SWE2004-41 : Principles in Programming(Spring 2015)

Homework#1

Due date : 2015. 05. 30

Problem Statement

Mike Mazemeister has recently built a large **maze** in his backyard. The j-th character of the i-th element of **maze** is 'X' if the square is an impassable bush; otherwise, it is a '.'. Mike wants his friend, Jumping Jim, to solve the maze. Jim will start in row **startRow** in column **startCol**.

Unlike typical maze solvers, Jim has the ability to jump through the maze, rather than simply walking. Jim's possible moves are described in **moveRow** and **moveCol**. The i-th element corresponds to a move Jim can make in which his current row is changed by **moveRow**[i], and his current column is changed by **moveCol**[i]. For example, if **moveRow** = {1, 0, -1} and **moveCol** = {3, -2, 5}, Jim's legal moves are (1,3), (0, -2), and (-1, 5). However, Jim cannot move outside the boundary of the maze, and he cannot land on an impassable bush.

Mike wants to make the maze impossible for Jim to exit, and can place the exit in any cell containing a '.' in the maze. If this turns out to be impossible, then Mike wants to make Jim's trip take as long as possible. Jim is smart, and he will always exit the maze in the minimum number of jumps that he can. Return the maximum number of jumps that Jim will make to exit the maze; if it is impossible for him to exit the maze, return -1 instead.

Definition

Class:	MazeMaker
Method:	longestPath
Parameters:	String[], int, int, int[], int[]
Returns:	int
Method signature:	int longestPath(String[] maze, int startRow, int startCol, int[] moveRow, int[] moveCol)
(be sure your method is public)	

Constraints

- **maze** will contain between 1 and 50 elements, inclusive.
- Each element of **maze** will contain between 1 and 50 characters, inclusive.
- Each element of **maze** will contain the same number of characters.
- Each character of **maze** will be either 'X' or '.'.
- **maze** will contain at least 2 '.' characters.
- **startRow** will be between 0 and N-1, inclusive, where N is the number of elements in **maze**.
- **startCol** will be between 0 and M-1, inclusive, where M is the number of characters in each element of **maze**.
- **maze**[**startRow**][**startCol**] will be '.'.
- **moveRow** will contain between 1 and 50 elements, inclusive.
- **moveCol** will contain the same number of elements as **moveRow**.
- Each element of **moveRow** and **moveCol** will be between -50 and 50, inclusive.

Input & Output Example

0)

```
{"...",
 "...",
 "..."}
0
1
{1, 0, -1, 0}
{0, 1, 0, -1}
```
Returns: 3

Here Jim can move up, down, left and right. Mike will set the exit in one of the bottom corners, which take Jim 3 steps to reach.

1)

```
{"...",
 "...",
 "..."}
0
1
{1, 0, -1, 0, 1, 1, -1, -1}
{0, 1, 0, -1, 1, -1, 1, -1}
```
Returns: 2

This is the same problem, but now Jim can move diagonally. With this, he can reach any section in at most two steps.

2)

```
{"X.X",
 "...",
 "XXX",
 "X.X"}
0
1
{1, 0, -1, 0}
{0, 1, 0, -1}
```
Returns: -1

Here Mike can place the exit in the empty section of the bottom row; Jim can never reach it.

3)

```
{".......",
 "X.X.X..",
 "XXX...X",
 "....X..",
 "X....X.",
 "......."}
```

5

0

{1, 0, -1, 0, -2, 1}

{0, -1, 0, 1, 3, 0}

Returns: 7

4)

{"......."}

0

0

{1, 0, 1, 0, 1, 0}

{0, 1, 0, 1, 0, 1}

Returns: 6

5)

{"..X.X.X.X.X."}

0

0

{2, 0, -2, 0}

{0, 2, 0, -2}

Returns: -1

Since Jim can only jump (and can't move to the side), Mike can place the exit anywhere except the start to prevent Jim from winning.