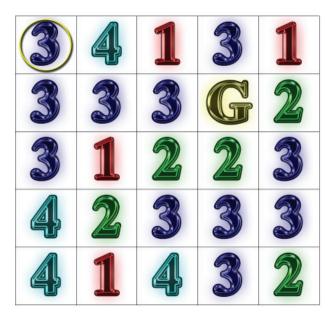
Rook Jumping Maze – Shortest Solution

Problem: Create a program that, for each of a sequence of specified Rook Jumping Mazes, prints the number of moves in a shortest solution, or "No solution." if no solution exists.

Here is an example of a Rook Jumping Maze:



A Rook Jumping Maze is defined as an *m*-by-*n* grid of *jump numbers*, a start cell (circled above), and a goal cell (marked "G" above). According to each cell's jump number, one may move that exact number of cells horizontally or vertically in a straight line. One may *not* wrap around edges, change directions mid-move, or move diagonally. One goal of a Rook Jumping Maze is to find the shortest path, i.e. the minimum number of moves, from the start cell to the goal cell. For the example above, there is a minimum of 13 moves: DRLUDLRULLRDU, where R, L, U, D stand for Right, Left, Up, and Down, respectively.

Input Format:

The input consists of a series of maze specifications followed by single line with a terminating zero (0).

Each maze specification begins with a line of space-separated integers providing:

- Number of rows (*m*)
- Number of columns (*n*)
- Start row
- Start column
- Goal row
- Goal column

Start and goal cells coordinates are zero-based, i.e. the 1st row and 1st column are at (0, 0).

What follows is a grid of jump numbers, i.e. m lines each representing a row of jump numbers having n space-separated integers. The integer in the ith row and jth column is the jump number for cell (i, j).

Output Format:

For each solvable maze, print a single line with an integer indicating the number of moves in the shortest solution. For each unsolvable maze, print a single line with the string "No solution."

Example Data:

EXAMPLE INPUT:

EXAMPLE OUTPUT:

```
13
No solution.
20
```