Stacks 2 : Nearest Smaller / Greater Element

Question

Given an integer array, find the index of nearest smaller element in left of index in array A.

If not possible → mark -1.

	0	1	2	3	4	5	6	7
A = [	8	2	4	7	7	5	3	10]

nearest smaller in left =	NA	NA	2	4	4	4	2	3
index = [	-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]
indices = [	-1	0	1	2	1	4	-1	6]

	0	1	2	3	4	5
A = [	4	5	2	10	8	2]
indices = [	-1	0	-1	2	2	-1]

Bruteforce

ti, iterate to left & find the nearest smaller element.

$TC = O(N^2)$ $SC = O(1)$

Observation

A = [8 _ _ _ _ 5 . . . _ .] (0 1 2 3 4 5 6 7 8)

for any element x, can index 0 be the answer?

if $X > 8 \Rightarrow X > 5$

since 5 is closer than 8, index 0 will NEVER be the answer.

	0	1	2	3	4	5	6	7
A =	8	2	4	7	7	15	3	10
ans =	-1	-1	1	2	2	4	1	6

possible answer indices: ~~0~~ 1 ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6 7

last index is checked first ⇒ LIFO

code

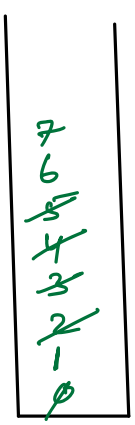
```
int ans[n]
for( i=0 to n-1) {
    while( !st.isEmpty() && A[st.peek()] >= A[i] ){
        st.pop()
    }
    if( st.isEmpty() )    ans[i] = -1
    else    ans[i] = st.peek()

    st.push(i)
}
return ans
```

TC = O(N)

SC = O(N)

	0	1	2	3	4	5	6	7
A = [	8	2	4	7	7	15	3	10]
ans = [	-1	-1	1	2	2	4	1	6]



$i=0$	while runs	
	X_0	times
$i=1$	X_1	times
$i=2$	X_2	
⋮		
$n-1$	X_{n-1}	times

total iterations $= X_0 + X_1 + X_2 \cdots + X_{n-1}$

$= N$

Question

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

	0	1	2	3	4	5	6
ratings :	3	2	6	5	8	7	9

[2 2 4 4 6 6 -1]

In this problem, we have to find nearest greater element in right.

```
for ( i = n-1 to 0 ) {
    while ( !st.isEmpty() && A[st.peek()] <= A[i] ) {
        st.pop()
    }
    if( st.isEmpty() )    ans[i] = -1
    else                  ans[i] = st.peek()

    st.push(i)
}

return ans
```

Question :

For all i, find nearest greater in the left

```
for ( i = 0 to n-1 ) {
    while ( !st.isEmpty() && A[st.peek()] <= A[i] ) {
        .
        .
        .
    }
```

Question

for all i, find the nearest smaller on right

```
for ( i = n-1  to  0 ) {
    while ( !st.isEmpty()  &&  A[st.peek()] >= A[i] ) {
        ⋮
    }
}
```

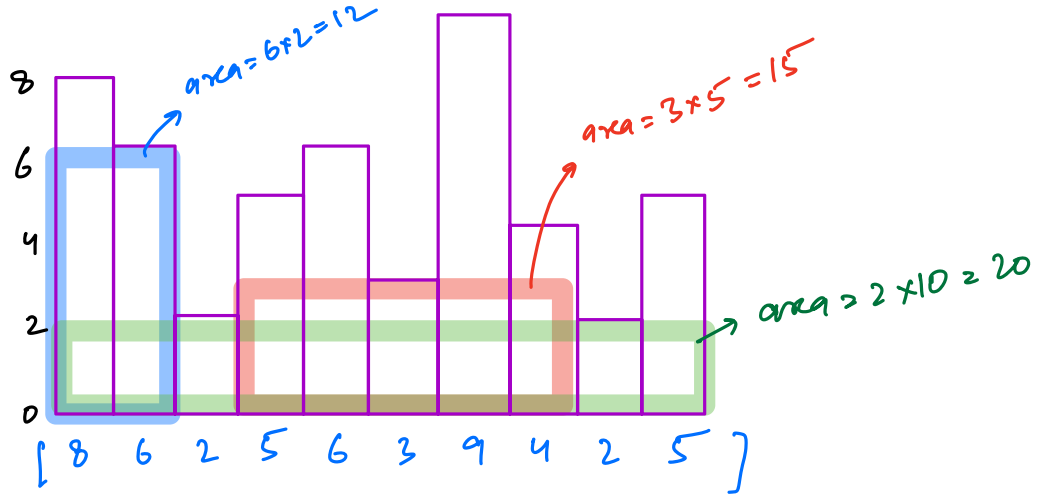
Question

Given an integer A,

i^{th} bar, A[i] = height of i^{th} bar

width of each bar = 1

find the area of largest rectangle formed by continous bars.

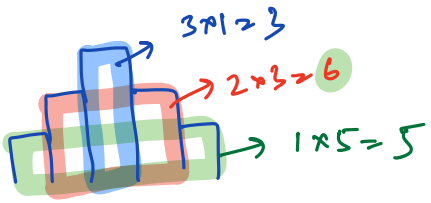


Height = min (A[i])
∀i

width = # bars

0 1 2 3 4 5 6 7 8 9

A = [1 2 3 2 1]



ans=6

Brute force

if subarray (i,j) find H & W & calculate area.

Max. area is answer.

```
aus=0

for( i=0 to n-1) {
    H = a[i]
    for(j=i to n-1) {
        H = min( H, a[j])
        W = j-i+1
        ans = max( ans, H*W)
    }
}
```

TC = O(N²)

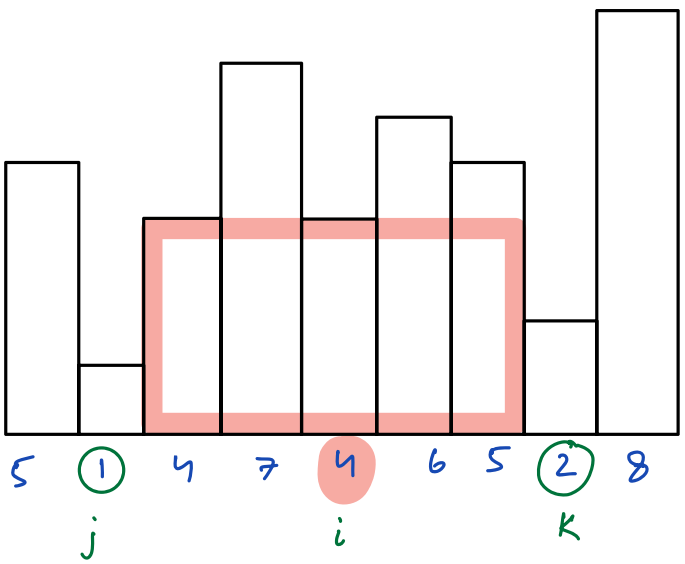
SC = O(1)

# of subarray ⟶ min element

# of width ⟶ height

⇓ reverse

# of height ⟶ width



nearest smaller element in left

nearest smaller element in right

H = a[i] W = K - j - 1 subarray: [j+1, K-1]

if there is no smaller in left

j = -1

if there is no smaller in right

K = n

Code

```
left[n]   // nearest smaller in left of i      TC=O(N)
right[n]  // nearest smaller in right of i     TC=O(N)

ans = 0
for(i=0 to n-1) {
    j = left[i]
    k = right[i]
    if (k == -1)   k = n
    ans = max( ans , a[i] * (k-j-1))
}
return ans
```

$TC = O(N + N + N)$

- create left array
- create right array
- main logic

$= O(N)$

$S(-) = O(N)$

Question

Given an integer array with distinct elements, find (max - min) of subarrays & return their sum as answer.

A = [2 5 3]

2 * (1-3) = -4

5 * (4-1) = 15

3 * (1-2) = -3

=> -4 + 15 - 3 = 8

	(max - min)
2	2 - 2 = 0
2 5	5 - 2 = 3
2 5 3	5 - 2 = 3
5	5 - 5 = 0
5 3	5 - 3 = 2
3	3 - 3 = 0
	8

Bruteforce

```
ans = 0
for (i = 0 to n-1) {
    mx = a[i], mi = a[i]
    for (j = i to n-1) {
        mx = max(mx, a[j])
        mi = min(mi, a[j])
        ans = ans + (mx - mi)
    }
}
return ans
```

TC = O(N^2)

SC = O(1)

```
        mi = min(mi, a[j])
        ans += mx - mi
    }
}
```

A = [1 2 3]

	(max - min)
1	1 - 1 = 0
1 2	2 - 1 = 1
1 2 3	3 - 1 = 2
2	2 - 2 = 0
2 3	3 - 2 = 1
3	3 - 3 = 0

4

Contribution Technique

$$ans = \sum_{i} \text{contribution of } a[i]$$

$$a[i] * \left(\begin{array}{c} \#\text{ subarrays} \\ \text{where } a[i] \text{ is} \\ \max \end{array} - \begin{array}{c} \#\text{ subarrays} \\ \text{where } a[i] \text{ is} \\ \min \end{array} \right)$$

A = [2 (0) | 10 (1, j) | 3 (2) | 5 (3) | 6 (4, i) | 4 (5) | 7 (6, K) | 1 (7) | 3 (8)]

subarrays a[4] is max ?

= 6

3 5 6
5 6
6

3 5 6 4
5 6 4
6 4

nearest greater in left (j) < start <= i [j+1, i]
⇒ (i - j)

i <= end < nearest greater in right (K) [i, K-1]
⇒ (K - i)

subarray = (i - j) × (K - i)

ans = $\sum_i$ contribution of a[i]

a[i] * (# subarrays where a[i] is max − # subarrays where a[i] is min)

(i - greaterLeft[i]) × (greaterRight[i] - i)

(i - smallerLeft[i]) × (smallerRight[i] - i)

TC = O(N + N + N + N + N)

= O(N)

SC = O(N)

	0	1	2
A =	2	5	3
greaterLeft =	-1	-1	1
greaterRight =	1	*3	*3
smallerLeft =	-1	0	0
smallerRight =	*3	2	*3

contribution of a[0] = (0 - (-1)) * (1 - 0) - (0 - (-1)) * (3 - 0)

=> 1 * 1 - (1 * 3) = (1 - 3) * 2 = -4

contribution of a[1], a[2] <- TODO