C950 Task-2 WGUPS Write-Up

Dennis J. Pothier

ID #011690569

WGU Email: dpothie@wgu.edu

19 July 2025

C950 Data Structures and Algorithms II

# A. Hash Table

A computer screen shot of a program

Description automatically generated

# B. Look-Up Functions

A computer screen shot of a program code

Description automatically generated

# C. Original Code

# C1. Identification Information

A black rectangular object with a black strip

Description automatically generated

# C2. Process and Flow Comments

A screenshot of a computer program

Description automatically generated

A computer screen shot of a program

Description automatically generated

A screen shot of a computer program

Description automatically generated

# D. Interface

A computer screen with white text

Description automatically generated

# D1. First Status Check

A screen shot of a computer

Description automatically generated

# D2. Second Status Check

A screenshot of a computer screen

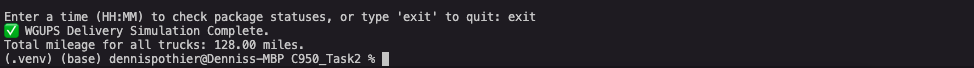
Description automatically generated

# D3. Third Status Check

A screenshot of a computer screen

Description automatically generated

# E. Screenshot of Code Execution



A computer screen shot of a program code

Description automatically generated

A computer screen shot of a program code

Description automatically generated

A screen shot of a computer program

Description automatically generated

# F1. Strengths of the Chosen Algorithm

The chosen algorithm is considered a perfect choice due to several different features. Using the nearest neighbor, the program could run in O(n2) time, which for 40 packages required just a few milliseconds. The algorithm is also adaptable to changes in the package data as they occur, as exemplified by package 9.

# F2. Verification of Algorithm

The algorithm complies with all requirements and criteria while rapidly delivering the packages. This algorithm got all packages delivered on or before their deadlines, and in some cases with significant improvements. All the package delivery was done at 128 miles, well under the 140-mile limit.

# F3. Other Possible Algorithms

Two other options would be Depth-First-Search and Dijkstra’s Shortest Path algorithms instead of the Nearest-Neighbor. Both would be great options to use in this scenario.

# F3a. Algorithm Differences

Depth-first search is exhaustive and relies on backtracking to work and pruning to stay practical, whereas nearest neighbor relies on being purely greedy and not revisiting earlier choices. Dijkstra’s, by contrast, isn’t a tour builder at all. It focuses on edge weights and defers route construction to a secondary heuristic. These solutions trade off the simple computational time required for nearest neighbor with being slightly more computationally exhaustive. Both would solve the problem, but they would add computational time, so it would run slower when there are much larger datasets.

# G. Different Approach

A more novel approach to this would be refining the data in the source to incorporate codes for each exception expected for packages. I.e., delayed would be D, wrong address would be W, and a grouped package or required truck number would be assigned G[13,14,15,16,17,19,20] and T[2], respectively, so that the program could handle these inputs programmatically. By doing this, the algorithm would have to be tweaked slightly to assign the packages to trucks, but it would reduce the need for the truck assignments to be modified with each different constraint daily. Also, factor in the weights of the packages because of fuel economy to get the most cost-effective solution.

# H. Verification of Data Structure

The solutions chaining provides near-constant time inserts and look-ups, making it a non-factor in algorithm complexity overall. It conforms to all the requirements of the problem.

# H1. Other Data Structures

Two suitable replacements for the custom hash table would be an open-addressing hash table with linear probing or a red-black tree.

# H1a. Data Structure Differences

The open-addressing design stores the entries in a bucket array and resolves the collisions by probing subsequent slots. This eliminates pointer overhead and possibly introduces primary clustering, which requires tombstone handling during deletions. A red-black tree would maintain the keys in a sorted order and guarantee O(logN) performance for all operations, which is beneficial to range queries. This does come at a cost for single-key look-ups and more complex rotation logic compared to the constant-time hashing of the given solution.