Homework 2

Com Sci 32 – Spring 2019

Problem 2

Given the algorithm, main function, and the maze provided, the following are the first 12 (r, c) coordinate pairs that are popped off the stack utilized in the algorithm -

(3, 5)

(3, 6)

(3, 4)

(2, 4)

(1, 4)

(1, 3)

(1, 2)

(1, 1)

(2, 1)

(3, 3)

(4, 5)

(5, 5)

Problem 4

Given the algorithm, main function and the maze provided, the following are the first 12 (r, c) coordinate pairs that are dequeued (or popped) from the queue utilized by the algorithm -

(3, 5)

(4, 5)

(3, 4)

(3, 6)

(5, 5)

(3, 3)

(2, 4)

(6, 5)

(5, 4)

(1, 4)

(7, 5)

(5, 3)

In the stack implementation, the algorithm proceeds to check if the maze is solvable by moving along by advancing to the coordinate pair which was most recently pushed into the stack – this coordinate pair will be legal to move to (i.e. will not be the character ‘X’ or off-limits with respect to the maze’s boundaries), and will be neighboring the present position (i.e. will be to the immediate south/ east/ west/ north of the coordinate pair which is currently recorded as the present position). Of course, as a starting point for the algorithm, present is chosen to be the initial position specified by the user and terminates when the present position equals the destination. However, it is possible that when the current position isn’t the destination and there are no other neighboring coordinates which can become the new position (along which the algorithm can continue traversing). In this case, the algorithm goes back to the previously visited position along the most recently taken path (i.e. the path that lead to the present position which turned out to be a dead end), and will continue exploring in a similar fashion along a neighboring position other than the one which turned out to be a dead end. If this position too has no other alternatives to go along (i.e. when the all legal neighboring positions turned out to be dead ends or have already been visited), it goes a step back again to continue moving along another legal candidate neighbor at that point (if any; if none, the process of tracing the way back and looking for legal unvisited neighboring positions is repeated). Employing this methodology, the algorithm will eventually reveal whether or not the maze is solvable between the given coordinate pairs. The priority of this algorithm is to proceed along a certain path without worrying about how the neighborhood of each point on the path can branch off until it encounters a dead end, even in which case backs off only minimally and to go on a separate path from there on (and doesn’t worry about neighborhoods at previous points in the dead-end path). In technical speak, depth is a priority of this algorithm (hence the preference of moving along a path instead of extensively analyzing neighborhoods unlike the queue implementation), and hence, it is a depth-first algorithm.

In the queue implementation, however, the algorithm, dequeues the present position, and if it wasn’t the destination, and enqueues all legal neighbors with respect to a present position. Next, it checks (if it exists) the first enqueued neighbor of the previous stage, and if it isn’t the destination, dequeues it and enqueues all legal neighbors with respect to that position. Next, it dequeues the second enqueued neighbor (if it exists) from the first stage and checks if it is the destination; if it isn’t, the algorithm enqueues the potential neighbors of the second neighbor of the first stage into the queue. In a similar fashion, after having checked whether or not any of the legal neighbors in the immediate layer of the present position are the destination, if none of them are, the algorithm would proceed to check the immediate neighborhood of all the neighbors of the present position. Thus, this algorithm’s search for destination proceeds in a layer by layer fashion, and its priority is extensively searching each layer for whether or not the destination was found. In technical speak, breadth is a priority of this algorithm (hence the preference of extensively searching the neighborhood of each point and not traversing a particular path, unlike the stack implementation), and hence, it is a breadth-first algorithm.