Analyzing Path finding Algorithms

COMP 4202 Project Report

Dawson Theroux (SN: 101106602)

Due: December 6th 2022

Introduction

Algorithms and graph theory are some of the fundamental pillars of Computer Science. Understanding how algorithsm work and their running times can be simple at times from a general point of view, but it can sometimes be hard to fully visualize what they are doing on a real dataset. The goal of this project was to implement various path finding algorithms used by route planning apps such as google maps to see analyze their average running times on real world data along with their search areas. The algorithms that where implemented and tested where Dijkstras Algorithm, A\* search with Euclidian distance heuristic, A\* search with Manhattan distance heuristic, Breadth Frist Search, and Depth First search. These algorithms where implemented in python and analyzed on a road network dataset for downtown Ottawa, downtown Toronto, and Manhattan. Although the results of this analysis where predictable from the start, it was very interesting to see the search area created by each of the different algorithms, and how long they each took with respect to each other on real world road data.

Implementation

Since the implementation was meant to use real world road and street data, my first task was to generate road and street data. To save computing time, I used ArcGIS to get road data and trim it down to a specific area of interest. This helped me limit the number of nodes and edges in the graph to save computer resources. Converting the shapefile data to python graph data was done using the NetworkX package. This package can take in shapefiles and with some modification can create a Graph object which can be used to run the path finding algorithms.

Although there exists many python packages that can run the various path finding algorithms talked about in this project, I was forced to implement them by hand in order to get the necessary data. The standard Dijkstras, A\*, and BFS algorithms are not able to return the search area as part of their default implementation. To mitigate this, I Implemented the algorithms myself and added an extra NetworkX Graph object which adds all explored edges and nodes. This graph can then be displayed along side the shortest path in order to show the search area of all of the algorithms.

Displaying the output was done using the MatPlotLib library for python. This library is a standard python library used in many projects to plot all kinds of graphs. Firstly, all of the graph data is projected onto the plane. Then the explored area is drawn on top of that in a different colour to show the search area of the algorithm. Finally the shortest path is displayed to show the shortest path found by the algorithm. (ADD THE COLOURS THAT THINGS ARE BEING DISPLAYED AS.)

Algorithms Analyzed

DFS:

BFS:

A\* (Euclidan and Manhattan Distance)

Dijkstra’s

Findings

Show the graphs that are generated. One for Ottawa and one for Toronto and one for manhattan. Also analyze execution time.

Talk about non consistent A\* heuristics

Talk about how DFS does not always find the optimal path (Which is known).

Insights

Talk about best case big O and worstcase bigO