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

/\* In each subsection below, briefly describe how each algorithm works in one paragraph and cite each of them (provide a reference at the end of this template). For example, if two students worked together and implemented the dynamic programming, backtracking, and branch-and-bound algorithms for 0/1 knapsack [1], briefly describe them in II.A, II.B, and II.C below, and cite Cormen’s book [1]. (This is just an example. Replace them with your algorithms.) \*/

## Dynamic Programming for 0/1 Knapsack

## Backtracking for 0/1 Knapsack

## Branch-and-Bound for 0/1 Knapsack

/\* The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. Please do not revise any of the current designations. \*/

# 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.