CHALLENGES

PRACTICE

COMPANIES

SOLVE

All Tracks > Data Structures > Advanced Data Structures > Segment Trees > Problem

Good Evening sweetheart!! [Medium]

Attempted by: 87 / Accuracy: 70% / Maximum Score: 30 /

2 Votes

Tag(s): Medium

PROBLEM

EDITORIAL

MY SUBMISSIONS

ANALYTICS

The obvious greedy solution is to choose the largest c[i] candles during the i^{th} evening. But the straight forward implementation will obviously get TLE.

We need a fast way of:

deciding which candles to use decreasing their height by 1 The types of operations involved hint that we need some data structure, probably a segment tree. The idea here is to sort the candles initially decreasing by height, and then choose to decrement the candles in a way that keeps them sorted.

Let's take an example, where we have candles of heights [5, 5, 4, 3, 3, 3, 3, 1] and we need 5 candles. If we choose the first 5, we get the resulting array [4, 4, 3, 2, 2, 3, 3, 3, 1] which is no longer sorted. Instead, we could decrease the first 3 elements, plus the last two elements equal to 3, getting [4, 4, 3, 3, 3, 3, 2, 2, 1] This way we preserve the values in decreasing order.

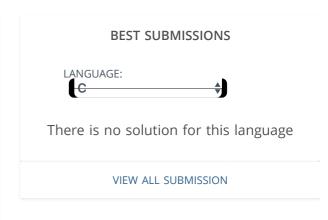
Now, let's see what operations the segment tree should support:

https://www.hackerearth.com/practice/data-structures/advanced-data...ees/practice-problems/algorithm/good-evening-sweetheart/editorial/

Decrement some range of elements Find the value of a given elements Find the first/last occurrence of a given value These are classical updates/queries.

For each evening we should do the following:

First find the value x on position c[i]



CONTRIBUTOR



SOCIAL SHARE

F

4

in

G+

Then find the first and the last occurrences of x Finally, having this information, we can decrement the prefix that ends before the first x, and then some of the last elements equal to x, as needed.

Code for same in C++:

```
include<br/>
<br/>bits/stdc++.h>
using namespace std;
#define int long long
#define rep(i,n) for(int i=0;i<(n);i++)
#define pb push back
#define all(v) (v).begin(),(v).end()
#define fi first
#define se second
typedef vector<int>vint;
typedef pair<int,int>pint;
typedef vector<pint>vpint;
template<typename A, typename B>inline void
chmin(A \&a,B b){if(a>b)a=b;}
template<typename A, typename B>inline void
chmax(A &a,B b){if(a<b)a=b;}</pre>
struct segtree{
    static const int SEG=1<<17;</pre>
    vint dat, put;
    segtree():dat(SEG*2),put(SEG*2){}
    inline void push(int k,int l,int r){
         dat[k] += put[k]*(r-1);
         if(k<SEG-1){
              put[k*2+1]+=put[k];
              put[k*2+2]+=put[k];
         put[k]=0;
    void update(int a,int b,int x,int
k=0, int l=0, int r=SEG) {
         push(k,l,r);
         if(r<=a|b<=1)return;</pre>
         if(a<=1&&r<=b){
              put[k]+=x;
              push(k,l,r);
              return;
         update(a,b,x,k*2+1,1,(1+r)/2);
         update(a,b,x,k*2+2,(1+r)/2,r);
         dat[k]=dat[k*2+1]+dat[k*2+2];
    int query(int a,int b,int k=0,int
1=0, int r=SEG) {
```

```
push(k,l,r);
         if(r<=a||b<=1)return 0;
         if(a<=l&&r<=b)return dat[k];</pre>
         return query(a,b,k*2+1,1,(1+r)/2)
         +query(a,b,k*2+2,(1+r)/2,r);
    }
};
int N,M;
int H[111111],C[111111];
signed main(){
    cin >> N >> M;
    rep(i,N)cin>>H[i];
    rep(i,M)cin>>C[i];
   sort(H,H+N);
    segtree seg;
    rep(i,N)seg.update(i,i+1,H[i]);
    rep(i,M){
         int x=seg.query(N-C[i],N-C[i]+1);
         int lb=-1,ub=N-C[i];
         while(ub-lb>1){
              int mid=(ub+lb)/2;
if(seq.query(mid, mid+1) == x)ub=mid;
              else lb=mid;
         int l=ub;
         lb=N-C[i],ub=N;
         while(ub-lb>1){
              int mid=(ub+lb)/2;
if(seg.query(mid,mid+1)==x)lb=mid;
              else ub=mid;
         int r=ub;
         if(x==0){
              cout<<i<<endl;
              return 0;
         seg.update(r,N,-1);
         seg.update(l,l+C[i]-(N-r),-1);
    cout<<M<<endl;
}
```

IS THIS EDITORIAL HELPFUL?