Nearest Smaller Element

Largest Rectangle in Histogram

(Max - Min) in Subarray

Nearest Smaller Element

Given an integer array A, find index of nearest smaller element on left for all i index in A[].
For all i, find j such that j<i and A[j] < A[i] and j is maximum.

	0	1	2	3	4	5	6	7
A[] →	8	2	4	9	7	6	3	10
NSL →	-1	-1	2	4	4	4	2	3
idx →	-1	-1	1	2	2	2	1	6

Brute Force:

For every i, calculate NSL

↓

we will travel from i-1 to 0

↓

return first element smaller than A[i]

TC: $O(N^2)$

SC: $O(1)$

Optimized:

For any idx >5 can idx 0 become nearest smaller element on left?

NO

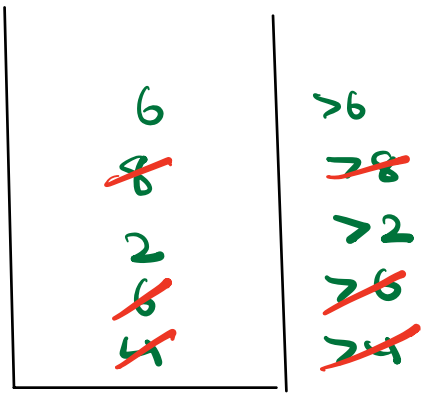
8 → ans etc >8

Ex 1.

4	6	2	8	6
-1	4	-1	2	2

x > 6 > 2

x > 8 > 6



6 4

↓

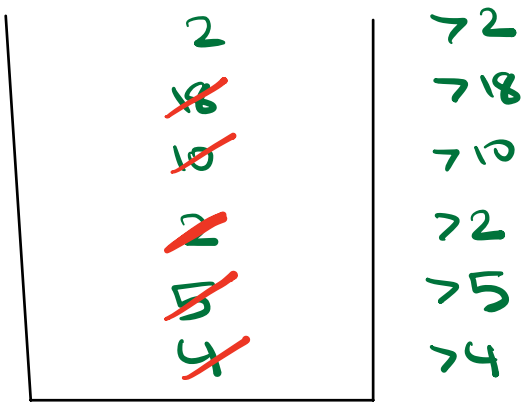
>6 4-6

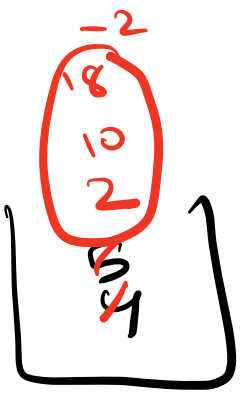
Stack contains potential ans elements

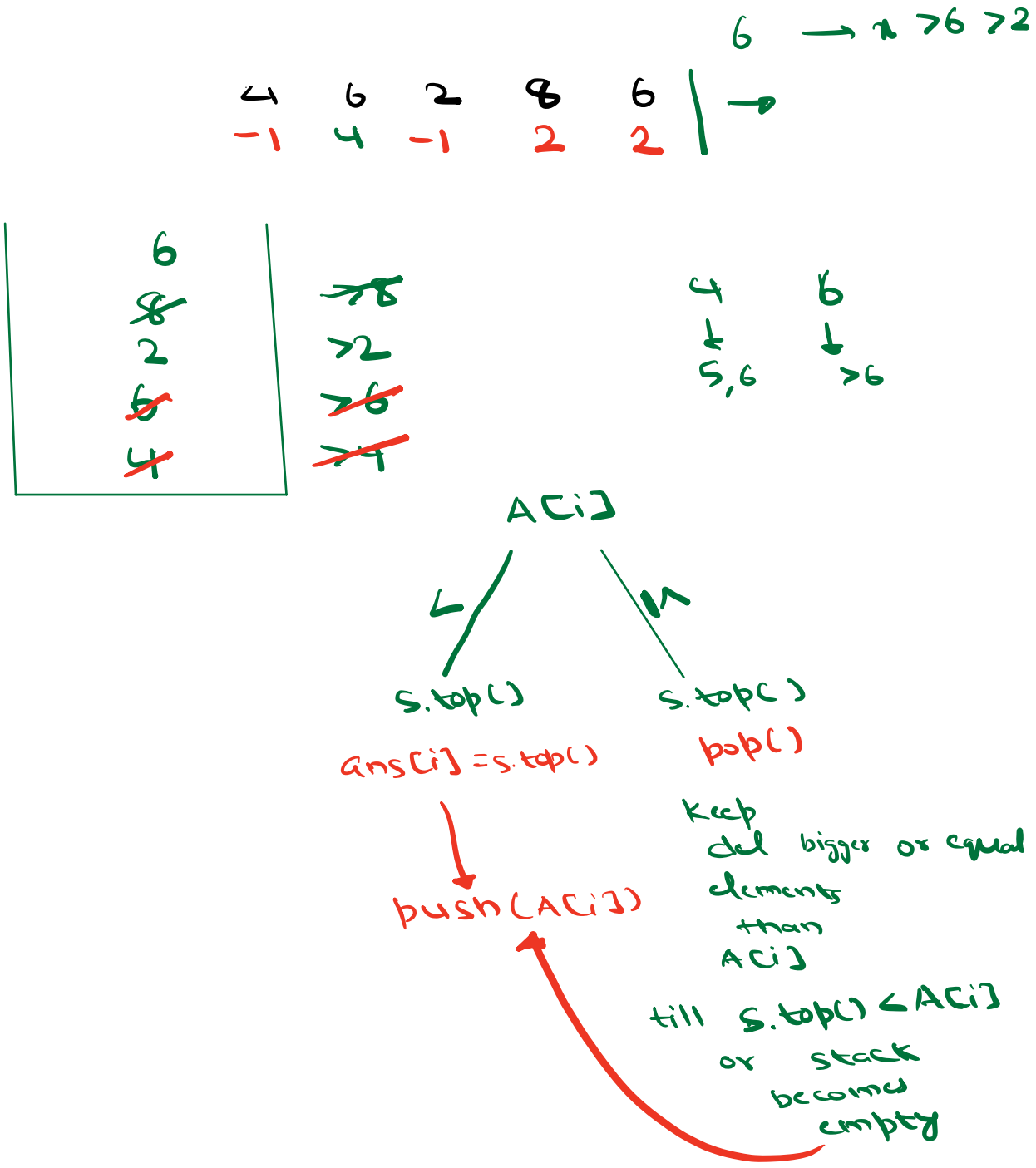
Ele are in increasing order from bottom to top

Ex 2

4	5	2	10	18	2
-1	4	-1	2	10	-1



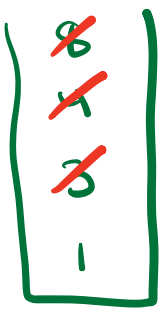



6 → 4 >6 >2
4 6 2 8 6
-1 4 -1 2 2
6
8
2
6
4
>8
>2
>6
>4
4
5,6
6
>6
A[i]
s.top()
ans[i] = s.top()
push(A[i])
s.top()
pop()
Keep del bigger or equal elements than A[i]
till s.top() < A[i] or stack becomes empty

```
// A[], int n
int ans[n]
stack<int> st
for (i=0 ; i<n ; i++) {
    // A[i]
    while (!st.empty() && st.top()>=A[i]){
        st.pop()
    }
    if (st.empty()) {
        ans[i]=-1
    }
    else {
        ans[i] = st.top()
    }
    st.push(A[i])
}
```

TC : O(N)
SC : O(N)

(2)



	0	1	2	3	4	5
A[] →	4	5	2	10	18	2
NSL →	-1	4	-1	2	10	-1
NSL idx	-1	0	-1	2	3	-1

```
int ans[n]
stack<int> st
for(i=0 ; i<n ; i++){
    // A[i]
①   while(!st.empty() && A[st.top()] >= A[i]){
②       st.pop()
    }
③   if(st.empty()){
④       ans[i]=-1
    }
    else{
⑤       ans[i] = st.top()
    }
⑥   st.push( i )
}
```

NSL

A[i]

stack ≥ A[i] → del

stack < A[i]

i → 0 to n-1

Q2. Find nearest smaller or equal element on left

Condition in line 1 changes $A[st.top()] > A[i]$

(don't delete duplicate from stack)

Q3. Find nearest greater element on left

$A[st.top()] <= A[i]$

Q4. Find nearest greater or equal element on left

$A[st.top()] < A[i]$

Q5. Find nearest smaller element on right

Reverse for loop $(i = n-1 ; i \geq 0 ; i--)$

Q6. Find nearest smaller or equal element on right

$A[st.top()] > A[i]$

Q7. Find nearest greater element on right

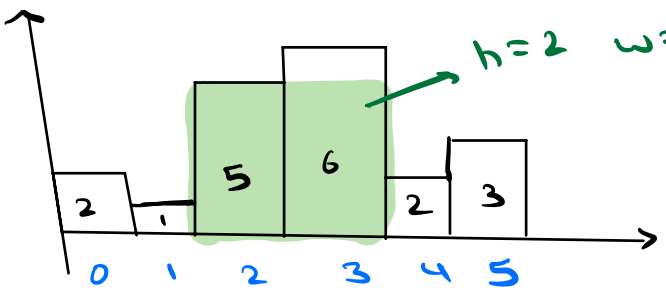
$A[st.top()] <= A[i]$

Q8. Find nearest greater or equal element on right

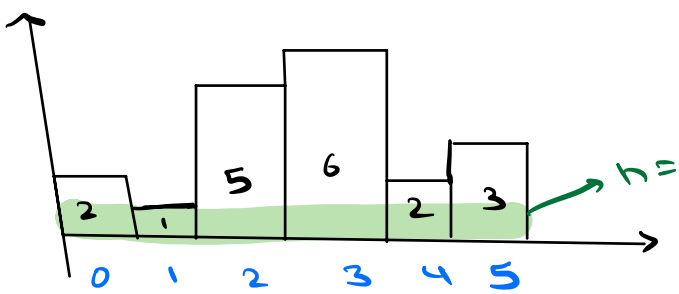
$A[st.top()] < A[i]$

10:28

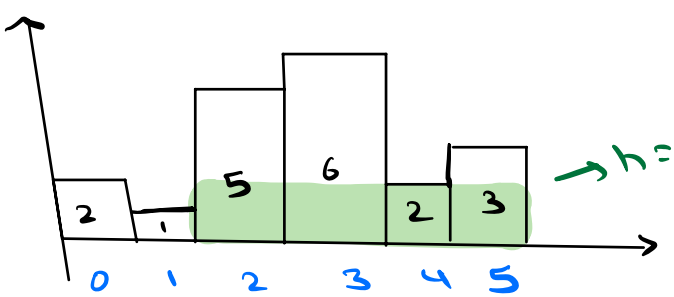
Q. Given an integer array A where
A[i] = height of i^{th} bar
width of each bar = 1
Find area of largest rectangle formed by continuous bars.



h = 2 w = 2
Area = 10

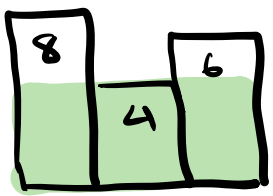


h = 1 w = 6
Area = 6



h = 2 w = 4
Area = 8

h = min of rectangle involved



h = 4
w = 3
Area = 12

Rectangle is b/w multiple bars

↓

left bar and right bar

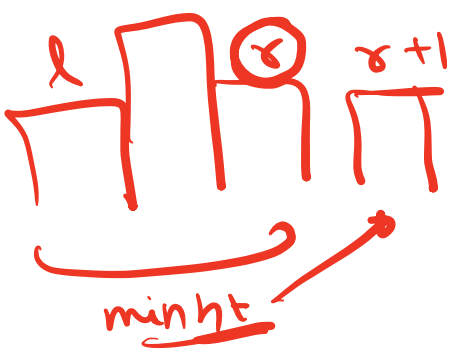
BF → Consider all pairs of left and right

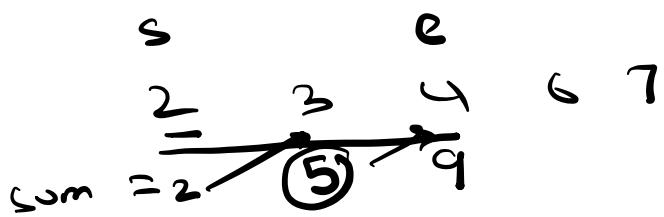
[a, b]

l → r

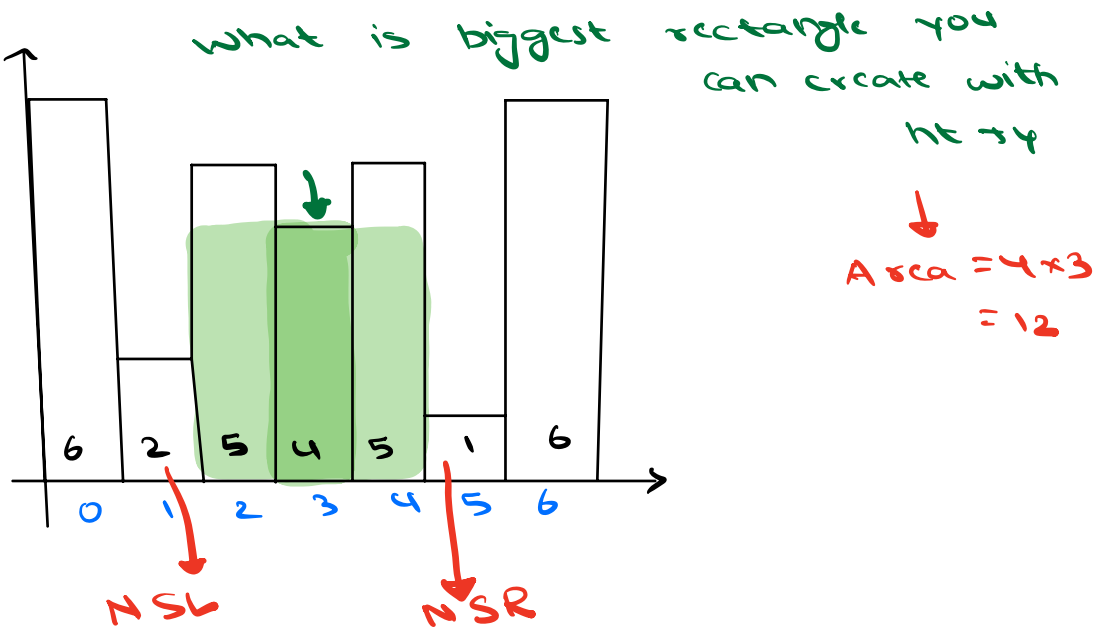
TC: $O(N^2)$

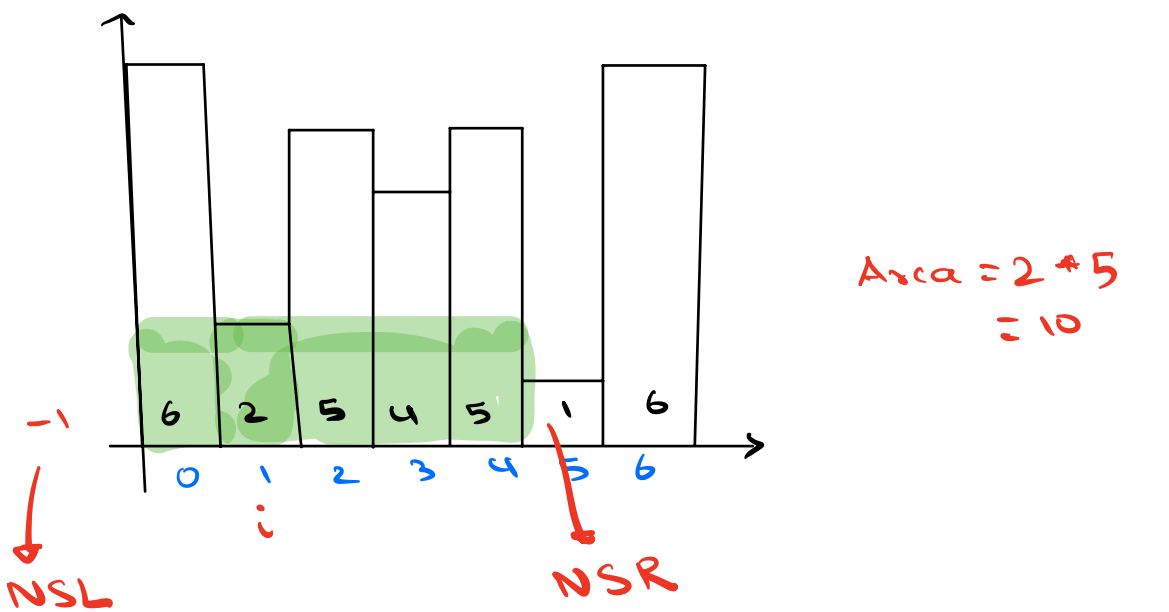
```
for (l = 0 ; l < N ; l++) {
    minht = height[l]
    for (r = l ; r < N ; r++) {
        minht = min (minht, height[r])
        width = r - l + 1
        area = max (area, minht * width)
    }
}
```

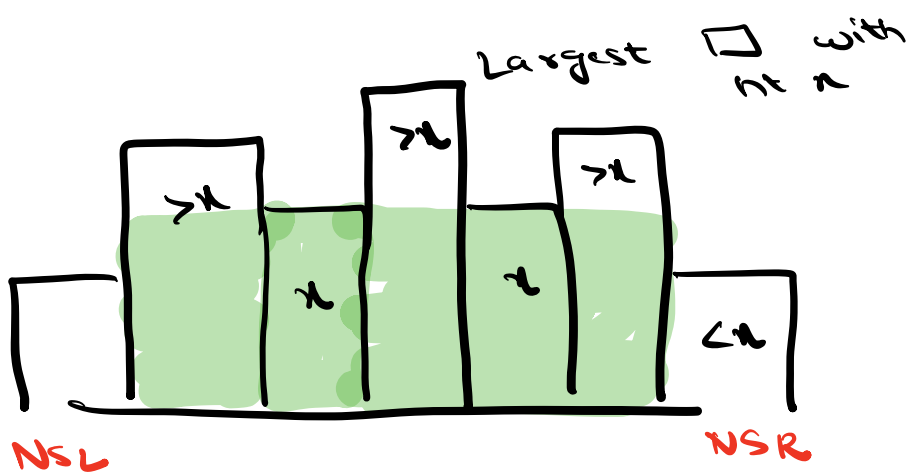


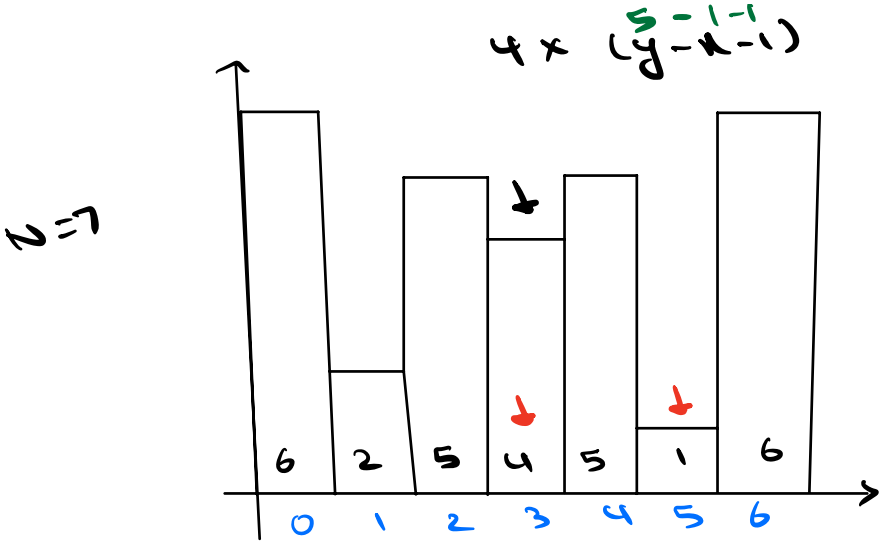


Optimized - Height of max area rectangle must be equal to height of one of the bars.


What is biggest rectangle you can create with ht = 4
Area = 4 * 3
= 12
6 2 5 4 5 1 6
0 1 2 3 4 5 6
NSL
NSR

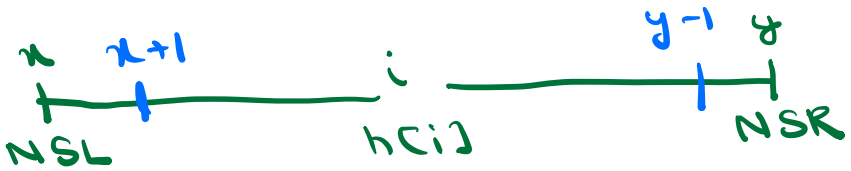

Area = 2 * 5
= 10
6 2 5 4 5 1 6
-1
0 1 2 3 4 5 6
i
NSL
NSR


Largest □ with ht x
>x
x
x
<x
NSL
NSR



For NSR_I if there's no such value $NSR_I = N$

[a, b] → b - a + 1



h[i] width → [x+1, y-1]

$= y-1-(x+1)+1$

$= y-1-x-1+1$

$width = y-x-1$

$Area = h[i] * (y-x-1)$

$= h[i] * (NSR[i] - NSL[i] - 1)$

idx idx

// h[], N

nSL[]

nSR[] (-1 → N)

```
for (i=0 ; i< N ; i++) {

    // what is largest □ that
       we can build with h[i]

    width = NSR[i] - NSL[i] - 1

    area = max (area, width * h[i])

}
```

TC: O(N)

SC: O(N)

Q. Given an integer array with distinct integers, find sum of (max - min) for all subarrays.

	0	1	2
A[ ] →	2	5	3

S	E	Max	Min	Max - Min
0	0	2	2	+2-2 0
0	1	5	2	+5-2 3
0	2	5	2	+5-2 3
1	1	5	5	+5-5 0
1	2	5	3	+5-3 2
2	2	3	3	+3-3 0

$2(1-3) + 5(4-1) + 3(1-2)$

$-4 + 15 + -3$

ans → 8

BF : Generate all subarrays

↓

Iterate it to get max, min, diff

↓

Add it to sum

```
for (s ——————— )
    for (e ——————— ) {
        for (i=s ; i<=e ; i++)
            max - min
    }
```

TC : $O(N^3)$

SC : $O(1)$

Optimized : Contribution of each ele

$$\sum (max_s - min)$$

ele → x

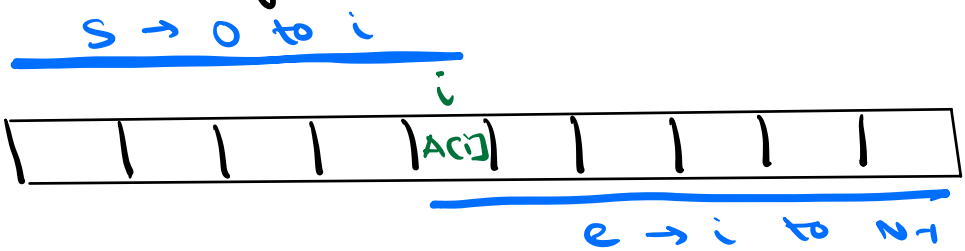
No. of subarrays in which ele x is maxm → a

x is minm → b

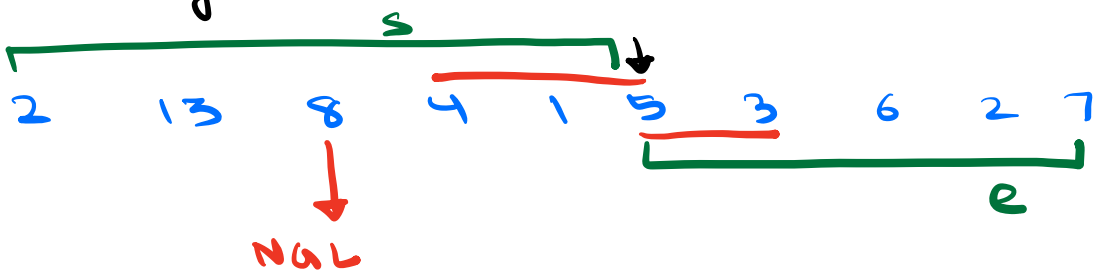
Contribution of x = $x(a-b)$

To find :

In how many subarrays A[i] is max



In how many subarrays 5 is max



S (3) e (2)

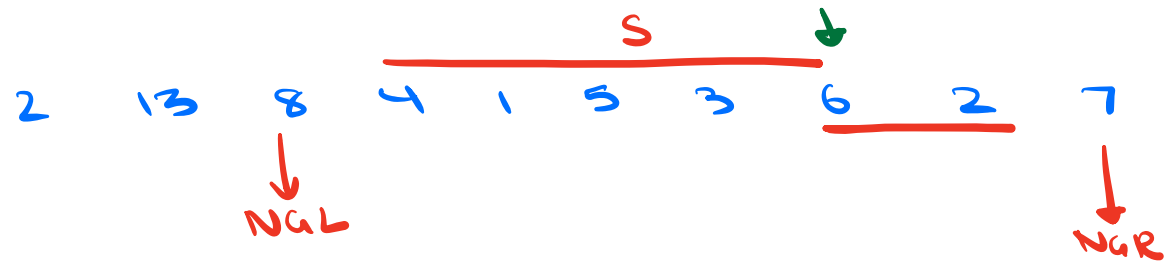
4 5

1 3

5

Ans = 3 × 2 = 6

In how many subarrays 6 is max



S (5) e (2)

4 6

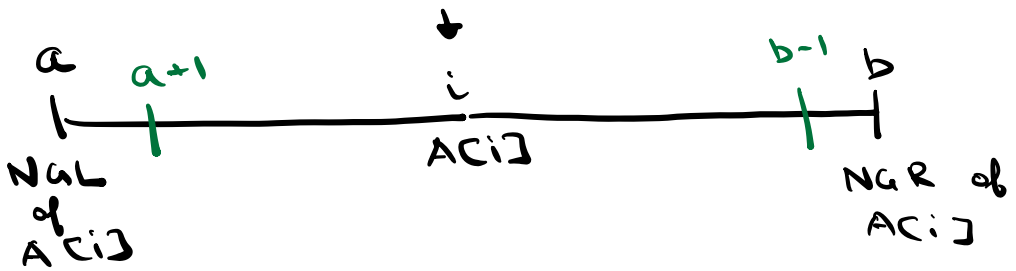
1

5 2

3

6

ans = 5 × 2 = 10



S → [a+1, i] → i − a − X + X

e → [i, b−1] → b − X − i + X

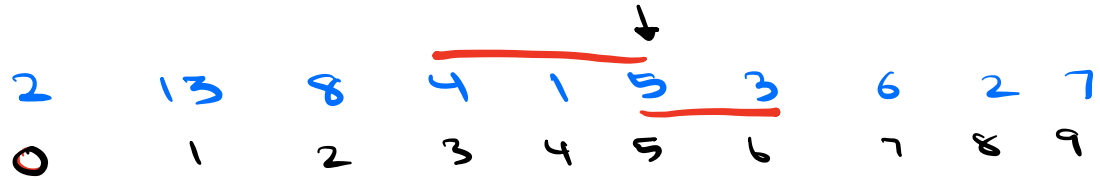
S e

i − a b − i

No. of subarrays = $(i-a)(b-i)$

$= (i - NGL[i])(NGR[i]-i)$

In how many subarrays 5 is max



NGL[i] → 2 NGR[i] → 7

No. = (5 - 2)(7-5)

= 3 * 2 = 6

```
int sum = 0
NGL[], NGR[], NSL[], NSR[]

for (i = 0 ; i < N ; i++) {

    No. of max = (i - NGL[i])(NGR[i]-i)
    No. of min = (i - NSL[i])(NSR[i]-i)

    sum += A[i] * (No. of Max - No. of Min)

}
```

TC : O(N)

SC : O(N)