CS 3050: Group Project

CS 3050

November 6, 2014

DUE: December 11

Topic: Maze Traversal using Graph Searching

Description

The purpose of this project is to utilize and build upon your knowledge in graph searching algorithms. In robotics, it's common to traverse through rooms or buildings and gather data. To do this the robot traverses through, using graph searching, until it finds the way out. In this project you will be implementing a maze traversal algorithm which finds the most direct route from any starting and stopping point. The algorithm needs to use a backtracking algorithm to step back through the maze after it's visited certain parts. An example maze that you should expect will be similar to the input file below.

Implementation

You will develop a graph traversal algorithm which finds a path from the start (S) to the end (E) points. The algorithm can only move one spot per movement (use backtracking when going to previous spaces). This means that it can't teleport to other parts of the maze. You cannot assume dimensions of the file as each line of the maze may be sized differently than other lines.

Some constraints of the program include:

1. The program can be written in C or C++.
2. Use a Makefile to compile and run the program.
3. Include a README file to tell the TA's how to run your program.
4. Your algorithm must work on babbage.
5. Always check for bad input files.
6. At the end of the program, show the final route from start to finish.
7. You may not use any graph theory libraries to solve this program.
8. All graphing algorithms should be implemented by hand.
9. Produce an error if

In the graph the characters mean the following:

1. # Wall or obstacle
2. Space: Empty spot that the program can move to.
3. S: Starting position in the maze.
4. E: Exit position in the maze.
5. There will be only one starting and stopping point.

Due Dates

This project is split into two parts.

December 1^{st} will be the first submission and presentation. This will cover reading the maze into a graph and getting started with the traversal algorithm.

The final submissions and presentations will due December 10^{th}.

Submission

This project can be done alone or in a group of 2 or 3 people.

You should place all your submission material into a folder named as follows: *pawprint1_pawprint2_pawprint3_project*. Where the *pawprint1_pawprint2_pawprint3* is the list of you and your group member pawprints. To prepare your submission, you must remove any data files and object files and executable from the folder.

You will submit one file, a tar-ball of the directory containing all the source code and appropriate Makefile(s). The naming convention should be: *pawprint1_pawprint2_pawprint3_project.tgz*
Use

```
tar czvf pawprint1_pawprint2_pawprint3_project.tgz pawprint1_pawprint2_pawprint3_project
```

YOU MUST RUN *make clean* PRIOR TO SUBMITTING THE PROGRAM. Submission of object files or program executable will result in deducted points.

Input File

```
            ###################
            #       #         ####################################
            #   #  E#                                            #
            #   #####         ################################## #
            #                 #                                # #
############################  #################                # #
#                                             #                # #
#       ##################                    #                # #
#                        #####                #                # #
#      ###############       #                #                # #
#      #                     #                #                # #
###################  ##########################                # #
                #                       #                      # #
                #             #     #   ######################## #
                #             #     #                            #
                ############  #     #   ##########################
                #             #     #   #
                #             #######   #
                #             #     #   #
                #             #     #   #
                #             #     #  S#
                #########################
```