Q→ Given an integer array, find the index of nearest smaller element on left i.e.
$\forall i$, find index j s.t. $A[j] < A[i]$, $j < i$ & j is maximum.

	0	1	2	3	4	5	6	7
A =	8	2	4	9	7	5	3	10
	-1	-1	1	2	2	2	1	6 (Ans)

	0	1	2	3	4	5	6	7
A =	4	6	10	11	7	8	3	5
	-1	0	1	2	1	4	-1	6 (Ans)

	0	1	2	3	4	5
A =	4	5	2	10	8	2
index →	-1	0	-1	2	2	-1
values →	-1	4	-1	2	2	-1

Bruteforce → $\forall i$, iterate j from $(i-1)$ to 0, first $A[j] < A[i] \Rightarrow j$ is ans.

```
∀i, ans[i] = -1
for i → 1 to (N-1) {
    for j → (i-1) to 0 {
        if (A[j] < A[i]) { ans[i] = j
                           break }
    }
} return ans
```

TC = $O(N^2)$ SC = $O(1)$

	0	1	2	3	4	5	6	7	
A = [	8	_	_	_	_	5	_	_	. . .]

For any index > 5 , can 8 (index 0) be the ans?

$i > 5$, if $A[0] < A[i]$

& $A[5] < A[0] \Rightarrow A[5] < A[i]$

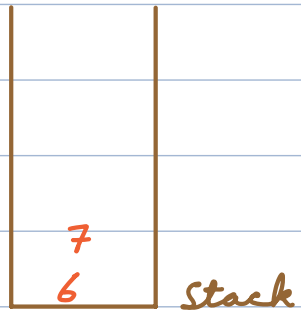
$\because i > 5 \Rightarrow$ index 5 will be closer wrt index 0

$\Rightarrow A[5]$ is always a better answer.

	0	1	2	3	4	5	6	7
A = [	8	2	4	9	7	5	3	10]
	-1	-1	1	2	2	2	1	6

check latest element first → use Stack

	~~i~~ 0	~~i~~ 1	~~i~~ 2	~~i~~ 3	~~i~~ 4	~~i~~ 5	~~i~~ 6	~~i~~ 7	i
A = [	4	6	10	11	7	8	3	5]	
ans →	-1	0	1	2	1	4	-1	6	




```
for i → 0 to (N-1) {
    while (!st.isEmpty() && A[st.peak()] >= A[i]) {
        st.pop()
    }
    if (st.isEmpty())   ans[i] = -1
    else                ans[i] = st.peak()
    st.push(i)
} return ans        TC = O(N)   SC = O(N)
```

Q2 → Find nearest smaller or equal on left.

Q3 → Find nearest greater element on left.

Q4 → Find nearest greater or equal on left.

Q5 → Find nearest smaller element on right.

```
for i → N-1 to 0 {
    while (! st.isEmpty () && A[st.peek ()] >= A[i]) {
        st.pop ()
    }
    if (st.isEmpty ())   ans[i] = -1
    else                 ans[i] = st.peek ()
    st.push (i)
}
```

TC = O(N) SC = O(N)

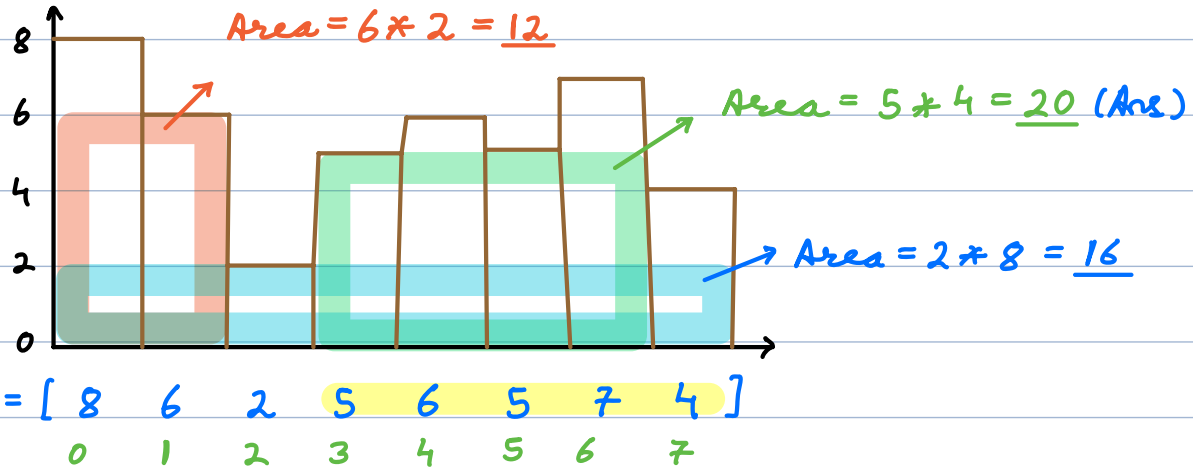
> <= <

Q6 → Find nearest smaller or equal on right.

Q7 → Find nearest greater element on right.

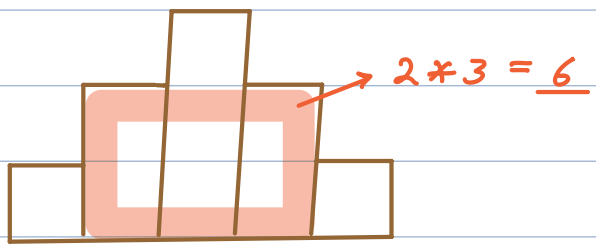
Q8 → Find nearest greater or equal on right.

Q9 → Given an integer array A[],
A[i] = height of i^{th} bar in histogram.
Width of each bar is 1.
Find the area of largest rectangle formed by continuous bars.



Area of bars (l — r) = $\forall_{i \to l \text{ to } r}$ min (A[i]) * (r - l + 1)

A = [1 2 3 2 1]
0 1 3 3 4



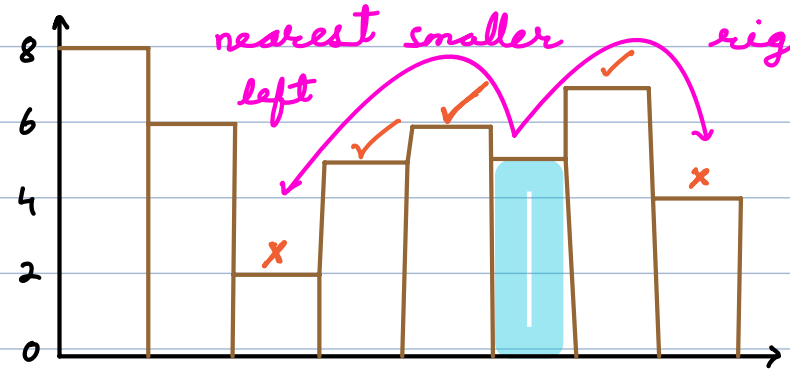
Bruteforce → ∀ subarray of bars calculate area & take maximum.

TC = O(N² * N) → O(N³) SC = O(1)

∀ subarrays → find height

base → height

Sol → ∀ A[i] as height, find max base.



(j k)

A = [8 6 2 5 6 5 7 4]

0 1 2 3 4 5 6 7

j i k

[(j+1) (k-1)] → k-j-1

Area = A[i] * (k-j-1)

```
ans = 0
for i → 0 to (N-1) {
    j = nearestSmallerLeft[i]  // if not present → -1
    k = nearestSmallerRight[i]  // if not present → N
    ans = max(ans, A[i] * (k-j-1))
}
return ans
```

TC = O(N+N+N) → O(N) SC = O(N+N+N) → O(N)

Q → Given an integer array with distinct values, for all subarrays find (max - min) & return its sum as the answer.

A = [2 5 3] (0 1 2)

	(max - min)
2	2 - 2 = 0
2 5	5 - 2 = 3
2 5 3	5 - 2 = 3
5	5 - 5 = 0

Bruteforce →

∀ subarray,
find max & min.
Calculate (max - min) &
take its sum as answer.

TC = $O(N^3)$ → $O(N^2)$

SC = $O(1)$

			(max - min)
1			1 - 1 = 0
1	2		2 - 1 = 1
1	2	3	3 - 1 = 2
2			2 - 2 = 0
2	3		3 - 2 = 1
3			3 - 3 = 0

4 (Ans)

A = [1 2 3] (0 1 2)

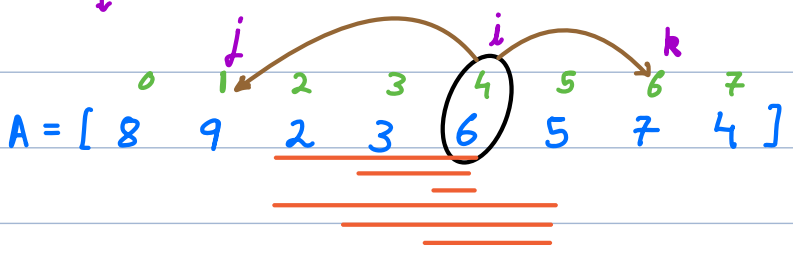
Contribution Technique

Ans = $\sum_{\forall i}$ [contribution of A[i]]

1 * (1 - 3)	-2
2 * (2 - 2)	0
3 * (3 - 1)	6
	4

4 (Ans)

A[i] * (# of subarrays where A[i] is max − # subarrays where A[i] is min)



Find # subarrays where A[4] is max → 6

j = nearest greater on left

k = nearest greater on right

start → [(j+1) i] → i - (j+1) + 1 = i - j

end → [i (k-1)] → k - 1 - i + 1 = k - i

 # subarrays = (i - j) * (k - i)

Find # subarrays where A[i] is min.

j = nearest smaller on left

k = nearest smaller on right

start → [(j+1) i] → i-(j+1)+1 = i-j

end → [i (k-1)] → k-1-i+1 = k-i

 # subarrays = (i-j) * (k-i)

// pre-compute → N+N+N+N+N = 5N

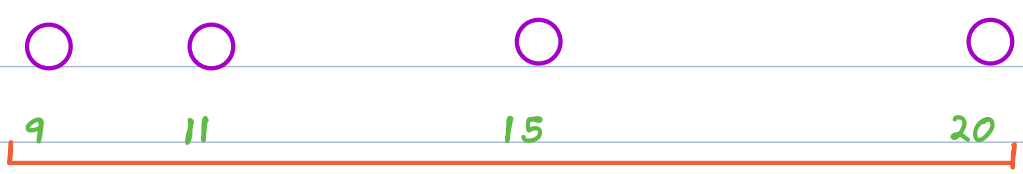
```
ans = 0
for i → 0 to (N-1) {
    gl = nearestGreaterLeft[i]
    gr = nearestGreaterRight[i]
    sl = nearestSmallerLeft[i]
    sr = nearestSmallerRight[i]
    ans += A[i] * ((i-gl) * (gr-i) -
                   (i-sl) * (sr-i))
}
```

} Array

return ans

TC = O(5N) → O(N)

SC = O(5N) → O(N)

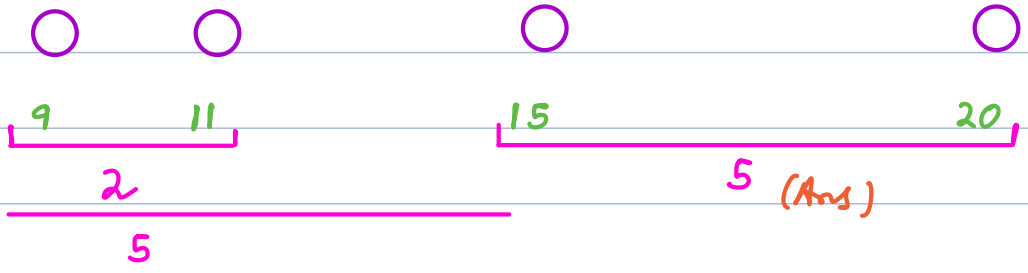


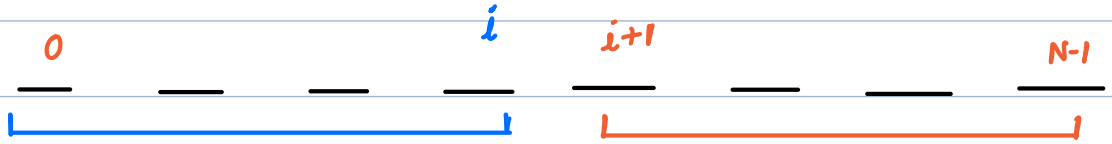
A[K] → position of K^{th} hole

1 board (maxA - minA)

9 + L = 20

⇒ L = 20 - 9 = 11





A[i] - A[0] A[N-1] - A[i+1]

// sort A

ans = Int_Max

for i → 0 to (N-2) {

    L = A[i] - A[0]

    R = A[N-1] - A[i+1]

    ler = max (L, R)

    ans = min (ans, ler)

}

TC = O(N log(N))
