Max Rectangle

Given a binary matrix. Find the maximum area of a rectangle formed only of 1s in the given matrix.

Input: n = 4, m = 4 **Output**: 8

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

Algorithm:

1. Run a loop to traverse through the rows.
2. Now If the current row is not the first row then update the row as follows, if matrix[i][j] is not zero then matrix[i][j] = matrix[i-1][j] + matrix[i][j].
3. Find the maximum rectangular area under the histogram, consider the ith row as heights of bars of a histogram.
4. Do the previous two steps for all rows and print the maximum area of all the rows.

Code: Implementation of finding Max area rectangle by MAH using NSL and NSR.

```
class Solution{
  public:
    int MAH(vector<int>&v)
    {
        int ans = 0;
        stack<int> st;
```

```
        int n = v.size();
        vector<int> nsl(n),nsr(n);
        for(int i=n-1;i>=0;i--)
        {
            if(st.empty()) nsr[i] = n;
            else if(v[st.top()] < v[i]) nsr[i] = st.top();
            else
            {
                while(!st.empty() && v[st.top()] >= v[i])
                {
                    st.pop();
                }
                if(st.empty()) nsr[i] = n;
                else nsr[i] = st.top();
            }
            st.push(i);
        }
        while(!st.empty()) st.pop();
        for(int i=0;i<n;i++)
        {
            if(st.empty()) nsl[i] = -1;
            else if(v[st.top()] < v[i]) nsl[i] = st.top();
            else
            {
                while(!st.empty() && v[st.top()] >= v[i])
                {
                    st.pop();
                }
                if(st.empty()) nsl[i] = -1;
                else nsl[i] = st.top();
            }
            st.push(i);
        }
        for(int i=0;i<n;i++)
        {
            ans = max(abs(nsr[i]-nsl[i]-1)*v[i],ans);
        }
        return ans;
    }

    int maxArea(int M[MAX][MAX], int n, int m)
    {
        int ans = 0;
```

```
        vector<int> v(m,0);
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<m;j++)
            {
                if(M[i][j] == 0)
                {
                    v[j] = 0;
                }
                else
                {
                    v[j]++;
                }
            }
            int a = MAH(v);
            ans = max(a,ans);
        }
        return ans;
    }
};
```

Code 2:

```
#include <bits/stdc++.h>
using namespace std;
#define MAX 100

int MAH(int A[], int n)
{
    if(n == 0) return 0;
    if(n == 1) return A[0];
    int mx = 0, i = 0;
    stack<int> st;
    for(i=0;i<n;i++)
    {
        if(st.empty() || A[i] >= A[st.top()])
            st.push(i);
        else
```

```
        {
            while(!st.empty() && A[st.top()] > A[i])
            {
                int temp = A[st.top()];
                st.pop();
                if(st.empty())
                {
                    mx = max(mx,temp*i);
                }
                else
                {
                    mx = max(mx, temp*(i-st.top()-1));
                }
            }
            st.push(i);
        }
    }
    while(!st.empty())
    {
        int temp = A[st.top()];
        st.pop();
        if(st.empty())
        {
            mx = max(mx,temp*i);
        }
        else
        {
            mx = max(mx, temp*(i-st.top()-1));
        }
    }
    return mx;
}

int maxArea(int M[MAX][MAX], int n, int m)
{
    int m1 = m;
    int maxarea = 0;
    maxarea = max(maxarea, MAH(M[0],m1));
    for(int i=1;i<n;i++)
```

```
    {
        for(int j=0;j<m;j++)
        {
            if(M[i][j] == 0)
            {
                M[0][j] = 0;
            }
            else
            {
                M[0][j] += M[i][j];
            }
        }
        maxarea = max(maxarea, MAH(M[0],m1));
    }
    return maxarea;
}

int main()
{
    int M[MAX][MAX];
    int n,m;
    cin >> n >> m;
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            cin >> M[i][j];
        }
    }
   cout << maxArea(M, n, m) << endl;
}
```

Time Complexity: O(N*M)

Space Complexity: O(M)