CS 575 Project Proposal

Team Members:
Sean Kunz

Problems:
I plan on working with single-source shortest path algorithms and self-balancing binary search trees. Specifically, I am looking to implement the A* Search Algorithm and a red-black tree.

Why these problems are important/interesting:
I will be applying these algorithms to a real-world situation. The graph the search algorithm will be working with will consist of all metropolitan statistical area principal cities [1] in the United States. The user will be able to input a source and destination city (e.g., New York City and Houston) and identify the shortest path between the two. By running the shortest path algorithm between many sets of large city pairs, we can develop a network around the country that can be used, for example, to determine if the Interstate Highway or railroad system is optimized.

The red-black tree will exist as a separate and intertwined implementation. The red-black tree will store the nodes from the previous graph and will include information such as urban area names, population, and IDs. Users will be able to query the tree in search of this information, as well as be able to add urban areas previously not included or remove cities they do not want to consider. The tree will also be used for storage as part of the shortest path problem.

Why these problems are significant/non-trivial:
I plan on implementing the shortest path algorithm with an additional weighting system that adds distance and population considerations to all possible edges. Furthermore, the creation of the graph will be non-trivial. I will need to geocode [2] all urban areas (~1,500) to obtain latitude/longitude data and match it with characteristic data. Once I have all latitude/longitude coordinates, I will need to generate edge data via distance calculations. Next, I will visualize the results by creating a website that uses the ArcGIS JavaScript API [3]. This will add an additional layer of interactivity that will allow the user to fully grasp what the shortest path between two cities would look like. Lastly, the red-black tree implementation will be fairly traditional, but will be integrated into the shortest-path algorithm problem, as well as existing as a standalone command-line program.

Why it is doable:
I plan on starting early and having the algorithm implementation done by the end of March. After that, I will work on the visualization task and creating any necessary presentations. I plan on writing most of the code in Go, with some code written in JavaScript for visualization

purposes. I don’t have much experience with the APIs I’m planning to use, but I am confident that I will be able to figure them out.

References:
[1] - https://en.wikipedia.org/wiki/List_of_United_States_urban_areas
[2] - https://developers.arcgis.com/rest/geocode/api-reference/geocoding-find-address-candidates.htm
[3] - https://developers.arcgis.com/javascript/latest/