Megan Perry

SWDV 610-Data Structures and Algorithms

Final Project

Magic BFS Maze Solver

4/27/2022

First, a few tips for running this program:

This program utilizes some module imports:

-“coolQueue” is made my me-I will include the file.

-“Graphics.py” is widely used in the university classes and is promoted in textbooks. This file can also be installed on your computer via command line or into Thonny (I have tried both methods). I will also include the file. I will include resources to help with this in this document.

-“time” is included with Python.

The Magic BFS Maze Solver utilizes the Breadth-First Search algorithm to find a target. It does this by starting at an element, and exploring the next, adjacent elements, and then the next adjacent elements to those, and so on-one “level” at a time. All of the paths are explored in a step wise fashion instead of one possible path at a time. If there is a solution, this is a great algorithm that ensures that you will actually reach it. The queue data structure is utilized with BFS search, due its the appropriate First In First Out behavior.

This project not only performs a BFS, but utilizes graphics so that the user can actually see the search occurring, as the different coordinate nodes are traversed and explored in a step-wise, per level fashion. When the end point is found, the program stops and the shortest/fastest (since edges aren’t weighted, it is possible to see what is shortest/quickest by simply seeing what took the least amount of steps/levels to reach the end) path is highlighted.

I have included two premade mazes that have different start and endpoints to quickly demonstrate the capabilities of this program. It can however, accept any maze that is formatted properly. The program can be exited by clicking on the maze window or by using the red “x” box at the corner.

This program has many improvement possibilities. Here are some that I would consider in the future:

-Creating a program to automate maze creation and also accept more user interaction/input with making mazes to run.

-Better graphics that are more attractive. Due to time constraints, I utilized the simplest graphics module that I know, but I did attempt Pygame first, before realizing that I needed a lot more time to learn and produce something reasonably good with that.

-The knowledge from this project and BFS could be built upon for other, more significant projects, such as for transportation routing/GPS systems.

Here are some resources for the Graphics.py module:

We used this module in SDWV600 Intro class. At that time I had to install it in order for it to work. For some reason it didn’t work when it was just in the same folder. It did work for me for this project without full install.

<https://mcsp.wartburg.edu/zelle/python/graphics.py>

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generatedOr in Thonny: