Stacks II

Madhan Kumar M S
Abhishek Sharma
Akansh Nirmal
amit khandelwal
Bhaveshkumar
Burhan
Gagan Kumar S
Gowtham
Murali krishna Talluri
Naval Oli
Pankaj Bhanu
Purusharth A
Rajat Sharma
Rajendra
Sanket Agarwal
Sanket Giri
Saurabh Ruikar
Shani Jaiswal
sharath r
Shrikanth
Subhashini
Subhranil Kundu
Suyash Gupta
Venkata Sribhavana Nandiraju
Vimal Kumar
Vishal Mosa
Yugesh v

→ Nearest smaller element on left.

→ Largest Rectangle

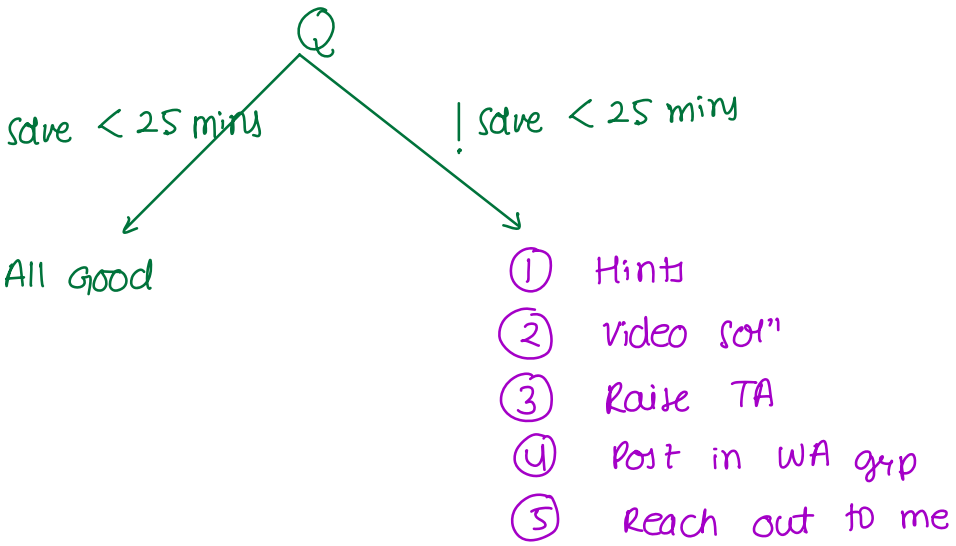
→ Sum of subarrays max min

Current PSP

↓

63 → 70%.

Problem solving Framework



Nearest Smaller Element on Left

* Given an integer array A, find the index of nearest smaller element on left $\forall$ i index in A[]

Formally, $\forall i$ find j such that

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
indices of nse	-1	-1	1	2	2	2	1	6

	0	1	2	3	4	5	6	7
A	4	6	10	11	7	8	3	5
indices of nse	-1	0	1	2	1	4	-1	6

	0	1	2	3	4	5
A	4	5	2	10	8	2
value	-1	4	-1	2	2	-1
indices of nse	-1	0	-1	2	2	-1

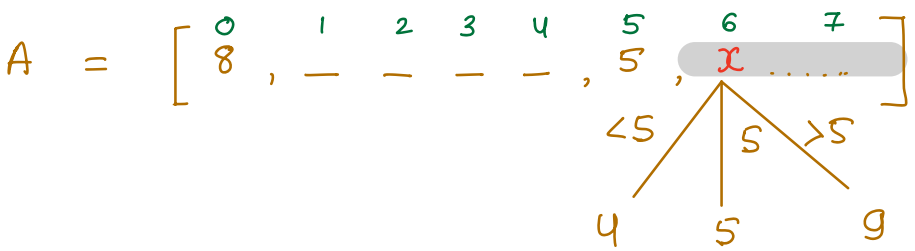
Bruteforce

```
∀i ⟶ 0 to N-1
    nse = -1
    ∀j ⟶ i-1 to 0   LIFO
        if Ai > Aj
            nse = j     break.
    print(nse)
```

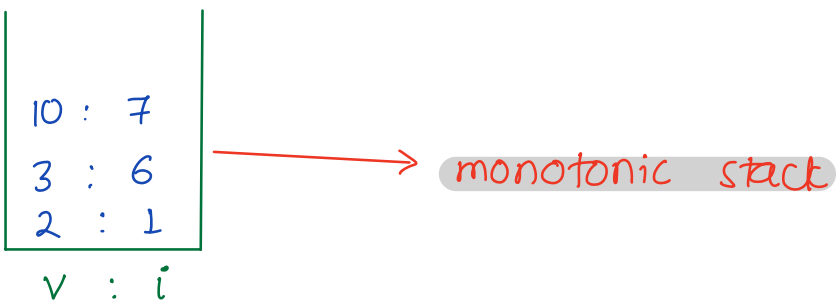
TC: $O(N^2)$

SC: $O(1)$

Observation

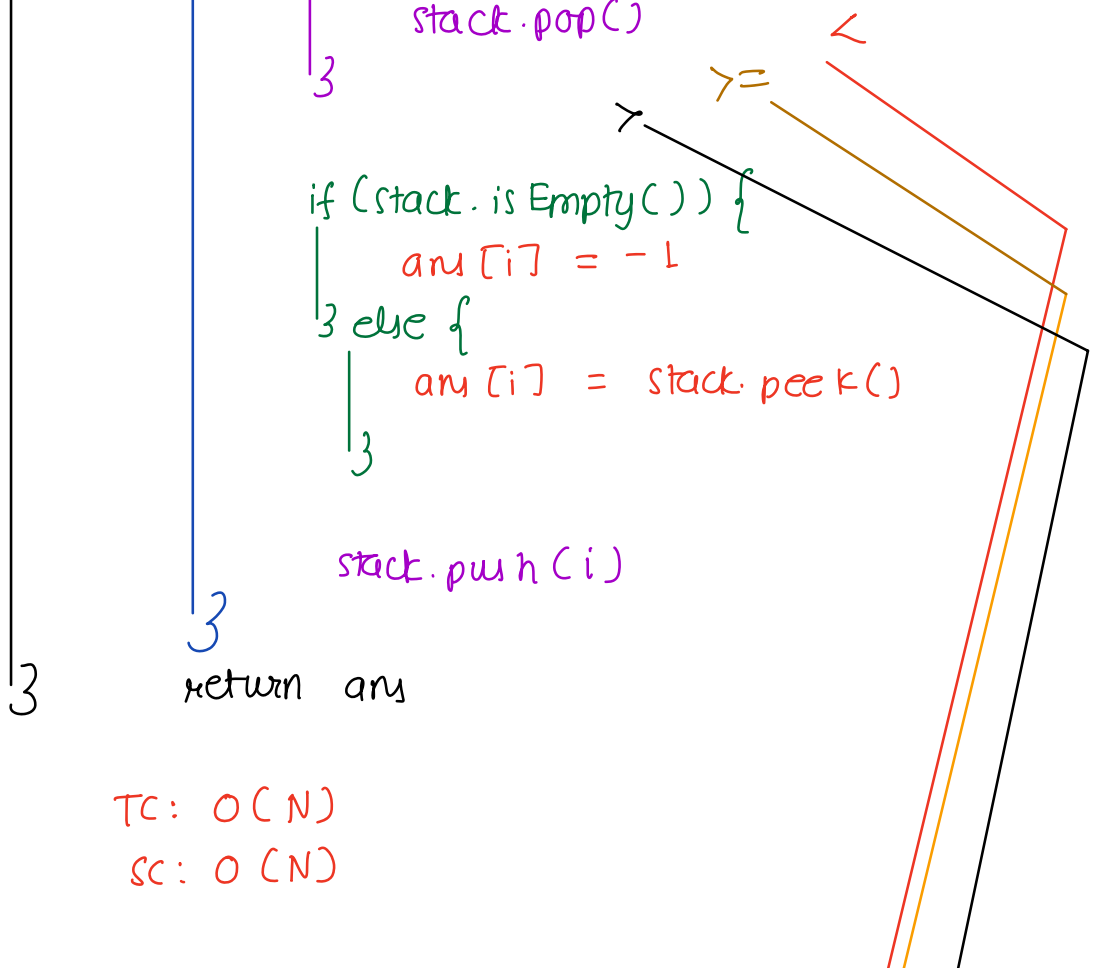


0	1	2	3	4	5	6	7
8	2	4	9	7	5	3	10
↑	↑	↑	↑	↑	↑	↑	↑
-1	-1	1	2	2	2	1	6



Pseudo code

```
int[]  nearestSmallerOnLeft ( int[] A ) {
        stack = []
        ans  =  [0.....0]  // n size
        for ( i=0 ; i<n ; i++ ) {
            while ( !stack.isEmpty()  &&
                    A[i]  <=  A[stack.peek()] ) {
                stack.pop()
            }

            if (stack.isEmpty()) {
                ans[i] = -1
            } else {
                ans[i]  =  stack.peek()
            }

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



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

Q> ∀ i , find nearest smaller or equal element on left
Q> ∀ i , find nearest greater element on left
Q> ∀ i , find nearest greater or eq element on left

Q> ∀ i , find nearest smaller element on right.

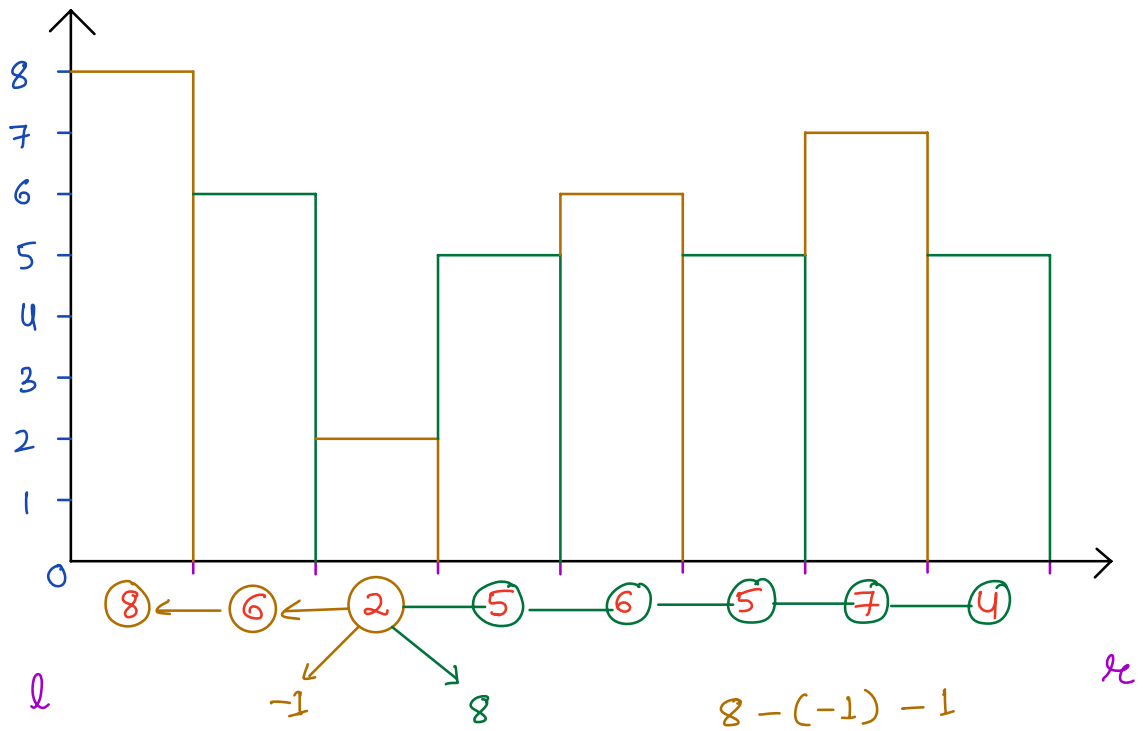
```
int[] nearestSmallerOnRight( int[] A ) {
    stack = []
    ans = [0.....0] // n size
    for ( i = n-1; i >= 0; i-- ) {
        while ( !stack.isEmpty() &&
                A[i] <= A[stack.peek()] ) {
            stack.pop()
        }

        if (stack.isEmpty()) {
            ans[i] = -1
        } else {
            ans[i] = stack.peek()
        }

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

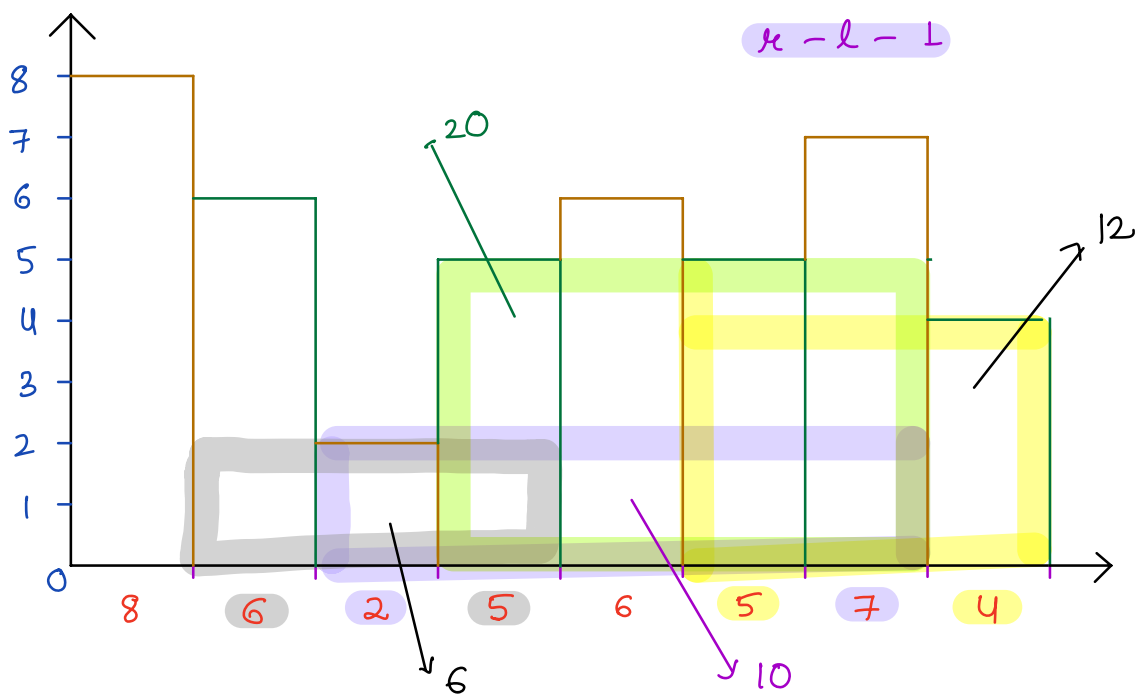
∀i nearest smaller or equal element on right ⟶ <

∀i nearest greater or equal element on right ⟶ >

∀i nearest greater or equal element on right ⟶ >=

Q> Given A[], where A[i] = height of i^{th} bar
width of each bar = 1
find the area of largest rectangle formed by continuous bars.





Given i , j area = min (A_i .... A_j) * (j - i + 1)

st end

A = 1 2 3 2 1

minH = 2 2*3 = 6

width = 3

Brute force cc

TC : O(N^3)

SC : O(1)

maxarea = 0

```
for l ⟶ 0 to N-1 {
    for r ⟶ l to N-1 {
        minH = ∞
        for i ⟶ l to r {
            minH = min (minH, A[i])
        }
        maxarea = max (maxarea, minH * (r-l+1))
    }
}
print (maxarea)
```

Carry Forward Technique

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

```
for l ⟶ 0 to N-1 {
    minH = ∞
    for r ⟶ l to N-1 {
        minH = min(minH, A[r])
        marea = max(marea, minH * (r-l+1))
    }
}
print(marea)
```

Break: 22:48

∀ width ⟶ min Height

TC: $O(N^2)$

∀ height ⟶ calculate width

Pseudocode

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

```
int largestRectangle(int[] A) {
    nseL // calculate
    nseR // calculate
    marea = 0
    for i ⟶ 0 to N-1 {
        l = nseL[i]
        r = nseR[i] // if r == -1 make r = N
        w = r - l - 1
        marea = max(marea, A[i] * w)
    }
    return marea
}
```

Q> Given a A[] with **distinct integers** ∀ subarrays, find (max - min) & return its sum as the answer

A = 2 5 3

	max	min	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

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

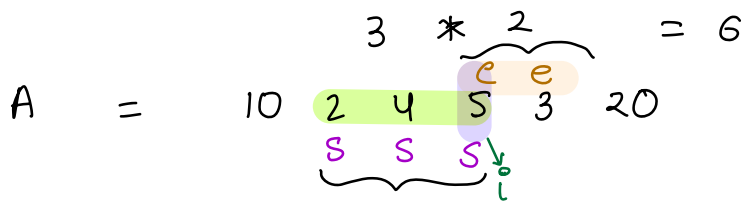
$15 + (-4) - 3 =$

A = 1 2 3

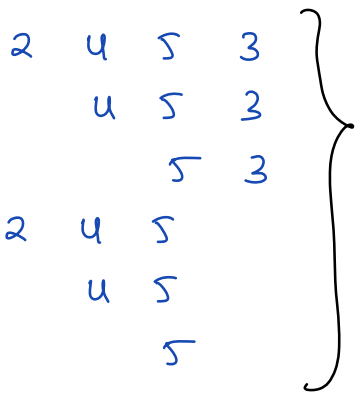
	max	min	max - min
1	1	1	0
1 2	2	1	1
1 2 3	3	1	2
2	2	2	0
2 3	3	2	1
3	3	3	0
			4

Bruteforce ∀ subarray add max-min to any

TC: $O(N^3)$



In how many subarrays is 5 my max element?



5 is max element in 6 subarrays.

count of subarrays where 5 is max =

* i − ngeL[i] // possibilities of start
ngeR[i] − i // possibilities of end

	0	1	2	3	4	5
	10	2	4	5	3	20
ngeL	-1	0	0	0	3	-1
ngeR	5	2	3	5	5	-1 → 6

Replace with N

start = 3 − 0 = 3 *
end = 5 − 3 = 2

= 6

count of subarrays where S is min =

i − nseL[i] // possibilities of start

* nseR[i] − i // possibilities of end

```
int sumMinMax ( A[] ) {

    nseL , nseR , ngeL , ngeR // create
    // also change -1 to N for right.
    ans = 0
    for i ⟶ 0 to N-1 {
        // max
        maxSubs = (i − ngeL[i]) * (ngeR[i] − i)
        // min
        minSubs = (i − nseL[i]) * (nseR[i] − i)

        contribution = A[i] * (maxSubs − minSubs)
        ans += contribution
    }
    return ans
}
```

TC : O(N)

SC : O(N)