**Algorithmic Problem Solving [23ECSE309]**

**Semester: VI 2023-24**

**Q-Box Assignment Set**

Student Name: Vineet Gajanan Pai

SRN: 01fe21bcs273

Branch: computer science

**Question 01**

Title: Max Tower height

Level: Easy

Concepts Tested: Problem solving

**Problem Statement:**

Vineet is given with a 2D matrix which contains the skyline of his city which is in 0’s and 1’s, number of 1’s in a column represent building height. He wanted to find the tallest building of his city with its position and height. If more than one buildings have same height than print details of left most one.

**Input Format:**

User first enters M, N which indicates the number of rows and column, followed by MxN values.

**Constraints:**

1 <= M,N <= 10000

Height of tallest building <= M

**Output Format:**

Two integers representing position and height of building.

**Solution:**

#include<iostream>

#include<cmath>

using namespace std;

int main()

{

int m,n,temp;

cin >> m >> n;

int mat[m][n];

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

{

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

{

cin >> mat[i][j];

}

}

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

{

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

{

if(mat[i][j]==1)

{

cout << j+1 <<' '<<m-i<<endl;

exit(0);

}

}

}

return 0;

}

**Sample Test Cases:**

Sample 1:

Sample1:

Input:

3 3

0 0 1

0 1 1

1 1 1

Output:

3 3

Explanation:

Third column contain the tallest building with height of 3.

Sample 2:

Input:

5 10

0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0

1 0 0 1 0 1 0 0 0 1

1 1 0 1 0 1 0 1 0 1

1 1 1 1 1 1 1 1 1 1

Output:

1 4

Explanation:

First column contains the tallest building with height of 4.

**Question 02**

Title: Max timber problem

Level: Medium

Concepts Tested: dp, problem solving

**Problem Statement:**

Kumarswamy is required to cut trees from his farm for a function as he won the MP election. He wants to maximize the amount of timber by cutting down trees, but his wife insists that he must not cut down two adjacent trees. He is given an integer array representing the trees in his farm and the values representing the amount of timber each tree will provide. Help him determine the positions of the trees he should cut to maximize the timber.

**Input Format:**

The first line contains an integer N which indicates the number of trees. The second line contains N space-separated integers representing the amount of timber from each tree.

**Constraints:**

1 <= N <= 1000000

1<=array[i]<=50

**Output Format:**

Integer representing max timber he can get

**Solution:**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

long long int max\_sum\_no\_adjacent(const vector<int>& arr) {

int n = arr.size();

if (n == 0) {

return 0;

}

if (n == 1) {

return arr[0];

}

vector<long long int> dp(n, 0);

dp[0] = arr[0];

dp[1] = max(arr[0], arr[1]);

for (int i = 2; i < n; ++i) {

dp[i] = max(arr[i] + dp[i-2], dp[i-1]);

}

return dp[n-1];

}

int main() {

int n;

cout << "Enter the number of elements: ";

cin >> n;

vector<int> arr(n);

cout << "Enter the elements of the array: ";

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

cin >> arr[i];

}

cout << "Max sum with no adjacent: " << max\_sum\_no\_adjacent(arr) << endl;

return 0;

}

**Sample Test Cases:**

Sample1:

Input:

5

3 2 5 10 7

Output:

15(3+5+7)

Sample2:

Input:

11

26 32 43 11 2 34 10 20 41 48 23

Output:

171