☆ Tom & Jerry in a Maze

After decades of chasing Jerry, Tom wants to make peace. Being skeptical, Jerry hides in an $n \times n$ maze. Tom decides the best way to make Jerry take him seriously is to collect all of the cheese pieces in the maze and give them to Jerry as a gift. A

Algorithmic Trader Coding Test

02 : 55
to test end

1
2
3
4

Complete the *minMoves* function in your editor. It has *3* parameters:

1. A *2D* array of integers, *maze*, denoting the maze where Jerry is hiding.
2. An integer, *x*, denoting the *x*-coordinate for Jerry's location.
3. An integer, *y*, denoting the *y*-coordinate Jerry's location.

Each cell in *maze* is labeled as follows:

- A path cell is represented by a *0*.
- A blocked cell (wall) is represented by a *1*.
- A cheese cell is represented by a *2*. Tom can move through a cheese cell just as he would a regular path cell.

Tom's initial position is *(0, 0)*. Your function must return an integer denoting the minimum number of moves that it will take for him to collect all the cheese and deliver it to Jerry at *(x, y)*; if the task is not possible, return -1.

Input Format

The locked stub code in your editor reads the following input from stdin and passes it to your function:

The first line contains an integer, *n*, denoting the number of rows in *maze*.

The second line contains an integer, *n*, denoting the number of columns in *maze*.

Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains *n* space-separated integers describing the respective elements of row *i* in *maze*.

The next line contains an integer, *x*, denoting the *x*-coordinate where Jerry is located in *maze*.

The next line contains an integer, *y*, denoting the *y*-coordinate where Jerry is located in *maze*.

Constraints

- $1 \le n \le 100$
- $0 \le$ *the number of cheese pieces* ≤ 10
- $1 \le x, y \le n$

Output Format

Your function must return an integer denoting the minimum number of moves Tom must make to collect all of the maze's cheese and deliver it to Jerry; if the task is not possible, return *−1*. This is printed to stdout by the locked stub code in your editor.

Sample Input 0

The following arguments are passed to your function:

maze = {{0, 2, 0}, {0, 0, 1}, {1, 1, 1}}
x = 1
y = 1

Sample Output 0

```
2
```

Sample Input 1

The following arguments are passed to your function:

maze = {{0, 1, 0}, {1, 0, 1}, {0, 2, 2}}
x = 1
y = 1

Sample Output 1

```
-1
```

Sample Input 2

The following arguments are passed to your function:

maze = {{0, 2, 0}, {1, 1, 2}, {1, 0, 0}}
x = 2
y = 1

Sample Output 2

```
5
```

Explanation

Sample Case 0:

The shortest path Tom can take to pick up all the cheese and deliver it to Jerry is *(0, 0)* → *(0, 1)* → *(1, 1)*. Because this involves *2* moves, we return *2*.

Sample Case 1:

It is not possible for Tom to reach Jerry, so we return *−1*.

Sample Case 2:

The shortest path Tom can take to pick up all the cheese and deliver it to Jerry is *(0, 0)* → *(0, 1)* → *(0, 2)* → *(1, 2)* → *(2, 2)* → *(2, 1)*. Because this involves *5* moves, we return *5*.

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

Original code   C ⌄   ⚙

```
#include ↔
```

```
/*
 * Complete the function below.
 */
int minMoves(int maze_size_rows, int maze_size_cols, int** maze, int x, int y) {


}

int main() {↔}

```

Line: 10 Col: 1

□ **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.*

About Privacy Policy Terms of Service