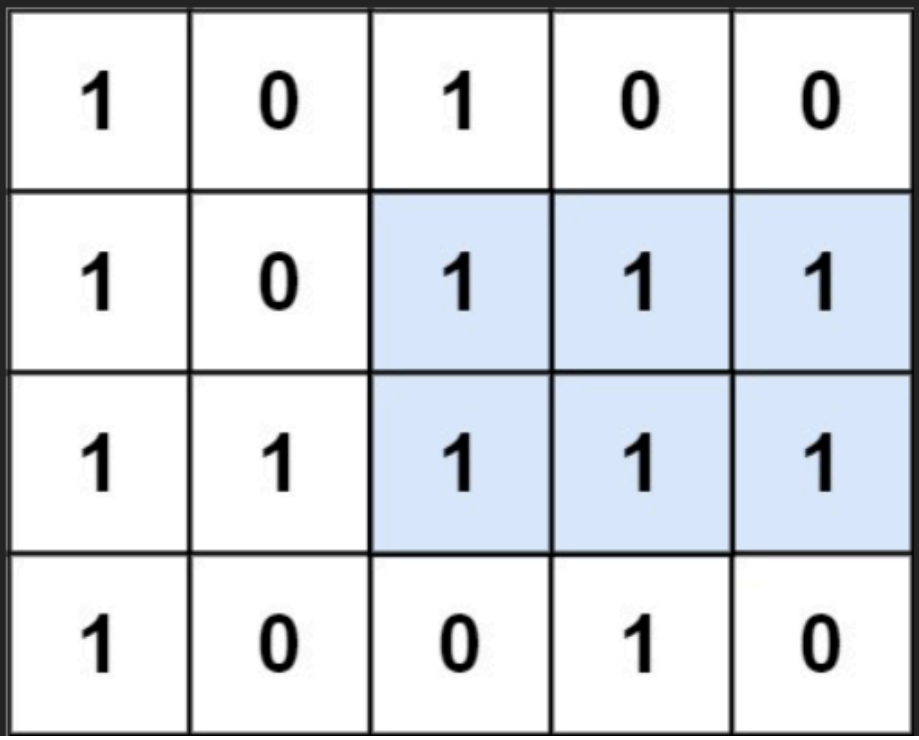
85. Maximal Rectangle

Solv

Hard | Topics | Companies

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:



```
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 <= rows, cols <= 200`
- `matrix[i][j]` is `'0'` or `'1'`.

Python:

```
class Solution:
    def area(self, heights: List[int]) -> int:
        stack = []
        maxArea = 0
        n = len(heights)

        for i in range(n + 1):
            h = 0 if i == n else heights[i]
            while stack and h < heights[stack[-1]]:
                height = heights[stack.pop()]
                width = i if not stack else i - stack[-1] - 1
                maxArea = max(maxArea, height * width)
            stack.append(i)
```

```
        return maxArea

    def maximalRectangle(self, matrix: List[List[str]]) -> int:
        if not matrix:
            return 0

        m, n = len(matrix), len(matrix[0])
        hist = [0] * n
        ans = 0

        for i in range(m):
            for j in range(n):
                if matrix[i][j] == '1':
                    hist[j] += 1
                else:
                    hist[j] = 0
            ans = max(ans, self.area(hist))

        return ans
```

JavaScript:

```
var maximalRectangle = function(matrix) {
    if (matrix.length === 0) return 0;
    let n = matrix[0].length;
    let height = Array(n).fill(0);
    let ans = 0;

    for (let row of matrix) {
        for (let i = 0; i < n; i++)
            height[i] = row[i] === '1' ? height[i] + 1 : 0;

        let stack = [];
        for (let i = 0; i <= n; i++) {
            let cur = i === n ? 0 : height[i];
            while (stack.length && height[stack[stack.length-1]] > cur) {
                let h = height[stack.pop()];
                let w = stack.length === 0 ? i : i - stack[stack.length-1] - 1;
                ans = Math.max(ans, h * w);
            }
            stack.push(i);
        }
    }
    return ans;
};
```

Java:

```
class Solution {
    public int maximalRectangle(char[][] matrix) {
        int m = matrix.length, n = matrix[0].length, ans = 0;
        int[]hist = new int[n];
        for(int i=0;i<m;i++){
            for(int j=0;j<n;j++){
                if(i == 0){
                    if(matrix[i][j] == '1')hist[j] = 1;
                    else hist[j] = 0;
                }
                else{
                    if(matrix[i][j]!='0')hist[j]+=1;
                    else hist[j] = 0;
                }
            }
            int area = area(hist);
            ans = Math.max(ans, area);
        }
        return ans;
    }
    public static int area(int[] heights) {
        int n = heights.length;
        int maxArea = 0;
        Stack<Integer> stack = new Stack<>();
        for (int i = 0; i <= n; i++) {
            int h = (i == n) ? 0 : heights[i];
            while (!stack.isEmpty() && h < heights[stack.peek()]) {
                int height = heights[stack.pop()];
                int width = stack.isEmpty() ? i : i - stack.peek() - 1;
                maxArea = Math.max(maxArea, height * width);
            }
            stack.push(i);
        }
        return maxArea;
    }
}
```