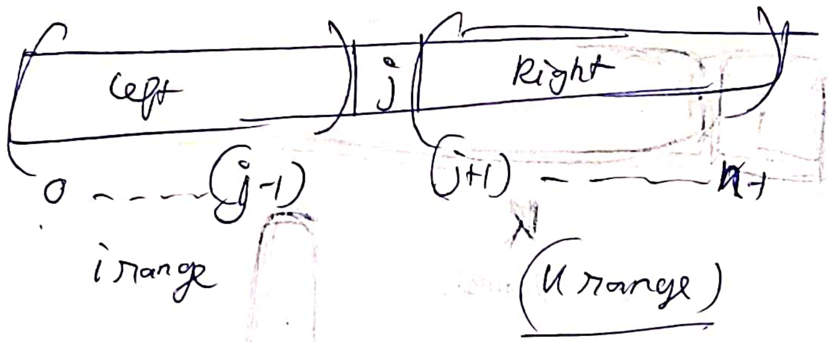
Seeing $(i < j < k)$

the format which came to my mind:



So in the left find the max for the j

in the right find the max for the j

But the twist comes with a condition

So

$prices[i] < prices[j] < prices[k]$.

then we can do a

ans $= (profits[i] + profits[j] + profits[k])$

So finding $\max(profits[i])$ for $i<j$

and $\max(profits[k])$ for $j<k$

won't work, because we also have to ensure

$prices[i] < prices[j] < prices[k]$.

So I thought about the problem a little hard,

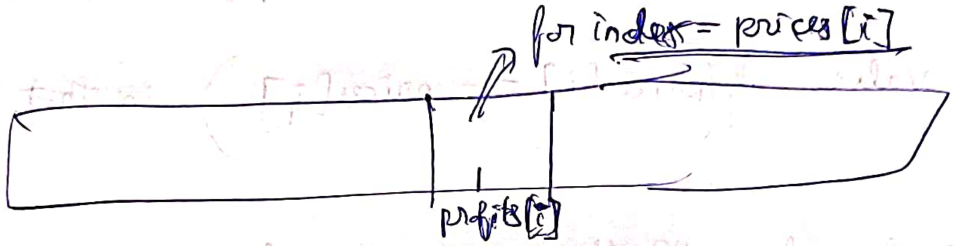
then an idea came to me.

What if?

For left side → i range

we use prices[i] as the index

we can use



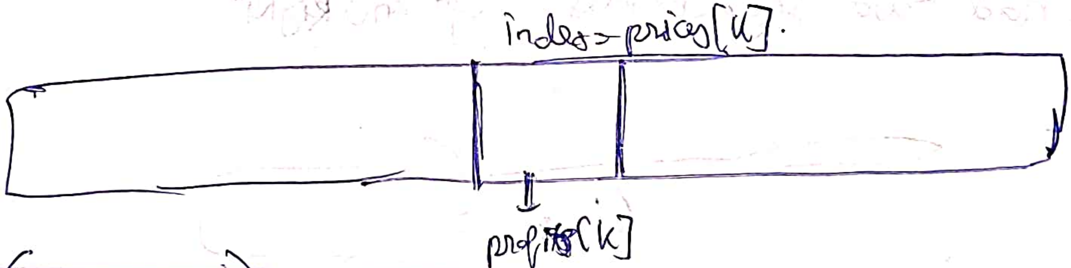
then for i range search, we can do a

(0 - prices[i] -1) area max find,

to find the maximum.

Similarly for

right side → k range



In this range, we can do a

max of (prices[j]+1, max_val)

profits[k] in this range

So I built a sparse segment tree, to store the max.

Now Our ~~use~~ use-case is prices[i] → to store profits[i].

Now its possible that a ~~pref~~ prices[i_1], we have a profit[i_1]

and the value of (prices[i] == prices[i_1]) in that case, we have

use

node.val = Math.max(node.val, ~~profit~~ val);

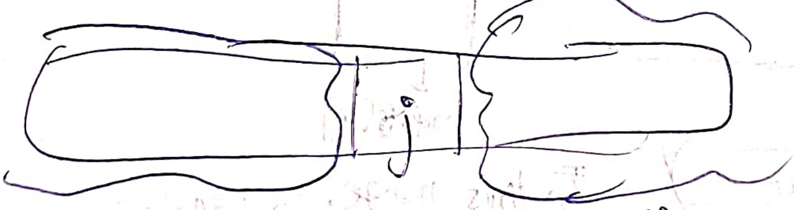
⇓

{new val that is going to get updated}

Coming back to our code

1st attempt.

I had two segment trees, left and right



left will have elements in this range

we update

arr[prices[i]] = Math.max(arr[prices[i]], profits[i]);

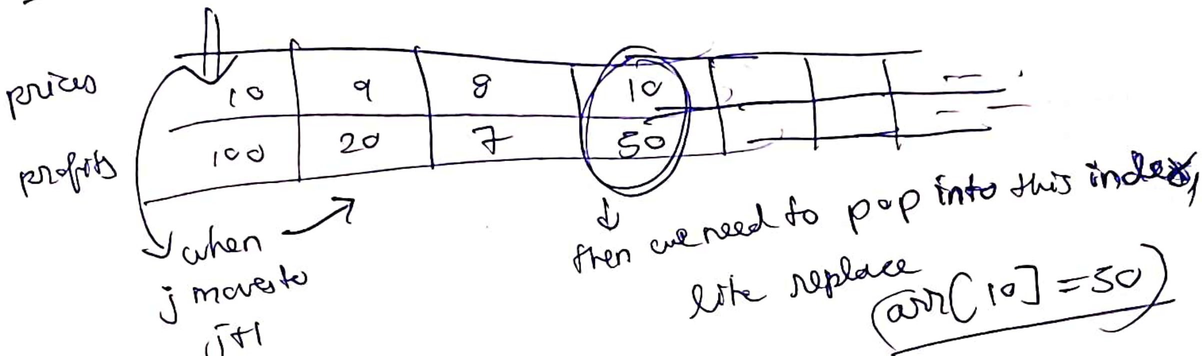
having default value as -1.

And for Right we had segment tree initially full but as j was progressing, we had to remove that from the tree

P.T.O.

Problem, with removing the element with the right segment is that we have to maintain what next value we update that index with.

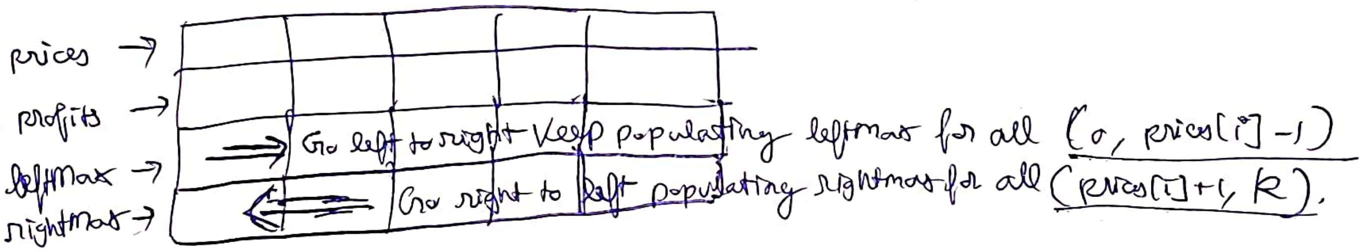
Example



To do that we had to maintain what ~~we~~ next value for prices[i] will be.

2nd attempt

After seeing one solution. Realized my attempt was complicating it too much, hence better approach keep two arrays



~~So~~ then for every index j just check

leftmax[j] != -1 && rightmax[j] != -1

ans = Max(ans, leftMax[j] + profits[j] + rightmax[j]);