Team: Electrical and Computer Engineers

CS225 Final Project SP 2022

March 25th, 2022

INTRODUCTION AND LEADING QUESTION

Throughout the past few decades, the rise in use of vehicles has become indispensable in major cities, resulting in the expansion of roads networks and routes. This resulted in the increasing complexity and difficulty of deciding on the best route to take whilst minimizing time. This problem is relevant within two instances: the most common scenario of a human deciding on the fastest route to take between two points on the map, but more recently applicable to autonomous vehicles.

A question then arises: given an undirected dataset of a specific area's road network and assuming ideal conditions (no traffic, obstacles, limitations, and uniform speed all throughout), what would be the best algorithm (from among breadth-first search and Dijkstra's algorithm) to deduce the fastest route between two points (road intersection) on a map?

To further explore this question and land at an accurate answer/solution to the above problem, we have decided to use the California Road Network[1] as our dataset as it provides an undirected network with nodes representing road intersections and edges representing roads connecting these intersections. The dataset provides this data as a set of two columns that depict the nodes that have an edge between them. This would be a great dataset to base our investigation on as it provides a real-world location that embodies the problem we're addressing with regards to complexity of routes and traffic, but it also stands as an exciting challenge as it requires our team to solve the challenging problems of suitably processing, storing, visualizing such a dataset in a suitable way to implement our algorithms and answer our leading question. As for algorithms, our decision to use breadth-first search and Dijkstra's algorithm to attempt to answer our leading question as well as force-directed graphs to allow us to represent our data stems from our knowledge from previous units as well as our research on the topic as these algorithms have been hypothesized to be effective in finding the shortest path between two nodes of an undirected graph, thus proving relevant to our leading question. After first processing and storing our data in a suitable manner, we will use these algorithms alongside a counter for how many nodes have been traversed to find the diameter (shortest path) between

two nodes as well as a way to track the elapsed time (an std module), comparing the time it took for each algorithm to land at the answer.

DATASET ACQUISITION AND PROCESSING

- **Data format**:

  We acquired the dataset from Stanford Large Network Dataset Collection. The dataset represents a road network in California. Nodes represent intersection and endpoints, and undirected edges represent the roads connecting these intersections or road endpoints. The input format of the dataset is in .txt. The dataset includes 1,965,206 nodes and 2,766,607 edges. We plan to use all of the data.

- **Data correction**:

  The dataset does not provide the length of edges, meaning that we cannot know the exact length of road that connects intersections or road endpoints. However, since we are assuming the ideal condition of roads (no traffic, obstacles, limitations, and uniform speed all throughout), the shortest path would simply be the least number of edges or nodes between two paths.

  However, for safety measures, a correction step we will take is making sure each row has two non-empty columns, and each of the two columns have the correct data type (integers) to avoid issues with empty/missing data-points, skipping to the next row if such an instance occurs.

- **Data storage**:

  The dataset will be stored in a graph. Since each node (intersection) has a varying number of edges (roads connecting to other intersections), we will need to define a graph implementation to suit our investigation. There will be unique data types that are created to represent the various elements of the graph such as edges and nodes. The struct for nodes will contain the node ID and the number of edges. The struct for edges will contain a pointer to a previous node and the next node. The space complexity for this graph will be O(V+E) where V is the number of vertices and E is the number of edges.

IMPLEMENTATION PLAN

We plan to use force directed graph drawing to represent our data, then use breadth-first search and Dijkstra's algorithm as means to find the diameter (shortest path) between two nodes. Highlighted below are more details for each algorithm:

- Breadth First Search (covered in class)
    - **Input**: nodes and edges given in the .txt file
    - **Output shown**: using Force directed graph drawing
    - By using force directed graph drawing, it would show the path from the start to the destination.
    - **Function efficiency**: Targets on O(V + E); V = vertices(nodes) on graph, E = edges on graph [2]
- Dijkstra's Algorithm (covered in class)
    - **Input**: nodes and edges given in the .txt file
    - **Output shown**: using Force directed graph drawing
    - By using force directed graph drawing, it would show the path from the start to the destination.
    - **Function efficiency:** Targets on O(V^2); V = vertices(nodes) on graph [3]
- Force directed graph drawing (not covered in class)
    - **Input:** file containing relevant data (may need to be modified to different file type, possibly a csv)
    - **Output shown:** graph
    - **Function efficiency:** Targets on O(n^3); n = number of nodes on graph

TIMELINE

Our final deliverable goals are as follows:

- A visualization of our dataset represented as graph of all the nodes and edges
- A set of functions that find the shortest path using BFS
- A set of functions that find the shortest path using Dijkstra's Algorithm
- A function/ set of functions that color the shortest path
- A table and graph that highlights the time it took for each function to land at the shortest path
- Unique data structures representing the graph and its elements (edges,nodes,etc.)

Planning ahead, we divided our investigation into several major milestones, setting a proposed timeline for each task.

- Data acquisition, processing and visualization: **(April 3, 2022)**
    - Establish project directory, foundation, and major files
    - Access dataset successfully
    - Come up with methods to store and visualize data
- Finish Force-Directed Graph that represents all the data **(April 10, 2022)**
- Finish 1st Shortest Path Algorithm (BFS or Dijkstra's) **(April 24, 2022)**
- Finish 2nd Shortest Path Algorithm **(May 3, 2022)**
- Finish Presentation + Report **(May 6, 2022)**

WORKS CITED

[1] California Road Network Dataset: https://snap.stanford.edu/data/roadNet-CA.html

[2] Force Directed Graph Efficiency: https://en.wikipedia.org/wiki/Force-directed_graph_drawing

[3] Dijkstra's Algorithm Efficiency: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm