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
- → Solve Reverse Polish Notation / Postfix Notation Expression.

2L + Stacks.

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]

	4	5	2	10	3	2
index	0	1	2	3	4	5
ans	X (-1)	4	X (-1)	2	2	X (-1)

arr: [4 6 10 11 7 8 3 5]

	4	6	10	11	7	8	3	5
index	0	1	2	3	4	5	6	7
ans	X (-1)	4	6	10	6	7	X (-1)	3

Idea:- For every element, iterate on its l.h.s. elements & find nearest smaller element on l.h.s.

```
ans[] → -1

for( i=0 ; 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
↓	↓	↓	↓	↓	↓	↓	
~~5~~	~~2~~	2	8	2	~~1~~	1	

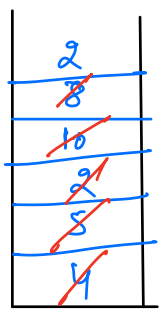
~~5~~	~~2~~	~~8~~	~~10~~	~~6~~	1

⇓

Stack.

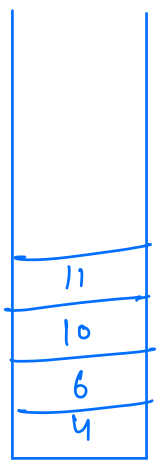
Eg: [4 5 2 10 8 2]

-1 4 -1 2 2 -1



Eg: [4 6 10 11 7 6 3 5]

ans → [-1 4 6 10 6 4 -1 3]



→ pop all greater & equal elements.

→ update ans.

→ push curr element in st.

pseudo-code:

```
Stack<Integer> st , ans[N]

for(i=0; i<N ; i++){
        while (!st.isEmpty() && st.peek() ≥ arr[i]){
                st.pop()
        }
        if ( st.isEmpty() ) { ans[i] = -1 }
        else    { ans[i] = st.peek() }

        st.push (arr[i]);
}
return ans[];
```

T.C → O(N)
S.C → O(N)

Q2: Nearest Smaller Element on r.h.s.

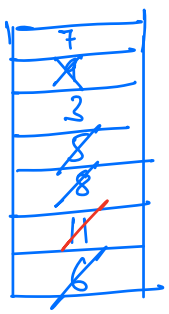
↓

① iterate from N-1 to 0

② Reverse the array & apply same method.

arr[]: [7 9 3 5 8 11 6]

7	9	3	5	8	11	6
3	3	-1	-1	6	6	-1



→ pop all greater & equal elements.

→ update ans.

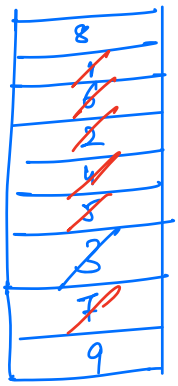
→ push curr element in st.

Q3) Nearest Greater Element on the Left.

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

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

→ pop all smaller & equal elements.
→ update ans.
→ push curr element in st.

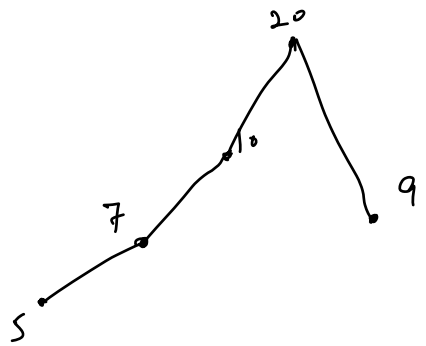



```
Stack<Integer> st , ans[N]

for(i=0; i<N ; i++){
        while (!st.isEmpty() && st.peek() ≤ arr[i]){
                st.pop()
        }
        if ( st.isEmpty() ) { ans[i] = -1 }
        else { ans[i] = st.peek() }

        st.push(arr[i]);
}
return ans[];
```

Q4) Nearest Greater Element on the Right.

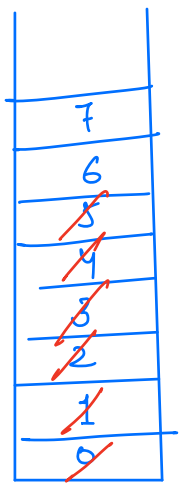
↳ {traverse from N-1 to 0}

→ index of next smaller element on left / right.



Eg.: arr: [4 6 10 11 7 6 3 5]
0 1 2 3 4 5 6 7

(indices of nearest smaller element on l.h.s) ans: [-1 0 1 2 1 0 -1 6]



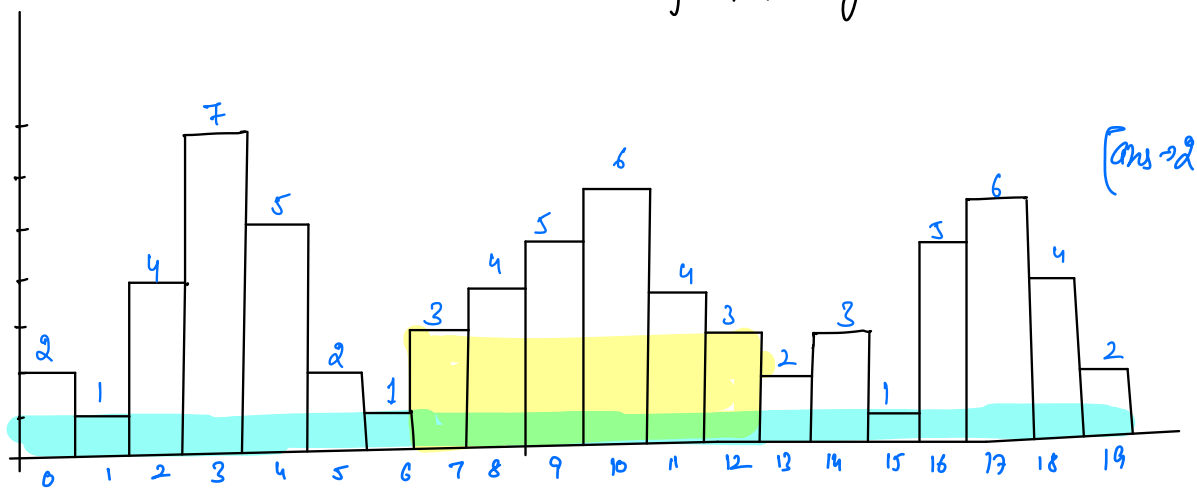
→ pop all greater & equal elements.
→ update ans.
→ push curr index in st.

pseudo-code.

```
Stack<Integer> st , ans[N];

for (i=0; i<N; i++) {
        while( !st.isEmpty() && arr[st.peek()] ≥ arr[i]){
            st.pop();
        }
        if (st.isEmpty() ) { ans[i] = -1 }
        else { ans[i] = st.peek() ; }

        st.push (i);
}
```

[T.C → O(N)
S.C → O(N)]

Q: Find the largest area of rectangle in histogram.
↳ formed by continuous bars.



[ans → 20]

2 20 12 7 10 8

[Max width possible with ht of current bar = r − l − 1.]

r → Nearest smaller on r.h.s

l → Nearest smaller on l.h.s

idea-1 for every bar,
traverse & find index of nearest smaller element on l.h.s (l) and
traverse & find index of nearest smaller element on r.h.s (r)

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

T.C → $O(N^2)$
↓
$O(N)$?

Step-1. Create nsl [N] (default → -1)

Step-2. Create nsr [N] (default → N)

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

[T.C → O(N)
S.C → O(N)]

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

$\underset{0}{2} \quad \underset{1}{5} \quad \underset{2}{3}$

	max	min	max - min
0-0	2	2	0
0-1	5	2	3
0-2	5	2	3
1-1	5	5	0
1-2	5	3	2
2-2	3	3	0
			8

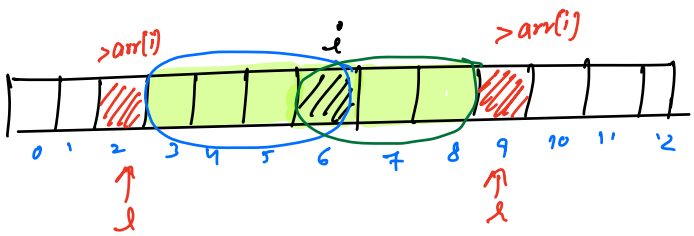
idea-1. Consider all subarrays. TC → $O(N^2)$

idea-2. Contribution?

$$\sum max_s - min_s$$

$$\underbrace{\sum max_s}_{\Downarrow} - \sum min_s$$

i^{th} element → in how many subarrays i^{th} element will be max.



subarray → st, end.

arr[i] → You need to find index of nearest greater element on l.h.s (l) & index of nearest greater element on r.h.s (r).

3-6
3-7
3-8
4-6
4-7
4-8
5-6
5-7
5-8
6-6
6-7
6-8

Contribution of arr[i] → $arr[i] * (i-l) * (r-i)$

↑ option for s.i

↑ options for e.i

pseudo-code.

default value

-1 N

```
ngl[N], ngr[N], nsl[N], nsr[N]

max_s = 0
for( i = 0; i < N; i++) {
        max_s += ( arr[i] * (i - ngl[i]) * (ngr[i] - i) )
}

min_s = 0
for( i = 0; i < N; i++) {
        min_s += ( arr[i] * ( i - nsl[i]) * (nsr[i] - i) );
}

return max_s - min_s ;
```

T.C → O(N)
S.C → O(N)

Reverse Polish Notation / Postfix Notation

infix. { 8 * 5 + 4 — op1 operator op2

prefix. { + * 8 5 4 — operator op1 op2

postfix. { 8 5 * 4 + — op1 op2 operator.

A + B.

prefix.
+ A B

postfix.
A B +

→ 10 + 3 * 4 - 7

10 3 4 * + 7 -

~~10~~	~~3~~	~~4~~	~~12~~	~~22~~	~~7~~	15

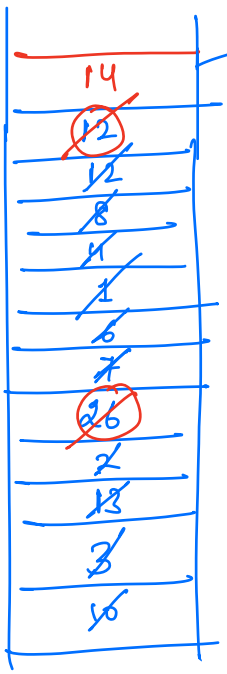
Stack

op2 → 7
op1 → 22

10 3 + 4 * 7 - X
13 4 * 7 -
52 7 -
45.

RPN / postfix. →

[10 3 + 2 * 7 6 - 4 8 + * -]



ans = 14.

op2 → ~~12~~ 12

op1 → ~~1~~ 26

traverse the expression (array)

- operand (int) → push in the stack.
- operator.

```
op2 = st.pop();
op1 = st.pop();
st.push( op1 operator op2)
```