lớn nha.

Độ phức tạp: $O(n \cdot k)$ với k là chi phí xử lý số lớn.



Chương trình:

```
Const
    tfi='LATGACH.INP';
    tfo='LATGACH.OUT';
    maxn=201;

Var
    T      :Longint;
    N      :longint;
    Fibo   :array[1..maxn] of string;
    fi,fo :text;

function add(s1,s2:string):string;
var
    i,x,y,t,nho   :longint;
    s,kq   :string;
begin
    kq:='';
    nho:=0;
    while length(s1)<length(s2) do s1:='0'+s1;
    while length(s2)<length(s1) do s2:='0'+s2;
    for i:=length(s1) downto 1 do
        begin
            val(s1[i],x);
            val(s2[i],y);
            t:=x+y+nho;
            str(t mod 10,s);
            kq:=s+kq;
            nho:= t div 10;
        end;
    if nho>0 then kq:='1'+kq;
    exit(kq);
end;

procedure Build;
var
    i      :longint;
begin
    Fibo[1]:='1';
    Fibo[2]:='1';
    for i:=3 to maxn do Fibo[i]:=Add(Fibo[i-1],Fibo[i-2]);
end;

procedure Run;
var
    i      :longint;
begin
    assign(fi,tfi);
    reset(fi);
    assign(fo,tfo);
    rewrite(fo);
    readln(fi,T);
    for i:=1 to T do
        begin
            readln(fi,N);
            writeln(fo,fibo[N+1]);
        end;
    close(fi);
    close(fo);
end;

Begin
    Build;
    Run;
End.
```



Problem 10: Xâu nhị phân

Mã bài: SPBINARY VNOI.INFO

Cho 2 số n và k ($2 \leq k \leq n \leq 600$)

Hãy đếm xem có bao nhiêu xâu nhị phân độ dài n mà không có quá k số 0 hoặc k số 1 nào liên tiếp nhau.

Input

Gồm 1 dòng duy nhất là 2 số n và k

Output

Gồm 1 dòng duy nhất là số dãy nhị phân thoả mãn

Example

Input:

3 2

Output:

6

Giải thích : Đó là các xâu 001 , 010 , 011 , 100 , 101 , 110.

Note:Nhớ xử lý số lớn.

Solution:

Gọi $B[i,j]$ là số xâu nhị phân độ dài i kết thúc là bít j có không có quá k chữ số 0 hoặc 1 liên tiếp nhau ($j=0$ hoặc $j=1$).

Ta có: $B[n,0]+B[n,1]$ chính là đáp án của bài toán.

Nếu $i \leq k$ thì $B[i,0]+B[1,0]=2^i$ chính là số xâu nhị phân độ dài i vì $i \leq k$ không bao giờ tồn tại k chữ số 0 hoặc 1 liên tiếp. Hay nói cách khác: $B[i,0]=B[i,1]=B[i-1,0]+B[i-1,1]$;

Nếu $i > k$ ta thấy: $B[i,0]=B[i-1,1]+B[i-2,1]+\dots+B[i-k,1]$ có nghĩa là nếu chữ số thứ i là không thì chữ số thứ $i-1, i-2, \dots, i-k$ là 1 để thoả mãn số chữ số 0 liên tiếp không quá k. Tương tự với $B[i,1]$ ta có $B[i,1]=B[i-1,0]+B[i-2,0]+\dots+B[i-k,1]$.

Ta thấy cách tính $B[i,0]$ và $B[i,1]$ rất giống nhau bây giờ ta sẽ cộng $B[i,0]$ và $B[i,1]$, gọi $L[i]$ là số xâu nhị phân độ dài i không có quá k chữ số 0 hoặc 1 liên tiếp nhau trong xâu $\Rightarrow L[i]=B[i,0]+B[i,1]$. Ta có kết quả sau:

Nếu $i \leq k$ thì $L[i]:=B[i,0]+B[i,1]=2*(B[i-1,0]+B[i-1,1])=2*L[i-1]$.

Nếu $i > k$ thì $L[i]:=B[i,0]+B[i,1]=(B[i-1,1]+B[i-2,1]+\dots+B[i-k,1])+(B[i-1,0]+B[i-2,0]+\dots+B[i-k,1])=(B[i-1,1]+B[i-1,0])+B[i-2,1]+B[i-2,0]+\dots+(B[i-k,1]+B[i-k,0])=L[i-1]+L[i-2]+\dots+L[i-k]$.

Tóm lại ta có:

+ $i \leq k: L[i]:=2*L[i-1]$.



+ $i > k : L[i] := \sum_{j=1}^{i-1} L[j]$

Nhưng công thức trên vẫn chưa đủ nhanh ta gọi $S[i] := \sum_{j=1}^i L[j] \Rightarrow S[i] := S[i-1] + L[i]$.

Như vậy ta có thể tính lại: và ta có thể dùng chính mảng L để thay thế mảng S ở dưới.

+ $i \leq k : L[i] := 2 * L[i-1]; S[i] := S[i-1] + L[i]$.

+ $i > k : L[i] := S[i-1] - S[i-k-1]; S[i] := S[i-1] + L[i]$.

Chương trình:

```
{ $MODE OBJFPC }
Const
    fi='SPBINARY.INP';
    fo='SPBINARY.OUT';
    maxn=600;

Type
    tstring=string;
    arr1=array[0..maxn] of tstring;
Var
    n, k      :longint;
    l         :arr1;
    f         :text;

{$R-}
Procedure nhap;
begin
    assign(f,fi);
    reset(f);
    readln(f,n,k);
    close(f);
end;

function add(s1,s2:tstring):tstring;inline;
var
    i,x,y,t,nho:longint;
    s,kq      :tstring;
begin
    while length(s1)<length(s2) do s1:='0'+s1;
    while length(s2)<length(s1) do s2:='0'+s2;
    nho:=0;
    kq:='';
    for i:=length(s1) downto 1 do
        begin
            val(s1[i],x);
            val(s2[i],y);
            t:=x+y+nho;
            str(t mod 10,s);
            kq:=s+kq;
            nho:=t div 10;
        end;
    if nho>0 then kq:='1'+kq;
    exit(kq);
end;

function sub(s1,s2:tstring):tstring;inline;
var
    i,x,y,t,nho:longint;
    s,kq      :tstring;
begin
```



```

while length(s1)<length(s2) do s1:='0'+s1;
while length(s2)<length(s1) do s2:='0'+s2;
nho:=0;
kq:='';
for i:=length(s1) downto 1 do
begin
    val(s1[i],x);
    val(s2[i],y);
    t:=x-y-nho;
    if t<0 then
        begin
            t:=t+10;
            nho:=1;
        end
    else nho:=0;
    str(t,s);
    kq:=s+kq;
end;
while (length(kq)>1) and (kq[1]='0') do delete(kq,1,1);
exit(kq);
end;

Procedure calcmu2;inline;{Neu i<=k}
var
    i      :longint;
    x      :tstring;
begin
    l[0]:='0';
    l[1]:='2';
    for i:=2 to k do
    begin
        x:=sub(l[i-1],l[i-2]);
        x:=add(x,x);
        l[i]:=add(x,l[i-1]);{Dung mang L thay the mang S}
    end;
end;

Procedure solution;inline;{Neu i>k}
var
    i,j      :longint;
begin
    for i:=k+1 to n do
    begin
        l[i]:=sub(l[i-1],l[i-k-1]);
        l[i]:=add(l[i],l[i-1]);{Dung mang L thay the mang S}
    end;
end;

Procedure xuat;
begin
    assign(f,fo);
    rewrite(f);
    write(f,sub(l[n],l[n-1]));
    close(f);
end;

begin
    nhap;
    calcmu2;
    solution;
    xuat;
end.

```

Độ phức tạp: $O(n.k)$ với k là chi phí xử lý số lớn.



Problem 11: Đường đi trên lưới

Đề thi học sinh giỏi quốc gia năm 2006

Cho một lưới ô vuông gồm m dòng và n cột. Các dòng được đánh số từ 1 đến m từ trên xuống dưới, các cột được đánh số từ 1 đến n từ trái qua phải. Ô nằm ở vị trí dòng i và cột j của lưới được gọi là ô (i, j) và khi đó, i được gọi là tọa độ dòng còn j được gọi là tọa độ cột của ô này. Trên ô (i, j) của lưới ghi số nguyên dương a_{ij} , $i = 1, 2, \dots, m$; $j = 1, 2, \dots, n$. Trên lưới đã cho, từ ô (i, j) ta có thể di chuyển đến ô (p, q) nếu các điều kiện sau đây được thỏa mãn:

- $j < n$; $i \leq p$; $j \leq q$ và $i + j < p + q$;
- a_{ij} và a_{pq} có ước số chung lớn hơn 1.

Ta gọi một cách di chuyển từ mép trái sang mép phải của lưới là cách di chuyển bắt đầu từ một ô có tọa độ cột bằng 1 qua các ô của lưới theo qui tắc di chuyển đã nêu và kết thúc ở một ô có tọa độ cột bằng n .

Yêu cầu: Tính số cách di chuyển từ mép trái lưới sang mép phải lưới.

Dữ liệu vào

- Dòng đầu tiên ghi 2 số nguyên dương m, n .
- Dòng thứ i trong số m dòng tiếp theo ghi n số nguyên dương $a_{i1}, a_{i2}, \dots, a_{in}$ là các số trên dòng thứ i của lưới

Kết quả

Ghi ra 1 số nguyên là phần dư của số lượng cách di chuyển tìm được cho 10^9 .

Hạn chế

Trong tất cả các test: $1 < m, n \leq 100$; $a_{ij} \leq 30000$, $i=1,2,\dots,m; j=1,2,\dots,n$. Có 50% số lượng test với $m, n \leq 50$.

Ví dụ

Dữ liệu	Kết quả
2 2	4
2 4	
6 8	

Solution:

Bài này tương tự bài đường đi trên lưới quen thuộc nhưng thay vì từ một ô chỉ đến được 3 ô thì ở đây từ một ô ta có thể đến được một số ô thỏa mãn theo yêu cầu trên. Bạn có thể TEST trên [VNOI.INFO](#) với mã bài là **NKPATCH**

Chương trình:

```
Const
  fi='NKPATCH.INP';
  fo='NKPATCH.OUT';
  maxn=100;
  e=10000000000;

Type
  arr1  =array[1..maxn,1..maxn] of integer;
  arr2  =array[1..maxn,1..maxn] of longint;
```



```

Var
  m,n      :longint;
  a        :arr1;
  l        :arr2;
  kq       :longint;
  f        :text;

procedure nhap;
var
  i,j      :longint;
begin
  assign(f,fi);
  reset(f);
  readln(f,m,n);
  for i:=1 to m do
    for j:=1 to n do
      read(f,a[i,j]);
  close(f);
end;

function UCLN(a,b:longint):longint;inline;
Var
  r      :longint;
begin
  while b<>0 do
    begin
      r:=a mod b;
      a:=b;
      b:=r;
    end;
  exit(a);
end;

Procedure solution;
var
  i,j      :longint;
  u,v      :longint;
begin
  for i:=1 to m do
    for j:=2 to n do
      l[i,j]:=0;
  for i:=1 to m do l[i,1]:=1;
  for i:=1 to m do
    for j:=1 to n do
      for u:=i downto 1 do
        for v:=j downto 1 do
          if (UCLN(a[i,j],a[u,v])>1) and (u+v<i+j) and (v<n)
then l[i,j]:=(l[i,j]+l[u,v]) mod e;
  kq:=0;
  for i:=1 to m do kq:=(kq+l[i,n]) mod e;
end;

procedure xuat;
begin
  assign(f,fo);
  rewrite(f);
  write(f,kq);
  close(f);
end;

begin
  nhap;
  solution;
  xuat;
end.

```

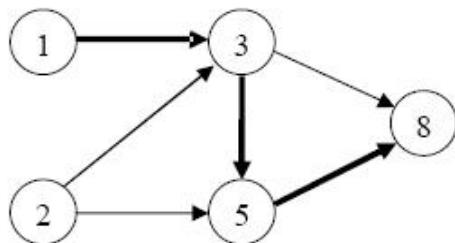


Problem 12: Lò cò

Đề thi học sinh giỏi quốc gia năm 2008

Nhảy lò cò là trò chơi dân gian của Việt Nam. Người trên hành tinh X cũng rất thích trò chơi này và họ đã cải biên trò chơi này như sau: Trên mặt phẳng vẽ n vòng tròn được đánh số từ 1 đến n. Tại vòng tròn i người ta điền số nguyên dương a_i . Hai số trên hai vòng tròn tùy ý không nhất thiết phải khác nhau. Tiếp đến người ta vẽ các mũi tên, mỗi mũi tên hướng từ một vòng tròn đến một vòng tròn khác. Quy tắc vẽ mũi tên là: Nếu có ba số a_i, a_j, a_k thỏa mãn $a_k = a_i + a_j$, thì vẽ mũi tên hướng từ vòng tròn i đến vòng tròn k và mũi tên hướng từ vòng tròn j đến vòng tròn k. Người chơi chỉ được di chuyển từ một vòng tròn đến một vòng tròn khác nếu có mũi tên xuất phát từ một trong số các vòng tròn, di chuyển theo cách mũi tên đã vẽ để đến các vòng tròn khác. Người thắng cuộc sẽ là người tìm được cách di chuyển qua nhiều vòng tròn nhất.

Ví dụ: Với 5 vòng tròn và các số trong vòng tròn là 1, 2, 8, 3, 5, trò chơi được trình bày trong hình dưới đây:



Khi đó có thể di chuyển được nhiều nhất qua 4 vòng tròn (tương ứng với đường di chuyển được tô đậm trên hình vẽ).

Yêu cầu

Hãy xác định xem trong trò chơi mô tả ở trên, nhiều nhất có thể di chuyển được qua bao nhiêu vòng tròn.

Dữ liệu

- Dòng đầu chứa số nguyên n ($3 \leq n \leq 1000$);
- Dòng thứ hai chứa số nguyên dương a_1, a_2, \dots, a_n ($a_i \leq 10^9$, $i=1, 2, \dots, n$).

Kết quả

Ghi ra số lượng vòng tròn trên đường di chuyển tìm được.

Ràng buộc

- 60% tests ứng với 60% số điểm của bài có $3 \leq n \leq 100$.

Ví dụ

Dữ liệu:

5
1 2 8 3 5

Kết quả

4

Các bạn có thể TEST trên [VNOL.INFO](#) với mã bài là **NKJUMP**



Solution:

Bài toán trên có thể đưa về đồ thị, nhưng nó có thể giải quyết hoàn toàn với DP kết hợp sắp xếp và tìm kiếm nhị phân.

Ta thấy thứ tự các số ghi trên vòng tròn không quan trọng, ta chỉ quan tâm đến giá trị ghi trên vòng tròn đó, mặt khác a_i dương với mọi i và để thỏa mãn $a_k = a_i + a_j$ thì $a_k \geq a_i$ và $a_k \geq a_j$ nên ta sẽ sắp xếp mảng a theo thứ tự tăng dần.

Gọi $L[i]$ là số vòng tròn lớn nhất trên đường di chuyển được và kết thúc tại ô i (tất nhiên sau khi sắp xếp) ta có:

$L[i] := \max(L[i], \max(L[j], L[k]))$ với $j, k \leq i$ và $j \neq k$ sao cho $a[k] + a[j] = a[i]$.

Nếu tính thăng ta phải dùng 3 vòng lặp vì vậy ta áp dụng thuật toán tìm kiếm nhị phân vì mảng a đã sắp xếp. Nếu biết i và j thì suy ra $a[k] = a[i] - a[j]$ ta sẽ tìm kiếm giá trị bày trên mảng a .

Chương trình:

```
Const
    fi='NKJUMP.INP';
    fo='NKJUMP.OUT';
    maxn=1000;

Type
    arr1 = array[0..maxn] of longint;

Var
    n      :longint;
    l,a    :arr1;
    kq    :longint;
    f      :text;

procedure nhap;
var
    i      :longint;
begin
    randomize;
    assign(f,fi);
    reset(f);
    readln(f,n);
    for i:=1 to n do read(f,a[i]);
    close(f);
end;

procedure sort(l,r:longint);{Sắp xếp mảng a theo thứ tự tăng dần}
var
    i,j      :longint;
    p,temp   :longint;
begin
    i:=l;
    j:=r;
    p:=a[l+random(r-l+1)];
    repeat
        while a[i]<p do inc(i);
        while a[j]>p do dec(j);
        if i<=j then
            begin
                if i<j then
                    begin
                        temp:=a[i];
                        a[i]:=a[j];
                        a[j]:=temp;
                    end;
                end;
            end;
    until i>j;
end;
```



```

                inc(i);dec(j);
            end;
        until i>j;
        if i<r then sort(i,r);
        if l<j then sort(l,j);
    end;

function binsearch(x,p,q:longint):longint;{Dua ra chi so cua phan tu bang x trong doan p..q}
var
    i,j,k    :longint;
begin
    if p>q then exit(0);
    i:=p;
    j:=q;
    while i<j do
        begin
            k:=(i+j) div 2;
            if a[k]<x then i:=k+1
            else j:=k;
        end;
    if a[i]=x then exit(i);
    exit(0);
end;

function max(a,b,c:longint):longint;
var
    d      :longint;
begin
    d:=a;
    if d<b then d:=b;
    if d<c then d:=c;
    exit(d);
end;

procedure solution;
var
    i,j,k    :longint;
begin
    for i:=0 to n do l[i]:=1;
    kq:=0;
    l[0]:=0;
    for i:=3 to n do
        for j:=i-1 downto 2 do
            begin
                k:=binsearch(a[i]-a[j],1,j-1);{k la chi so can tim}
                if k>0 then l[i]:=max(l[i],l[k]+1,l[j]+1);
                if kq<l[i] then kq:=l[i];
            end;
end;

procedure xuat;
begin
    assign(f,fo);
    rewrite(f);
    write(f,kq);
    close(f);
end;

begin
    nhap;
    sort(l,n);
    solution;
    xuat;
end.

```

Độ phức tạp: $O(n^2 \log n)$



Problem 13: Trò chơi với bảng số

Đề thi học sinh giỏi quốc gia năm 2009

Trò chơi với bảng số là trò chơi tham gia trúng thưởng được mô tả như sau: Có một bảng hình chữ nhật chia ra làm n ô vuông, đánh số từ trái qua phải bắt đầu từ 1. Trên ô thứ i ghi một số nguyên dương a_i , $i=1,2..,n$. Ở mỗi một lượt chơi, người tham gia trò chơi được quyền lựa chọn một số lượng tùy ý các ô trên bảng số. Giả sử theo thứ tự từ trái sang phải, người chơi lựa chọn các ô $i_1, i_2, .., i_k$. Khi đó điểm người chơi đạt được là:

$$a_{i_1} - a_{i_2} + a_{i_3} - a_{i_4} + \dots + (-1)^{k-1} a_{i_k}$$

Yêu cầu: Hãy tính số điểm lớn nhất có đạt được từ 1 lượt chơi.

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

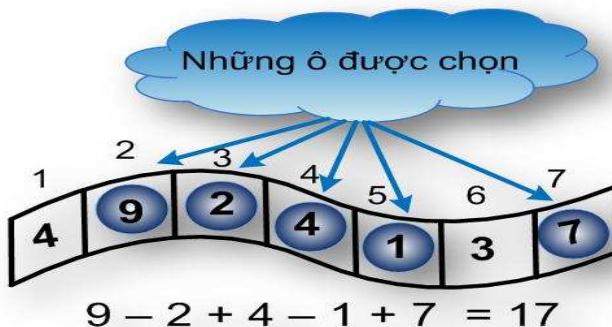
- Dòng đầu tiên chứa số nguyên dương n ($n \leq 10^6$) là số lượng ô trên bảng số.
- Dòng thứ 2 chứa n số nguyên dương $a_1, a_2, .., a_n$ ($a_i \leq 10^4, i=1,2..,n$) ghi trên bảng số

Các số liên tiếp trên một dòng cách nhau ít nhất một dấu cách.

Kết quả: Đưa ra file văn bản LINEGAME.OUT một số duy nhất là số điểm lớn nhất đạt được từ một lượt chơi.

Ví dụ:

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



Ràng buộc: 60% số test ứng với 60% số điểm của bài có $1 \leq n \leq 20$.

Solution:

Đây là một bài quy hoạch động khá hay. Trước hết ta có thể ăn được 60% số Test bằng **duyệt**, hoặc **Quy hoạch động** với độ phức tạp $O(n^2)$, thậm chí làm **Quy hoạch động** với độ phức tạp $O(n)$ mà dùng hai mảng một chiều 10^6 phần tử chúng ta cũng chỉ đạt 60% số test.

Tôi xin trình bày ngắn gọn công thức sau với độ phức tạp $O(n)$.

Gọi $F[i, 1]$ là số điểm lớn nhất có thể đạt được khi xét tới ô thứ i và ô cuối cùng mang dấu '+', tương tự $F[i, 2]$ là số điểm lớn nhất có thể đạt được khi xét tới ô thứ i và ô cuối cùng mang dấu '-' ta thấy:



$F[i,1]=\max(F[i-1,2]+a[i], F[i-1,1]).$

$F[i,2]=\max(F[i-1,1]-a[i], F[i-1,2]).$

Với công thức trên chương trình có thể chạy với $n=10^6$ trong 1s là cùng, nhưng đối với Time Limit nhỏ hơn như 0.5s thì rất dễ quá thời gian một cách đáng tiếc.

Ta để ý như sau: Tính $F[i,1]$ và $F[i,2]$ chỉ phụ thuộc vào $F[i-1,1]$ và $F[i-1,2]$ như vậy ta có thể dùng 3 biến $m1, m2, m3$ với vai trò như sau: **m1 lưu $F[i-1,1]$, m2 lưu $F[i-1,2]$, m3 là trung gian và sau khi tính xong $F[i,1]$ và $F[i,2]$ thì $m1, m2$ lại được ghi đè lên giá trị của $m1$ và $m2$ lúc trước**. Trong quá trình tính ta luôn cập nhật $m1$ và $m2$ với Max.

Trong chương trình chính ta tính luôn $m1, m2$ song song với đọc mảng a . Nói chung chương trình dưới đây chỉ sử dụng 7 biến kiểu nguyên và 1 biến tệp. Rất tiết kiệm bộ nhớ và CT ngắn gọn.

Chương trình:

```
Const
    fi='LINEGAME.INP';
    fo='LINEGAME.OUT';

Var
    n,a,i    :longint;
    m1,m2,m3:int64;
    max      :int64;
    f        :text;
Begin
    assign(f,fi);
    reset(f);
    max:=0;m1:=0;m2:=0;
    readln(f,n);
    for i:=1 to n do
        begin
            read(f,a);{Doc a[i]}
            m3:=m1;{Giu lai F[i-1,1] de tinh m2}
            if m1<m2+a then m1:=m2+a;{m1=F[i,1]=Max(F[i-1,2]+a[i],F[i-1,1])}
            if m2<m3-a then m2:=m3-a;{m2=F[i,2]=Max(F[i-1,1]+a[i],F[i-1,2])}
            if m1>max then max:=m1;{Cap nhat F[i,1] voi Max}
            if m2>max then max:=m2;{Cap nhat F[i,2] voi Max}
        end;
    close(f);
    assign(f,fo);
    rewrite(f);
    write(f,max);
    close(f);
End.
```

Độ phức tạp: $O(n)$



Problem 14: BULLS AND COWS

Mã bài: CTNBULLS VNOI.INFO

Nông dân john muốn sắp xếp n con bò(bao gồm cả bò đực và bò cái) của ông ta trên 1 hàng .Ông ta biết rằng những con bò đực rất hung hăng – Nếu 2 con bò đực đứng gần nhau thì chúng sẽ trở nên hung dữ và bắt đầu húc nhau , vì thế chúng sẽ phá hỏng mất trật tự trên hàng mà ông ta vừa sắp xếp được.

Theo kinh nghiệm lâu năm của mình , john biết được nếu ở giữa hai con bò đực có ít nhất K con bò cái thì sẽ ngăn chặn được việc chúng húc nhau (+_+)

Vì thế , ông ta nhờ các Vcoders giúp đỡ để tính toán xem có bao nhiêu cách sắp xếp đàn bò của ông ta lại sao cho không có “chiến tranh” xảy ra giữa những chú bò (^_^), (Tất cả những con bò đực và những con bò cái đều giống nhau). Hai dãy (B1..Bn) và (A1..An) được cho là khác nhau nếu tồn tại một vị trí i ($1 \leq i \leq n$) sao cho $A_i \neq B_i$

Giới hạn: $1 \leq N \leq 100000$. $0 \leq k \leq n$.

Input : gồm 1 dòng duy nhất chứa 2 số n và k cách nhau 1 dấu cách

Output: gồm duy nhất một số là kết quả theo modun 2111992

Eg:

Input :

4 2

Output

6

Giải thích Output(C=bò cái , B:bò đực)

CCCC
BCCC
CBCC
CCBC
CCCB
BCCB

Có 1/3 số test với $n \leq 15$

Solution:

Gọi $L[i,j]$ ($i=1,2..n$; $j=0..1$) số cách xếp n con bò mà con bò đứng cuối là j ($j=0$ là bò cái và $j=1$ là bò đực).

Dễ thấy: $L[1,0]:=1; L[1,1]:=1; L[0,0]:=0; L[0,1]:=0;$

Nếu $i \leq k+1$ có nghĩa trong hàng có tối đa 1 con bò đực khi đó:

$L[i,1]:=1;$
 $L[i,0]:=L[i-1,0]+L[i-1,1];$ {Nếu con bò đứng cuối là bò cái thì con bò đứng trước nó là đực hay cái đều được}



Nếu $i > k+1$ có nghĩa là trong hàng có thể có từ hai con bò đực trở lên và giữa chúng phải có ít nhất k con bò cái:

$$L[i,0]:=L[i-1,0]+L[i-1,1] \{Tương tự như trên\}$$

$$L[i,1]:=L[i-k-1,1]+L[i-k-2,1]+\dots+L[1,1]+1. (1)$$

Chứng minh (1): $L[i,1]:=L[i-k-1,1]+L[i-k-2,1]+\dots+L[1,1]+1$ giả sử trong hàng có hai con bò đực trở lên vì giữa hai con bò đực phải có ít nhất k con bò cái nên con bò thứ i là bò đực thì các con bò sau có thể là bò đực là $i-k-1, i-k-2..1$ và tất nhiên giữa hai con bò đực trên toàn là bò cái nên $L[i,1]:=L[i-k-1,1]+L[i-k-2,1]+\dots+L[1,1]$, còn nếu trong hàng chỉ có duy nhất 1 con bò đực và con bò đó ở cuối hàng nên ta có một cách nữa nên $L[i,1]:=L[i-1,1]+L[i-2,1]+\dots+L[1,1]+1$

Ta sẽ lại dùng mảng S có ý nghĩa $S[i]:=\sum_{j=1}^i L[j, 1]$ và tính $S[i]$ bằng công thức $S[i]=S[i-1]+L[i,1]$ như vậy thay vì phải tính $L[i-k-1,1]+L[i-k-2,1]+\dots+L[1,1]+1$ mất một vòng lặp thì ta sẽ có luôn $L[i,1]:=S[i-k-1]+1$;

Chương trình:

```

Const
    fi='CTNBULLS.INP';
    fo='CTNBULLS.OUT';
    maxn=100000;
    e=2111992;

Type
    arr1 =array[0..maxn+1,0..1] of int64;

Var
    n,k :longint;
    l :arr1;
    s :array[0..maxn+1] of int64;
    kq :int64;
    f :text;

Procedure nhap;
begin
    assign(f,fi);
    reset(f);
    read(f,n,k);
    close(f);
end;

Procedure solution;
var
    i :longint;
begin
    l[0,0]:=0;
    l[0,1]:=0;
    l[1,0]:=1;
    l[1,1]:=1;
    s[1]:=1;
    for i:=2 to k+1 do
        begin
            l[i,0]:=(l[i-1,0]+l[i-1,1]) mod e;
            l[i,1]:=1;
            s[i]:=s[i-1]+1;
        end;
    for i:=k+2 to n do
        begin
            l[i,0]:=(l[i-1,0]+l[i-1,1]) mod e;
            l[i,1]:=(s[i-k-1]+1) mod e;
            s[i]:=(s[i-1]+l[i,1]) mod e;
        end;
end;

```



```

kq:=(l[i,0]+l[i,1]) mod e;

end;

Procedure xuat;
begin
    assign(f,fo);
    rewrite(f);
    write(f,kq);
    close(f);
end;

begin
    nhap;
    solution;
    xuat;
end.

```

Độ phức tạp: $O(n)$

Bàn luận:

Từ các bài trên ta thấy rất nhiều bài toán **DP** sử dụng mảng cộng dồn $S[i]=\sum_{j=1}^i L[j]$ hay $S[i]$ là tổng của các số trên HCN mà cạnh trên trái là $(1,1)$ và cạnh dưới phải là (i,j) . Từ việc tính toán trước mảng S thì ta có thể rút ngắn thuật toán từ $O(n^3)$ về $O(n^2)$ hay từ $O(n^2)$ về $O(n)$. Nói chung là tiết kiệm được rất nhiều chi phí về thời gian và ta cũng chạy được những TEST lớn như **BINARY2** mà ta sẽ giới thiệu ở mục sau chẳng hạn.

Nhưng các bạn nên chú ý chỉ sử dụng mảng cộng dồn S trong các trường hợp sau:

- + Chi phí tính mảng S **nhỏ hơn chi phí tính trực tiếp theo công thức Quy hoạch động** và nếu thay mảng S vào công thức tính ta rút ngắn chi phí tính theo công thức DP như từ $O(n^2)$ về $O(n)$ chẳng hạn.
- + **Không gian lưu trữ mảng S phải không quá lớn**, như bảng S trong bài **BONUS** nếu nâng lên $n \leq 10^5$ thì mảng S vì là mảng hai chiều nên tốn bộ nhớ xấp xỉ 37GB quá lớn đối với máy tính.
- + Trong công thức **DP** ta phải tính tổng giá trị của một số đại lượng liên tiếp nhau như bài **CTNBULLS** $L[i,1]:=L[i-k-1,1]+L[i-k-2,1]+..+L[1,1]+1$, hoặc trong bài **BONUS** tổng các vuông cạnh k ô dưới phải là (i,j) .
- + Cách tính mảng S không quá phức tạp như bài: **CTNBULLS** là $S[i]=S[i-1]+L[i,1]$ và trong bài **BONUS** là $S[i,j]:=S[i-1,j]+S[i,j-1]+a[i,j]-S[i-1,j-1]$;

Chú ý khi sử dụng mảng cộng dồn:

- + Vì mảng cộng dồn S tốn thêm bộ nhớ để lưu trữ nên trong một số bài toán nhất định ta có thể dùng chính mảng sử dụng trong công thức **DP** để lưu trữ chính mảng S ví dụ bài **SPBINARY**.
- + Nếu tính $S[i]$ dựa vào **một đại lượng thay đổi** trong công thức **DP** thì $S[i]$ được tính song song với công thức **DP** ví dụ: **SPBINARY**, **CTNBULLS**.. còn nếu việc tính $S[i]$ dựa vào một **đại lượng không đổi trong suốt chương trình** biết trước như bài **BONUS** hay **BONUS2** việc tính $S[i]$ dựa vào mảng a không đổi trong suốt chương trình thì mảng S sẽ được tính sẵn từ trước coi như một phần trong thủ tục khởi tạo.



Bài tập tự luyện:

Problem 1: Rectangles Perimeter

Mã bài: MMAXPER VNOL.INFO

Cho n hình chữ nhật đánh số từ 1 đến n , các hình chữ nhật này được đặt tiếp xúc với trục Ox và nằm kề nhau từ trái qua phải theo thứ tự chỉ số. Mỗi hình chữ nhật có thể tiếp xúc với trục Ox theo bất kỳ cạnh cạnh. Cần tính độ dài lớn nhất của đường gấp phía trên.

INPUT

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

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

Ví dụ:

```
5
2 5
3 8
1 10
7 14
2 5
```

OUTPUT

Ghi ra độ dài lớn nhất tìm được

Ví dụ:

```
68
```

Problem 2: Nối mạng

Mã bài: NKCABLE VNOL.INFO

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

Dữ liệu

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



Kết quả

Ghi ra độ dài của cáp nối cần sử dụng.

Ví dụ

Dữ liệu:

6
2
2
3
2
2

Kết quả

7

Problem 3: Dãy con tăng dài nhất (bản khó)

Mã bài: LIS VNOL.INFO

Cho một dãy gồm N số nguyên ($1 \leq N \leq 30000$). Hãy tìm dãy con tăng dài nhất trong dãy đó. Các số trong phạm vi longint.

Input

- Dòng đầu tiên gồm số nguyên N.
- Dòng thứ hai gồm N số mô tả dãy.

Output In ra số lượng phần tử của dãy con.

Gồm một số nguyên duy nhất là đáp số của bài toán

Example

Input:

5
2 1 4 3 5

Output:

3

Problem 4: Dãy con dài nhất có tổng chia hết cho K

Mã bài: QBSEQ VNOL.INFO

Cho một dãy gồm n ($n \leq 1000$) số nguyên dương A_1, A_2, \dots, A_n và số nguyên dương k ($k \leq 50$). Hãy tìm dãy con gồm nhiều phần tử nhất của dãy đã cho sao cho tổng các phần tử của dãy con này chia hết cho k.

Input



Dòng đầu tiên chứa hai số n, k ghi cách nhau bởi ít nhất 1 dấu trống.

Các dòng tiếp theo chứa các số A₁, A₂, ..., A_n được ghi theo đúng thứ tự cách nhau ít nhất một dấu trống hoặc xuống dòng

Output

Gồm 1 dòng duy nhất ghi số lượng phần tử của dãy con dài nhất thỏa mãn

Example

Input:

```
10 3  
2 3 5 7  
9 6 12 7  
11 15
```

Output:

```
9
```

Problem 5: Xâu con chung dài nhất

Mã bài: QBSTR VNOL.INFO

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

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

Input

Dòng 1: chứa xâu A

Dòng 2: chứa xâu B

Output

Chỉ gồm một dòng ghi độ dài xâu C tìm được

Example

Input:

```
abc1def2ghi3  
abcdefghi123
```

Output:

```
10
```

Problem 6: Chuỗi đối xứng

Mã bài: NKPALIN VNOL.INFO

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



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

Dữ liệu vào

Gồm một dòng duy nhất chứa chuỗi s, chỉ gồm những chữ cái in thường.

Kết quả

Gồm một dòng duy nhất là một xâu con đối xứng dài nhất của xâu s. Nếu có nhiều kết quả, chỉ cần in ra một kết quả bất kỳ.

Giới hạn

Chuỗi s có độ dài không vượt quá 2000.

Ví dụ

Dữ liệu mẫu

lmevxkeyzl

Kết quả

level

Problem 7: Hội trường

Mã bài: NKREZ VNOL.INFO

Nhà trường có một phòng hội trường. Có những yêu cầu muốn sử dụng phòng hội trường này, mỗi yêu cầu cho biết thời điểm bắt đầu và thời điểm kết thúc. Nhà trường có thể chấp nhận hoặc từ chối đổi với một yêu cầu.

Yêu cầu: hãy giúp nhà trường chọn các yêu cầu sử dụng hội trường sao cho tổng thời gian hội trường được sử dụng là lớn nhất.

Dữ liệu

Dòng đầu tiên chứa một số nguyên dương n ($n \leq 10000$), số yêu cầu.

Mỗi dòng trong số n dòng tiếp theo chứa 2 số nguyên dương p và k ($0 \leq p < k \leq 30000$), mô tả một yêu cầu bắt đầu tại thời điểm p và kết thúc tại thời điểm k.

Kết quả

Gồm một dòng duy nhất là tổng thời gian lớn nhất mà hội trường được sử dụng

Ví dụ

Dữ liệu:

12

1 2

3 5



0 4
6 8
7 13
4 6
9 10
9 12
11 14
15 19
14 16
18 20

Kết quả
16

Problem 8: Đường đi có tổng lớn nhất

Mã bài: QBMAX VNOL.INFO

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

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

Input

Dòng 1: Ghi hai số m, n là số hàng và số cột của bảng.

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

Output

Gồm 1 dòng duy nhất ghi tổng lớn nhất tìm được

Example

Input:

5 7
9 -2 6 2 1 3 4
0 -1 6 7 1 3 3
8 -2 8 2 5 3 2
1 -1 6 2 1 6 1
7 -2 6 2 1 3 7

Output:

41

Problem 9: Mua vé tàu hỏa

Mã bài: QBTICKET VNOL.INFO



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

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

Khoảng cách giữa hai nhà ga (X)

Khoảng cách $0 < X \leq L_1 \rightarrow$ Giá vé C1

Khoảng cách $0 < X \leq L_2 \rightarrow$ Giá vé C2

Khoảng cách $0 < X \leq L_3 \rightarrow$ Giá vé C3

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

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

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

Input

Dòng đầu tiên ghi các số nguyên L1, L2, L3, C1, C2, C3 ($1 \leq L_1 \leq L_2 \leq L_3 \leq 10^9$; $1 \leq C_1 \leq C_2 \leq C_3 \leq 10^9$) theo đúng thứ tự liệt kê ở trên.

Dòng thứ hai chứa số lượng nhà ga N ($2 \leq N \leq 100000$)

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

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

Output

Gồm 1 dòng duy nhất ghi chi phí nhỏ nhất tìm được

Example

Input:

3 6 8 20 30 40

7

2 6

3

7

8

13

15

23



Output:

70

Problem 10: LÁT GẠCH 3

Mã bài: LATGACH3 VNOL.INFO

Đếm số cách lát hình chữ nhật $3 \times n$ bằng các domino 2×1 ? Dữ liệu vào gồm nhiều testcase kết thúc là số -1 . Mỗi testcase là một số nguyên n , $0 \leq n \leq 30$.

Với mỗi test case, in ra số nguyên là đáp số trên một dòng.

SAMPLE INPUT

2

8

12

-1

SAMPLE OUTPUT

3

153

2131

Problem 11: BLAST

Mã bài: MBLAST VNOL.INFO

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

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

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

Input

Dòng 1 chứa A, dòng 2 chứa B, chỉ gồm chữ thường a-z và số kí tự ≤ 2000 .

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

Output

Khoảng cách nhỏ nhất.

Sample

BLAST.INP

```
cmc
snmn
2
```



BLAST.OUT

10

BLAST.INP

koiv

ua

1

BLAST.OUT

5

BLAST.INP

mj

jao

4

BLAST.OUT

12



CODE:

Problem 1:

```
Const
  fi='NMAXPER.INP';
  fo='NMAXPER.OUT';
  maxn=1000;

Type
  arr1 =array[0..maxn] of longint;

Var
  n      :longint;
  a,b    :arr1;
  l      :arr1;
  p      :arr1;
  f      :text;

procedure nhap;
var
  i      :longint;
begin
  assign(f,fi);
  reset(f);
  readln(f,n);
  for i:=1 to n do readln(f,a[i],b[i]);
  close(f);
end;

function max(a,b:longint):longint;
begin
  if a>b then exit(a);
  exit(b);
end;

procedure solution;
var
  i      :longint;
begin
  l[1]:=a[1];
  p[1]:=b[1];
  for i:=2 to n do
    begin
      l[i]:=max(l[i-1]+abs(b[i]-b[i-1])+a[i],p[i-1]+abs(b[i]-a[i-1])+a[i]);
      p[i]:=max(p[i-1]+abs(a[i]-a[i-1])+b[i],l[i-1]+abs(a[i]-b[i-1])+b[i]);
    end;
end;

procedure xuat;
begin
  assign(f,fo);
  rewrite(f);
  write(f,max(l[n],p[n]));
  close(f);
end;

begin
  nhap;
  solution;
  xuat;
end.
```



Problem 2:

```
Const
    fi='NKCABLE.INP';
    fo='NKCABLE.OUT';
    maxn=25000;

Var
    n      :longint;
    a      :array[0..maxn-1] of longint;
    l,p    :array[0..maxn] of longint;
    f      :text;

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

function min(a,b:longint):longint;
begin
    if a<b then exit(a);
    exit(b);
end;

procedure solution;
var
    i,j      :longint;
begin
    l[0]:=0;
    a[0]:=0;
    l[1]:=0;
    if n>=2 then
        begin
            l[2]:=a[1];
            l[3]:=a[1]+a[2];
            for i:=4 to n do
                l[i]:=min(l[i-1]+a[i-1],l[i-2]+a[i-1]);
        end;
end;

procedure xuat;
begin
    assign(f,fo);
    rewrite(f);
    write(f,l[n]);
    close(f);
end;

begin
    nhap;
    solution;
    xuat;
end.
```

Problem 3:

```
Const
    fi='LIS.INP';
    fo='LIS.OUT';
```



```

maxn=40000;

Type
    arr1      =array[0..maxn] of longint;

Var
    n          :longint;
    a          :arr1;
    h          :arr1;
    lmax      :longint;
    f          :text;

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

function binsearch(x:longint;l:Longint):longint;
var
    i,j,k    :longint;
begin
    i:=0;
    j:=l;
    while i<>j do
        begin
            k:=(i+j) div 2;
            if a[h[k]]<x then i:=k+1
                else j:=k;
        end;
    if a[h[i]]<x then exit(i);
    exit(i-1);
end;

procedure solution;
var
    i,k      :longint;
begin
    h[0]:=0;
    a[0]:=-maxlongint;
    lmax:=0;
    for i:=1 to n do h[i]:=n+1;
    a[n+1]:=maxlongint;
    for i:=1 to n do
        begin
            k:=binsearch(a[i],lmax);
            if lmax<k+1 then lmax:=k+1;
            if a[h[k+1]]>a[i] then h[k+1]:=i;
        end;
end;

procedure xuat;
begin
    assign(f,fo);
    rewrite(f);
    write(f,lmax);
    close(f);
end;

begin
    nhap;

```



```

        solution;
        xuat;
end.

Problem 4:

Const
  fi='QBSEQ.INP';
  fo='QBSEQ.OUT';
  maxn=1000;
  maxk=50;

Type
  arr1 =array[1..maxn] of longint;
  arr2 =array[0..maxn,0..maxk] of longint;

Var
  n, k      :integer;
  a          :arr1;
  sum       :longint;
  l          :arr2;
  f          :text;

procedure nhap;
var
  i          :integer;
begin
  assign(f,fi);
  reset(f);
  readln(f,n,k);
  sum:=0;
  for i:=1 to n do
    begin
      read(f,a[i]);
      sum:=sum+a[i];
    end;
  close(f);
end;

function submod(a,b:longint):longint;
var
  c          :longint;
begin
  c:=(a-(b mod k)) mod k;
  if c<0 then c:=c+k;
  exit(c);
end;

procedure solution;
var
  i,j      :integer;
begin
  l[0,0]:=0;
  for i:=1 to k-1 do l[0,i]:=maxk;
  for i:=1 to n do
    for j:=0 to k-1 do
      begin
        l[i,j]:=l[i-1,j];
        if l[i,j]>l[i-1,submod(j,a[i])]+1 then
          l[i,j]:=l[i-1,submod(j,a[i])]+1;
      end;
end;

procedure xuat;
begin

```



```

        assign(f,fo);
        rewrite(f);
        write(f,n-1[n,sum mod k]);
        close(f);
end;

begin
    nhap;
    solution;
    xuat;
end.

```

Problem 5:

```

Const
    fi='QBSTR.INP';
    fo='QBSTR.OUT';

Var
    a,b      :array[1..1000] of char;
    m,n      :integer;
    l        :array[0..256,0..256] of integer;
    f        :text;

procedure nhap;
var
    i      :integer;
begin
    assign(f,fi);
    reset(f);
    n:=0;
    m:=0;
    while not(eoln(f)) do
        begin
            inc(m);
            read(f,a[m]);
        end;
    while not(eof(f)) do
        begin
            inc(n);
            read(f,b[n]);
        end;
    close(f);
end;

function max(a,b:integer):integer;
begin
    if a>b then exit(a);
    exit(b);
end;

procedure solution;
var
    i,j      :integer;
begin
    for i:=1 to m do l[i,0]:=0;
    for j:=0 to n do l[0,j]:=0;
    for i:=1 to m do
        for j:=1 to n do
            if a[i]=b[j] then l[i,j]:=l[i-1,j-1]+1
            else
                l[i,j]:=max(l[i-1,j],l[i,j-1]);
end;

procedure xuat;

```



```

begin
    assign(f,fo);
    rewrite(f);
    write(f,l[m,n]);
    close(f);
end;

begin
    nhap;
    solution;
    xuat;
end.

```

Problem 6:

```

Const
    fi='NKPALIN.INP';
    fo='NKPALIN.OUT';
    maxn=2000;

{$H+}
Type
    arr1 =array[0..maxn,0..maxn] of integer;

Var
    s      :string;
    s1     :string;
    k      :integer;
    l      :arr1;
    f      :text;

procedure nhap;
begin
    assign(f,fi);
    reset(f);
    readln(f,s);
    close(f);
end;

procedure latnguoc(var s:string);
var
    i      :integer;
begin
    s1:='';
    for i:=length(s) downto 1 do s1:=s1+s[i];
end;

procedure solution;
var
    i,j      :integer;
begin
    for i:=0 to length(s) do
        begin
            l[i,0]:=0;
            l[0,i]:=0;
        end;
    for i:=1 to length(s) do
        for j:=1 to length(s1) do
            if s[i]=s1[j] then l[i,j]:=l[i-1,j-1]+1
            else
                begin
                    l[i,j]:=l[i-1,j];
                    if l[i,j]<l[i,j-1] then
                        l[i,j]:=l[i,j-1];
                end;

```



```

end;

procedure ghikq(i,j:integer);
begin
    if k=0 then exit;
    if s[i]=s1[j] then
        begin
            dec(k);
            ghikq(i-1,j-1);
            write(f,s[i]);
            exit;
        end
    else
        begin
            if l[i,j]=l[i-1,j] then ghikq(i-1,j)
                else ghikq(i,j-1);
        end;
end;

procedure xuat;
begin
    assign(f,fo);
    rewrite(f);
    k:=l[length(s),length(s)];
    ghikq(length(s),length(s));
    close(f);
end;

begin
    nhap;
    latnguoc(s);
    solution;
    xuat;
end.

```

Problem 7:

```

{$MODE OBJFPC}
Const
    fi='NKREZ.INP';
    fo='NKREZ.OUT';
    maxn=10000;

Type
    arr1 =array[1..maxn] of integer;
    arr2 =array[1..maxn] of longint;

Var
    n      :longint;
    p,k   :arr1;
    l     :arr2;
    kq    :int64;
    f     :text;

{$R-}
procedure nhap;
var
    i      :longint;
begin
    assign(f,fi);
    reset(f);
    readln(f,n);
    for i:=1 to n do readln(f,p[i],k[i]);
    close(f);
end;

```



```

procedure swap(var a,b:integer);
var
    c      :integer;
begin
    c:=a;
    a:=b;
    b:=c;
end;

procedure sort(l,r:longint);inline;
var
    i,j      :longint;
    pivot   :integer;
begin
    i:=l;
    j:=r;
    pivot:=k[(i+j) div 2];
    repeat
        while pivot>k[i] do inc(i);
        while pivot<k[j] do dec(j);
        if i<=j then
            begin
                if i<j then
                    begin
                        swap(p[i],p[j]);
                        swap(k[i],k[j]);
                    end;
                inc(i);
                dec(j);
            end;
        until i>j;
        if i<r then sort(i,r);
        if l<j then sort(l,j);
    end;

procedure solution;inline;
var
    i,j      :longint;
begin
    l[1]:=k[1]-p[1];
    kq:=l[1];
    for i:=2 to n do
        begin
            l[i]:=k[i]-p[i];
            for j:=i-1 downto 1 do
                if p[i]>=k[j] then
                    if l[i]<l[j]+k[i]-p[i] then
                        l[i]:=l[j]+k[i]-p[i];
                if kq<l[i] then kq:=l[i];
        end;
end;

procedure xuat;
begin
    assign(f,fo);
    rewrite(f);
    write(f,kq);
    close(f);
end;

begin
    nhap;
    sort(1,n);
    solution;
    xuat;
end.

```



Problem 8:

```
Const
    fi='QBMAX.INP';
    fo='QBMAX.OUT';
    maxn=100;

Var
    m,n      :integer;
    a        :array[1..maxn,1..maxn] of integer;
    b        :array[0..maxn,0..maxn] of longint;
    max     :longint;
    f        :text;

Procedure nhap;
var
    i,j      :integer;
begin
    assign(f,fi);
    reset(f);
    readln(f,m,n);
    for i:=1 to m do
        for j:=1 to n do
            read(f,a[i,j]);
    close(f);
end;

function maxs(a,b:longint):longint;
begin
    if a>b then exit(a);
    exit(b);
end;

Procedure optimize;
var
    i,j      :integer;
begin
    for j:=1 to m do b[1,j]:=a[1,j];
    for j:=2 to n do
        for i:=1 to m do
            begin
                b[i,j]:=b[i,j-1]+a[i,j];
                if i>1 then b[i,j]:=maxs(b[i,j],b[i-1,j-1]+a[i,j]);
                if i<m then b[i,j]:=maxs(b[i,j],b[i+1,j-1]+a[i,j]);
            end;
    max:=-maxlongint;
    for j:=1 to m do if max< b[j,n] then max:=b[j,n];
end;

procedure xuat;
var
    i,j      :integer;
begin
    assign(f,fo);
    rewrite(f);
    write(f,max);
    close(f);
end;

begin
    nhap;
    optimize;
    xuat;
end.
```



Problem 9:

```
Const
    fi='QBTICKET.INP';
    fo='QBTICKET.OUT';
    maxn=10000;

Type
    arr1 = array[0..maxn] of qword;

Var
    N      :longint;
    l1,l2,l3:longint;
    c1,c2,c3:longint;
    a      :arr1;
    s,t    :longint;
    l      :arr1;
    kq     :qword;
    f      :text;

procedure nhap;
var
    i      :longint;
begin
    assign(f,fi);
    reset(f);
    readln(f,l1,l2,l3,c1,c2,c3);
    readln(f,n);
    readln(f,s,t);
    a[1]:=0;
    for i:=2 to n do readln(f,a[i]);
    close(f);
end;

procedure solution;
var
    i,j,k    :longint;
begin
    for i:=1 to n do l[i]:=high(qword);
    l[s]:=0;
    for i:=s+1 to t do
        for j:=i-1 downto s do
            begin
                if (a[i]-a[j]<=l1) and (l[i]>l[j]+c1)
                    then l[i]:=l[j]+c1;
                if (a[i]-a[j]<=l2) and (l[i]>l[j]+c2)
                    then l[i]:=l[j]+c2;
                if (a[i]-a[j]<=l3) and (l[i]>l[j]+c3)
                    then l[i]:=l[j]+c3;
                if a[i]-a[j]>l3 then break;
            end;
end;

procedure xuat;
begin
    assign(f,fo);
    rewrite(f);
    write(f,l[t]);
    close(f);
end;

begin
    nhap;
    solution;
    xuat;
end.
```



Problem 10:

```
Const
    fi='M3TILE.INP';
    fo='M3TILE.OUT';
    maxn=40;

Var
    n      :longint;
    L      :array[0..maxn] of qword;
    f,f1  :text;

Procedure Build;
Var
    i      :longint;
begin
    L[0]:=1;
    L[1]:=1;
    L[2]:=3;
    for i:=3 to maxn do
        if i mod 2=0 then L[i]:=l[i-2]+2*l[i-1]
        else l[i]:=l[i-1]+l[i-2];
end;

Procedure solution;
Begin
    assign(f,fi);
    reset(f);
    assign(f1,fo);
    rewrite(f1);
    n:=0;
    while n<>-1 do
        begin
            read(f,n);
            if n>-1 then
                if n mod 2=0 then writeln(f1,L[n])
                else writeln(f1,0);
        end;
    close(f);
    close(f1);
end;

begin
    Build;
    solution;
end.
```

Problem 11:

```
Const
    fi='MBLAST.INP';
    fo='MBLAST.OUT';
    maxn=2000;

Type
    arr     =array[0..maxn,0..maxn] of longint;

Var
    a,b      :ansistring;
    k        :longint;
    m,n      :longint;
    L        :arr;
    f        :text;
```



```

Procedure nhap;
begin
    assign(f,fi);
    reset(f);
    readln(f,a);
    readln(f,b);
    read(f,k);
    close(f);
end;

Procedure init;
var
    i,j      :longint;
begin
    m:=length(a);
    n:=length(b);
    for i:=0 to m do
        for j:=0 to n do
            L[i,j]:=maxlongint;
    for i:=0 to m do L[i,0]:=i*k;
    For j:=0 to n do L[0,j]:=j*k;
end;

Function Min(a,b:longint):longint;
begin
    if a<b then exit(a);
    exit(b);
end;

Procedure solution;
var
    i,j      :longint;
begin
    for i:=1 to m do
        for j:=1 to n do
            begin
                L[i,j]:=Min(L[i-1,j-1]+abs(ord(a[i])-ord(b[j])),L[i-1,j]+k);
                L[i,j]:=Min(L[i,j-1]+k,L[i,j]);
            end;
end;

Procedure xuat;
begin
    assign(f,fo);
    rewrite(f);
    write(f,l[m,n]);
    close(f);
end;

Begin
    nhap;
    init;
    solution;
    xuat;
end.

```



Selection 2: Skill of Dynamic Program lift



Problem 1: SPBINARY2

Mã bài: BINARY2 VNOL.INFO

Đề bài tương tự SPBINARY, nhưng giới hạn lớn hơn

Cho 2 số n và k ($2 \leq k \leq n \leq 10^6$)

Hãy đếm xem có bao nhiêu xâu nhị phân độ dài n mà không có quá k số 0 hoặc k số 1 liên tiếp nhau.

Input

Gồm 1 dòng duy nhất là 2 số n và k.

Output

Gồm 1 dòng duy nhất là số dãy nhị phân thỏa mãn (module 10^9).

Example

Input:

3 2

Output:

6

Solution:

Có lẽ thuật toán thì không cần nói nhiều. Vấn đề ở đây là TimeLimit quá chật 0.1s ☺. Nếu dùng phép mod chắc chắn sẽ quá thời gian vì phép mod máy tính chậm hơn cộng trừ rất nhiều, vì vậy ta thay phép mod bằng phép trừ và cộng điều này rất quan trọng trong một số bài toán.

Chương trình:

```
{$MODE OBJFPC}
Const
  fi='BINARY2.INP';
  fo='BINARY2.OUT';
  maxn=1000000;
  e=1000000000;
Var
  n,k      :longint;
  l        :array[0..maxn] of int64;
  kq      :int64;
  f        :text;

{$R-,Q-}
procedure nhap;
begin
  assign(f,fi);
  reset(f);
  readln(f,n,k);
  close(f);
end;

procedure solution;
var
```



```

i      :longint;
ik     :int64;
begin
l[0]:=0;
l[1]:=2;
ik:=2;
For i:=2 to k do
begin
    ik:=ik*2;
    if ik>e then ik:=ik-e;
    L[i]:=ik+L[i-1];
    if L[i]>e then L[i]:=L[i]-e;
end;
For i:=k+1 to n do
begin
    L[i]:=L[i-1]-L[i-k-1]+L[i-1];
    if L[i]<0 then L[i]:=L[i]+e;
    if L[i]>e then L[i]:=L[i]-e;
end;
kq:=L[n]-L[n-1];
if kq<0 then kq:=kq+e;
if kq>e then kq:=kq-e;

end;

procedure xuat;
begin
assign(f,fo);
rewrite(f);
write(f,kq);
close(f);
end;

begin
nhap;
solution;
xuat;
end.

```

Độ phức tạp: $O(n)$



Problem 2: Card Sorting

Mã bài: MCARDS VNOL.INFO

Mav có n^* c quân bài; mỗi quân bài 1 màu; và mỗi màu có n quân bài. Mav muốn sắp xếp các quân bài cùng màu nằm cạnh nhau và chúng theo thứ tự tăng dần về giá trị.



Input

Dòng đầu là 2 số C (số màu), ($1 \leq C \leq 4$), và N, số quân bài cùng màu mỗi loại, ($1 \leq N \leq 100$).

N^*C dòng tiếp theo mỗi dòng là 2 số nguyên X, Y; $1 \leq X \leq C$, $1 \leq Y \leq N$, là màu của quân bài và giá trị quân bài đó.

Thứ tự quân bài ban đầu (từ trái sang phải) chính là thứ tự dữ liệu input.

Output

Số lần chuyển bài ít nhất để thu được dãy bài được sắp theo yêu cầu trên.

Sample

MCARDS.INP

```
2 2
2 1
1 2
1 1
2 2
```

MCARDS.OUT

```
2
```

MCARDS.INP

```
3 2
3 2
2 2
1 1
3 1
```



2 1
1 2

MCARDS.OUT
2

Solution:

Các bạn hãy chú ý các quân bài cùng màu phải nằm cạnh nhau và tăng dần theo giá trị. Vì vậy tại sao chúng ta không số hóa các màu:

VD:

Giả sử dãy các quân bài thu được khi thực hiện ít lần nhất là dãy màu theo thứ tự x_1, x_2, \dots, x_k ($1 \leq k \leq 4$)
Ta gọi y_1, y_2, \dots, y_k là số tương ứng với màu trên và ta số hóa sao cho $y_1 < y_2 < \dots < y_k$.

Ta xét các con bài cùng màu thì quân bài nào có giá trị lớn hơn sẽ được xếp trước con bài có giá trị nhỏ hơn.
Như vậy ta thấy dãy số để số hóa các con bài là một dãy đơn điệu tăng dần.

Ta có số thao tác chuyển ít nhất $d=n-\text{Max}$ với Max là độ dài dãy con dài nhất tăng dần của dãy số y sau khi đã số hóa từ màu và giá trị các quân bài của đế bài.

Vấn đề còn lại là số hóa thế nào thôi. Ta dùng thuật toán quay lui duyệt tất cả các hoán vị của C màu sau đó dùng mảng hàng $gt[i]$ là số được mã hóa của màu thứ i trong hoán vị sau đó dùng thuật toán QHD tìm độ dài dãy con không giảm dài nhất.

Chương trình:

```
Const
    fi='MCARDS.INP';
    fo='MCARDS.OUT';
    maxn=100;
    maxc=4;

Type
    arr1 = array[0..maxn*maxc] of longint;

Var
    c,n :longint;
    m,y :arr1;
    a :array[0..maxn*maxc+1] of int64;
    l :arr1;
    p :array[1..4] of boolean;
    x :array[1..4] of longint;
    gt :array[1..4] of int64;
    kq :longint;
    f :text;

procedure nhap;
var
    i :longint;
begin
    assign(f,fi);
    reset(f);
    readln(f,c,n);
    for i:=1 to n*c do readln(f,m[i],y[i]);
    close(f);
end;

procedure init;
begin
```



```

kq:=maxlongint;
fillchar(p,sizeof(p),true);
end;

procedure optimize;
var
  i,j      :longint;
  max      :longint;
begin
  for i:=1 to n*c do l[i]:=1;
  l[0]:=0;
  max:=0;
  for i:=2 to n*c do
    for j:=i-1 downto 1 do
      if a[i]>=a[j] then
        if l[i]<l[j]+1 then
          l[i]:=l[j]+1;
  for i:=1 to n*c do
    if max<l[i] then max:=l[i];
  if kq>n*c-max then kq:=n*c-max;
end;

procedure best;
var
  i      :longint;
begin
  gt[x[1]]:=1;
  for i:=2 to c do gt[x[i]]:=gt[x[i-1]]*1000;
  for i:=1 to n*c do
    a[i]:=gt[m[i]]+y[i];{So hoa vao mang a}
  optimize;
end;

procedure try(i:longint);
var
  j      :longint;
begin
  for j:=1 to c do
    if p[j] then
      begin
        x[i]:=j;
        p[j]:=false;
        if i=c then best
          else try(i+1);
        p[j]:=true;
      end;
end;
end;

procedure xuat;
begin
  assign(f,fo);
  rewrite(f);
  write(f,kq);
  close(f);
end;

begin
  nhap;
  init;
  try(1);
  xuat;
end.

```

Độ phức tạp: $O(n^2C!)$



Problem 3: Lát gạch 4

Mã bài: LATGACH4 VNOL.INFO

Cho một hình chữ nhật kích thước $2 \times N$ ($1 \leq N \leq 10^9$). Hãy đếm số cách lát các viên gạch nhỏ kích thước 1×2 và 2×1 vào hình trên sao cho không có phần nào của các viên gạch nhỏ thừa ra ngoài, cũng không có vùng diện tích nào của hình chữ nhật không được lát.

Input

Gồm nhiều test, dòng đầu ghi số lượng test T ($T \leq 100$). T dòng sau mỗi dòng ghi một số N .

Output

Ghi ra T dòng là số cách lát tương ứng lấy phần dư cho 111539786.

Example

Input:

```
3  
1  
2  
3
```

Output:

```
1  
2  
3
```

Solution:

Có lẽ không cần đề cập nhiều đến thuật toán đây chính là bài LATGACH có điều $n \leq 10^9$ vì vậy thay vì tính $L[i] = L[i-1] + L[i-2]$ trực tiếp ta sẽ dùng **nhân ma trận** thực ra tính lũy thừa ma trận bằng chia để trị.

Chương trình:

```
Const  
  tfi='LATGACH4.INP';  
  tfo='LATGACH4.OUT';  
  maxn=1000000000;  
  e=111539786;  
Type  
  Matrix = array[0..1,0..1] of qword;  
Var  
  T      :longint;  
  n      :longint;  
  fi, fo :text;  
  A      :Matrix;  
  AC     :Matrix;  
  kq     :qword;  
  
Procedure Init;  
begin  
  A[0,0]:=1;A[0,1]:=1;A[1,0]:=1;A[1,1]:=0;{Khoi tao ma tran ban dau}  
end;  
  
Procedure nhap;
```



```

begin
    readln(fi,n);
end;
Function Mul_Matrix(C,B:Matrix):Matrix;{Nhan hai ma tran B va C}
var
    A      :Matrix;
    i,j,k :longint;
begin
    For i:=0 to 1 do
        for j:=0 to 1 do
            begin
                A[i,j]:=0;
                for k:=0 to 1 do A[i,j]:=(A[i,j]+B[i,k]*C[k,j]) mod e;
            end;
    exit(A);
end;
Function Matrix_n(A:Matrix;n:longint):Matrix;{Ma tran A mu n}
var
    C      :Matrix;
begin
    if n=1 then exit(A);
    if n mod 2=0 then
        begin
            C:=Matrix_n(A,n div 2);
            C:=Mul_Matrix(C,C);
            exit(C);
        end;
    if n mod 2=1 then
        begin
            C:=Matrix_n(A,n div 2);
            C:=Mul_Matrix(C,C);
            C:=Mul_Matrix(C,A);
            exit(C);
        end;
    end;
Procedure xuat;
begin
    if n>2 then
        begin
            AC:=Matrix_n(A,n);
            kq:=AC[0,0];
        end
    else kq:=n;
    writeln(fo,kq);
end;
Procedure Process;
var
    i      :longint;
begin
    assign(fi,tfi);
    reset(fi);
    assign(fo,tfo);
    rewrite(fo);
    readln(fi,T);
    for i:=1 to T do
        begin
            nhap;
            xuat;
        end;
    close(fi);
    close(fo);
end;
begin
    Init;
    process;
end.

```



Problem 4: Siêu đối xứng

Mã bài: NKSP VNOL.INFO

Một xâu có độ dài lớn hơn 1 chỉ gồm các chữ cái la tinh in thường được gọi là đối xứng, nếu ta đọc xâu đó từ trái sang phải và từ phải sang trái là như nhau. Một xâu được gọi là siêu đối xứng, nếu nó là xâu đối xứng hoặc được tạo thành bằng cách ghép liên tiếp từ nhiều xâu đối xứng.

Yêu cầu: Cho một xâu S, hãy đếm số xâu con siêu đối xứng của S.(Xâu con của một xâu S là một đoạn liên tiếp các ký tự của S)

Dữ liệu

Chứa xâu S với độ dài không vượt quá 1000.

Kết quả

Ghi ra số xâu con tìm được.

Ví dụ

Dữ liệu

abacdc

Kết quả

3

Solution:

Đây là một bài toán không khó nhưng nếu không cài đặt cẩn thận sẽ rất dễ chạy quá thời gian ta có thể phát biểu thuật toán như sau:

Gọi $L[i,j] = \text{TRUE or FALSE}$ có nghĩa là đoạn $[i,j]$ là một đoạn đối xứng hay không.

Khi xét đoạn $[i,j]$ ta có các trường hợp sau:

+ Nếu $S[i]=S[j]$ thì nếu $L[i+1,j-1]=\text{TRUE}$ thì đoạn $[i,j]$ là đối xứng.

+ Nếu $L[i,j]=\text{FALSE}$ thì nếu $[i,j]$ là đoạn đối xứng thì tồn tại k sao cho: $i \leq k \leq j$ và $L[i,k]$ và $L[k+1,j]$ đều là TRUE.

Tất nhiên nếu $L[i,j]=\text{TRUE}$ thì tăng đếm lên 1.

Chương trình:

```
Const
  fi='NKSP.INP';
  fo='NKSP.OUT';
  maxn=1000;
Type
  arr1    =array[0..maxn,0..maxn] of boolean;
Var
  n       :longint;
  l       :arr1;
  S       :ansistring;
  kq      :qword;
  f       :text;
procedure nhap;
begin
  assign(f,fi);
  reset(f);
```



```

read(f,s);
n:=length(s);
close(f);
end;

procedure solution;
var
    i,j      :longint;
    k        :longint;
    dem      :qword;
begin
    kq:=0;
    for i:=1 to n do
        for j:=1 to n do
            l[i,j]:=false;
    for i:=1 to n-1 do
        if s[i]=s[i+1] then
            begin
                l[i+1,i]:=true;
                inc(kq);
            end;
    for i:=0 to n do l[i,i]:=true;
    for i:=3 to n do
        begin
            dem:=0;
            for j:=i-2 downto 1 do
                if l[i,j]=false then
                    begin
                        if (s[j]=s[i]) and L[i+1,j-1] then
                            begin
                                inc(dem);
                                l[i,j]:=true;
                            end;
                        if l[i,j]=false then
                            begin
                                for k:=j+2 to i do
                                    if l[k-1,j] and l[i,k] then break;
                                    if (k>=j+2) and (k<i) then
                                        if l[k-1,j] and l[i,k] then
                                            begin
                                                inc(dem);
                                                l[i,j]:=true;
                                            end;
                            end;
                        end;
                    end;
            kq:=kq+dem;
        end;
    end;
procedure xuat;
begin
    assign(f,fo);
    rewrite(f);
    write(f,kq);
    close(f);
end;
begin
    nhap;
    solution;
    xuat;
end.

```

Độ phức tạp: $O(n^3)$



Problem 5: Số nhị phân có nghĩa

Mã bài: BINARY VNOL.INFO

Cho số nguyên không âm N ($N < 2^{31}$). Hãy xác định xem trong phạm vi từ 0 tới N có bao nhiêu số mà trong dạng biểu diễn nhị phân của nó có đúng K chữ số 0 có nghĩa.

Ví dụ: N = 18, K = 3 có 3 số:

1. 8 = 1000

2. 17 = 10001

3. 18 = 10010

Input

Gồm một số dòng, mỗi chứa hai số nguyên N và K cách nhau một dấu cách.

Output

Úng với mỗi bộ N, K ở Input đưa ra số lượng tìm được.

Example

Input:

18 3

8 1

Output:

3

4

Solution:

Ta làm quen với một dạng QHD mới **Quy hoạch động cấu hình**.

Gọi m là số chữ số của n khi viết trong hệ nhị phân. ($n \leq 2^{31} \Rightarrow m \leq 31$)

Chú ý các số đang xét ở đây đều hiểu là viết trong hệ nhị phân

Số bé hơn n thì có 2 khả năng :

1) Số đó có $m' < m$ chữ số :

- Chữ số đầu tiên(trái nhất) phải bằng 1

\Rightarrow k chữ số 0 phải nằm trong $m'-1$ chữ số còn lại

\Rightarrow Có $C[k, m'-1]$ số

($C[x, y]$ là tổ hợp chập x của y, tính cái này bằng quy hoạch động $C[0, i] = 1$, $C[i, j] = C[i-1, j] + C[i-1, j-1]$)

2) Số đó có m chữ số :

+ Xét lần lượt các chữ số của n từ phải qua trái

($n = x_1x_2..x_m$, số bé hơn n có dạng $y_1y_2y_3..y_m$)

Giả sử xét đến chữ số thứ i

Nếu $x_i=0$ thì $y_i=0 \rightarrow$ giảm k đi 1 đơn vị

Nếu $x_i=1$ thì $y_i=0$ hoặc $y_i=1$



Với $y_i=0$ thì các vị trí $y_{i+1}, y_{i+2} \dots y_m$ sẽ có $k-1$ chữ

số 0 => Có $C[k-1, m-i]$ số

Với $y_i=1$ thì ta xét tiếp số n

Chú ý trường hợp $k>=m$ => không có số nào thỏa mãn

Trong lúc xét số n cần phải kiểm tra xem k = 0 hay không để break.

Chương trình:

```
Const
    tfi='BINARY.INP';
    tfo='BIBARY.OUT';
    maxn=32;

Type
    arr1      =array[1..maxn] of longint;

Var
    a          :qword;
    k          :longint;
    n          :longint;
    x          :arr1;
    kq         :qword;
    C          :array[0..maxn,0..maxn] of qword;
    fi,fo     :text;

Procedure nhap;
begin
    readln(fi,a,k);
end;

Procedure Baseh(a:int64);{Doi a ra he nhi phan}
var
    y          :arr1;
    i          :longint;
begin
    n:=0;
    While a>0 do
        begin
            inc(n);
            y[n]:=a mod 2;
            a:=a div 2;
        end;
    for i:=n downto 1 do x[i]:=y[n-i+1];
end;

Procedure Calc;{Tinh C[k,n]}
var
    i,j      :longint;
begin
    For i:=0 to maxn do
        for j:=0 to maxn do
            C[i,j]:=0;
    For i:=0 to maxn do C[0,i]:=1;
    For i:=1 to maxn do
        For j:=i to maxn do
            C[i,j]:=C[i-1,j-1]+C[i,j-1];
end;

Procedure Solution;
var
    i,j      :longint;
    dem     :longint;
```



```

begin
    kq:=0;
    if n<0 then exit;
    If k=1 then kq:=1;
    if (n<=0) or (k>=n) then exit;
    dem:=0;
    For i:=1 to n do
        if x[i]=0 then inc(dem);
    if dem=k then inc(kq);
    For i:=1 to n-1 do kq:=kq+C[k,i-1];
    For i:=2 to n do
        begin
            if x[i]=0 then dec(k);
            if (x[i]=1) and (k>0) then kq:=kq+C[k-1,n-i];
            if k=0 then exit;
        end;
    end;

Procedure xuat;
begin
    Writeln(fo,kq);
end;

Procedure Process;
begin
    assign(fi,tfi);
    reset(fi);
    assign(fo,tfo);
    rewrite(fo);
    While not eof(fi) do
        begin
            nhap;
            Baseh(a);
            Solution;
            Xuat;
        end;
    close(fi);
    close(fo);
end;

Begin
    Calc;
    Process;
end.

```

Độ phức tạp: $O(n)$ với n là số chữ số của a khi chuyển sang hệ nhị phân.



Problem 6: Chặt cây

Mã bài: OPTCUT VNOI.INFO

Bạn cần chặt một thanh gỗ ra thành n đoạn, mỗi đoạn có độ dài a_i . Các đoạn được chặt phải có độ dài theo đúng thứ tự a_1, a_2, \dots, a_n từ trái sang phải.

Tại mỗi bước, bạn có thể chặt một nhát chia một thanh gỗ làm hai, và chi phí cho nhát chặt này bằng độ dài của thanh gỗ trước khi chặt.

Thứ tự chặt khác nhau sẽ cho ra tổng chi phí khác nhau khi chặt thanh gỗ thành n đoạn yêu cầu.

Ví dụ bạn cần chặt một thanh gỗ độ dài 20 ra thành 4 đoạn độ dài 3, 5, 2 và 10 theo thứ tự.

Khi chặt từ trái sang phải:

20 chặt thành 3 và 17, chi phí 20.

17 chặt thành 5 và 12, chi phí 17.

12 chặt thành 2 và 10, chi phí 12.

Tổng chi phí: 49

Khi chặt từ phải sang trái:

20 chặt thành 10 và 10, chi phí 20.

10 chặt thành 8 và 2, chi phí 10.

8 chặt thành 3 và 5, chi phí 8.

Tổng chi phí: 38

Bạn hãy tìm cách chặt có tổng chi phí nhỏ nhất.

Dữ liệu

Dòng 1: n ($1 \leq n \leq 2000$)

Dòng 2: n số nguyên dương a_1, a_2, \dots, a_n , biết rằng độ dài của thanh gỗ $a_1+a_2+\dots+a_n \leq 500000$

Kết quả

Một số nguyên duy nhất là chi phí nhỏ nhất tìm được.

Solution:

Ở đây chúng ta sẽ đề cập đến một hàm tên là *Quadrangle Inequalities* để tăng tốc độ thuật toán. Tôi xin dịch và trích toàn bộ từ tài liệu tên là *DpSpeedup* của tác giả Ngô Minh Đức.



Cho w là ma trận $n \times n$. Chúng ta cần tính bảng f bảng quy hoạch động $n \times n$ như sau:

- +) $f[i,i]=w[i,i]=0;$
- +) $f[i,j]=w[i,j]+\min(f[i,k-1]+f[k,j])$ với $i < k \leq j$. (1.1)

Tất nhiên $f[1,n]$ chính là kết quả.

Nếu sử dụng trực tiếp (1.1) thì tính mảng f mất $O(n^3)$. Chúng ta sẽ chỉ ra cách tính mảng f trong thời gian $O(n^2)$.

Dinh ly 1.1: Nếu w thỏa mãn $w[i,j]+w[i',j'] \leq w[i',j]+w[i,j']$ ($i \leq i' \leq j \leq j'$) thì ta nói w là thỏa mãn **Quadrangle Inequalities** (QI).

Dinh ly 1.2: Nếu $w[i,j] \leq w[i',j']$ ($i \leq i' \leq j \leq j'$) thì ta nói w là đơn điệu.

Dinh ly 1.3: Đặt $k[i,j]=\text{Min}(t \mid f[i,j]=w[i,j]+f[i,t-1]+f[t,j])$ với $i < j$, $k[i,i]=i$, hay nói theo một cách khác thì $k[i,j]$ là chỉ số nhỏ nhất để cực tiểu hóa (1.1).

Heth qua 1.1: Nếu w đơn điệu và thỏa mãn QI thì f cũng thỏa mãn QI.

Nên ta có $k[i,j-1] \leq k[i,j] \leq k[i+1,j]$ (*)

Dựa vào (*) ta có thể tính mảng f trong $O(n^2)$ các bạn có thể tự chứng minh kết quả này.

Quay trở lại với bài toán: Ta làm như sau:

Gọi $L[i,j]$ là chi phí nhỏ nhất khi chặt cây trong đoạn $[i,j]$.

Khi đó $L[i,j]:=a[i]+a[i+1]..+a[j]+\min(L[i,k-1]+L[k,j])$ ($i < k \leq j$) rồi để tính mảng L trong $O(n^2)$ ta áp dụng (*).

Chuong trinh:

```
Const
    fi='OPTCUT.INP';
    fo='OPTCUT.OUT';
    maxn=2000;

Type
    arr1 = array[0..maxn] of longint;
    arr2 = array[0..maxn,0..maxn] of longint;

Var
    n      :longint;
    a,s   :arr1;
    L     :arr2;
    k     :arr2;
    f     :text;

procedure nhap;
var
    i      :longint;
begin
    s[0]:=0;
    assign(f,fi);
    reset(f);
    readln(f,n);
    for i:=1 to n do
        begin
            read(f,a[i]);
            s[i]:=s[i-1]+a[i];
        end;
    close(f);
end;

function Min(a,b:int64):int64;
```



```

begin
    if a<b then exit(a);
    exit(b);
end;

Procedure solution;
var
    i,j      :longint;
    ik       :longint;
    len      :longint;
    best     :int64;
begin
    for i:=1 to n do
        begin
            k[i,i]:=i;
            l[i,i]:=0;
        end;
    for i:=1 to n-1 do
        begin
            l[i,i+1]:=a[i]+a[i+1];
            k[i,i+1]:=i+1;
        end;
    for len:=3 to n do
        for i:=1 to n-len+1 do
            begin
                j:=i+len-1;
                best:=high(longint);
                for ik:=k[i,j-1] to k[i+1,j] do
                    if best>L[i,ik-1]+l[ik,j]+s[j]-s[i-1] then
                        begin
                            best:=L[i,ik-1]+l[ik,j]+s[j]-s[i-1];
                            k[i,j]:=ik;
                        end;
                L[i,j]:=best;
            end;
    end;

Procedure xuat;
begin
    assign(f,fo);
    rewrite(f);
    write(f,l[1,n]);
    close(f);
end;

begin
    nhap;
    solution;
    xuat;
end.

```

Độ phức tạp: $O(n^2)$



Problem 7: Dãy số vòng tròn

Mã bài: PTMSEQ VNOL.INFO

Leaxtanh là cháu của nhà thông thái Anhxtanh hiện tại đang làm 1 học sinh chuyên tin. Anh đã đem lòng yêu mến cô bạn chuyên toán cùng khóa từ lâu nhưng chưa được đáp ứng. Vì là 1 học sinh chuyên toán nên cô nàng muốn thử thách leaxtanh bằng một bài toán như sau :

Trên một vòng tròn người ta đánh dấu n vị trí. Các vị trí được đánh số thứ tự từ 1 đến n theo chiều kim đồng hồ. Tại vị trí i người ta ghi số nguyên a[i] ($i=1..n$). Cần tìm cách chọn ra dãy con độ dài k liên tiếp ($0 < k < n$) trên vòng tròn (theo chiều kim đồng hồ) để tổng các số hạng trong dãy con này là lớn nhất.

Các bạn hãy nể tình Anhxtanh giúp Leaxtanh nhé.

Input

- Dòng đầu tiên ghi số n

Có 2 kiểu input

- Ghi ra các số a[i], mỗi số cách nhau 1 dấu cách.

- Ghi ra các số a[i], mỗi số trên 1 dòng.

Output

1 dòng duy nhất là tổng các số hạng của dãy tìm được

Example

Input:

7
2 -4 1 -7 4 6 -1

Output:

11

Giới hạn $|a[i]| \leq 70000$; $n \leq 1094782$;

Solution:

Gọi mini,maxi lần lượt là phần tử nhỏ nhất và lớn nhất trong mảng a.

Gọi summax,summin lần lượt là tổng các số liên tiếp trên dãy a lớn nhất và nhỏ nhất.

Gọi Sum là tổng các phần tử của dãy a.

Ta chú ý ($0 < k < n$).

Nếu maxi < 0 => **kq=maxi** vì cả dãy a các phần tử đều <0.

Nếu mini > 0 => **kq=sum-max** vì các phần tử dãy a đều >0.

Trong các trường hợp còn lại: **kq=Max(summax,sum -summin)**.



Chương trình:

```
Const
    tfi='PTMSEQ.INP';
    tfo='PTMSEQ.OUT';
Var
    min,max :int64;
    sum      :int64;
    smax     :int64;
    smin     :int64;
    n        :longint;
    s        :int64;
    k        :int64;
    kq       :int64;
    a        :int64;
    fi,fo   :text;

Procedure Solution;
var
    i        :longint;
Begin
    assign(fi,tfi);
    reset(fi);
    assign(fo,tfo);
    rewrite(fo);
    read(fi,n);
    min:=high(int64);
    max:=-high(int64);
    smax:=-high(int64);
    smin:=high(int64);
    if n<=1 then exit;
    s:=0;
    k:=0;
    for i:=1 to n do
        begin
            read(fi,a);
            sum:=sum+a;
            if min>a then min:=a;
            if max<a then max:=a;
            s:=s+a;
            if s<a then s:=s+a;
            if s>smax then smax:=s;
            k:=k+a;
            if k>a then k:=a;
            if k<smin then smin:=k;
        end;
    if min>=0 then write(fo,sum-min)
    else
        begin
            if max<0 then write(fo,max)
            else
                begin
                    kq:=smax;
                    if kq<sum-smin then kq:=sum-smin;
                    write(fo,kq);
                end;
        end;
    close(fi);
    close(fo);
end;

Begin
    solution;
end.
```

Độ phức tạp: $O(n)$



Problem 8: Xâu Fibonacci

Mã bài: LQDFIBO VNOL.INFO

Cho 2 xâu khác rỗng S_1 và S_2 có độ dài không lớn hơn 100. Xét các dãy $F_1, F_2, F_3, \dots, F_n$ trong đó

$$F_1 = S_1$$

$$F_2 = S_2$$

....

$$F_k = F_{k-1} + F_{k-2} \text{ với } k > 2$$

Cho xâu S không quá 100 ký tự và số nguyên N ($3 \leq N \leq 10^3$). Hãy xác định S xuất hiện bao nhiêu lần trong F_n .

Input

Dòng 1 chứa số nguyên N

Dòng 2 chứa xâu S_1

Dòng 3 chứa xâu S_2

Dòng 4 chứa xâu S

Output

Đưa ra một dòng chứa kết quả

Example

Input:

8

A

B

AB

Output:

8

Solution:

Gọi $L[i]$ là số lần xuất hiện của xâu S trong $F[i]$. Như vậy ta có $L[i] = L[i-2] + L[i-1] + R$ với R là số lần xuất hiện của xâu S ở cuối xâu $F[i-1]$ và đầu xâu $F[i-2]$.

Ta thấy: khi $F[i]$ đủ lớn có nghĩa là $\text{length}(F[i]) = 2 * \text{length}(S)$. Gọi Sa chứa $\text{length}(S)-1$ ký tự đầu tiên $F[i]$ và Ya chứa $\text{length}(S)-1$ ký tự cuối cùng của $F[i]$. Gọi Yb là $\text{length}(S)-1$ ký tự đầu của $F[i+1]$, Sb là $\text{length}(S)$ ký tự cuối của $F[i+1]$.

Gọi $\text{Count}(\text{Str:String}):longint$ là số lần xuất hiện của xâu S trong Str .

Như vậy ta có: $F[i]=Sa..Ya$; $F[i+1]=Yb..Sb$.

$F[i+2]=Yb..(SbSa)..Ya$.

$F[i+3]=Yb..SbSa..(YaYb)..Sb$.

$F[i+4]=Yb..SbSa..YaYb..SbYb..SbSa..(SbYb)..SbSa..Ya$.

$F[i+5]=Yb..SbSa..YaYb..SbYb..SbSa..(YaYb)..SbSa..YaYb..Sb$.

Như vậy: $L[i+2]=L[i]+L[i+1]+\text{Count}(Sb+Sa)$;

Cứ từ $F[i+3]$ trở đi $L[x]=L[x-1]+L[x-2]+\text{Count}(Ya+Yb)$; với $x \geq i+3$ và sau mỗi lần hoán đổi Ya cho Sb .



Chương trình:

```
Const
    fi='LQDFIBO.INP';
    fo='LQDFIBO.OUT';
    maxn=1000;

Type
    arr1 = array[0..maxn] of string;

Var
    n :longint;
    s,s1,s2 :string;
    L :arr1;
    f :text;

Procedure nhap;
begin
    assign(f,fi);
    reset(f);
    readln(f,n);
    readln(f,s1);
    readln(f,s2);
    readln(f,s);
    close(f);
end;

Function Count(stri:ansistring):string;
var
    dem :longint;
    kq :ansistring;
    d :string;
    i,j :longint;
begin
    dem:=0;
    i:=1;
    while i+length(s)-1<=length(stri) do
        begin
            kq:='';
            for j:=i to i+length(s)-1 do kq:=kq+stri[j];
            if kq=s then inc(dem);
            inc(i);
        end;
    str(dem,d);
    exit(d);
end;

Function add(s1,s2:string):string;
var
    i,x,y,t,nho:longint;
    kq,si :string;
begin
    while length(s1)<length(s2) do s1:='0'+s1;
    while length(s2)<length(s1) do s2:='0'+s2;
    nho:=0;
    kq:='';
    for i:=length(s1) downto 1 do
        begin
            val(s1[i],x);
            val(s2[i],y);
            t:=x+y+nho;
            str(t mod 10,si);
            kq:=si+kq;
            nho:=t div 10;
        end;
    if nho>0 then kq:='1'+kq;
```



```

        exit(kq);
end;

Procedure Swap(var a,b:ansistring);
var
    c      :ansistring;
begin
    c:=a;a:=b;b:=c;
end;

Procedure Solution;
var
    stri   :ansistring;
    Sa,Sb :ansistring;
    Ya,Yb :ansistring;
    i,j    :longint;
begin
    for i:=0 to n do L[i]:='0';
    L[1]:=Count(s1);
    L[2]:=Count(s2);
    i:=3;
    while (i<n+1) and (length(s1)+length(s2)<2*length(s)) do
        begin
            stri:=s2+s1;s1:=s2;s2:=stri;
            inc(i);
        end;
    stri:=s2+s1;s1:=s2;s2:=stri;j:=i-1;
    if length(s)>1 then { Tim Sa,Ya,Sb,Yb}
        begin
            inc(j);
            L[j]:=Count(stri);
            Sa:=Copy(stri,1,length(s)-1);
            Ya:=Copy(stri,length(stri)-length(s)+2,length(s)-1);
            inc(j);
            stri:=s2+s1;s1:=s2;s2:=stri;
            L[j]:=Count(stri);
            Yb:=Copy(stri,1,length(s)-1);
            Sb:=Copy(stri,length(stri)-length(s)+2,length(s)-1);
            inc(j);stri:=s2+s1;
            L[j]:=Count(stri);
        end;
    For i:=j+1 to n do
        begin
            L[i]:=add(L[i-2],L[i-1]);
            L[i]:=add(L[i],Count(Ya+Yb));
            swap(Ya,Sb);{Hoan doi Ya cho Sb}
        end;
end;

Procedure Xuat;
begin
    assign(f,fo);
    rewrite(f);
    write(f,L[n]);
    close(f);
end;

Begin
    Nhap;
    Solution;
    Xuat;
End.

```

Độ phức tạp: **O(nk)** với k là chi phí xử lí số nguyên lớn và chi phí cho hàm Count.



Bàn luận:

Nói chung ở phần 2 ta làm quen với một số bài toán QHĐ ở mức cao hơn. Một số dạng QHĐ mới QHĐ cấu hình và quan sát một số kĩ thuật tăng tốc thuật toán và các Solution tinh tế. Như ta đã thấy hàm QI mang lại cải thiện đáng kể cho QHĐ chúng ta nên ghi nhớ hàm này và cách dùng của nó.

Ở phần 3 tôi sẽ giới thiệu với các bạn các cấu trúc dữ liệu nâng cao, các truy vấn thường gặp và ứng dụng của chúng trong các thuật toán Quy hoạch động như: [Interval Tree\(IT\)](#), [Binary Index Tree \(BIT\)](#), [Binary Search Tree \(BST\)](#), [Priority Queue](#) (Heap)... Các bài toán như: Tìm Min,Max của tập hợp động, Range Maximum Query (RMQ)...



Bài tập tự luyện:

Problem 1: Tiền bạc luôn là thứ quý giá

Mã bài: DTGAME VNOL.INFO

Tiền bạc vẫn luôn là thứ có giá trị đối với mỗi con người, kể cả với *pirate*. Vì vậy, khi hòn đảo xinh đẹp của tên cướp biển khét tiếng này sắp bị... giải tỏa, hắn đã tranh thủ vơ vét cho đến đồng tiền vàng cuối cùng. Trên hòn đảo có N mỏ vàng nằm cạnh nhau trên một đường thẳng, đánh số từ 1 đến N. Mỏ vàng i có giá trị là p_i . Đội thi công sẽ có nhiệm vụ san lấp hết N mỏ vàng này. Nhưng tại mỗi thời điểm, *pirate* cố gắng ngăn cản việc san lấp bằng cách xây một bức tường ngăn cách hai mỏ vàng liên tiếp nào đó, vậy là cụm mỏ còn lại được chia làm 2 phần. Đội thi công sẽ chọn một trong hai phần đó để san lấp hết, thế là *pirate* giữ lại được phần còn lại và hắn sẽ nhận được số đồng tiền vàng đúng bằng tổng giá trị của những mỏ còn lại đó. Công việc cứ tiếp tục cho đến khi chỉ còn lại một mỏ vàng duy nhất. Đội thi công biết điều đó, cho nên họ đã có chiến thuật san lấp để cực tiểu hóa số tiền của *pirate*. Với lòng tham không đáy, *pirate* quyết tâm lấy càng nhiều vàng càng tốt. Hãy giúp con người khốn khổ vì tiền này !

Input

Dữ liệu vào gồm nhiều dòng:

- Dòng 1: Một số nguyên dương N ($1 \leq n \leq 2000$).
- Dòng 2...n+1: Mỗi dòng ghi giá trị p_i của từng mỏ vàng theo thứ tự từ 1 đến n ($1 \leq p_i \leq 1000$).

Output

Dữ liệu ra gồm 1 dòng duy nhất ghi ra số đồng tiền vàng lớn nhất có thể vơ vét được.

Example

Input:

5
8
6
2
4
2

Output:

Giải thích: Đầu tiên *pirate* chia mỏ vàng thành 2 phần [1,2] và [3,4,5]. Xe ui sẽ san lấp phần [1,2] và *pirate* nhận được $p_2 + p_3 + p_4 = 8$ đồng tiền vàng. Tiếp theo chia [3,4,5] thành [3] và [4,5]. Xe ui san lấp [4,5] và *pirate* thu thêm $p_2 = 2$ đồng tiền vàng. Công việc kết thúc vì chỉ còn 1 mỏ, vậy tổng cộng thu được là $8 + 2 = 10$ đồng tiền vàng. Không có cách nào giúp hắn thu thêm tiền.

Problem 2: Xâu Fibonacci 2

Mã bài: LQDFIBO2 VNOL.INFO

Cho 2 xâu khác rỗng S_1 và S_2 có độ dài không lớn hơn 100. Xét các dãy $F_1, F_2, F_3, \dots, F_n$ trong đó

$$\begin{aligned}F_1 &= S_1 \\F_2 &= S_2\end{aligned}$$



....

$F_k = F_{k-1} + F_{k-2}$ với $k > 2$

Cho xâu S không quá 100 ký tự và số nguyên N ($3 \leq N \leq 10^9$). Hãy xác định S xuất hiện bao nhiêu lần trong F_n sau khi modun 15111992

Input

Dòng 1 chứa số nguyên N

Dòng 2 chứa xâu S_1

Dòng 3 chứa xâu S_2

Dòng 4 chứa xâu S

Output

Đưa ra một dòng chứa kết quả

Example

Input:

8

A

B

AB

Output:

8

Problem 3: Hình chữ nhật 0 1

Mã bài: QBRECT VNOI.INFO

Cho một bảng kích thước $M \times N$, được chia thành lưới ô vuông đơn vị M dòng N cột ($1 \leq M, N \leq 1000$)

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

Yêu cầu:

Hãy tìm một hình chữ nhật gồm các ô của bảng thỏa mãn các điều kiện sau:

1 - Hình chữ nhật đó chỉ gồm các số 1

2 - Cạnh hình chữ nhật song song với cạnh bảng

3 - Diện tích hình chữ nhật là lớn nhất có thể

Input



Dòng 1: Ghi hai số M, N

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

Output

Gồm 1 dòng duy nhất ghi diện tích của hình chữ nhật tìm được x

Example

Input:

```
11 13
0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 1 1 1 0 0 0 0
0 1 1 1 1 1 1 1 1 1 0 0 0
1 1 1 1 1 1 1 1 1 1 1 0 0
0 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 1 1 1 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 1 1
0 0 0 0 0 1 0 0 0 0 0 1 1
```

Output:

49

Problem 4: Hình vuông 0 1

[Mã bài: QBSQUARE VNOL.INFO](#)

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

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

Yêu cầu:

Hãy tìm một hình vuông gồm các ô của bảng thỏa mãn các điều kiện sau:

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

2 - Cạnh hình vuông song song với cạnh bảng.

3 - Kích thước hình vuông là lớn nhất có thể

Input

Dòng 1: Ghi hai số m, n



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

Output

Gồm 1 dòng duy nhất ghi kích thước cạnh của hình vuông tìm được

Example

Input:

```
11 13
0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 1 1 1 0 0 0 0
0 1 1 1 1 1 1 1 1 0 0 0 0
1 1 1 1 1 1 1 1 1 1 0 0 0
0 1 1 1 1 1 1 1 1 1 0 0 0
0 0 1 1 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 1 1
0 0 0 0 1 0 0 0 0 0 1 1
```

Output:

```
7
```

Problem 5: Đếm hình chữ nhật trên bảng 0-1

Mã bài: CREC01 VNOL.INFO

Cho một bảng ô vuông kích thước $M \times N$. Mỗi ô của bảng chứa một số 0 hoặc 1. Hãy đếm số hình chữ nhật con của bảng mà có các cạnh song song với các cạnh của bảng và gồm toàn số 1.

Input

Dòng đầu chứa hai số nguyên M, N . ($1 \leq M, N \leq 1000$)

M dòng sau, mỗi dòng chứa N kí tự 0/1.

Output

In ra số lượng hình chữ nhật thỏa mãn.

Example

Input:

```
4 3
111
101
111
001
```

Output:

```
24
```



Problem 6: Đếm hình chữ nhật

Mã bài: LQDRECT VNOL.INFO

Cho một bảng kích thước $M \times N$, được chia thành lưới ô vuông đơn vị M dòng N cột.
($1 \leq M \leq 1000; 1 \leq N \leq 300$)

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

Yêu cầu:

Đếm số hình chữ nhật gồm các ô của bảng thỏa mãn các điều kiện sau:

1 - Hình chữ nhật đó có 4 ô ở 4 đỉnh là 4 ô khác nhau

2- 4 ô ở đỉnh đều là số 1

3- Cạnh hình chữ nhật song song với cạnh bảng

Input

Dòng 1: Ghi hai số M, N

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

Output

Gồm 1 dòng duy nhất ghi số hình chữ nhật thỏa mãn yêu cầu .

Example

Input:

```
4 4
1 0 0 1
0 1 1 1
1 1 1 1
1 1 1 1
```

Output:

```
14
```

Problem 7: Dãy số

Mã bài: NKSEQ VNOL.INFO

Cho dãy số nguyên a_1, a_2, \dots, a_n ($1 \leq n \leq 100000$), mỗi số không vượt quá 10000. Dãy số này được viết trên một vòng tròn. Nghĩa là, khi cắt vòng tròn tại vị trí j , ta thu được:

$a_j, a_{j+1}, \dots, a_n, a_1, a_2, \dots, a_{j-1}$

Vị trí j được gọi là vị trí tốt, nếu các điều kiện sau đây được thỏa mãn:



- $a_j > 0$
- $a_j + a_{j+1} > 0$
- ...
- $a_j + a_{j+1} + \dots + a_n > 0$
- $a_j + a_{j+1} + \dots + a_n + a_1 > 0$
- ...
- $a_j + a_{j+1} + \dots + a_n + a_1 + a_2 + \dots + a_{j-2} > 0$
- $a_j + a_{j+1} + \dots + a_n + a_1 + a_2 + \dots + a_{j-2} + a_{j-1} > 0$

Yêu cầu: hãy đếm số vị trí tốt.

Dữ liệu vào

- Dòng đầu tiên chứa số nguyên n.
- Dòng thứ 2 chứa dãy số a₁, a₂,...,a_n.

Kết quả

In ra 1 số nguyên duy nhất là số vị trí tốt.

Ví dụ

Dữ liệu mẫu

```
5
0 1 -2 10 3
```

Kết quả

```
2
```

Problem 8: Đếm chuỗi đối xứng

Mã bài: QBPAL VNOL.INFO

Trong một buổi học viết chữ, Bòm phát hiện trong một số từ khi bỏ đi một số ký tự thì đọc ngược hay đọc xuôi đều giống nhau. Bòm cảm thấy thú vị, và cậu tiếp tục thử xóa các ký tự khác, kết quả là có thêm nhiều từ đối xứng nữa: II, I, O, C... Nhưng nếu với một từ dài, cứ thử từng cách xóa như vậy thì thật mất thời gian. Bạn hãy viết chương trình giúp Bòm tính số cách xóa sao cho từ thu được đối xứng. Hai cách xóa chỉ khác nhau bởi thứ tự xóa các ký tự thì coi như trùng nhau.

Input

Một dòng duy nhất là từ cần tính số cách xóa, từ này chỉ chứa các chữ cái in hoa A, B, .., Z. (Độ dài từ không quá 120)

Output

Một số duy nhất là số cách xóa.

Example

Input:
IOICAMP

Output:

9



Selection 3: Special Data Structure



A.Stack and Queue

Stack và **Queue** là hai kiểu cấu trúc dữ liệu trùu tượng có lẽ là dễ hiểu và cài đặt đơn giản nhất, nhưng ứng dụng của chúng lại không hề nhỏ.

Hầu hết chúng được sử dụng trong các bài toán đồ thị, ngoài ra chúng còn một số ứng dụng trong Quy hoạch động, kết hợp để làm giải độ phức tạp. . .

Dưới đây chúng ta sẽ xét những bài toán về **Stack** và **Queue** điển hình và có những solution tinh tế.

Trước khi nói về vấn đề này chúng ta sẽ đề cập tới một loại **Queue** mới đó là **Double Ended Queue** (Hàng đợi hai đầu, trang bị thêm hai thao tác nữa là lấy phần tử ở cuối hàng đợi, và đẩy một phần tử vào đầu hàng đợi) và **Priority Queue** (Hàng đợi có độ ưu tiên- **Heap**).



Problem 1: Mass of Molecule

Duyên hải Bắc Bộ 2008

Hóa chất chỉ gồm các nguyên tố C, H, O có trọng lượng 12, 1, 16 tương ứng.

Nó được biểu diễn dạng "nén", ví dụ COOHHH là CO₂H₃ hay CH(COOH)(COOH) là CH(CO₂H)₃. Nếu ở dạng nén thì số lần lặp ≥ 2 và ≤ 9 .

Tính khối lượng hóa chất.

Input

Gồm một dòng mô tả hóa chất không quá 1000 kí tự chỉ gồm C, H, O, (,), 2,...,9.

Output

Khối lượng của hóa chất, luôn ≤ 100000 .

Sample

MASS.INP

COOH

MASS.OUT

45

MASS.INP

CH(CO₂H)₃

MASS.OUT

148

MASS.INP

((CH)₂(OH₂H)(C(H))O)₃

MASS.OUT

222

Solution:

Đây là một bài khá dễ giải bằng Stack đơn thuần chỉ cần chú ý một chút là được.

Vì đây có cả dấu '(' và ')' nên ta để Element có kiểu **AnsiString**;

Ta sẽ làm như sau:



- + Đọc INPUT vào chuỗi kí tự S.
- + Duyệt i lần lượt từ 1 đến Length(S) với mỗi S[i] ta xét như sau:
 - + Nếu $S[i] \in \{C, H, O\}$ thì ta đẩy {12, 1, 16} vào Stack
 - + Nếu $S[i] = 2, 3..9$ thì ta làm như sau:
 - Đổi $S[i]$ ra số gán vào y.
 - Lấy một phần tử ra khỏi Stack giả sử gán vào x.
 - Đổi x ra số gán vào k sau đó gán $k := y * k$;
 - Đẩy k ra chuỗi gán vào x. Đẩy x vào Stack.
 - + Nếu $S[i] =)$ thì :
 - $k := 0$;
 - Lấy một phần tử ra khỏi Stack giả sử gán vào x.
 - Đổi x ra số gán vào y sau đó $k := k + y$;
 - Lặp đi lặp lại các công việc trên đến khi nào phần tử lấy ra là '(' thì thôi.
 - Đổi k ra chuỗi gán vào x rồi đẩy x vào Stack.
 - + Nếu $S[i] = ('$ thì đẩy $S[i]$ vào Stack.
- + Việc đơn giản còn lại chỉ là lần lượt lấy các phần tử ra khỏi Stack đến khi nào Stack rỗng thì thôi và tính tổng của chúng, tổng đó chính là đáp án của bài toán.

VD: $S = 'C(OH)2'$;

$i := 1$; Stack = (); $S[1] = 'C'$ đẩy '12' vào Stack.

$i := 2$; Stack = ('12'); $S[2] = '('$ đẩy '(' vào Stack.

$i := 3$; Stack = ('12', '('); $S[3] = 'O'$ đẩy '16' vào Stack.

$i := 4$; Stack = ('12', '(', '16'); $S[4] = 'H'$ đẩy '1' vào Stack.

$i := 5$; Stack = ('12', '(', '16', '1'); $S[5] =)'$ thực hiện:

+ sum := 0;

+ Lấy Stack[Top] ra khỏi Stack: $y := 1$; Top := Top - 1;

+ $k := k + y = 1$;

+ Lấy Stack[Top] ra khỏi Stack gán vào x, $x < > '()$ nên $a := 16$;

+ $k := k + y = 17$; Top := Top - 1;

+ Lấy Stack[Top] ra khỏi Stack gán vào x, $x = '()$ nên dừng lại.

+ Đẩy k vào Stack.

$i := 6$; Stack = ('12', '17'); $S[6] = '2'$; lấy Stack[Top] ra khỏi Stack gán vào x; Đổi x, Stack[Top] ra số gán vào k và y ; $k := y * k$; Đẩy k vào Stack.

Stack = ('12', '34');

Kq = 12 + 34 = 46;

Vậy kết quả cần tìm là 46.

Thực ra bài toán trên chỉ khó ở phần xử lý chuỗi '(...)' và '...A' với A = 2..9;

Nếu chuỗi có dạng '(...)' thì ta chỉ việc tính tổng khối lượng các phần tử trong dấu ngoặc rồi đẩy nó vào trong Stack làm như vậy khi trường '(...A)' với A = 2..9 chúng ta coi '(...)' như là một nguyên tố có khối lượng là Stack[Top] và xử lý chung là lấy một phần tử ở Top của Stack ra và nhân với hệ số A rồi lại đẩy vào Stack.

Bạn có thể Test bài này trên [VNOI.INFO](#) với mã bài là: **MMASS**



Chương trình:

```
Const
    fi='MASS.INP';
    fo='MASS.OUT';
    Maxn = 1000;
    Number =['2','3','4','5','6','7','8','9'];

Type
    Element      =AnsiString;
    TStack       =Array[0..Maxn] of Element;

Var
    Stack     :TStack;
    Top      :longint;
    kq       :longint;
    s        :Ansistring;
    f        :text;

Procedure Nhap;
begin
    assign(f,fi);
    reset(f);
    read(f,s);
    close(f);
end;

Procedure Init;
begin
    Top:=0;
end;

Function Is_Empty:boolean;
begin
    Is_Empty:=Top=0;
end;

Function Is_Full:boolean;
begin
    Is_Full:=Top=Maxn;
end;

Procedure PushS(x:Ansistring);
begin
    if not Is_Full then
        begin
            inc(Top);
            Stack[Top]:=x;
        end;
end;

Procedure PopS(var x:Ansistring);
begin
    if not Is_Empty then
        begin
            x:=Stack[Top];
            dec(Top);
        end;
end;

Procedure Solution;
var
    i          :longint;
    x          :string;
    y          :longint;
    k          :longint;
```



```

begin
    kq:=0;
    for i:=1 to length(s) do
        begin
            if s[i]='H' then PushS('1');
            if s[i]='O' then PushS('16');
            if s[i]='C' then PushS('12');
            if s[i]='(' then PushS(s[i]);
            if s[i]=')' then
                begin
                    k:=0;
                    while Stack[Top]<>'(' do
                        begin
                            PopS(x);
                            val(x,y);
                            k:=k+y;
                        end;
                    PopS(x);
                    str(k,x);
                    PushS(x);
                end;
            if s[i] in Number then
                begin
                    val(s[i],y);
                    PopS(x);
                    val(x,k);
                    k:=k*y;
                    str(k,x);
                    PushS(x);
                end;
            end;
        kq:=0;
        While Top>0 do
            begin
                PopS(x);
                val(x,y);
                kq:=kq+y;
            end;
    end;

Procedure Xuat;
begin
    assign(f,fo);
    rewrite(f);
    write(f,kq);
    close(f);
end;

begin
    Nhap;
    Init;
    Solution;
    Xuat;
end.

```



Problem 2: Những con đường quanh nông trang

Mã bài: VRATF VNOI.INFO

Các con bò của nông dân John có sở thích là hay đi khám phá những vùng xung quanh nông trang. Ban đầu, tất cả N ($1 \leq N \leq 1,000,000,000$) con bò tập trung thành 1 nhóm và cùng bắt đầu chuyến đi trên 1 con đường. Cho tới khi gặp một ngã ba đường thì chúng đôi khi chọn cách chia làm 2 nhóm nhỏ hơn (mỗi nhóm ít nhất 1 bò) và mỗi nhóm lại tiếp tục hành trình trên con đường của nhóm chúng. Khi một trong những nhóm này gặp 1 ngã ba khác thì nhóm này lại có thể tách ra tiếp, và cứ như vậy.

Các con bò đã hình thành nên 1 quy tắc về việc chia nhóm như sau: nếu chúng có thể chia thành 2 nhóm mà chênh lệch số bò của 2 nhóm là đúng bằng K ($1 \leq K \leq 1000$) thì tại ngã ba đó chúng sẽ chia làm 2; nếu không thì chúng sẽ dừng cuộc hành trình và đứng ở đó nhẩm nháp cỏ non.

Giả sử rằng luôn có những ngã ba mới trên các con đường, hãy tính xem cuối cùng có bao nhiêu nhóm bò tất cả.

Dữ liệu

- Dòng 1: 2 số nguyên cách nhau bởi dấu cách: N và K

Kết quả

- Dòng 1: Một số nguyên cho biết số lượng nhóm bò sau cùng.

Ví dụ

Dữ liệu

6 2

Giải thích:

Có 6 con bò và độ chênh lệch khi xét chia nhóm là 2.

Kết quả

3

Giải thích:

Cuối cùng có 3 nhóm bò (1 nhóm có 2 bò, 1 nhóm có 1 và 1 nhóm có 3).

6
/\
2 4
/\
1 3

Solution:

Bài trên các bạn có thể dùng đệ quy để giải một cách hữu hiệu, nhưng chúng ta đang học Stack và Queue nên tôi xin trình bày cách sử dụng hàng đợi để giải quyết bài này:



Gọi kq là kết quả cần tìm. Ban đầu khởi tạo Queue có duy nhất một phần tử đó là số bò ban đầu và kq:=0;

Thực hiện các công việc sau chừng nào Queue rỗng thì thôi:

+Lấy một phần tử ra khỏi hàng đợi gán vào n.

+ Nếu $((n-k) \bmod 2=1)$ or $(n \leq k)$ thì tăng kq lên 1: $kq:=kq+1$; { Khi $n \leq k$ thì có nghĩa không thể chia được nữa, $(n-k) \bmod 2 = 1$ có nghĩa là không thể chia thành hai nhóm mà chênh lệch số lượng đúng bằng k, khi đó chúng sẽ dừng lại và gãm cỏ, khi đó ta có thêm một nhóm mới nên $kq:=kq+1$ }

+ Trái lại thì ta có thể chia thành hai nhóm mà chênh lệch số lượng đúng bằng k, như vậy nhóm một sẽ có $a=(n-k) \bmod 2$ con bò, nhóm 2 sẽ có $b=n-a$ con bò.Ta đẩy a và b vào hàng đợi.

Note: Tuy $n \leq 10^9$ nhưng ta chỉ cần khai báo số phần tử tối đa của Queue là 10000 là quá đủ rồi.

Chương trình: Đây là một bài toán khá đơn giản vì vậy tôi chỉ viết thủ tục chính các thủ tục còn lại các bạn tự viết.

```
Procedure solution;
Var
    a,b      :longint;
Begin
    PushQ(n);
    kq:=0;
    {kq và n là biến khai báo o chương trình chính}
    While not Empty do
        Begin
            PopQ(n);
            if (n mod 2=1) or (n<=k) then inc(kq)
                else
                    Begin
                        a:=(n-k) div 2;
                        b:=n-a;
                        PushQ(a);
                        PushQ(b);
                    End;
        End;
End;
```

Tôi xin giới thiệu các bạn thủ tục đệ quy như sau: Hàm Find($n:\text{longint}$): longint trả về số nhóm bò cuối cùng khi có n con bò:

```
Function find(n:longint):longint;
Begin
    if (n mod 2=1) or (n<=k) then exit(1)
        else
            find:=find((n-k) div 2)+find((n-k) div 2+k);
End;
```

Ta thấy thủ tục đệ quy trên khá ngắn gọn tuy với $n \leq 1000000000$ nó chạy tương đối nhanh nhưng khi mở rộng ra nó tỏ ra chậm hơn nhiều so với cách sử dụng hàng đợi. Tuy nhiên khi thi tôi khuyên nên sử dụng một cách hợp lý các thuật toán, không nhưng tốc độ nhanh mà còn đòi hỏi cài đặt càng đơn giản càng tốt.Nếu là tôi, tôi sẽ chọn cách đệ quy ☺.



Problem 3: Huyền thoại Lục Vân Tiên

Mã bài: MINK VNOL.INFO

Lục Vân Tiên cũng giống Samurai Jack , bị Quan Thái Sư đẩy vào vòng xoáy thời gian và bị chuyển tới tương lai của những năm 2193 .

Ở thời đại này , Tráng sỹ phải là người thông thạo máy tính , gõ bàn phím lia lịa như đấu sỹ thời xưa múa kiếm ấy và phải qua một cuộc thi lập trình mới được phong danh hiệu .

Để vượt qua vòng loại , Vân Tiên cần tham gia cuộc thi sát hạch . Ban Giám Khảo cuộc thi sát hạch gồm có N người , họ đều là các cao thủ trong giới IT . Các thành viên trong Ban Giám Khảo được đánh số từ 1 -> N và mỗi người lại có một chỉ số sức mạnh gọi là APM (Actions Per Minute) . Các giám khảo sẽ xếp hàng lần lượt từ 1 -> N . Mỗi thí sinh sẽ phải đấu với K vị giám khảo và K vị giám khảo này phải đứng liền thành 1 đoạn (Tức là i , i+1 , i+2 , ... i+K-1) , chỉ cần thắng 1 vị giám khảo thì sẽ vượt qua vòng loại .

Tuy nhiên thí sinh không được chọn xem những giám khảo nào sẽ đấu với mình .

Vân Tiên rất lo vì lỡ may đụng độ với những vị giám khảo nào "khó nhằn" thì sẽ tiêu mất . Nên chiến thuật của Vân Tiên là tập trung hạ vị giám khảo có chỉ số APM thấp nhất trong số K vị . Bạn hãy lập trình để giúp Lục Vân Tiên xác định được ở tất cả các phương án thi chỉ số APM của vị giám khảo thấp nhất sẽ là bao nhiêu

Có tất cả $N-k+1$ phương án :

Phương án 1 : Vân Tiên phải đấu với vị 1 -> vị k

Phương án 2 : Vân Tiên phải đấu với vị 2 -> vị k+1

...

Phương án $N-k+1$: Vân Tiên phải đấu với vị $N-k+1$ -> vị N .

($1 \leq N \leq 17000$, chỉ số APM của 1 giám khảo ≥ 1 và ≤ 2 tỉ , $1 \leq K \leq N$) .

Input

Dòng 1 : số T là số test .

Tiếp theo là T bộ test , mỗi bộ test có format như sau :

Dòng 1 : N k

Dòng 2 : N số nguyên dương $A[1]$, ... $A[N]$.

Output

Kết quả mỗi test ghi ra trên dòng , dòng thứ i gồm $N-k+1$ số , số thứ j tương ứng là chỉ số APM của vị giám khảo yếu nhất trong phương án j .

Example

Input:

2

4 2

3 2 4 1

3 3

1 2 3

Output:

2 2 1

1



Solution:

Ta có thể phát biểu bài toán trên ngắn gọn như sau:

Cho một dãy số gồm n phần tử $a[1..n]$ và một số nguyên dương k. Tìm và in ra theo câu sau:

```
Min(a[1],a[2],a[3]..a[k]);  
Min(a[2],a[3],a[4]..a[k+1]);  
Min(a[3],a[4],a[5]..a[k+2]);  
...  
Min(a[n-k+1],a[n-k+2],..a[n]);
```

Và tất nhiên cũng có thể là tìm Max. Có thể nói đây là một bài toán rất hay về việc sử dụng Queue và Stack để tìm Min Max trong một đoạn tịnh tiến với thời gian là $O(n)$.

Lời giải quen thuộc và đơn giản nhất:

```
For i:=1 to n-k+1 do  
  Begin  
    min:=a[i];  
    for j:=i+1 to i+k-1 do  
      if min>a[j] then min:=a[j];  
    writeln(f,min);{Min(a[i],a[i+1],...,a[i+k-1])}  
  End;
```

Với thuật toán trên ta có độ phức tạp $O(n.k)$ không thể chạy với đề bài trên.

Tôi xin trình bày ngắn gọn tư tưởng thuật toán sử dụng **DQueue** (Double Ended Queue) như chúng ta đã trình bày ở trên như sau:Ở đây nếu chỉ nói là đẩy một phần tử vào Queue ta hiểu là thao tác PushR tức là đẩy vào cuối Queue.

Trước tiên ta khai báo kiểu dữ liệu cho một phần tử là một bản ghi (**record**) gồm 2 trường: value: giá trị của một phần tử của Queue, cs: Chỉ số của phần tử ở mảng a được đưa vào Queue.

Ta duy trì Queue như sau: Giả sử xét đến phần tử thứ i là $a[i]$:

+ Lấy ra một phần tử của Queue gán vào x.Nếu $x \leq a[i]$ thì ta đẩy x và $a[i]$ vào Queue

+ Trái lại tức là $x > a[i]$ ta lặp lại công việc lấy ra một phần tử của Queue đến khi nào Queue rỗng hoặc gặp một phần tử nhỏ hơn hoặc bằng $a[i]$ thì thôi.Sau đó chúng ta lại đẩy $a[i]$ vào Queue.

+ Tại sao chúng ta lại làm vậy: Ta nhận xét nếu làm như trên thì Queue luôn duy trì ở trạng thái là một dãy tăng dần xét các phần tử từ Head đến Top về giá trị (value), mặt khác vẫn đảm bảo trong Queue các phần tử luôn có chỉ số tăng dần từ Head đến Top. Như vậy phần tử nhỏ nhất trong Queue chính là phần tử ở đầu Queue hay chính là $Queue[Head].value$.Và $\min(a[i-k+1],a[i-k+2]..a[i])$ là **phần tử Queue[Head].cs của mảng a**.

+ Nếu $i \geq k$ có nghĩa là i là phần tử cuối của đoạn $(i-k+1)..i$ ta suy ra :

$$\min(a[i-k+1],a[i-k+2]..a[i]) = Queue[Head].value.$$

Ta cần chú ý: Vì kết quả cần tìm nằm trong đoạn $a[i-k+1],a[i-k+2]..a[i]$ nên phần tử đầu của Queue tức là Head luôn phải thỏa mãn $Head \geq i-k+1$ thì $Queue[Head]$ mới là kết quả cần tìm. Vì vậy để $Head \geq i-k+1$ khi xét đến $a[i]$ thì khi lấy $Queue[Head]$ ra khỏi Queue để đưa ra kết quả nếu $y = Queue[head].cs \leq i-k+1$ thì ta **không đẩy** lại $x = Queue[head]$ vào



đầu của Queue nữa (PushL) trong trường hợp $y > i - k + 1$ có nghĩa là $i - k + 2 \leq y \leq i$ thì ta lại **đẩy lại x vào Queue** vì đoạn tiếp theo cần tìm là $\text{Min}(a[i-k+2], a[i-k+3] \dots a[i+1])$, nên ta phải giữ lại và tất nhiên các phần tử còn lại trong Queue cũng có chỉ số trong đoạn đó.

Chương trình:

```

Const
    fi='MINK.INP';
    fo='MINK.OUT';
    maxn=17000;

Type
    Element =record
        value :longint;
        cs     :longint;
    end;
    arr1   =array[1..maxn] of longint;
    DQueue =array[1..maxn] of Element;

Var
    T      :longint;
    n, k   :longint;
    a      :arr1;
    Queue  :DQueue;
    Head, Top:longint;
    f, f1  :text;

Procedure nhap;
var
    i      :longint;
begin
    readln(f,n,k);
    for i:=1 to n do read(f,a[i]);
end;

Procedure InitQ;
begin
    Top:=0;
    Head:=1;
end;

Procedure PushR(x,y:longint);{Day phan tu x co chi so trong mang a la y vao cuoi Queue}
begin
    inc(Top);
    Queue[Top].value:=x;
    Queue[Top].cs:=y;
end;

Procedure PushL(x,y:longint);{Day phan tu x co chi so trong mang a la y vao dau Queue}
begin
    dec(Head);
    Queue[Head].value:=x;
    Queue[Head].cs:=y;
end;

Procedure PopR(var x,y:longint);{Lay ra phan tu o cuoi Queue gan gia tri vao x, chi so vao y}
begin
    x:=Queue[Top].value;
    y:=Queue[Top].cs;
    dec(Top);
end;

Procedure PopL(var x,y:longint);{Lay ra phan tu o dau Queue gan gia tri vao x, chi so vao y}
begin
    x:=Queue[Head].value;
    y:=Queue[Head].cs;
end;

```



```

y:=Queue[Head].cs;
inc(Head);
end;

Procedure Solution;
var
  i,j      :longint;
  x        :longint;
  y        :longint;
  cuoi    :longint;
begin
  InitQ;
  for i:=1 to n do
    If Head>Top then
      begin
        PushR(a[i],i);
        If i>=k then write(f1,a[i],' ');
        if k=1 then PopL(x,y);
      end
    else
      begin
        PopR(x,y);
        if x<=a[i] then
          begin
            PushR(x,y);
            PushR(a[i],i);
          end
        else
          begin
            while (x>a[i]) and (Top>=head) do PopR(x,y);
            If x<=a[i] then PushR(x,y);
            PushR(a[i],i);
          end;
        if i>=k then
          begin
            PopL(x,y);
            write(f1,x,' ');{x=Min(a[i-k+1]..a[i])}
            if y>i-k+1 then PushL(x,y);
          end;
      end;
  end;
end;

Procedure Run;
var
  i      :longint;
begin
  assign(f,fi);
  reset(f);
  Readln(f,T);
  assign(f1,fo);
  rewrite(f1);
  for i:=1 to T do
    begin
      nhap;
      solution;
      writeln(f1);
    end;
  close(f);
  close(f1);
end;

begin
  run;
end.

```



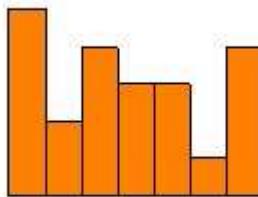
Problem 4: Bán dừa

Mã bài: KPLANK VNOL.INFO

Nếu các bạn biết câu chuyện thương tâm "ăn dừa leo trả vàng" của Pirate hẳn đã phải khóc hết nước mắt khi anh ấy, vì lòng thương chim, đã bán rẻ trái dừa leo siêu bụi của mình.

Dừa leo cũng đã bị chim to lấy đi rồi, Pirate giờ chuyển sang nghề bán dừa để bù lỗ. Bất đắc dĩ thôi, vì trên đảo toàn là dừa...

Nhưng mà bán cái gì thì đầu tiên cũng phải có biển hiệu đã. Pirate quyết định lùng sục trên đảo các mảnh ván còn sót lại của những con tàu đãm để ghép lại thành tấm biển. Cuối cùng anh cũng tìm được N tấm ván hình chữ nhật, tấm thứ i có chiều rộng là 1 đơn vị và chiều dài là a_i đơn vị. Pirate dựng đứng chúng trên mặt đất và dán lại với nhau để được một mảnh ván to hơn (xem hình minh họa).



Việc cuối cùng chỉ là đem mảnh ván này đi cưa thành tấm biển thôi. Nhưng hóa ra đây lại là công việc khó khăn nhất. Pirate rất thích hình vuông và muốn tấm biển của mình càng to càng tốt, nhưng không ở trên đảo lại không có nhiều dụng cụ đo đạc. Không êke, không thước đo độ, nên Pirate chỉ còn cách dựa vào cạnh của N tấm ván ban đầu để cưa cho thẳng thắn. Pirate chỉ có thể cưa theo những đoạn thẳng chứa một cạnh nào đó (dọc hoặc ngang) của các tấm ván.

Hãy giúp anh ấy cưa được tấm biển lớn nhất có thể.

Input

- Dòng thứ nhất: ghi số nguyên N - số tấm ván.
- N dòng tiếp theo: mô tả độ cao của các tấm ván theo thứ tự trái sang phải sau khi đã dán lại.

Output

- Một số nguyên duy nhất là độ dài cạnh của tấm biển lớn nhất có thể cưa được.

Giới hạn

- Độ cao của các tấm ván là các số nguyên dương không vượt quá 10^9 .
- $1 \leq N \leq 10^6$.
- 60% số test có $1 \leq N \leq 2000$.
- 80% số test có $1 \leq N \leq 10^5$.

Example

Input:

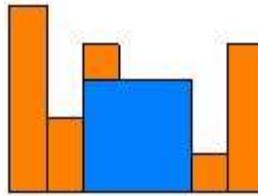


7
5
2
4
3
3
1
4

Output:

3

Giải thích: Hình dưới đây minh họa phương án tối ưu.



Note:

+ Phân biệt **tấm ván** và **tấm biển**.

+ Các tấm ván cắt ra làm tấm biển phải trong **một đoạn liên tiếp các tấm ván từ i đến j.**(1.1)

+ Vì không có thước nên muôn **Pirate** cắt được theo chiều ngang thì phải có **một tấm ván làm mấu** có nghĩa là trong tấm biển cần tìm phải **có ít nhất một tấm ván giữ nguyên không bị cắt.** (1.2)

+ Để đạt được 60% đến 80% số test chỉ việc xét i là cạnh của tấm biển cần tìm sau đó tìm left nhỏ nhất $\leq i$ và right lớn nhất $\geq i$ thỏa mãn $a[left..right] \geq a[i]$ nếu $right-left = a[i]$ thì cập nhật $a[i]$ với Max. Chú ý: Nếu $a[i] \leq Max$ thì ta không cần xét tấm ván i nữa.

Solution:

Gọi $left[i]$ và $right[i]$ lần lượt là gần i nhất thỏa mãn:

- + $left[i] \leq i; right[i] \geq i$.
- + $Min(a[left[i]..right[i]]) \geq a[i]$.
- + $a[left[i-1]] < a[i]$ và $a[right[i]+1] > a[i]$.

Như vậy ta dễ dàng thấy nếu $right[i]-left[i]+1 \geq a[i]$ thì ta sẽ cắt được tấm biển có cạnh là $a[i]$ và chứa tấm ván i.

Bây giờ ta tính lần lượt $left[i]$ và $right[i]$ bằng Stack.



Note:

Nếu $a[i-1] < a[i]$ thì suy ra $left[i]:=i$ trái lại nếu $a[i-1] \geq a[i]$ thì $left[i]$ luôn nhỏ hơn hoặc bằng $left[i-1]$ vì trong đoạn $left[i-1]..i-1$ các phần tử luôn lớn hơn hoặc bằng $a[i-1]$ mà $a[i-1] \geq a[i]$ nên \Rightarrow các phần tử trong đoạn $left[i-1]..i$ luôn lớn hơn hoặc bằng $a[i]$.

Tương tự ta có nhận xét trên đối với right.

Ta dùng Stack lưu lại $left[i], left[left[i]]$, và i để thu hẹp phạm vi tìm kiếm $left[i]$, thay vì phải duyệt từ $i-1$ đến khi nào gặp phần tử nhỏ hơn, thì với những phần tử có chiều cao lớn hơn hoặc bằng $a[i]$ thì nếu $a[i+1] \leq a[i]$ thì các phần tử đó cũng thỏa mãn $i+1$ nên ta chỉ cần **tìm tiếp từ $left[i-1]$ trở xuống**, và ta chỉ cần lưu i vào Stack để so sánh $a[i+1]$ với $a[i]$ thôi **thay vì phải lưu đoạn từ $left[i]$ đến i** . Còn nếu $a[i+1] < a[i]$ thì $left[i]$ luôn bằng i.

Nếu Stack rỗng thì ta có luôn có $left[i]:=i$ và đẩy i vào Stack.

Nếu Stack khác rỗng thì ta làm như sau:

Khởi tạo $y:=i$;

Chừng nào Stack chưa rỗng và $a[Stack[Top]] \geq a[i]$ thì $\Rightarrow Pop(x)$; (x chính là $Stack[Top]$) ta có $y=left[x]$.

Khi đó $left[i]:=y$; Đẩy i vào Stack. Ta thấy trong cả hai trường hợp i đều được đẩy vào Stack để phục vụ cho việc tìm $left[i+1]$.

Tính right[i] tương tự như tính $left[i]$ chỉ khác duyệt các phần tử theo thứ tự ngược lại.

Chương trình:

```
Const
  tfi='KPLANK.INP';
  tfo='KPLANK.OUT';
  maxn=1000000;

Type
  arr1 =array[1..maxn] of longint;
  TStack =array[0..maxn+1] of longint;

Var
  n      :longint;
  a      :arr1;
  left   :arr1;
  right  :arr1;
  Stack  :TStack;
  Top    :longint;
  kq     :longint;
  fi,fo :text;

Procedure nhap;
var
  i      :longint;
begin
  assign(fi,tfi);
  reset(fi);
  readln(fi,n);
  for i:=1 to n do readln(fi,a[i]);
```



```

        close(fi);
end;

procedure Init;
begin
    Top:=0;
end;

Procedure Push(x:longint);
begin
    inc(Top);
    Stack[Top]:=x;
end;

Procedure Pop(var x:longint);
begin
    x:=Stack[Top];
    dec(Top);
end;

Procedure solution;
var
    i          :longint;
    x,y       :longint;
begin
    {Tim left[i]}
    Init;
    for i:=1 to N do
        begin
            if Top=0 then
                begin
                    Push(i);
                    left[i]:=i;
                end
            else
                begin
                    y:=i;
                    While (Top>0) and (a[Stack[Top]]>=a[i]) do
                        begin
                            Pop(x);
                            {Neu a[x]>=a[i] thi a[left[x]..x]>=a[i]}
                            y:=left[x];
                        end;
                    left[i]:=y;
                    Push(i);
                end;
        end;
    {Tim right[i]}
    Init;
    for i:=N downto 1 do
        begin
            if Top=0 then
                begin
                    Push(i);
                    right[i]:=i;
                end
            else
                begin
                    y:=i;
                    While (Top>0) and (a[Stack[Top]]>=a[i]) do
                        begin
                            Pop(x);
                            {Neu a[x]>=a[i] thi a[x..right[x]]>=a[i]}
                            y:=right[x];
                        end;
                    right[i]:=y;
                end;
        end;
end;

```



```

                Push(i);
            end;
        end;
        kq:=0;
        for i:=1 to n do
            if (right[i]-left[i]+1>=a[i]) and (a[i]>kq) then kq:=a[i];
    end;

Procedure xuat;
begin
    assign(fo,tfo);
    rewrite(fo);
    write(fo,kq);
    close(fo);
end;

Begin
    Nhap;
    Solution;
    Xuat;
End.

```



Problem 5: Giải mã văn tự MAYA

IOI 2006

Giải mã văn tự MayA là một nhiệm vụ phức tạp hơn so với các nghiên cứu trước đây. Trên thực tế, sau gần 200 năm người ta chưa làm sáng tỏ gì nhiều lầm trong lĩnh vực này. Chỉ trong phạm vi ba thập niên cuối này mới có những tiến bộ đáng kể trong nghiên cứu.

Văn tự MayA đặt cơ sở dựa vào các hình vẽ nhỏ, được biết dưới dạng các nét vạch biểu diễn âm tiết. Từ trong tiếng May A thường được viết dưới dạng ô vuông chứa một số các nét vạch. Đôi khi một từ bị bỏ đọc thành nhiều ô hoặc một ô lại chứa nhiều nét vạch hơn số nét cần thiết cho một từ.

Một trong số các vấn đề liên quan tới giải mã văn tự May A nảy sinh khi xác định trình tự đọc âm tiết. Khi điền các vạch vào ô vuông, đôi khi người May A lại quy trình trình tự đọc dựa trên các tiêu chuẩn thẩm mỹ riêng chứ không theo một quy luật chung. Điều này dẫn đến việc, ngay cả khi đã biết rõ âm tiết của nhiều nét vạch, các nhà khảo cổ học cũng không dám khẳng định chắc chắn cách phát âm của từ.

Các nhà khảo cổ đang khảo sát một từ W cụ thể. Họ biết những nét gạch tạo thành từ đó, nhưng không biết hết các cách vẽ chúng. Biết bạn đến tham dự IOI-06, họ đề nghị bạn giúp đỡ. Các nhà khảo cổ sẽ chọn bạn biết g nét gạch tạo thành từ W và dãy S các nét vạch (theo trình tự xuất hiện) của câu đang khảo sát. Hãy xác định các khả năng xuất hiện từ W trong câu được khảo sát.

NHIỆM VỤ

Cho các nét vạch tạo thành từ W và dãy S các nét vạch trong bản văn tự chạm trổ. Hãy lập trình xác định số khả năng xuất hiện từ W trong S. Vì mọi trình tự xuất hiện các nét vạch trong W đều là chấp nhận được, các nhà khảo cổ yêu cầu bạn tìm số lượng dãy các nét vạch liên tiếp trong S, mỗi dãy tương ứng với hoán vị g nét vạch trong W.

HẠN CHÉ

$1 \leq g \leq 3000$ số lượng vạch trong W

$g \leq |S| \leq 3\,000\,000$ Số nét vạch trong dãy S

INPUT

Chương trình của bạn phải đọc dữ liệu từ file writing.in

WRITING.INP	Ý nghĩa
4 11	Dòng 1: Chứa 2 số nguyên g và $ S $ viết rời nhau.
cAda	Dòng 2: Chứa g nét vạch liên tiếp nhau tạo thành W . Mỗi nét vạch tương ứng với một ký tự. Các ký tự hợp lệ là ‘a’-‘z’ và ‘A’-‘Z’; ký tự hoa và thường là khác nhau.
AbrAcadAbRa	Dòng 3: Chứa $ S $ ký tự liên tiếp biểu diễn S . Valid characters are ‘a’-‘z’ and ‘A’-‘Z’; Các ký tự hợp lệ là ‘a’-‘z’ và ‘A’-‘Z’; ký tự hoa và thường là khác nhau. .



OUTPUT

Chương trình của bạn phải ghi kết quả ra file writing.out

WRITING.OUT	Ý nghĩa
2	Dòng 1: Phải chứa số khả năng xuất hiện W trong S.

CHẤM ĐIỂM

Có một bộ phận Tests được đánh giá tổng cộng 50 điểm, mỗi test thoả mãn ràng buộc $g \leq 10$.

Note: Các kí tự xét trong hang đá xuất hiện W phải liên tiếp nhau.

Solution:

Bài này thực chất không khó, có khá nhiều cách giải và cách đơn giản nhất là xét tất cả các từ có độ dài g trong S rồi sắp xếp từ đó và so sánh với từ W(cũng đã sắp xếp), vì các từ trong khoảng ‘a’..’z’ và ‘A’..’Z’ ta có thể dùng thuật toán đếm phân phối tuy nhiên thuật toán trên chỉ giải quyết được 75% bài toán. Sau đây tôi xin trình bày cách dùng Queue để giải quyết triệt để bài toán trên: Queue lưu chỉ số các kí tự trong hang đá.

Gọi $dd[i]$ là số kí tự có mã ASCII là i trong từ W. $d[i]$ là số kí tự có mã ASCII là i trong Queue.

Giả sử xét đến kí tự thứ i trong từ S, tất nhiên đã xét hết các kí tự từ 1 đến $i-1$ ta có: Gọi $k = \text{ord}[S[i]]$ (Mã ASCII kí tự $S[i]$)

Nếu $d[k]+1 > dd[k]$ (số kí tự $S[i]$ trong Queue cộng thêm $S[i]$ nhiều hơn trong từ W) thì ta phải lấy ra trong Queue (tất nhiên là lấy ở front) đến khi nào $d[k]+1 = dd[k]$ thì thôi. Và nếu $dd[k] > 0$ (Có kí tự $S[i]$ trong W) thì đẩy i vào Queue.

Nếu số kí tự trong Queue đúng bằng g ($\text{Queue}[\text{rear}] - \text{Queue}[\text{front}] + 1 = g$ tăng kq lên 1 (1)) thì có nghĩa trong hang đá các kí tự thứ $\text{Queue}[\text{front}]$ đến $\text{Queue}[\text{rear}]$ tạo thành từ W.

Chứng minh:

Nếu $S[i]$ không xuất hiện trong W thì cũng không được đưa vào Queue và khi đó vì $S[i]$ không xuất hiện trong W nên $dd[k]=0$ và $d[k]=0$ ($k = \text{ord}[S[i]]$) nên $d[k]+1$ luôn lớn hơn $dd[k]$ ta lấy đến khi nào Queue rỗng thì thôi.

Nếu $S[i]$ xuất hiện trong W mà $d[k]+1 \leq dd[k]$ có nghĩa là trong Queue chưa đủ $dd[k]$ kí tự $S[i]$ thì ta đẩy i vào Queue khi đó $d[k]$ tăng lên 1.

Nếu $S[i]$ xuất hiện trong W mà $d[k]+1 > dd[k]$ có nghĩa là trong Queue có số kí tự $S[i]$ lớn hơn $dd[k]$ (nói thê thôi chứ trong Queue chỉ chứa tối đa $dd[k]$ kí tự $S[i]$ thôi nha) thì ta lấy bớt phần tử ra khỏi Queue để thỏa mãn $d[k]=dd[k]$, ta thấy các phần tử được lấy ra ở front thỏa mãn các kí tự trong Queue là các kí tự liên tiếp trong S.

Vậy ta thấy trong Queue luôn chứa các kí tự có trong W và số lượng các kí tự này luôn \leq số kí tự trong W (1) luôn đúng

Note: Cài đặt Queue bằng danh sách vòng .



Chương trình:

```
Const
  tfi='DMAYA.INP';
  tfo='DMAYA.OUT';
  maxs=3000000;
  maxg=3000;

Type
  arr1 =array[1..maxs] of char;
  TQueue =array[0..maxg-1] of longint;
  arr3 =array[1..maxg] of char;
  arr2 =array[60..200] of longint;

Var
  m,n :longint;
  w :arr3;
  S :arr1;
  Queue :TQueue;
  front :longint;
  rear :longint;
  dem :longint;
  kq :longint;
  d :arr2;
  dd :arr2;
  fi,fo :text;

Procedure Init;
var
  i :longint;
begin
  for i:=60 to 200 do
    begin
      d[i]:=0;
      dd[i]:=0;
    end;
  kq:=0;
  dem:=0;
  Front:=0;
  rear:=Maxg-1;
end;

Procedure nhap;
var
  i :longint;
begin
  assign(fi,tfi);
  reset(fi);
  readln(fi,m,n);
  for i:=1 to m do
    begin
      read(fi,w[i]);
      inc(dd[ord(w[i])]);
    end;
  readln(fi);
  for i:=1 to n do read(fi,s[i]);
  close(fi);
end;

Procedure Push(x:longint);
begin
  rear:=(rear+1) mod maxg;
  Queue[rear]:=x;
  inc(d[ord(S[x])]);
  inc(dem);
end;
```



```

end;

Procedure Pop(x:longint);
begin
    x:=Queue[front];
    front:=(front+1) mod maxg;
    dec(d[ord(S[x])]);
    dec(dem);
end;

Procedure Solution;
var
    i,j      :longint;
    x        :longint;
    k        :longint;
begin
    For i:=1 to n do
        begin
            k:=ord(S[i]);
            while (d[k]>dd[k]-1) and (dem>0) do Pop(x);
            if dd[k]>0 then Push(i);
            if (Queue[rear]-Queue[front]+1=m) and (dem>0) then inc(kq);
        end;
end;

Procedure xuat;
begin
    assign(fo,tfo);
    rewrite(fo);
    write(fo,kq);
    close(fo);
end;

Begin
    Init;
    Nhaph;
    Solution;
    Xuat;
End.

```

Độ phức tạp: **O(n)**

Các bạn có thể Test bài này trên **VNOI.INFO** với mã bài là **PBCWRI**



Problem 6: Hàng đợi có độ ưu tiên

Mã bài: QBHEAP VNOL.INFO

Cho trước một danh sách rỗng. Người ta xét hai thao tác trên danh sách đó:

Thao tác "+V" (ở đây V là một số tự nhiên ≤ 1000000000): Nếu danh sách đang có ít hơn 15000 phần tử thì thao tác này bổ sung thêm phần tử V vào danh sách; Nếu không, thao tác này không có hiệu lực.

Thao tác "-": Nếu danh sách đang không rỗng thì thao tác này loại bỏ tất cả các phần tử lớn nhất của danh sách; Nếu không, thao tác này không có hiệu lực

Input

Gồm nhiều dòng, mỗi dòng ghi một thao tác. Thứ tự các thao tác trên các dòng được liệt kê theo đúng thứ tự sẽ thực hiện

Output

Dòng 1: Ghi số lượng những giá trị còn lại trong danh sách.

Các dòng tiếp theo: Liệt kê những giá trị đó theo thứ tự giảm dần, mỗi dòng 1 số

Example

Input:

```
+1  
+3  
+2  
+3  
-  
+4  
+4  
-  
+2  
+9  
+7  
+8  
-
```

Output:

```
4  
8  
7  
2  
1
```

Solution:

Đây là một bài toán điển hình về Heap, các thao tác không có gì ngoài hai thủ tục chính của Heap.

Chương trình:



```

Const
  fi='QBHEAP.INP';
  fo='QBHEAP.OUT';
  maxn=15000;
  maxHeap=maxn;

Type
  HeapMax =array[1..maxHeap] of longint;
  arr1   =array[1..maxn] of longint;

Var
  Heap    :HeapMax;
  Top     :longint;
  dTop    :longint;
  kq      :arr1;
  f       :text;

Procedure InitHeap;
begin
  Top:=0;
end;

Procedure UpHeap(i:longint);
var
  c,r    :longint;
  value   :longint;
begin
  value:=Heap[i];
  c:=i;
  repeat
    r:=c div 2;
    if (r=0) or (Heap[r]>=value) then break;
    Heap[c]:=Heap[r];
    c:=r;
  until false;
  Heap[c]:=value;
end;

Procedure DownHeap(i:longint);
var
  c,r    :longint;
  value   :longint;
begin
  value:=Heap[i];
  c:=i;
  repeat
    r:=2*c;
    if Heap[r]<Heap[r+1] then inc(r);
    if (Heap[r]<=value) or (r>Top) then break;
    Heap[c]:=Heap[r];
    c:=r;
  until
  false;
  Heap[c]:=value;
end;

Procedure Push(V:longint);
begin
  if Top<maxn then
    begin
      inc(Top);
      Heap[Top]:=V;
      if Top>1 then UpHeap(Top);
    end;
end;

```



```

Procedure Pop(var x:longint);
begin
    if Top>0 then
        begin
            x:=Heap[1];
            Repeat
                Heap[1]:=Heap[Top];
                dec(Top);
                DownHeap(1);
            Until (Top<=0) or (Heap[1]<>x);
        end;
end;

Procedure Solution;
var
    c      :char;
    V      :longint;
    x      :longint;
begin
    assign(f,fi);
    reset(f);
    While not eof(f) do
        begin
            read(f,c);
            if c='+' then
                begin
                    read(f,V);
                    Push(V);
                end
            else Pop(x);
            writeln(f);
        end;
    dTop:=0;
    While Top>0 do
        begin
            Pop(x);
            inc(dTop);
            kq[dtop]:=x;
        end;
end;

Procedure Print;
var
    i      :longint;
begin
    assign(f,fo);
    rewrite(f);
    writeln(f,dTop);
    For i:=1 to dTop do writeln(f,kq[i]);
    close(f);
end;

Begin
    InitHeap;
    Solution;
    Print;
End.

```



Problem 7: Double Queue

Mã bài: MSE07B VNOL.INFO

Ngân hàng BIG-Bank mở một chi nhánh ở Bucharest và được trang bị một máy tính hiện đại với các công nghệ mới nhập, C2#, VC3+ ... chỉ chuỗi mỗi cái là không ai biết lập trình.

Họ cần một phần mềm mô tả hoạt động của ngân hàng như sau: mỗi khách hàng có một mã số là số nguyên K, và khi đến ngân hàng giao dịch, họ sẽ nhận được 1 số P là thứ tự ưu tiên của họ. Các thao tác chính như sau

0 Kết thúc phục vụ

1 K P Thêm khách hàng K vào hàng đợi với độ ưu tiên P

2 Phục vụ người có độ ưu tiên cao nhất và xóa khỏi danh sách hàng đợi

3 Phục vụ người có độ ưu tiên thấp nhất và xóa khỏi danh sách hàng đợi.

Tất nhiên là họ cần bạn giúp rồi.

Input

Mỗi dòng của input là 1 yêu cầu, và chỉ yêu cầu cuối cùng mới có giá trị là 0. Giả thiết là khi có yêu cầu 1 thì không có khách hàng nào khác có độ ưu tiên là P.

$K \leq 10^6$, và $P \leq 10^7$. Một khách hàng có thể yêu cầu phục vụ nhiều lần và với các độ ưu tiên khác nhau.

Output

Với mỗi yêu cầu 2 hoặc 3, in ra trên 1 dòng mã số của khách hàng được phục vụ tương ứng. Nếu có yêu cầu mà hàng đợi rỗng, in ra số 0.

Sample

Input :

2

1 20 14

1 30 3

2

1 10 99

3

2

2

0

Ouput:

0

20

30

10

0



Solution:

Tư tưởng bài này rất đơn giản quản lí một lúc 2 Heap là HeapMin và HeapMax, cái khó trong việc cài đặt đồng thời 2 Heap một lúc.

Chương trình:

```
Const
  tfi='MSE07B.INP';
  tfo='MSE07B.OUT';
  maxn=1000000;

Type
  THeap = record
    nHeap :longint;
    value :array[1..maxn] of longint;
    pos   :array[1..maxn] of longint;
  end;
  arr1  =array[1..maxn] of longint;

Var
  p      :arr1;
  HeapMax :THeap;
  HeapMin :THeap;
  fi,fo  :text;

Procedure InitHeap;
var
  i      :longint;
begin
  HeapMax.nHeap:=0;
  HeapMin.nHeap:=0;
  for i:=1 to maxn do
    begin
      HeapMax.pos[i]:=0;
      HeapMin.pos[i]:=0;
    end;
end;

Function Tmp(a,b,k:longint):boolean;
begin
  if k=1 then
    begin
      if a>=b then exit(true);
      exit(false);
    end
  else
    begin
      if a<=b then exit(true);
      exit(false);
    end;
end;

Procedure UpHeap(i,k:longint;var Heap:THeap);
var
  r,c      :longint;
begin
  With Heap do
    begin
      r:=pos[i];
      repeat
        c:=r div 2;
```



```

        if (c=0) or Tmp(p[value[c]],p[i],k) then break;
        value[r]:=value[c];
        pos[value[c]]:=r;
        r:=c;
    until false;
    value[r]:=i;
    pos[i]:=r;
end;

Procedure DownHeap(i,k:longint;var Heap:THeap);
var
    r,c      :longint;
begin
    With Heap do
        begin
            r:=pos[i];
            repeat
                c:=2*r;
                if (c<nHeap) and Tmp(p[value[c]],p[value[c+1]],3-k) then
c:=c+1;
                if (c>nHeap) or Tmp(p[value[c]],p[i],3-k) then break;
                value[r]:=value[c];
                pos[value[c]]:=r;
                r:=c;
            until false;
            value[r]:=i;
            pos[i]:=r;
        end;
end;

Procedure Push(i,v,k:longint;var Heap:THeap);
begin
    With Heap do
        if pos[i]=0 then
            begin
                inc(nHeap);
                p[i]:=v;
                value[nHeap]:=i;
                pos[i]:=nHeap;
                UpHeap(i,k,Heap);
            end
        else
            begin
                p[i]:=v;
                UpHeap(i,k,Heap);
                DownHeap(i,k,Heap);
            end;
    end;

Procedure Pop(i,k:longint;var Heap:THeap;var ms:longint);
begin
    if Heap.nHeap=0 then
        begin
            ms:=0;
            exit;
        end;
    With Heap do
        begin
            ms:=value[i];
            if i<>nHeap then
                begin
                    value[i]:=value[nHeap];
                    pos[value[nHeap]]:=i;
                    pos[ms]:=0;
                    dec(nHeap);
                end;
        end;
end;

```



```

UpHeap (value[i],k,Heap) ;
DownHeap (value[i],k,Heap) ;
end
else
begin
    pos[ms]:=0;
    dec(nHeap);
end;
end;
end;

Procedure Solution;
var
    i,j,u,v :longint;
    k,p,t   :longint;
begin
    assign(fi,tfi);
    reset(fi);
    assign(fo,tfo);
    rewrite(fo);
    Repeat
        read(fi,u);
        if u=0 then break;
        if u=1 then
            begin
                readln(fi,k,p);
                Push(k,p,1,HeapMax);
                Push(k,p,2,HeapMin);
            end;
        if u=2 then
            begin
                Pop(1,1,HeapMax,v);
                writeln(fo,v);
                if v<>0 then Pop(HeapMin.pos[v],2,HeapMin,t);
            end;
        if u=3 then
            begin
                Pop(1,2,HeapMin,v);
                writeln(fo,v);
                if v<>0 then Pop(HeapMax.pos[v],1,HeapMax,t);
            end;
    until false;
    close(fi);
    close(fo);
end;
begin
    solution;
end.

```



Problem 8: Trồng hoa

Mã bài: KDIFF VNOI.INFO

Pirate là một người rất yêu hoa. Anh ấy trồng một luống hoa trước cửa nhà mình. Luống hoa được chia thành các ô đất, mỗi ô đất trồng một bông hoa. Tuy nhiên, vì đang bị đau chân nên Pirate ấy không thể chăm sóc luống hoa một cách hoàn hảo nhất. Kết quả là các bông hoa của anh có xấu đẹp không đều nhau.

Để cải thiện tình hình, Pirate quyết định chỉ để lại hai khóm hoa rời nhau, mỗi khóm gồm một số các bông hoa đứng liền tiếp nhau. Để ngôi nhà của mình trông thật xinh đẹp, hai khóm hoa kia phải được chọn lựa kỹ càng. Anh dùng đôi mắt thẩm mỹ tinh tường của mình (gọi là "sắc kế") để đánh giá độ xinh đẹp của các bông hoa, được thể hiện bằng các số nguyên không âm. Căn cứ vào đó, một khóm hoa đạt tiêu chuẩn khi và chỉ khi chênh lệch độ xinh đẹp giữa hai bông hoa bất kì trong khóm không quá một giá trị cho trước. Pirate muốn hai khóm hoa có càng nhiều bông hoa càng tốt. Bạn hãy giúp anh ấy xác định xem có thể chọn được nhiêu nhất bao nhiêu bông nhé.

Input

- Dòng 1: Hai số nguyên N - số bông hoa trên luống hoa, K - chênh lệch độ xinh đẹp tối đa của hai bông hoa bất kì trong một khóm.
- N dòng tiếp theo: Mỗi dòng là một số nguyên thể hiện độ xinh đẹp của một bông hoa.

Output

- Một số nguyên duy nhất là số bông hoa được chọn của hai khóm hoa.

Giới hạn

- $1 \leq N \leq 3 * 10^5$.
- 30% số test có $1 \leq N \leq 30$.
- 50% số test có $1 \leq N \leq 10^3$.
- Các số trong dữ liệu vào đều là số nguyên không âm không quá 10^9 .

Example

Input:

5 2
1
3
2
5
4

Output:

5

Giải thích: hai khóm hoa được chọn là (1, 2, 3) và (4, 5).

Solution:

Gọi L[i] là số hoa nhiều nhất được chọn khi xét các bông từ 1 tới i.

Ta có $L[i]=\text{Max}(L[i-1], i-ik+1)$ với ik là chỉ số nhỏ nhất thỏa mãn $\text{Max}(a[ik], a[ik+1], \dots, a[i]) - \text{Min}(a[ik], a[ik+1], \dots, a[i]) \leq k$.

Gọi R[i] là số hoa nhiều nhất được chọn khi xét các bông hoa từ n về i.



Ta có $R[i]=\max(L[i+1], ik-i+1)$ với ik là chỉ số lớn nhất thỏa mãn $\max(a[ik], a[ik-1], \dots, a[i]) - \min(a[ik], a[ik-1], \dots, a[i]) \leq k$.

Ta có $kq=\max(L[i]+R[i+1])$ với $i=1..n-1$.

Vì $n \leq 3 \cdot 10^5$ nên chúng ta sẽ phải tính ik trong thời gian **logn** ta sẽ sử dụng cấu trúc dữ liệu nâng cao để cài đặt, chúng ta có thể sử dụng hàng đợi hai đầu **Double Ended Queue**.

Nhưng ở đây tôi sẽ cài đặt theo cách khác mà tìm Max, Min trong thời gian **O(1)**, chèn thêm, hoặc xóa bớt một phần tử trong thời gian **O(logn)** đó chính là **Heap**, chúng ta sử dụng 2 Heap là HeapMin và HeapMax. Khi đó tính chênh lệch giữa bông hoa xinh đẹp nhất và bông hoa xấu nhất là $p[HeapMax.value[1]] - p[HeapMin.value[1]]$ (1)

Tương tự như trên chừng nào $(1) > k$ thì lấy ra khỏi 2 Heap phần tử có chỉ số nhỏ nhất.

Chương trình:

```
Const
    fi='KDIFF.INP';
    fo='KDIFF.OUT';
    maxn=300000;

Type
    THeap = record
        value :array[0..maxn+1] of longint;
        pos :array[0..maxn+1] of longint;
        nHeap :longint;
    end;
    arr1 = array[0..maxn+1] of longint;

Var
    N, ik, kq :longint;
    HeapMin :THeap;
    HeapMax :THeap;
    L, R, p :arr1;
    f :text;

Procedure Nhap;
var
    i :longint;
begin
    assign(f,fi);
    reset(f);
    readln(f,n,ik);
    for i:=1 to n do readln(f,p[i]);
    close(f);
end;

Procedure Init;
var
    i :longint;
begin
    HeapMin.nHeap:=0; HeapMax.nHeap:=0;
    for i:=1 to n do
        begin
            HeapMax.pos[i]:=0;
            HeapMin.pos[i]:=0;
        end;
end;

Procedure UpHeap(i,k:longint;var Heap:THeap);
var
    r,c :longint;
begin
    With Heap do
        begin
            r:=pos[i];
            Repeat
                c:=r div 2;
                if c=0 then break;
                if pos[c]<pos[r] then
                    begin
                        pos[i]:=pos[c];
                        pos[c]:=r;
                    end;
                r:=c;
            Until c=0;
        end;
end;
```



```

                if ((k=1) and (p[i]<=p[value[c]])) or ((k=2) and
(p[i]>=p[value[c]])) then break;
                        value[r]:=value[c];
                        pos[value[r]]:=r;
                        r:=c;
                        until false;
                        value[r]:=i;
                        pos[i]:=r;
                end;
end;

Procedure DownHeap(i,k:longint;var Heap:THeap);
var
    r,c      :longint;
begin
    With Heap do
        begin
            r:=pos[i];
            repeat
                c:=2*r;
                if (c<nHeap) and (((k=1) and (p[value[c]]<p[value[c+1]])) or
((k=2) and (p[value[c]]>p[value[c+1]]))) then c:=c+1;
                if c>nHeap then break;
                if ((k=1) and (p[i]>=p[value[c]])) or ((k=2) and
(p[i]<=p[value[c]])) then break;
                value[r]:=value[c];
                pos[value[r]]:=r;
                r:=c;
            until false;
            value[r]:=i;
            pos[i]:=r;
        end;
end;

Procedure Push(i,k:longint;var Heap:THeap);
begin
    With Heap do
        begin
            inc(nHeap);
            value[nHeap]:=i;
            pos[i]:=nHeap;
            UpHeap(i,k,Heap);
        end;
end;

Procedure Pop(i,k:longint;var Heap:THeap);
var
    r      :longint;
begin
    With Heap do
        begin
            r:=pos[i];
            pos[i]:=0;
            if r<>nHeap then
                begin
                    value[r]:=value[nHeap];
                    pos[value[r]]:=r;
                    dec(nHeap);
                    DownHeap(value[r],k,Heap);
                    UpHeap(value[r],k,Heap);
                end
            else dec(nHeap);
        end;
end;

Procedure Solution;

```



```

var
  i,j,k    :longint;
  u,v      :longint;
begin
  Init;
  L[1]:=1; k:=1;
  Push(1,1,HeapMax);
  Push(1,1,HeapMin);
  for i:=2 to n do
    begin
      Push(i,1,HeapMax);
      Push(i,2,HeapMin);
      u:=p[HeapMax.value[1]];
      v:=p[HeapMin.value[1]];
      While u-v>ik do
        begin
          Pop(k,1,HeapMax);
          Pop(k,2,HeapMin);
          inc(k);
          u:=p[HeapMax.value[1]];
          v:=p[HeapMin.value[1]];
        end;
      l[i]:=l[i-1];
      if l[i]<i-k+1 then l[i]:=i-k+1;
    end;
  Init;
  Push(n,1,HeapMax);
  Push(n,2,HeapMin);
  r[n]:=1;k:=n;
  for i:=n-1 downto 1 do
    begin
      Push(i,1,HeapMax);
      Push(i,2,HeapMin);
      u:=p[HeapMax.value[1]];
      v:=p[HeapMin.value[1]];
      While u-v>ik do
        begin
          Pop(k,1,HeapMax);
          Pop(k,2,HeapMin);
          dec(k);
          u:=p[HeapMax.value[1]];
          v:=p[HeapMin.value[1]];
        end;
      r[i]:=r[i+1];
      if r[i]<k-i+1 then r[i]:=k-i+1;
    end;
  kq:=0;
  for i:=1 to n-1 do if l[i]+r[i+1]>kq then kq:=l[i]+r[i+1];
end;

Procedure xuat;
begin
  assign(f,fo);
  rewrite(f);
  write(f,kq);
  close(f);
end;

Begin
  Nhap;
  Solution;
  Xuat;
End;

```

Độ phức tạp: O($n \log n$)



B.Interval Tree

Interval Tree (IT) là một cấu trúc quản lí đoạn khá hiệu quả mà cài đặt ngắn gọn dễ dàng hơn Binary Search Tree rất nhiều mà không phức tạp hơn Binary Index Tree là mấy.

Cấu trúc nguyên thủy của Interval Tree dùng để giải quyết các bài toán hình học, nhưng ở đây ta chỉ xét quản lí đoạn đơn giản hơn, có nghĩa là không có nhiều tác dụng như cấu trúc nguyên thủy, một số tài liệu gọi đây là Segment Tree (Cây quản lí đoạn), nhưng ở Việt Nam cái tên Interval Tree thông dụng hơn.

Xét cấu trúc của Interval Tree:

- + Nút thứ nhất quản lí đoạn $[1..n]$.
- + Nút thứ hai quản lí đoạn $[1..k]$, nút thứ ba quản lí đoạn $[k+1,n]$ với $k=(1+n)$ div 2.
- + Tổng quát nút thứ i quản lí đoạn $[L,R]$ thì nút 2^i quản lí đoạn $[L,mid]$, nút 2^i+1 quản lí đoạn $[mid+1,R]$ với $mid=(L+R)$ div 2.

Người ta đã chứng minh được số nút của Interval Tree luôn nhỏ hơn hoặc bằng 4^n , với n là số phần tử của dãy cần quản lí, vì vậy ta chỉ cần khai báo số phần tử tối đa của IT là $4^{\max n}$.

Trên mỗi nút sẽ lưu một thông tin phụ trợ quản lí nút đó như Max, Sum.... để trả lời các truy vấn cũng như thực hiện các phép biến đổi, bài toán cần những thông tin nào thì ta sẽ bổ xung thông tin đó.

Ứng dụng của Interval Tree rất đa dạng như trong các bài toán Quy hoạch động, các bài toán về dãy số, tập hợp động ... (Như thế là quá đủ ☺).



Problem 1: Giá trị lớn nhất

Mã bài: QMAX VNOL.INFO

Cho một dãy gồm n phần tử có giá trị ban đầu bằng 0.

Cho m phép biến đổi, mỗi phép có dạng (u, v, k) : tăng mỗi phần tử từ vị trí u đến vị trí v lên k đơn vị.

Cho q câu hỏi, mỗi câu có dạng (u, v) : cho biết phần tử có giá trị lớn nhất thuộc đoạn $[u, v]$

Giới hạn

- $n, m, p \leq 50000$
- $k > 0$
- Giá trị của một phần tử luôn không vượt quá $2^{31}-1$

Input

- Dòng 1: n, m
- m dòng tiếp theo, mỗi dòng chứa u, v, k cho biết một phép biến đổi
- Dòng thứ $m+2$: p
- p dòng tiếp theo, mỗi dòng chứa u, v cho biết một phép biến đổi

Output

- Gồm p dòng chứa kết quả tương ứng cho từng câu hỏi.

Example

Input:

6 2
1 3 2
4 6 3
1
3 4

Output:

3

Solution:

Đây là một bài toán sử dụng **Interval Tree** căn bản, điều khó nhất của thuật toán này xử lý phép biến đổi tăng trong thời gian $O(m)$, rất may bài toán này cho phép biến đổi trước nên ta chỉ cần xử lý offline không quá phức tạp.

Gọi mảng ban đầu là a (tất nhiên $a[i]=0$, $i=1..n$), sau một phép biến đổi (u,v,k) thì ta làm

$a[u]:=a[u]+k; a[v+1]:=a[v+1]-k;$

Sau khi xong m phép biến đổi thì ta làm $a[i]:=a[i-1]+a[i]$, $i=2..n$. Ta thấy ngay dãy a thu được chính là dãy a sau m phép biến đổi trên.

Công việc còn lại chỉ là dùng IT rồi trả lời các truy vấn (Query).



Chương trình:

```
Const
    tfi='QMAX.INP';
    tfo='QMAX.OUT';
    maxn=50000;

Type
    arr1 =array[1..maxn+1] of longint;
    arr2 =array[1..4*maxn] of longint;

Var
    n,m,q :longint;
    a :arr1;
    max :arr2;
    l,r :arr2;
    fi,fo :text;

Procedure Init;
var
    i :longint;
begin
    for i:=1 to n do a[i]:=0;
end;

Function maxs(a,b:longint):longint;
begin
    if a>b then exit(a);
    exit(b);
end;

Procedure BuildTree(x,i,j:longint);
var
    k :longint;
begin
    l[x]:=i;
    r[x]:=j;
    if i=j then max[x]:=a[i]
    else
        begin
            k:=(i+j) div 2;
            BuildTree(2*x,i,k);
            BuildTree(2*x+1,k+1,j);
            Max[x]:=Maxs(Max[2*x],Max[2*x+1]);
        end;
end;

Function Query(i,j,x:longint):longint;
begin
    if (i>r[x]) or (j<l[x]) then exit(-maxlongint);
    if (i<=l[x]) and (j>=r[x]) then exit(max[x]);
    Query:=Maxs(Query(i,j,2*x),Query(i,j,2*x+1));
end;

Procedure Solution;
var
    i :longint;
    u,v,k :longint;
begin
    assign(fi,tfi);
    reset(fi);
    assign(fo,tfo);
    rewrite(fo);
    readln(fi,n,m);
    Init;
    for i:=1 to m do
```



```
begin
    readln(fi,u,v,k);
    a[u]:=a[u]+k;
    a[v+1]:=a[v+1]-k;
end;
for i:=1 to n do a[i]:=a[i-1]+a[i];
BuildTree(1,1,n);
readln(fi,q);
for i:=1 to q do
begin
    readln(fi,u,v);
    writeln(fo,Query(u,v,1));
end;
close(fi);
close(fo);
end;

Begin
    Solution;
End.
```



Problem 2: Giá trị lớn nhất version 2

Mã bài: QMAX2 VNOL.INFO

Giống bài "Giá trị lớn nhất" ở trên.

Input

- n: số phần tử của dãy ($n \leq 50000$).
- m: số lượng biến đổi và câu hỏi ($m \leq 100000$).
- + biến đổi có dạng: 0 x y value
- + câu hỏi có dạng : 1 x y.

Output

Ghi ra trả lời cho lần lượt từng câu hỏi.

Example

Input:

```
6 3  
0 1 3 3  
0 4 6 4  
1 1 6
```

Output:

```
4
```

Solution:

Bài này gần tương tự bài QMAX có điều ta phải xử lí các phép biến đổi online, vì truy vấn xen kẽ với biến đổi. Ta gọi add[i] là giá trị cộng theo nút i trong IT có nghĩa là nếu nút i quản lí đoạn từ L[i] đến R[i] thì tất cả các phần tử trong đoạn đó được cộng thêm add[i].

Chương trình:

```
Const  
  tfo='QMAX2.OUT';  
  tfi='QMAX2.INP';  
  maxn=50000;  
  
Type  
  arr1 =array[1..4*maxn] of longint;  
  
Var  
  Max :arr1;  
  add :arr1;  
  L,R :arr1;  
  n,m :longint;  
  fi,fo :text;  
  
Procedure BuildTree(x,i,j:longint);  
var  
  k :longint;  
begin  
  L[x]:=i;  
  R[x]:=j;
```



```

Max[x]:=0;
Add[x]:=0;
if i=j then exit;
if i<>j then
begin
    k:=(i+j) div 2;
    BuildTree(2*x,i,k);
    BuildTree(2*x+1,k+1,j);
end;
end;

Function GetMax(a,b:longint):longint;
begin
    if a>b then exit(a);
    exit(b);
end;

Procedure Update(x,i,j,k:longint);
begin
    if (L[x]>=i) and (R[x]<=j) then
begin
        inc(add[x],k);
        exit;
    end;
    if (L[x]>j) or (R[x]<i) then exit;
    Update(2*x,i,j,k);
    Update(2*x+1,i,j,k);
    Max[x]:=GetMax(Max[2*x]+add[2*x],Max[2*x+1]+add[2*x+1]);
end;

Function Q(x,i,j,k:longint):longint;
var
    xx,xy :longint;
begin
    if (L[x]>=i) and (R[x]<=j) then exit(Max[x]+add[x]+k);
    if (L[x]>j) or (R[x]<i) then exit(-maxlongint);
    k:=k+add[x];
    xx:=Q(2*x,i,j,k);
    xy:=Q(2*x+1,i,j,k);
    Q:=GetMax(xx,xy);
end;

Procedure Xuat;
var
    i,j,k,u,v:longint;
begin
    assign(fi,tfi);reset(fi);
    assign(fo,tfo);rewrite(fo);
    read(fi,n,m);
    BuildTree(1,1,n);
    for i:=1 to m do
begin
    read(fi,k,u,v);
    if k=0 then
begin
        read(fi,j);
        Update(1,u,v,j);
    end
    else writeln(fo,Q(1,u,v,0));
end;
close(fi);close(fo);
end;

Begin
    Xuat;
End.

```



Problem 3: Maximum Sum

Mã bài: KGSS VNOL.INFO

You are given a sequence $A[1], A[2], \dots, A[N]$ ($0 \leq A[i] \leq 10^8$, $2 \leq N \leq 10^5$). There are two types of operations and they are defined as follows:

Update:

This will be indicated in the input by a 'U' followed by space and then two integers i and x.

U i x, $1 \leq i \leq N$, and $x, 0 \leq x \leq 10^8$.

This operation sets the value of $A[i]$ to x.

Query:

This will be indicated in the input by a 'Q' followed by a single space and then two integers i and j.

Q x y, $1 \leq x < y \leq N$.

You must find i and j such that $x \leq i, j \leq y$ and $i < j$, such that the sum $A[i]+A[j]$ is maximized. Print the sum $A[i]+A[j]$.

Input

The first line of input consists of an integer **N** representing the length of the sequence. Next line consists of N space separated integers $A[i]$. Next line contains an integer **Q**, $Q \leq 10^5$, representing the number of operations. Next Q lines contain the operations.

Output

Output the maximum sum mentioned above, in a separate line, for each Query.

Example

Input:

```
5
1 2 3 4 5
6
Q 2 4
Q 2 5
U 1 6
Q 1 5
U 1 7
Q 1 5
```

Output:

```
7
9
11
12
```



Solution:

Đề bài dịch lại:

Cho dãy số $A[1], A[2], \dots, A[n]$ ($0 \leq A[i] \leq 10^8$, $2 \leq n \leq 10^5$). Có 2 thao tác là:

U i x : Gán $A[i]=x$

Q x y: Tìm 2 số i, j sao cho $x \leq i, j \leq y$, $i < j$ và $A[i]+A[j]$ đạt GTLN, in ra GTLN ấy.

Input:

Dòng đầu: n (số phần tử dãy số)

Dòng 2: là n số nguyên $A[1], A[2], \dots, A[n]$.

Dòng 3: là Q (Số thao tác và truy vấn).

Q dòng tiếp theo thể hiện một thao tác.

Output:

Với mỗi truy vấn ‘ Q ’ in ra kết quả cần tìm trên một dòng.

Thực ra bài này không khó, phép $U i x$ đã quá quen thuộc, vấn đề là xử lí truy vấn, vẫn để tìm Max trong đoạn $[x, y]$ là rất dễ dàng, ta nhận thấy $A[i]+A[j]$ lớn nhất thì $A[i]$ và $A[j]$ là hai số lớn nhất trong đoạn $[x, y]$ vì vậy ta chỉ cần tìm Max 2 lần là được, sau khi tìm được lần 1 ta gán lại số đó là $-oo$ để lần tìm thứ 2 không tìm trùng lại, sau khi tìm xong ta lại trả lại giá trị cho phần tử tìm được ở lần đầu.

Cách thứ 2 là sau khi tìm lần thứ nhất ta được phần tử $A[ik]$ là Max trong đoạn $[x, y]$ lần 2 ta tìm Max trong đoạn $[x, ik-1]$ và $[ik+1, y]$ sau đó tìm Max của 2 kết quả vừa tìm được. Tất nhiên cách thứ 2 sẽ chạy nhanh hơn vì không phải Update nhiều lần.

Chương trình dưới đây cài đặt theo cách thứ 2.



Chương trình:

```
Const
    tfi='KGSS.INP';
    tfo='KGSS.OUT';
    maxn=100000;

Type
    Element =record
        Max :longint;
        addr :longint;
    end;
    Interval=array[1..4*maxn] of longint;
    Segment =array[1..4*maxn] of longint;
    TPos    =array[1..4*maxn] of longint;
    TLeaf   =array[1..maxn] of longint;
    arrl   =array[1..maxn] of longint;

Var
    n,q      :longint;
    a       :arrl;
    Tree   :Interval;
    Pos    :TPos;
    L,R    :Segment;
    Leaf   :TLeaf;
    oo     :Element;
    fi,fo   :text;

Function Max(a,b:longint):longint;
begin
    if a>b then exit(a);
    exit(b);
end;

Function MaxE(a,b:Element):Element;
begin
    if a.Max>b.Max then exit(a);
    exit(b);
end;

Procedure BuildTree(i,j,x:longint);
var
    k      :longint;
begin
    L[x]:=i;
    R[x]:=j;
    if i=j then
        begin
            Leaf[i]:=x;
            Tree[x]:=a[i];
            Pos[x]:=i;
        end
    else
        begin
            k:=(i+j) div 2;
            BuildTree(i,k,2*x);
            BuildTree(k+1,j,2*x+1);
            Tree[x]:=Max(Tree[2*x],Tree[2*x+1]);
            if Tree[x]=Tree[2*x] then Pos[x]:=Pos[2*x]
            else Pos[x]:=Pos[2*x+1];
        end;
end;

Procedure Update(i,v:longint);
var
    x      :longint;
```



```

begin
    x:=Leaf[i];
    Tree[x]:=v;
    While x>1 do
        begin
            x:=x div 2;
            Tree[x]:=Max(Tree[2*x],Tree[2*x+1]);
            if Tree[x]=Tree[2*x] then Pos[x]:=Pos[2*x]
                else Pos[x]:=Pos[2*x+1];
        end;
    end;

Function Query(x,i,j:longint):Element;
var
    x1,x2    :Element;
begin
    if (j<L[x]) or (i>R[x]) then exit(oo);
    if (i<=L[x]) and (j>=R[x]) then
        begin
            x1.Max:=Tree[x];
            x1.addr:=Pos[x];
            exit(x1);
        end;
    x1:=Query(2*x,i,j);
    x2:=Query(2*x+1,i,j);
    Query:=MaxE(x1,x2);
end;

Procedure Solution;
var
    c      :char;
    i      :longint;
    u,v    :longint;
    x1,x2    :Element;
begin
    oo.Max:=-Maxlongint;
    assign(fi,tfi);
    reset(fi);
    assign(fo,tfo);
    rewrite(fo);
    readln(fi,n);
    for i:=1 to n do read(fi,a[i]);
    BuildTree(1,n,1);
    readln(fi,q);
    for i:=1 to q do
        begin
            readln(fi,c,u,v);
            if c='U' then Update(u,v);
            if c='Q' then
                begin
                    x1:=Query(1,u,v);
                    x2.Max:=0;
                    if u<=x1.addr-1 then x2:=Query(1,u,x1.addr-1);
                    if v>=x1.addr+1 then
                        writeln(fo,x1.Max+x2.Max);
                end;
        end;
    close(fi);
    close(fo);
end;

Begin
    Solution;
End.

```



Problem 4: Đoạn con có tổng lớn nhất

Mã bài: GSS VNOL.INFO

Cho dãy số $a[1], a[2], \dots, a[n]$ ($|a[i]| \leq 15000$, $n \leq 50000$).

Hàm $q(x, y) = \max \{ \text{tổng}(a[i]+a[i+1]+\dots+a[j]), x \leq i \leq j \leq y \}$.

Cho m câu hỏi dạng x, y ($1 \leq x \leq y \leq n$). ($m \leq 50000$) \rightarrow hãy tính các $q(x, y)$.

Input

- Dòng đầu là n .
- Dòng thứ hai là dãy a .
- Dòng thứ 3 là m .
- m dòng tiếp theo mỗi dòng là 1 cặp số x, y .

Output

\rightarrow Lần lượt ghi ra các $q(x, y)$ tương ứng. Mỗi kết quả ghi trên 1 dòng.

Example

Input:

3
-1 2 3
1
1 2

Output:

2

Solution:

Ta bổ xung thêm 2 thông tin cho mỗi nút của IT là MaxL và MaxR tương ứng là đoạn con trái $[x, i]$ đạt Max và đoạn con phải $[j, y]$ đạt Max ($i, j \in [x, y]$) tất nhiên nút đó quản lý đoạn $[x, y]$.

Ta cần cập nhật MaxL và MaxR một cách hợp lí.



Chương trình:

```
Const
    tfi='GSS.INP';
    tfo='GSS.OUT';
    maxn=50000;

Type
    arr1 =array[1..4*maxn] of longint;
    Interval=record
        Max :arr1;
        L,R :arr1;
        maxl :arr1;
        maxr :arr1;
        sum :arr1;
    end;
    Element =record
        MaxL :longint;
        MaxR :longint;
        Max :longint;
        Sum :longint;
    end;
    arr2 =array[1..maxn] of longint;

Var
    n,m :longint;
    a :arr2;
    Tree :Interval;
    fi,fo :text;

Procedure Nhap;
var
    i :longint;
begin
    assign(fi,tfi);
    reset(fi);
    readln(fi,n);
    for i:=1 to n do read(fi,a[i]);
    readln(fi,m);
end;

Function Maxs(a,b:longint):longint;
begin
    if a>b then exit(a);
    exit(b);
end;

Procedure BuildTree(x,i,j:longint);
var
    k :longint;
begin
    With Tree do
        begin
            L[x]:=i;
            R[x]:=j;
            if i=j then
                begin
                    MaxL[x]:=a[i];
                    MaxR[x]:=a[i];
                    Sum[x]:=a[i];
                    Max[x]:=a[i];
                end
            else
                begin
                    k:=(i+j) div 2;
                    BuildTree( 2*x,i,k);
                end
        end
end;
```



```

BuildTree(2*x+1, k+1, j);
Sum[x]:=Sum[2*x]+Sum[2*x+1];

Max[x]:=Maxs(Maxs(Max[2*x], Max[2*x+1]), MaxR[2*x]+MaxL[2*x+1]);
MaxL[x]:=Maxs(MaxL[2*x], Sum[2*x]+MaxL[2*x+1]);
MaxR[x]:=Maxs(MaxR[2*x+1], Sum[2*x+1]+MaxR[2*x]);
end;
end;

Function Q(x,i,j:longint):Element;
var
    xx,yy,zz:Element;
begin
    With Tree do
        begin
            xx.Max:=-Maxlongint;
            xx.MaxL:=-Maxlongint;
            xx.MaxR:=-Maxlongint;
            xx.Sum:=0;
            if (L[x]>j) or (R[x]<i) then exit(xx);
            if (L[x]>=i) and (R[x]<=j) then
                begin
                    xx.Max:=Max[x];
                    xx.MaxL:=MaxL[x];
                    xx.MaxR:=MaxR[x];
                    xx.Sum:=Sum[x];
                    exit(xx);
                end;
            xx:=Q(2*x,i,j);
            yy:=Q(2*x+1,i,j);
            if (xx.MaxR<>-Maxlongint) and (yy.MaxL<>-Maxlongint) then
                zz.Max:=Maxs(Maxs(xx.Max,yy.Max),xx.MaxR+yy.MaxL)
            else zz.Max:=Maxs(xx.Max,yy.Max);
            if yy.MaxL<>-Maxlongint then
                zz.MaxL:=Maxs(xx.MaxL,xx.Sum+yy.MaxL)
            else zz.MaxL:=xx.MaxL;
            if xx.MaxR<>-Maxlongint then
                zz.MaxR:=Maxs(yy.MaxR,xx.MaxR+yy.Sum)
            else zz.MaxR:=yy.MaxR;
            zz.Sum:=xx.Sum+yy.Sum;
            exit(zz);
        end;
    end;

Procedure Xuat;
var
    i      :longint;
    u,v    :longint;
begin
    assign(fo,tfo);
    rewrite(fo);
    for i:=1 to m do
        begin
            readln(fi,u,v);
            writeln(fo,Q(1,u,v).Max);
        end;
    close(fi);
    close(fo);
end;

Begin
    Nhap;
    BuildTree(1,1,n);
    Xuat;
end.

```



Problem 5: Xếp hàng

Mã bài: NKLINKEUP VNOL.INFO

Hàng ngày khi lấy sữa, N con bò của bác John ($1 \leq N \leq 50000$) luôn xếp hàng theo thứ tự không đổi. Một hôm bác John quyết định tổ chức một trò chơi cho một số con bò. Để đơn giản, bác John sẽ chọn ra một đoạn liên tiếp các con bò để tham dự trò chơi. Tuy nhiên để trò chơi diễn ra vui vẻ, các con bò phải không quá chênh lệch về chiều cao.

Bác John đã chuẩn bị một danh sách gồm Q ($1 \leq Q \leq 200000$) đoạn các con bò và chiều cao của chúng (trong phạm vi $[1, 1000000]$). Với mỗi đoạn, bác John muốn xác định chênh lệch chiều cao giữa con bò thấp nhất và cao nhất. Bạn hãy giúp bác John thực hiện công việc này!

Dữ liệu

- Dòng đầu tiên chứa 2 số nguyên N và Q.
- Dòng thứ i trong số N dòng sau chứa 1 số nguyên duy nhất, là độ cao của con bò thứ i.
- Dòng thứ i trong số Q trong tiếp theo chứa 2 số nguyên A, B ($1 \leq A \leq B \leq N$), cho biết đoạn các con bò từ A đến B.

Kết quả

Gồm Q dòng, mỗi dòng chứa 1 số nguyên, là chênh lệch chiều cao giữa con bò thấp nhất và cao nhất thuộc đoạn tương ứng.

Ví dụ

Dữ liệu:

6 3
1
7
3
4
2
5
1 5
4 6
2 2

Kết quả

6
3
0

Solution:

Đây là một bài toán cơ bản nhưng có ứng dụng lớn trong bài toán **LCA (Least common ancestor)** - Tổ tiên chung gần nhất mà sẽ đề cập ở phần đồ thị.

Bài toán trên còn có tên là RMQ (Range Minimum(Maximum) Query).

Cài đặt với 2 IT một quản lí Min một quản lí Max trong đoạn L[x] đến R[x].



Chương trình:

```
Const
    tfi='NKLINEUP.INP';
    tfo='NKLINEUP.OUT';
    maxn=50000;

Type
    IT      =array[1..4*maxn] of longint;
    arr1   =array[1..maxn] of longint;
    Segment =array[1..4*maxn] of longint;

Var
    n,q      :longint;
    a       :arr1;
    MaxT   :IT;
    MinT   :IT;
    L,R     :Segment;
    fi,fo   :text;

Procedure nhap;
var
    i      :longint;
begin
    assign(fi,tfi);
    reset(fi);
    readln(fi,n,q);
    for i:=1 to n do readln(fi,a[i]);
end;

Function Max(a,b:longint):longint;
begin
    if a>b then exit(a);
    exit(b);
end;

Function Min(a,b:longint):longint;
begin
    if a<b then exit(a);
    exit(b);
end;

Procedure BuildTree(x,i,j:longint);
var
    k      :longint;
begin
    L[x]:=i;
    R[x]:=j;
    if i=j then
        begin
            MaxT[x]:=a[i];
            MinT[x]:=a[i];
        end
    else
        begin
            k:=(i+j) div 2;
            BuildTree(2*x,i,k);
            BuildTree(2*x+1,k+1,j);
            MaxT[x]:=Max(MaxT[2*x],MaxT[2*x+1]);
            MinT[x]:=Min(MinT[2*x],MinT[2*x+1]);
        end;
end;

Function Qmax(x,i,j:longint):longint;
begin
    if (j<L[x]) or (i>R[x]) then exit(-maxlongint);
```



```

        if (i<=L[x]) and (j>=R[x]) then exit(MaxT[x]);
        Qmax:=Max(Qmax(2*x,i,j),Qmax(2*x+1,i,j));
    end;

Function Qmin(x,i,j:longint):longint;
begin
    if (j<L[x]) or (i>R[x]) then exit(maxlongint);
    if (i<=L[x]) and (j>=R[x]) then exit(MinT[x]);
    Qmin:=Min(Qmin(2*x,i,j),Qmin(2*x+1,i,j));
end;

Function Query(i,j:longint):longint;
var
    Ma,Mi    :longint;
begin
    Ma:=Qmax(1,i,j);
    Mi:=Qmin(1,i,j);
    Query:=Ma-Mi;
end;

Procedure Solution;
var
    u,v,i    :longint;
begin
    assign(fo,tfo);
    rewrite(fo);
    for i:=1 to q do
        begin
            readln(fi,u,v);
            writeln(fo,Query(u,v));
        end;
end;

Procedure Xuat;
begin
    close(fi);
    close(fo);
end;

begin
    Nhap;
    BuildTree(1,1,n);
    Solution;
    Xuat;
End.

```



Problem 6: Dãy con tăng

Mã bài: IS Tài liệu THPTCMGVC 2011

(Tài liệu tập huấn phát triển chuyên môn giáo viên trường THPT chuyên môn Tin Học năm 2011)

Cho dãy số nguyên dương $a[1], a[2] \dots a[n]$, phần tử $a[i]$ có trọng số $w[i]$. Mỗi dãy con $a_{i1}, a_{i2}, \dots a_{ik}$ thỏa mãn:

$$\left\{ \begin{array}{l} 1 \leq i_1 < i_2 < \dots < i_k \leq N \\ a_{i1} < a_{i2} < a_{i3} < \dots < a_{ik} \end{array} \right.$$

được gọi là một dãy con tăng của a.

Yêu cầu: Trong số các dãy con tăng của a tìm dãy con có tổng trọng số lớn nhất có thể.

Dữ liệu: Vào từ file IS.INP

- Dòng 1 chứa số nguyên dương n ($n \leq 10^5$).
- Dòng 2 chứa n số nguyên $a[1], a[2]..a[n]$ ($a[i] \leq 10^5$).
- Dòng 3 chứa n số nguyên dương $w[1], w[2]..w[n]$ ($w[i] \leq 10^4$).

Kết quả: Ghi ra file IS.OUT ghi duy nhất tổng trọng số lớn nhất tìm được.

Solution:

Đây là một bài toán Quy hoạch động diễn hình dùng Interval Tree.

Gọi $L[i]$ là trọng số lớn nhất của các dãy con tăng kết thúc tại $i \Rightarrow L[i] = \text{Max}(L[j] + w[i] | j < i, a[j] < a[i])$.

Vấn đề tìm j trong thời gian nhanh nhất có thể. Ta xây dựng mảng $g[0..10^5]$ với ý nghĩa $g[i]$ là tổng trọng số lớn nhất của dãy con kết thúc là i . Khi tính xong $L[i]$ nếu $L[i] > g[a[i]]$ thì cập nhật $g[a[i]] = L[i]$.

\Rightarrow Việc tìm j chính là truy vấn Max của đoạn $g[0..a[i]-1]$, ta có thể thực hiện dễ dàng với IT.



Chương trình:

```
Const
    fi='IS.INP';
    fo='IS.OUT';
    maxn=100000;

Type
    Interval=array[0..4*maxn] of int64;
    arr1    =array[0..maxn] of longint;
    arr2    =array[0..4*maxn] of longint;

Var
    n      :longint;
    a      :arr1;
    w      :arr1;
    g      :Interval;
    l,r   :arr2;
    left   :arr1;
    Maxa  :longint;
    kq    :int64;
    f     :text;

Procedure nhap;
var
    i      :longint;
begin
    assign(f,fi);
    reset(f);
    readln(f,n);
    Maxa:=0;
    for i:=1 to n do
        begin
            read(f,a[i]);
            if Maxa<a[i] then Maxa:=a[i];
        end;
    for i:=1 to n do read(f,w[i]);
    close(f);
end;

Function Max(a,b:int64):int64;
begin
    if a>b then exit(a);
    exit(b);
end;

Procedure BuildTree(i,j,x:longint);
var
    k      :longint;
begin
    l[x]:=i;
    r[x]:=j;
    if i=j then
        begin
            left[i]:=x;
            g[x]:=0;
        end
    else
        begin
            k:=(i+j) div 2;
            BuildTree(i,k,2*x);
            BuildTree(k+1,j,2*x+1);
            g[x]:=Max(g[2*x],g[2*x+1]);
        end;
end;
```



```

Procedure Update(i:longint;v:int64);
var
    x      :longint;
begin
    x:=left[i];
    g[x]:=v;
    while x>1 do
        begin
            x:=x div 2;
            g[x]:=Max(g[2*x],g[2*x+1]);
        end;
end;

Function Query(x,i,j:longint):int64;
begin
    if (i>r[x]) or (j<l[x]) then exit(low(int64));
    if (i<=l[x]) and (j>=r[x]) then exit(g[x]);
    Query:=Max(Query(2*x,i,j),Query(2*x+1,i,j));
end;

Procedure Solution;
var
    i,j      :longint;
    best     :int64;
begin
    kq:=0;
    for i:=1 to n do
        begin
            best:=Query(1,0,a[i]-1)+w[i];
            Update(a[i],best);
            if best>kq then kq:=best;
        end;
end;

Procedure Xuat;
begin
    assign(f,fo);
    rewrite(f);
    write(f,kq);
    close(f);
end;

Begin
    Nhap;
    BuildTree(0,Maxa,1);
    Solution;
    Xuat;
End.

```



C.Binary Search Tree

Binary Search Tree là một cấu trúc có thể nói là tổng quát nhất để giải quyết các bài toán Quy hoạch động, Danh sách, quản lí phạm vi,... nói chung mọi bài toán **Interval Tree**, **BIT** đều có thể giải bằng **BST**.

Có thể nói người ta tìm ra **BST** sau đó tùy vào bài toán thì **Interval Tree** và **Binary Index Tree** mới ra đời vì **BST** cài đặt tương đối phức tạp và mất nhiều thời gian.

Ở đây chủ yếu chúng ta sẽ khảo sát vai trò đầu tiên của **BST** là cây quản lí phạm vi.

BST cung cấp cho ta thao tác chèn, xóa, tìm kiếm, truy vấn trong thời gian logn nếu sử dụng **BST** cân bằng. Có nhiều loại **BST** cân bằng, nhưng thông dụng nhất đối với học sinh chuyên Tin nước ta hiện nay là **Splay Tree** và **Treap**, còn đối với học sinh đại học là **AVL Tree** và **Red Black Tree** (Cây đỏ đen).

Dưới đây chúng ta chỉ khảo sát 2 loại là **Splay Tree** và **Treap**.

Splay Tree thể hiện ý tưởng thú vị nhất về **BST** cân bằng. Thao tác làm giảm độ sâu của cây được thực hiện ngay khi có thao tác truy nhập nút. Trong trường hợp các khóa hay cụm khóa tìm kiếm có tần suất lớn thì nó thể hiện ưu thế về mặt tốc độ.

Treap là sự kết hợp của **Tree** và **Heap** mỗi nút gán một độ ưu tiên ngẫu nhiên, mỗi khi thực hiện các thao tác chèn xóa hay truy vấn thì các nút sẽ được cập nhật tùy theo độ ưu tiên của nút. Cụ thể với hai con trỏ x,y trên **Treap**:

Nếu y nằm trong nhánh con trái của x thì $y^.value \leq x^.value$.

Nếu y nằm trong nhánh con phải của x thì $y^.value \geq x^.value$.

Nếu y là hậu duệ của x thì $y^.priority \leq x^.priority$.

Nếu so sánh về tốc độ thì **Treap** có tốc độ tốt hơn **Splay Tree** hay **AVL Tree**. Nếu biểu diễn danh sách thì **Splay Tree** có ưu thế hơn vì có hai thủ tục Split và Join để tách ghép cây thuận tiện cho việc cắt chia danh sách.



Problem 1: Giá trị lớn nhất version 3

Mã bài: QMAX3VN VNOL.INFO

Tiếp tục hành trình khám phá sức mạnh của các cấu trúc dữ liệu đặc biệt, hai nhà thám hiểm *pirate* và *duyhung123abc* lại lao vào những thử thách với những bài toán học búa mới. Nhưng những gì họ thu thập được chẳng là bao. Chán chường và bế tắc, họ quyết định lật giờ những bài toán đã giải để xem lại những thành quả đã đạt được. Đột nhiên, một ánh sáng lóe lóe lên trong đầu họ: "*Tại sao không kết hợp một số bài toán lại với nhau !*". Và đây, **QMAX3VN** đã ra đời, tự nhiên như cái cách mà lửa đã đến với loài người.

Chuyện kể rằng vào một buổi sáng nọ trong một doanh trại quân đội, các quân nhân đang tập trung thành một hàng ngang để chuẩn bị tập thể dục buổi sáng. Nhưng vấn đề là không phải tất cả họ đều dậy cùng một lúc. Ban đầu, hàng ngang chẳng có người nào. Sau một lúc, lại có một anh chàng quân nhân hộc tốc chạy ra. Mỗi anh chàng lại thích đứng cạnh ai đó, nên cứ nhát quyết chen vào một vị trí nào đó trong hàng. Thật là một cảnh tượng hỗn độn và không thể chấp nhận. Mặt ai cũng cúi gầm xuống đất chuẩn bị chịu sự trừng phạt của viên chỉ huy. Viên chỉ huy, tuy nghiêm khắc nhưng cũng rất nhân hậu, đã đưa một trò chơi như sau để cho các anh chàng một bài học. Mỗi khi ông ta đưa ra một **khẩu lệnh** (x,y) thì lập tức các quân nhân phải chỉ vào **người cao nhất** trong các quân nhân đang đứng từ vị trí x đến vị trí y trong hàng. Khẩu lệnh sẽ được phát ra **liên tục** từ khi có một quân nhân bước vào hàng. Hình phạt đang chờ những người làm sai khẩu lệnh. Sân tập ướt đẫm những giọt mồ hôi lo lắng của các cậu lính trẻ. Họ đang cần sự giúp đỡ của bạn!

Input

Dữ liệu vào gồm nhiều dòng:

- Dòng thứ 1 : Một số nguyên n ($1 \leq n \leq 100000$): Số sự kiện sẽ diễn ra trong buổi tập thể dục.
- Dòng thứ 2 đến dòng thứ $n+1$: Mỗi dòng là một trong hai sự kiện:
 - **A x y** : Một quân nhân có chiều cao x vừa chen vào giữa vị trí thứ $y-1$ và y trong hàng ($-10^9 \leq x \leq 10^9$; $1 \leq y \leq k+1$, với k là số anh lính hiện có trong hàng).
 - **Q x y** : Khẩu lệnh của viên chỉ huy. Hãy tìm chiều cao của người cao nhất trong các quân nhân đang đứng từ vị trí thứ x đến vị trí thứ y trong hàng ($1 \leq x \leq y \leq k$, với k là số anh lính hiện có trong hàng).

Output

Dữ liệu ra gồm nhiều dòng:

- Với mỗi khẩu lệnh của viên chỉ huy, ghi trên một dòng chiều cao của người mà các quân nhân cần phải chỉ vào.

Example

Input:

10
A 1 1
A 2 2
Q 1 2
A 3 1
A 4 3
Q 2 4
A 5 2
Q 2 3
A 6 3
Q 1 4

Output:



2
4
5
6

Solution:

Đây là một bài toán điển hình dung BST quản lý danh sách với 2 thủ tục chính là chèn thêm phần tử x vào vị trí y trong danh sách và truy vấn tìm phần tử lớn nhất trong đoạn từ x đến y, dưới đây tôi sẽ cài đặt bằng Splay Tree

Chương trình:

```
Const
    tfi='QMAX3VN.INP';
    tfo='QMAX3VN.OUT';

Type
    Pnode  =^TNode;
    Tnode   =record
        value :longint;
        l,r,p :pnode;
        len,max:longint;
    end;

Var
    n      :longint;
    root   :pnode;
    sentinel:Tnode;
    nilT   :pnode;
    fi,fo  :text;

Procedure InitTree;
begin
    sentinel.len:=0;
    sentinel.max:=low(longint);
    sentinel.value:=0;
    nilT:=@sentinel;
    root:=nilT;
end;
Function max(a,b,c:longint):longint;
var
    d      :longint;
begin
    d:=a;
    if d<b then d:=b;
    if d<c then d:=c;
    exit(d);
end;
Procedure Update(x:Pnode);
begin
    x^.len:=x^.L^.len+x^.R^.len+1;
    x^.max:=max(x^.L^.max,x^.R^.max,x^.value);
end;

Procedure setl(x,y:pnode);
begin
    y^.p:=x;
    x^.l:=y;
end;
Procedure setr(x,y:pnode);
begin
    y^.p:=x;

```



```

        x^.r:=y;
end;

function nodeat(i:longint;x:pnode):pnode;
var
    sl      :longint;
begin
    repeat
        sl:=x^.L^.len+1;
        if sl=i then break;
        if sl>i then x:=x^.l
                    else
                        begin
                            i:=i-sl;
                            x:=x^.r;
                        end;
    until false;
    exit(x);
end;

Procedure uptree(x:pnode);
var
    y,z      :pnode;
begin
    y:=x^.p;
    z:=y^.p;
    if x=y^.l then
        begin
            setl(y,x^.r);
            setr(x,y);
        end
    else
        begin
            setr(y,x^.l);
            setl(x,y);
        end;
    if y=z^.l then setl(z,x)
                else setr(z,x);
    update(y);
    update(x);
end;

Procedure splay(x:pnode);
var
    y,z      :pnode;
begin
    repeat
        y:=x^.p;
        if y=nilt then break;
        z:=y^.p;
        if z<>nilt then
            begin
                if (x=y^.l)=(y=z^.l) then uptree(y)
                                         else uptree(x);
            end;
        uptree(x);
    until false;
end;

Procedure split(T:pnode;i:longint;var t1,t2:pnode);
begin
    if i=0 then
        begin
            t1:=nilt;
            t2:=t;
            exit;

```



```

        end;
t1:=nodeat(i,t);
splay(t1);
t2:=t1^.r;
t2^.p:=nilt;
t1^.r:=nilt;
update(t1);
end;

function Join(t1,t2:pnode):pnode;
begin
  if t1=nilt then exit(t2);
  while t1^.r<>nilt do t1:=t1^.r;
  splay(t1);
  setr(t1,t2);
  update(t1);
  exit(t1);
end;

Procedure insert(i,v:longint);
var
  t1,t2  :pnode;
begin
  if i>root^.len then i:=root^.len+1;
  split(root,i-1,t1,t2);
  new(root);
  root^.value:=v;
  root^.p:=nilt;
  setl(root,t1);
  setr(root,t2);
  update(root);
end;

Function query(i,j:longint):longint;
var
  t1,t2,t3:pnode;
begin
  split(root,j,t2,t3);
  split(t2,i-1,t1,t2);
  query:=t2^.max;
  root:=Join(Join(t1,t2),t3);
end;

Procedure Solution;
var
  c      :char;
  x,y    :longint;
  i      :longint;
begin
  assign(fi,tfi);reset(fi);
  assign(fo,tfo);rewrite(fo);
  readln(fi,n);
  for i:=1 to n do
    begin
      readln(fi,c,x,y);
      if c='A' then insert(y,x);
      if c='Q' then writeln(fo,query(x,y));
    end;
  close(fi);close(fo);
end;

begin
  InitTree;
  solution;
end.

```



Problem 2: Giá trị lớn nhất version 4

Mã bài: QMAX4 VNOL.INFO

Cho 1 tập S ban đầu không có phần tử nào. Máy tính sẽ đưa ra n lệnh có dạng sau :

- I x y : Chèn số x vào tập s giữa 2 vị trí y-1 và y ($-10^9 \leq x \leq 10^9$)

Quy định : Với k là số phần tử trong tập S , y = 1 thì x được chèn vào đầu dãy, y = k+1 thì x được chèn vào cuối dãy.

- S x y : Đổi chỗ vị trí 2 phần tử thứ x và y ($x \neq y, 1 \leq x, y \leq k$)
- D x : Xóa phần tử thứ x ra khỏi tập S ($1 \leq x \leq k$)
- Q x y : Tìm giá trị lớn nhất từ vị trí x đến vị trí y. ($1 \leq x \leq y \leq k$)

Yêu cầu : Cho n lệnh. Hãy trả lời mỗi truy vấn

Input

- Dòng đầu là số n ($n \leq 10^5$)
- N dòng tiếp theo là các lệnh có mẫu như trên

Output

Gồm một số dòng , mỗi dòng trả lời cho 1 truy vấn theo thứ tự từ trên xuống

Example

Input:

7
I 1 1
I 5 2
I 2 3
S 1 2
Q 1 2
D 1
Q 1 1

Output:

5
1

Solution:

Đây là một bài toán tổng quát hơn bài QMAX3VN với thêm hai thao tác là đổi chỗ và xóa. Ta vẫn dùng **Splay Tree** để biểu diễn danh sách.



Chương trình:

```
Const
  tfi='QMAX4.INP';
  tfo='QMAX4.OUT';

Type
  Pnode  =^TNode;
  TNode  =record
    value:longint;
    L,R,P:Pnode;
    len  :longint;
    max  :longint;
  end;

Var
  n      :longint;
  sentinel:TNode;
  nilt   :Pnode;
  root   :Pnode;
  fi,fo  :text;

Procedure InitTree;
begin
  sentinel.value:=-maxlongint;
  sentinel.max:=-maxlongint;
  sentinel.len:=0;
  nilt:=@sentinel;
  root:=nilt;
end;

Function Max(a,b:longint):longint;
begin
  if a>b then exit(a);
  exit(b);
end;

Procedure Update(x:Pnode);
begin
  x^.len:=x^.L^.len+x^.R^.len+1;
  x^.max:=Max(x^.value,Max(x^.L^.max,x^.R^.max));
end;

Function NodeAt(i:longint;x:Pnode):Pnode;
var
  sL      :longint;
begin
  repeat
    sL:=x^.L^.len+1;
    if sl=i then break;
    if sl>i then x:=x^.L
                else
                  begin
                    x:=x^.R;
                    i:=i-sl;
                  end;
  until false;
  exit(x);
end;

Procedure SetLink(x,y:Pnode;ok:boolean);
begin
  y^.P:=x;
  if ok then x^.R:=y
```



```

            else x^.L:=y;
end;

Procedure UpTree(x:Pnode);
var
    y,z      :Pnode;
begin
    y:=x^.P;
    z:=y^.P;
    if x=y^.L then
        begin
            SetLink(y,x^.R,False);
            SetLink(x,y,True);
        end
    else
        begin
            SetLink(y,x^.L,True);
            SetLink(x,y,False);
        end;
    SetLink(z,x,y=z^.R);
    Update(y);
    Update(x);
end;

Procedure Splay(x:Pnode);
var
    y,z      :Pnode;
begin
    Repeat
        y:=x^.P;
        if y=nilt then break;
        z:=y^.P;
        if z<>nilt then
            begin
                if (x=y^.L)=(y=z^.L) then UpTree(y)
                    else UpTree(x);
            end;
        UpTree(x);
    Until false;
end;

Procedure Split(T:Pnode;i:longint;var T1,T2:Pnode);
begin
    if i=0 then
        begin
            T1:=nilt;
            T2:=T;
        end
    else
        begin
            T1:=NodeAt(i,T);
            Splay(T1);
            T2:=T1^.R;
            T2^.P:=nilt;
            T1^.R:=nilt;
            Update(T1);
        end;
end;

Function Join(T1,T2:Pnode):Pnode;
begin
    if T1=nilt then exit(T2);
    while T1^.R<>nilt do T1:=T1^.R;
    Splay(T1);
    SetLink(T1,T2,True);
    Update(T1);

```



```

        exit(T1);
end;

Procedure Insert(i,v:longint);
var
  T1,T2  :Pnode;
begin
  if i>root^.len then i:=root^.len+1;
  Split(root,i-1,T1,T2);
  New(root);
  root^.value:=v;
  root^.P:=nilT;
  SetLink(root,T1,False);
  SetLink(root,T2,True);
  Update(root);
end;

Procedure Delete(i:longint);
var
  x,T1,T2 :Pnode;
begin
  x:=NodeAt(i,root);
  Splay(x);
  T2:=x^.R;
  T2^.P:=nilT;
  T1:=x^.L;
  T1^.P:=nilT;
  root:=Join(T1,T2);
end;

Procedure Updates(i,j:longint);
var
  x,y      :Pnode;
  v,u      :longint;
begin
  x:=NodeAt(i,root);
  y:=NodeAt(j,root);
  u:=x^.value;
  v:=y^.value;
  x^.value:=v;
  y^.value:=u;
  Splay(x);
  Update(x);
  Splay(y);
  Update(y);
  root:=y;
end;

Function Q(i,j:longint):longint;
var
  T1,T2,T3:Pnode;
begin
  Split(root,j,T2,T3);
  Split(T2,i-1,T1,T2);
  Q:=T2^.max;
  root:=Join(Join(T1,T2),T3);
end;

Procedure Solution;
var
  i,x,y  :longint;
  c       :char;
begin
  assign(fi,tfi);
  reset(fi);
  assign(fo,tfo);

```



```

rewrite(fo);
readln(fi,n);
for i:=1 to n do
begin
  read(fi,c);
  if c='I' then
  begin
    readln(fi,x,y);
    Insert(y,x);
  end;
  if c='S' then
  begin
    readln(fi,x,y);
    Updates(x,y);
  end;
  if c='D' then
  begin
    readln(fi,x);
    Delete(x);
  end;
  if c='Q' then
  begin
    readln(fi,x,y);
    writeln(fo,Q(x,y));
  end;
end;
close(fi);
close(fo);
end;

Begin
InitTree;
Solution;
End.

```



Problem 3: Tráo bài

Mã bài: CARDS VNOL.INFO

Cho bộ bài gồm n lá bài được xếp thành dây thứ tự từ 1 tới n, đầu tiên người ta ghi vào mỗi lá bài một số nguyên là số thứ tự ban đầu của lá bài đó. Xét phép tráo $S(i,m,j)$: Lấy ra khỏi bộ bài m lá liên tiếp bắt đầu từ lá bài thứ i, sau đó chèn m lá bài này vào trước lá bài thứ j trong số n-m lá bài còn lại $1 \leq i, j \leq n-m+1$. Quy ước nếu $j = n-m$ thì m lá bài lấy ra sẽ được đưa vào cuối dây

Ví dụ với $n = 9$

Bộ bài ban đầu : (1,2,3,4,5,6,7,8,9)

Thực hiện $S(1,5,2)$: (1,2,3,4,5,6,7,8,9) \rightarrow (6,1,2,3,4,5,7,8,9)

Thực hiện tiếp $S(5,4,6)$: (6,1,2,3,4,5,7,8,9) \rightarrow (6,1,2,3,9,4,5,7,8)

Thực hiện tiếp $S(8,2,1)$: (6,1,2,3,9,4,5,7,8) \rightarrow (7,8,6,1,2,3,9,4,5)

Yêu cầu : Hãy cho biết số ghi trên các lá bài sau khi thực hiện x phép tráo bài cho trước

Input

- Dòng 1 : Chứa 2 số nguyên n và x ($n, x \leq 10^5$)
- x dòng tiếp theo là bộ ba số nguyên i,m,j là một phép tráo $S(i,m,j)$

Output

Gồm 1 dòng duy nhất chứa n số nguyên. số thứ i là số ghi trên lá bài thứ i.

Example

Input:

9 3
1 5 2
5 4 6
8 2 1

Output:

7 8 6 1 2 3 9 4 5

Solution:

Ta thấy ở bài **CARDSHUF** ở dưới thì việc rút ra một lá bài và chèn vào một vị trí mới chỉ đơn giản là dùng 2 thủ tục Delete và Insert trên **BST**. Nhưng ở đây chúng ta cần rút một số lá bài liên tiếp, ta thấy tác dụng **Splay Tree** trong trường hợp này tốt hơn **Treap**, **AVL**, hay **RB Tree**. Ta dựa vào 2 thủ tục của **Splay Tree** là Split(Tách thành 2 BST con T1,T2) và Join(Ghép 2 BST con thành một BST).

Để rút ra m lá bài từ vị trí i, ta tách BST ban đầu thành 2 BST: T1,T2 sao cho T1:là BST biểu diễn i-1 vị trí đầu của lá bài con T2 biểu diễn các lá bài ở vị trí hiện tại. Sau đó lại tách T2 thành 2 BST con: T2,T3, T2: chứa m phần tử



đầu, T3: chứa đoạn còn lại, sau đó ghép 3 cây với nhau theo thứ tự T1,T3 rồi ghép T2.Việc cuối cùng là duyệt cây theo thứ tự giũa.

Chương trình:

```
Const
  tfi='CARDS.INP';
  tfo='CARDS.OUT';

Type
  PNode  =^TNode;
  TNode  =record
    value :longint;
    P,L,R :Pnode;
    len   :longint;
  end;

Var
  n,x      :longint;
  nilT     :Pnode;
  sentinel:TNode;
  root    :Pnode;
  fi,fo   :text;

Procedure Init;
begin
  sentinel.len:=0;
  sentinel.value:=0;
  nilT:=@sentinel;
  root:=nilT;
end;

Procedure Update(x:Pnode);
begin
  x^.len:=x^.L^.len+x^.R^.len+1;
end;

Procedure SetL(x,y:Pnode);
begin
  y^.P:=x;
  x^.L:=y;
end;

Procedure SetR(x,y:Pnode);
begin
  y^.P:=x;
  x^.R:=y;
end;

Function NodeAt(x:Pnode;i:longint):Pnode;
var
  sL      :longint;
begin
  repeat
    sL:=x^.L^.len+1;
    if sL=i then break;
    if sL>i then x:=x^.L
    else
      begin
        i:=i-sL;
        x:=x^.R;
      end;
  until false;
  exit(x);
end;
```



```

Procedure UpTree(x:Pnode);
var
    y, z      :Pnode;
begin
    y:=x^.P;
    z:=y^.P;
    if y^.L=x then
        begin
            SetL(y,x^.R);
            SetR(x,y);
        end
    else
        begin
            SetR(y,x^.L);
            SetL(x,y);
        end;
    if z^.L=y then SetL(z,x)
        else SetR(z,x);
    Update(y);
    Update(x);
end;

Procedure Splay(x:Pnode);
var
    y, z      :Pnode;
begin
    repeat
        y:=x^.P;
        if y=nilT then break;
        z:=y^.P;
        if z<>nilT then
            begin
                if (y=z^.L)=(x=y^.L) then UpTree(y)
                    else UpTree(x);
            end;
        UpTree(x);
    until false;
end;

Procedure Split(T:Pnode;i:longint;var T1,T2:Pnode);
begin
    if i=0 then
        begin
            T1:=nilT;
            T2:=T;
        end
    else
        begin
            T1:=NodeAt(T,i);
            Splay(T1);
            T2:=T1^.R;
            T1^.R:=nilT;
            T2^.P:=nilT;
            Update(T1);
        end;
end;

Function Join(T1,T2:Pnode):Pnode;
begin
    if T1=nilT then exit(T2);
    while T1^.R<>nilT do T1:=T1^.R;
    Splay(T1);
    SetR(T1,T2);
    Update(T1);
    exit(T1);

```



```

end;

Procedure Insert(i,v:longint);
var
  T1,T2  :Pnode;
begin
  if i>root^.len then i:=root^.len+1;
  Split(root,i-1,T1,T2);
  New(Root);
  root^.value:=v;
  root^.P:=nilT;
  SetL(root,T1);
  SetR(root,T2);
  Update(root);
end;

Procedure Query(i,m,j:longint);
var
  t1,t2,t3      :pnode;
begin
  Split(root,i-1,t1,t2);
  Split(t2,m,t2,t3);
  t1:=Join(t1,t3);
  Split(t1,j-1,t1,t3);
  root:=Join(join(t1,t2),t3);
end;

Procedure Visit(u:pnode);
begin
  if u<>nilT then
    begin
      Visit(u^.l);
      write(fo,u^.value,' ');
      Visit(u^.r);
    end;
end;

Procedure Solution;
var
  i,j,m,k :longint;
begin
  assign(fi,tfi);
  reset(fi);
  assign(fo,tfo);
  rewrite(fo);
  readln(fi,n,x);
  for i:=1 to n do Insert(i,i);
  for k:=1 to x do
    begin
      readln(fi,i,m,j);
      Query(i,m,j);
    end;
  Visit(root);
  close(fi);
  close(fo);
end;

begin
  Init;
  Solution;
end.

```



Problem 4: Cards shuffling

Mã bài: CARDHUF VNOL.INFO

Phú ông có bộ bài gồm n lá bài. Phú ông xếp chúng thành tập và ghi vào mỗi lá bài số thứ tự ban đầu của lá bài đó trong tập bài (vị trí các lá bài được đánh số từ 1 tới n từ trên xuống dưới).

Tiếp theo Phú ông tiến hành tráo tập bài, mỗi phép tráo kí hiệu bởi $S(i, j)$: rút ra lá bài thứ i và chèn lên trên lá bài thứ j trong số n - 1 lá bài còn lại ($1 \leq i, j \leq n$), quy ước rằng nếu $j = n$ thì lá bài thứ i sẽ được đặt vào vị trí cuối cùng của tập bài.

Ví dụ ($n=6$):

$$(1, \boxed{2}, 3, 4, 5, 6) \xrightarrow{S(2,3)} (1, 3, \boxed{2}, 4, 5, 6)$$

$$(\boxed{1}, 3, 2, 4, 5, 6) \xrightarrow{S(1,2)} (3, \boxed{1}, 2, 4, 5, 6)$$

$$(3, 1, 2, \boxed{4}, 5, 6) \xrightarrow{S(4,5)} (3, 1, 2, 5, \boxed{4}, 6)$$

$$(\boxed{3}, 1, 2, 5, 4, 6) \xrightarrow{S(1,6)} (1, 2, 5, 4, 6, \boxed{3})$$

Sau x phép tráo bài, Phú ông đưa cho Bờm tập bài và thách Bờm dùng ít phép tráo nhất để xếp lại các lá bài về vị trí ban đầu. Hãy giúp Bờm thực hiện điều đó.

Dữ liệu

- Dòng đầu tiên chứa hai số nguyên dương n, x.
- x dòng tiếp theo, dòng thứ p chứa hai số nguyên i_p, j_p cho biết phép tráo thứ p của Phú ông là $S(i_p, j_p)$.

Kết quả

- Một số duy nhất là số phép tráo cần thực hiện để đưa các lá bài về vị trí ban đầu.

Ví dụ

Dữ liệu:

6 4
2 3
1 2
4 5
1 6

Kết quả:

2

Giới hạn

$1 \leq n, x \leq 10^5$.

Solution:

Giả sử sau x lần tráo bài ta được trạng thái bộ bài như sau: x_1, x_2, \dots, x_n .

Như vậy để đưa bộ bài về ban đầu ta làm tương tự bài **MCARDS** tìm dãy con tăng dài nhất của dãy x gọi là l_{max} .



⇒ Số thao tác ít nhất sẽ bằng: n-lmax.

Vấn đề là $1 \leq x, n \leq 10^5$, ta sẽ sử dụng **BST**(Binary Search Tree) cân bằng mà cụ thể là cây **Splay** biểu diễn danh sách để xác định trạng thái của bộ bài sau x lần tráo trong thời gian **O(nlogn)**.

Sau đó làm tương tự bài **LIS** tìm độ dài dãy con tăng dài nhất trong thời gian **O(nlogn)**.

Các thao tác chính trên cây **Splay** là **Delete(i)** xóa quân bài thứ i ra khỏi bộ bài, **Insert(i,v)** chèn vào vị trí i của bộ bài quân bài v.

Chương trình:

```
Const
    fi='CARDSHUF.INP';
    fo='CARDSHUF.OUT';
    maxn=100000;

Type
    Pnode = ^Tnode;
    Tnode = record
        value : longint;
        l,p,r : Pnode;
        len : longint;
    end;
    arr1 = array[0..maxn+1] of longint;

Var
    n : longint;
    root : Pnode;
    nilt : Pnode;
    sentinel:tnode;
    a : Trrl;
    x : longint;
    kq : longint;
    Node : longint;
    f : text;

{$R-}

Procedure InitTree;
begin
    sentinel.value:=0;
    sentinel.len:=0;
    nilt:=@sentinel;
    root:=nilt;
end;

Procedure Update(x:pnode);
begin
    x^.len:=x^.L^.len+x^.R^.len+1;
end;

Procedure SetL(x,y:pnode);
begin
    y^.P:=x;
    x^.L:=y;
end;

Procedure SetR(x,y:pnode);
begin
    y^.P:=x;
    x^.R:=y;
end;

function NodeAt(i:longint;x:pnode):pnode;
var
    sl : longint;
```



```

begin
repeat
  sl:=x^.L^.len+1;
  if sl=i then break;
  if sl>i then x:=x^.L
    else
      begin
        i:=i-sl;
        x:=x^.R;
      end;
  until false;
exit(x);
end;

Procedure UpTree(x:Pnode);
var
  y,z      :Pnode;
begin
  y:=x^.P;
  z:=y^.P;
  if x=y^.L then
    begin
      SetL(y,x^.R);
      SetR(x,y);
    end
  else
    begin
      SetR(y,x^.L);
      SetL(x,y);
    end;
  if y=z^.L then SetL(z,x) else SetR(z,x);
  Update(y);
  Update(x);
end;

Procedure Splay(x:Pnode);
var
  y,z      :Pnode;
begin
repeat
  y:=x^.P;
  if y=nilt then break;
  z:=y^.P;
  if z<>nilt then
    begin
      if (x=y^.L)=(y=z^.L) then UpTree(y)
        else UpTree(x);
    end;
  UpTree(x);
until false;
end;

Procedure Split(t:Pnode;i:longint;var t1,t2:Pnode);
begin
  if i=0 then
    begin
      t1:=nilt;
      t2:=t;
    end
  else
    begin
      t1:=NodeAt(i,t);
      Splay(t1);
      t2:=t1^.R;
      t2^.P:=nilt;
      t1^.R:=nilt;
    end;
end;

```



```

        Update(t1);
    end;
end;

function Join(t1,t2:Pnode):Pnode;
begin
    if t1=nilt then exit(t2);
    while t1^.R<>nilt do t1:=t1^.R;
    Splay(t1);
    SetR(t1,t2);
    Update(t1);
    exit(t1);
end;

Procedure Insert(i,v:longint);
var
    t1,t2:pnode;
begin
    if i>root^.len then i:=root^.len+1;
    Split(root,i-1,t1,t2);
    new(root);
    root^.value:=v;
    root^.P:=nilt;
    SetL(root,t1);
    SetR(root,t2);
    Update(root);
end;

Procedure Delete(i:longint);
var
    x,t1,t2:Pnode;
begin
    x:=NodeAt(i,root);
    Splay(x);
    Node:=x^.value;
    t1:=x^.L;t1^.P:=nilt;
    t2:=x^.R;t2^.P:=nilt;
    Dispose(x);
    root:=Join(t1,t2);
end;

Procedure InorderTravelsal(x:Pnode);
begin
    if x<>nilt then
        begin
            InorderTravelsal(x^.L);
            inc(n);
            a[n]:=x^.value;
            InorderTravelsal(x^.R);
        end;
end;

Procedure Nhap;
var
    u,v      :longint;
    i         :longint;
begin
    assign(f,fi);
    reset(f);
    readln(f,n,x);
    for i:=1 to n do insert(i,i);
    for i:=1 to x do
        begin
            readln(f,u,v);
            delete(u);
            Insert(v,Node);

```



```

        end;
n:=0;
close(f);
end;

Procedure Solution;
var
H      :arr1;
i,k    :longint;

Function Search(x:longint):longint;
var
i,j,k  :longint;
begin
i:=0;
j:=kq;
While i<>j do
begin
k:=(i+j) div 2;
if a[h[k]]<x then i:=k+1
else j:=k;
end;
if i>kq then i:=kq;
while a[h[i]]>x do dec(i);
exit(i);
end;

Begin
kq:=0;
for i:=1 to n do h[i]:=n+1;
a[n+1]:=maxlongint;
a[0]:=-maxlongint;
h[0]:=maxlongint;
for i:=1 to n do
begin
k:=Search(a[i]);
if a[h[k+1]]>=a[i] then h[k+1]:=i;
if kq<k+1 then kq:=k+1;
end;
kq:=n-kq;
end;

Procedure Xuat;
begin
assign(f,fo);
rewrite(f);
write(f,kq);
close(f);
end;

Begin
InitTree;
Nhap;
InorderTravelsal(root);
Solution;
Xuat;
End.

```

Độ phức tạp: $O(n \log n)$



Bài tập tự luyện:

Problem 1: Chiến trường Ô qua

Mã bài: KAGAIN VNOL.INFO

Lại nói về Lục Vân Tiên , sau khi vượt qua vòng loại để trở thành Tráng Sỹ , anh đã gặp được Đôrêmon và được chú mèo máy cho đi qua giang về thế kỷ 19 . Trở lại quê hương sau nhiều năm xa cách , với tấm bằng Tráng Sỹ hạng 1 do Liên Đoàn Type Thuật cấp , anh đã được Đức Vua cử làm đại tướng thống lĩnh 3 quân chống lại giặc Ô Qua xâm lăng . Đoàn quân của anh sẽ gồm N đại đội , đại đội i có $A[i] (> 0)$ người . Quân sỹ trong 1 đại đội sẽ đứng thành 1 cột từ người 1 -> người $A[i]$, như vậy binh sỹ sẽ đứng thành N cột . Vì Vân Tiên quyết 1 trận sẽ đánh bại quân Ô Qua nên đã cử ra 1 quân đoàn hùng mạnh nhất . Trong sự cũ chép rằng , quân đoàn của Vân Tiên cử ra lúc đó là một nhóm các đại đội có chỉ số liên tiếp nhau (tức là đại đội i , $i + 1 , \dots , j$) . Vì sự sách thì mỗi một hết cả nên chỉ biết được mỗi thé . Ngoài ra theo giang hồ đồn đại thì sức mạnh của 1 quân đoàn = số người của đại đội ít người nhất * số đại đội được chọn . Nhiệm vụ của bạn là dựa trên các thông số của các nhà khảo cổ có được , hãy cho biết quân đoàn mà Vân Tiên đã chọn ra là từ đại đội nào đến đại đội nào . Chủ ý nếu có nhiều phương án thì ghi ra phương án mà chỉ số của đại đội đầu tiên được chọn là nhỏ nhất .

Input

Dòng 1 : Số T là số bộ test .

T nhóm dòng tiếp theo , mỗi nhóm dòng mô tả 1 bộ test . Nhóm dòng thứ i :

Dòng 1: N ($<= 30000$)

Dòng 2: N số nguyên mô tả N số $A[1], A[2], \dots, A[N]$ (các số nguyên dương $<= 30000$).

Output

Kết quả mỗi test ghi ra trên 1 dòng , gồm 3 số : sức mạnh quân đoàn mạnh nhất , chỉ số của đại đội đầu tiên và chỉ số của đại đội cuối cùng được chọn .

Example

Input:

```
2
4
3 4 3 1
4
1 2 1 3
```

Output:

```
9 1 3
4 1 4
```

Gợi ý: Tương tự bài **KPLANK**.

Problem 2: Những con đường quanh nông trang

Tương tự **Problem 2** ở trên nhưng sử dụng Stack.

Problem 3: Hình vuông 0 1



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

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

Yêu cầu:

Hãy tìm một hình vuông gồm các ô của bảng thỏa mãn các điều kiện sau:

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

2 - Cạnh hình vuông song song với cạnh bảng.

3 - Kích thước hình vuông là lớn nhất có thể

Input

Dòng 1: Ghi hai số m, n

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

Output

Gồm 1 dòng duy nhất ghi kích thước cạnh của hình vuông tìm được

Example

Input:

```
11 13
0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 1 1 1 0 0 0 0
0 1 1 1 1 1 1 1 1 1 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 0 0
0 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 1 1 1 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 1 1
0 0 0 0 0 1 0 0 0 0 0 1 1
```

Output:

7

Gọi ý: Dùng mảng pre[i] với ý nghĩa là cột xa nhất mà $h[pre[i]] \geq h[i]$ với $h[i]$ là độ dài cột i toàn chữ số 1 kết thúc tại i.



Problem 4: Hoán vị dài nhất

Mã bài: NKLP VNOL.INFO

Cho dãy A gồm N phần tử A_1, A_2, \dots, A_N là các số nguyên. Một dãy con của dãy A là dãy gồm các phần tử liên tiếp A_U, A_{U+1}, \dots, A_V trong đó $1 \leq U \leq V \leq N$. Một dãy con B có độ dài K của A được coi là đáng quan tâm nếu dãy B là một hoán vị của K số $1, 2, \dots, K$.

Nhiệm vụ của bạn là tìm một dãy con đáng quan tâm dài nhất của A.

Dữ liệu

- Dòng thứ nhất ghi số N là số phần tử của dãy A.
- Dòng thứ hai ghi N số A_1, A_2, \dots, A_N .

Kết quả

Một số duy nhất là độ dài lớn nhất tìm được.

Giới hạn

- $1 \leq N \leq 100\,000$.
- $1 \leq A_U \leq N$.

Ví dụ

Dữ liệu:

5
4 1 2 1 3

Kết quả

3

Problem 5: BALL GAME

Mã bài: BALLGAME VNOL.INFO

Hôm qua AnhDQ mời cô gái của mình đi ăn món gà KFC (vốn là sở trường của hai người). Hai người quyết định gọi phần ăn Chicky (dành cho trẻ em :d), vừa tiết kiệm, mà còn được tặng kèm món đồ chơi ngộ nghĩnh có tên là "Bắt gà" :d Sau một hồi loay hoay, AnhDQ phát hiện ra rằng trò chơi của bọn trẻ con nước ngoài này khá là giống với trò chơi gụ của trẻ con Việt Nam :d (tất nhiên là hiện đại hơn ^^). Sau đó AnhDQ quyết định rủ nàng đi xem phim ở MegaStar :> trên đường ghé qua khu StarBowl, cầm trên tay món đồ "Bắt gà", AnhDQ chợt nảy ra một trò chơi thú vị kết hợp giữa những quả bóng bowling và trò chơi gụ dân gian, và chàng quyết định hai nhóc Alice và Bob sẽ là những vị khách đầu tiên tham gia trò chơi mới này! Chàng tạm đặt tên cho trò chơi là "Trò chơi với những trái bóng":

AnhDQ làm sẵn một cái máng có N lỗ xếp liền nhau được đánh số từ 1..N và một đường rãnh giúp đưa các quả bóng vào các lỗ này. Đường rãnh này có thiết kế đặc biệt như sau:

- Chỉ có thể ném bóng vào từ một trong hai đầu của đường rãnh.
- Tại một thời điểm chỉ có một quả bóng được ném vào đường rãnh.



- Nếu bóng được ném vào từ đâu nào thì đường rãnh sẽ đưa quả bóng tới lỗ **trống gần đầu đó nhất**.
- Thời gian đưa bóng tới một lỗ bắt kì là không đáng kể.

AnhDQ làm ra M quả bóng đặc biệt cho Alice và Bob chơi, những quả bóng này có đặc điểm như sau:

- Chúng có cùng hình dạng, kích thước; vậy nên có thể rơi vào bất kì lỗ nào trên máng.
- Mỗi quả bóng có một độ xoáy (có thể giống hoặc khác nhau); nếu một quả bóng có độ xoáy là x thì nó sẽ xoáy trên miệng lỗ mà nó rơi vào trong x phút trước khi rơi hẳn xuống lỗ.

Lại nói về cái máng:

- Mỗi lỗ trên máng có sức chứa vô hạn.
- Một lỗ được gọi là **trống** nếu không có quả bóng nào đang xoáy trên miệng lỗ đó; tại thời điểm một quả bóng dừng việc xoáy và bắt đầu rơi vào lỗ thì lỗ đó cũng được coi là trống.

- Trò chơi bắt đầu ở thời điểm 0, Alice đứng ở đầu rãnh phía bên trái (gần lỗ thứ nhất) còn Bob đứng ở đầu rãnh phía bên phải (gần lỗ thứ N).

- Tại các thời điểm khác nhau, Alice hoặc Bob ném một trong M quả bóng vào rãnh.

- Trò chơi kết thúc khi một trong các trường hợp sau xảy ra:

++ Alice và Bob ném hết M quả bóng; khi đó hai nhóc **hòa nhau**; thời gian của ván chơi là thời điểm mà quả bóng cuối cùng ngừng xoáy và rơi xuống lỗ.

++ Alice hoặc Bob ném một quả bóng vào rãnh trong lúc không có lỗ nào trên máng còn trống; khi đó nhóc nào ném quả bóng đó sẽ là người **thua cuộc** ;d; thời gian của ván chơi là thời điểm mà nhóc thua cuộc ném quả bóng đó.

AnhDQ đi chơi về và thấy hai nhóc đang ngồi thở hổn hển vì ném bóng :)) AnhDQ liền hỏi hai nhóc xem ai thắng thì hai nhóc cười toe toét nói là.. không nhớ @) Bó tay, AnhDQ liền gặng hỏi chúng về diễn biến ván chơi, hai nhóc chanh chòe nhau kể ra các lần ném bóng của mình, được mô tả bởi hai số u, v, trong đó u là thời điểm ném quả bóng và v là độ xoáy của quả bóng đó.

Nhưng chúng kể cũng chẳng có thứ tự gì hết, càng làm AnhDQ đau đầu hơn, chàng cố ghi lại thông tin mà chúng kể và nhờ các VOJ-er(s) tài giỏi giúp chàng phân định kết quả của ván chơi xem sao. Chàng rất hồi hộp về trò chơi mới này của mình.

Dữ liệu

- Dòng đầu tiên chứa hai số N, M.
- M dòng tiếp theo, mỗi dòng bắt đầu bởi một chữ cái 'A' hoặc 'B', tương ứng với lời kể của Alice hoặc Bob; tiếp theo là hai số u, v tương ứng.

Kết quả

- Nếu ván chơi kết thúc với kết quả hòa, in ra như sau:

++ Dòng đầu tiên in ra **DRAW**.

++ Dòng thứ hai in ra **Game lasts: T minute(s)**; trong đó T là thời gian của ván chơi.

++ M dòng tiếp theo in ra "biên bản" của ván chơi; mỗi dòng ghi **Alice takes the hole: H** hoặc **Bob takes the hole: H** tương ứng với một lượt ném bóng của Alice hoặc Bob; trong đó H là lỗ trống tương ứng mà quả bóng của lượt ném đó rơi vào. Biên bản này viết ra theo **thứ tự thời gian**.

- Nếu ván chơi kết thúc với một nhóc thua cuộc, in ra như sau:

++ Dòng đầu tiên in ra **Alice loses at her turn: R** nếu Alice thua cuộc, hoặc in ra **Bob loses at his turn: R** nếu Bob thua cuộc; với ý nghĩa: nhóc tương ứng thua cuộc sau lần ném thứ R của mình.

++ Dòng thứ hai in ra **Game lasts: T minute(s)**; trong đó T là thời gian của ván chơi.

Ví dụ



Dữ liệu:

2 2
A 1 10
B 2 20

Kết quả:

DRAW

Game lasts: 22 minute(s)

Alice takes the hole: 1

Bob takes the hole: 2

Dữ liệu:

1 2
A 1 10
B 2 20

Kết quả:

Bob loses at his turn: 1

Game lasts: 2 minute(s)

Giới hạn

- $N, M \leq 10^5$.
- $u, v \leq 10^9$.

Problem 6: BALL GAME 2**Mã bài: BALLGAME2 VNOL.INFO**

(Nguồn gốc từ bài BALLGAME)

Cho một dãy các lỗ có sức chứa vô hạn được đánh số từ 1 đến N. Trò chơi tiến hành như sau:

_ Tại thời điểm nào đó, một người đứng ở vị trí x bắt kè và ném bóng.

_ Quả bóng được ném đi sẽ lọt vào lỗ **trống** gần nó nhất

_ Mỗi quả bóng có một độ xoáy (có thể giống hoặc khác nhau); nếu một quả bóng có độ xoáy là x thì nó sẽ xoáy trên miệng lỗ mà nó rơi vào trong x phút trước khi rơi hẳn xuống lỗ.

_ Một ô được định nghĩa là trống khi không có quả bóng nào đang xoáy trên miệng lỗ đó, nếu tại một lỗ có một quả bóng đang xoáy thì thời điểm quả bóng ngừng xoáy cũng là thời điểm lỗ bắt đầu trống.

Cho các truy vấn có dạng T X Y với T là thời điểm ném bóng, X là vị trí ném bóng, và Y là độ xoáy. Với mỗi truy vấn yêu cầu in ra vị trí của lỗ trống gần vị trí X nhất (nếu X trống thì tất nhiên vị trí này sẽ là X). Nếu tồn tại 2 ô trống đều gần X nhất thì in ra vị trí bên trái.

Input

_ Dòng đầu chứa 2 số N và M (với N là số lỗ và M là số truy vấn) ($N \leq 100000; M \leq 100000$)



_ M dòng sau mỗi dòng chứa 2 số T X Y với ý nghĩa như trên. (lưu ý là các truy vấn dc cung cấp theo thứ tự thời gian)
($0 \leq T, Y \leq 10^9$; $1 \leq X \leq N$)

Output

_ Với mỗi truy vấn, in ra trên một dòng vị trí lỗ trống mà quả bóng đó sẽ rơi vào, nếu ko tồn tại vị trí nào thì in ra 0

Example

Input:

```
5 7  
3 3 7  
7 4 8  
10 5 4  
13 4 3  
15 2 8  
16 5 2  
18 3 7
```

Output:

```
3  
4  
5  
3  
2  
5  
3
```

Problem 7: BALL GAME 3

[Mã bài: BALLGAME3 VNOL.INFO](#)

Cho N quả bóng (đánh số 1..N) có màu được đánh dấu bởi một trong các kí tự thuộc bảng mã ASCII, xếp thành một hàng. Bạn hãy viết một chương trình làm các công việc sau:

1. Lấy ra một đoạn liên tiếp dài nhất các quả bóng có cùng màu, nếu có nhiều đoạn thỏa mãn thì lấy đoạn trái nhất.
2. Nếu đoạn lấy ra có số lượng bóng lớn hơn 1 thì in ra trên một dòng theo định dạng:
 $C P_1 P_2 P_3 \dots$
trong đó C là màu của đoạn bóng được lấy ra, tập P gồm chỉ số của các quả được lấy ra theo thứ tự tăng dần; giữa C và các P_i cách nhau bởi đúng 1 dấu trống và không có dấu trống hay kí tự thừa.
3. Nếu đoạn lấy ra có số lượng bóng bằng 1 thì kết thúc chương trình.
4. Nếu dãy còn lại rỗng thì kết thúc chương trình.
5. Nếu dãy còn lại không rỗng thì tiến hành ghép phần bên trái và bên phải đoạn vừa lấy ra tạo thành dãy mới, các quả bóng được giữ nguyên chỉ số ban đầu.
6. Lặp lại bước 1.

Yêu cầu

Viết một chương trình thỏa mãn lưu đồ trên, đáp ứng thời gian thực hiện tỉ lệ thuận với kích thước của output đáp án.



Dữ liệu

- Gồm một dòng duy nhất chứa N kí tự viết liên tiếp thể hiện dãy bóng ban đầu.

Kết quả

In ra như hướng dẫn của đề bài.

Ví dụ

Dữ liệu:

XDDTVXXXDVVVVD

Kết quả:

X 6 7 8
V 10 11 12
D 2 3
D 9 13

Giới hạn

- $N \leq 500,000$.

Problem 8: Tham Ăn

Mã bài: SMATHDOG VNOL.INFO

Tiếp theo chiến lược “quy hoạch động”, Bòm huấn luyện cho chú chó của mình chiến lược “tham ăn” trong một sân chơi được biểu diễn bởi mặt phẳng trực chuẩn. Ban đầu chú chó xuất phát ở điểm $(0,0)$ và nó phải đứng im cho tới khi được gọi. Trò chơi diễn ra trong N lượt, lượt thứ i của trò chơi diễn ra như sau:

Bòm di chuyển đến vị trí (x_i, y_i) , cầm c_i cái bánh và gọi chú chó. Chú chó có quyền đứng im hoặc di chuyển theo các phương song song theo trục tọa độ để đến chỗ Bòm nếu độ dài quãng đường di chuyển không vượt quá K . Nếu chú chó có thể di được đến chỗ Bòm và quyết định di chuyển, nó sẽ được thưởng toàn bộ c_i cái bánh, ngược lại nó sẽ phải đứng nhìn Bòm ăn hết luôn c_i cái bánh đó. Hết lượt chơi này chú chó lại phải đứng im và trò chơi tiếp tục ở lượt $i+1$.

Yêu cầu: Cho biết trước các tọa độ (x_i, y_i) và số bánh c_i tại các lượt chơi, hãy giúp chú chó tội nghiệp của Bòm kiếm được nhiều bánh nhất

Input

- Dòng 1 chứa hai số nguyên dương N, K .
- N dòng tiếp theo, dòng thứ i chứa ba số nguyên dương x_i, y_i, c_i .
- Ràng buộc: $N \leq 10^5$, tất cả các số còn lại trong file dữ liệu đều là số nguyên dương $\leq 10^3$.

Output

- Gồm một số nguyên duy nhất là số bánh chó kiếm được trong trò chơi theo phương án của bạn.



Example

Input:

8 4

1 2 2

1 5 9

4 3 3

6 4 4

7 7 8

8 2 5

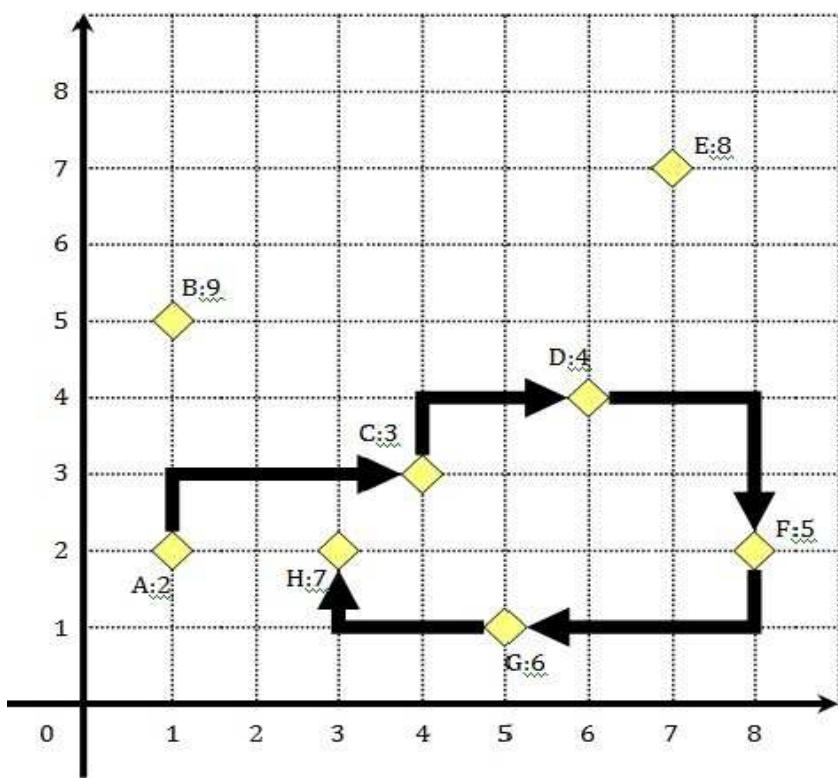
5 1 6

3 2 7

Output:

27

Đây là ví dụ với 8 lượt chơi và vị trí của Bờm tại 8 lượt chơi đó lần lượt là A, B, C, D, E, F, G, H.
Trong phương án tối ưu chó chỉ phải bỏ 9 bánh tại điểm B và 8 bánh tại điểm E



Problem 9: Số lượng dãy con tăng

Mã bài: NTSEQ VNOI.INFO

Cho dãy số $a_1, a_2, a_3, \dots, a_n$. Hãy đếm số lượng dãy con tăng dài nhất của dãy số trên.

Một dãy con độ dài k của dãy a được xác định bởi một bộ chỉ số $(u_1 \leq u_2 \leq u_3 \leq \dots \leq u_k)$ ($1 \leq u_i \leq n$). Hai dãy con $(u_1, u_2, u_3, \dots, u_k)$ và $(v_1, v_2, v_3, \dots, v_t)$ được gọi là khác nhau nếu $k \neq t$ hoặc tồn tại một vị trí i sao cho $u_i \neq v_i$.

Input

- Dòng đầu tiên ghi số nguyên dương n ($n \leq 10^5$)
- Dòng tiếp theo ghi n số nguyên mô tả dãy a ($a_i \leq 10^9$)

Output

- Một dòng duy nhất ghi kết quả theo module 1000000007

Example

Input:

6
1 1 2 2 3 3

Output:

8

Problem 10: Bus

Mã bài: NKBUS VNOI.INFO

Một xe buýt của công ty có nhiệm vụ đón nhân viên đến trụ sở làm việc. Trên hành trình, xe buýt sẽ tiếp nhận nhân viên đứng chờ ở các điểm hẹn nếu như xe còn chỗ trống. Xe buýt có thể đỗ lại để chờ những công nhân chưa kịp đến điểm hẹn.

Cho biết thời điểm mà mỗi nhân viên đến điểm hẹn của mình và thời điểm qua mỗi điểm hẹn của xe buýt. Giả thiết rằng xe buýt đến điểm hẹn đầu tiên tại thời điểm 0 và thời gian xếp khách lên xe được bằng 0.

Xe buýt cần phải chờ một số lượng nhiều nhất các nhân viên có thể được đến trụ sở. Hãy xác định khoảng thời gian ngắn nhất để xe buýt thực hiện công việc.

Dữ liệu vào

Dòng đầu tiên chứa 2 số nguyên dương n, m theo thứ tự là số điểm hẹn và số chỗ ngồi của xe buýt

Dòng thứ i trong số n dòng tiếp theo chứa số nguyên t_i là thời gian cần thiết để xe buýt di chuyển từ điểm hẹn thứ i đến điểm hẹn thứ $i+1$ (điểm hẹn thứ $n+1$ sẽ là trụ sở làm việc của công ty) và số nguyên k là số lượng nhân viên đến điểm hẹn i , tiếp theo k số nguyên là các thời điểm đến điểm hẹn của k nhân viên.

Kết quả



Gồm một dòng duy nhất, là thời gian ngắn nhất tìm được.

Giới hạn

$1 \leq n \leq 200000$, $1 \leq m \leq 20000$

Tổng số nhân viên không vượt quá 200000.

Kết quả không vượt quá $2^{31}-1$.

Ví dụ

Dữ liệu mẫu

3 2
3 2 4 3
1 3 6 3 7
5 1 5

Kết quả

10

Giải thích: Trên đường đến công ty có 3 trạm xe buýt. Từ trạm 1 đến trạm 2, trạm 2 đến trạm 3, và từ trạm 3 đến công ty lần lượt mất 3, 1 và 5 đơn vị thời gian. Xe buýt có thể đi như sau: đến thẳng trạm 2, đón người thứ 2, đến trạm 3, chờ 1 đơn vị thời gian để đón người duy nhất ở trạm này, và cuối cùng đến công ty. Tổng cộng xe buýt đi mất $3 + 1 + 1 + 5 = 10$ đơn vị thời gian.

Problem 11: Even Palindrome

Mã bài: PALDR VNOL.INFO

Xâu Palindrome là một chuỗi kí tự có tính chất là đọc giống nhau theo cả 2 chiều (trái qua phải hoặc phải qua trái). Bạn cần xác định xem một chuỗi kí tự cho trước có thể được biểu diễn dưới dạng ghép của một số xâu Palindrome độ dài chẵn hay không.

Lưu ý: Một chuỗi kí tự có thể được biểu diễn bởi dạng ghép của một số bất kỳ các xâu Palindrome độ dài chẵn.

Input

Dòng đầu chứa T, số lượng bộ test. T dòng tiếp theo, mỗi dòng chứa một chuỗi kí tự tương ứng với bộ test đó.

Output

Chứa đúng T dòng, mỗi dòng cho một bộ test tương ứng. Bạn phải in ra "YES" nếu chuỗi kí tự có thể được biểu diễn bởi ghép của các xâu Palindrome độ dài chẵn, hoặc in ra "NO" trong trường hợp ngược lại. (Không in dấu ngoặc kép).

Example

Input:

3
madam
aA



aabb

Output:

NO
NO
YES

Constraints

Độ dài chuỗi $\leq 10^6$

Problem 12: Sequences

Mã bài: SPSEQ VNOL.INFO

W. là 1 dãy các số nguyên dương. Nó có các đặc điểm sau:

- Độ dài của dãy là 1 số lẻ: $L = 2*N + 1$
- $N + 1$ số nguyên đầu tiên của dãy tạo thành 1 dãy tăng
- $N + 1$ số nguyên cuối của dãy tạo thành 1 dãy giảm
- Không có 2 số nguyên nào cạnh nhau trong dãy có giá trị bằng nhau

Ví dụ: **1, 2, 3, 4, 5, 4, 3, 2, 1** là 1 dãy W. độ dài **9**. Tuy nhiên, dãy **1, 2, 3, 4, 5, 4, 3, 2, 2** không là 1 dãy W.

Yêu cầu: Trong các dãy con của dãy số cho trước, tìm dãy W. có độ dài dài nhất.

Input

Dòng 1: số nguyên dương N ($N \leq 100000$), độ dài dãy số.

Dòng 2: N số nguyên dương a_i ($a_i \leq 10^9$).

Output

1 số nguyên dương duy nhất là độ dài dãy W. dài nhất.

Example

Input:

10
1 2 3 4 5 4 3 2 1 10

Output:

9

Input:

19
1 2 3 2 1 2 3 4 3 2 1 5 4 1 2 3 2 2 1



Output:

9

Problem 13: Tháp Hà Nội

Mã bài: CHNTOWER VNOL.INFO

Bài toán Tháp Hà Nội trở thành nổi tiếng vào năm 1883, sau bài báo của Luca là một nhà toán học người Pháp. Tháp là mỗi cọc đĩa đường kính giảm dần từ dưới lên trên. Bài toán đặt ra là cần chuyển chòng đĩa sang một cọc khác sử dụng một cọc trung gian sao cho trong quá trình chuyển đĩa không có đĩa nào có đường kính lớn hơn bị đặt lên trên đĩa có đường kính nhỏ hơn.

Yêu cầu: Giải 3 toán tháp Hà Nội tổng quát. Cho M cọc và tháp N đĩa ($3 < M \leq 64$, $1 \leq N \leq 64$), hãy xác định số lần chuyển đĩa tối thiểu cần thực hiện để chuyển chòng đĩa từ cọc xuất phát sang cọc đích sử dụng M-2 cọc còn lại như cọc trung gian.

Input

Gồm nhiều dòng, mỗi dòng chứa 2 số nguyên N,M ghi cách nhau theo thứ tự là số đĩa và số cọc trong bài toán tháp Hà Nội

Output

Mỗi dòng ghi số lần chuyển tối thiểu cần thực hiện

Example**Input:**

5 3

Output:

31



Một số đề thi tuyển chọn

Pirate contests 1 Problem

Bài 1: Đì chơi

Mã bài: KTOUR

Điều tối kỵ khi có nhiều bạn gái là gì? Là để họ gặp nhau...

Tuy nhiên, vào một ngày nọ tai họa bỗng nhiên ập xuống đầu Pirate khi tất cả N cô bạn gái của anh ta cùng một lúc muốn đi chơi với anh chàng. Dĩ nhiên là anh ấy không bao giờ bó tay chịu bị "ngũ mã phanh thây", nên đã sắp xếp hẹn mỗi em trên một hòn đảo khác nhau. Các hòn đảo này có vị trí rất đặc biệt. Nếu nhìn vào bản đồ biển (là hệ trực tọa độ xy), thì các hòn đảo này là các điểm nằm trên một đường thẳng song song với trực hoành.

Giờ điều duy nhất là Pirate bắn khoăn là dắt các em đi đâu đây nhỉ? Có M chốn hẹn hò lãng mạn trên vùng biển này, mỗi nơi được biểu diễn bởi một điểm trên bản đồ. Pirate muốn "giữ sức" và "giữ tiền" để có đi chơi hết với mọi em. Thế anh quyết định là sẽ dẫn mỗi em đến nơi hẹn hò gần nhất với hòn đảo họ đang đứng. Sau khi xếp lịch xong, Pirate tự hỏi mình phải đi quãng đường xa nhất là bao nhiêu để chuẩn bị tiền đồ xăng (đã qua rồi cái thời cướp biển và cảnh buồn phiêu du)?

Input

- Dòng thứ nhất: hai số nguyên N - số hòn đảo hẹn hò, và Y0 - tọa độ y của các hòn đảo.
- N dòng tiếp theo: mô tả tọa độ x của các hòn đảo.
- Dòng thứ N + 2: số nguyên M - số địa điểm hẹn hò lãng mạn.
- M dòng tiếp theo: mô tả tọa độ (x, y) của các nơi hẹn hò.

Output

- Một dòng duy nhất ghi ra khoảng cách lớn nhất mà Pirate phải đi từ một hòn đảo đến nơi hẹn hò tương ứng (làm tròn đến 6 chữ số thập phân sau dấu chấm).

Giới hạn

- Mọi số trong input đều là số nguyên không âm và không vượt quá 10^5 .
- 60% số test có $1 \leq N, M \leq 1000$.

Example



Input:

```
2 1  
1  
2  
2  
0 2  
1 0
```

Output:

```
1.414214
```

Giải thích: Có 2 hòn đảo nằm ở $(1, 1)$ và $(2, 1)$ và 2 địa điểm hẹn hò ở $(0, 2)$ và $(1, 0)$. Vì Pirate chỉ chọn nơi hẹn hò gần nhất với hòn đảo đang đứng, nên từ hai hòn đảo, anh ấy đều đi đến nơi hẹn hò đầu tiên với khoảng cách lần lượt là 1 và 1.414214. Đáp án của bài toán là khoảng cách lớn nhất 1.414214.

Bài 2: Sửa cầu

Mã bài: KBUILD

Vì lo lắng Pirate sẽ buồn chán khi một thân một mình ở đảo hoang, bạn gái Pirate từ trong đất liền dự định sang chơi với anh ấy.

Pirate đang sinh sống trên một quần đảo gồm N đảo. Vì các đảo khá gần nhau nên chẳng cần thuyền bè gì, Pirate chỉ cần đốn đại cây dừa nào đó và bắc ngang là có thể đi được từ đảo này sang đảo khác. Vì không muốn hủy hoại môi trường nên anh ấy chỉ đốn N - 1 cây dừa làm cầu, vừa đủ để từ một đảo bất kì đi đến được hết mọi đảo còn lại.

Nhưng mà "Môi son, má đào, chân guốc cao gót làm sao em qua cầu dừa???". Lo lắng sợ bạn gái sẽ rơi xuống biển và bị cá đuối nắn nát, Pirate hộc tốc đi sửa chữa các cây cầu dừa. Anh đưa ra một lịch trình như sau: vào mỗi ngày sẽ đi kiểm tra mọi cây cầu trên đường đi từ đảo a đến đảo b.

Tuy nhiên, lịch trình đó khá là phi khoa học. Thực hiện, xong rồi, Pirate mới ngớ ra là không biết mình có bỏ sót cây cầu nào không?

Input

- Dòng thứ nhất: số nguyên N - số hòn đảo.
- N - 1 dòng tiếp theo: mỗi dòng gồm hai số nguyên a và b - có một cây cầu dừa nối đảo a và đảo b.
- Dòng thứ N + 1: số nguyên M - số ngày kiểm tra.
- M dòng tiếp theo: mỗi dòng gồm hai số nguyên a và b - ngày hôm đó, Pirate sẽ kiểm tra các cây cầu trên đoạn đường từ đảo a đến đảo b.

Output

- Một số nguyên duy nhất thể hiện số cây cầu chưa được kiểm tra.



Giới hạn

- $1 \leq N, M \leq 200000$
- 60% số test có $1 \leq N, M \leq 5000$
- 80% số test có $1 \leq N, M \leq 50000$

Example

Input:

```
6
1 2
2 3
2 4
4 5
4 6
2
3 6
5 6
```

Output:

```
1
```

Giải thích: Ngày thứ nhất, Pirate kiểm tra các cây cầu (2, 3), (2, 4) và (4, 6). Ngày thứ hai, anh kiểm tra các cây cầu (5, 4) và (4, 6). Cây cầu duy nhất chưa được kiểm tra là (1, 2).

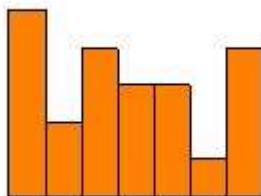
Bài 3: Bán dừa

Mã bài: KPLANK

Nếu các bạn biết câu chuyện thương tâm "ăn dừa leo trả vàng" của Pirate hẳn đã phải khóc hết nước mắt khi anh ấy, vì lòng thương chim, đã bán rẻ trái dừa leo siêu bự của mình.

Dừa leo cũng đã bị chim to lấy đi rồi, Pirate giờ chuyển sang nghề bán dừa để bù lỗ. Bất đắc dĩ thôi, vì trên đảo toàn là dừa...

Nhưng mà bán cái gì thì đầu tiên cũng phải có biển hiệu đã. Pirate quyết định lùng sục trên đảo các mảnh ván còn sót lại của những con tàu đắm để ghép lại thành tấm biển. Cuối cùng anh cũng tìm được N tấm ván hình chữ nhật, tấm thứ i có chiều rộng là 1 đơn vị và chiều dài là a_i đơn vị. Pirate dựng đứng chúng trên mặt đất và dán lại với nhau để được một mảnh ván to hơn (xem hình minh họa).



Việc cuối cùng chỉ là đem mảnh ván này đi cưa thành tấm biển thôi. Nhưng hóa ra đây lại là công việc khó khăn nhất. Pirate rất thích hình vuông và muốn tấm biển của mình càng to càng tốt, nhưng khổ nỗi trên đảo lại không có nhiều dụng cụ đo đạc. Không êke, không thước đo độ, nên Pirate chỉ còn cách dựa vào cạnh của N tấm ván ban đầu để cưa cho thảng thới. Pirate chỉ có thể cưa theo những đoạn thẳng chứa một cạnh nào đó (dọc hoặc ngang) của các tấm ván.

Hãy giúp anh ấy cưa được tấm biển lớn nhất có thể.

Input

- Dòng thứ nhất: ghi số nguyên N - số tấm ván.
- N dòng tiếp theo: mô tả độ cao của các tấm ván theo thứ tự trái sang phải sau khi đã dán lại.

Output

- Một số nguyên duy nhất là độ dài cạnh của tấm biển lớn nhất có thể cưa được.

Giới hạn

- Độ cao của các tấm ván là các số nguyên dương không vượt quá 10^9 .
- $1 \leq N \leq 10^6$.
- 60% số test có $1 \leq N \leq 2000$.
- 80% số test có $1 \leq N \leq 10^5$.

Example

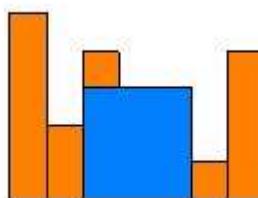
Input:

7
5
2
4
3
3
1
4

Output:

3

Giải thích: Hình dưới đây minh họa phương án tối ưu.



Pirate contests 2 Problem

Bài 1:Bán hàng

Mã bài: KTREEC

Quả là trái dừa đã cho Pirate một ý tưởng thiên tài khi chuyển sang bán cocktail. Bây giờ Pirate đã mở rộng mạng lưới của mình sang khắp các hòn đảo bên cạnh. Tại mỗi hòn đảo, anh mở một cửa hàng cocktail. Tùy theo sản vật của từng hòn đảo mà mỗi cửa hàng chuyên về một loại cocktail nhất định. Tuy nhiên, do hệ thống đường nối các hòn đảo với nhau vẫn chưa được cải thiện (vẫn dùng các cây cầu dừa để vừa đủ cho từ mỗi hòn đảo chỉ có một đường đi duy nhất đến mọi hòn đảo khác) nên khi giao hàng cho khách gặp một vấn đề nhỏ: nhiều khi khách hàng lại không muốn thưởng thức sản phẩm trên hòn đảo của mình mà lại muốn dùng những loại cocktail khác. Để tiết kiệm chi phí đi lại, Pirate phải chọn cửa hàng cung ứng loại cocktail tương ứng gần với hòn đảo của người khách nhất để đi giao hàng...

Input

- Dòng thứ nhất: ghi số nguyên N - số hòn đảo.
- N - 1 dòng tiếp theo: mỗi dòng ghi hai số nguyên x và y - có một cây cầu dừa nối hai hòn đảo x và y.
- Dòng thứ N + 1: ghi số nguyên C - số loại cocktail mà các cửa hàng cung ứng.
- Dòng thứ N + 2: ghi N số nguyên - loại cocktail chuyên biệt của từng hòn đảo.
- Dòng thứ N + 3: ghi số nguyên M - số các vị khách.
- M dòng tiếp theo: mỗi dòng mô tả hai số nguyên x và y - người khách ở đảo x yêu cầu cocktail loại y.

Output

- Gồm M dòng, mỗi dòng ghi ra khoảng cách (tính bằng số cây cầu dừa) cửa hàng gần nhất phải đi để giao hàng cho người khách tương ứng.

Giới hạn:

- Trong mỗi test, $1 \leq C \leq N \leq 10^4$, $1 \leq M \leq 10^4$.
- 60% số test có $1 \leq C \leq 10^2$.

Example

Input:

```
8
1 2
1 3
2 7
2 8
3 4
4 5
4 6
4
```



```
1 2 4 2 3 1 3 2
8
1 4
2 4
3 4
4 4
5 4
6 4
7 4
8 4
```

Output:

```
1
2
0
1
2
2
3
3
```

Giải thích: Chỉ có cửa hàng 3 chuyên về loại cocktail 4, nên kết quả chính là khoảng cách từ hòn đảo 3 đến các hòn đảo còn lại.

Bài 2: Pha chế

Mã bài: KMIX

Sau khi thu hoạch đủ trái cây, Pirate đưa hết chúng vào nhà máy tinh chế để chuẩn bị cho sự nghiệp bán cocktail của mình. Đầu tiên, Pirate thử nghiệm trên hai loại quả là dâu và cam. Tuy nhiên, kinh doanh cocktail cũng không phải là công việc dễ dàng. Nhà máy chỉ sản xuất được một số loại cocktail nhất định. Mỗi loại có nồng độ cam và dâu khác nhau (nồng độ được tính theo đơn vị phần tử). Vấn đề là mỗi vị khách lại có khẩu vị khác nhau và họ yêu cầu Pirate phải pha chế được đúng loại cocktail có x phần tử dâu và y phần tử cam thì họ mới trả tiền. Để tính toán lợi nhuận, Pirate muốn xác định trước xem có thể đáp ứng yêu cầu của từng vị khách hay không.

Input

- Dòng thứ nhất: ghi một số nguyên N - số loại cocktail có sẵn.
- N dòng tiếp theo: mỗi dòng ghi hai số nguyên - nồng độ dâu và cam của từng loại cocktail.
- Dòng thứ N + 2: ghi một số nguyên M - số lượng các vị khách.
- M dòng tiếp theo: mỗi dòng ghi hai số nguyên - nồng độ dâu và cam yêu cầu của từng vị khách.

Output

- Gồm M dòng, mỗi dòng ghi 'YES' nếu yêu cầu của vị khách tương ứng được thỏa mãn và 'NO' nếu ngược lại.

Giới hạn



- Trong mỗi test, $1 \leq N, M \leq 10^5$. Nồng độ của các loại cocktail là các số nguyên không âm không quá 10^9 .
- 60% số test có $1 \leq N \leq 10^2$.
- 80% số test có $1 \leq N \leq 10^3$.

Example

Input:

```
3
0 10
20 30
30 10
2
10 30
20 20
```

Output:

```
NO
YES
```

Giải thích: ta có thể đáp ứng yêu cầu của vị khách thứ hai bằng cách pha 3 loại cocktail theo tỉ lệ 1 : 3 : 2.

Bài 3: Thu hoạch

Mã bài: KCOLLECT

Công việc buôn bán dừa của Pirate không mấy quan cho lắm, khiến anh đêm ăn không ngon ngày ngủ không yên, chỉ biết chui đầu vào xem "Rôbô trái cây". Một ngày nọ, đang nằm ngủ dưới gốc dừa, bỗng một trái dừa rơi vào đầu anh ấy. Cũng giống như Newton, Pirate cũng cầm trái dừa lên, ngắm nghía và... rủa: "Khi thật, sao xứ này toàn là dừa thế này!". Tức giận lên, Pirate quyết trồng thêm các loại trái cây khác vào hòn đảo của mình.

Đến mùa thu hoạch, Pirate đặt hàng một "Rôbô trái cây" để giúp mình hái quả. Khu vườn của Pirate có hình chữ nhật, và được chia thành $M \times N$ ô vuông bằng nhau. Trong mỗi ô vuông có một cây thuộc một loại quả khác nhau, đánh số từ 0 đến 9. Không phải vô tình mà chúng được đánh số như vậy, con số đó thể hiện giá trị kinh tế của các loại cây.

Tuy nhiên, nhìn mặt con Rôbô trái cây này có vẻ ngu ngú nên trong lần đầu tiên thử việc, Pirate muốn test AI của nó. Cụ thể là Rôbô phải tuân theo các quy định sau:

- a. Tại mỗi ô, Rôbô chỉ có thể đi sang hướng đông hoặc hướng nam sang ô kè cạnh.
- b. Có một số ô đặc biệt mà tại đó Rôbô có thể đi được thêm hướng tây hoặc hướng bắc sang ô kè cạnh (chỉ một trong hai).



c. Rôbô không được đi vào những ô có cây dùa (Pirate cǎm thù dùa).

d. Rôbô được đi qua một ô nhiều lần. Khi đi qua một ô, Rôbô phải hái hết quả ở cây trong ô đó. Lợi nhuận thu được sẽ bằng chỉ số của loại cây vừa được thu hái. Và sau này, không thể đạt thêm lợi nhuận gì từ ô đó nữa.

Xuất phát từ ô ở góc tây bắc của khu vườn, hãy giúp Rôbô trái cây xác định hành trình để đạt được lợi nhuận tối đa.

Input

- Dòng thứ nhất: ghi hai số nguyên M và N - kích thước của khu vườn.
- M dòng tiếp theo: mỗi dòng ghi N kí tự liên tiếp nhau mô tả khu vườn:

+ '0' - '9': các loại trái cây;

+ '#': cây dùa;

+ 'W': được quyền đi theo hướng tây;

+ 'N': được quyền đi theo hướng bắc.

Output

- Ghi một số nguyên duy nhất là lợi nhuận tối đa đạt được.

Giới hạn

- Trong mọi test, $1 \leq M, N \leq 100$.
- 60% số test có $1 \leq M, N \leq 20$.

Example

Input:

2 3
264
3WW

Output:

15

Giải thích: Rôbô sẽ đi theo hành trình như sau $(1, 1) \rightarrow (1, 2) \rightarrow (1, 3) \rightarrow (2, 3) \rightarrow (2, 2) \rightarrow (2, 1)$ ($\hat{o}(i, j)$ là ô ở dòng i và cột j). Tổng lợi nhuận sẽ là $2 + 6 + 4 + 3 = 15$.



Pirate contests 3 Problem

Bài 1: Trồng hoa

Mã bài: KDIFF

Pirate là một người rất yêu hoa. Anh ấy trồng một luống hoa trước cửa nhà mình. Luống hoa được chia thành các ô đất, mỗi ô đất trồng một bông hoa. Tuy nhiên, vì đang bị đau chân nên Pirate ấy không thể chăm sóc luống hoa một cách hoàn hảo nhất. Kết quả là các bông hoa của anh có xấu đẹp không đều nhau.

Để cải thiện tình hình, Pirate quyết định chỉ để lại hai khóm hoa rời nhau, mỗi khóm gồm một số các bông hoa đứng liên tiếp nhau. Để ngôi nhà của mình trông thật xinh đẹp, hai khóm hoa kia phải được chọn lựa kỹ càng. Anh dùng đôi mắt thẩm mỹ tinh tường của mình (gọi là "sắc kế") để đánh giá độ xinh đẹp của các bông hoa, được thể hiện bằng các số nguyên không âm. Căn cứ vào đó, một khóm hoa đạt tiêu chuẩn khi và chỉ khi chênh lệch độ xinh đẹp giữa hai bông hoa bất kì trong khóm không quá một giá trị cho trước. Pirate muốn hai khóm hoa có càng nhiều bông hoa càng tốt. Bạn hãy giúp anh ấy xác định xem có thể chọn được nhiều nhất bao nhiêu bông nhé.

Input

- Dòng 1: Hai số nguyên N - số bông hoa trên luống hoa, K - chênh lệch độ xinh đẹp tối đa của hai bông hoa bất kì trong một khóm.
- N dòng tiếp theo: Mỗi dòng là một số nguyên thể hiện độ xinh đẹp của một bông hoa.

Output

- Một số nguyên duy nhất là số bông hoa được chọn của hai khóm hoa.

Giới hạn

- $1 \leq N \leq 3 * 10^5$.
- 30% số test có $1 \leq N \leq 30$.
- 50% số test có $1 \leq N \leq 10^3$.
- Các số trong dữ liệu vào đều là số nguyên không âm không quá 10^9 .

Example

Input:

```
5 2
1
3
2
5
4
```

Output:



Giải thích: hai khóm hoa được chọn là (1, 2, 3) và (4, 5).

Input:

5 2
1
3
5
2
4

Output:

4

Giải thích: hai khóm hoa được chọn là (1, 2) và (4, 5).

Bài 2: Học sử

Mã bài: KHISTORY

Pirate rất thích học môn Sử. Một hôm, anh tình cờ tìm được một cuốn sách sử địa phương. Chăm chú đọc, Pirate phát hiện ra rằng quần đảo của mình có một lịch sử rất huy hoàng...

Vào năm XYZ trước Công Nguyên, quần đảo của Pirate được gồm nhiều hòn đảo nhỏ ở gần nhau. Khi kinh tế của mỗi hòn đảo ngày càng phát triển dẫn đến nhu cầu các hòn đảo phải trao đổi hàng hóa. Vậy là những cây cầu dừa được xây dựng ở một số cặp hòn đảo để có thể được từ đảo này sang đảo kia và ngược lại. Theo quy luật tự nhiên, khi có trao đổi hàng hóa nhảy vọt sẽ dẫn đến nhu cầu liên kết và mở rộng thị trường. Vì vậy, các hòn đảo dần tập hợp lại và mở ra một thời đại mới, thời đại của các quốc gia trên biển.

Một quốc gia là tập hợp của một số hòn đảo, mỗi hòn đảo chỉ thuộc vào một quốc gia. Nhận thấy rằng các cây cầu dừa rất mỏng manh nhưng lại vô cùng trọng yếu, các hòn đảo chỉ liên kết lại thành một quốc gia khi và chỉ khi chúng vẫn đi được đến nhau nếu có bất cứ một cây cầu dừa nào nối hai hòn đảo thành viên không sử dụng được.

Sự phân chia này dẫn đến việc các hòn đảo có nền kinh tế tương đương có điều kiện vô cùng thuận lợi để phát triển. Hệ quả tất yếu là sự phân hóa giữa các nước quốc gia. Một số quốc gia trở nên hùng mạnh hơn hẳn các quốc gia khác. Ta gọi đó là các siêu cường.

Sách sử ghi lại rằng một quốc gia hoặc là một siêu cường, hoặc có cầu dừa nối trực tiếp tới một siêu cường. Hơn nữa, sách có lưu ý rằng số siêu cường là ít nhất có thể nhưng không có ghi chép nào khác nữa về chúng. Dựa vào sơ đồ các cây cầu dừa của khác hòn đảo khi xưa, Pirate muốn xác định xem vào thuở sơ khai, quần đảo có bao nhiêu siêu cường. Bạn hãy giúp anh ấy nhé.

Input

- Dòng 1: Hai số nguyên N - số hòn đảo, M - số lượng các cây cầu dừa.
- M dòng tiếp theo: Mỗi dòng chứa hai số nguyên, mô tả các cặp hòn đảo có cầu dừa nối với nhau.

Output



- Một số nguyên duy nhất là số siêu cường.

Giới hạn

- $1 \leq N, M \leq 10^5$.
- 30% số test có $1 \leq N, M \leq 20$.
- Giữa một cặp hòn đảo chỉ có tối đa một cây cầu dừa nối chúng.

Example

Input:

6 7
1 2
2 3
3 1
4 5
5 6
6 4
1 4

Output:

1

Giải thích: có hai quốc gia là (1, 2, 3) và (4, 5, 6). Chúng có đường nối trực tiếp với nhau, vì vậy siêu cường có thể là một trong hai.

Input:

15 19
1 2
2 3
3 1
4 5
5 6
6 4
7 8
8 9
9 7
10 11
11 12
12 10
13 14
14 15
15 13
1 4
1 7
1 10
1 13

Output:

1



Giải thích: có năm quốc gia là (1, 2, 3), (4, 5, 6), (7, 8, 9), (10, 11, 12), (13, 14, 15). Để số siêu cường là ít nhất thì chỉ có thể có một siêu cường là (1, 2, 3) vì mọi quốc gia khác đều có cầu dẫn nối trực tiếp đến nó.

Bài 3: Mật mã

Mã bài: KPASS

Hôm nay, Pirate sẽ truyền dạy một kinh nghiệm quý báu cho các anh chàng có nhiều bạn gái...

Tất nhiên là việc lưu số điện thoại vào danh bạ có nhiều lợi ích, nhưng khi bạn gái trả bài: "Anh có nhớ số em không?", trong thâm tâm các nàng luôn muốn câu trả lời bạn là "Anh chỉ nhớ mỗi số em thôi!". Khi số lượng bạn gái tăng lên cũng đồng nghĩa với số lượng số điện thoại bạn cần phải ghi nhớ cũng tăng lên một cách chóng mặt, nhất là với các nàng thích xài SIM khuyến mãi.

Một sự cố thường gặp nhất đó là bạn nhầm số của bé X sang bé Y. Thé là bé X móc ngay điện thoại ra bấm số và có một cuộc "nội chiến" xảy ra.

Vậy làm sao để ghi nhớ các số điện thoại một cách nhanh chóng và an toàn?

Pirate đã nghĩ ra một quy trình mã hóa số điện thoại như sau:

- Giả sử số điện thoại của bạn được biểu diễn bằng dãy số a_1, a_2, \dots, a_N ($0 \leq a_i \leq 9$ với $1 \leq i \leq N$).
- Đặt $s_1 = a_1$, $s_i = s_{i-1} + a_i$ (với $1 < i \leq N$).
- Tìm số thứ tự từ điển của dãy $\{s_N\}$ trong tập các dãy được sinh ra từ tất cả các số điện thoại theo nguyên tắc trên. Trừ số đó đi một đơn vị.
- Trong dãy số N bit biểu diễn nhị phân của số vừa nhận được, gọi K là dãy bit 0 liên tiếp nhau dài nhất. Tìm thứ tự từ điển của dãy N bit trên trong tập các dãy nhị phân N bit có không quá K bit 0 liên tiếp, gọi số đó là X .

Nhiệm vụ của bạn là giúp Pirate viết một chương trình giúp tìm ra một số điện thoại nếu biết được N , X và K .

Lưu ý: dãy số $\{a_N\}$ được gọi là lớn hơn dãy số $\{b_N\}$ khi và chỉ khi tồn tại một chỉ số i sao cho $a_j = b_j$ (với mọi $1 \leq j < i$) và $a_i > b_i$. Trong một tập hợp các dãy số có cùng tính chất nào đó, thứ tự từ điển của một dãy số là số lượng dãy số trong tập không lớn nó (bao gồm cả chính nó).

Input

- Ghi ba số nguyên N , X và K .

Output

- Gồm một dòng duy nhất ghi ra dãy số $\{a_N\}$ ban đầu, các số cách nhau bởi dấu cách.

Giới hạn



- $1 \leq N, K \leq 30; 1 \leq X \leq 10^9$.
- 30% số test có $1 \leq N, K \leq 10$.
- Dữ liệu vào đảm bảo có kết quả.

Example

Input:

4 3 2

Output:

0 0 0 4

Giải thích: dãy nhị phân 4 bit thứ 3 có không quá 2 bit 0 liên tiếp là 0100. Đây là biểu diễn của số 4. Từ đó suy ra được dãy $\{s_N\}$ là (0, 0, 0, 4). Vậy dãy $\{a_N\}$ là (0, 0, 0, 4).

Input:

4 5 1

Output:

0 0 1 1

Giải thích: dãy nhị phân 4 bit thứ 5 có không quá 1 bit 0 liên tiếp là 1011. Đây là biểu diễn của số 11. Từ đó suy ra được dãy $\{s_N\}$ là (0, 0, 1, 2). Vậy dãy $\{a_N\}$ là (0, 0, 1, 1).



VNOI MARATHON 11

Round 1:

Bài 1: Pirate đăng trí

Mã bài: VMDEGREE

Những cây cầu dừa nối những hòn đảo của Pirate đang gây trở ngại lớn cho anh ấy khi việc kinh doanh ngày càng mở rộng. Một ngày nọ, Pirate quyết định thiết kế một hệ thống đường mới nối các hòn đảo với nhau. Mỗi đường nối sẽ rộng hơn trước và cho phép di chuyển theo hai chiều. Để tiết kiệm chi phí, giữa hai hòn đảo chỉ nên có một đường nối. Bản vẽ đã xong hoàn thành, giờ chỉ còn việc bắt tay vào xây dựng. Tuy nhiên, do tính đăng trí kinh niên của mình, Pirate đã làm mất bản vẽ thiết kế. Lục tung đống giấy tờ thì chỉ còn lại một mảnh giấy thống kê về số hòn đảo (khác) có đường nối trực tiếp đối với từng hòn đảo.

Việc nâng cấp không thể chậm trễ được, Pirate cần phải bắt tay vào vẽ lại bảng thiết kế dựa vào bảng thống kê trên. Nhưng khổ một nỗi, anh vẫn đang nghi ngờ tính chính xác của bảng thống kê này. Vì thế, trước hết, các bạn hãy giúp Pirate xác định xem có tồn tại một hệ thống đường thỏa mãn bảng thống kê tìm được không.

Input

- Dòng thứ nhất ghi một số nguyên T – số bộ test.
- Tiếp theo là T test, mỗi test được mô tả như sau:
 - + Dòng đầu tiên ghi một số nguyên N – số lượng các hòn đảo.
 - + N dòng tiếp theo: mô tả bảng thống kê.

Output

- Gồm T dòng, mỗi dòng ghi “YES” nếu có tồn tại một hệ thống đường thỏa mãn bảng thống kê trong test tương ứng, hoặc “NO” nếu ngược lại.

Example

Input:

1

2

2



1

1

Output:

YES

Giới hạn

- $1 \leq T \leq 10$.
- $1 \leq N \leq 10^5$.
- 60% bộ test có $1 \leq N \leq 10^3$.

Mỗi số trong input là một số nguyên không âm không quá 10^9 .

Bài 2: Nối điểm

Mã bài: VMLINES

Trường mầm non SuperKids tổ chức một buổi kiểm tra tư duy hình học của các cháu. Bài kiểm tra như sau: Mỗi bé được phát một tờ giấy trên đó có $2N$ điểm hoàn toàn phân biệt: N điểm xanh và N điểm vàng, các điểm xanh đánh số từ 1 tới N và các điểm vàng cũng được đánh số từ 1 tới N . Lần lượt với $i = 1, 2, \dots, N$, nếu đoạn thẳng nối từ điểm xanh thứ i tới điểm vàng thứ i không có điểm chung với những đoạn thẳng đã nối thì bé phải nối đoạn thẳng đó, ngược lại bé phải thông báo ngay là không nối được ở lượt thứ i và bài kiểm tra kết thúc, giá trị i này khi đó được gọi là đáp số của bài kiểm tra. Nếu bé có thể nối được tất cả N đoạn thẳng thì đáp số quy ước là -1.

Vì số lượng điểm khá lớn nên cô giáo muốn nhanh chóng biết đáp số để chấm bài cho các bé. Hãy giúp các cô giáo của trường SuperKids biết đáp số của bài kiểm tra.

Input

- Dòng 1 chứa số nguyên dương $N \leq 10^5$
- dòng tiếp theo, dòng thứ i chứa 4 số nguyên x_i, y_i, x'_i, y'_i trong đó (x_i, y_i) là tọa độ điểm xanh thứ i còn (x'_i, y'_i) là tọa độ điểm vàng thứ i . Các tọa độ là số nguyên có giá trị tuyệt đối không quá 10^9 .

Output

- Ghi ra một số nguyên duy nhất là đáp số của bài.

Example

Input:

5

4 5 1 4

2 1 2 4



3 3 6 3
6 2 4 4
5 1 3 1

Output:
4

Round 2:

Bài 1: Cái Giếng

Mã bài: WELL

Ngày xưa ngày xưa, có một tên sọ vợ tên là Pirate. Hắn sọ vợ nhất làng nhưng lại muôn ra oai trước mọi người. Thế là hắn bèn xây cho nhà mình một cái giếng. Mỗi lần vợ bị rượt chạy trối chết, hắn đều nhầm thảng cái giếng rồi chạy vòng vòng quanh nó. Người ngoài nhìn vào thấy hai vợ chồng đang chạy quanh cái giếng, chẳng biết ai đang đuổi đánh ai cả, cứ tưởng tên này đang “dạy vợ”. Muốn mình càng nổi tiếng hơn nữa, hắn quyết định là cái giếng của nhà mình phải thật là đặc biệt. Vậy là hắn bắt tay vào thiết kế...

Giếng nhà Pirate là một khối hình trụ tròn rỗng. Giếng gồm nhiều vòng, vòng này chồng lên vòng kia. Mỗi vòng được xây bằng các viên đá ống bằng nhau. Viên đá ở vòng trên phải được đặt thẳng hàng với viên đá ở vòng dưới làm cho khi nhìn ngang từ bên ngoài, cái giếng trông giống như một bảng hình chữ nhật gồm các ô vuông.

Mỗi viên đá ống có thể được sơn mặt ngoài bằng một trong hai màu: trắng hoặc đen. Sơn giếng xong rồi, Pirate bèn khắc lên đó một câu đố nhằm lưu danh thiên cổ. Câu đố như sau: “Có bao nhiêu cách sơn giếng khác nhau? Trong đó, có bao nhiêu cách để nhìn từ ngoài vào không thấy bất cứ hình vuông 2x2 nào được sơn hoàn toàn bằng một màu?”.

Hai cách sơn gọi là khác nhau nếu không thể “xoay” để cách này trùng với cách kia. Nếu hình dung cái giếng như một bảng hai chiều, phép “xoay” ở đây là việc dời cột bên trái nhất (cột 1) sang phía ngoài cùng bên phải.

Cái giếng đúng là tầm thường nhưng câu đố đúng là bất hủ đấy, các bạn thử giải xem!

Input

- Dòng thứ nhất là số nguyên T - số bộ test.
- T dòng tiếp theo: mỗi dòng gồm 3 số nguyên: K - loại câu hỏi; M - số vòng của giếng; và N - số viên đá trên mỗi vòng.
 - + Nếu $K = 1$, bạn phải trả lời câu hỏi có bao nhiêu cách sơn khác nhau.
 - + Nếu $K = 2$, bạn phải trả lời câu hỏi trong tất cả các cách sơn khác nhau, có bao nhiêu cách mà không có hình vuông 2x2 nào được sơn bằng một màu.

Output



- Gồm T dòng: mỗi dòng là số cách sơn giêng ở test tương ứng.

Giới hạn

- $1 \leq T \leq 15$.
- Với $K = 1$, $1 \leq M \leq 10000$, $1 \leq N \leq 20$.
- Với $K = 2$, $1 \leq M * N \leq 64$.
- 40% số test có $1 \leq M * N \leq 20$.

Example

Input:

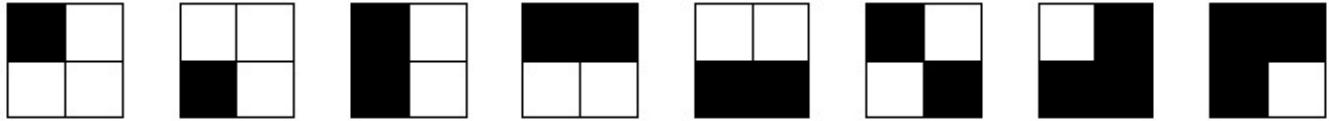
1

2 2 2

Output:

8

Giải thích: ta có 8 cách sơn như sau:



Lưu ý rằng các cách sơn như sau được xem là như nhau:

$$(BW, WW) = (WB, WW)$$

$$(WW, BW) = (WW, WB)$$

$$(BW, BW) = (WB, WB)$$

$$(BW, WB) = (WB, BW)$$

$$(BB, BW) = (BB, WB)$$

$$(BW, BB) = (WB, BB)$$

Bài 2: Sắp xếp đội hình

Mã bài: STRATEGY



Sau thất bại của đội tuyển Anh tại World Cup 2010, huấn luyện viên Fabio Capello quyết định sẽ áp dụng một công nghệ mới cho đội tuyển Anh tại Euro 2012. Ông giao nhiệm vụ cho HLV phó thu thập thông tin về vị trí trên sân của các cầu thủ đối phương và sẽ sắp xếp cầu thủ của mình sao cho diện tích mặt sân mà kiểm soát là nhiều nhất có thể.

Sân bóng là một hình chữ nhật có góc trái dưới ở (0, 0) góc phải trên ở (120, 90). Bạn hãy sắp xếp 11 cầu thủ của đội tuyển Anh sao cho các vị trí này không trùng với các vị trí của đối phương và diện tích sân mà đội tuyển Anh kiểm soát là lớn nhất có thể. Một vị trí trên sân sẽ được kiểm soát bởi đội nào có cầu thủ ở gần vị trí đó hơn.

Input

Dữ liệu vào ghi 11 dòng, mỗi dòng ghi 2 số x y nguyên là vị trí của cầu thủ đối phương

Output

Dữ liệu ra ghi 11 dòng, mỗi dòng là 2 số nguyên x y là vị trí của cầu thủ đội tuyển Anh.

Điểm được cho theo độ tốt của đáp án.

Example

Input:

```
1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
10 10
11 11
```

Output:

```
12 12
13 13
14 14
15 15
16 16
17 17
18 18
19 19
20 20
21 21
22 22
```

Lưu ý: Trong lúc thi, BTC sẽ chấm trước 40% số test, điểm tối đa các bạn submit đúng cả 40% số test sẽ là 40 điểm.



Round 3:

Bài 1: Mã khóa bí mật

Mã bài: MAKHOA3

Hè đã về!! ConanKudo đã tự hứa sẽ làm một việc gì đó có ích trong hè này, và bây giờ là lúc anh bắt đầu.

ConanKudo đi thuyền tới một hòn đảo giàu vàng. Sau nhiều ngày tìm kiếm, anh phát hiện ra một chiếc hòm đã bị khóa. Hiển nhiên ConanKudo không thể phá khóa vì hệ thống kích nổ sẽ hoạt động, và cái mà anh nhận được sẽ chỉ là một đống tro. Do đó, việc tìm ra chìa khóa để mở chiếc hòm là rất cần thiết.

Chìa khóa này là một bảng kích thước $M \times N$ chỉ chứa các số 0 hoặc 1 ($0 < M, N \leq 30$). Các hàng được đánh số từ 1 đến M , và các cột được đánh số từ 1 đến N . Từ những người đến trước (và thất bại trong việc mở khóa), ConanKudo đã thu thập được $M+N$ thông tin. Mỗi thông tin tương ứng với một hàng hoặc một cột của bảng, và có 1 trong 2 dạng:

Dạng 1: $K_i A(i,1) A(i,2) \dots A(i,K_i)$ có ý nghĩa là: Trên hàng i , có K_i đoạn được tạo bởi 'dãy dài nhất các số 1 liên tiếp', và độ dài các đoạn lần lượt (từ trái qua phải) là $A(i,1) A(i,2) \dots A(i,K_i)$. ('Dãy dài nhất các số 1 liên tiếp' là dãy gồm các số 1 liên tiếp và không là một dãy con của dãy các số 1 liên tiếp khác).

Dạng 2: $K_j B(j,1) B(j,2) \dots B(j,K_j)$ có ý nghĩa là: Trên cột j , có K_j đoạn được tạo bởi 'dãy dài nhất các số 1 liên tiếp', và độ dài các đoạn lần lượt (từ trên xuống dưới) là $B(j,1) B(j,2) \dots B(j,K_j)$.

Có M thông tin dạng 1 tương ứng với M hàng, và N thông tin dạng 2 tương ứng với N cột.

Chỉ 10 phút sau khi ConanKudo đưa thông tin về mã khóa, ll931110, với tốc độ code như gió, đã đưa ra được output. Tuy nhiên ll931110 đưa ra không phải chỉ 1 đáp án, mà những T ($T \leq 5$) đáp án khác nhau. Nhiệm vụ của bạn là kiểm tra xem đáp án nào là đáp án đúng (thỏa mãn tất cả $M+N$ thông tin).

Input

Dòng 1: 2 số nguyên M, N cách nhau bởi dấu cách

M dòng tiếp, dòng thứ i là $K_i A(i,1) A(i,2) \dots A(i,K_i)$ cho biết thông tin ứng với hàng i

N dòng tiếp, dòng thứ j là $K_j B(j,1) B(j,2) \dots B(j,K_j)$ cho biết thông tin ứng với cột j

Dòng tiếp theo chứa số T - số đáp án khác nhau mà ll931110 đưa ra.

Tiếp theo là T nhóm dòng, mỗi nhóm gồm M dòng, mỗi dòng gồm N ký tự mô tả kết quả mà ll931110 đưa ra.

Chú ý:



- Các dòng trống thừa có thể xuất hiện ở bất kỳ vị trí nào trong file.
- Nếu một hàng của mã không có số 1 nào, thì ràng buộc tương ứng sẽ gồm duy nhất 1 số 0

Output

Gồm T dòng, dòng thứ i là YES nếu kết quả thứ i của ll931110 đưa ra thỏa mãn tất cả $M+N$ thông tin, ngược lại in ra NO.

Example

Input

3 3

1 1

1 1

1 1

1 1

1 1

1 1

2

100

010

001

100

010

000



Output

YES

NO

Bài 1: Mã khóa bí mật

Mã bài: MAKHOA2

Hè đã về!! ConanKudo đã tự hứa sẽ làm một việc gì đó có ích trong hè này, và bây giờ là lúc anh bắt đầu.

ConanKudo đi thuyền tới một hòn đảo giàu vàng. Sau nhiều ngày tìm kiếm, anh phát hiện ra một chiếc hòm đã bị khóa. Hiện nhiên ConanKudo không thể phá khóa vì hệ thống kích nổ sẽ hoạt động, và cái mà anh nhận được sẽ chỉ là một đống tro. Do đó, việc tìm ra chìa khóa để mở chiếc hòm là rất cần thiết.

Chìa khóa này là một bảng kích thước $M \times N$ chỉ chứa các số 0 hoặc 1 ($0 < M, N \leq 30$). Các hàng được đánh số từ 1 đến M , và các cột được đánh số từ 1 đến N . Từ những người đến trước (và thất bại trong việc mở khóa), ConanKudo đã thu thập được $M+N$ thông tin. Mỗi thông tin tương ứng với một hàng hoặc một cột của bảng, và có 1 trong 2 dạng:

Dạng 1: $K_i A(i,1) A(i,2) \dots A(i,K_i)$, có ý nghĩa là: Trên hàng i , có K_i đoạn được tạo bởi 'dãy dài nhất các số 1 liên tiếp', và độ dài các đoạn lần lượt (từ trái qua phải) là $A(i,1) A(i,2) \dots A(i,K_i)$. ('Dãy dài nhất các số 1 liên tiếp' là dãy gồm các số 1 liên tiếp và không là một dãy con của dãy các số 1 liên tiếp khác).

Dạng 2: $K_j B(j,1) B(j,2) \dots B(j,K_j)$, có ý nghĩa là: Trên cột j , có K_j đoạn được tạo bởi 'dãy dài nhất các số 1 liên tiếp', và độ dài các đoạn lần lượt (Từ trên xuống dưới) là $B(j,1) B(j,2) \dots B(j,K_j)$.

Có M thông tin dạng 1 tương ứng với M hàng, và N thông tin dạng 2 tương ứng với N cột.

Cho kích thước bảng mã khóa, và các thông tin, hãy giúp ConanKudo xác định mã khóa.

Trong ít nhất 50% test, bài toán chỉ có 1 nghiệm duy nhất.

Input

Dòng 1: 2 số nguyên M, N cách nhau bởi dấu cách

M dòng tiếp, dòng thứ i là $K_i A(i,1) A(i,2) \dots A(i,K_i)$ cho biết thông tin ứng với hàng i

N dòng tiếp, dòng thứ j là $K_j B(j,1) B(j,2) \dots B(j,K_j)$ cho biết thông tin ứng với cột j

Chú ý:

- Các dòng trống thừa có thể xuất hiện ở bất kỳ vị trí nào trong file.



- Nếu một hàng của mã không có số 1 nào, thì rằng buộc tương ứng sẽ gồm duy nhất 1 số 0

Output

Gồm **M** dòng, mỗi dòng **N** ký tự 0 hoặc 1, thể hiện mã khóa.

Điểm của bạn được cho như sau:

Gọi **D** là số hàng và cột thỏa mãn điều kiện. Điểm lớn nhất có thể đạt được của một test là 2.5 điểm.

- Nếu $D \leq \max(M, N)$, bạn được 0 điểm

- Nếu $D = M + N$ bạn được 2.5 điểm

- Trường hợp còn lại, bạn được:

$$\frac{D - \max(M, N)}{M + N - \max(M, N)} \times 2 + 0.5$$

Example

Input	Output	
10 10	1000110001	
3 1 2 1	1000111011	
3 1 3 2	1000110101	
4 1 2 1 1	0101010001	
4 1 1 1 1	0010010001	
3 1 1 1	0001001000	
	0011011000	
2 1 1	0001001000	
2 2 2	0001001000	
2 1 1	0001001000	
2 1 1		
2 1 1		
1 3		
1 1		
2 1 1		
2 1 5		
1 3		
2 5 1		
2 1 5		
1 1		
1 1		
1 5		



Với output như trên, bạn sẽ được 2.5 điểm cho test này

Trong lúc thi, bài của bạn sẽ được chấm với 50% số test của bài. Điểm tối đa bạn có thể đạt được là 5

Bài 1: Tham ăn

Mã bài: SMATHDOG

Tiếp theo chiến lược “quy hoạch động”, Bòm huấn luyện cho chú chó của mình chiến lược “tham ăn” trong một sân chơi được biểu diễn bởi mặt phẳng trực chuẩn. Ban đầu chú chó xuất phát ở điểm $(0,0)$ và nó phải đứng im cho tới khi được gọi. Trò chơi diễn ra trong N lượt, lượt thứ i của trò chơi diễn ra như sau:

Bòm di chuyển đến vị trí (x_i, y_i) , cầm c_i cái bánh và gọi chú chó. Chú chó có quyền đứng im hoặc di chuyển theo các phương song song theo trực tọa độ để đến chỗ Bòm nếu độ dài quãng đường di chuyển không vượt quá K . Nếu chú chó có thể đi được đến chỗ Bòm và quyết định di chuyển, nó sẽ được thưởng toàn bộ c_i cái bánh, ngược lại nó sẽ phải đứng nhìn Bòm ăn hết luôn c_i cái bánh đó. Hết lượt chơi này chú chó lại phải đứng im và trò chơi tiếp tục ở lượt $i+1$.

Yêu cầu: Cho biết trước các tọa độ (x_i, y_i) và số bánh c_i tại các lượt chơi, hãy giúp chú chó tội nghiệp của Bòm kiếm được nhiều bánh nhất

Input

- Dòng 1 chứa hai số nguyên dương N, K .
- N dòng tiếp theo, dòng thứ i chứa ba số nguyên dương x_i, y_i, c_i .
- Ràng buộc: $N \leq 10^5$, tất cả các số còn lại trong file dữ liệu đều là số nguyên dương $\leq 10^3$.

Output

- Gồm một số nguyên duy nhất là số bánh chú chó kiếm được trong trò chơi theo phương án của bạn.

Example

Input:

8 4

1 2 2

1 5 9

4 3 3



6 4 4

7 7 8

8 2 5

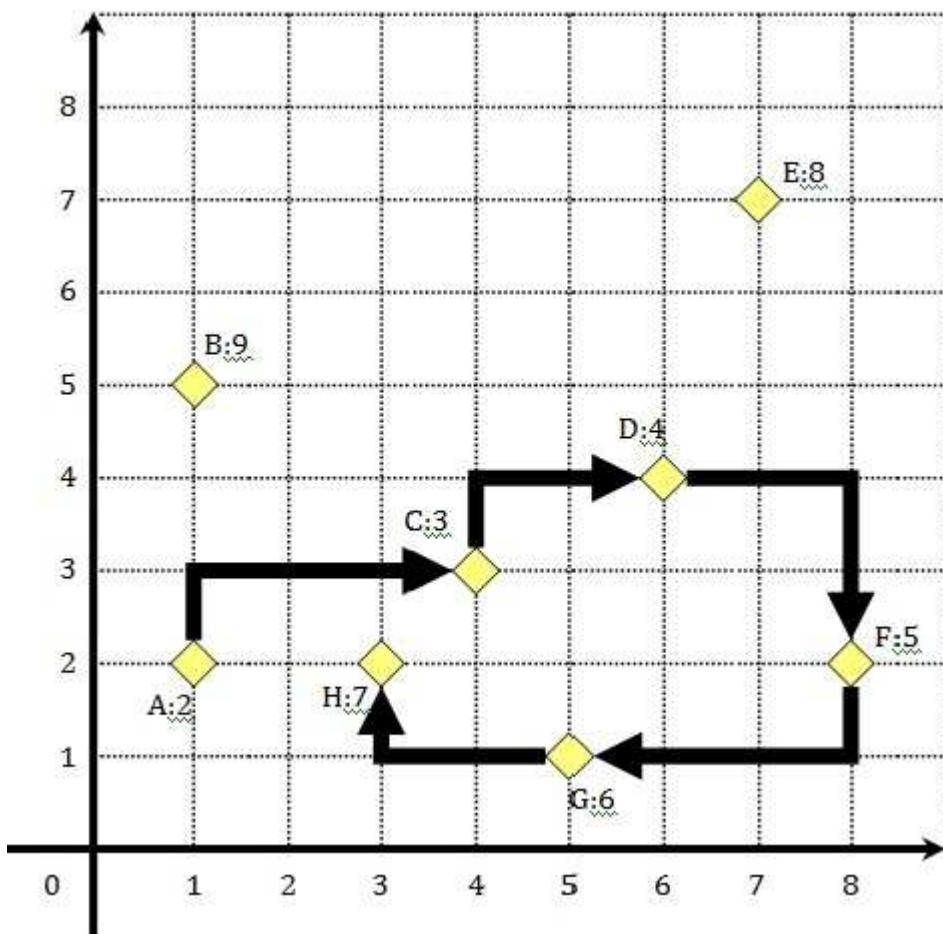
5 1 6

3 2 7

Output:

27

Đây là ví dụ với 8 lượt chơi và vị trí của Bờm tại 8 lượt chơi đó lần lượt là A, B, C, D, E, F, G, H.
Trong phương án tối ưu chú chó chỉ phải bỏ 9 bánh tại điểm B và 8 bánh tại điểm E



Round 4:

Bài 1: Tổng trên ma trận!

Mã bài: VMMTFIVE

Cho một bảng số 5×5 . Nhiệm vụ của bạn là sẽ phải điền vào ma trận sao cho tổng của các phần tử trên mỗi hàng và mỗi cột bằng một số nguyên cho trước. Mỗi phần tử trong bảng số từ 1 đến 25 và không có hai phần tử bất kì nào giống nhau.

Input

- Dòng thứ nhất gồm 5 số là tổng của các số từ dòng thứ 1 đến dòng thứ 5 của bảng số.
- Dòng thứ hai gồm 5 số là tổng của các số từ cột thứ 1 đến cột thứ 5 của bảng số.

Output

- Gồm 5 dòng, mỗi dòng 5 số thể hiện bảng 5×5 là kết quả của bạn. Nếu có nhiều đáp án, hãy in ra một đáp án bất kì. Dữ liệu đầu vào luôn luôn có kết quả.

Example

Input:

60 86 59 38 82

61 59 57 89 59

Output:

15 5 9 25 6
17 10 23 20 16
12 19 3 18 7
13 14 1 2 8
4 11 21 24 22

Bài 2: Đồ thị 3 phía

Mã bài: VM3PHIA

Một đồ thị vô hướng được gọi là đồ thị 3 phía nếu tồn tại một cách chia tập đỉnh V thành 3 tập V_1, V_2, V_3 khác rỗng sao cho mọi cặp đỉnh u, v có cạnh nối thì u, v thuộc 2 tập con khác nhau. Cho đồ thị $G = \langle V, E \rangle$ là một đồ thị 3 phía, tìm cách chia tập V thành 3 tập V_1, V_2, V_3 khác rỗng thỏa mãn.



Input

- Dòng đầu tiên ghi hai số N và M trong đó N là số đỉnh của đồ thị, M là số cạnh của đồ thị.
- M dòng tiếp theo, mỗi dòng ghi 2 số u v thể hiện cạnh nối giữa u và v. ($u \leftrightarrow v$)

Output

- In ra xâu N ký tự. Kí tự thứ i là 1/2/3 tương ứng với đỉnh i thuộc tập V1/V2/V3.
- Dữ liệu đảm bảo luôn có đáp án. Bạn chỉ cần đưa ra một đáp án bất kỳ.

Example

Input:

5 6
1 2
1 5
1 4
2 3
3 5
5 4

Output:

12132

Giới hạn: $3 \leq N \leq 200$, trong 20% số test, $N \leq 15$.

Bài 3: Kỷ lục đồ DOMINO

Mã bài: VM DOMINO

Một kỷ lục thế giới về xếp domino đồ đã được ghi nhận vào hôm 17/11/2006. Kỷ lục này thuộc về Hà Lan khi 4.079.381 quân domino đã lần lượt đổ xuống theo phản ứng dây chuyền trong tiếng vỗ tay reo hò của các cổ động viên. Những người tổ chức sự kiện Ngày Domino ở Hà Lan cho biết, 4.079.381 quân domino đã lần lượt đổ xuống trong vòng 2 giờ đồng hồ. Những quân domino đã di động uyển chuyển trên nền những điệu nhạc cổ điển và đương đại là nét đặc biệt nhất của màn trình diễn domino. Tác giả Robin Paul Weijers nói: “Hơn 4 triệu quân domino, điều này chưa bao giờ xảy ra. Chúng tôi còn thành công trong việc khiến cho những quân bài domino nhảy múa trong tiếng nhạc. Tôi rất hạnh phúc vì đã thành công”.

Với màn trình diễn tuyệt vời này, những kỷ lục gia domino Hà Lan đã phá vỡ kỷ lục của chính họ lập được năm 2005 với 4.002.136 quân bài domino. Sắp tới, Bờm dự định xây dựng một công trình lớn hơn để phá kỷ lục của người Hà Lan. Công trình sẽ bao gồm 2 công đoạn chính:

- Công đoạn 1: Xếp M^*N-T quân domino vào các ô còn trống trên hình chữ nhật kích thước M^*N ($M, N \leq 16$), trong hình chữ nhật đó có T ô đã được đặt trước T vật trang trí.
- Công đoạn 2: Xếp R^*L quân domino thành một dãy độ dài L ($L \leq 10^6$), mỗi hàng có đúng R ($R \leq 8$) quân (có thể được hiểu như xếp vào hình chữ nhật kích thước $R * L$).



Điểm đặc đáo trong công trình này là sự phối màu giữa các quân domino lân cận chung cạnh. Các quân domino được xếp bằng hai loại domino, loại 1 có màu xanh nhạt và loại 2 có màu xanh đậm. Quân domino ở vị trí ô (i,j) sẽ phải thỏa mãn điều kiện: nếu $i+j$ lẻ thì màu quân domino này sẽ phải có màu không nhạt hơn các quân ở các ô chung cạnh (nếu có), nếu $i+j$ chẵn thì màu quân domino này sẽ phải có màu không đậm hơn các quân ở các ô chung cạnh (nếu có).

Để xây dựng công trình, Bờm muốn biết số lượng cách xếp khác nhau của công đoạn 1 và công đoạn 2. Hai cách xếp được gọi là khác nhau nếu khi chồng khít 2 cách lên nhau (không xoay hoặc lật) có ít nhất một quân khác màu.

Input

- Dòng 1: gồm 1 số nguyên dương K ($K \leq 10^9$), các kết quả tìm được sẽ mod cho K ,
- Dòng 2: bắt đầu là 3 số nguyên dương M, N, T ($M, N \leq 16, M * N \leq T$), trong đó M, N là kích thước hình chữ nhật trong công đoạn 1, T là số lượng ô trong hình chữ nhật đã đặt vật trang trí, tiếp theo là T cặp số, cặp số (i, j) là tọa độ ô đã đặt vật trang trí;
- Dòng 3: gồm 2 số nguyên dương R, L ($R \leq 8, L \leq 10^6$) là kích thước hình của công đoạn 2.

Output

- Dòng 1: số cách xếp công đoạn 1 khác nhau mod K ;
- Dòng 2: số cách xếp công đoạn 2 khác nhau mod K .

Example

Input:

1000
5 5 1 3 3
3 10000

Output:

240
593

Chú ý: Có 50% số test $M, N \leq 10$ và $L \leq 10000$;

Bài 4: Di chuyển mã

Mã bài: KNMANO

Ván đấu cuối cùng của giải cờ vua VNOI Chess 2011 đã diễn ra, và bạn đang phải chạm trán với kì thủ số 1 của Việt Nam – đại kiện tướng quốc tế Lê Quang Liêm. Sau hơn 3 giờ chiến đấu, bạn đã tìm được 1 ô xung yếu trong ván cờ (ở vị trí T), và mục tiêu của bạn là phải chiếm được ô này càng nhanh càng tốt. Để thực hiện được việc này, bạn có một quân mã ở ô A , và dĩ nhiên, bạn mong muốn đưa quân mã này đến ô T sau càng ít bước càng tốt.



Tuy nhiên, đối phương của bạn cũng hiểu rõ tầm quan trọng của ô T, và để ngăn chặn kế hoạch của bạn, đại kiện tướng cũng có một quân mã ở ô B (A và B khác nhau và khác T). Cụ thể, sau mỗi bước, bạn và kiện tướng sẽ luân phiên di chuyển quân mã của mình. Bạn sẽ tìm cách đưa quân mã của mình đến ô T càng sớm càng tốt, trong khi kiện tướng Liêm sẽ tìm cách di chuyển quân mã của mình để ngăn chặn kế hoạch của bạn. Mục tiêu bạn cần đạt được là đưa quân mã của mình đến ô T trong thời gian ngắn nhất mà không bị quân mã của Liêm ăn được, luôn giả sử rằng Liêm luôn chơi tốt nhất có thể (lưu ý, kể cả nếu như quân mã của Liêm đến được T trước mà không thể chặn được mã của bạn, thì bạn vẫn đạt được mục tiêu).

Lưu ý:

- Bàn cờ có kích thước $8 * 8$, được chia thành 8 hàng dọc (được kí hiệu từ ‘a’ đến ‘h’ từ trái sang phải) và 8 hàng ngang (kí hiệu từ 1 đến 8 từ dưới lên trên)
- Ở mỗi bước, quân mã đi theo hình chữ ‘L’ và không bị cản. Quân mã không được phép nhảy ra khỏi bàn cờ
- Đây là thế cờ giả lập, nên bạn có thể giả sử rằng, ngoài 2 quân mã của bạn và của Liêm, tất cả những ô còn lại trên bàn cờ đều là ô trống.
- Trong 20% số test, bên Trắng luôn có chiến thuật đạt được mục tiêu, bất chấp mọi cách cản trở của Đen

Input:

- Dòng đầu tiên gồm duy nhất 1 số nguyên dương T ($T \leq 10$) là số bộ test
- T dòng tiếp theo, mỗi dòng gồm 3 xâu kí tự mô tả vị trí ban đầu quân mã của bạn, quân mã của Liêm, và ô mục tiêu. Các xâu này được cách nhau bởi đúng 1 dấu cách

Output

- Gồm T dòng, mỗi dòng in ra số bước ít nhất để quân mã của bạn đến được ô mục tiêu mà không bị ăn, hoặc -1 nếu việc đó là không thể thực hiện được.

Example:

Input

2
a1 e1 d4
h8 e5 g3

Output

2
-1

Giải thích:



- Ở bàn cờ đầu tiên, bạn đưa quân mã của mình đến ô b3, rồi đến d4 (2 nước). Lưu ý, nếu ở nước đầu tiên, bạn đưa mã đến c2, mã của Liêm sẽ ăn mã của bạn và kế hoạch thất bại

- Ở bàn cờ thứ hai, bắt kê bạn đưa mã sang f7 hay g6, mã của Liêm cũng có thể “túm” được mã của bạn, nên bạn không thể đưa được quân mã sang g3.

Bài 5: Xì trum

Mã bài: SMURFS

Lão phù thủy ác độc Gargamel đã đuổi các Xì trum nhỏ bé ra khỏi ngôi làng của họ, khiến họ phải vượt qua ranh giới giữa thế giới Xì trum huyền diệu để lọt vào thế giới hiện đại.



Các Xì trum cần phải tìm mọi cách để đánh bại Gargamel và trở về ngôi làng xinh đẹp của mình. Họ quyết định ... đến Trung Quốc học võ Thiếu Lâm.

Trụ Trì của Thiếu Lâm Tự rất khâm phục ý chí chiến đấu của các Xì trum, tuy nhiên để được nhận làm đệ tử Thiếu Lâm, các Xì trum cần phải vượt qua vòng thử thách trí tuệ của ngài, bằng cách giành chiến thắng trong trò chơi với các chuỗi tràng hạt. Sau khi bàn bạc, các Xì trum quyết định cử Brainy Smurf – người nổi tiếng thông thái của thế giới Xì trum – đứng ra thi đấu với Trụ Trì.

Các chuỗi tràng hạt trong thiếu Lâm tự là các chuỗi vòng được kết lại một cách ngẫu nhiên từ 3 loại hạt bằng gỗ quý, mỗi loại mang một màu sắc riêng: màu nâu, màu đỏ, và màu vàng. Trong trò chơi, trụ trì sẽ có N chuỗi tràng hạt. Hai người chơi sẽ lần lượt thực hiện lượt đi của mình, **Brainy được ưu tiên đi trước**.

Tại mỗi lượt đi, người chơi sẽ được lần lượt lấy ra **tối thiểu là một**, và **tối đa là 3 đoạn** gồm các hạt cùng màu **liên tiếp** trong một hoặc một số chuỗi tràng hạt. Người chơi phải lấy ra **ít nhất là 1 hạt**.

Tất nhiên, để lấy được các hạt ra khỏi chuỗi thì người chơi sẽ phải **cắt chuỗi tràng hạt** tại một vị trí thích hợp. Trong trò chơi này, quy định rằng nhát cắt chuỗi tràng hạt phải được thực hiện ở vị trí dây **nằm giữa 2 hạt khác màu nhau** (trừ trường hợp chuỗi chỉ có 1 hạt), và chỉ được cắt ở **đầu đoạn hạt cần lấy**. Sau khi cắt dây



thì chuỗi tràng hạt được duỗi ra thành một dây tràng hạt, và người chơi có thể lấy được các hạt cần lấy ở **1 trong 2 đầu dây**. Ở những lần lấy hạt tiếp theo, để lấy các hạt ở 2 đầu dây thì người chơi không cần cắt, nhưng nếu muốn lấy các hạt ở giữa dây thì người chơi vẫn phải thực hiện một nhát cắt với quy định như trên. Lưu ý: người chơi **không được phép nối** dây tràng hạt đã bị cắt lại thành chuỗi vòng như ban đầu, mà trò chơi được thực hiện tiếp với dây tràng hạt thu được.

Ai không thể thực hiện được lượt đi của mình nữa là người thua cuộc. Người còn lại sẽ giành phần thắng.

Hai người chơi, một là Trụ Trì Thiếu Lâm không những tinh thông võ học, mà còn mưu lược bậc thầy, người kia là Brainy Smurfs thông thái nổi danh khắp giới Xì trum, đều đi những nước đi tối ưu. Ai sẽ là người chiến thắng? Liệu Brainy có giúp các Xì trum vượt qua thử thách khó khăn này hay không?

Dữ liệu

- Dòng đầu chứa một số nguyên **T** là số lượng trận đấu, tiếp theo là mô tả của các trận đấu.

- Mỗi trận đấu được mô tả như sau:

- Dòng đầu chứa một số nguyên **N**, thể hiện số chuỗi tràng hạt dùng trong trò chơi.
- **N** dòng tiếp theo, mỗi dòng mô tả một chuỗi tràng hạt. Đầu tiên là một số nguyên **L** thể hiện số lượng hạt trong chuỗi, tiếp theo là 1 dấu cách, sau đó là một chuỗi gồm **L** kí tự thuộc một trong 3 loại: ‘N’, ‘D’, ‘V’, mỗi kí tự thể hiện màu của một hạt trong chuỗi tràng hạt (nâu, đỏ, vàng). Các hạt được liệt kê **theo thứ tự** tương ứng trong chuỗi tràng hạt, theo **chiều kim đồng hồ**.

Kết quả

- Gồm **T** dòng, mỗi dòng ghi kết quả của trận đấu tương ứng: in ra “Brainy” nếu Brainy giành chiến thắng, “Trụ Trì” nếu Trụ Trì giành chiến thắng.

Giới hạn

- Trong tất cả các test: $1 \leq T \leq 10$

- Trong 10% số test: $N = 1$, và $1 \leq L \leq 10$.

- Trong 20% số test: $1 \leq N \leq 100$, và $1 \leq L \leq 100$.

- Trong 70% số test: $1 \leq N \leq 1000$, và $1 \leq L \leq 1000$.

Ví dụ

Dữ liệu	Kết quả
2	
1	
5 NNDND	
1	
4 NDVD	
	Brainy Trụ Trì

