Amazon, MS, Twitter, FB, Uber, LinkedIn

Q1 Nearest smallest on the left.

Given array of integers, for every index i, find the nearest index on the left side which is smaller than A[i]

	0	1	2	3	4	5
Eg1	4	5	2	10	8	2
ans	-1	0	-1	2	2	-1

	0	1	2	3	4	5	6
Eg2	5	2	8	10	12	6	1
ans	-1	-1	1	2	3	1	-1

Brute: Iterate on each elem, and for each elem, keep going left till you get a smaller no.

TC: $O(n^2)$

8 x x x x 5 x x x x

0	1	2	3	4	5	6
5	2	8	10	12	6	1
-1	-1	1	2	3	1	-1

Consider a box

$\boxed{6 \qquad\qquad\qquad\qquad}$

contains indexes

Any elem > 5 is obviously > 2.
Thus 5 is useless.

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

$\boxed{6 \quad 7 \qquad\qquad\qquad}$

What is such a data structure?
Stack

- while top elem is bigger,
  remove top.

Code

```
ans[n]
stack<int> s
for (i=0 ; i<n ; i++) {
    while ( !s.empty() && A[s.top] >= A[i] ){
        s.pop()
    }
    if ( s.size() > 0 )
        ans[i] = s.top()
    else
        ans[i] = -1
    s.insert(i)
}
return ans
```

TC: $O(n)$

SC: $O(n)$

inserts → n

deletions → n

total ops ⇒ $2n$

nsl   nsr

ngl   ngr

Variations

1) Get dist from nearest smallest on left

dist = i - nsl[i]

2) Nearest greater on left

while (A[s.top()] <= A[i])
    s.pop()

3) Nearest smaller on right

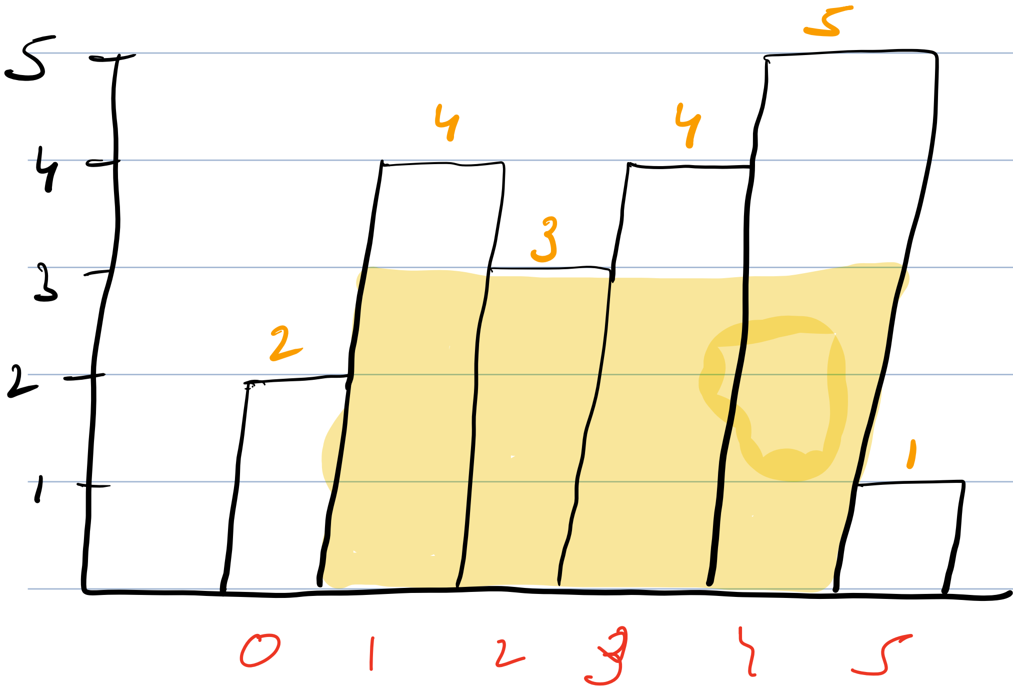
Iterate from right to left

for (i=n-1 ; i>=0 ; i--)

if (s.empty)
    ans[i] = n

bar graph

Q2 Given a continuous histogram, find max rectangular area not exceeding the histogram.



ans = 3x4 = 12

Brute ⇒ Take all pairs of bars as endpoints & calc area.

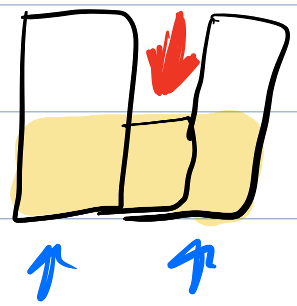
Let end point be i & j

width = j - i + 1

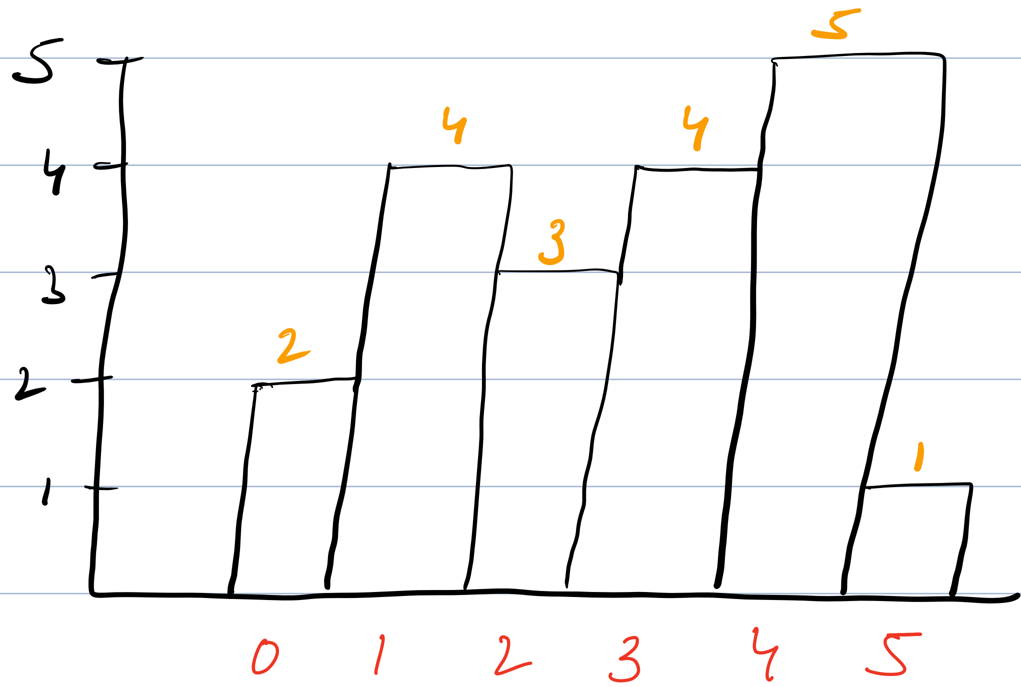
Height = min of all bars [i:j]

area = width * height

Obs: The min height b/w any 2 is actually the height of a bar between the endpoints.



Idea Consider each bar as minimum and calc max area for that bar.



What do we need?

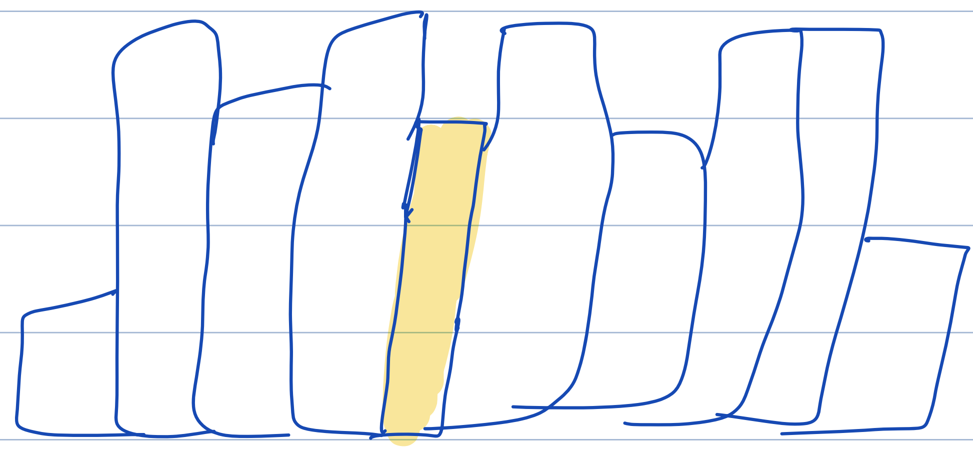
nearest smaller on left &
nearest smaller on right

Code

```
ans = 0
for (i=0 ; i<n ; i++){
    height = arr[i]
    x1 = nearest smallest on left
    x2 = nearest smallest on right
    width = x2 - x1 - 1
    area = height * width
    ans = max(ans, area)
}
```

TC: $O(n)$

SC: $O(n)$



x_1 $\quad$ i $\quad$ x_2

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

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

$[x_1, i]$ $\quad$ $i - x_1$

03 Sum of max−min over all subarrays

Eg ⇒ 2 5 3

2 →	2 − 2	5 →	5 − 5	3 →	3 − 3
2,5 →	5 − 2	5,3 →	5 − 3	2,5,3 →	5 − 2

Brute: For all subarrays, calc max & min
do ans += max − min

Idea ⇒ $\sum_{\text{all subarrays}} (\max - \min)$

$= \sum_{\text{all subarrays}} (\max) - \sum_{\text{all subarrays}} (\min)$

Now, how to get sum(max) for all subarrays.

⇒ Contribution technique

For sum(max)

get nearest bigger on left and
nearest bigger on right array

$[x_1, i]$

0 1 2 3 4 5 6 7 8

12, 3, 5, 1, 8, 9, 7, 3, 11

ngl[5] = idx 0

ngr[5] = idx 8

- Why this is required ?

All elems b/w ngl[i] & i are smaller?

All elems b/w i & ngr[i] are smaller

so any subarray formed this way would have this i idx as max

$\Rightarrow$ left = i - ngl[i]

right = ngr[i] - i

subarrays = left * right

$$\text{contribution} = \sum_{i=0}^{n-1} \text{subarrays} * a[i]$$

Similarly for min, use nsl & nsr

For min

$\Rightarrow$ left = i - nsl[i]

right = nsr[i] - i

subarrays = left * right

contribution = $\sum_{i=0}^{n-1}$ subarrays * a[i]

{done}

TC : } O(n)
SC : }