X

Week 3 Programming Assignment: Frog Jumping

Due on 2019-09-20, 23:59 IST

- Select your language (C/C++/Java/Python2/Python3)
- Paste your code into the submission window.
- There are some public test cases and some (hidden) private test cases.
- "Compile and run" will evaluate your submission against the public test cases.
- "Submit" will evaluate your submission against the hidden private test cases. There are 6 private testcases in all, each with equal weightage. You will get feedback about which private test cases pass or fail, though you cannot see the actual test cases.
- Ignore warnings about "Presentation errors".

Frog Jumping

(INOI 2005) (http://www.iarcs.org.in/inoi/2005/inoi2005/inoi2005-qpaper.pdf)

The latest hit on TV is a jumping game played on a giant rectangular chessboard. Each participant dresses up in a green frog suit and starts at the top left corner of the board. On every square there is a spring-loaded launcher that can propel the person either to the right or down.

Each launcher has two quantities *R* and *D* associated with it. The launcher can propel the person upto *R* squares to the right and upto *D* squares down. The participant can set the direction of the launcher to *Right* or *Down* and set the number of squares to jump to any number between *1* and *R* squares when jumping right, or between *1* and *D* squares when jumping down. The winner is the one who can reach bottom right corner of the chessboard in the smallest number of jumps.

For instance, suppose you have 3 x 4 chessboard as follows. In each square

For instance, suppose you have 3×4 chessboard as follows. In each square, the pair of numbers indicates the quantities *(R,D)* for the launcher on that square.

(1,2)(1,2)(1,2)(2,1)
(3,1)(1,1)(1,2)(1,2)
(1,1)(1,1)(1,2)(2,2)

Here, one way to reach the bottom right corner is to first jump 1 square right, then jump 2 squares down to the bottom row, then jump right two times, one square a time, for a total of 4 jumps. Another way is to first jump 1 square down, then jump 3 squares right to the last column and finally jump one square down to the bottom right corner, for a total of 3 jumps. On this board, it is not possible to reach the bottom right corner in fewer than 3 jumps.

Your task is to write a program to calculate the smallest number of jumps needed to go from the top left corner to the bottom right corner, given the layout of the launchers on the board.

Solution hint

Set up a graph in which the (i,j) positions are vertices and use breadth first search.

Input format

The first line of the input contains two positive integers *M* and *N*, giving the dimensions of the chessboard. *M* is the number of rows of the board and *N* is the number of columns. This is followed by *2M* lines of input: *M* lines describing the *R* values of the launchers followed by *M* lines describing the *D* values of the launchers. Line $1+i$, $1 \leq i \leq M$, has *N* integers, describing the *R* values for row *i*. Line $M+1+i$, $1 \leq i \leq M$, has *N* integers, describing the *D* values for row *i*.

Output format

The output should be a single integer, the minimum number of jumps required to reach the bottom right square from the top left square on the given chessboard.

Test data

You may assume that $1 \leq M \leq 250$ and $1 \leq N \leq 250$.

Sample input

```
3 4
1 1 1 2
3 1 1 1
1 1 1 2
2 2 2 1
1 1 2 2
1 1 2 2
```

Sample output

```
3
```

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	3 4 1 1 1 2 3 1 1 1 1 1 1 2 2 2 2 1 1 1 2 2 1 1 2 2	3\n	3 \n	Passed
Test Case 2	10 10 1 1 3 4 1 5 2 1 5 1 1 3 4 2 1 4 3 4 5 1 3 4 4 3 3 1 1 2 2 3 4 2 3 2 4 3 5 4 4 3 1 1 3 4 1 3 1 1 3 5 5 4 1 3 3 2 5 5 4 2 2 3 1 5 1 5 2 1 4 4 1 4 2 4 1 5 1 1 2 4 5 5 2 4 3 4 4 2 5 5 1 1 3 3 5 2 4 4 5 2 1 1 3 4 1 5 2 1 5 1 1 3 4 2 1 4 3 4 5 1 3 4 4 3 3 1 1 2 2 3 4 2 3 2 4 3 5 4 4 3 1 1 3 4 1 3 1 1 3 5 5 4 1 3 3 2 5 5 4 2 2 3 1 5 1 5 2 1 4 4 1 4 2 4 1 5 1 1 2 4 5 5 2 4 3 4 4 2 5 5 1 1 3 3 5 2 4 4 5 2	7\n	7 \n	Passed
Test Case 3	20 20 4 5 5 5 3 3 6 3 4 6 5 2 7 5 8 6 8 8 6 4 5 2 2 8 5 3 8 1 6 8 5 4 7 8 7 1 5 1 8 3 5 2 6 7 7 1 6 6 6 6 7 2 6 7 3 1 2 2 2 1 5 8 7 2 4 4 6 6 3 7 5 7 1 8 4 5 3 8 5 5 3 5 7 4 8 5 4 5 7 3 7 6 1 8 6 7 2	7\n	7 \n	Passed

```
8 39 48 36 30 8 8 3 18 26 22 31 43
23 48 5 9 11 44 46 7 36 36 3 36 5
 16 47 49 20 14 29 10 13 16 40 20
 23 42 39 49 15 20 41 39 17 48 48
 30 43 45 36 28 30 39 13 36 6 11 3
5 25 24 15 35 36 30 26 6
7 3 19 46 1 33 15 43 21 31 40 20 1
2 33 14 48 12 43 38 25 29 43 35 15
17 11 29 3 48 10 28 7 12 46 2 14 3
0 16 9 1 48 48 22 10 32 36 9 44 30
46 20 10 38 6 24 6 18 4 9 16 14 36
22 27 34 25 43 13 40 1 15 39 50 37
48 32 22 6 27 1 3 46 11 40 3 36 48
21 42 6 38 7 43 11 33 28 35 27 41
 26 27 5 14 27 41 14 10 14 19 36 1
7 22 33 29 13 35 16 12 7 7 17 44 1
3 12 4 48 39 38 24 31 13 3 36 29 2
9 28 42 40 44 12 25 12 35 9 40 48
 45 7 9 2 14 28 47 28 41 1 27 29 4
0 3 10 5 5 47 33 35 26 24 24 21 35
50 32 20 8 21 19 4 30 29 7 43 8 4
 22 48 4 49 29 45 1 40 49 7 38 31
 41 13 6 14
```

Due Date Exceeded.
6 out of 6 tests passed.
You scored 100.0/100.

Your last recorded submission was :

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e5;
vector<int> adj[N];
int m,n;
int vis[N];
int lvl[N];
void bfs(int u){
    queue<int>q;
    lvl[u]=0,vis[u]=1;
    q.push(u);
    while(!q.empty()){
        int u=q.front();
        q.pop();
        for(int i: adj[u]){
            if(not vis[i]){
                lvl[i]=1+lvl[u];
                vis[i]=1;
                q.push(i);
            }
        }
    }

}
int main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    cin>>m>>n;
    int count=0;
    for(int i=1;i<=m*n;++i){
        int a;
        cin>>a;
        if((i-1)%n==0)count++;
        for(int j=i+1;j<=i+a and j<=count*n ;++j)
            adj[i].push_back(j);
    }
```

```
37      for(int i=1;i<=m*n;++i){
38          int a;
39          cin>>a;
40          for(int j=i;j+n<=n*m and a;j+=n,a--)
41              adj[i].push_back(j+n);
42      }
43      bfs(1);
44      cout<<lvl[n*m]<<endl;
45
46
47      return 0;
48  }
```