Q.1 Nearest Smaller Element

[Google] [FB] [Amazon] [Apple] Given an Array of +ve integers, for every i, find the nearest element on the left side which is smaller than A[i].

A: [4, 2, 5, 10, 8, 2]
NSL[]: -1 -1 2 5 5 -1

A: [6, 10, 11, 12, 7]
NSL[]: -1 6 10 11 6

Brute force

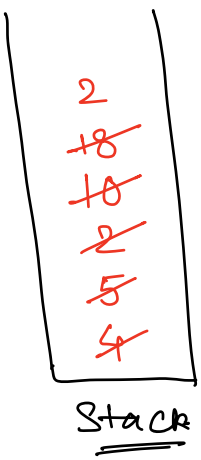
for every i :
    iterate from (i-1) to 0 :
        return the first element smaller than A[i].

TC: $O(N^2)$

Ex A: 4 6 2 8 6
NSL[]: -1 4 -1 2 2

Ex

	0	1	2	3	4	5
A[]	4	5	2	10	18	2
NSL[]:	-1	4	-1	2	10	-1



st.top()

- $< A[i]$
  - ans[i] = st.top();
  - st.push(A[i])
- $\geq A[i]$
  - keep popping from stack till stack is empty OR st.top() < A[i].
  - if (st.isEmpty())
    ans[i] = -1
    else
    ans[i] = st.top()
  - st.push(A[i])

```
ans[]:
Stack<int> st;
for(i=0; i<N; i++) {
    while(!st.isEmpty() && st.top()>=A[i]){
        st.pop();
    }
    if( st.isEmpty() ) {
        ans[i] = -1;
    }
    else {
        ans[i] = st.top();
    }
    st.push(A[i]);
}
```

TC: O(N)

SC: O(N)

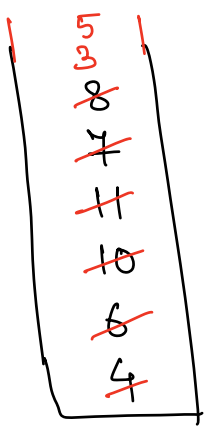
Q.2 Find the index of Nearest Smaller element on the left.

```
ans[] :
Stack<int> st;
for(i=0; i<N; i++) {
    while(!st.isEmpty() && A[st.top()]>=A[i]){
        st.pop();
    }
    if( st.isEmpty() ) {
        ans[i] = -1;
    }
    else {
        ans[i] = st.top();
    }
    st.push(i);
}
```

Quiz

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

ans[] : -1 4 6 10 6 7 -1 3



Q.3 Get the distance of the nearest smaller on the left.

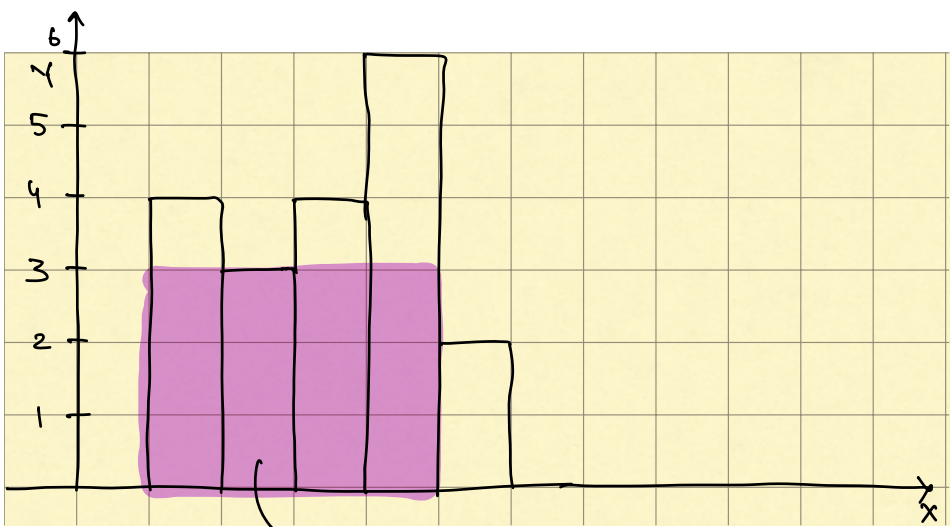
Q.4 Find the nearest smaller on the right side (N-1 ⇒ 0)

Q.5 Find the nearest greater on the left side.

Q.6 Find the nearest greater on the right side. (N-1 ⇒ 0)

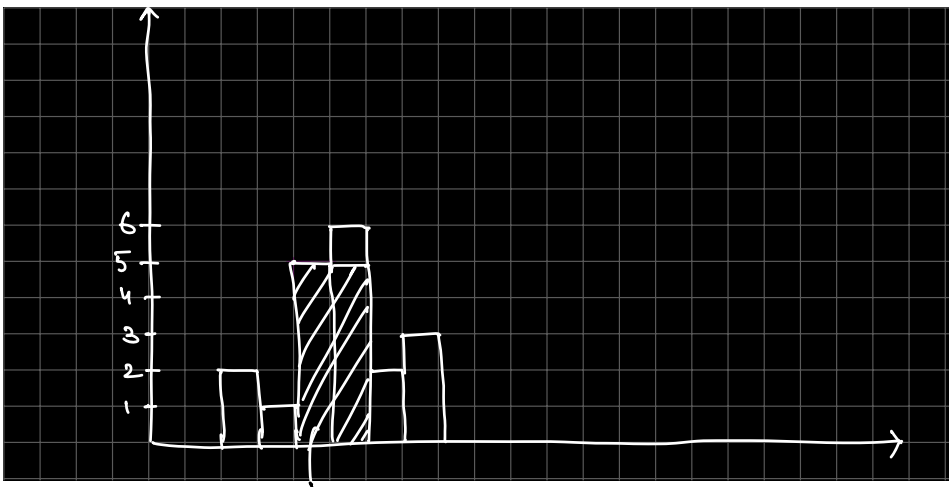
Q.7 Largest rectangle area in a Histogram.

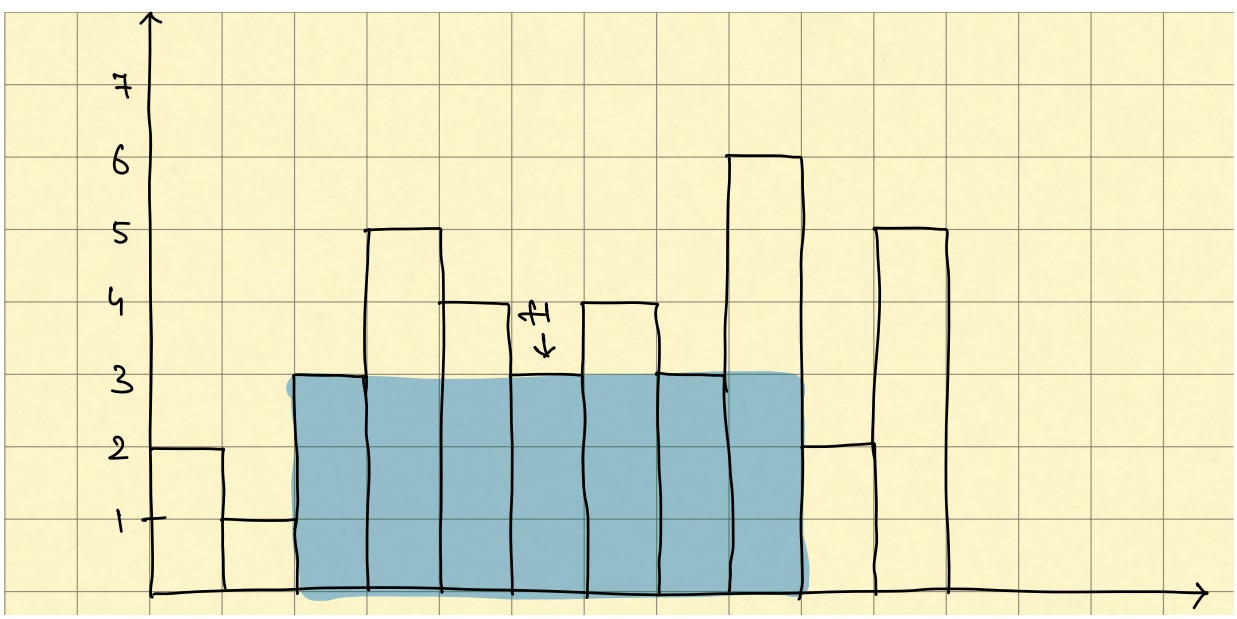
Amazon



A = 4 x 3

= 12





A : [2 1 3 5 4 3 4 3 6 2 5]

0 1 2 3 4 5 6 7 8 9 10

S = 1

i = 5

e = 9

wd = e − s − 1

height = H[i]

Area = wd × height.

e : index of Nearest smaller element on Right

s : index of Nearest smaller element on left

O(N) ⇐ NSL [] ⇒ Nearest smaller element on left

O(N) ⇐ NSR [] ⇒ Nearest smaller element on Right

O(N) {

```
for( i= 0; i< N; i++) {
    wd = NSR[i] - NSL[i] - 1
    ht = A[i]
    area = wd * ht
    max_area = max(max_area, area);
}
```

}

TC : O(N)

SC : O(N)

Q.8 Given an Array :
Find the sum of (max - min) for all possible subarrays.

Google.

A : [2 , 5 , 3]
(0) (1) (2)

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
				8

Brute force

Iterate over all the subarrays, find max & min and keep adding (max - min) to ans.

TC : $O(N^3)$

Contribution Technique

$$ans = (2\times1 - 2\times3) + (5\times4 - 5\times1) + (3\times1 - 3\times2)$$
$$= -4 + 15 - 3 = 8$$

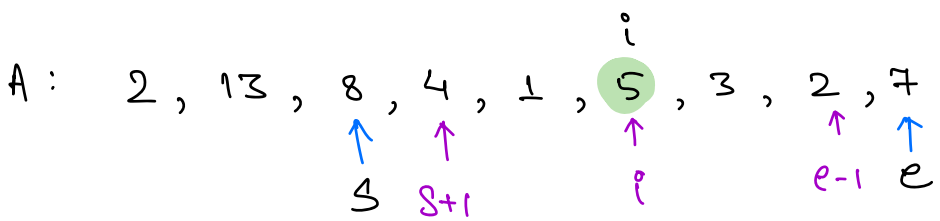
⇒ For every A[i], find in how many subarrays it will be present as MAX & MIN.

⇒ for every element A[i]:

find count of subarrays in which A[i] is MAX ⇒ (x)

find count of subarrays in which A[i] is MIN ⇒ (y)

$$ans += (A[i]*x - A[i]*y)$$



s: Nearest Greater Element on left

e: Nearest Greater Element on right

No. of subarrays in which A[i] will be as MAX

$$x = (i - s) * (e - i)$$

$[s+1, i]$ $[i, e-1]$

No. of subarrays in which A[i] will be as MIN

$$y = (i - s) * (e - i)$$

$[s+1, i]$ $[i, e-1]$

s: Nearest Smaller Element on left

e: Nearest Smaller Element on right.

for every i :

ans += $(x * A[i] - y * A[i])$

⇒ Build NGL[], NGR[], NSL[], NSR[];

```
for (i = 0; i < N; i++) {
    max = (i - NGL[i]) * (NGR[i] - i) * A[i]
    min = (i - NSL[i]) * (NSR[i] - i) * A[i]
    ans += (max - min);
}
```

TC : O(N)

SC : O(N)