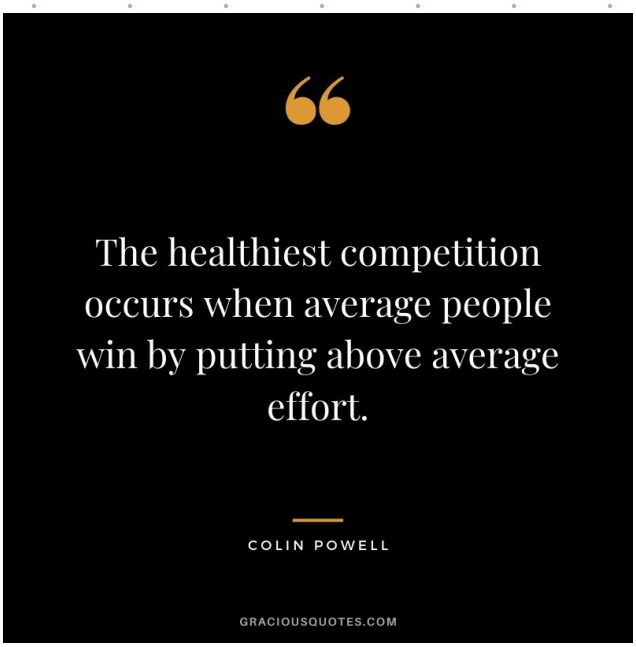
Stacks lab



Good Morning

content

01. Left smaller ele index
02. Double character Trouble
03. Largest Area histogram
04. sum of (max-min) in all subarrays

* Given an integer A[]. Find the nearest smaller element idx on left.

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

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

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

	0	1	2	3	4	5	6	7
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, traverse from i-1 to 0 & search for nearest smaller ele idx

```
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;
```

Optimised solution

A[ ] = { 8 _ _ _ 5 x _ _ }
0 1 2 3 4 5

Can index 0 ever be the answer for x?

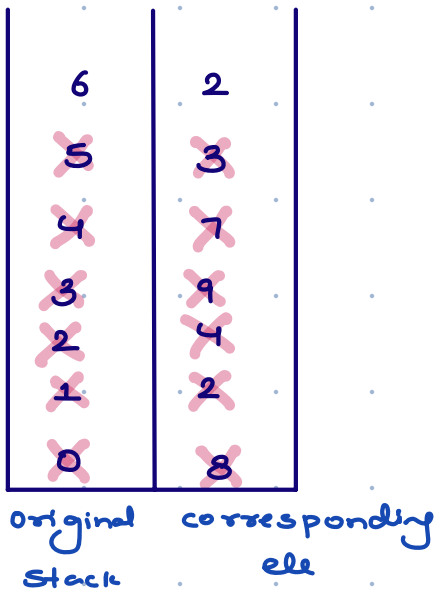
01. $x \leq 5 \Rightarrow ans \neq 0$

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

03 $x > 8 \Rightarrow ans = 4$ idx $(ans \neq 0)$

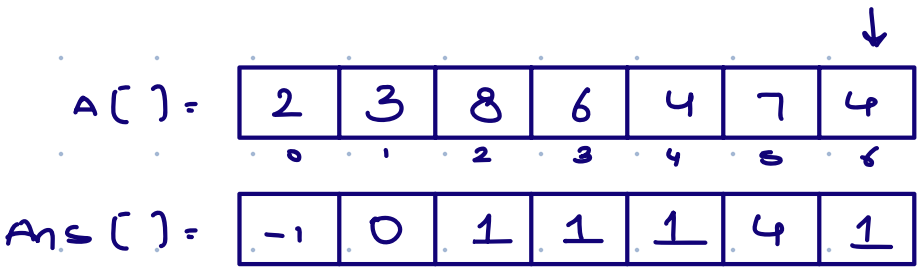
Conclusion → For x, if a smaller no. is present on RHS of larger no., then larger no. can never be the answer.

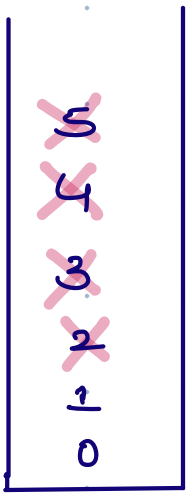
	0	1	2	3	4	5	6 ↓
A[ ] =	8	2	4	9	7	3	2
ans[ ]	-1	-1	1	2	2	1	-1




```
Stack<I> st = new Stack<>();
int[] ans = new int[n];
for(i=0; i<n; i++){
    while(st.size()>0 && A[st.peek()] ≥ A[i])
        st.pop();
    if(st.size()==0) ans[i]=-1
    else ans[i] = st.peek();
    st.push(i);
}
```

TC : O(N)
SC : O(N)





$\forall i$, find nearest smaller or equal ele idx on left.

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

$\forall i$, find nearest greater ele idx on left.

A[st.peek()] ≤ A[i]

Double Trouble

Given a string str. Remove equal pair of adjacent characters. Return string without adjacent duplicates.

str = "abbd" ⇒ "ad"

str = "abccbd" ⇒ abbd ⇒ ad

str = "abbbd" ⇒ abd

str = abbcbbcacx

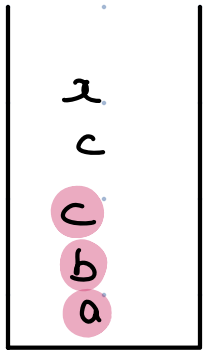
⇒ acbbcacx

⇒ accacx

⇒ aacx ⟶ cx

Approach → At every character, need to look at previously visited character

str = "abccbacx"
↑



ans = x c

⇓

reverse (ans)

⇒ cx

```java
public class Solution {
    public String solve(String A) {
        Stack<Character>st=new Stack<>();

        for(int i=0;i<A.length();i++){
            char ch=A.charAt(i);

            if(st.size()==0 || st.peek()!=ch) st.push(ch);
            else st.pop();
        }

        StringBuilder sb=new StringBuilder();
        while(st.size()>0){
            sb.append(st.pop());
        }
        return sb.reverse().toString();
    }
}
```

8:11 → 8:13 AM

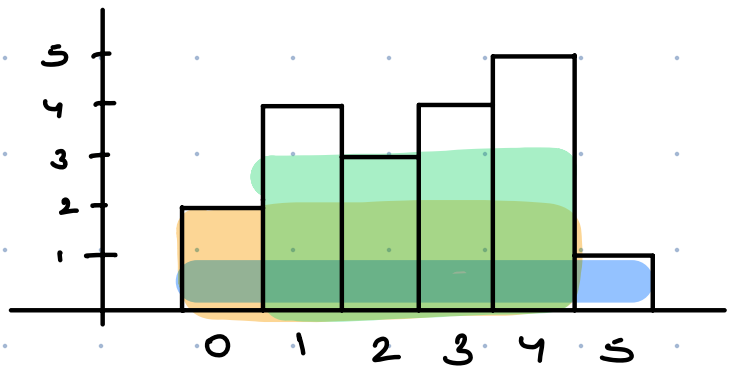
Histogram Area

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

Note :- Every histogram is of width = 1

Area of rectangle = ht * width

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

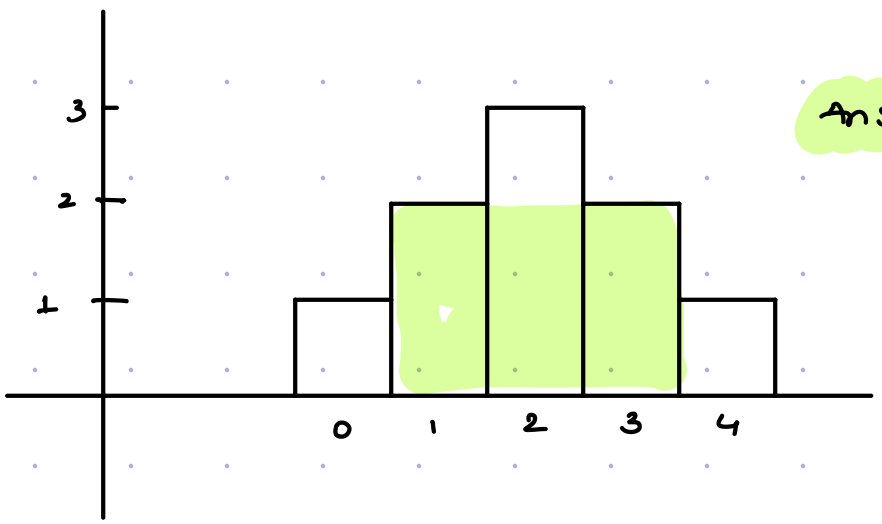


Area = 6

Area = 10

Area = 12 → final ans

A[] = { 1 2 3 2 1 }



Ans = 2 * 3 = 6

Brute force → Consider all possible combination of st point & end point.

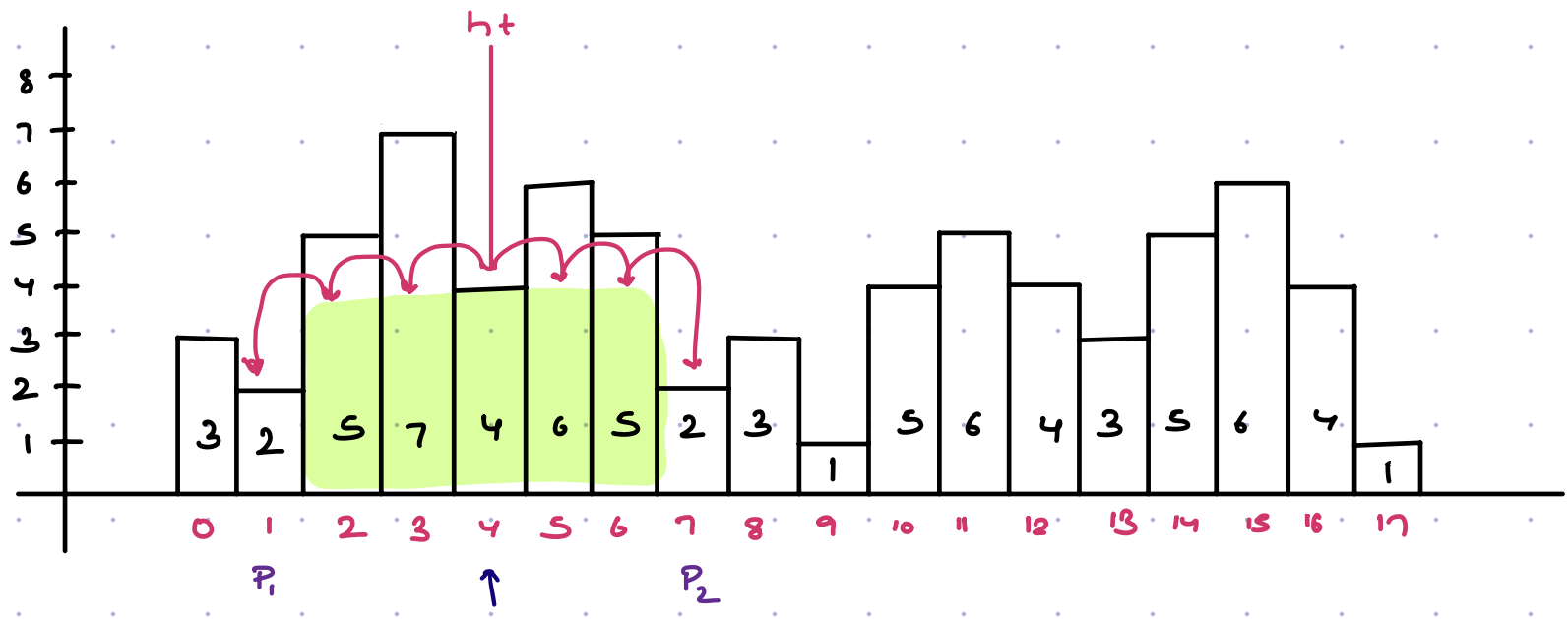
```
ans=0
for ( i=0; i<n; i++ )
    ht= A[i];
    for ( j=i ; j<n; j++ )
        ht= min ( ht , A[j])
        width = j-i+1;
        area = ht * width
        ans= Max ( area, ans);
```

TC : $O(n^2)$

SC : $O(1)$

ar [ ] =

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



left = -1 -1 1 2 1

right = 1 9 4 4 7

Idea → Consider each histogram as height of rectangle & expand on left & right

- Nearest smaller ele idx on left (P_1)
- Nearest smaller ele idx on right (P_2)

$width = P_2 - P_1 - 1$

```
int rectangleArea(int[] ht)
    int[] left = nearestSmallerEleIdxOnLeft(ht)      → default = -1
    int[] right = nearestSmallerEleIdxOnRight(ht)    → default = n
    int ans = 0, area = 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(area, ans);
    }
    return ans;
}
```

TC: O(N)
SC: O(N)

* Sum of (Max - Min) for all subarrays

Q Given a distinct integer array A. For all subarrays, find (Max - Min) & return the sum of all these differences.

$$A[] = \{2 \underset{0}{} \quad 5 \underset{1}{} \quad 3 \underset{2}{}\}$$

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 }
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

Bruteforce → Generate all subarrays & for every subarr, find min & max

get the difference & add it to a global ans

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

TC : O(N³)
SC : O(1)

* Contribution Technique

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

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

$A[] = \{ \overset{0}{2} \ \overset{1}{5} \ \overset{2}{3} \}$

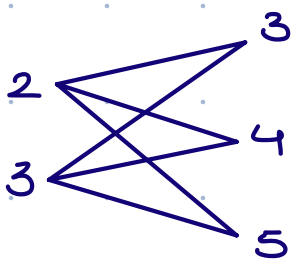
Subarr		Max - Min	contribution of A[i]
[0 0]	$= \{2\}$	$2-2=0$	$2*(1-3) = -4$
[0 1]	$= \{2,5\}$	$5-2=3$	$5*(4-1) = 15$
[0 2]	$= \{2,5,3\}$	$5-2=3$	$3*(1-2) = -3$
[1 1]	$= \{5\}$	$5-5=0$	Ans = 8
[1 2]	$= \{5,3\}$	$5-3=2$	
[2 2]	$= \{3\}$	$3-3=0$	

Ans = 8

Q In how many subarrays, will A[3] act as max.

$A[] = \{ \overset{0}{1} \ \overset{1}{8} \ \overset{2}{3} \ \overset{3}{5} \ \overset{4}{4} \ \overset{5}{2} \ \overset{6}{11} \ \overset{7}{7} \}$

st end



ans = 2 * 3 = 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

subarr where A[i] is maximum = (i - j) * (k - i)

Q In how many subarrays, will A[3] act as minimum?

	0	1	2	3	4	5	6	7
A[] = {	1	8	3	5	4	2	11	7 }

p ← i → q

p = next smaller ele idx on left

q = next smaller ele idx on right

subarr where A[i] is minimum = (i - p) * (q - i)

j = next greater ele idx on LHS

k = next greater ele idx on RHS

p = next smaller ele idx on left

q = next smaller ele idx on right

```
for (i=0; i<n; i++) {
    ans += ((i-j)*(k-i)) * ((i-p)*(q-i))
}
```