Stacks 2 : Nearest Smaller / Greater Element

Ques1 → Given an integer array, find the index of nearest smaller element of left of index in array A.

If it is not present → mark -1

	0	1	2	3	4	5	6	7
A =	8	2	4	7	7	5	3	10
ans →	-1	-1	1	2	2	2	1	6

Bruteforce → fix i, iterate to left & find nearest smaller element.

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

A = [8 _ _ _ _ 5 _ _ _ _]

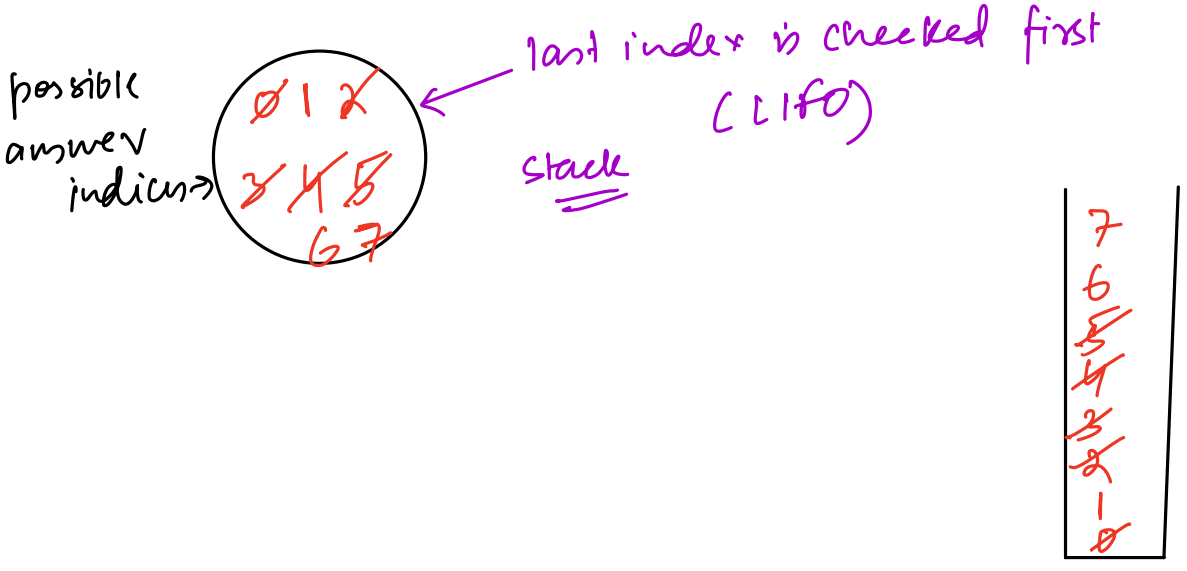
for any element X can index 0 be the answer?

if X > 8 ⇒ X > 5

~~index = 0~~

for any index > 5 closer index is 5 wrt. 0

	0	1	2	3	4	5	6	7
A =	8	2	4	7	7	15	3	10
	-1	-1	1	2	2	4	1	6




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

    st.push(i)
}
return ans
```

TC = O(N)

SC = O(N)

Q2 → ith, find the nearest smaller or equal element on left

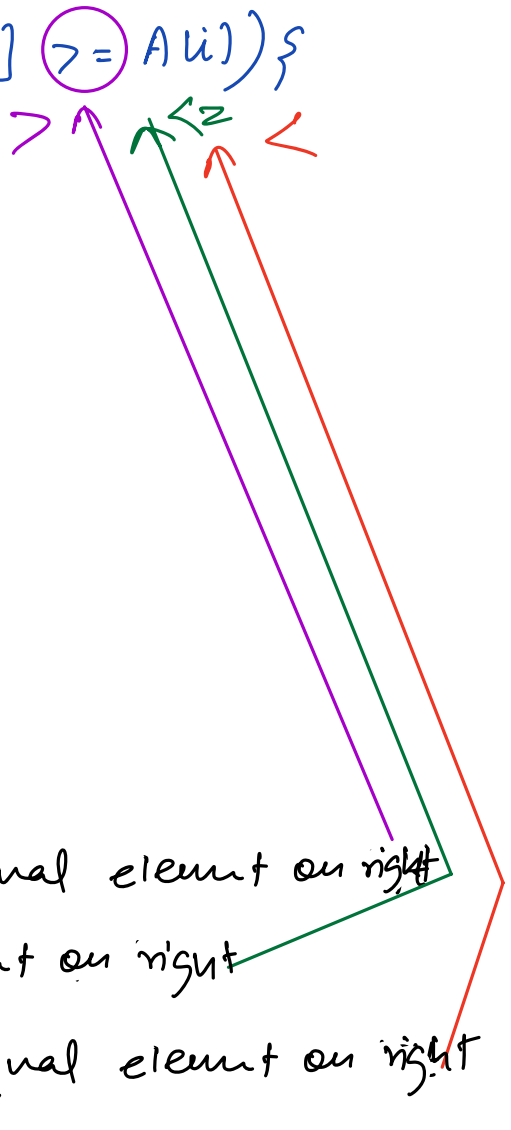
Q3 → ith, find the nearest greater element on left

Q4 → ti, find the nearest greater or equal element on left

Q5 → ti, find the 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)
}
return ans
```



Q6 → ti, find the nearest smaller or equal element on right

Q7 → ti, find the nearest greater element on right

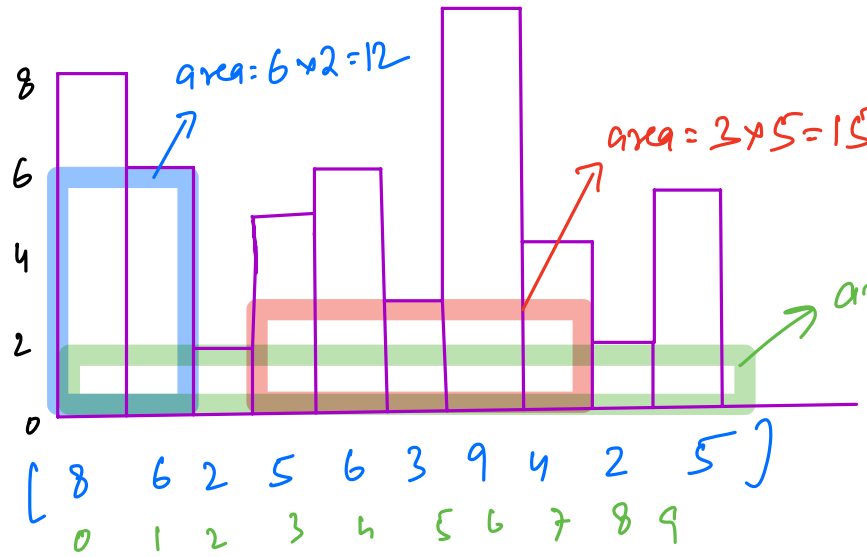
Q8 → ti, find the nearest greater or equal element on right

Ques → Given an integer array A,

$\forall i$, A[i] = height of i^{th} bar

width of each bar = 1

find the area of largest rectangle formed by continous bars.



Height = $\min_{\forall i}(A[i])$

width = # bars

Bruteforce : $\forall$ subarray i,j find H & W & calculate area. Max area is ans.

TC = $O(N^2)$

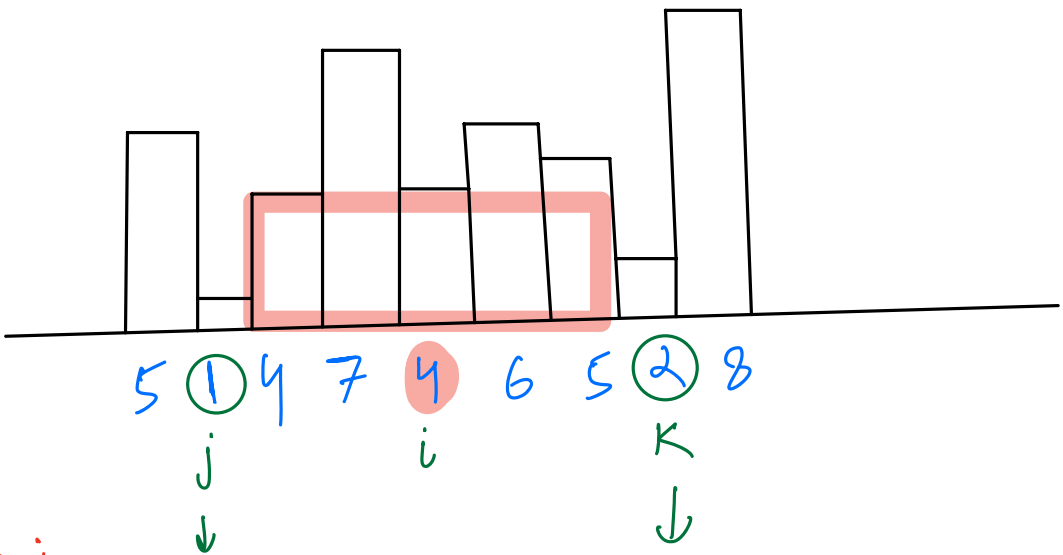
```
ans=0
for (i = 0 to N-1) {
    H = a[i]
    for (j = i to N-1) {
        H = min(H, a[j])
        W = j-i+1
        ans = max(ans, H*W)
```

```
    }
}
return ans
```

# of subarray ⟶ min element

# of width ⟶ height

# of height ⟶ width



steps :

1. nearest smaller element on left

2. nearest smaller elemt is right

H = a[i]

W = K-1-(j+1)+1 = K-j-1

subarray → [j+1, K-1]

j = -1 K = n

W = n-(-1)-1

= n+1-1 = n

3.
```
ans = 0
for (i = 0 to n-1) {
    j = left[i]
    K = right[i]
```

```
        if (k == -1)  k = n
        am = max( am , a[i] * (k-j-1))
    }
    return am
```

TC = O(N + N + N)
- N → creak left[] array
- N → creak right[] array
- N → main logic

$= O(N)$

$SC = O(N)$

Ques Given an integer array with **distinct elements**, find (max - min) **of subarrays** & returns their **sum** as answer.

$A = [2\ 5\ 3]$

			(max - min)	
2		→	2 - 2	= 0
2	5	→	5 - 2	= 3
2	5	3 →	5 - 2	= 3
5		→	5 - 5	= 0
5	3	→	5 - 3	= 2
3		→	3 - 3	= 0
				8

contribution technique

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

$= -4 + 15 - 3$

$= 8$

Brute force:

$TC = O(N^2)$

$SC = O(1)$

```
ans=0
for ( i=0 to n-1) {  [i,....)
    mx = a[i] , mn = a[i]
    for (j=i  to n-1) {  [i,j]
        mx = max(mx, a[j])
        mn = min(mn , a[j])
        ans += (mx - mn)
    }
}
return ans
```

TC = $O(N^2)$

SC = $O(1)$

Contribution technique

$$ans = \sum_{i} \text{contribution of } a[i]$$

$a[i]$ * (# subarrays where $a[i]$ is max − # subarrays where $a[i]$ is min)

A = [2 10 3 5 6 4 7 1 3]

0 1 2 3 4 5 6 7 8

subarrays where a[4] is max? = 6

3 5 6

5 6

6

nearest greater < start <= i } [j+1, i) → i-j
on left (j)

i <= end < nearest greater } [i, K-1] → K-1-i+1
on right (K) → K-i

3 5 6 4
5 6 4
6 4

subarrays = (i-j) * (K-i)

ans = $\sum_i$ contribution of a(i)

a(i) * (# subarrays where a(i) is max − # subarrays where a(i) is min)

(i - greaterleft(i)) * (greaterright(i) - i)

(i - smallerLeft(i)) * (smallerRight(i) - i)

TC = O(N + N + N + N + N)

= O(N)

SC = O(N)