Michael Chau

2018-05-06

Final Description

1) The Problem

I wrote a program to solve the problem of finding the shortest path in a maze. My program outputs the distance of the shortest path. For this problem, I used binary mazes where 0 is passable and 1 is not passable. My search function assumes the maze starts at the upper-left corner and ends on the bottom-right of the matrix.

2) Steps and algorithms used to solve the problem

I implemented a breadth-first search for the graph with a queue data structure. I chose BFS because I’m most familiar with it and it seems practical. I chose queue because I needed a FIFO process of checking each node for BFS.

The BFS process starts with using the first node as the root. I have a queue for each node visited and a dictionary saving the node as a key and its distance from the root as its value. As we go through the queue, we check to see if the current node is at the end of the matrix by using the length of the matrix list and length of the first list in the matrix. If it isn’t, we’ll add all neighbor nodes that are open paths to the queue to check for later. We’ll also add those neighbor nodes to the distance dictionary with the value of the current node’s distance plus one more since it’s an extra step. The search will constantly add nodes to the queue until the end of the matrix is found.

3) Description of test case

For the first test case, I inputted a matrix that had no paths to the end. My solution correctly prompted users that there was not a path in the maze.

For the second, I had 2 connecting paths that both lead to the end. The search correctly avoided overlapping the 2 paths because of the check on visited nodes. My solution correctly solved the problem and found the distance of the shortest path.

For the last couple of mazes, I put 2 separate paths in each matrix: where the distance was drastically different from one another. I also checked to see if direction mattered by having different directions on each long path, but it didn’t. The solution worked as expected.

4) Issues

In my research, I found mazes that had unequal columns. My solution does not work for those kinds of mazes. I also realized that it doesn’t matter if the first node is a 0 or 1, but that’s not really an issue. We can pretend that’s a door. In the future, I would like to output the whole path to the end, either by displaying each direction or maybe creating a new matrix with only the shortest path shown.