Stacks 2

Good Evening

Today's content

01. Nearest smaller on left/right
02. Nearest greater on left/right
03. Largest Area histogram
04. Sum of (max-min) in all subarrays

01. Given an integer A[]. Find the nearest smaller ele index on left.

	0	1	2	3	4	5
A[6] = {	4	5	2	10	3	12 }
Ans[6] = {	-1	0	-1	2	2	4 }

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

	0	1	2	3	4	5
A[] = {	4	5	2	10	8	2 }
Ans[] = {	-1	0	-1	2	2	-1 }

Brute force → For an element at 'i' idx, iterate from i-1 to 0 & look for the nearest smaller element.

```
int[] ans = new int[n];
for (i=0; i<n; i++) {
    ans[i] = -1;
    for (j=i-1; j≥0; j--) {
        if (A[j] < A[i]) {
            ans[i] = j
            break;
        }
    }
}
return ans;
```

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

Observations

$A[] = \{ \underset{0}{8}, \_, \_, \_, \underset{4}{5}, \underset{5}{x}, \_, \_ \}$

Can idx 0' ele (8) ever be the answer for x?

01. $x \leq 5 \Rightarrow$ ans $\neq$ 0' idx
02. $5 < x \leq 8 \Rightarrow$ ans = 4' idx (ans $\neq$ 0' idx)
03. $x > 8 \Rightarrow$ ans = 4' idx (ans $\neq$ 0' idx)

Conclusion → If a smaller no. is present on RHS of a larger no. then this larger no. can never contribute in the answer

	0	1	2	3	4	5	6	7	8	9
A[] =	8	2	4	9	7	5	3	10	10	11
Ans[] =	-1	-1	1	2	2	2	1	6	6	8

```
Stack<I> st = new Stack<>();

int[] ans = new int[n]

for (i=0; i<n; i++){

    while (st.size()>0 && A[i] ≤ A[st.peek()]){

        st.pop();

    }

    if (st.size()==0) ans[i] = -1;

    else ans[i] = st.peek();

    st.push(i);

}
```

Tc : O(n)

Sc : O(n)

∀ i , find nearest smaller or equal ele idx on left.

$$A[i] < A[st.peek()]$$

∀ i, find nearest greater ele idx on left

$$A[i] \geq A[st.peek()]$$

Updating popping condition in above code to get the answer for these variations.

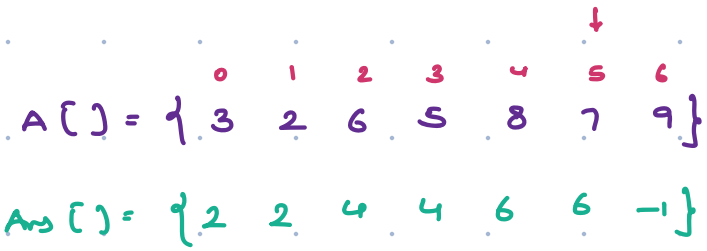
A person uses **Google Maps** to find the nearest restaurants and picks one based on its proximity. Unfortunately, after visiting, they realized that the restaurant didn't meet their expectations.

Task

Let's break it down with a simple example. You have a list of restaurants and their **ratings**. For each restaurant, we're going to find the next restaurant to the **right** on the list that's not just close but also has a **higher rating** than the **current** one. If there's no better option on the list, we'll say there's **none** available.

Problem

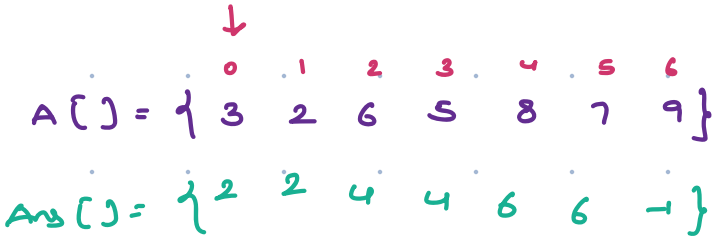
Given a sequence of restaurants listed on Google Maps with their ratings, create a tool that helps users discover the rating of the next higher-rated restaurant to the right for each listed establishment.

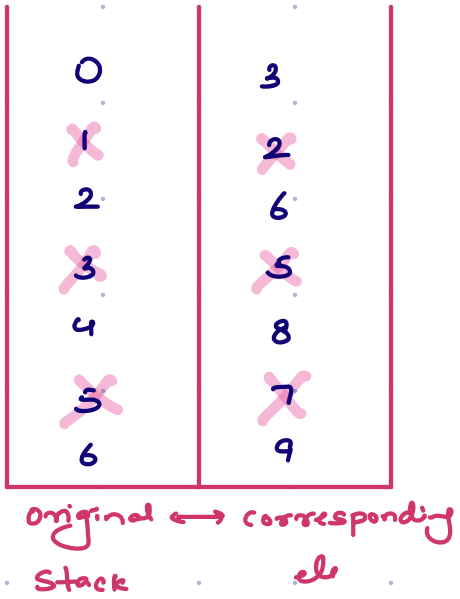


* Next greater element on RHS

```
Stack <I> st = new Stack <>();

int [] ans = new int [n]

for (i = n-1 ; i ≥ 0; i--)
{
    while (st.size() > 0 && A[i] ≥ A[st.peek()]) {
        st.pop();
    }

    if (st.size() == 0) ans[i] = -1;

    else ans[i] = st.peek();

    st.push(i);
}
```

Tc : O(n)
Sc : O(n)





A[i] < A[st.peek()]

not remove

curr ele ≥ A[st.peek()]

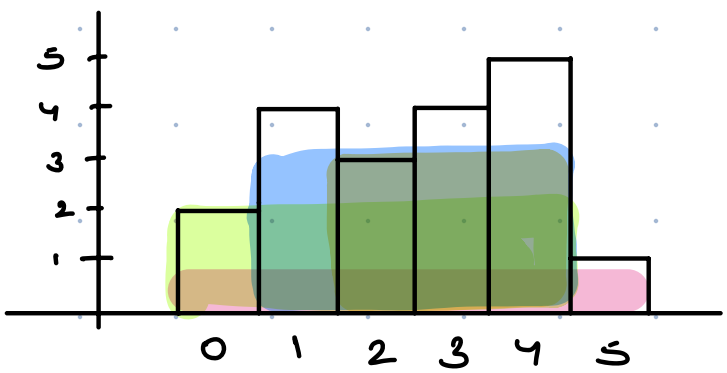
remove

Histogram Area

Given continous block of Histogram find max Rectangular area which can be present within histogram

Note :- Every histogram is of width = 1

Ex : ar[6] = { 2 4 3 4 5 1 }
0 1 2 3 4 5



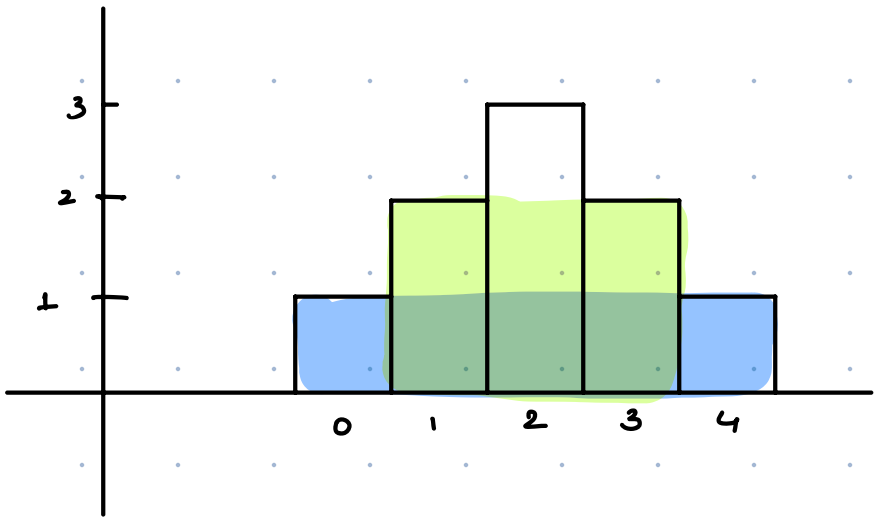
Area = 6

Area = 9

Area = 10

Area = 12 → find ans

A[] = { 1 2 3 2 1 }



Area = 5

Area = 2 * 3

Ans

Brute force → Consider all the possible combination of starting point & ending point of histogram

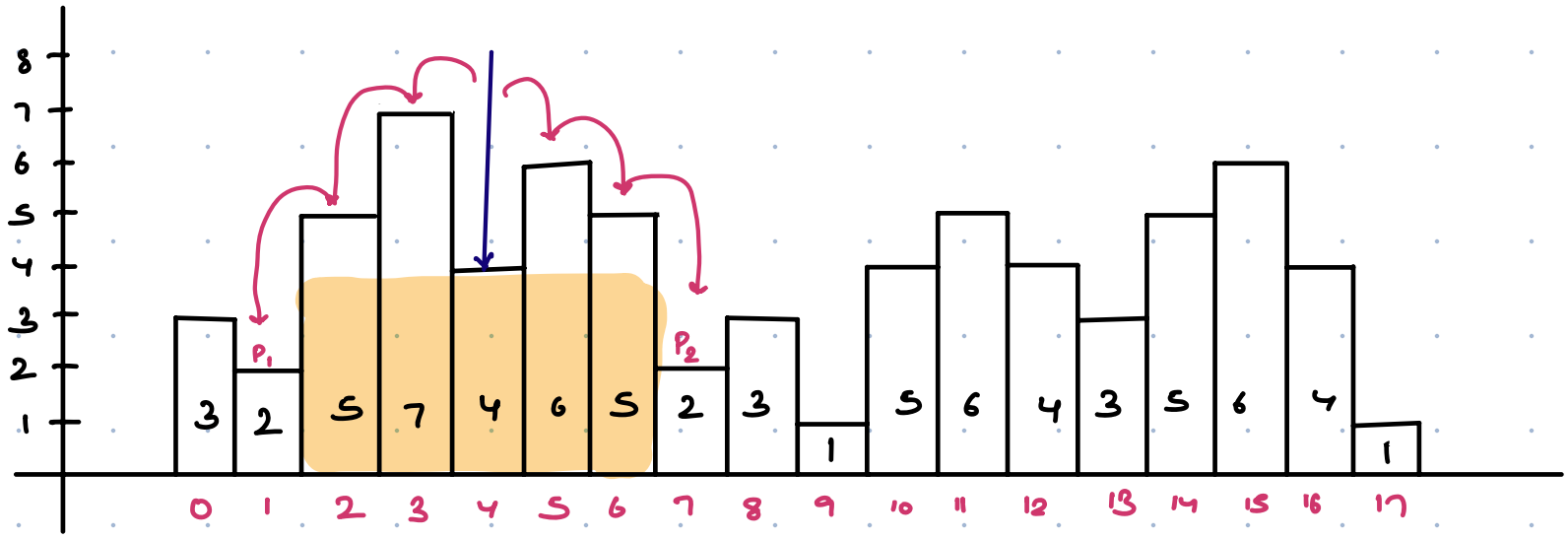
```
ans = 0
for ( i = 0 ; i < n ; i++ ) {
    minht = A[i];
    for ( j = i ; j < n ; j++ ) {
        minht = min ( minht, A[j] )
        width = j - i + 1
        Area = minht * width;
        ans = max (ans, Area);
    }
}
```

Tc : O(n²)

Sc : O(1)

ar[ ] =

3	2	5	7	4	6	5	2	3	1	5	6	4	3	5	6	4	1



Idea → Consider each histogram as height of my rectangle & figure out how much to expand on

LHS & RHS

Nearest smaller ele idx on left

P_1

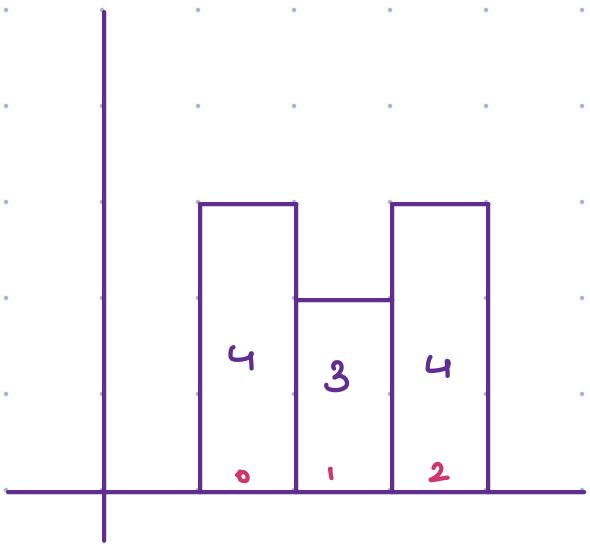
Nearest smaller ele idx on right.

P_2

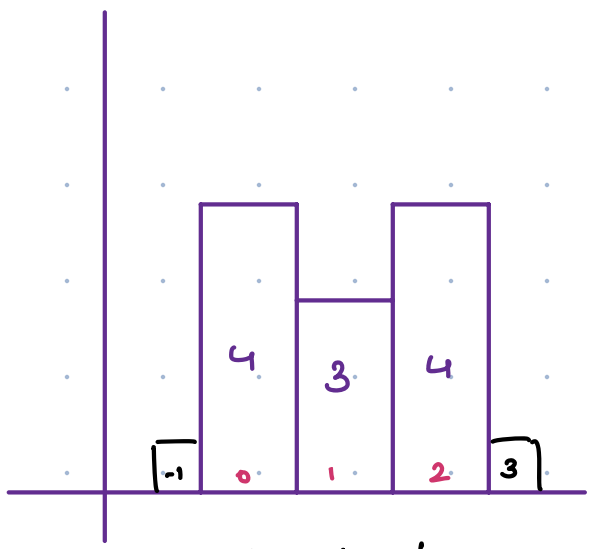
width = $P_2 - P_1 - 1$

```
int rectangularArea ( int [] ht)
{
    int [] left = nearest smaller idx on left (ht);  // default val = -1
    int [] right = nearest smaller idx on right (ht); // default val = n
                                                      & not -1
    int area = 0, ans = 0
    for (i=0; i<n; i++){
        int h = ht[i];
        P1 = left[i]
        P2 = right[i]
        area = h * (P2 - P1 - 1)
        ans = Math.max(ans, area)
    }
    return ans;
}
```

TC : O(n)
SC : O(n)



P_1 =	-1	-1	1
P_2 =	1	-1	-1
width =	1	-1	-3



P_1 =	-1	-1	1
P_2 =	1	3	3
width =	1	3	1

Q3 Given an integer array with distinct elements, for all the subarrays, find (max-min) & return its sum as the answer

$A[] = \{2, 5, 3\}$ (indices 0, 1, 2)

subarr		Max-min
[0 0]	= {2}	2-2 = 0
[0 1]	= {2,5}	5-2 = 3
[0 2]	= {2,5,3}	5-2 = 3
[1 1]	= {5}	5-5 = 0
[1 2]	= {5,3}	5-3 = 2
[2 2]	= {3}	3-3 = 0

Ans = 8

$A[] = \{1, 2, 3\}$ (indices 0, 1, 2)

subarr		max-min
[0 0]	= {1}	1-1 = 0
[0 1]	= {1,2}	2-1 = 1
[0 2]	= {1,2,3}	3-1 = 2
[1 1]	= {2}	2-2 = 0
[1 2]	= {2,3}	3-2 = 1
[2 2]	= {3}	3-3 = 0

Ans = 4

BF → Calculate max & min for all subarrays & find their difference

Add it to global answer

```
for (i=0; i<n; i++)
    for (j=i; j<n; j++) {
        max = -∞
        min = +∞
        for (int k=i; k≤j; k++) {
            max = Math.max (max, A[k]);
            min = Math.min (min, A[k]);
        }
        int diff = max - min;
        ans += diff;
    }
```

TC: $O(n^3)$
SC: $O(1)$

* Contribution Technique

$$Ans = \sum_{i=0}^{n-1} \text{contribution of } A[i].$$

$$\text{contribution of } A[i] = A[i] * \left(\begin{array}{l} \#\text{ no. of subarrays} \\ \text{where } A[i] \text{ is} \\ \text{maximum} \end{array} - \begin{array}{l} \#\text{ of subarrays} \\ \text{where } A[i] \text{ is} \\ \text{minimum} \end{array} \right)$$

A[] = {2, 5, 3}

subarr		Max-min
[0 0]	= {2}	2-2 = 0
[0 1]	= {2,5}	5-2 = 3
[0 2]	= {2,5,3}	5-2 = 3
[1 1]	= {5}	5-5 = 0
[1 2]	= {5,3}	5-3 = 2
[2 2]	= {3}	3-3 = 0

Ans = 8

contribution of A[i]

2 * (1 - 3) = -4

5 * (4 - 1) = 15

3 * (1 - 2) = -3

8

Q1. In how many subarrays, will 5 act as maximum?

A[] = {1 8 3 5 4 2 11 7}

(indices 0 1 2 3 4 5 6 7)

j ← i → k

st

2

3

end

3

4

5

Ans = 6

j = next greater ele idx on LHS

k = next greater ele idx on RHS

st [j+1 i] = i - j

end [i k-1] = k - i

# of subarrays where i^{th} ele is maximum $= (i-j) * (k-i)$

01. In how many subarrays, will 5 act as minimum?

	0	1	2	3	4	5	6	7
A[] = {	1	8	8	5	9	2	11	7 }
	p	←		i		→	q	

st	end
1	3
2	4
3	

p = next smaller ele on LHS

q = next smaller ele on RHS

$st[p+1 \quad i] = i - p$

$end[i \quad q-1] = q - i$

# of subarrays where i will act as minimum $= (i-p) * (q-i)$

$$Ans = \sum_{i=0}^{n-1} A[i] * \Big((i - greaterLeft[i]) * greaterRight[i] - i) - (i - smallerLeft[i]) * smallerRight[i] - i) \Big)$$

Final sol