UCA_2018_Practice_Test_7

08 : 07
to test end

Counting Groups

A *2D* array, *m*, is an $n \times n$ matrix where each cell contains either the value *0* or the value *1*. Any two cells (x_1, y_1) and (x_2, y_2) in *m* fall into the same *group* if $|x_1 - x_2| + |y_1 - y_2| = 1$ and both cells contain the value *1*.

Complete the *countGroups* function in your editor. It has *2* parameters:

1. An $n \times n$ *2D* array of integers, *m*, where the value of each element $m_{i,j}$ (where $0 \le i,j < n$) is a binary integer (i.e., a *0* or *1*).
2. An array of *q* integers, *t*, where the value of each element t_k (where $0 \le k < q$) is a group size for which you must find the number of groups in *m*.

Your function must go through each of the *q* integers in array *t* and, for each t_k (where $0 \le k < q$), find the number of groups in *m* having size t_k. It must then add the result to index *k* of a *q*-element array of integers to be returned by the function (we'll call this array *ret*).

After finding the result for each element in *t*, your function must return the *ret* array. Recall from the above paragraph that this is a *q*-element array of integers where each element *k* ($0 \le k < q$) denotes the number of groups of size t_k in array *m*.

Input Format
The locked stub code in your editor reads the following input from stdin and passes it to your function:
The first two lines both contain an integer, *n*, denoting the number of rows in array *m*.
The second line contains an integer, *n*, denoting the number of columns in array *m*.
Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains *n* space-separated binary integers describing the respective elements of row *i* in *m*.
The next line contains an integer, *q*, denoting the number of test cases.
Each line *k* of the *q* subsequent lines (where $0 \le k < q$) contains an integer describing element *k* in array *t*.

Constraints

- $1 \le n \le 10^3$
- $1 \le q \le n$
- $1 \le t_k \le n^2$

Output Format
Your function must return an array of integers where each element *k* denotes the number of groups of size t_k present in array *m*. This is printed to stdout by the locked stub code in your

```
10
10
1 1 1 1 1 1 1 1 1 1
1 1 1 1 0 0 0 0 0 0
1 1 1 0 0 0 1 1 1
1 1 0 0 1 0 0 1 1 1
1 0 1 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
5
1
10
20
2
6
```

Sample Output 1

```
2
2
1
1
1
```

Sample Input 2

```
5
5
1 0 1 1 0
0 1 0 0 1
1 0 1 1 0
1 0 1 1 0
0 1 0 0 1
5
1
2
3
4
5
```

UCA_2018_Practice_Test_7

08 : 07
to test end

```
5
2
0
1
0
```

Explanation

Sample Case 1:

t_0 = *1*: *m* has two groups of this size, so index *0* in our return array should contain the value *2*.
t_1 = *10*: *m* has two groups of this size, so index *1* in our return array should contain the value *2*.
t_2 = *20*: *m* has one group of this size, so index *2* in our return array should contain the value *1*.
t_3 = *2*: *m* has one group of this size, so index *3* in our return array should contain the value *1*.
t_4 = *6*: *m* has one group of this size, so index *4* in our return array should contain the value *1*.

Sample Case 2:

t_0 = *1*: *m* has five groups of this size, so index *0* in our return array should contain the value *5*.
t_1 = *2*: *m* has two groups of this size, so index *1* in our return array should contain the value *2*.
t_2 = *3*: *m* has zero groups of this size, so index *2* in our return array should contain the value *0*.
t_3 = *4*: *m* has one group of this size, so index *3* in our return array should contain the value *1*.
t_4 = *5*: *m* has zero groups of this size, so index *4* in our return array should contain the value *0*.

YOUR ANSWER

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour. Start tour ×

Draft saved 08:21 pm Original code C++

```
#include ↔

using namespace std;

```

UCA_2018_Practice_Test_7

08 : 07
to test end

```



int dfs(vector<vector<int> >&arr, int i, int j, int count)
{
    if(i<0 || i>=arr.size() || j<0 || j>=arr[0].size() || arr[i][j] =
        return count;
    else
    {
        arr[i][j] =0;
        count++;
        int dir[4][2] ={ {-1,0}, {1,0},{0,-1}, {0,1}};
        int k;
        for(k=0;k<4;k++)
        {
            int newi = i + dir[k][0];
            int newj = j+ dir[k][1];
            count = dfs(arr,newi,newj, count);
        }
         return count;
    }
}


vector <int> countGroups(vector < vector<int> > arr, vector <int> t)
    int i , j;
    map<int,int> m;map<int,int>::iterator it;vector<int>res;
    for(i=0;i<arr.size();i++)
    {
        for(j=0;j<arr[0].size();j++)
        {
            if(arr[i][j])
            {
                int k = dfs(arr,i,j,0);
                cout<<k<<endl;
                 it = m.find(k);
                    if (it == m.end())
                    {
                       m[k] =1;
                    }
                else
                    m[k]++;
            }
        }
```

UCA_2018_Practice_Test_7

08 : 07
to test end

```


    for(i=0;i<t.size();i++)
    {
        if(m.find(t[i]) != m.end())
            res.push_back(m[t[i]]);
        else
            res.push_back(0);
    }
return res;
}


```

```
int main()
{↔}

```

Line: 34 Col: 46

☐ Test against custom input

Run Code

Submit code & Continue

(You can submit any number of times)

Download sample test cases *The input/output files have Unix line endings. Do not use Notepad to edit them on windows.*

Compiled successfully. All available test cases passed!

Tip: Debug your code against custom input

Test Case #1:	✓	Test Case #6:	✓
Test Case #2:	✓	Test Case #7:	✓
Test Case #3:	✓	Test Case #8:	✓
Test Case #4:	✓	Test Case #9:	✓
Test Case #5:	✓	Test Case #10:	✓