Experiment Title 5

Student :Anjali Singh **UID: 20BCS9239**
Branch: CSE **Section/Group:- 607-A**
Semester: 5th **Date:07/10/2022**
Subject :- Competitive Coding **Subject Code: 20CSP-314**

Aim/Overview of the practical: IMPLEMENTATION OF GRAPH

Q1. Snakes and Ladders: The Quickest Way Up

Markov takes out his Snakes and Ladders game, stares at the board and wonders: "If I can always roll the die to whatever number I want, what would be the least number of rolls to reach the destination?"

Rules The game is played with a cubic die of 6 faces numbered 1 to 6.

1. Starting from square 1, land on square 100 with the exact roll of the die. If moving the number rolled would place the player beyond square 100, no move is made.
2. If a player lands at the base of a ladder, the player must climb the ladder. Ladders go up only.
3. If a player lands at the mouth of a snake, the player must go down the snake and come out through the tail. Snakes go down only.

Function Description

Complete the quickestWayUp function in the editor below. It should return an integer that represents the minimum number of moves required.

quickestWayUp has the following parameter(s):

- ladders: a 2D integer array where each $ladders[i]$ contains the start and end cell numbers of a ladder
- snakes: a 2D integer array where each $snakes[i]$ contains the start and end cell numbers of a snake

Input Format

The first line contains the number of tests, t.

For each testcase:

- The first line contains n, the number of ladders.
- Each of the next n lines contains two space-separated integers, the start and end of a ladder.
- The next line contains the integer m, the number of snakes.
- Each of the next m lines contains two space-separated integers, the start and end of a snake.

Constraints

$1 \leq t \leq 10$

$1 \leq n, m \leq 15$

The board is always 10×10 with squares numbered 1 to 100.

Neither square 1 nor square 100 will be the starting point of a ladder or snake.

A square will have at most one endpoint from either a snake or a ladder.

Output Format

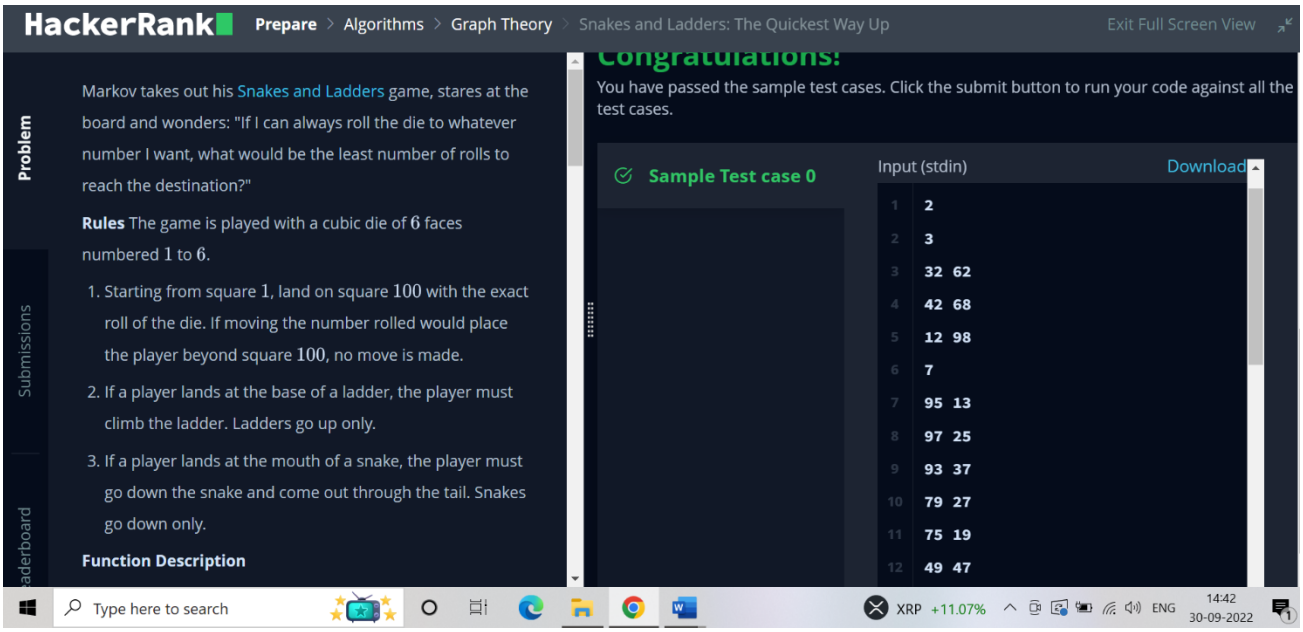
For each of the t test cases, print the least number of rolls to move from start to finish on a separate line. If there is no solution, print `-1`.

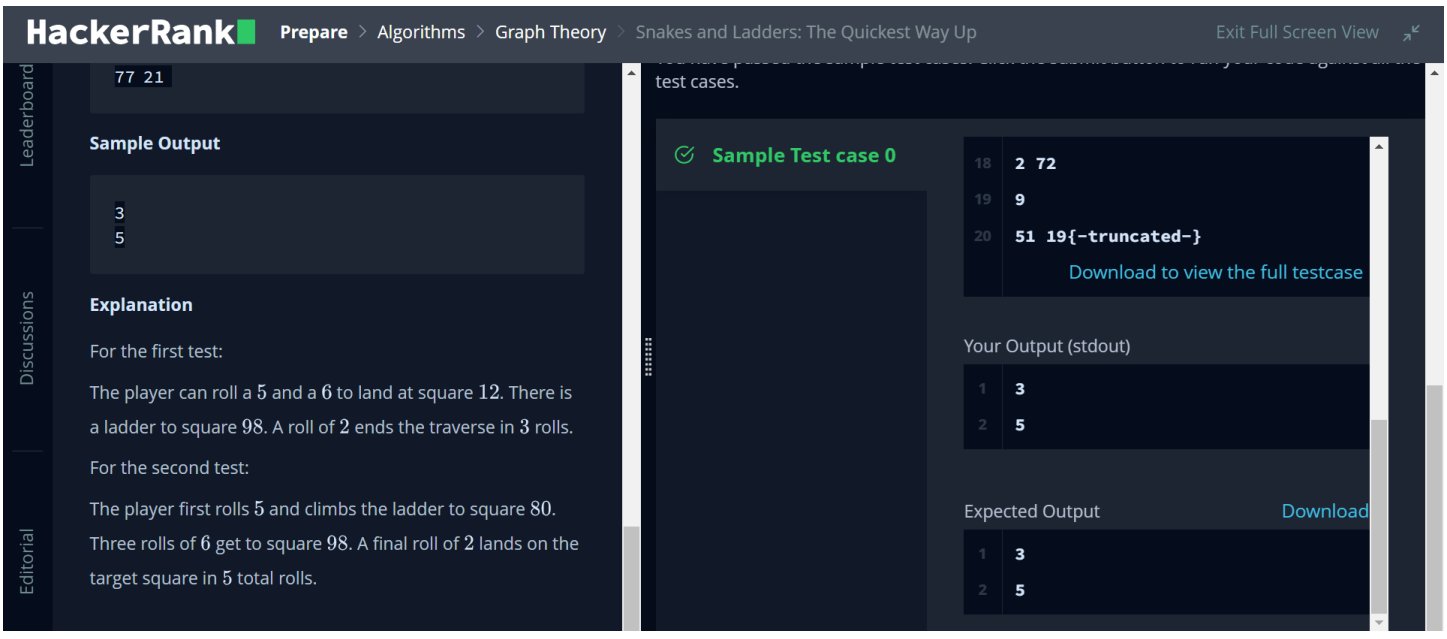
CODE:-

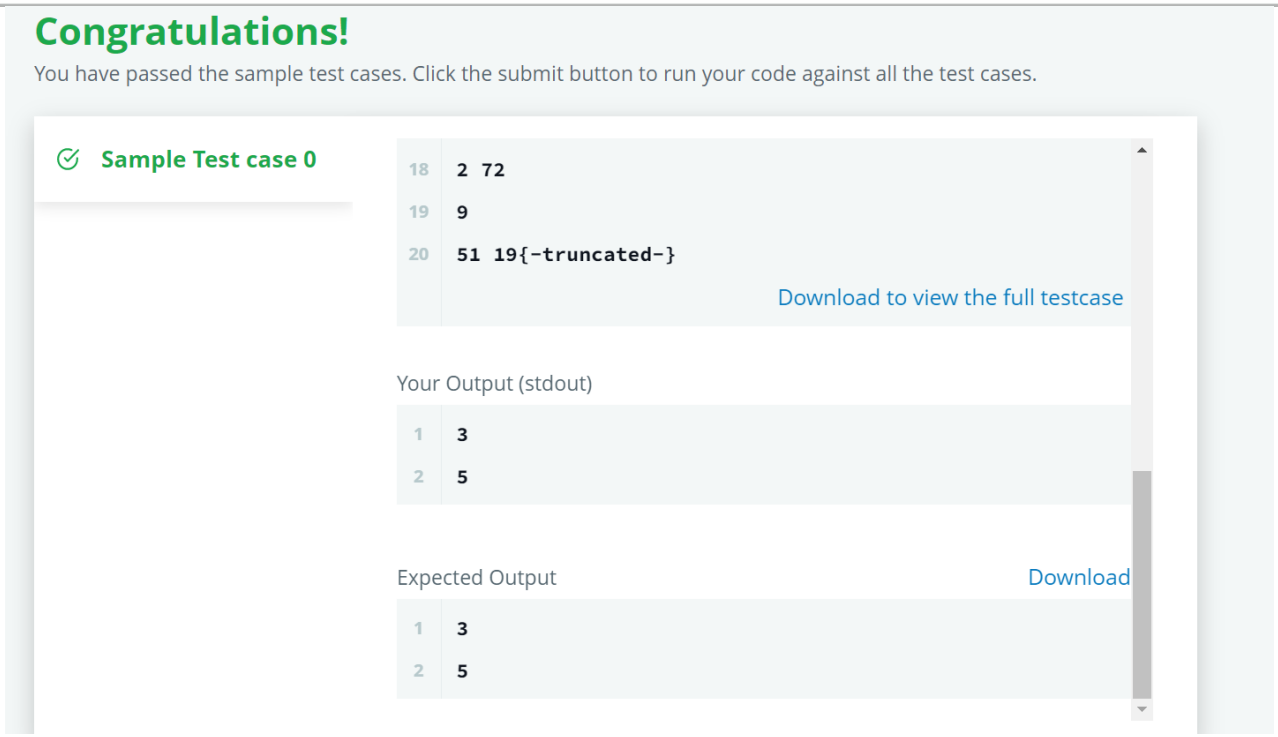
```
  #include <bits/stdc++.h>
#define ll long long
#define INF 999999999
using namespace std;
ll n, m, i, j, t, a, b;
vector < int > v(101,-1);
vector < int > dis(101,INF);

void func(int x){
    if(x>=100) return;
            //cout<<x<<endl;
```

```
        if(v[x]!=-1) if(dis[v[x]]>dis[x]){dis[v[x]]=dis[x]; func(v[x]); return;}
        for(int k=1; k<=6; k++)
        {
           if(x+k>100) break;
            if(dis[x+k]>dis[x]+1){
            dis[x+k]=dis[x]+1;
            func(x+k);
            }
        }
        return;
}
int main(){
        cin>>t;
        while(t>0) {
        cin>>n;
        for(i=1; i<=n; i++){
cin>>a>>b;
            v[a]=b;
        }
        cin>>m;
       for(i=1; i<=m; i++){
            cin>>a>>b;
            v[a]=b;
        }
        dis[1]=0;
        func(1);
        if(dis[100]==INF) cout<<-1<<endl;
        else
            cout<<dis[100]<<endl;
            for(j=1; j<=101; j++){
                dis[j]=INF;
                v[j]=-1;
            }
           t--;
       }
    return 0;
}
```

OUTPUT:-


HackerRank
Prepare > Algorithms > Graph Theory > Snakes and Ladders: The Quickest Way Up
Exit Full Screen View
Problem
Submissions
Leaderboard
Markov takes out his Snakes and Ladders game, stares at the board and wonders: "If I can always roll the die to whatever number I want, what would be the least number of rolls to reach the destination?"
Rules The game is played with a cubic die of 6 faces numbered 1 to 6.
1. Starting from square 1, land on square 100 with the exact roll of the die. If moving the number rolled would place the player beyond square 100, no move is made.
2. If a player lands at the base of a ladder, the player must climb the ladder. Ladders go up only.
3. If a player lands at the mouth of a snake, the player must go down the snake and come out through the tail. Snakes go down only.
Function Description
Congratulations!
You have passed the sample test cases. Click the submit button to run your code against all the test cases.
Sample Test case 0
Input (stdin)
Download
2
3
32 62
42 68
12 98
7
95 13
97 25
93 37
79 27
75 19
49 47
Type here to search
XRP +11.07%
ENG
14:42
30-09-2022

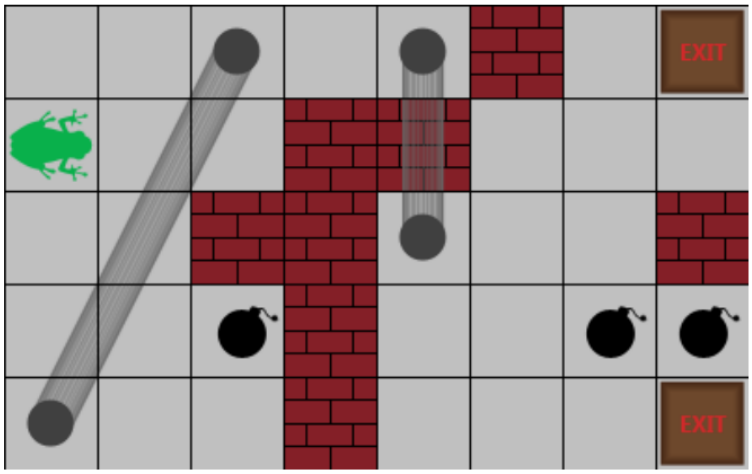

HackerRank
Prepare > Algorithms > Graph Theory > Snakes and Ladders: The Quickest Way Up
Exit Full Screen View
Leaderboard
Discussions
Editorial
77 21
Sample Output
3
5
Explanation
For the first test:
The player can roll a 5 and a 6 to land at square 12. There is a ladder to square 98. A roll of 2 ends the traverse in 3 rolls.
For the second test:
The player first rolls 5 and climbs the ladder to square 80. Three rolls of 6 get to square 98. A final roll of 2 lands on the target square in 5 total rolls.
test cases.
Sample Test case 0
2 72
9
51 19{-truncated-}
Download to view the full testcase
Your Output (stdout)
3
5
Expected Output
Download
3
5



Q2. Frog in Maze

Alef the Frog is in an $n \times m$ two-dimensional maze represented as a table. The maze has the following characteristics:

- Each cell can be free or can contain an obstacle, an exit, or a mine.
- Any two cells in the table considered adjacent if they share a side.
- The maze is surrounded by a solid wall made of obstacles.
- Some pairs of free cells are connected by a bidirectional tunnel.



When Alef is in any cell, he can randomly and with equal probability choose to move into one of the adjacent cells that don't contain an obstacle in it. If this cell contains a mine, the mine explodes and Alef dies. If this cell contains an exit, then Alef escapes the maze.

When Alef lands on a cell with an entrance to a tunnel, he is immediately transported through the tunnel and is thrown into the cell at the other end of the tunnel. Thereafter, he won't fall again, and will now randomly move to one of the adjacent cells again. (He could possibly fall in the same tunnel later.)

It's possible for Alef to get stuck in the maze in the case when the cell in which he was thrown into from a tunnel is surrounded by obstacles on all sides.

Your task is to write a program which calculates and prints a probability that Alef escapes the maze.

Input Format

The first line contains three space-separated integers n, m and k denoting the dimensions of the maze and the number of bidirectional tunnels.

The next n lines describe the maze. The i'th line contains a string of length m denoting the i'th row of the maze. The meaning of each character is as follows:

- `#` denotes an obstacle.
- `A` denotes a free cell where Alef is initially in.
- `*` denotes a cell with a mine.

CODE-

```
#include<cstdio>


char M[25][25]; // map
int T[25][25][2]; // tunnels
double P[2][25][25];

const int D[4][2] = {{-1,0}, {1, 0}, {0,-1}, {0,1}};
int h,w,t;

void calc(int in, int out) {
    for(int x=0;x<w;x++)
        for(int y=0;y<h;y++) {
            if(M[y][x] == '*' || M[y][x] == '#')
                P[out][y][x] = 0.0;
            if(M[y][x] == '%')
```

```
                P[out][y][x] = 1.0;
            if(M[y][x] == 'O' || M[y][x] == 'A') {
                int count = 0; double suma = 0.0;
                int px=x, py=y;
                if(T[y][x][0] != -1) {px = T[y][x][0]; py = T[y][x][1];}

                for(int i=0;i<4;i++) {
                    int x2 = px+D[i][0], y2 = py + D[i][1];
                    if(x2 < 0 || x2 >= w || y2 < 0 || y2 >= h)continue;
                    if(M[y2][x2] == '#')continue;
                    suma += P[in][y2][x2];
                    count++;
                }
                if(count == 0)
                    P[out][y][x] = 0.0;
                else P[out][y][x] = suma / count;
            }
        }
}

double get_ans(int p) {
    for(int i=0;i<h;i++)
        for(int j=0;j<w;j++)
            if(M[i][j] == 'A')
                return P[p%2][i][j];
    return -1.0;
}

int main() {
    scanf("%d%d%d", &h, &w, &t);

    for(int i=0;i<h;i++)
        scanf("%s", M[i]);

    for(int i=0;i<h;i++)
        for(int j=0;j<w;j++)
```

```
            T[i][j][0] = T[i][j][1] = -1;

    for(int i=0;i<t;i++){
        int x0, y0, x1, y1;
        scanf("%d%d%d%d", &y0, &x0, &y1, &x1);
        x0--;y0--;x1--;y1--;
        T[y0][x0][0] = x1;
        T[y0][x0][1] = y1;
        T[y1][x1][0] = x0;
        T[y1][x1][1] = y0;
    }

    const int limit = 80000;

    for(int i=0;i<limit;i++) {
        calc(i%2, (i+1)%2);
        // for(int y=0;y<h;y++){
        // for(int x=0;x<w;x++)printf("%.3lf|", P[(i+1)         %2][y][x]);
        //     printf("\n");
        // } printf("\n");
        //printf("%lf\n", get_ans(i+1));
    }
    printf("%lf\n", get_ans(limit));


}
```

OUTPUT:-

- Tunnels don't connect adjacent cells.

Output Format

Print one real number denoting the probability that Alef escapes the maze. Your answer will be considered to be correct if its (absolute) difference from the true answer is not greater than 10^{-6}.

Sample Input 0

```
3 6 1
###*00
0#0A%0
###*00
2 3 2 1
```

Sample Output 0

```
0.25
```

test cases.

Sample Test case 0

Input (stdin) Download

```
3 6 1
###*00
0#0A%0
###*00
2 3 2 1
```

Your Output (stdout)

```
0.250000
```

Expected Output Download

```
0.25
```

You solved this challenge. Would you like to challenge your friends?

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message

```
Success
```

Input (stdin) Download

```
3 6 1
###*00
0#0A%0
###*00
2 3 2 1
```

Expected Output Download

```
0.25
```