ECE242 - Project 1 Arrays and Stacks

Overview

In this assignment, you will implement a simple puzzle solver that discovers a route through a maze from the start to finish. This will give you a chance to get familiar with the stack data structure.

Introduction

A maze is stored in a two-dimensional array as shown in Figure 1. Each block represents either a path or a wall, and only the locations of the two external portals are known. Your goal for this project is to implement a program in Java that will guide two robots through the maze. One robot will start from portal 1 and proceed to portal 2 and the other robot will start from portal 2 and proceed to portal 1.

Robot movements will occur randomly based on the result of a random number generator. If the random number generator creates a 0, the first robot moves. The generation of a 1 causes the second robot to move. For each movement, a robot moves 1 block. The direction of movement is arbitrary, although a robot should not proceed forward over a previously-visited block (its OK to go back). The robot is blocked by a wall or the maze boundary and must try a different path. After every movement, the robot should be in a different block than where it started unless a feasible path through the maze does not exist. If both robots reach their destination portals or one robot attempts to move to a block already occupied by the other robot, the program ends. Please do not use recursion for this program. **Hint: add a block to a robot's stack when you visit it for the first time. You may want to have separate stacks for each robot.**

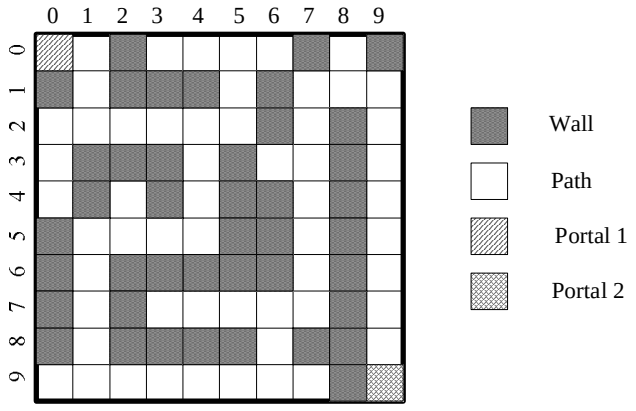

Figure 1. An example maze

Assignment Details

- ✓ Input the maze map in Figure 1 to a two-dimensional array named **Maze**, and print out the content of the maze array. 0 represents one portal cell and -1 represents the other portal cell, 1 and 2 represent a path and a wall accordingly.
- ✓ Implement a class, called **CellStorage**, which uses an array-based stack.. The stack should

at least have the following methods: push(cell), pop(), isEmpty(). Add more methods as needed.

- ✓ The two robots should alternate movements. Robot1 marks a visited cell with 7, and Robot2 marks it with 8. If both robots have visited a cell, it should be marked with a 9.
- ✓ Print the maze after each move.
- ✓ When the target portal is reached, or the robots collide, print "mission completed", and stop the program.
- ✓ Print the path to go out if the solution exists.
- ✓ Make sure your code can recognize a maze without an available route. Change cell Maze[8][9] from a path to a wall. Rerun your program. Show that it is able to print "No route found!"
- ✓ Make sure your solution is not limited to the maze in Figure 1. We may use a different maze for grading.

Grading

- ✓ You should submit your completed code, by the date posted on the course website. You will receive 75 points for your code running accurately, and 25 points for the thoroughness and usefulness of your comments which should be included in the code.
- ✓ Printing the initial maze is worth 5 points. Printing the intermediate mazes is worth 5 points. Implementing **CellStorage** class is worth 15 points. Traversing the maze correctly is worth 30 points. If the program ends with the right result, 10 points will be awarded. Identifying a maze without an available route is worth 10 points.
- ✓ Comments in your code should describe your design decisions and how the code operates. Please see the Project page on the course website for further information on comments.