Lab 7 Report

**Introduction:**

The purpose of this lab was to demonstrate our knowledge of traversing graphs. We had to make a custom maze by asking the user how many walls they would like to be removed from the maze. We had to display the number of cells that the maze has. We then had to remove the number of walls that the user wants. Once we removed the number of walls the user wanted removed, we have to display a message saying that the maze is either unsolvable, has one unique path, or has more than one path, of course this all depends on the number of walls that the user input. If the user wants a number that is less than the number of cells minus one, then the maze is not guaranteed to have a path, if the user puts a number equal to the number of cells minus one then there’s a unique path, and if the user inputs a number greater than the number of cells minus one then there is at least one path. Once the user has entered the number of walls, they’d like to be removed we make the maze, then create an adjacency list that matches the maze created. This lab also required us to find a path by using the breadth first search algorithm and the depth first search algorithm.

**Proposed Solution Design and Implementation:**

Displaying the message was quite simple, I just made several if statements revolved around the number that the user input. Removing the walls was somewhat difficult, initially I ran into the problem of the same wall being removed because of course I was choosing the walls at random. So, I created a list that would represent the indices of the wall that I wanted to remove, it was still generated randomly, but I would check to see if the index was already in the list before adding it to prevent duplicates. This resulted in the proper number of walls being removed.

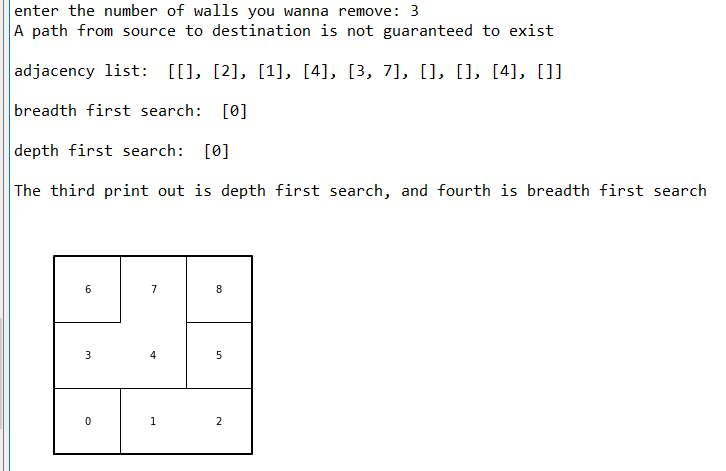
In order to create the adjacency list, I had to compare the list of walls before any walls were removed with the list of walls that didn’t include the removed walls. After comparing both of the lists I was able to determine which walls were removed. I made a new list just for all of the walls that were removed. Each element in the removed wall list is a list of length two. Each element in that list of length two is both and index and a value, it has kind of an inverse relationship, when one of them is an index the other is the value and vice versa. So, I append the values in the wall removed list to the adjacency list.

For the breadth first search I made a visited list to keep track of all of the nodes that have been visited. The breadth first search method that I made uses a queue to store all of the node’s neighbors (the things connected to the node). I pop the first element in the list and then add it to the visited list if it wasn’t already there, then I add its neighbors to the queue. This process continues until the current node equals the desired destination.

The depth first search is very similar to the breadth first search method. Really the only difference between the two is that I used a stack instead of using a queue to store the node’s neighbors. This resulted in a different path. The breadth first search method resulted in a methodical search, checks each neighbor, whereas the depth first search makes the deepest path, then has to backtrack if that path doesn’t result to the desired location.

**Experimental Results:**

I tested the methods that I made by changing the size of the maze several times. To check the number of walls being and adjacency lists I used a smaller maze because it was simply much easier to see if the correct number of walls were being removed and the adjacency list is much easier to follow as well. Below are some pictures of the results I got using a maze with 3 rows by 3 columns which I removed 3 walls.



I was unable to determine a path from the zero cell to the cell in the upper right-hand corner. I was able to get a breadth first search and depth first search working properly though. Below are some print outs of my breadth and depth first searches, the paths that the generated.

A screenshot of a cell phone

Description automatically generated

To get the runtimes for the adjacency list builder, I increased the number of cells that made up the maze, I did keep the number of walls removed to the same as the number of cells for testing purposes.

Unfortunately, I was unable to obtain accurate times for my breadth and depth first searches due to the fact that many of the times a wall would come between the starting point and the ending point, preventing the search to travel the whole graph to get to the end point. Because the walls get removed randomly it sometimes does not remove a wall blocking the path from the staring point to the end point.

**Conclusion:**

I learned that breadth first search and depth first search are practically coded the same, the only difference between the two were that breadth first search uses a queue and depth first search uses a stack. One other thing I learned in this lab is that was pretty important was how to make the adjacency list. I learned that I had to use the initial wall list and the wall list after the walls were removed to create the adjacency list instead of using a disjoint forest set. I just compared the two wall lists and then saw which walls were removed then added the removed walls to the adjacency list.

**Appendix Source Code:**

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

“I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class.”

Joey Roe