Nearest Smaller Element

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

	0	1	2	3	4	5	6	7
ex →	8	2	4	9	7	6	3	10
elem	-1	-1	2	4	4	4	2	3
idx →	-1	-1	1	2	2	2	1	6

Brute force :- for every i, we will travel from i-1 to 0, when first element smaller than arr[i];

Optimized idea :-

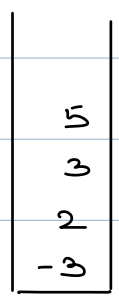
arr[] = [8, *, *, *, *, 5, *, *, *, *]

4, 6, 2, 8, 6

-1 4 -1 2 2

4, 5, 2, 10, 18, 2

-1 4 -1 2 10 -1



st.top()

< A[i]

ans[i] = st.top();
st.push(ans[i])

>= A[i]

st.pop(),
till stack is
empty, or
st.top() < A[i].

```
ans -> [];                                   T.C -> O(n)
st -> stack();                               S.C -> O(n) -> stack.
for  i -> 0  to  n-1
    while (!st.empty() && st.top >= A[i]){
        st.pop();
    }
    if (st.isEmpty()){
        ans[i] = -1;
    }
    else{
        ans[i] = st.top();
    }
    st.push(A[i]);
}
```

	0	1	2	3	4	5
arr →	4,	5,	2	, 10,	18,	2
nsh →	-1	4	-1	2	10	-1
nsRight →	-1	0	-1	2	3	-1 ✓

ans → [];

st → stack();

T.C → O(n)

S.C → O(n)

for i → 0 to n-1

while (!st.empty() && A[st.top] >= A[i]){

st.pop();

}

if (st.isEmpty()){

ans[i] = -1;

}

else {

ans[i] = st.top();

}

st.push(i);

}

Ques Get the dist of nth. ✓

Ques find nearest smaller to right.

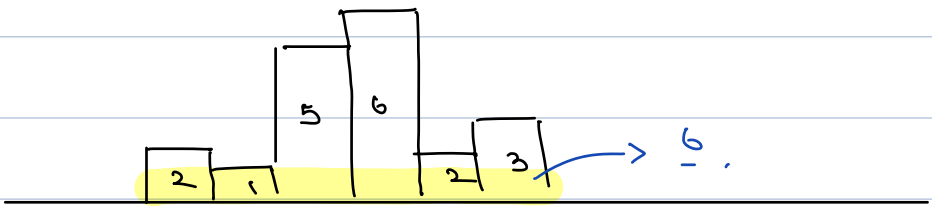
↓

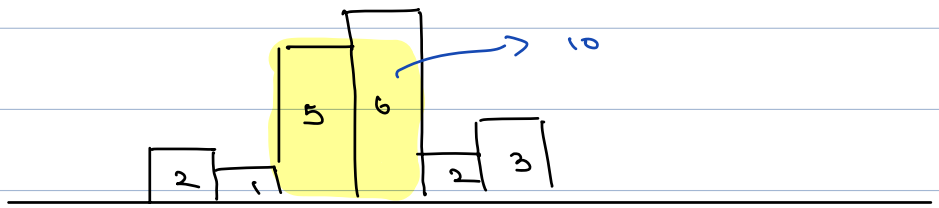
Travel 0 to n

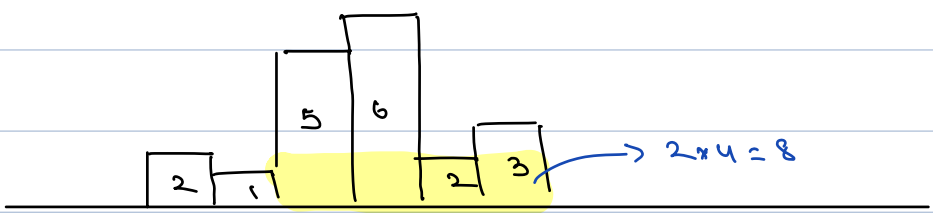
Ques. find nearest greater to left.

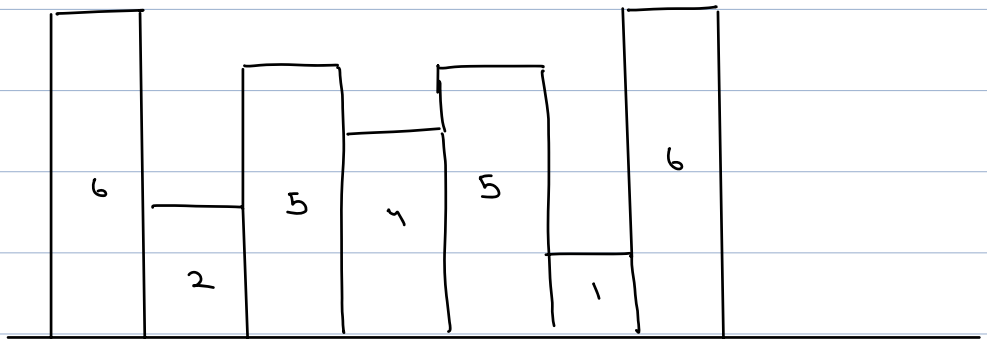
Ques find nearest greater to right.

Ques Largest rectangle area in a histogram.

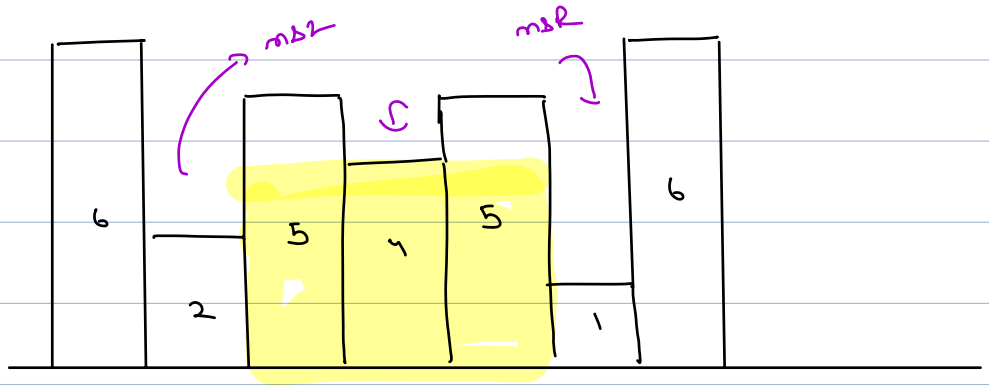


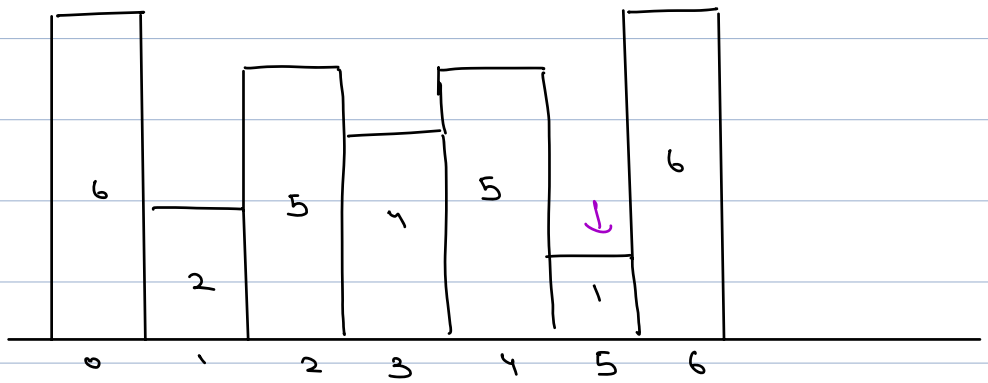






Claim :- height of max area rectangle, must be equal to height of one of the buildings.

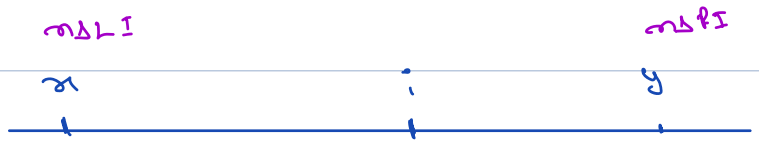




	0	1	2	3	4	5	6
val →	6	2	5	4	5	1	6
nsli →	-1	-1	1	1	3	-1	5
nsri	1	5	3	5	5	7	7

→ (7 - (-1) - 1) * 1 ⇒ 7

→ (5 - 1 - 1) x 4 ⇒ 12



$[x+1, y-1] \rightarrow y-1-x-1+x$

$\Rightarrow (y-x-1)$

width

area = h[i] * (nsri[i] - nsli[i] - 1);

nsl [] —>

nsr [] —>

T.C → O(n)

S.C → O(n)

```
for every height h[i]
    width = (nsr[i] - nsl[i-1])
    height = h[i];
    area = Max(area, width * height);
}
return area;
```

Ques Given an array, find the sum of (max - min) for all subarrays.

s	e	max	min	max - min
0	0	2	2	0
0	1	5	2	3
0	2	5	2	3
1	1	5	5	0
1	2	5	3	2
2	2	3	3	0

Ans → 8

$2(1-3) + 5(4-1) + 3(1-2)$
$\Rightarrow 8$

↓ contribution technique.

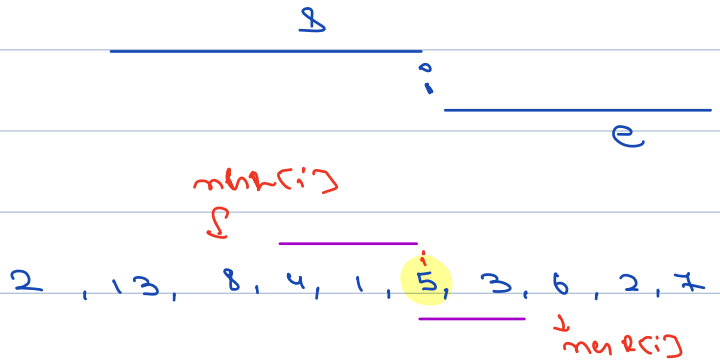
Brute force

1 → check all subarrays

$O(n^3)$ ↻

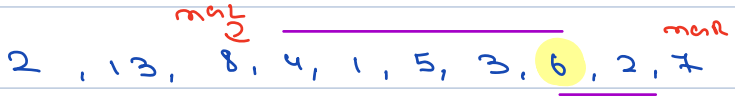
To find :-

In how many subarrays A[i] is max.



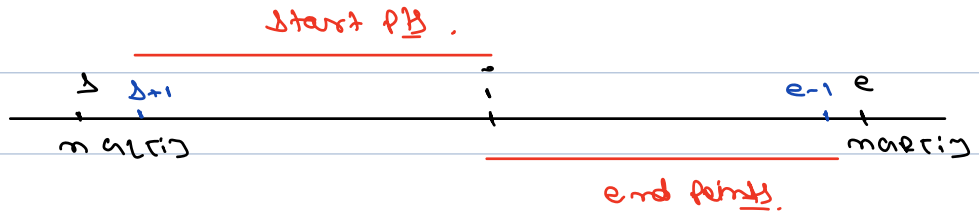
Ques In how many subarrays 5 will be max ?

Ans : $3 * 2 = 6$



Ques In how many subarrays 6 will be max ?

$5 \times 2 = 10$.



(no. of start points) * (no. of end pts)

(s+1 to i) * (i to e-1)

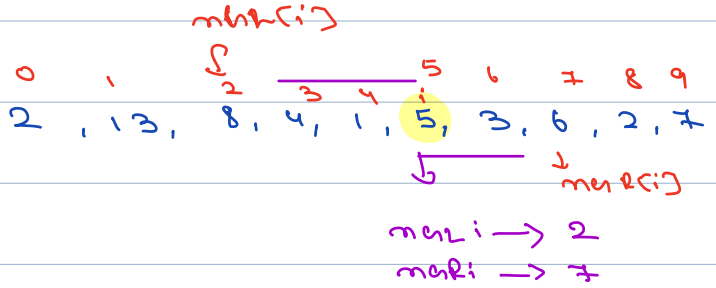
$\Rightarrow (i - s - x + 1) * (e - x - i + 1)$

$\Rightarrow (i - s) * (e - i)$.

$\Rightarrow \rightarrow (i - nsl[i]) * (nsr[i] - i)$

↓

no. of subarrays in which i will be Maxm.



$(i - nsl[i]) * (nsr[i] - i)$

$\rightarrow (5 - 2) * (7 - 5) \Rightarrow 3 \times 2 \Rightarrow 6$

for i → 0 to n-1

No. of max = $(i - nsl[i]) * (nsr[i] - i)$

No. of min = $(i - nsl[i]) * (nsr[i] - i)$

ans += (No. of max − No. of min) * arr[i]

3

return ans;

T.C → O(n)

S.C → O(n)