CS575: Final Project Report

**Project Title: Viewing U.S. Urbanized Areas Through the Lens of Data Structures and Algorithms**

**Team Member(s): Sean Kunz**

# Problem

In this project, I tackled two problems. The first of these problems involved managing user data. User data in this case was a list of all urbanized areas in the United States [ref here]. Overall, almost 1,500 entries are included. This dataset can be used by many different specialists, such as transportation planners, demographers, and spatial analysts. While not quite as large as some other datasets, any code using this data could be improved by using a faster query method than iterating over a list. To accomplish this goal, a self-balancing red-black tree can be used. A red-black tree can improve query times by limiting the depth of a tree through balancing techniques.

The second problem I solved is finding the shortest path between two points. There are many different ways of doing this, with most solutions falling into two categories: single-pair shortest path algorithms and all-pairs shortest path algorithms. In this project, I worked towards modeling an ideal form of the United States Interstate Highway System, which is best suited to a single-pair shortest path algorithm, as all-pairs shortest path algorithms would include smaller areas that don’t necessarily need highway/train stops, resulting in unnecessary computation. I implemented the A\* search algorithm to assist in this modeling, where users can input any two cities as input and receive a path between the two as an output.

Lastly, I needed to obtain data for all of the nodes and edges for the graph used in the shortest path algorithm. I also used the node data as an input to the red-black tree. I utilized several APIs and algorithms such as Euclidean distance to accomplish this task.

# Algorithms

When working on this project, I focused on implementing the red-black tree data structure and the A\* Shortest Path Search Algorithm. I derived my implementation for the red-black tree from Cormen’s Algorithm book [1], Wikipedia [ref here], and the University of South Florida [ref here]. I used Cormen’s Algorithm book and Wikipedia for the A\* Shortest Path Search Algorithm implementation.

## Red-Black Tree

A red-black tree is a form of a self-balancing binary search tree. In the act of self-balancing, the tree can maintain close to ideal search conditions. To do this, the tree marks each node as either red or black while inserting and deleting from the tree. These nodes can change color throughout the process.

There are four properties that must be maintained by the tree: each node is either red or black, all NIL values are considered black, a red node does not have a red child, and every path from a given node to any of its descendants goes through the same number of black nodes. These properties are maintained through six different insert cases and six different delete cases.

## A\* Search Algorithm

The A\* Search Algorithm is a single pair shortest path algorithm. The algorithm finds the shortest path by maintaining a set of paths originating from the start until it finds the destination node. In this way, it is fairly similar to Dijkstra’s Shortest Path Algorithm, but differentiates itself from the use of heuristics. The heuristic simply adds a value to the already calculated distance to further estimate cost. All nodes at the frontier of the search are stored in a priority queue and the node with the smallest distance + heuristic value is taken from the queue.

# Software Design and Implementation

/\* Briefly describe how you designed and implemented your software. Also, describe which tools you used and which parts you implemented yourself from scratch. \*/

## Software Design

## Implementation and Tools Used

## Performance Evaluation (Optional)

/\* If you have compared the performance of several algorithms, describe 1) performance metrics (e.g., latency or throughput) and 2) results using the defined metrics. To illustrate your results, plot graphs or tables and clearly explain them. Note that performance evaluation is not required but optional, since correctly implementing N+1 non-trivial algorithms is the first priority. If you haven’t done any performance evaluation just leave this section empty. \*/

# Project outcomes

* Turn in your source code through Brightspace
* Provide the link (URL) to your YouTube video here
* Provide the link to your presentation slides here

##### References

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, Introduction to Algorithms, Third Edition, The MIT Press, 2009.
2. Michael Ernst, How to Write a Technical Paper, <https://homes.cs.washington.edu/~mernst/advice/write-technical-paper.html> .

Provide all references and clearly specify whether you have used any existing implementations (in this case, clearly describe which parts you have extended and why your extensions are non-trivial) or you have implemented your project from scratch to avoid plagiarism. We will use automated tools that detect plagiarism across online/offline sources.