Jaime Vargas

Maan Singh

CS146 – Potika

Project 2 Report

How the code works.

This program works with a maze class, which displays and edits the mazes and performs the searches, a cell class, which acts as the vertices, which holds Boolean values dictating it's walls, status of visit(white, brown or black), and neighbors, and an MazeJUnit class which performs Junit testing. The maze and cells are initialized when a new maze class is made, and the constructor calls a buildMaze() method which creates a perfect maze, which is randomized in which walls it has torn down to complete it. This works with a method that finds all adjacent cells to a chosen cell and checks if it has neighboring cells with all their walls intact, of which it then chooses a random neighbor and knocks down the wall between the two cells. The DFS and BFS methods are called to perform the appropriate searches which call upon the solve method at the end to return the visualization to be displayed with the shortest path to solve the maze.

The Junits test a 4x4, 5x5, 6x6, 7x7, 8x8, and 10x10 maze and displays three mazes for each, a blank unsolved one, one that shows the steps of the searching algorithms, and one that shows the shortest path.

Issues

The hardest part of the project was creating and visualizing the initial maze. This was hard because we had trouble making it scalable, and the maze dimensions got bigger, especially knocking down neighboring walls. We resolved this by using an adjacency list to hold neighbors and reading the textbook a lot. We also used an adjacency matrix to handle the maze itself. This took the longest and is why we turned the project in a few days late. Once that was taken care of, the BFS and DFS were relatively straightforward.

Reflections

This project was fun and it showed us how the searches work pretty well. Prior to finishing this project, we were still a bit unsure of how exactly BFS and DFS worked and we didn't know how to properly implement them in algorithms, just knew the concept of it and could draw it out.