Stacks 2: Nearest Smaller / Greatest Element

Question 1:- Given an integer array A, find the index of nearest smaller on left for all 'i' index in A[].
Formally, for all 'i' find 'j' such that

$A[j] < A[i]$, $j < i$ & j is maximum

Example:-

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

Arr = [8 , 2 , 4 , 9 , 7 , 5 , 3 , 10]

Element	Nearest Prev ~~Next~~ Smaller Element	Prev Index of ~~Next~~ Smaller element
8	-1	-1
2	-1	-1
4	2	1
9	4	2
7	4	2
5	4	2
3	2	1
10	3	6

Quiz 1 :- Nearest Smaller element on left [NSL]

A = [4, 6, 10, 11, 7, 8, 3, 5]

↳ NSL = [-1 4 6 10 6 7 -1 3]

Quiz 2 :- Nearest Smaller element on Left [NSL]

A = [4, 5, 2, 10, 8, 2]

↳ NSL = [-1 4 -1 2 2 -1]

Brute force :- For every i, we will travel from (i-1) to 0, return first smaller ~~than~~ then arr[i];

TC → O(N²)

↳ OPTIMIZATION

Quiz 3 :- If A = [8, x, x, x, x, 5, x, x, x, x ...]

For any element >5 can the element 8 become nearest smaller element on Left ?

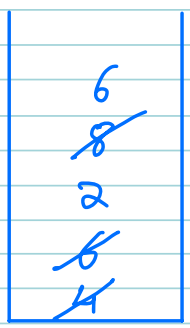
↳ No

⇒ APPROACH

↳ 1st Iteration

Ex:- 4, 6, 2, 8, 6

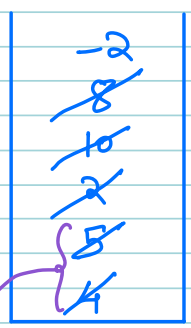
Ans:- -1 4 -1 2 2



Iteration 2

Ex:- 4 5 2 10 8 -2

Ans -1 4 -1 2 2 -1



Q Do I need to store 4 when 5 comes?

↳ yes, for 5 on Right 4 will be the Ans

These can never be Ans when 2 comes.

Conclusion :- Store the element in Stack if it could be potential Ans otherwise just popping all other.

* Observe Stack will always be decreasing

element comes (A[i])

st.top() < A[i]
↳ ans = st.top()
↳ st.push(A[i])

st.top() >= A[i]
↳ st.pop() till top is smaller or st is empty

PSEUDO CODE

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

TC → O(N)
SC → O(N)

Followup :- Instead of element we need to return Index.

Ex :- arr → 4, 5, 2, 10, 18, 2
(indices: 0, 1, 2, 3, 4, 5)

NSL → -1 4 -1 2 10 -1

NSLI → -1 0 -1 2 3 -1
(Index)

PSEUDO CODE

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

VARIATIONS

Q1 Get the distance of Nearest Smaller Left.

↳ If index is in Stack, then Ans = i - St.top()

Q2 Find nearest smallest to Right

↳ Travel R to L

Q3 Find nearest greater to Left

↳ Condition in while loop will be <=

Q4 Find nearest greater to right.

↳ Travel from R to L

10:04 → 10:14

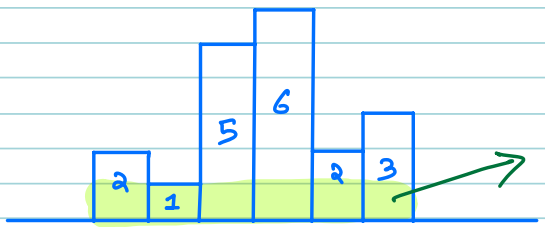
Question 2 :- Given an integer array A, where

→ A[i] = height of ith bar

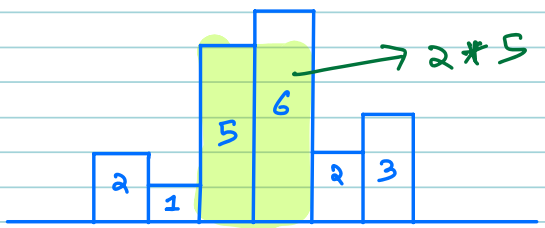
→ width of each bar is 1.

Find the area of the largest rectangle formed by continious bars.

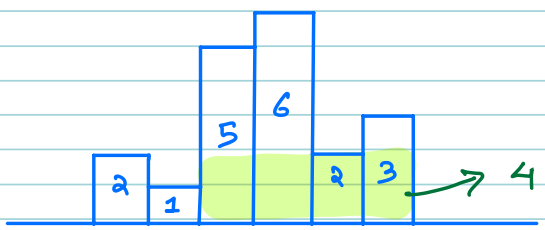
Ex



→ 6 × 1 ⇒ 6.



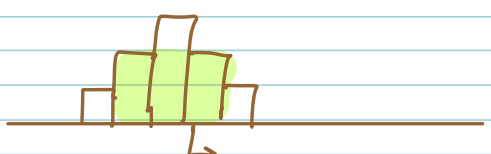
→ 2 * 5 ⇒ 10 Ans



→ 4 * 2 ⇒ 8

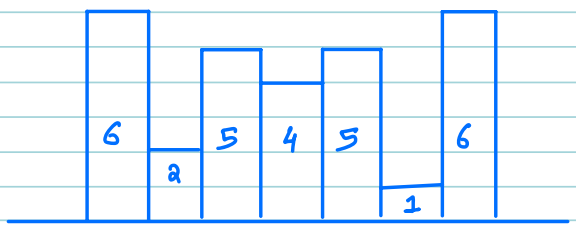
Quiz 4 :- Find the area of the largest rectangle formed by continious bars.

Bars = [1 2 3 2 1]

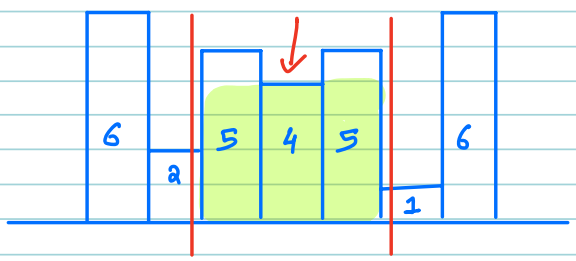


→ 2 × 3 ⇒ 6 Ans

APPROACH



Observation :- The height of rectangle will be any element in the given array.



Q what will be the max Rectangle with height 4.

↳ 2 will Stop the Left Expansion of Rectangle

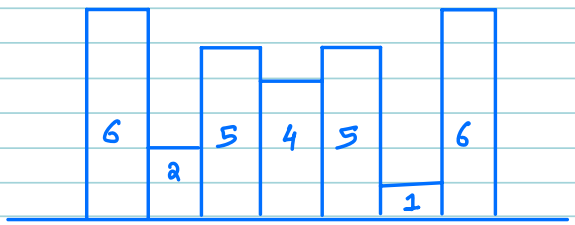
Nearest Smaller Left

↳ 1 will Stop the Right Expansion of Rectangle.

Nearest Smaller Right.

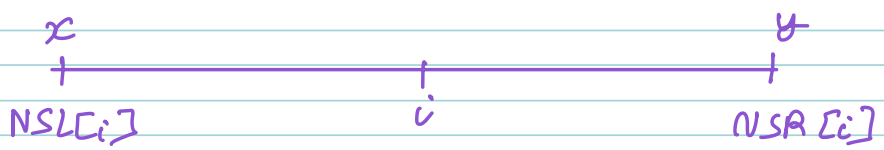
Observation :- We have to just check NSL & NSR of ith index

DRY RUN



Index →	0	1	2	3	4	5	6
NSLI →	-1	-1	1	1	3	-1	5
NSRI →	1	5	3	5	5	7	7

Maths



Window / Rectangle → $[x+1, y-1]$ ⇒ $(y-1)-(x+1)+1$

⇒ $y-1-x-1$

⇒ $y-x-1$

⇒ $NSR[i] - NSL[i] - 1$

Area = $A[i] * [NSR[i] - NSL[i] - 1]$

PSEUDO CODE

```
ans = 0;
NSLI[]
NSRI[]          } TODO
for (i -> 0 to n-1) {
    area = A[i] * [NSRI[i] - NSLI[i]-1];
    ans = Max(ans, area);
}
return ans;
```

TC -> O(N)

SC -> O(N)

Question 3 :- Given an integer array with distinct integers, for all subarray find (max − min) & return its sum.

	0	1	2
Ex :-	2	5	3

→ Total Subarrays :- $\frac{3*4}{2}$

→ 6

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
			Sum →	8

Brute force

↳ check all Subarrays

TC → $O(N^3)$

Quiz 5 :- Given an integer array, A with distinct integers, for all subarrays find (max − min) & return its sum as Ans.

A = [1 2 3]

S	e	Max	Min	Max - Min
0	0	1	1	0
0	1	2	1	1
0	2	3	1	2
1	1	2	2	0
1	2	3	2	1
2	2	3	3	0
				4

↳ OPTIMIZATION

Q Do we Need to Form all Subarrays?

↳ NO

OBSERVATION [Contribution Technique]

↳ See how many times Max & Min is coming.

i.e. $5(4-1) + 2(1-3) + 3(1-2)$

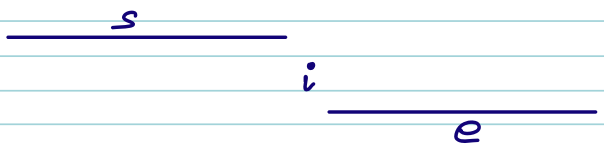
$\Rightarrow 5 * 3 + 2(-2) + 3(-1)$

$\Rightarrow 15 - 4 - 3$

$\Rightarrow 8$

Q To Find :-

In how many subarrays A[i] will be maximum.



Ex :- 2 , 13 , 8 , 4 , 1 , 5 , 3 , 6 , 2 , 7

5 Can't be max beyond this point

5 can't be Max beyond this point

Q :- In how many subarrays 5 will be max ?

↳ How many number will act like start = 3 [4, 1, 5]

↳ How many number will act like End = 2 [5, 3]

Total subarray => 3 x 2 => 6 Ans

Ex :- 2 , 13 , 8 , 4 , 1 , 5 , 3 , 6 , 2 , 7

start

End

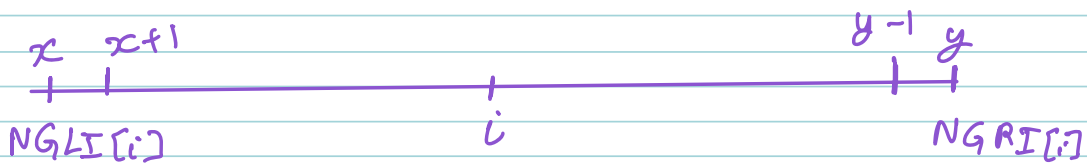
Q In how many subarray 6 will be max ?

=> 5 x 2 = 10

OBSERVATIONS

1. Left Boundary is Nothing just Next greater on Left (NGL)
2. Right Boundary is Nothing just Next greater on Right (NGR)

Maths



Starting point → [x+1 , i]

⇒ i − x − 1 + 1

⇒ i − x

⇒ i − NGLI[i]

End point → [i , y − 1]

⇒ y − 1 − i + 1

⇒ y − i

⇒ NGRI[i] − i

∴ No of subarray in which i is most = (i − NGLI[i]) * (NGRI[i] − i)

Validation of formula

Ex :- 2 , 13 , 8 , 4 , 1 , 5 , 3 , 6 , 2 , 7

i = 5

NGLI = 2

NGRI = 7

$\Rightarrow (i - NGLI[i]) * (NGRI[i] - i)$

$\Rightarrow (5 - 2) * (7 - 5)$

$\Rightarrow 3 * 2 \Rightarrow 6$

∴ No of Subarray in which :- $(i - NSLI[i]) * (NSRI[i] - i)$
i is Min

PSEUDO CODE

```
ans = 0
NGLI[]
NGRI[]
NSLI[]
NSRI[]
                } TODO

for (i -> 0 to N-1) {
    max = (i - NGLI[i])(NGRI[i] - i);
    min = (i - NSLI[i])(NSRI[i] - i);

    ans += A[i]*(max - min);
}

return ans;
```

TC → O(N)

SC → O(N)