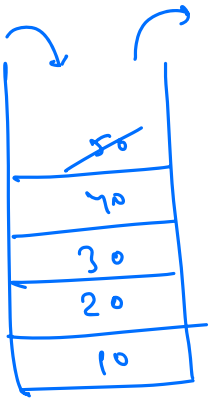
→ Quick Revision

→ Remove Equal Pair of Consecutive Elements

→ Largest Area in Histogram

→ Sum of max. min for all subarrays.

① Implement stack using linkedlist



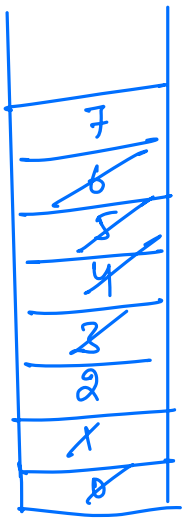
~~(1)~~ Add at tail, remove from tail → O(n)

(2) Add at head, remove from head.

② Nearest smaller on Lhs

arr[] - [5 4 2 7 6 9 15 3]
0 1 2 3 4 5 6 7

nsl[] - [-1 -1 -1 2 2 4 5 2]



→ pop all greater elements.

→ update ans

→ push current index in stack.

Code →

```
Stack<int> st;
nsl[N];
for(i=0; i<N; i++){
    // pop all greater elements
    while(st.size() > 0 && arr[st.peek()] >= arr[i]){
        st.pop();
    }
    // update ans
    if(st.size() == 0) { nsl[i] = -1 }
    else { nsl[i] = st.peek() }
    // push current index in stack
    st.push(i);
}
return nsl[];
```

T.C → O(N)
S.C → O(N)

③ Nearest smaller on r.h.s

⇓

iterate from r.h.s.

Q → Given a string, remove equal pair of consecutive elements till possible

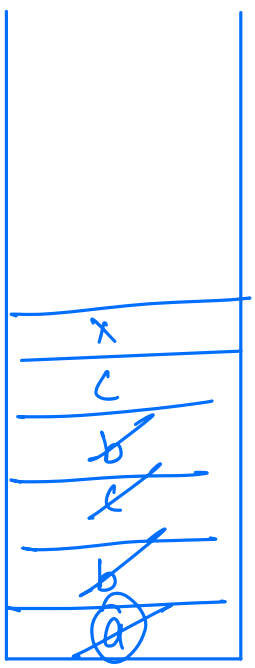
Str → abcdd ans → abc

Str → abcddc → abcc ans → ab

Str → abbcbbcacx → accacx → cx

Str → abbb → ab

abbcbbcacx



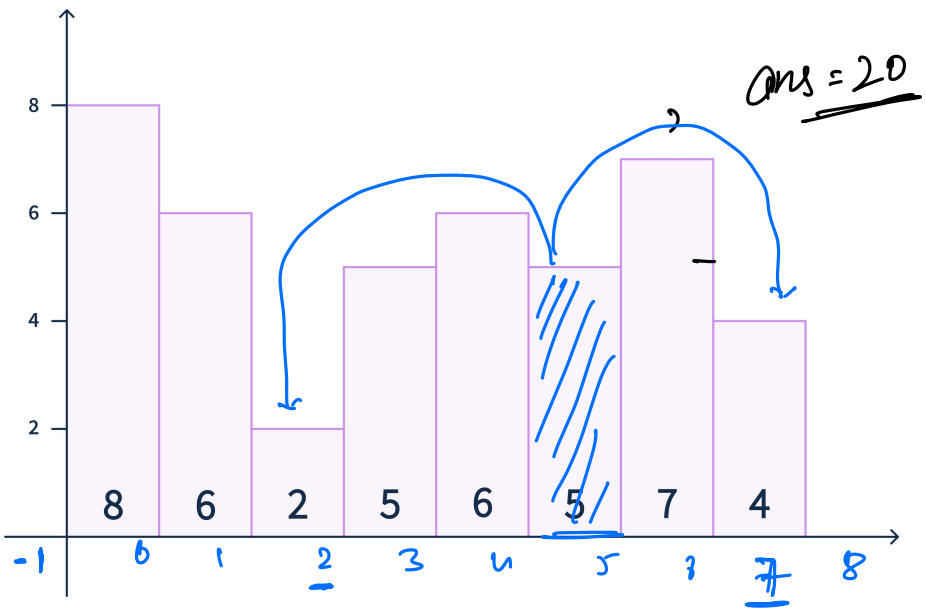
ans → cx

code -->

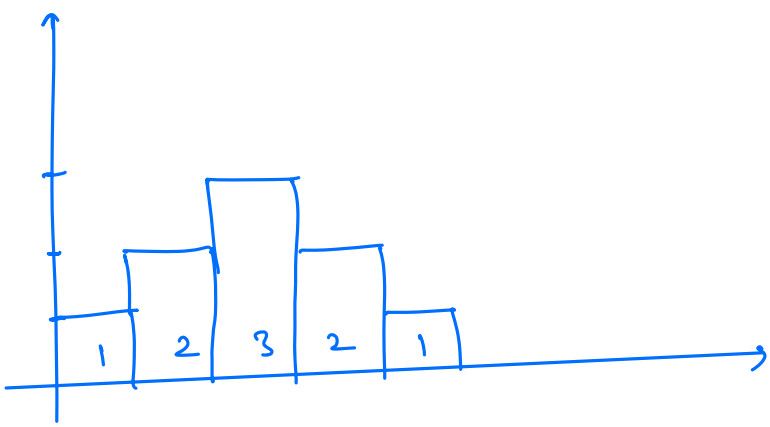
```
Stack < character > st;

for( i = 0; i < N; i++){
    if( st.size() == 0 || st.peek() != str[i]){
        st.push( str[i]);
    }
    else{
        st.pop();
    }
}

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

T.C → O(N)
S.C → O(N)

< *Question* > : Find the largest area of rectangle (formed by continuous bars) in histogram.



arr[] → [1, 2, 3, 2, 1]



5, 6, 3, 6, 5
max

ans = 6

B.f - for every bar,

→ find first bar with height less than or equal to current bar on l.h.s

→ find first bar with height less than or equal to current bar on r.h.s.

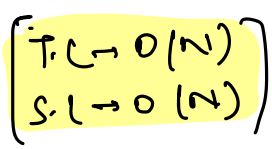
T.C → O(N²)
S.C → O(1)

optimisation → find nearest smaller on l.h.s,
nearest smaller on r.h.s.

→ arr[i] × (nsr[i] - nsl[i] - 1)

consider this factor for every bar.

```java
public class Solution {
    public int largestRectangleArea(int[] arr) {
        int[] nsl = new int[arr.length];
        int[] nsr = new int[arr.length];

        Stack<Integer> st = new Stack<>();
        for(int i = 0; i < arr.length; i++){
            while(st.size() > 0 && arr[st.peek()] >= arr[i]){
                st.pop();
            }
            if(st.size() == 0){
                nsl[i] = -1;
            }else{
                nsl[i] = st.peek();
            }
            st.push(i);
        }

        st = new Stack<>();
        for(int i = arr.length - 1; i >= 0; i--){
            while(st.size() > 0 && arr[st.peek()] >= arr[i]){
                st.pop();
            }
            if(st.size() == 0){
                nsr[i] = arr.length;
            }else{
                nsr[i] = st.peek();
            }
            st.push(i);
        }

        int ans = 0;
        for(int i = 0; i < arr.length; i++){
            ans = Math.max(ans, arr[i] * (nsr[i] - nsl[i] - 1));
        }
        return ans;
    }
}
```



< *Question* > : Find sum of (max-min) for all subarrays.

$1 \le N \le 10^5$

arr : [1 2 3]

0 1 2

$\sum$ max - min

$\Rightarrow \sum$ max $-$ $\sum$ min

	max	min	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
	14	10	4

BF Idea

Consider all subarrays & for every subarray find min and max.

[T.C → $O(N^3)$, S.C → $O(1)$]

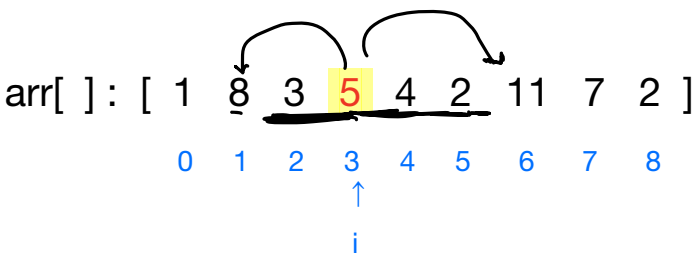
↓ carry forward

T.C → $O(N^2)$

```
ans = 0;
for( i = 0; i < N; i++) {
    min = arr[i], max = arr[i]
    for( j = i; j < n; j++) {
        min = Math.min( min, arr[j])
        max = Math.max( max, arr[j])
        ans += max - min;
    }
}
return ans;
```

T.C → O(N²)
S.C → O(1)

< *Question* > : In how many subarrays, ith element is the maximum element?



options for si

2
2

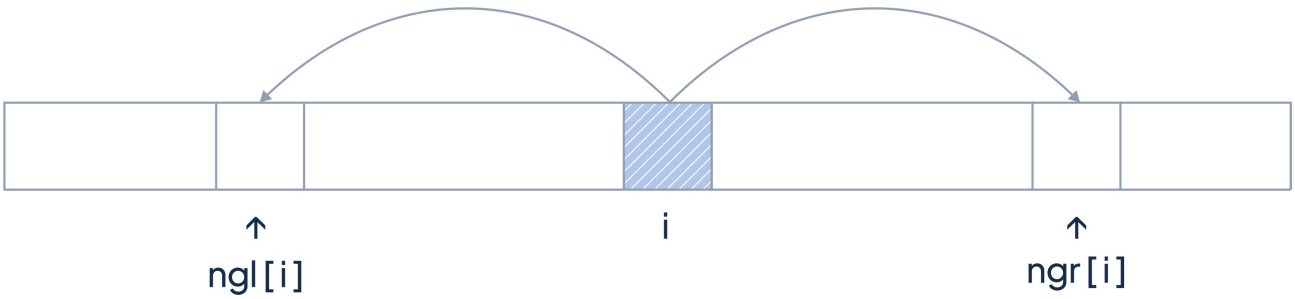
options for ei

3
4
5

[3,5]
[3,5,4]
[3,5,4,2]
[5]
[5,4]
[5,4,2]

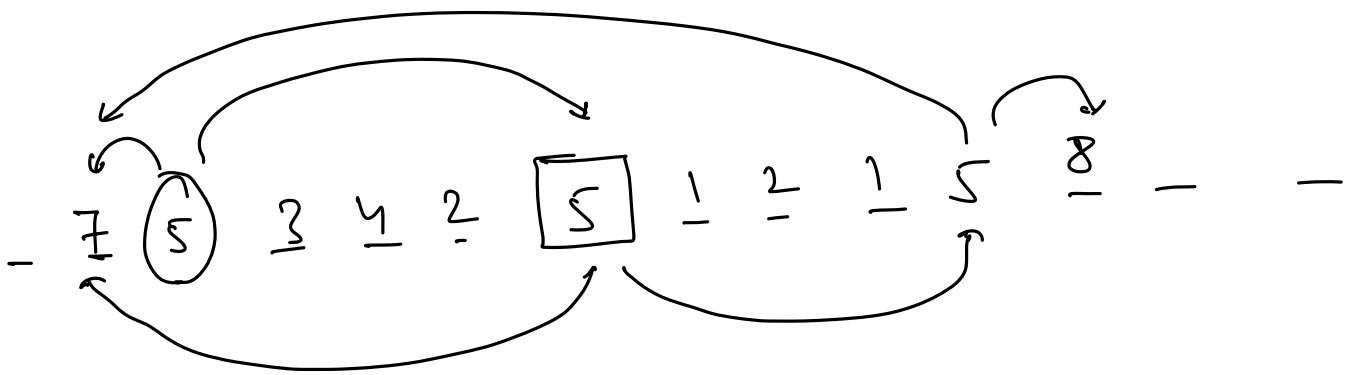
(i - ngl(i)) x (ngr(i) - i)

- ***Generalisation***



total subarrays where arr(i) is max → $(i - ngl(i)) \times (ngr(i) - i)$

total subarrays where arr(i) is min → $(i - nsl(i)) \times (nsr(i) - i)$



nsl ngl

nsr ngr

< / > Code

```
    st = new Stack<>();
    for(int i = arr.length - 1; i >= 0; i--){
        while(st.size() > 0 && arr[st.peek()] <= arr[i]){
            st.pop();
        }
        if(st.size() == 0){
            ngr[i] = arr.length;
        }else{
            ngr[i] = st.peek();
        }
        st.push(i);
    }

    long ans = 0;
    int mod = (int)1e9 + 7;
    for(int i = 0; i < arr.length; i++){
        long maxContribution = arr[i] * (i - ngl[i]) * (ngr[i] - i);
        long minContribution = arr[i] * (i - nsl[i]) * (nsr[i] - i);

        ans = (ans + (maxContribution - minContribution) + mod) % mod; // check
    }
    return (int)ans;
}
```

check it out

T.C → O(N)
S.C → O(N)

Contest 2: Searching, Linked List, Stacks, Queues & Trees,