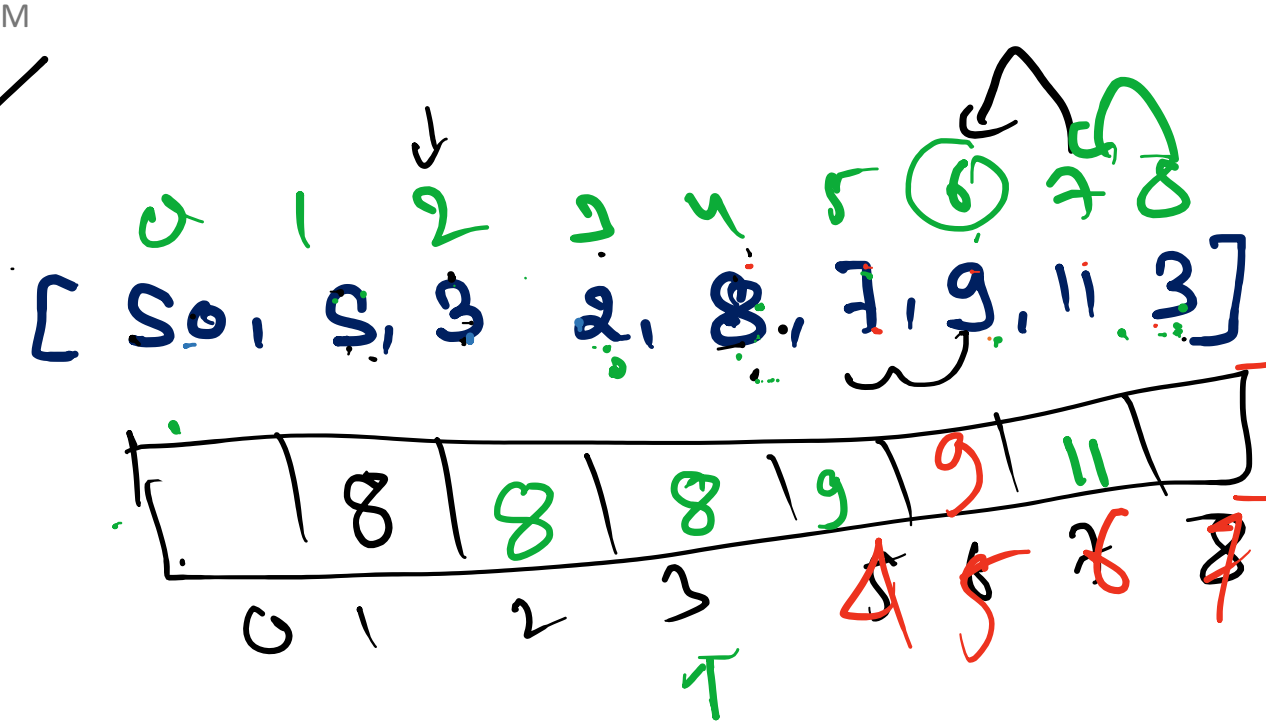


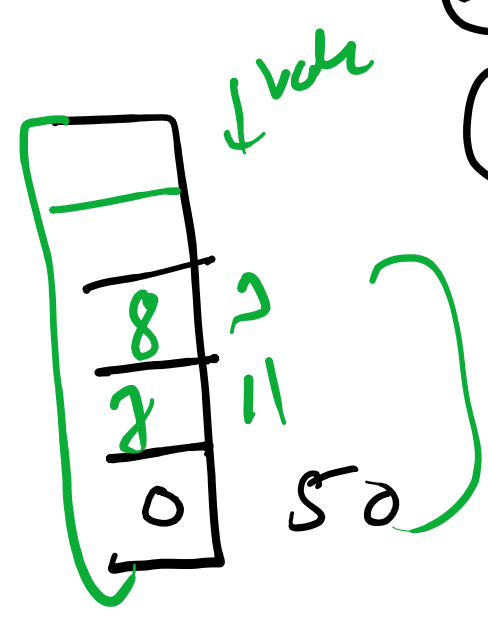
20 ← 801000  
2400



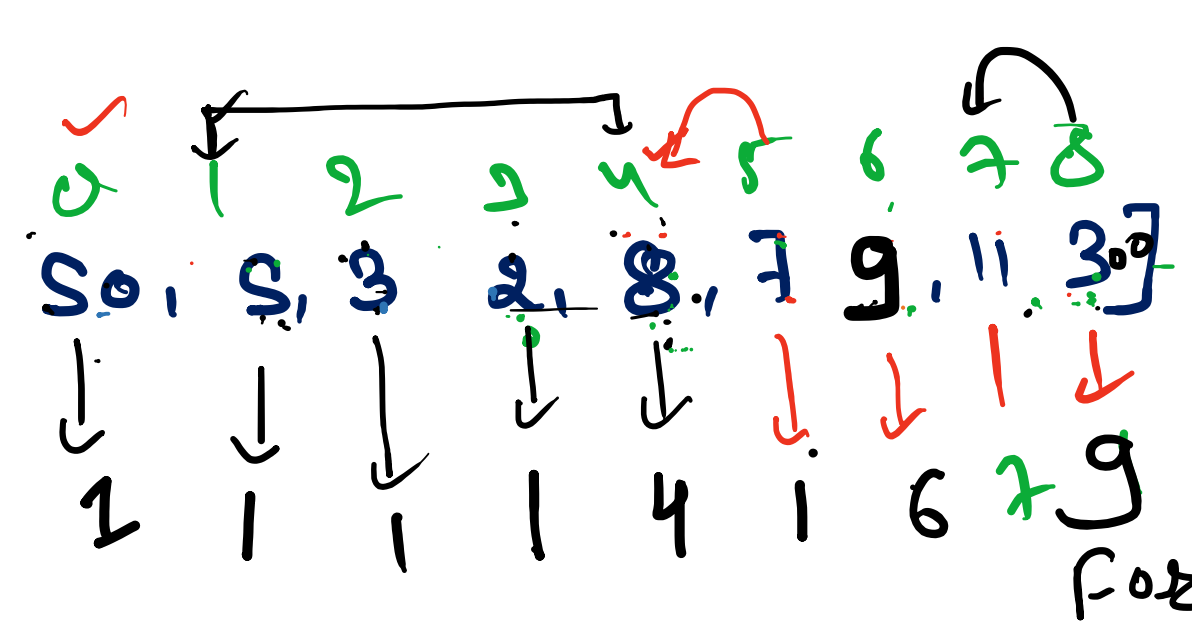
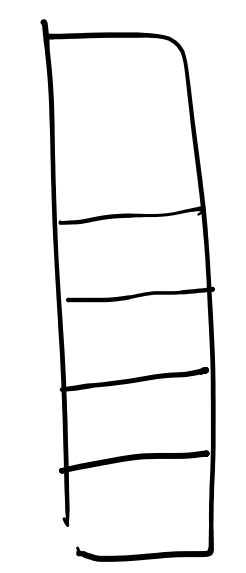
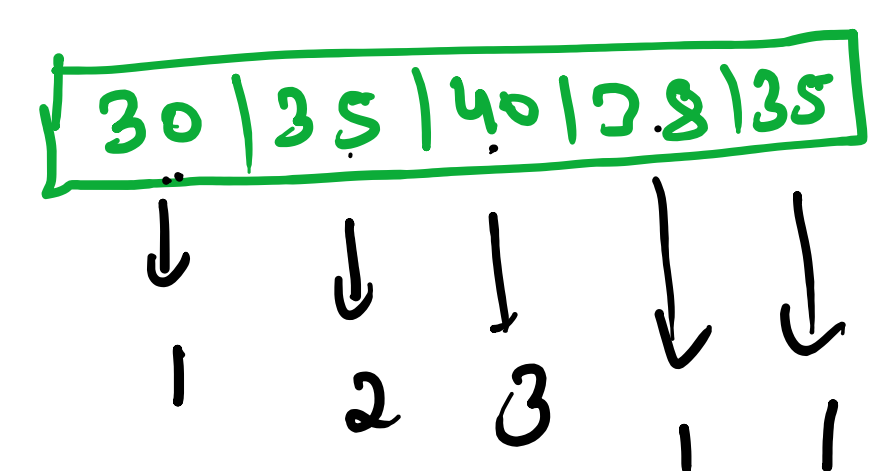
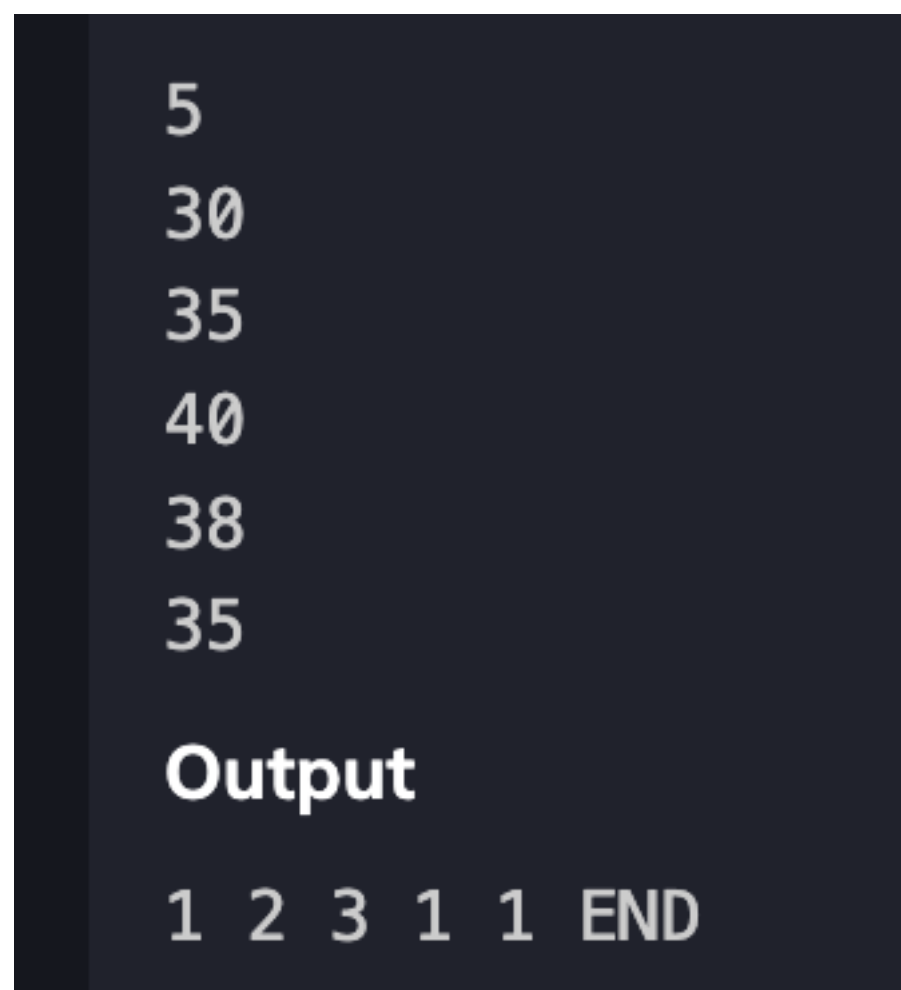
$O(n)$  ✓

- ① next great
- ② pre gr
- ③ next smol
- ④ pre smol

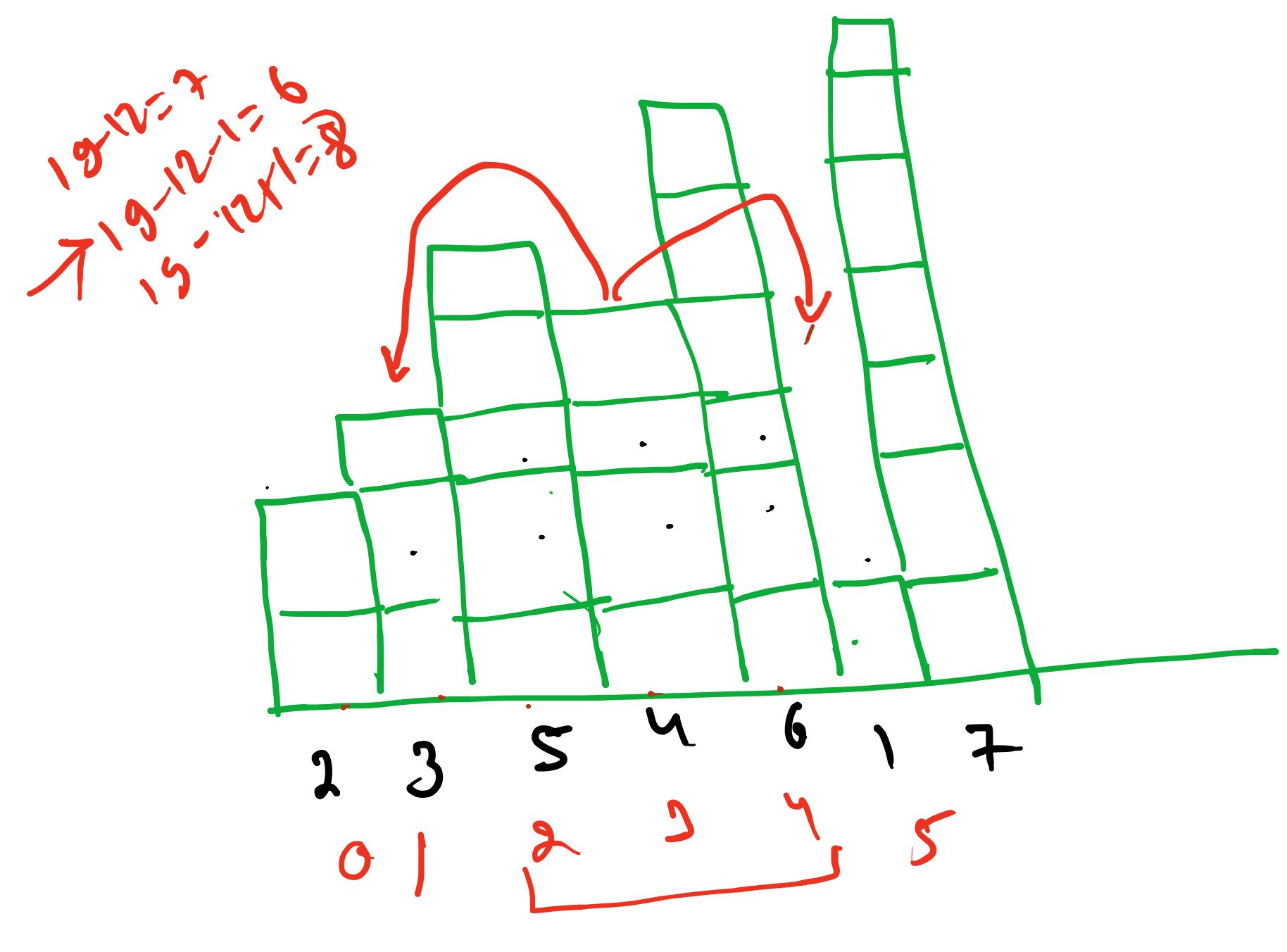
```
for (ci=0; i<arr.length; i++) {  
    while (!st.isEmpty() && arr[i] > arr[st.peek()]) {  
        ans[st.pop()] = arr[i];  
    }  
    st.push(i);  
}
```



```
while (!st.isEmpty() && arr[st.peek()] > arr[st.pop()]) {  
    ans[st.pop()] = arr[st.peek()];  
}
```

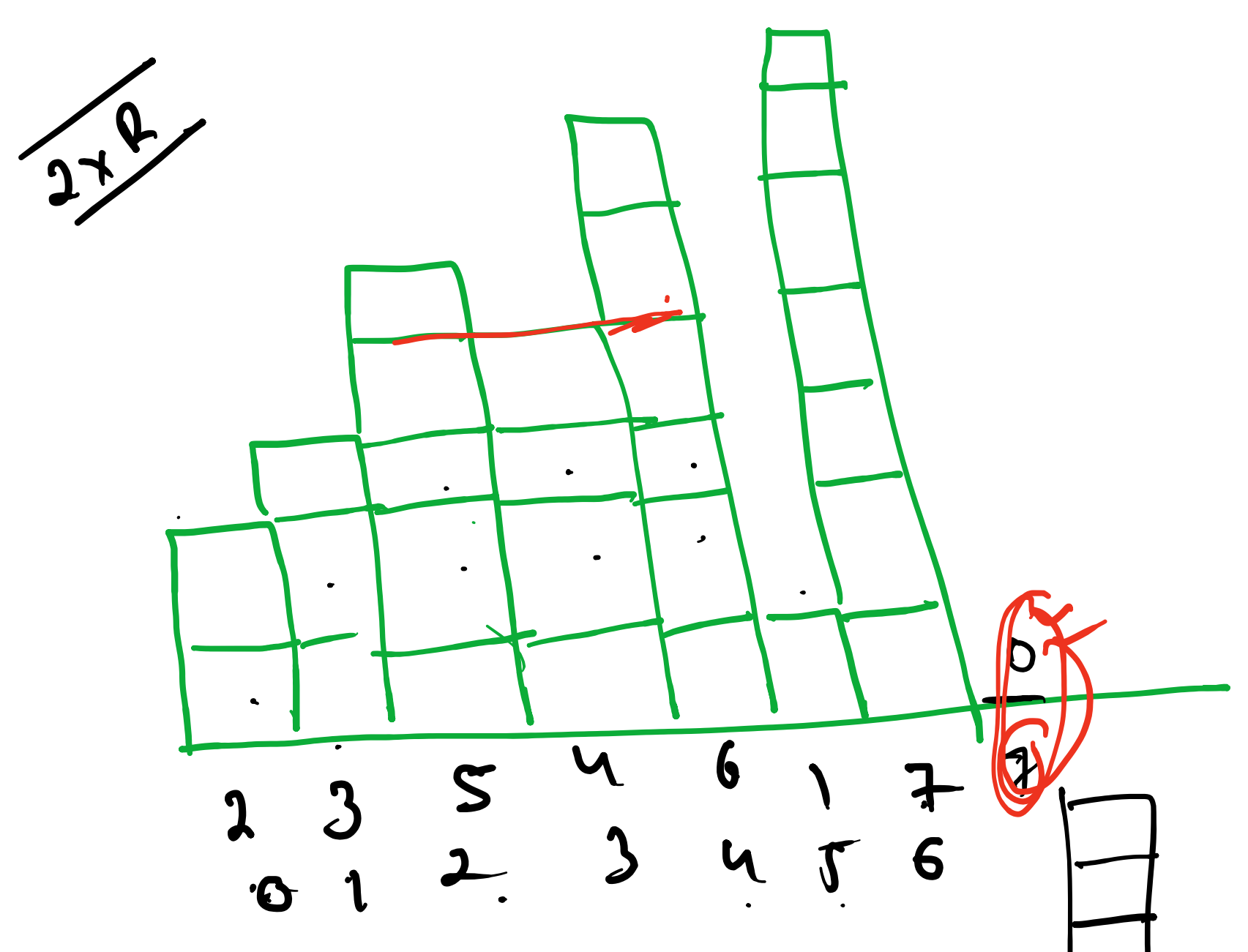


```
for (ci=0; i<arr.length; i++) {  
    while (!st.isEmpty() && arr[i] > arr[st.peek()]) {  
        ans[st.pop()] = arr[st.peek()];  
    }  
    if (st.isEmpty()) {  
        ans[i] = i+1;  
    }  
    st.push(i);  
}
```



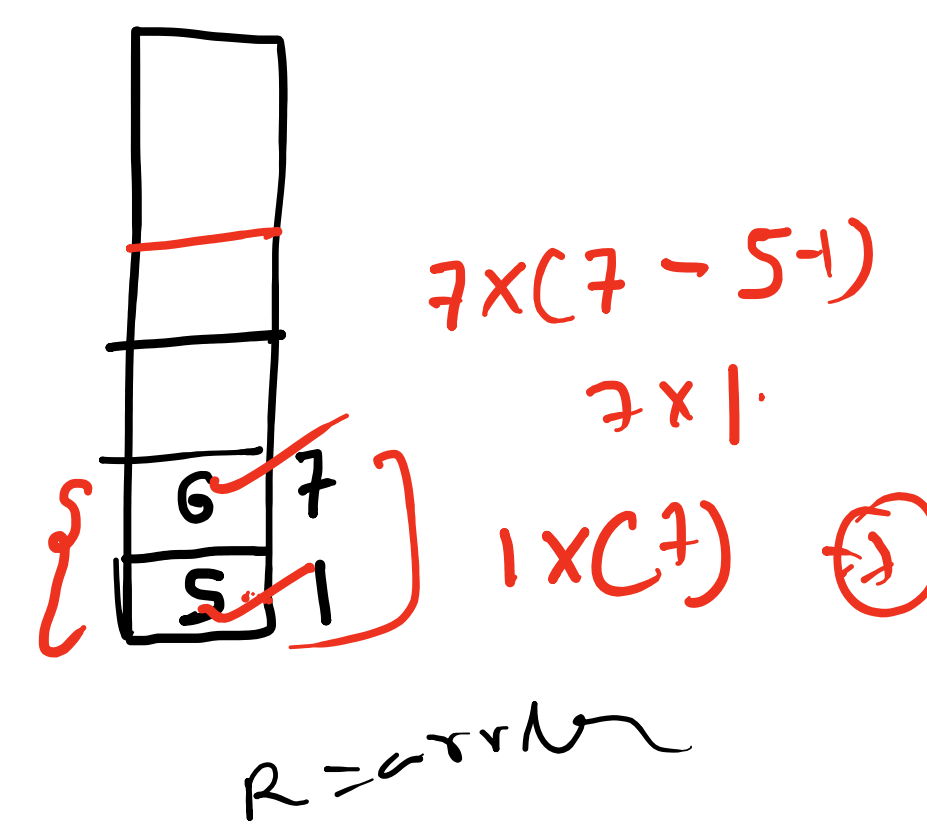
[2, 3, 5, 4, 6, 1, 7]

- 1 5
- 2 4
- 3 5
- 4 12
- 5 6
- 6 7
- 7 7

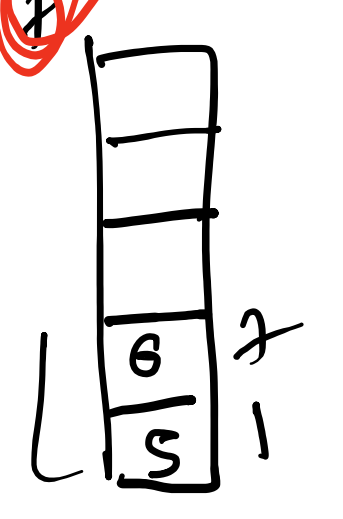


[2, 3, 5, 4, 6, 1, 7]

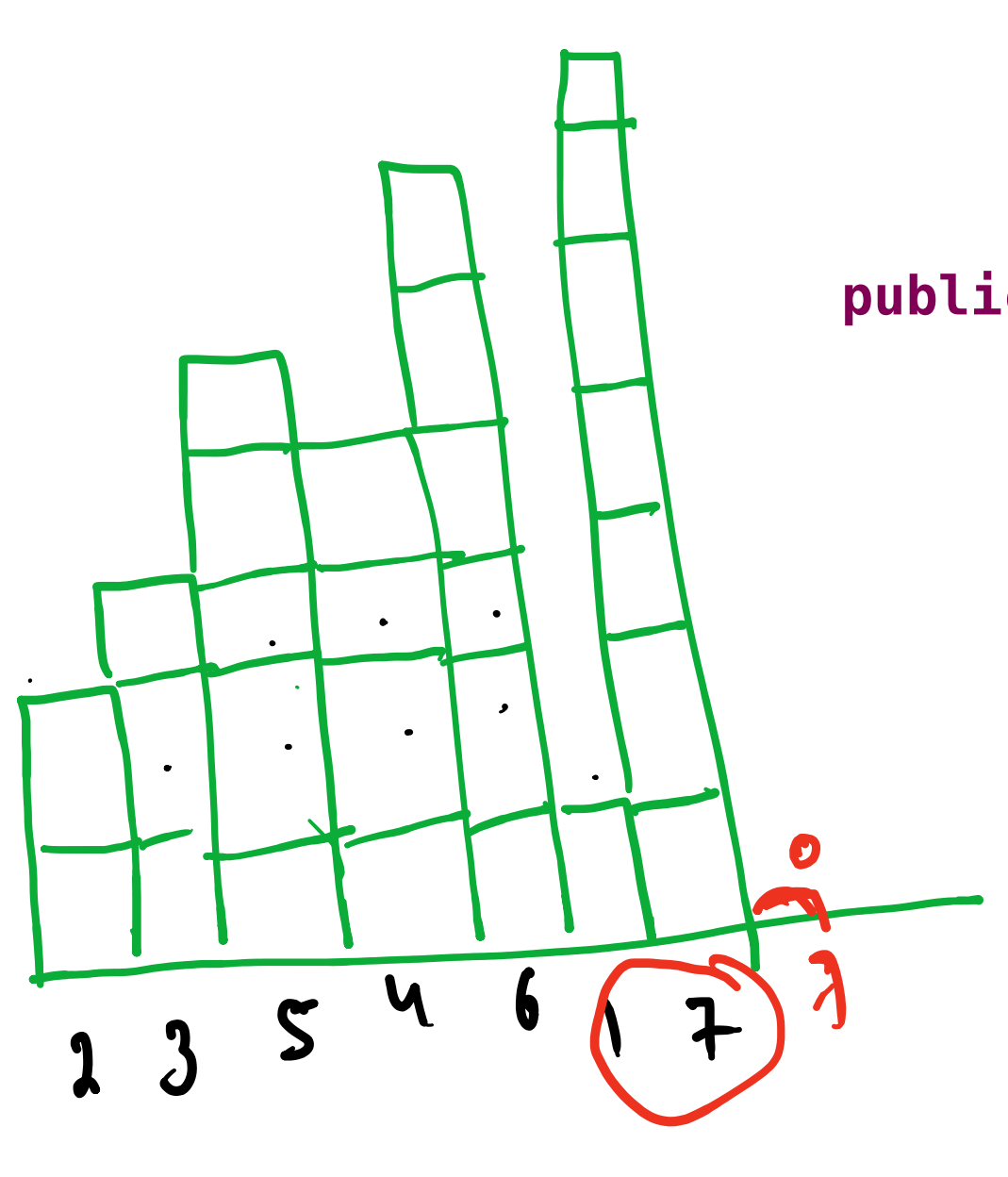
- 5 x (3-1-1) = 5
- 6 x (5-3-1) = 6
- 4 x (5-1-1) = 12
- 3 x (5-0-1) = 12
- 2 x (5) = 10



[2, 3, 5, 4, 6, 1, 7]



```
for (ci=0; i<arr.length; i++) {  
    while (!st.isEmpty() && arr[i] > arr[st.peek()]) {  
        int h = arr[st.pop()];  
        if (st.isEmpty()) {  
            ans = max(ans, h*x);  
        }  
        else {  
            L = st.peek();  
            ans = max(ans, h*(R-L-1));  
        }  
    }  
    st.push(i);  
}
```



```
public static int largestArea(int[] heights) {  
    Stack<Integer> st = new Stack<>();  
    int area = 0;  
    for (int i = 0; i < heights.length; i++) {  
        while (!st.isEmpty() && heights[i] < heights[st.peek()]) {  
            int h = arr[st.pop()];  
            if (st.isEmpty()) {  
                ans = max(ans, h*x);  
            }  
            else {  
                L = st.peek();  
                ans = max(ans, h*(R-L-1));  
            }  
        }  
        st.push(i);  
    }  
}
```

[2, 3, 5, 4, 6, 1, 7]

