Given a rows x cols binary matrix filled with 0's and 1's, find the largest rectangle containing only 1's and return *its area*.

**Example 1:**

A grid of numbers and symbols

Description automatically generated

**Input:** matrix = [["1","0","1","0","0"],["1","0","1","1","1"],["1","1","1","1","1"],["1","0","0","1","0"]]

**Output:** 6

**Explanation:** The maximal rectangle is shown in the above picture.

**Example 2:**

**Input:** matrix = [["0"]]

**Output:** 0

**Example 3:**

**Input:** matrix = [["1"]]

**Output:** 1

**Constraints:**

* rows == matrix.length
* cols == matrix[i].length
* 1 <= row, cols <= 200
* matrix[i][j] is '0' or '1'.

class Solution {

    public int maximalRectangle(char[][] matrix) {

        int hi[] = new int[matrix[0].length];

        int max = 0;

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

            for(int j =0;j<matrix[0].length;j++){

                if(matrix[i][j]=='1'){

                    hi[j]++;

                }

                else{

                    hi[j]=0;

                }

            }

            int[] left = leftmin(hi);

            int[] right = rightmin(hi);

            for(int k = 0;k<hi.length;k++){

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

                mul = mul\*hi[k];

                max = Math.max(mul,max);

            }

        }

        return max;

    }

    public int[] leftmin(int a[]){

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

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

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

            int num = a[i];

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

                st.pop();

            }

            if(st.isEmpty()){

                arr[i] = 0;

            }

            else{

                arr[i] = st.peek()+1;

            }

            st.push(i);

        }

        return arr;

    }

    public int[] rightmin(int a[]){

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

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

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

            int num = a[i];

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

                st.pop();

            }

            if(st.isEmpty()){

                arr[i] = a.length-1;

            }

            else{

                arr[i] = st.peek()-1;

            }

            st.push(i);

        }

        return arr;

    }

}