Department of Electrical and Computer Engineering

ECX4235 – Data Structures and Algorithms

Mini Project - 2013-14

Due Date: 02. 06. 2014

Exiting A Maze

Consider the scenario of a trapped mouse trying to exit a maze. The mouse tries to exit the maze by systematically trying all possible routes. If it reaches a dead end, it retraces its steps to the last position and begins at least one more untried paths. By retaining the information that allows for resuming the search after a dead end is reached, the mouse uses backtracking.

The maze is implemented as a two dimensional character array in which passages are marked with 0s, walls by 1s, exit position by the letter ***e***, and the initial position of the mouse by the letter ***m*** as shown in figure 1.

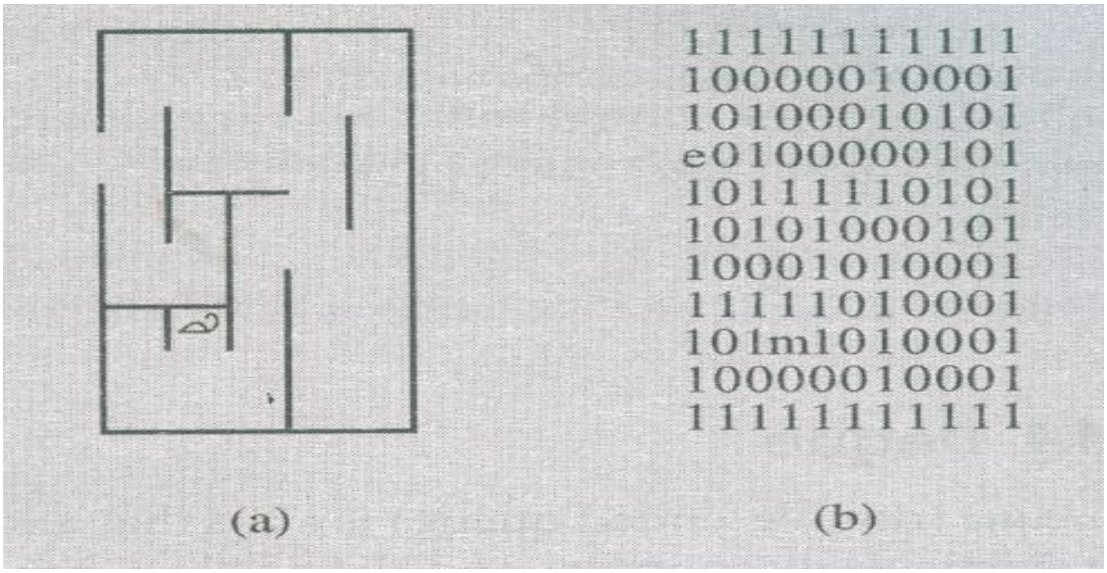

Figure 1

The program uses two stacks; one to initialize the maze and another to implement backtracking. The user enters a maze one line at a time and it can have any number of rows and columns. You have to assume that all columns are of equal lengths as well as the rows and contain any number of 1s, any number of 0s one **e** and one **m** only. The rows are pushed on the

stack **mazeRows** in the order they are entered after attaching one at the beginning and at the end.

A second stack, mazeStack is used in the process of escaping the maze. To remember untried paths for subsequent tries, the position of the untried neighbours of the current position (if any) are stored on a stack and always in the same order, first upper neighbor, then lower, then left, and finally right. After stacking the open avenues on the stack, the mouse takes the top most position and tries to follow it by first storing untried neighbours and then trying the topmost position and so forth until it reaches the exit or exhausts all possibilities and finds itself trapped. To avoid falling into an infinite loop of trying paths that have been already investigated, each visited position of the maze is marked with a period.

The stack stores coordinates of positions of cells. This could be done for instance, using two integer stacks, for x and y coordinates. Another possibility is to use one integer stack with both coordinates stored in one integer variable with the help of the shifting operation.

To understand the scenario consider the below example.

After the user enters the maze

1 1 0 0
0 0 0 e
0 0 m 1

The maze is immediately surrounded with a frame of 1 s

1 1 1 1 1 1
1 1 1 0 0 1
1 0 0 0 e 1
1 0 0 m 11
1 1 1 1 1 1

entryCell and currentCell are initialized to (3 3) and exitCell to (2 4) because curerntCell is not equal to exitCell, all four neighbors of the current cell (3 3) are tested, and only two of them are candidates for processing, namely, (3 2) and (2 3); therefore, they are pushed onto the stack. The stack is checked to see whether it contains any position, and because it is not empty, the topmost position (3 2) becomes current. An example is given in figure2.

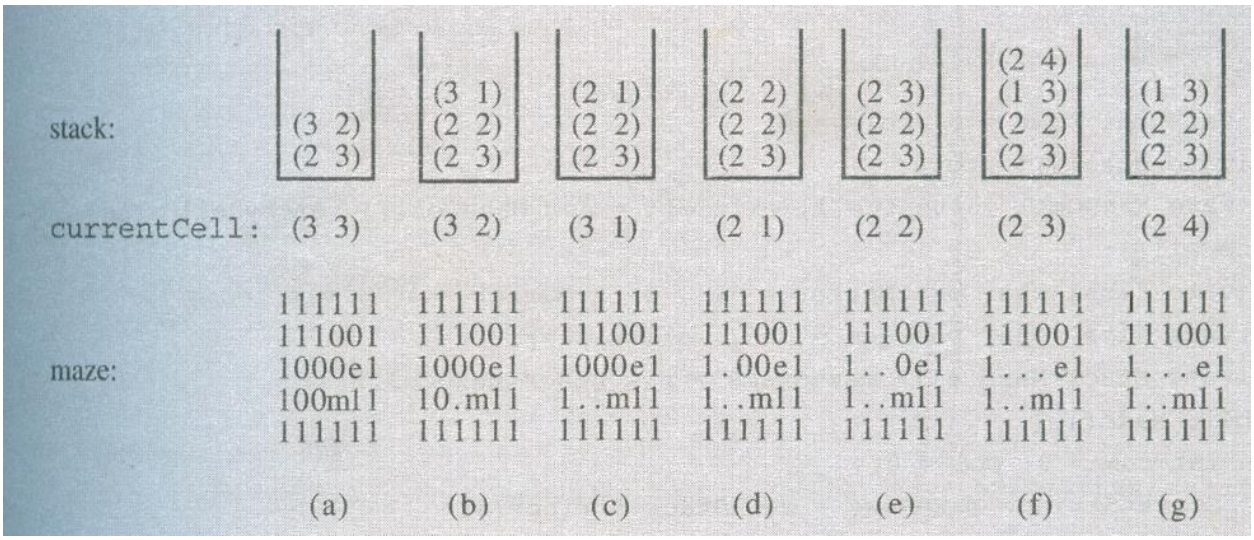

Figure 2

As per the mini project requirements you need to

1. Write a pseudo code algorithm to implement the above scenario.
2. Implement the above written pseudo code algorithm in Java programming language.
3. Write a simple report stating all the assumptions made, and scenario in detail with the pseudo code.

The evaluation will be done based on the report and your demonstration of the implementations.

Note:

To get eligibility for the subject ECX4235 minimum of 40% has to be taken from the mini project.