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.

ex →

	0	1	2	3	4	5	6	7
	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

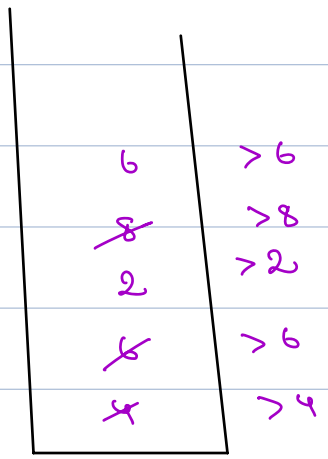
Brute force :- for every i, we will travel from i-1 to 0, when first element smaller than arr[i];

Optimized idea :-

arr[] = [8, *, *, *, *, 5, *, *, *, *] (>5)

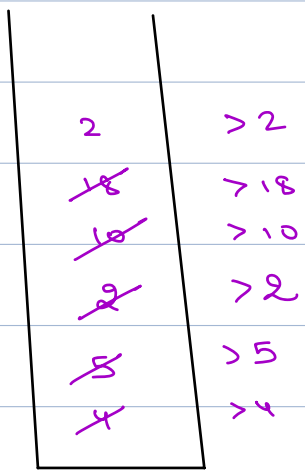
4, 6, 2, 8, 6

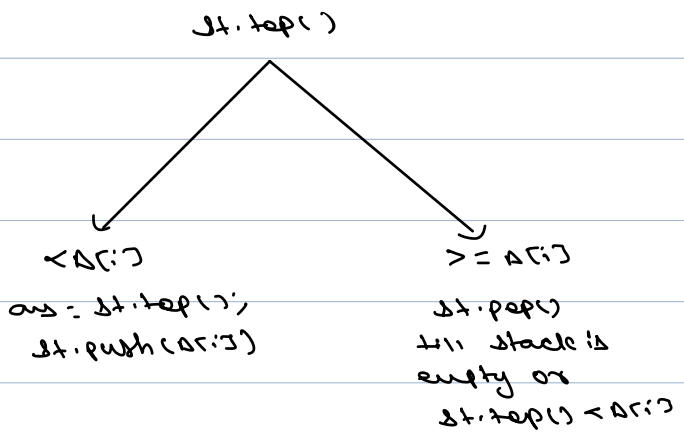
-1 4 -1 2 2

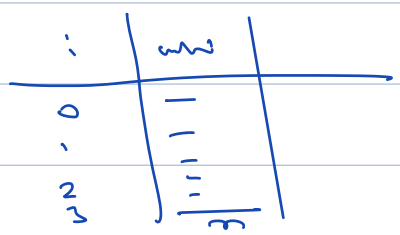


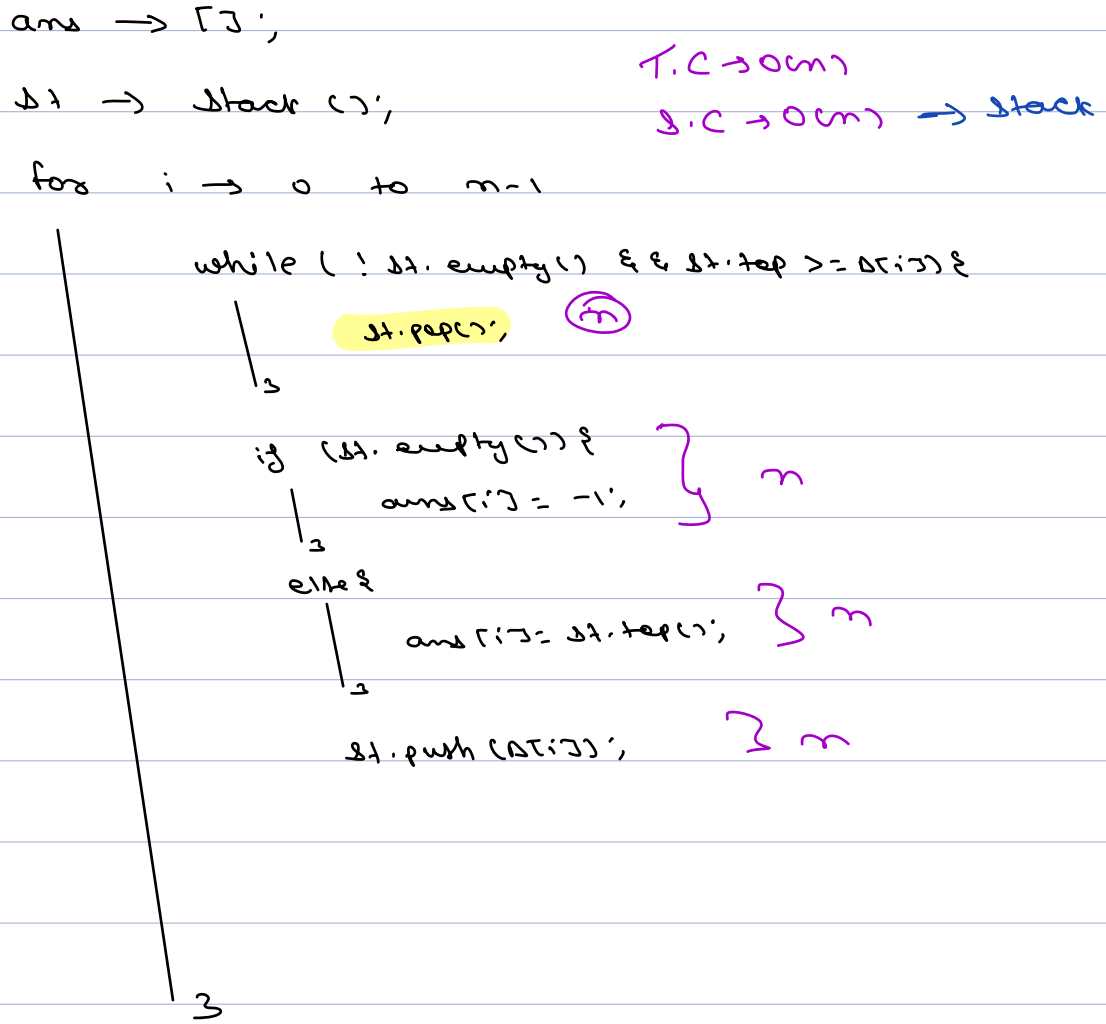
4, 5, 2, 10, 18, 2

-1 4 -1 2 10 -1








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

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

ans → [];

st → stack();

T.C → O(n)

S.C → O(n) → stack

for i → 0 to n-1

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

st.pop(); } n

if (st.empty()) {

ans[i] = -1; } n

else {

ans[i] = st.top(); } n

st.push(i);

Ques Get the dist of nth.

Ques find nearest smaller to right.

↓

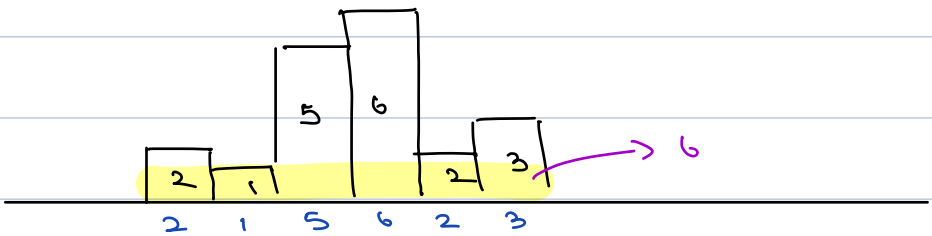
Travel 0 to n

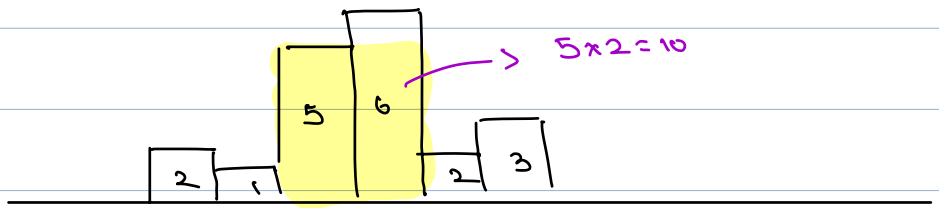
Ques. find nearest greater to left.

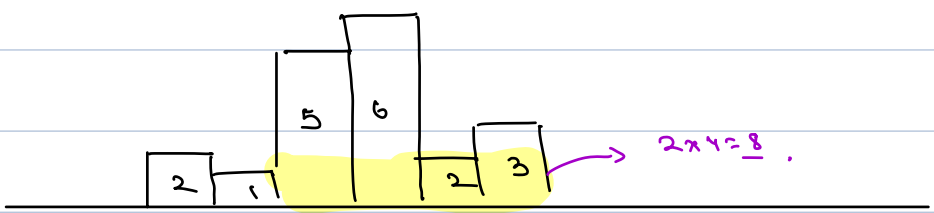
Ques find nearest greater to right.

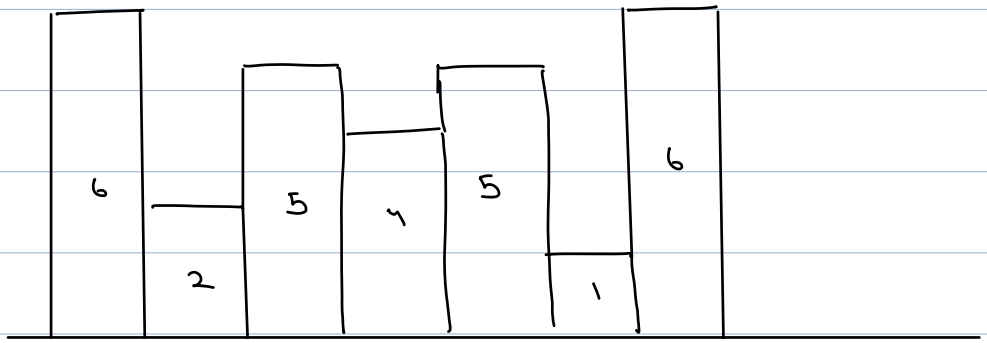
Break 9:55 pm - 10:05 pm

Ques Largest rectangle area in a histogram.

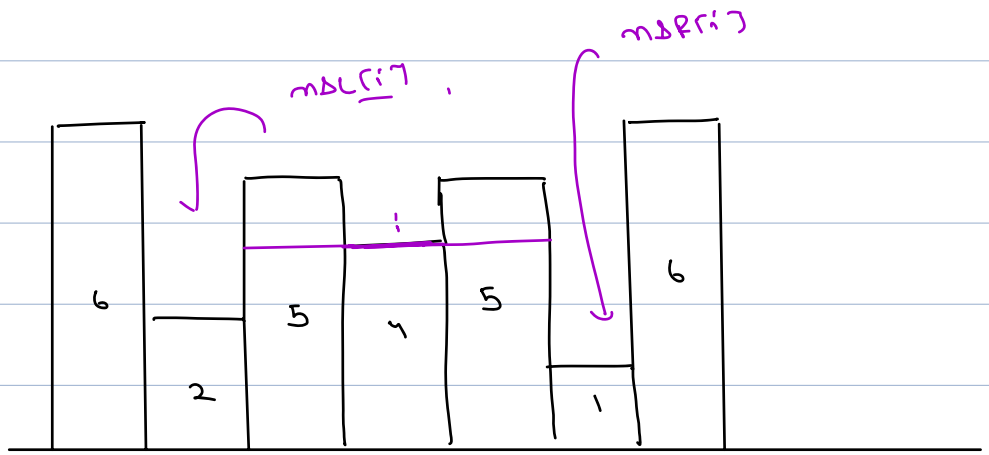




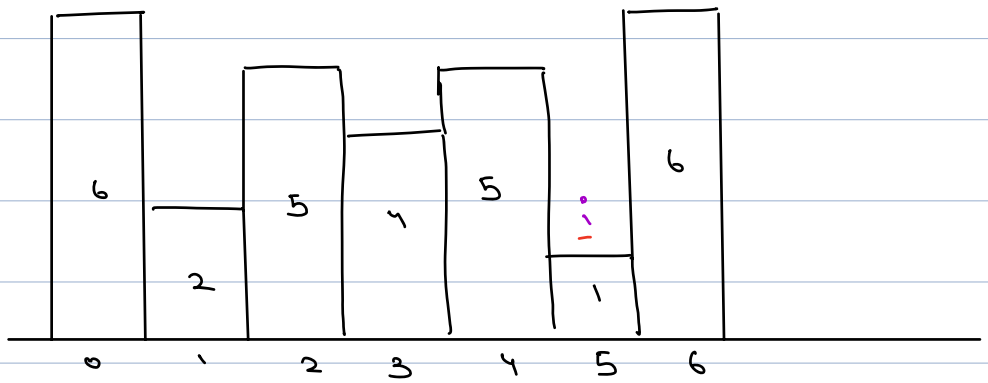




claim :- height of max area rectangle, must be equal to height of one of the buildings.



width = right[i] - left[i] - 1



val →	6	2	5	4	5	1	6
nsl →	-1	-1	1	1	3	-1	5
nsr →	1	5	3	5	5	7	7

[x+1, y-1] <= [i-k, i+x]

x + y - x - 1 - k

<= y - x - k

right[i] - left[i] - 1

area = h[i] * (right[i] - left[i] - 1)

nsl[] →

nsr[] →

T.C → O(n)
S.C → O(n)

for every height h[i]

area = h[i] * (nsr[i] - nsl[i] - 1)

↳ width.

ans = max(ans, area)

}

return ans;

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
				8

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

Brute force

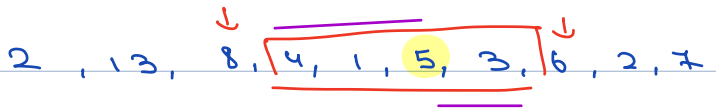
1 → check all subarrays

$O(n^3)$

To find :-

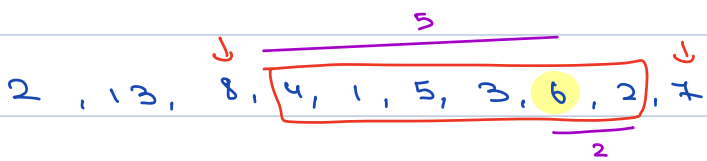
In how many subarrays A[i] is max.

3 * 2 = 6



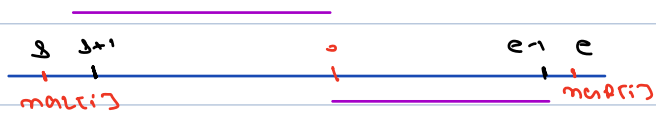
Ques In how many subarrays 5 will be max ?

Ans = 3 * 2 = 6.



Ques In how many subarrays 6 will be max ?

5 * 2 = 10.



start points → [s+1, i]

↓

i - s - 1 + 1

(i - s)

end points [i to e-1]

↓

e - 1 - i + 1

↓

(e - i)

*

no. of subarrays in which i is max

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

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

no. of subarrays in which i is min

$(i - msl[i])(msr[i] - i)$

0	1	2	3	4	5	6	7	8	9
2	13	8	4	1	5	3	6	2	7

$msl \rightarrow 2$

$msr \rightarrow 7$

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

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

// mgl, mgr, msl, msr .

for i → 0 to n-1

max = (i - mgl[i]) (mgr[i] - i)

min = (i - msl[i]) (msr[i] - i)

ans += (max - min) * arr[i]

return ans;

T.C → O(n)

S.C → O(n)