C950 Task-2 WGUPS Write-Up

(Task-2: The implementation phase of the WGUPS Routing Program).

(Zip your source code and upload it with this file)

Dustin Snelgrove

ID # 011633071

WGU Email: dsnelg1@wgu.edu

Date 08/19/2025

C950 Data Structures and Algorithms II

# A. Hash Table

A screen shot of a computer

AI-generated content may be incorrect.

# B. Look-Up Functions

# A screen shot of a computer program AI-generated content may be incorrect.A computer screen shot of a program AI-generated content may be incorrect.

# C. Original Code

# A screen shot of a computer AI-generated content may be incorrect.

# C1. Identification Information

A screenshot of a computer

AI-generated content may be incorrect.

# C2. Process and Flow Comments

A black background with colorful text

AI-generated content may be incorrect.A computer screen shot of a program code

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

# D. Interface

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

# D1. First Status Check

A screenshot of a computer screen

AI-generated content may be incorrect.

# D2. Second Status Check

A screenshot of a computer

AI-generated content may be incorrect.

# D3. Third Status Check

A screen shot of a computer

AI-generated content may be incorrect.

# E. Screenshot of Code Execution

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# F1. Strengths of the Chosen Algorithm

I used a greedy nearest neighbor with deadline tie-breaks.  
Strengths:

* It keeps routes short by always choosing the next closest feasible stop while favoring earlier deadlines.
* It adapts at each step using package fields like *earliest\_ready, truck\_only,* and *group\_id*, so delays and constraints are handled in real time.
* It is simple and runs in polynomial time, which makes it fast and predictable for this problem size.

# F2. Verification of Algorithm

My latest run produced total miles **82.7**, which is under the 140-mile limit.  
All 9:00 and 10:30 deadlines are met. Package 9 is delivered after the 10:20 address correction time.  
Truck-2-only packages ride on truck 2. The group {13, 14, 15, 16, 19, 20} travels together. Delayed packages do not load before 9:05 AM. The interface lets a user enter a time to view per-truck statuses as at the hub, en route, or delivered with delivery time, plus total miles.

# F3. Other Possible Algorithms

 Clarke–Wright savings heuristic for VRP.

 Simulated annealing to improve routes.

# F3a. Algorithm Differences

Clarke–Wright forms routes by merging stops that produce the largest distance savings. My approach grows each route locally from the current position and deadline rank.

Simulated annealing explores route permutations and can accept temporary worse moves to escape local minima. My approach is deterministic and greedy, not stochastic.

# G. Different Approach

If I did this again I would add a short 2-opt improvement pass per truck to remove route crossings, use a small nearest-neighbor cache to shrink the inner loop, and move all hard-coded constraint sets into a simple config block. I would also add small unit tests for the hash table and the time math.

# H. Verification of Data Structure

The solution uses a custom hash table with separate chaining and resizing keyed by package ID. Each record stores address, deadline, city, zip, weight, status, time\_loaded, and time\_delivered. Lookups and updates are average O(1), which supports the router and the status interface. The same table type maps address text to distance indices.

# H1. Other Data Structures

* Balanced binary search tree keyed by package ID.
* Sorted array of package records with binary search on package ID.

# H1a. Data Structure Differences

The tree guarantees O(log n) insert and lookup and maintains ordering, but it is slower than average O(1) hash lookups and uses more pointer overhead. The sorted array gives O(log n) searches but O(n) inserts or resorting when records change; it is cache-friendly but less flexible when package assignments change during the day.

# I. Sources

I did not directly quote any of the resources I used but the main learning resources I used are below:

*The Traveling Salesman Problem (TSP). WGU*

<https://www2.seas.gwu.edu/~simhaweb/champalg/tsp/tsp.html>

*C950: Data Structures and Algorithms II. zyBooks.*

<https://learn.zybooks.com/zybook/WGUC950Template2023>

*Know Thy Complexities! Big-O Cheat Sheet.*

<https://www.bigocheatsheet.com/>