Agenda

i) Design min Stack

ii) Nearest Smaller **
- ↳ on left
- ↳ on right

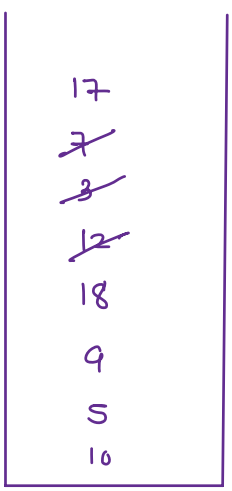
iii) Largest Area histogram

Q.1 Design a Stack that supports push, pop, top and min functions in O(1) time.

10 5 9 18 12 3 7 17

MinStack st = new MinStack();

st.push(10)
st.push(5)
st.push(9)
st.push(18)
st.push(12)
st.push(3)
st.push(7)
st.min() → 3
st.pop()
st.pop()
st.min() → 5
st.top() → 12
st.pop()
st.push(17)
st.top() → 17



class MinStack {

void push(int x)

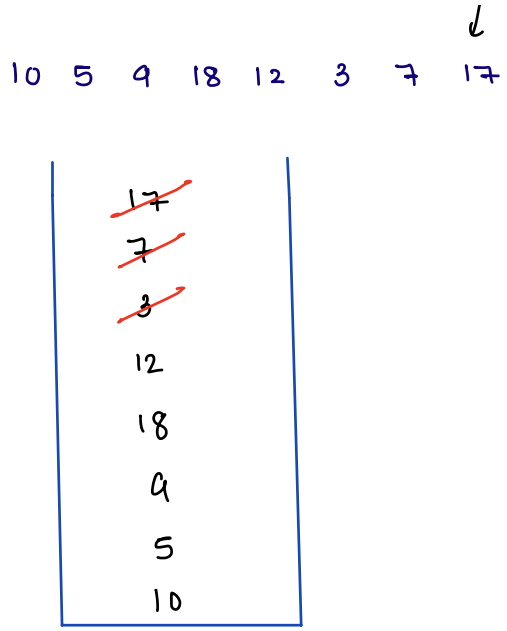
void pop()

O(1) int top()

int getmin()

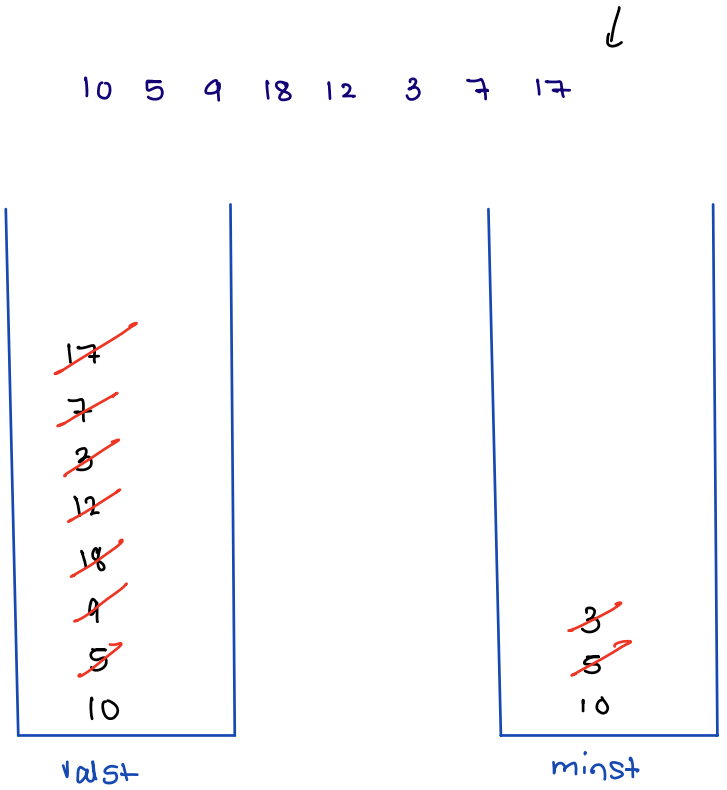
}

3



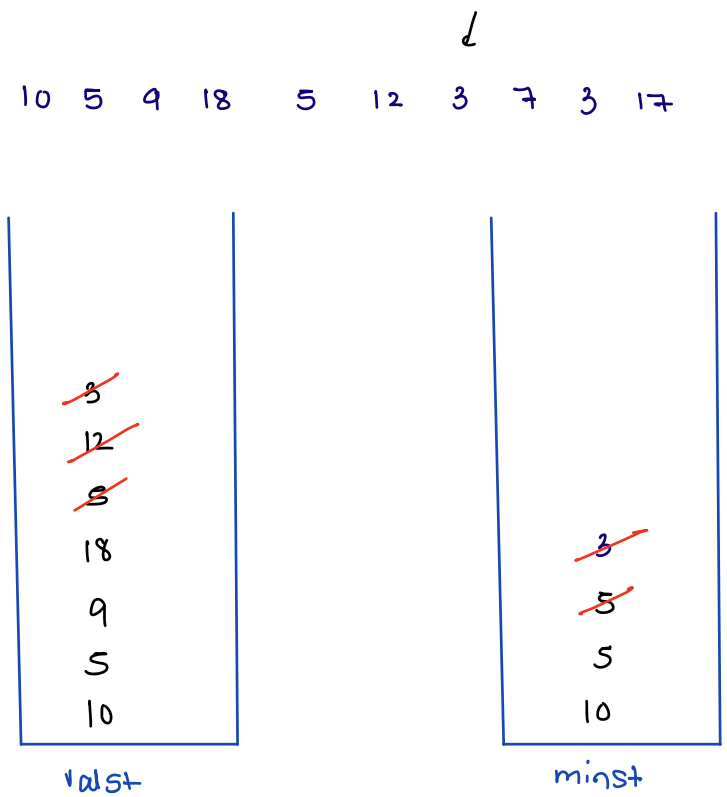
min = ∞ 10 5 3

idea: we need separate stack to manage min values.



x <= minst.peek()

→ minst.push(x);


i
10 5 9 18 5 12 3 7 3 17
5
12
5
18
9
5
10
valst
3
5
5
10
minst

class MinStack {

→ two stack : valst, minst

void push (int x) {

→ manage size=0 condition

→ add x in valst

→ add x in minst (x <= minst.peek())

}

void pop () {

→ manage size=0 condition

→ temp= pop from valst

→ pop from minst (temp == minst.peek())

}

int top () {

→ manage size=0 condition

→ return valst.peek()

}

int getmin() {

→ manage size=0 condition

→ return minst.peek()

}

}

```java
class Solution {
    Stack<Integer>st = new Stack<>();
    Stack<Integer>minSt = new Stack<>();

    public void push(int x) {
        if(st.size() == 0) {
            st.push(x);
            minSt.push(x);
        }
        else {
            st.push(x);

            if(x < minSt.peek()) {
                minSt.push(x);
            }
        }
    }

    public void pop() {
        if(st.size() == 0) {
            return;
        }
        else {
            int top = st.pop();

            if(top == minSt.peek()) {
                minSt.pop();
            }
        }
    }

    public int top() {
        if(st.size() == 0) {
            return -1;
        }
        else {
            return st.peek();
        }
    }

    public int getMin() {
        if(st.size() == 0) {
            return -1;
        }
        else {
            return minSt.peek();
        }
    }
}
```

Q.2 **Nearest / next smaller on Left.**

for every element find out the value of nearest smaller on left.
↳ in terms of distance

A = [10 16 5 9 12 8 25 7 13]

ans -1 10 -1 5 9 5 8 5 7

A = [18 3 13 14 5 24 4]

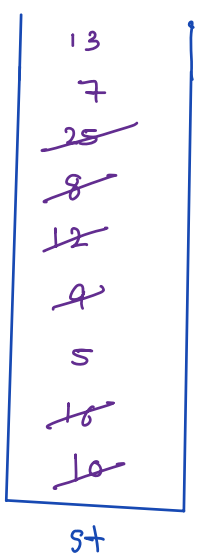
ans -1 -1 3 13 3 5 3

Expected TC: O(n)

What ?

A = [10 16 5 9 12 8 25 7 13]

ans -1 10 -1 5 9 5 8 5 7

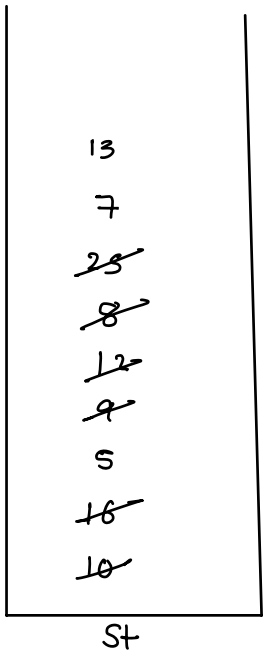


A = [10 16 5 9 12 8 25 7 13]

ans = -1 10 -1 5 9 5 8 5 7

```
ans[n]
ans[0] = -1, st.push(A[0]);

for ( i → 1 to A.length-1 ) {
  while (st.size() >0 && st.peek() >= A[i]) {
        st.pop();
  }
  if (st.size() ==0) {
        ans[i] = -1;
  }
  else {
        ans[i] = st.peek();
  }
  st.push(A[i]);
}
```



TC: O(n)

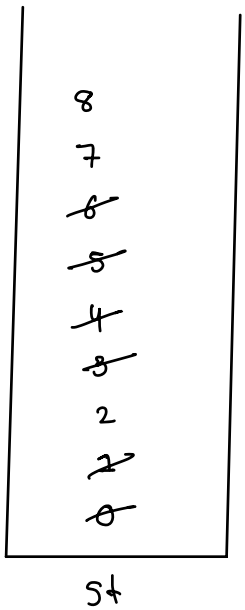
SC: O(n)

Nearest / next smaller on left (Index based)

for every element find out the index of next nearest smaller on left.

	0	1	2	3	4	5	6	7	8
A =	[10	16	5	9	12	8	25	7	13]
ans	-1	0	-1	2	3	2	5	2	7

```
for ( i → 1 to A.length-1 ) {
    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);
}
```



Q.3 Nearest / next smaller on right.

for every element find the value of nearest / next small on right

A = [10 16 5 9 12 8 25 7 3]

ans 5 5 3 8 8 7 7 3 -1

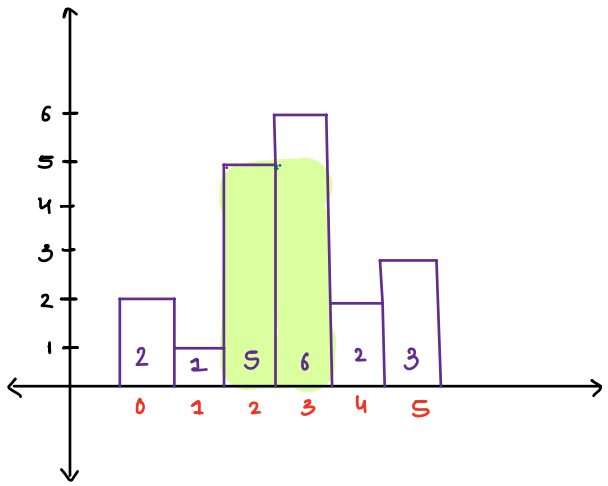
same logic as that of next smaller left, the only change is direction of travelling should be right to left.

Nearest / next smaller on right (Index based)

for every element find the idx of nearest / next small on right

8.4 Largest area histogram

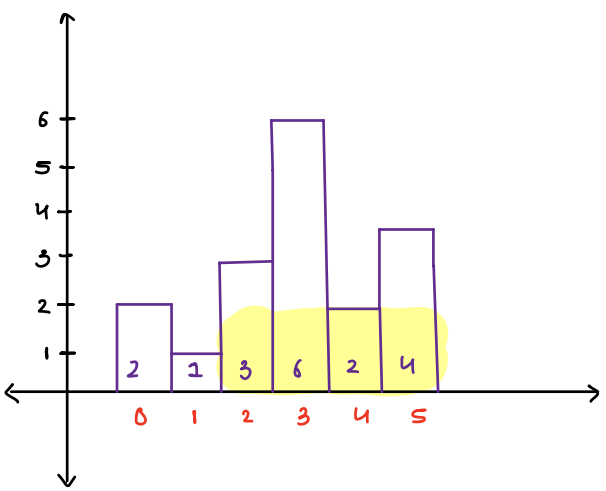
A = [2 1 5 6 2 3]



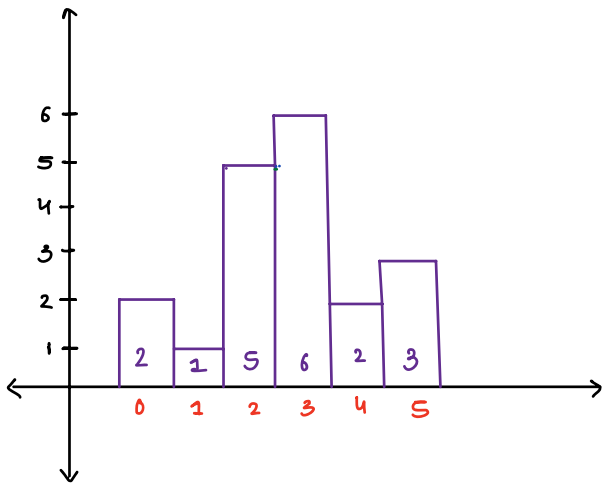
Return the area of Largest rectangle possible ?

ans = 5 * 2 = 10

A = [2 1 3 6 2 4]

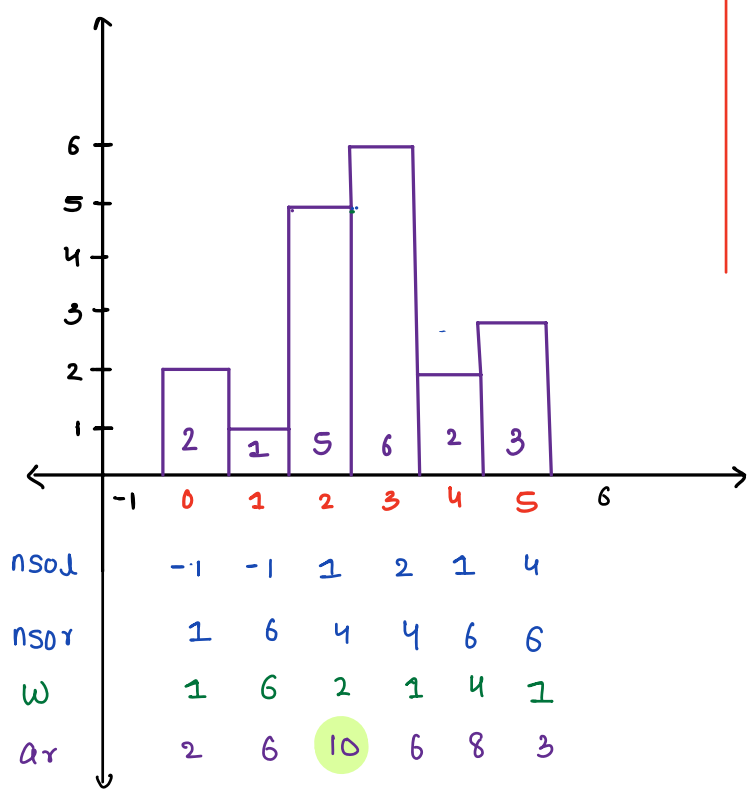


ans = 2 * 4 = 8



i	h	w	ar
0	2	1	2
1	1	6	6
2	5	2	10
3	6	1	6
4	2	4	8
5	3	1	3

```
h = A[i]
w = nsor[i] - nsol[i] - 1
ar = h * w
if (ar > max) {
    max = ar;
}
```



nsol	-1	-1	1	2	1	4
nsor	1	6	4	4	6	6
w	1	6	2	1	4	1
ar	2	6	10	6	8	3

Doubts

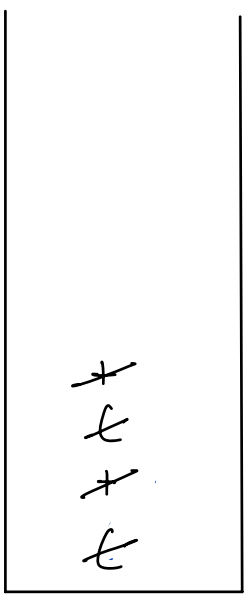
redundant

((a+b)) → 1 (yes redundant brack is present)

(a+(a+b)) → 0 (no redundant brackets)

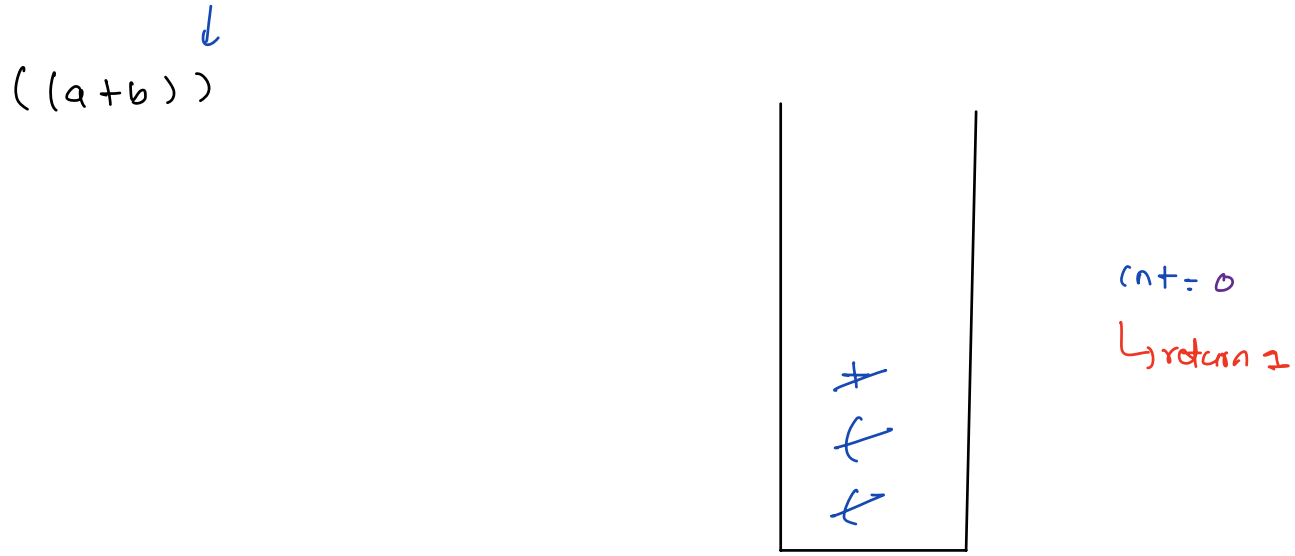
str: (a+(a+b))

```
for (i -> 0 to str.length()-1) {
  if (ch == '(') {
      st.push(ch);
  }
  else if (ch == '+' || ch == '-' ||
           ch == '*' || ch == '/')
      st.push(ch);
  }
  else if (ch == ')') {
      // keep count of content (oprs) till opening
      int cnt = 0;
      while (st.peek() != '(') {
          cnt++;
          st.pop();
      }
      st.pop();
      if (cnt == 0) { return 1; }
  }
}
return 0;
```



st (char)

cnt = 0, 2


((a+b))
cnt=0
return 2