COP-3337 Programming II

Programming Assignment 7

FIU Knight Foundation School of Computing & Info. Sciences

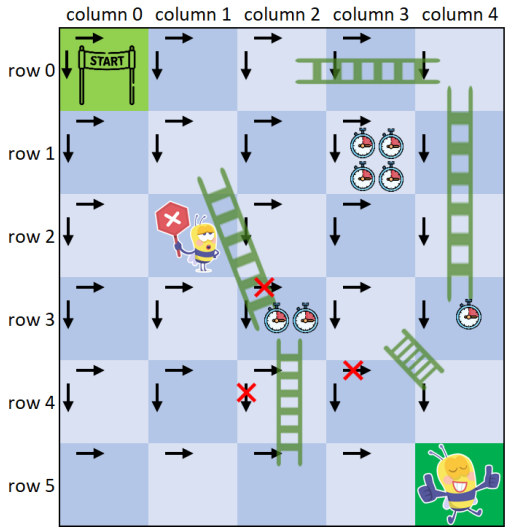
In this assignment, you are asked to complete a Java program that finds the shortest path that a hare needs to take to go through a grid-shape maze. The hare enters the maze from a specific square (start) and leaves the maze at another specific square (target).

1 Program Input and Maze Map

The program starts by asking the user to enter the name of a txt file containing the maze map. The file needs to be a tab-delimited rectangular shape table with multiple rows and columns. Here is an example of input file and the maze it represents:

Table 1: Example of input file

```
S\t N\t R2\t N\t D3
N\t R1,D2\t N\t W4\t N
N\t DE\t N\t N\t N
N\t N\t XR,D2,W2\t D1,R1\t W1
N\t N\t XD\t XR\t N
N\t N\t N\t N\t T
```



As you see in the example, content of each table cell is a string *str* that can be interpreted in the following way:

- *str.equals("S")*: cell is the start square where hare starts the journey. This cell is always at row 0 and column 0 (unless you want to do the extra-credit part of the assignment).
- *str.equals("T")*: cell is the target square where hare ends the journey. This cell is always at the last row and last column (unless you want to do the extra-credit part of the assignment).
- *str.equals("N")*: cell has no special property. Hare can move either one square to the right or one square to the bottom of the map (unless you want to do the extra-credit part of the assignment).
- *str.equals("DE")*: cell is a dead-end which means that if hare enters this cell, there is no way out of it.
- *str.contains("W")*: cell is a waiting square! If hare enters this cell, it needs to stay and wait in the cell for a specific units of time before leaving it. The length of waiting time is determined by the number that comes after 'W' (e.g. "W5" means that hare needs to stay and wait for 5 consequent steps before moving out of the cell).
- *str.contains("XR")*: cell is a no-right square! If hare enters this cell, it can't exit by moving to the right square.
- *!str.contains("XR") && str.contains("R")*: cell has a ladder on it! If hare enters this cell, it can either move normally (one step to the neighboring squares) or use the ladder to move multiple squares to the right. The length of ladder comes after the letter 'R' (e.g. "R3" means that there is a horizontal ladder of length 3).
- *str.contains("XD")*: cell is a no-down square! If hare enters this cell, it can't exit by moving down.
- *!str.contains("XD") && str.contains("D")*: cell has a ladder on it! If hare enters this cell, it can either move normally (one step to the neighboring squares) or use the ladder to move down multiple squares. The length of ladder comes after the letter 'D' (e.g. "D4" means that there is a vertical ladder of length 4, while "R3,D4" means that there is a ladder that moves hare 3 cells to right and 4 cells to the bottom of the maze).

2 Finding the Shortest Path

Given a maze, there may be many paths that allows hare to move from start to the target square. Some paths took longer than others. The goal of the program is to find the shortest path using the "Maze.solve" method available in the starter code.

Example of Shortest Path in a Maze

Consider the maze given in Table 1. There are many paths from start to target square. Below, you can find some of them and their corresponding lengths:

- $(0,0) \rightarrow (1,0) \rightarrow (2,0) \rightarrow (3,0) \rightarrow (4,0) \rightarrow (5,0) \rightarrow (5,1) \rightarrow (5,2) \rightarrow (5,3) \rightarrow (5,4)$. Length: 9
- $(0,0) \rightarrow (0,1) \rightarrow (0,2) \rightarrow (0,4) \rightarrow (3,4) \rightarrow (3,4) \rightarrow (4,4) \rightarrow (5,4)$. Length: 7
- $(0,0) \rightarrow (1,0) \rightarrow (1,1) \rightarrow (3,2) \rightarrow (3,2) \rightarrow (3,2) \rightarrow (5,2) \rightarrow (5,3) \rightarrow (5,4)$. Length: 8
- $(0,0) \rightarrow (1,0) \rightarrow (1,1) \rightarrow (2,1) \rightarrow$ dead-end. Length: ∞.

After considering all possible paths, the shortest path is the one with length 7 (second path in the list).

Recursive Solution

The current draft of the method recursively solves the maze assuming that all of the maze cells with the exception of source and target are normal squares (i.e. hare can move to either the right or bottom neighbors).

You need to manipulate the implementation of this method so that it can find the shortest path given a maze containing all kinds of squares (normal, waiting, dead-end, no-right, no-down and squares with ladders). Hint: only focus on manipulating *Maze.solveHelper* method.

3 50% Extra Credit

You'll get extra 50% if you can find the shortest path in the maze when hare, by default, is free to go in all four directions (left, right, up, down) and there are ladders moving hare up, down, right, left, up-right, up-left, down-right, and down-left. Also, there are two more cell types: no-up (represented by "XU" in the input file) and no-left (represented by "XL" in the input file).

4 Submissions

You need to submit a *.zip* file compressing the following folders:

- the packages containing all the java source files related to the assignment (*.java* files).
- **A readme file clearly explaining what parts have/haven't been implemented.**