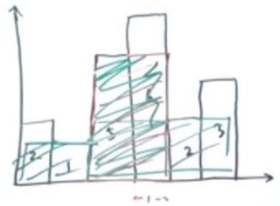
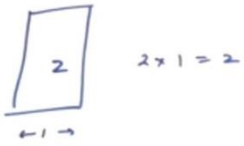
Largest Rectangle in Histogram

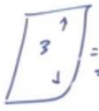


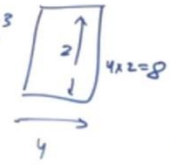
2 × 4 = 8 ↑↑

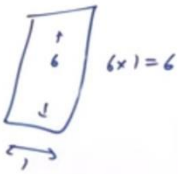
5 × 2 = 10

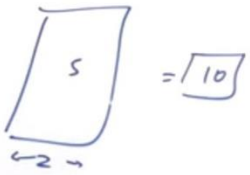
Largest Rectangle in Histogram











Largest Rectangle in Histogram

width x a[i]

(5 − 2 + 1) x a[i]

(right smaller − left smaller + 1) x a[i]

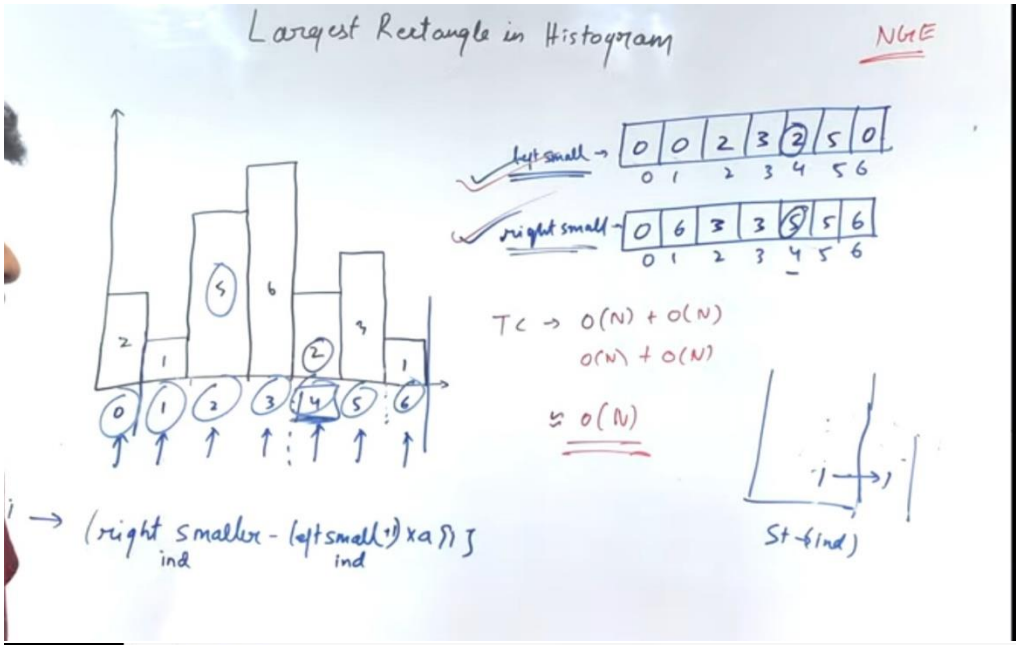
Largest Rectangle in Histogram

width x a[i]

(5 − 2 + 1) x a[i]

(right smaller − left smaller + 1) x a[i]

≈ O(N²)




```cpp
class Solution {
public:
    int largestRectangleArea(vector<int>& heights) {
        int n = heights.size();
        stack<int> st;
        int leftSmall[n], rightSmall[n];
        for(int i = 0;i<n;i++) {
            while(!st.empty() && heights[st.top()] >= heights[i]) {
                st.pop();
            }

            if(st.empty()) leftSmall[i] = 0;
            else leftSmall[i] = st.top() + 1;
            st.push(i);
        }

        // clear the stack to be re-used
        while(!st.empty()) st.pop();

        for(int i = n-1;i>=0;i--) {
            while(!st.empty() && heights[st.top()] >= heights[i]) {
                st.pop();
            }

            if(st.empty()) rightSmall[i] = n-1;
            else rightSmall[i] = st.top() - 1;

            st.push(i);
        }

        int maxA = 0;
        for(int i = 0;i<n;i++) {
            maxA = max(maxA, heights[i] * (rightSmall[i] - leftSmall[i] + 1));
        }
        return maxA;
    }
};
```

TUF

```cpp
class Solution {
public:
    int largestRectangleArea(vector<int>& heights) {
        int n = heights.size();
        stack<int> st;
        int leftSmall[n], rightSmall[n];
        for(int i = 0;i<n;i++) {
            while(!st.empty() && heights[st.top()] >= heights[i]) {
                st.pop();
            }

            if(st.empty()) leftSmall[i] = 0;
            else leftSmall[i] = st.top() + 1;
            st.push(i);
        }

        // clear the stack to be re-used
        while(!st.empty()) st.pop();

        for(int i = n-1;i>=0;i--) {
            while(!st.empty() && heights[st.top()] >= heights[i]) {
                st.pop();
            }

            if(st.empty()) rightSmall[i] = n-1;
            else rightSmall[i] = st.top() - 1;

            st.push(i);
        }

        int maxA = 0;
        for(int i = 0;i<n;i++) {
            maxA = max(maxA, heights[i] * (rightSmall[i] - leftSmall[i]));
        }
        return maxA;
    }
};
```

Line no 33 can be written on line 27 to reduce one more iteration

TUF