**Max rectangle(Hard level)**

**Company Tags**

[**Flipkart**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Flipkart) [**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft) [**Samsung**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Samsung) [**MakeMyTrip**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=MakeMyTrip) [**Directi**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Directi) [**Intuit**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Intuit) [**Google**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Google)

Statement.

Given a binary matrix **M**of size **n X m**. Find the maximum area of a rectangle formed only of **1s** in the given matrix.

**Example 1:**

**Input:**

n = 4, m = 4

M[][] = {{0 1 1 0},

{1 1 1 1},

{1 1 1 1},

{1 1 0 0}}

**Output:** 8

**Explanation:** For the above test case the

matrix will look like

0 1 1 0

1 1 1 1

1 1 1 1

1 1 0 0

the max size rectangle is

1 1 1 1

1 1 1 1

and area is 4 \*2 = 8.

Java Code

//{ Driver Code Starts

import java.util.\*;

class FindMinCost

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

int t = sc.nextInt();

while(t > 0)

{

int n = sc.nextInt();

int m = sc.nextInt();

int arr[][] = new int[n][m];

for(int i=0; i<n; i++)

{

for(int j=0; j<m; j++ )

{

arr[i][j] = sc.nextInt();

}

}

System.out.println(new Solution().maxArea(arr, n, m));

t--;

}

}

}

class Solution {

public int maxArea(int matrix[][], int n, int m) {

int[] temp = new int[m];

int maxArea = 0;

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

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

if(matrix[i][j] == 0) {

temp[j] = 0;

} else {

temp[j] += matrix[i][j];

}

}

maxArea = Math.max(maxArea, largestRectangleArea(temp));

}

return maxArea;

}

public int largestRectangleArea(int[] heights) {

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

int area=0;

int n=heights.length;

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

while(!st.isEmpty() && (i==n || heights[i]<=heights[st.peek()])){

int h=heights[st.pop()];

int w=st.isEmpty()? i : i-st.peek()-1;

area=Math.max(area, h\*w);

}

st.push(i);

}

return area;

}

}