Question: https://leetcode.com/problems/maximal-rectangle/submissions/

Intuition :

1) Pick one row

2) Do summation of each index till that row

i) if any index value is 0 then put 0 else previous summation + 1 3) Pass this array to get max area (84. Largest Rectangle in Historgram)4) Update max area

84. Largest Rectangle in Histogram

Intuition :1) Max area will always have atleast one full bar height on any index2) Find largest rectangle including each bar one by one.

a) For each bar, We have to find it's left limit & right limit (to know the maximum width)

b) Find it's left limit (where we find any index's value is smaller than current index in left side array of curr index)

c) Find it's right limit (where we find any index's value is smaller than current index in right side array of curr index

3) Take the maximum of all the max area found by each bar.

4) calculate area

width \* height

where width = right limit - left limit + 1

height = curr index's value5) Update max area & return it

Code:  
class Solution {

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

if(matrix.length == 0) return 0;

int maxArea = 0;

int row = matrix.length;

int col = matrix[0].length;

int[] dp = new int[col];

for(int i=0;i<row;i++){

for(int j=0;j<col;j++){

dp[j] = matrix[i][j] == '1' ? dp[j]+1 : 0;

}

//treating dp[j] as histogram, solving max area problem there and updating the max area

maxArea = Math.max(maxArea, findMaxAreaInHistogram(dp));

}

return maxArea;

}

//84. Largest Rectangle in Histogram code

public int findMaxAreaInHistogram(int[] dp){

int len = dp.length;

int maxArea = 0;

int[] left = new int[len];

int[] right = new int[len];

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

//traversing left to right, finding left limit

for(int i=0;i<len;i++){

if(stack.isEmpty()){

stack.push(i);

left[i] = 0;

}else{

while(!stack.isEmpty() && dp[stack.peek()] >= dp[i])

stack.pop();

left[i] = stack.isEmpty() ? 0 : stack.peek()+1;

stack.push(i);

}

}

//doing empty to stack

while(!stack.isEmpty())

stack.pop();

//traversing right to left, find right limit

for(int i=len-1;i>=0;i--){

if(stack.isEmpty()){

stack.push(len-1);

right[i] = len - 1;

}else{

while(!stack.isEmpty() && dp[stack.peek()] >= dp[i])

stack.pop();

right[i] = stack.isEmpty() ? len-1 : stack.peek()-1;

stack.push(i);

}

}

//traversing the array , caculating area

int[] area = new int[len];

for(int i=0;i<len;i++){

area[i] = (right[i] - left[i] + 1) \* dp[i];

maxArea = Math.max(maxArea, area[i]);

}

return maxArea;

}

}

Github Link :<https://lnkd.in/ecwtJeaz>