Google
Facebook
Amazon
Apple

## Q Nearest smaller element.

Given an array of +ve integers.
For every i, find the nearest element on the left side of i which is smaller than A[i]

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 4 | 2 | 5 | 10 | 8 | 2 |
| -1 | -1 | 2 | 5 | 5 | -1 |

| | | | | |
|---|---|---|---|---|
| 6, | 10, | 11, | 12, | 7 |
| -1 | 6 | 10 | 11 | 6 |

### Brute force

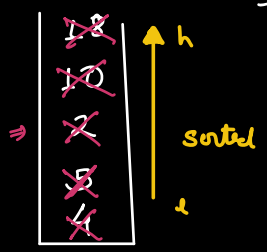
for every i,
  iterate from (i-1) to 0
    set 1st element smaller than A[i]

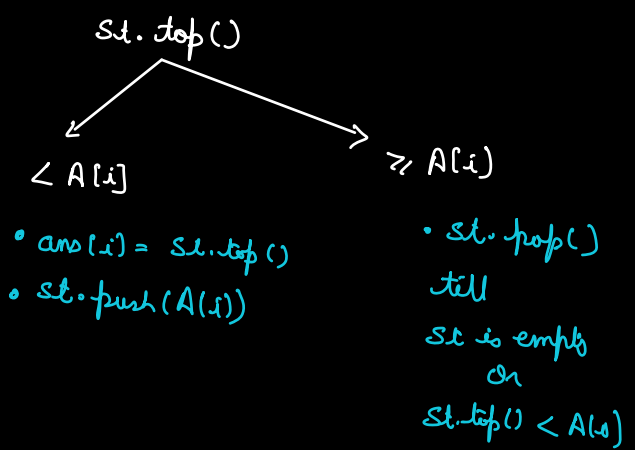
TC: $O(N^2)$

| | | | | |
|---|---|---|---|---|
| 4 | 6 | 2 | 8 | 6 |
| -1 | 4 | -1 | 2 | 2 |


⇒ Stack

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 4 | 5 | 2 | 10 | 18 | 2 |
| -1 | 4 | -1 | 2 | 10 | -1 |



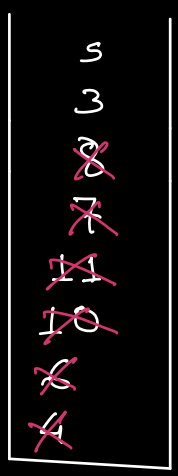




```
ans → [ ];
St → Stack<Integer>()
for(i=0; i<N; i++) {
    while ( !St.isEmpty() && St.top() >= A[i]) {
            St.pop();
    }
    if ( St.isEmpty()) {
            ans[i] = -1;
    }
    else {
            ans[i] = St.top();
    }
    St.push(A[i]);
}
```

Q Find the index of nearest smaller on left.

```
ans → [ ];
st → Stack<Integer>()
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 );
}
```

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 4, | 6, | 10, | 11, | 7, | 8, | 3, | 5 |
| -1 | 4 | 6 | 10 | 6 | 7 | -1 | 3 |




TC : O(N) ⇒ [2N → N push + N pop]

SC : O(N)

Q3 Get the distance of nearest smaller on the left side ?

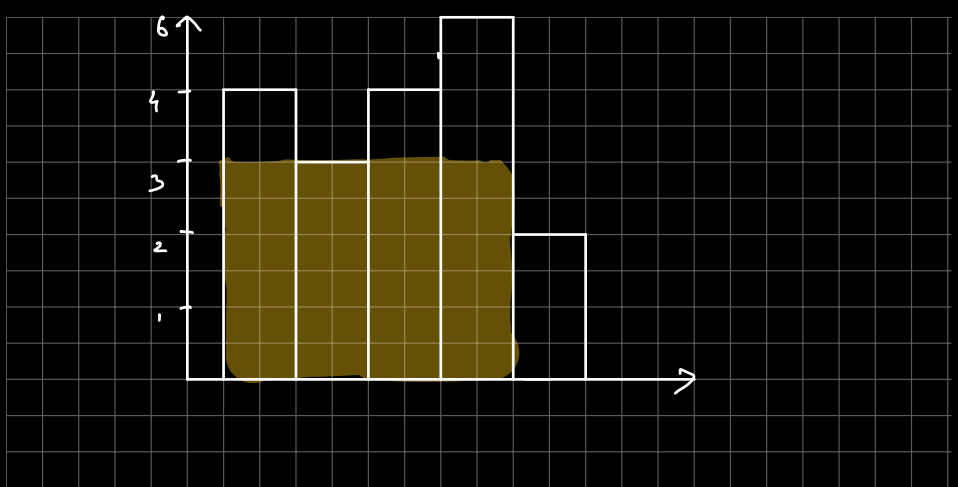
Q4 Find the nearest smaller on the right side ? (N-1) → 0

Q5 Find the nearest greater on left ?

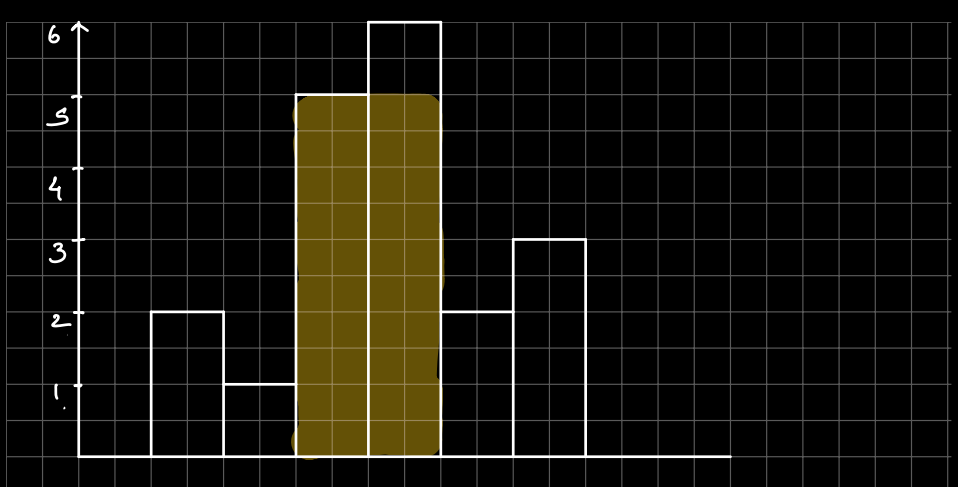
Q6 Find the nearest great on right ? (N-1) → 0

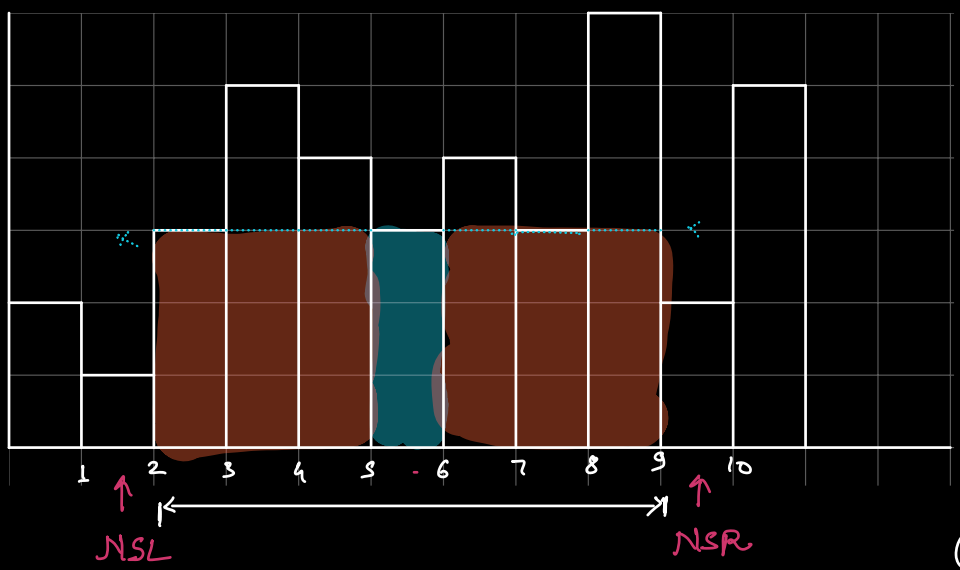
Amazon

Q Largest rectangle area in a histogram.




3.5






$(a,b) = b-a-1$

$[a,b] = b-a+1$

(NSL, NSR)

(2, 10)

$\Rightarrow 10-2-1$

$= 7$

NSL → [ ] // Nearest smaller on left O(N)

NSR → [ ] // Nearest smaller on right O(N)

for every height H[i] : O(N)

{

```
wd = NSR[i] - NSL[i] - 1;
ht = H[i];
area = max(area, wd x ht);
```

}

TC : O(N)

SC : O(N)

Google Q Scaler

Given an array.
Find the sum of (max - min) for all possible subarrays. ⇒ (subseq → sorty)

$\overset{0}{2}$ $\overset{1}{5}$ $\overset{2}{3}$

| 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 |

ans ⇒ 8

Brute Force

O(N³)

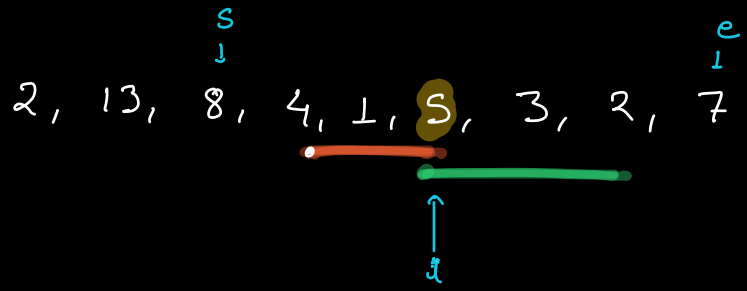
(2 × 1) − (2 × 3)
+
(5 × 4) − (5 × 1) ⇒ 8
+
(3 × 1) − (3 × 2)

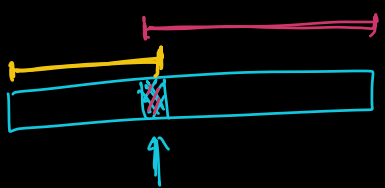
# Contribution Tech

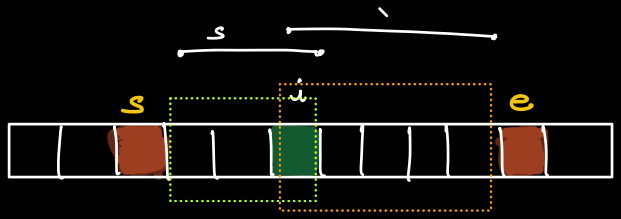
for every element;

find count of subarrays in which it is max

find count of subarrays in which it is min






No of subarrays where A(i) is max $= (i-s) * (e-i)$

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

(s, i] $= i - s$

s → index of NGL[i]

e → index of NGR[i]

NGL ⟶ [ ]  NSL → [ ]

NGR → [ ]  NSR → [ ]

// $\sum_{max_{sub}} = \sum_{i=0}^{N-1} (i - NGL(i)) \times (NGR[i] - i) \times A[i]$

// $\sum_{min_{sub}} = \sum_{i=0}^{N-1} (i - NSL(i)) \times (NSR[i] - i) \times A[i]$

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