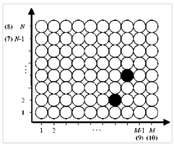
Rotten from the Core

locked

Problem | Submissions | Leaderboard | Discussions



Submissions: 120
Max Score: 100
Difficulty: Medium

Rate This Challenge:

More

На схемата са показани ябълки, подредени в N реда и M стълба. Една или две от ябълките са гнили (черните). От допира с тях след един ден и съседните им здрави също загниват. Напишете програма, която определя колко здрави ябълки ще останат след T дни.

Input Format

От първия ред на стандартния вход: N, M, T. От следващите един или два реда: по 2 числа – ред и стълб, в които се намира гнила ябълка в първоначалния момент.

Constraints

0 < N ≤ M ≤1000 0 < T ≤ 100

Output Format

На един ред на стандартния изход да се изведе търсеният брой здрави ябълки.

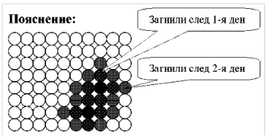
Sample Input 0

```
8 10 2
4 8
2 7
```

Sample Output 0

```
59
```

Explanation 0



Current Buffer (saved locally, editable) C++

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
#include <queue>
using namespace std;

struct Element{
    int xCord;
    int yCord;
    bool visited;
    int day;
    Element(int x,int y){
        xCord=x;
        yCord=y;
        visited=0;
        day=0;
    }
    Element()=default;


};
int main() {
    int M;
    int N;
    cin>>N;
    cin>>M;
    Element** matrix=new Element*[N];
    for(int i=0;i<N;i++){
        matrix[i]=new Element[M];
    }
    for(int i=0;i<N;i++){
        for(int j=0;j<M;j++){
            matrix[i][j]=Element(j,i);
        }
    }
    int days;
    cin>>days;
    int xCordFirst;
    int yCordFirst;
    int counterForApples=0;
    while(cin>>yCordFirst>>xCordFirst){
         --xCordFirst;
          queue<Element> apples;
    apples.push(matrix[N-yCordFirst][xCordFirst]);
        if((matrix[N-yCordFirst][xCordFirst]).visited==0){
            (matrix[N-yCordFirst][xCordFirst]).visited=1;
             ++counterForApples;
        }
    int counterForDays=1;

    while(1){
        if(apples.front().xCord+1<M){
            if((matrix[apples.front().yCord][apples.front().xCord+1]).visited==0){
            ++counterForApples;
                (matrix[apples.front().yCord][apples.front().xCord+1]).day=apples.front().day+1;
            (matrix[apples.front().yCord][apples.front().xCord+1]).visited=1;
                apples.push(matrix[apples.front().yCord][apples.front().xCord+1]);
            }



       }
        if(apples.front().xCord-1>=0){
            if((matrix[apples.front().yCord][apples.front().xCord-1]).visited==0){
                ++counterForApples;
                (matrix[apples.front().yCord][apples.front().xCord-1]).day=apples.front().day+1;
            apples.push(matrix[apples.front().yCord][apples.front().xCord-1]);

            (matrix[apples.front().yCord][apples.front().xCord-1]).visited=1;
             }



       }
        if(apples.front().yCord-1>=0){
            if((matrix[apples.front().yCord-1][apples.front().xCord]).visited==0){
            ++counterForApples;
                  (matrix[apples.front().yCord-1][apples.front().xCord]).day=apples.front().day+1;
            apples.push(matrix[apples.front().yCord-1][apples.front().xCord]);
            (matrix[apples.front().yCord-1][apples.front().xCord]).visited=1;
             }



       }
        if(apples.front().yCord+1<N){
            if((matrix[apples.front().yCord+1][apples.front().xCord]).visited==0){
            ++counterForApples;
                  (matrix[apples.front().yCord+1][apples.front().xCord]).day=apples.front().day+1;
            apples.push(matrix[apples.front().yCord+1][apples.front().xCord]);

            (matrix[apples.front().yCord+1][apples.front().xCord]).visited=1;
             }


       }
       apples.pop();
       if(apples.front().day==days){
           break;
       }
    }
    }

    cout<<N*M-counterForApples;
    return 0;
}
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code