Motion Planning of Robots:
A Depth First Search Solution

By: Dustin Hengel, Madison Williams, and Bradley Worthen

Team 10

Strategy of Completion

Over the course of time the final project was assigned, all three members met 5 times for a combined 18 hours. While brainstorming possible solutions, we came to the realization of how different our coding methodologies were and decided that it would be best to do as much together as possible. This was intended to limit the confusion and time spent figuring out each member's code away from each other, increase communication to integrate the best combination of ideas, and have three sets of eyes be able to troubleshoot simultaneously if necessary to refactor problematic pieces of code. However, each member did spend some individual time working on the solution.

Individual Focuses

The solution was broken up into three major parts. One part was dedicated to correctly reading in the input and constructing a graph, another part was focused on implementing the Depth First Search algorithm on the graph, and the final portion was concentrated on displaying the output. While every part was worked on by every member, certain tasks within each part utilized the divide and conquer strategy. Madison and Bradley largely implemented the code for reading in the input and inserting it into the graph structure after a group discussion and Dustin went on to another class. Dustin then took the code and readjusted it to line up with the fundamental portion of the depth first search algorithm, his focus. Madison and Bradley wrote helper functions. Dustin was the primary troubleshooter; Madison worked on the printing functionality, "Read Me" document, and helped build test cases; and Bradley also worked on writing test cases and documented this report.

Algorithm Complexity

The adjacency list used to store the graph within the program takes O(1) time to add each vertex and edge and has O(|V| + |E|) in regards to overall storage. Each character space within each line is accounted for as a vertex within the graph and the number of edges is dependent on the amount of possible adjacent movements. The Depth First Search algorithm runs in O(|V| + |E|) time. To push and pop nodes on the stack each takes O(1) time. Printing the graph and freeing the malloced space in the graph both take O(|V|) time. Overall the solution runs in O(|V| +|E|) time.

A Breakdown of the Code

The program starts by error checking for correct command line arguments. If the arguments are suitable for a proper run, then the file is opened, and error checked for a proper file open. If the file opens successfully, then the graph is to be created. This is done by reading each character of the file and assigning it a state to keep track if the character is a wall or obstacle, moveable space, or robot starting point or end point. Each moveable space, robot, and endpoint is added to an adjacency list, the method by which we keep record of the graph. The number of characters between each new line character keeps track of each character's position as a node within the graph. This positioning system is then used to calculate which spaces are

available for the robots to access and extends an edge between such vertexes. Error checking is also done to ensure that a character for each robot and edge exists. If one does not, the program prints a message to alert the program user of such error. Once the graph is correctly constructed, the file is closed, and error checking occurs to ensure such. An iterative Depth First Search utilizing a stack over recursion in order to prevent stack overflow errors. Pairs of occupied spaces are tracked until each corresponding robot is in its correct finishing spot unless such solution does not exist, and all combinations have been tested. Then it is determined and displayed that there is not a proper solution. If there is a proper solution, it is printed in accordance with the answer given to the printing prompt. The remaining blocks of memory are freed before exiting the program.