Given an array of integers heights representing the histogram's bar height where the width of each bar is 1, return *the area of the largest rectangle in the histogram*.

**Example 1:**

A graph showing the number of the same number

Description automatically generated with medium confidence

**Input:** heights = [2,1,5,6,2,3]

**Output:** 10

**Explanation:** The above is a histogram where width of each bar is 1.

The largest rectangle is shown in the red area, which has an area = 10 units.

**Example 2:**

A diagram of a number

Description automatically generated with medium confidence

**Input:** heights = [2,4]

**Output:** 4

Java code:

class Solution {

    public int largestRectangleArea(int[] heights) {

        int[] left = leftsmall(heights);

        int[] right = rightsmall(heights);

        int max = 0;

        for(int i =0;i<heights.length;i++){

            int mul= right[i]-left[i]+1;

            mul = mul\*heights[i];

            max = Math.max(mul,max);

        }

        return max;

    }

    public int[] leftsmall(int[] heights){

        Stack<Integer> st = new Stack<>();

        int[] arr = new int[heights.length];

        for(int i =0;i<heights.length;i++){

            int num = heights[i];

            while(!st.isEmpty() && num<=heights[st.peek()]){

                st.pop();

            }

            if(st.isEmpty()){

                arr[i] = 0;

            }

            else{

                arr[i] = st.peek()+1;

            }

            st.push(i);

        }

        return arr;

    }

        public int[] rightsmall(int[] heights){

        Stack<Integer> st = new Stack<>();

        int[] arr = new int[heights.length];

        for(int i =heights.length-1;i>=0;i--){

            int num = heights[i];

            while(!st.isEmpty() && num<=heights[st.peek()]){

                st.pop();

            }

            if(st.isEmpty()){

                arr[i] = heights.length-1;

            }

            else{

                arr[i] = st.peek()-1;

            }

            st.push(i);

        }

        return arr;

    }

}

Finding the left small and right small indexes:  
            int mul= right[i]-left[i]+1;

            mul = mul\*heights[i];

            max = Math.max(mul,max);

formula : (right -left +1)\* currentnum