CS 225 Data Structures

Project Proposal (jcampo37-ahuert30-psalin2-samuild2)

**Leading Question**

As a member of Code for Venezuela, I came across a problem where there was a use case of finding the best route between two cities, optimizing for places where there is gasoline available. Crazy? Well, Venezuela is under a severe scarcity of all kinds of resources.

**Dataset Acquisition and Processing**

Code for Venezuela’s [datasets](https://data.humdata.org/organization/stats/c4v) were among the most downloaded on HDX in the third quarter of 2021. HDX is an open platform for sharing data across crises and organizations, built by OCHA. Many really large organizations use it to operate better. Many established and renowned organizations such as the United Nations, UNHCR, iMMAP, and many more are using Code for Venezuela’s datasets to inform large humanitarian aid projects related to Venezuela.

We will be using the [Transportation Access & Mobility](https://data.humdata.org/dataset/open_monthly_transportation_access_mobility) dataset, which is data crowdsourced monthly from Venezuelans through the Premise Data mobile application. The data tracks transport accessibility and mobility indicators at a monthly granularity. We will download it through the website, store it on our computers, and to start the processing, we will start identifying and handling the missing or erroneous values.

The data format is easy to work with as it’s a CSV file with a row per each survey response. How the Transportation Access & Mobility dataset data will be used in the graph data structures will be used to provide the information about the given graph. The queries for the graph database could be like below:

* Get the list of all nodes, edges and attributes of the graph. Find the nodes and edges which are connected to the node. Then, for each node, find the neighbors by all possible paths. How many neighbors does it have?
* Get the distance between the two nodes.
* Get the shortest path between two nodes.
* Get the Euler tour (or Hamiltonian circuit) for the graph.
* Get the connected components (or components) for the graph.

**Graph Algorithms**

First, the shortest distance algorithmwill be implemented. This algorithm will be performed only once by finding the shortest distance between all pairs of nodes in the graph. Then, we will implement the single-source shortest path algorithm. This algorithm will be performed by finding the shortest distance between each node in the graph to the source node. Finally, we will implement the all-pairs shortest path algorithm, which will be performed by finding the shortest distance between each node in the graph to all other nodes in the graph. **What are the inputs you expect each method to take in?** The inputs for each algorithm will be the graph data structures, which can be implemented as an adjacency list or an adjacency matrix. Each method will also be passed the starting node for the traversals. What is the output you expect each method to return? The output for each algorithm will be the shortest distance between each pair of nodes in the graph data structure. How will you test your algorithms? As an example, we will add a method to add a node to the graph and compare the results of the shortest distance between the new nodes to the other nodes in the graph. How will you test your code for correctness and completeness? We will test the algorithms by adding nodes and checking the results of each of the algorithms. How will you test your code for performance? We will test the performance of the algorithms by adding nodes and checking the results of each of the algorithms. How will you prepare your code for production? We will add a method to add a node to the graph and compare the results of the shortest distance between the new nodes to the other nodes in the graph. All of the mentioned test will be done through test cases and manual testing alike.

**Timeline**

11/18 - Learn to interpret the data correctly (CSV) to represent it in a Graph.

11/25 - Write out the strong base to start with for algorithms that will be used.

12/2 - Implement the algorithms into the data correctly.

12/9 - Test and debug

12/13 - Revise and polish code if necessary.