INTERVAL TREE

Bài viết này muốn giới thiệu với các bạn 1 công cụ rất hữu dụng được sử dụng nhiều trong các bài toán trên dãy số, hoặc được quy về các bài toán xử lí trên dãy số, đặc biệt là các bài toán có nhiều công việc cần xử lí và nhiều truy vấn xen kẽ nhau. Công cụ được nói đến ở đây chính là INTERVAL TREE(hay còn gọi là segmet tree), tức là cây đoạn.

Interval tree là cây đoạn, vậy thì nó là 1 cây, và các nút của nó thì lưu thông tin của 1 đoạn xác định. Interval tree là 1 cây nhị phân mà mỗi nút không phải là lá đều có đúng 2 nút con. Nếu nút A lưu thông tin của đoạn từ i..j thì 2 nút con của nó, A1 và A2, lần lượt lưu thông tin của các đoạn i..m và m+1..j, với m=(i+j) div 2 là phần tử giữa của đoạn.

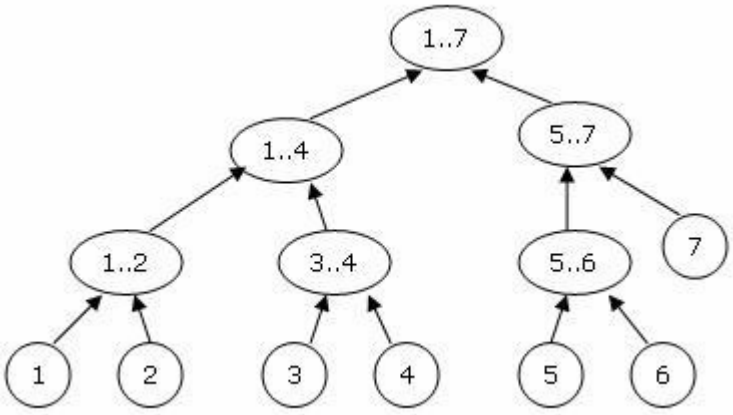
Vì đây là cây nhị phân, lại có đầy đủ 2 nút con, để đơn giản thì ta có thể biểu diễn cây chỉ = 1 mảng 1 chiều, với ý nghĩa như sau:

Nút 1 là nút gốc, nút k(nếu không phải là nút lá) thì có 2 con là các nút k*2(nút con trái) và k*2+1(nút con phải).

Nút 1 sẽ lưu thông tin của đoạn 1..n (với n là độ dài dãy số).

Vậy nút 2 sẽ lưu thông tin đoạn 1..(n+1) div 2 và nút 3 sẽ lưu thông tin đoạn (n+1) div 2+1..n.

Tương tự theo định nghĩa thì ta có thể biết được nút thứ k sẽ lưu thông tin của 1 đoạn xác định nào đấy. Sau đây là hình ảnh 1 cây INTERVAL TREE với n=7:



Có 1 vấn đề là với 1 đoạn n phần tử, thì mảng biểu diễn cây sẽ có bao nhiêu phần tử là đủ, người ta đã chứng minh được cây interval tree chỉ cần tối đa là 4*n-5 phần tử, vì vậy khi khai báo 1 cây interval tree, ta thường khai báo là 1 mảng 1..4*n.

Trên interval tree có 2 thao tác cần xử lí, 1 là cập nhật thay đổi cho đoạn i..j và 2 là lấy thông tin cần có của đoạn i..j, nhưng mỗi nút của inteval tree chỉ lưu những đoạn xác định, vì vậy 2 thao tác này có thể cần tác động đến nhiều nút.

Để dễ hình dung, ta lấy 1 ví dụ cụ thể:

Cho 1 dãy n phần tử (n<=10^5), ban đầu mỗi phần tử có giá trị 0. Có q<=10^5 truy vấn, mỗi truy vấn là 1 trong 2 thao tác sau:1 là gán giá trị v(v>0) cho phần tử ở vị trí i, 2 là tìm giá trị lớn nhất trong đoạn i..j bất kì.

Cách đơn giản nhất là ta có 1 mảng A[1..100000], với thao tác 1 thì gán A[i]=v, thao tác 2 thì cho 1 vòng for từ i đến j để tìm max, nhưng cách này trong trường hợp có nhiều truy vấn thứ 2 thì sẽ chạy rất lâu, trường hợp xấu nhất là n*q. Dễ thấy là cách này không thể chạy trong thời gian cho phép.

Cách dùng interval tree như sau:

Với truy vấn 1, ta sẽ cập nhật là kết quả max cho tất cả các đoạn mà chứa phần tử thứ i, những đoạn còn lại thì không làm gì cả vì không ảnh hưởng đến.

Với truy vấn 2, ta cần tìm kết quả max trong số tất cả những đoạn nằm gọn trong i..j, tức là những đoạn có điểm đầu >=i và điểm cuối <=j.

Gọi T[1..400000] of longint là mảng để lưu trữ cây interval tree. T[i] là giá trị lớn nhất trong đoạn mà nút k lưu giữ thông tin.

Khi gặp truy vấn 1, ta làm như trong thủ tục sau:

```
procedure truyvan1(k,l,r,i:longint);
var m:longint;
begin
1.if (l<i)or(r>i) then exit;
2 if (l=r) then
   begin
      T[k]:=v;exit;
   end;
3.m:=(l+r) div 2;
4.truyvan1(k*2,l,m,i);
5.truyvan1(k*2+1,m+1,r,i);
6.T[k]:=max(T[k*2],T[k*2+1]);
end;
```

Ta sẽ phân tích thủ tục truyvan1:

-Thủ tục này có các tham số k,l,r,i với ý nghĩa: l và r là điểm đầu và điểm cuối của đoạn mà nút k lưu trữ thông tin, i chính là phần tử cần gán giá trị v.-Trong thủ tục có 1 biến m để xác định phần tử nằm giữa đoạn l..r.

-Câu lệnh 1 có ý nghĩa là ta sẽ bỏ qua không làm gì với các đoạn không chứa phần tử thứ i(l>i hoặc r<i)

-Câu lệnh 2 tức là, khi đã đến nút thứ k biểu diễn đoạn i..i thì ta chỉ cần gán T[k]=v, vì tất nhiên sau khi gán, v sẽ là giá trị lớn nhất của đoạn i..i.

-Câu lệnh 3 xác định phần tử nằm giữa l..r.

-Câu lệnh 4 là 1 lời gọi đệ quy, để ý các tham số thì dễ dàng nhận ra, câu lệnh này gọi đến nút con trái của nút k, tức nút k*2 để ta cập nhật cho nút đó.

-Câu lệnh 5 tương tự câu lệnh 4 nhưng là gọi để cập nhật cho nút con phải.

-Câu lệnh 6: Sau khi đã cập nhật cho 2 nút con trái và phải thì đã xác định được giá trị lớn nhất từ l..m và m+1..r, vậy thì để xác định giá trị lớn nhất của đoạn l..r ta chỉ cần lấy giá trị lớn nhất của 2 đoạn kia=> T[k]:=max(T[k*2],T[k*2+1]).

Khi gọi thủ tục truy vấn 1, ta sẽ gọi từ nút gốc, tức là gọi truyvan1(1,1,n,i);

Vậy là đã xong yêu cầu thứ nhất, nhưng cái ta cần để xem chương trình có hoạt động tốt hay không lại là kết quả của yêu cầu thứ 2, vì vậy thủ tục truyvan1 là để giúp thủ tục truyvan2 sau đây tìm được kết quả với đoạn i..j bất kì.

```
Procedure truyvan2(k,l,r,i,j:longint);
var m:longint;
begin
1.if (l>j)or(r<i) then exit;
2.if (i<=l)and(j>=r) then
   begin
      res:=max(res,T[k]);exit;
   end;
3:m:=(l+r) div 2;
4:truyvan2(k*2,l,m,i,j);
5:truyvan2(k*2+1,m+1,r,i,j);
end;
```

Phân tích:Thủ tục truy vấn 2 này có các tham số với ý nghĩa như thủ tục 1, có thêm tham số j vì cần lấy kết quả trên đoạn i..j.

Mỗi đoạn l..r sẽ có 3 khả năng với đoạn i..j.

a: l..r không giao i..j, trường hợp này bỏ qua không làm gì cả(câu lệnh 1).

b: l..r nằm gọn trong i..j, trường hợp này thì ta chỉ cần tối ưu kết quả khi so sánh với T[k] vì: Ta đã xác định được phần tử lớn nhất trong đoạn l..r nhờ thủ tục 1, và do l..r nằm gọn trong i..j nên không cần đi vào 2 nút con của nó.(câu lệnh 2).

c: l..r không nằm gọn trong i..j nhưng có 1 phần giao với i..j, trường hợp này thì ta sẽ đi vào 2 nút con của nó để lấy kết quả, đến khi nào gặp phải 2 trường hợp đầu.(câu lệnh 4 và 5).

Suy nghĩ kĩ sẽ thấy thủ tục truy vấn 2 không bỏ sót cũng như tính thừa phần tử nào ngoài đoạn i..j.

Khi gọi thủ tục truyvan2 ta cũng gọi từ nút gốc truyvan2(1,1,n,i,j). Sau thủ tục thi in ra res là kết quả cần tìm.

Người ta cũng chứng minh được rằng, độ phức tạp của mỗi thủ tục truyvan1 và truyvan2 không quá log(n). Vì vậy thuật toán dùng interval tree cho bài này có độ phức tạp không quá q*log(n)=10^5*log(10^5), có thể chạy trong thời gian cho phép.

Đây là 1 trong những ví dụ cơ bản nhất về interval tree, hi vọng các bạn đã có thể hiểu và cài đặt nó.Trong khi nghiên cứu về các bài tập sau đây, ta sẽ tìm hiểu 1 vài kiểu sử dụng interval tree khác. Đây cũng là 1 trong những cấu trúc dữ liệu thường được sử dụng trong các kì thi Olympic Tin học trên thế giới.

.

Xếp hàng(NKLINEUP)

Hàng ngày khi lấy sữa, N con bò của bác John ($1 \le N \le 50000$) 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 ($1 \le Q \le 200000$) 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 qủa

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 qủa

6

3

0

Thuật toán:

Đây là 1 bài toán xử lí trên dãy số và cần truy cập đến những đoạn A..B bất kì trong dãy, vì vậy interval tree là 1 trong những lựa chọn không tồi. Bài này chúng ta có 1 điều may mắn là không cần phải cập nhật lại chiều cao của các con bò, vì vậy thông tin trong cây interval tree là cố định và ta sẽ tạo cây interval tree dựa trên mảng chiều cao của các con bò. Mỗi đoạn thì ta cần in ra chênh lệch độ cao con bò cao nhất và con bò thấp nhất, vì vậy chúng ta cần tìm được giá trị lớn nhất và giá trị nhỏ nhất trong các phần tử từ A đến B. Ta có thể dùng 1 cây interval tree với mỗi nút lưu 2 thông tin, giá trị lớn nhất và giá trị nhỏ nhất trong đoạn mà nó biểu diễn, cũng có thể dùng 2 cây

interval tree, 1 cây dùng để lưu giá trị lớn nhất, cây còn lại là giá trị nhỏ nhất. Ở đây ta gọi 2 cây này là maxd và mindPhần tạo ta có thể làm rất đơn giản dựa trên ý tưởng sau:

Nếu l=r thì Maxd[k]=Mind[k]=A[l];

Nếu không thì Maxd[k]=max(Maxd[k*2],Maxd[k*2+1]) và Mind[k]:=min(Mind[k*2],Mind[k*2+1]);

Khi muốn tìm kết quả thì dựa vào mảng Maxd để tìm GTLN trên đoạn A..B, dùng mảng Mind để tìm GTNN trên đoạn A..B, việc này làm tương tự như trong ví dụ của bài viết giới thiệu về Interval tree phía trên. Chú ý là khi tìm max hay tìm min ta đều phải đi đến những nút giống nhau(do đi đến những nút nào thì chỉ phụ thuộc A và B) nên mỗi lần tìm chỉ cần gọi chung 1 thủ tục.

Giá trị lớn nhất(QMAX)

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, q <= 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

Thuật toán:

Bài này thì ta có m phép biến đổi tăng dãy trước rồi mới yêu cầu tìm giá trị lớn nhất trong các đoạn chứ không phải xen kẽ nhau, vì vậy ta sẽ tìm cách xây dựng dãy số sau m phép biến đổi. Khi có được mảng giá trị sau m phép biến đổi, công việc còn lại của ta là tạo 1 cây interval tree với mỗi nút lưu giá trị lớn nhất của đoạn mà nó quản lí trong khi các phần tử của mảng đã xác định, với mối truy vấn tìm GTLN thì ta có thể làm không mấy khó khăn.

Vấn đề bây giờ là xây dựng dãy số sau m phép biến đổi.

Ta có thể sử dụng 1 kĩ thuật đơn giản những rất hiệu quả như sau.

Giả sử mảng ta cần có là mảng A[0..n+1], lúc đầu A[i]=0 với mọi i.

Mỗi yêu cầu u,v,k tức là tăng các phần tử từ vị trí u đến vị trí v lên k đơn vị, ta làm như sau: A[u]:=A[u]+k;A[v+1]:=A[v+1]-k;

Sau khi đọc xong m phép biến đổi và làm như trên, cuối cùng là tính mảng A:

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

Các bạn có thể tự chứng minh tính đúng đắn hay có thể viết đoạn chương trình này ra và kiểm nghiệm lại. Như vậy ta đã có thể giải quyết trọn vẹn bài toán.

Giá trị lớn nhất ver2(QMAX2)

Giống bài "Giá trị lớn nhất" ở trên.

Input

- n: số phần tử của dãy (n <= 50000).
- m: số lượng biến đổi và câu hỏi (m <= 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

Thuật toán:

Bài này khác bài QMAX ở trên là các phép biến đổi và các câu hỏi xen kẽ nhau, vì vậy ta không thể tạo trước 1 cây interval được. Để giải quyết bài toán này, ta cần phải thực hiện cập nhật cũng như lấy kết quả xen kẽ nhau. Nhưng ở đây thì các công việc này không hề đơn giản.

Vừa có được kết quả, ta cũng cần phải đảm bảo 2 công việc này không được có độ phức tạp vượt quá log(n).

Để làm bài này, ta gọi F[k] là giá trị lớn nhất trong đoạn mà nút k quản lí, trong lúc cập nhật, muốn đảm bảo thủ tục này không vượt quá log(n) thì khi đi đến 1 nút mà nằm gọn trong đoạn x..y thì ta không được đi vào các nút con của nó nữa(nếu không sẽ đi đến tất cả các đoạn có trong đoạn x..y và độ phức tạp tỷ lệ với n). Nếu như vậy, ta cần có 1 mảng phụ T để lưu lại giá trị cần tăng lên cho tất cả các phần tử của đoạn này, khi ta truy vấn đến 1 nút k đồng thời ta tăng T[k] lên cho T[k*2], T[k*2+1] và F[k]. (do T[k] là giá trị cần tăng lên cho tất cả những phần tử của đoạn nút k quản lí, nên các nút con của nó cũng phải tăng lên T[k]). Sau khi đã cập nhật cho mảng F và các nút con, ta gán lại T[k]=0 để tránh trường hợp cộng lại nhiều lần. Nếu đã đến đoạn nằm gọn trong đoạn x..y thì ta tăng mỗi biến F[k], T[k*2], T[k*2+1] lên v. Khi muốn lấy kết quả, ta vẫn làm như bài QMAX, nhưng đến mỗi nút con, ta vẫn cần thực hiện tăng giá trị T[k] cho T[k*2], T[k*2+1] và F[k] để lấy được kết quả. Tức là khi đến được 1 nút nào đó(trong bất kì thao tác nào, cập nhật hay

lấy kết quả) thì đều thực hiện như vậy. Điều này đảm bảo là ta có thể lấy được giá trị chính xác, đây là kiểu cây IT có thông tin truyền từ nút cha xuống nút con. Các bạn có thể tham khảo chương trình để thấy rõ hơn.

Bật đèn(LITES)

Bác John giữ cho đàn bò thông minh bằng cách để chúng chơi các đồ chơi phát triển trí tuệ. Một trong các trò chơi là các ngọn đèn trong chuồng. Mỗi trong số N (2 <= N <= 100,000) con bò được đánh số từ 1..N có treo một ngọn đèn màu.

Vào đầu buổi tối, tất cả đèn đều tắt. Đàn bò điều khiển các ngọn đèn bằng N công tắc; bấm công tắc i đổi trạng thái của đèn i từ tắt sang bật hoặc ngược lại.

Đàn bò đọc và thực thi một danh sách gồm M (1 <= M <= 100,000) thao tác mô tả bởi một trong hai số nguyên (0 <= thao tác <= 1).

Thao tác thứ nhất (mô tả bởi số 0) theo sau bởi hai số nguyên S_i và E_i (1 <= S_i <= E_i <= N) cho biết công tắc đầu và công tắc cuối. Đàn bò sẽ bấm mỗi công tắc từ S_i đến E_i đúng một lần.

Thao tác thứ hai (mô tả bởi số 1) yêu cầu đàn bò đến xem có bao nhiêu ngọn đèn giữa S_i và E_i (1 <= S_i <= E_i <= N) đang bật. Hãy giúp bác John đảm bảo rằng đàn bò trả lời đúng bằng cách xử lý danh sách và trả về các kết quả đúng.

Dữ liệu

* Dòng 1: Hai số nguyên cách nhau bởi khoảng trắng: N và M

* Dòng 2..M+1: Mỗi dòng chứa một thao tác với ba số nguyên cách nhau bởi khoảng trắng: thao tác, S_i, và E_i

Kết quả

* Dòng 1..số truy vấn: Với mỗi truy vấn, in ra kết quả là một số nguyên trên một dòng.

Ví dụ

Dữ liệu:
4 5
0 1 2
0 2 4
1 2 3
0 2 4
1 1 4

Kết quả:
1
2

Thuật toán:

Ta có thể sử dụng INTERVAL TREE để làm như sau:

Bài này có tư tưởng khá giống với bài QMAX2, thông tin được truyền từ nút cha xuống nút con bất cứ khi nào có thể. Ta có 1 mảng B kiểu logic, B[i]=true nghĩa là những đèn trong đoạn mà nút thứ i quản lí cần thay đổi trạng thái. Khi đến 1 nút, nếu b[i]=true thì cần thay đổi trạng thái của B[i*2] và B[i*2+1] đồng thời gán lại B[i]=false(đã thực hiện thay đổi rồi thì phải gán lại false), nếu gọi mảng sl[i] là số lượng đèn đang bật trong đoạn do nút i quản lí thì nếu gặp b[i]=true phải tiến hành sửa sl[i], sl[i] hiện tại là giá trị của số đèn đang tắt(do khi thay đổi trạng thái thì bật chuyển thàng tắt). Vậy chỉ cần 1 phép trừ là có thể tính được số lượng đèn đang bật trong đoạn. Các công việc này cần làm ở đầu ở cả 2 thủ tục cập nhật và lấy kết quả.

Khi thực hiện thao tác cập nhật kết quả cho đoạn x..y thì nếu gặp những đoạn nằm gọn trong x..y thì ta lại thực hiện thay đổi trạng thái như trên. Phần lấy kết quả thì không khó, các bạn chỉ cần nhớ là trong thủ tục lấy kết quả cũng cần thực hiện truyền thông tin từ nút cha cho nút con(bất cứ khi nào có thể).

A Marble Game(MARBLE)

Trong những ngày hè rảnh rỗi, ktuan thường chơi bắn bi trên một bảng hình vuông gồm NxN ô vuông nhỏ. Trò chơi được thực hiện như sau:

- Ban đầu, ktuan đặt K vật cản vào K ô vuông của bảng.
- Sau đó, ktuan thực hiện lần lượt Q lượt chơi. Ở lượt chơi thứ i, ktuan lần lượt bắn D_i viên bi từ ngoài bảng vào một trong 4 đường biên của bảng. Kích thước của mỗi viên bi đúng bằng kích thước của một ô vuông nhỏ. Viên bi sẽ đi qua các ô thuộc cùng một hàng / cột cho đến khi đi ra ngoài bảng hoặc gặp một vật cản hay viên bi khác. Nếu có vật cản hay viên bi khác ở ngay ô đầu tiên thì viên bi đó không được đưa vào bảng.
- Ở mỗi lượt bắn, ktuan ghi lại tổng số ô mà các viên bi đã đi qua.

Bạn hãy viết chương trình mô phỏng lại trò chơi và với mỗi lượt bắn, in ra tổng số ô mà các viên bi của lượt bắn đó đã đi qua.

Dữ liệu

- Dòng đầu ghi 3 số N, K, Q.
- K dòng sau, mỗi dòng ghi một cặp số (u,v) thể hiện toạ độ (dòng, cột) của một vật cản.

- Q dòng sau, mỗi dòng ghi 4 giá trị c, D, u, v. Ký tự c có thể là 'L', 'R', 'T', hoặc 'B' cho biết viên bi được đưa vào từ biên trái, phải, trên hoặc dưới của bảng. (u,v) thể hiện toạ độ ô đầu tiên mà viên bi được đưa vào. Đây phải là một ô nằm trên biên của bảng ứng với ký tự c. D là số lượng viên bi sẽ bắn ở lượt chơi này.

Kết quả

- Với mỗi lượt chơi, in ra tổng số ô mà các viên bi của lượt đó đã đi qua

Ví dụ

Dữ liệu
5 1 3
3 3

L 2 3 1
T 1 1 1
B 5 5 5

Kết quả
3
2
25

Giải thích

- Viên bi đầu tiên của lượt 1 sẽ đi qua 2 ô (3,1) và (3,2) trước ghi gặp vật cản ở ô (3,3)
- Viên bi tiếp theo của lượt 1 sẽ đi qua ô (3,1) trước khi gặp viên bi ở ô (3,2). Vậy tổng số ô bị đi qua ở lượt này là 3.
- Viên bi đầu tiên của lượt 2 sẽ đi qua 2 ô (1,1) và (2,1) trước khi gặp viên bi ở ô (3,1).
- Mỗi viên bi của lượt cuối cùng đều không gặp vật cản và sẽ đi ra ngoài bảng sau khi đi qua 5 ô.

Giới hạn

- $N \leq 50000$, $K \leq 10$, $Q \leq 100000$
- Trong 1/3 số test, N và Q không vượt quá 1000.
- Thời gian: 3-10s/test.

Thuật toán:

-Với mỗi lần bắn, ta cần phải tìm được ô chướng ngại vật đầu tiên. Bài này cũng có tư tưởng tương tự bài QMAX2, thông tin truyền từ nút cha xuống nút con, nhưng 1 điều đặc biệt là chúng ta chỉ cần thông tin của những đoạn 1 phần tử(do bắn chỉ trên 1 đường thẳng). Vì vậy thông tin chỉ được truyền xuống mà không truyền lên. Thao tác tìm chướng ngại vật đầu tiên trên đường bắn thì không có gì khó khăn.(cứ đi mãi cho đến nút quản lí hàng/cột hiện tại). Phần còn lại là phải cập nhật sau

khi bắn, thao tác này làm như QMAX2, phải truyền thông tin cho các nút con bất cứ khi nào có thể. Vậy ta cần 4 cây INTERVAL TREE, H1,H2,C1,C2 với ý nghĩa:

H1: Quản lí theo hàng, giả sử nút k quản lí từ hàng i đến hàng j thì H1[k] là chỉ số cột nhỏ nhất trong các chỉ số cột có chướng ngại vật từ hàng i đến hàng j. H1 dùng để xác định ô chướng ngại vật đầu tiên trong thao tác bắn L.

H2: Tương tự H1 nhưng thay bằng chỉ số lớn nhất. H2 dùng để xác định ô chướng ngại vật đầu tiên trong thao tác bắn R.

C1: Tương tự H1 những C1 quản lí các cột và C1[k] là chỉ số hàng nhỏ nhất có chướng ngại vật từ cột i đến cột j. C1 dùng để xác định ô chướng ngại vật đầu tiên trong thao tác bắn T.

C2: Tương tự C1 nhưng là chỉ số hàng lớn nhất. C2 dùng để xác định ô chướng ngại vật đầu tiên trong thao tác bắn B.

Các bạn cũng có thể thay 4 cây bằng 2 cây(H và C) nhưng mỗi cây có 2 thông tin là chỉ số nhỏ nhất và lớn nhất của cột/hàng.

Mỗi khi có thao tác bắn L và R, sau khi xác định được hàng i sẽ thêm những viên bi từ vị trí nào đến vị trí nào thì thực hiện cập nhật trên 2 mảng H1, H2.Tương tự với 2 thao tác còn lai, lúc này ta cập nhật trên 2 mảng C1, C2.Cuối cùng xin lưu ý các bạn những trường hợp mà không có chướng ngại vật trên hàng hoặc cột, khi đó ta không cần cập nhật, và kết quả chính là d*n. Các bạn cũng cần chú ý những trường hợp mà số ô đưa vào nhiều hơn số ô tối đa có thể đưa vào hàng/cột, lúc này ta chỉ kết quả cho những ô đưa được vào bảng thôi(1 số ô bị bắn ra ngay khi gặp chướng ngại vật ở biên). Bài này các bạn nên khai báo kết quả kiểu int64 để tránh bị tràn số.