Today's content:

- Next Smaller Element
  - on l.h.s
  - on r.h.s
- Next Greater Element (NGE)
  - on l.h.s
  - on r.h.s
- Index of nearest smaller.
- Largest Rectangle in Histogram.
- Sum of (max - min) for all subarrays

Nearest Smaller Element

Given an array of size N. for every index i, find the nearest element which is smaller then ith element on left side.

Eg - arr: [4 5 2 10 3 2]
(0 1 2 3 4 5)

ans - [-1 4 -1 2 2 -1]

(N)

Quiz: [4, 6, 10, 11, 7, 8, 3, 5]
(0 1 2 3 4 5 6 7)

ans. [-1 4 6 10 6 7 -1 3]

(N)

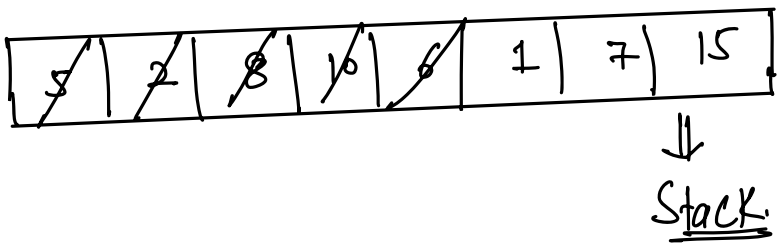
B.f. → for every element, traverse on lhs elements and find the first smaller element.

```
ans[N], ∀i, ans[i] = -1
for(i = 1; i < N; i++){
    for(j = i-1; j >= 0; j--){
        if(arr[j] < arr[i]){
            ans[i] = arr[j];
            break;
        }
    }
}
return ans[];
```

T.C → O(N^2)
S.C → O(1)

Eg → [5 2 8 10 6 1 7 15]
0 1 2 3 4 5 6 7

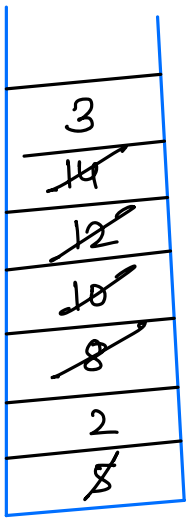
ans - [-1 -1 2 8 2 -1 1 7]



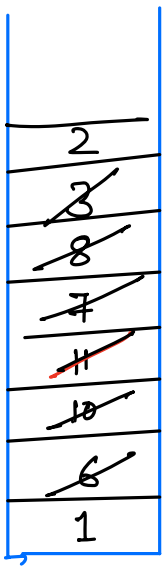
⇓

Stack.

[5, 2, 8, 10, 12, 14, 3]
-1 -1 2 8 10 12 2



[1, 6, 10, 11, 7, 8, 3, 2]
-1 1 6 10 6 7 1 1



→ pop all greater element

→ update ans

→ push curr element in stack

∴ An element is touched at max 2 times -

max iterations possible = 2·N

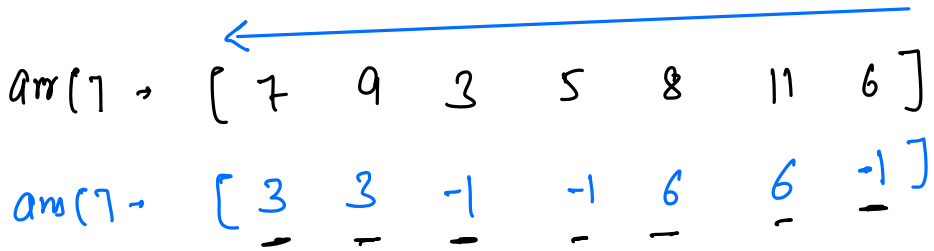
T.C → O(N)

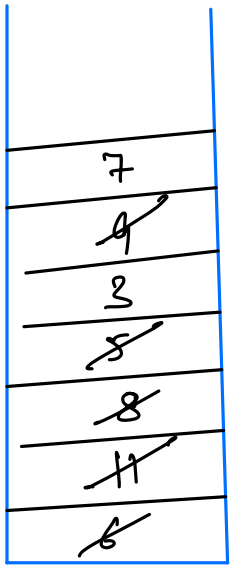
pseudo-code

```
Stack<integer> st;

int ans[N],    fi, ans[i] = -1

for( i = 0; i < N; i++){

    //1. pop all greater elements

     while(st.size()>0 && st.top() ≥ arr[i]){
     [
          st.pop();
     ]
     }

    //2. update the ans

       if( st.size() == 0){
       [
            ans[i] = -1;
       ]
       else {
            ans[i] = st.top();
       [
       }

    //3. push curr element in the stack

        st.push(arr[i]);

}

return ans[];
```

T.C → O(N)
S.C → O(N)

Nearest Smaller Element on R.H.S



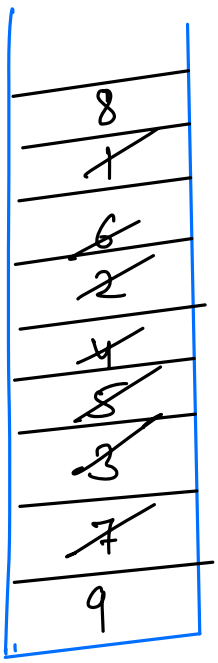


- pop all greater & equal elements
- → update the answer
- → push curr element

Q2) **Nearest Greater Element on the Left.**

arr- [9 7 3 5 4 2 6 1 8]
0 1 2 3 4 5 6 7 8

ans- [-1 9 7 7 5 4 7 6 9]



→ pop all smaller and equal elements

→ update the answer

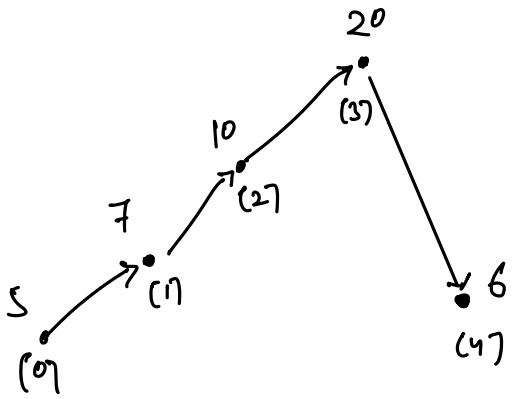
→ push curr element

#code..

```
Stack < integer> st;

int ans[N],    fi, ans[i] = -1

for( i = 0; i < N; i++){

    //1. pop all smaller and equal elements

    while(st.size()>0 && st.top() <= arr[i]){
        st.pop();
    }

    //2. update the ans

    if( st.size() == 0){
        ans[i] = -1;
    }
    else {
        ans[i] = st.top();
    }

    //3. push curr element in the stack

    st.push(arr[i]);
}

return ans[];
```

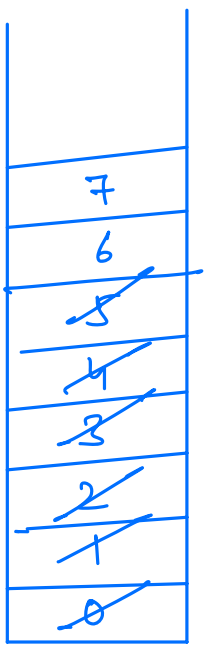
T.C → O(N)
S.C → O(N)

Index of nearest smaller Element on left



arr[] → [4 6 10 11 7 6 3 5]
0 1 2 3 4 5 6 7

ans[] → [-1 0 1 2 1 0 -1 6]



→ pop all greater and equal elements

→ update the answer

→ push curr index

(index in stack)

pseudo-code:

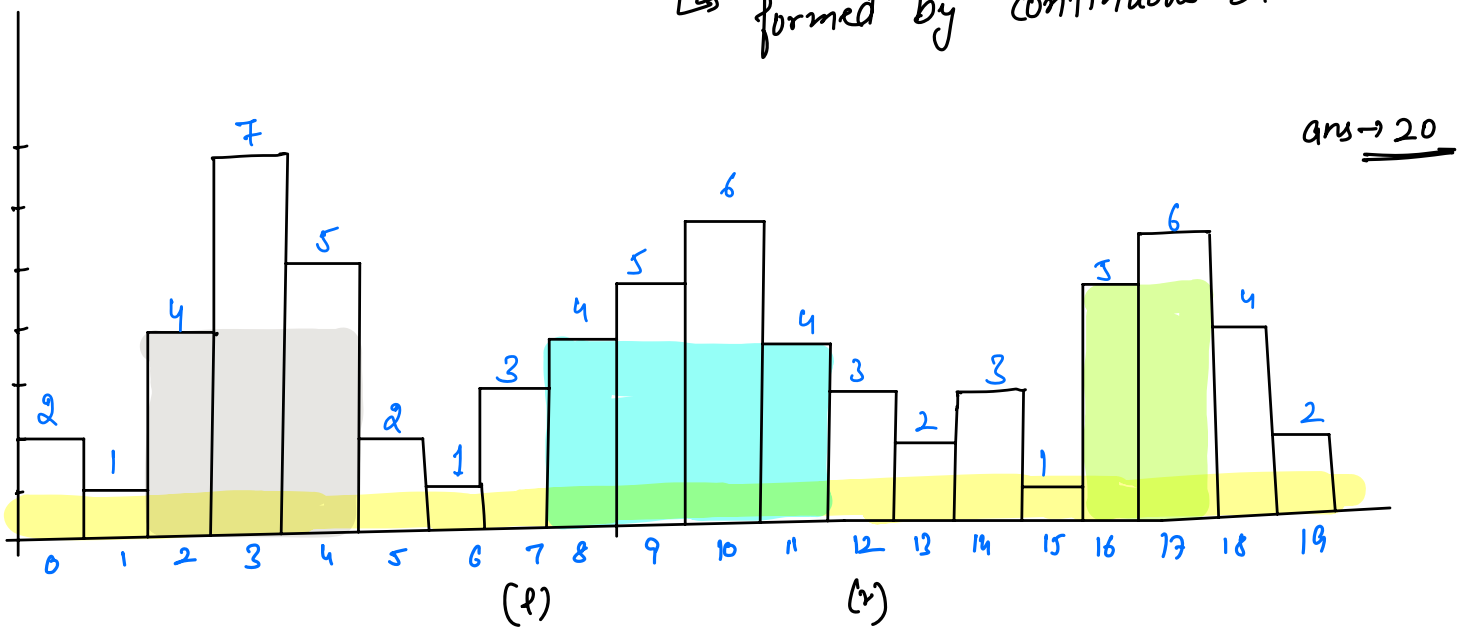
```
Stack < integer > st;

int ans[N],   ∀i, ans[i] = -1

for( i = 0; i < N; i++) {

    //1. pop all greater and equal elements

    while(st.size()>0 && arr[st.top()] ≥ arr[i]) {
        st.pop();
    }

    //2. update the ans

    if( st.size() == 0){
        ans[i] = -1;
    }
    else {
        ans[i] = st.top();
    }

    //3. push curr index in the stack

    st.push(i);

}

return ans[];
```

T.C → O(N)
S.C → O(N)

[# Break.. 10:32 → 10:40]

Q: Find the largest area of rectangle in histogram.

↳ formed by continuous bars.



Max width possible with ht. of current bar = $r - l - 1$

r ⇒ index of nearest smaller element on r.h.s

l ⇒ index of nearest smaller element on l.h.s

idea → for every bar,

find index of nearest smaller element on l.h.s (l)

find index of nearest smaller element on r.h.s (r)

$$\text{Max}_{i=0}^{N-1} \left[arr[i] * (r - l - 1) \right]$$

pseudo code.

```
Create nsl [N] ;   // default value => -1
Create nsr [N] ;   // default value => N

ans = 0
for( i = 0; i < N; i++){
{
        ans = Max( ans, arr[i] * ( nsr[i] - nsl[i] - 1)) ;
}
return ans;
```

ht. * width. → area.

[T.C → O(N)
S.C → O(N)]

Q: Find sum of (max - min) for all subarrays.

arr → [2 (0) 5 (1) 3 (2)]

	max	min	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 → ans.

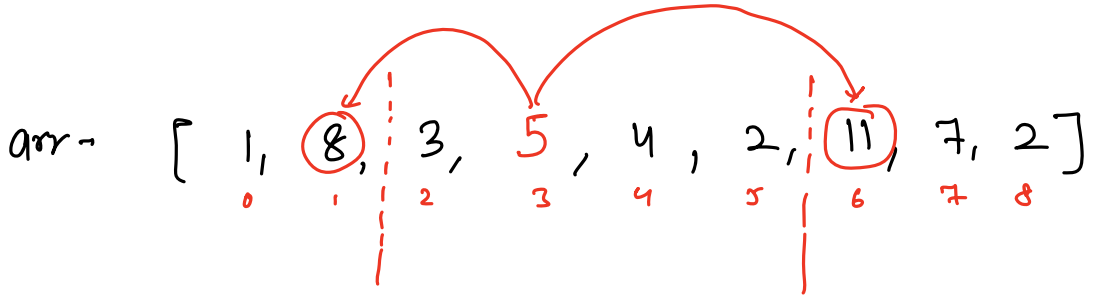
idea-1. → Consider all subarrays. T.C → $O(N^2)$

idea-2. Contribution technique?

$$\sum (max - min)_s$$

$$= \left(\sum max_s\right) - \sum min_s$$

ith element. → In how many subarrays, ith element is maximum?

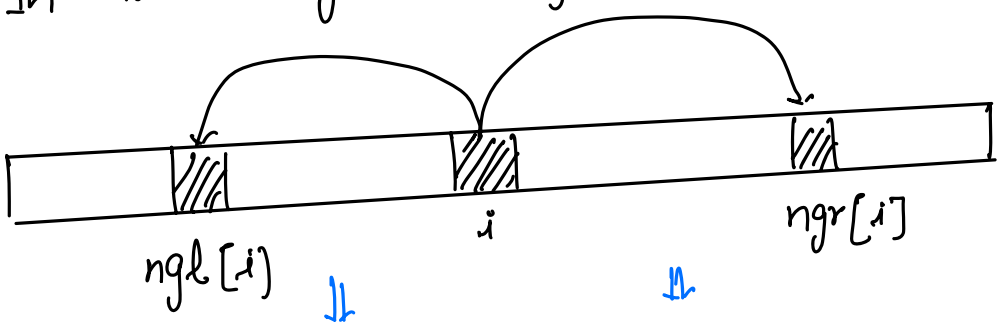


si	ei
2	3
3	4
	5

6 subarrays

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

ith element → In how many subarrays ith element is max?



ngl[i] = options for si

ngr[i] = options for ei

$(i - ngl[i]) * (ngr[i] - i)$

Contribution of arr[i] as max element =

$arr[i] * (i - ngl[i]) * (ngr[i] - i)$

Contribution of arr[i] as min element =

$arr[i] * (i - nsl[i]) * (nsr[i] - i)$

pseudo-code:

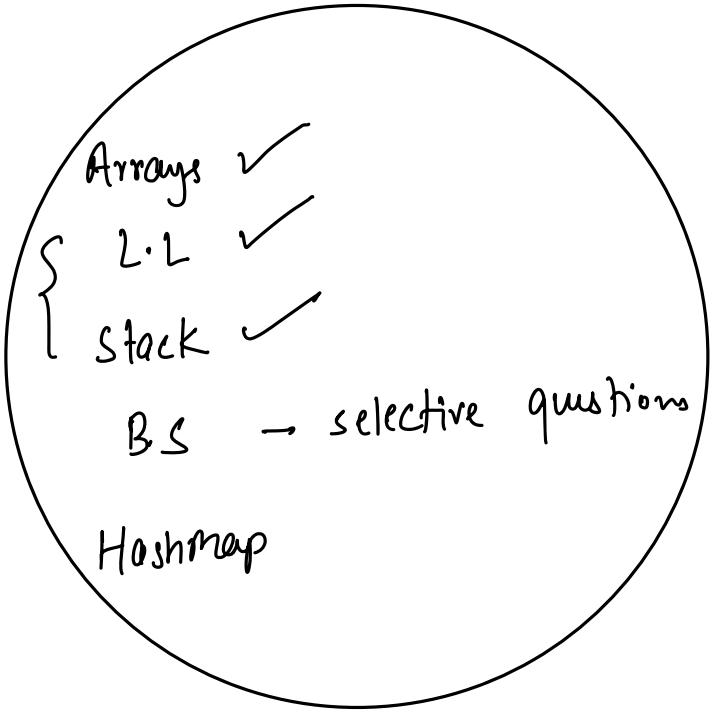
```
//Create nsl[N], nsr[N], ngl[N], ngr[N];

maxs = 0 , mins = 0

for( i = 0; i < N; i++) {
    maxs = maxs + (arr[i] * (i - ngl[i]) * (ngr[i] - i));
    mins = mins + (arr[i] * (i - nsl[i]) * (nsr[i] - i));
}

return maxs - mins ;
```

T.C → O(N)
S.C → O(N)

→ # duplicates.

Target → clear backlogs.

B.M ✗
Maths ✗
Recursion ✗
Sorting



24 hrs. → office
→ personal

1 lecture → 4 assignment problems.

1 week → 3 lectures → 10-12 assignment problems.

10 hours. → [10-12 problems]