WGUPS Routing Program - Task 2 Write-Up

Student Name: Joao Marcelo Martins Miranda

Student ID: 010548237

Email Address: jmir142@wgu.edu

Date: 01/10/2025

Stated Problem:

The goal of this project is to develop an optimized routing program for the Western Governors University Package Service (WGUPS). The application aims to compute the most efficient delivery routes while adhering to specific constraints:

* A maximum total mileage of 140 miles across all trucks.
* Two trucks and three drivers are available for deliveries.
* Each truck can carry a maximum of 16 packages.
* Package constraints, such as delivery deadlines and delayed addresses, must be respected.

The project employs Python as the primary programming language to design, implement, and test the routing solution.

Algorithm Overview:

The Nearest Neighbor Algorithm was selected to optimize delivery routes. This algorithm prioritizes the shortest distance from the truck's current location to the next destination. It ensures that the total mileage remains within the constraints while efficiently sequencing deliveries.

Implementation Steps:

1. Initial Setup:

* Each truck is loaded with packages based on constraints (e.g., deadlines, delays).
* The starting location for each truck is the hub.

1. Route Optimization:

* For each truck, the algorithm identifies the package with the closest destination to the current location.
* The truck delivers the package and updates its current location.
* The process repeats until all packages are delivered.

1. Dynamic Updates:

* Package statuses are updated in real-time (e.g., "At Hub," "En Route," "Delivered").
* Address corrections (e.g., package #9) are handled dynamically during runtime.

Pseudocode

**function deliver\_packages\_nearest\_neighbor(truck, package\_hash\_table):**

**current\_location ← truck's starting address**

**current\_time ← truck's departure time**

**while there are packages in truck:**

**nearest\_package ← find package with shortest distance to current\_location**

**remove nearest\_package from truck's package list**

**add nearest\_package to delivered packages list**

**distance ← calculate distance from current\_location to nearest\_package's address**

**travel\_time ← distance / truck's speed**

**current\_time ← current\_time + travel\_time**

**truck's mileage ← truck's mileage + distance**

**current\_location ← nearest\_package's address**

**update nearest\_package's status to "Delivered"**

**set nearest\_package's departure time to current\_time - travel\_time**

**set nearest\_package's delivery time to current\_time**

Time Complexity

Worst Case: O(n^2), where is the number of packages on a truck.

Best Case: O(1), if the truck is empty.

Space Complexity

Overall: O(n), for maintaining the package list.

Programming Environment:

Integrated Development Environment (IDE):

* PyCharm Community Edition 2024.3.1.1

Python Version:

* Python 3.13.1

External Libraries:

* csv (for data parsing)
* datetime (for time tracking)

Hardware:

* Acer Laptop (Intel Core i5, 8GB RAM, 128GB Storage)

Data Structures (Rubric Requirement A)

Hash Table

The primary data structure used is a hash table to store package information. Each package is indexed by its unique ID for quick access. The hash table efficiently supports insertion, deletion, and lookup operations, ensuring optimal performance.

Implementation

The hash table is implemented in the CreateHashMap class, which uses a custom hash function for indexing. Chaining is employed to handle collisions.

Strengths:

* Constant-time lookups ().
* Ideal for managing package data with unique identifiers.

Weaknesses:

* Potential for collisions, mitigated by using chaining.

Alternative Data Structures

1. Graphs: Useful for representing distances as edges and locations as vertices. This could enhance scalability for larger datasets.
2. Binary Search Trees (BST): Suitable for sorting packages based on attributes like deadlines or addresses.

Verification and Strengths of the Algorithm (Rubric Requirements B4, F1, F2)

Strengths:

1. Ensures all packages are delivered within the 140-mile limit.
2. Efficiently accounts for constraints, such as delivery deadlines and delays.
3. Scales well with the given problem size ( packages per truck).

Verification

The algorithm meets all requirements by:

* Delivering all packages within their deadlines.
* Keeping the total mileage below the maximum allowed.

Alternatives

1. A-star Algorithm: Uses heuristics to improve route selection.
2. Dijkstra's Algorithm: Finds the shortest path in weighted graphs but may be computationally expensive for large datasets.

Scalability and Efficiency (Rubric Requirement C4)

While the current solution is designed for the task constraints, scalability can be improved by:

* Automating truck loading based on package attributes.
* Using advanced data structures like graphs for distance optimization.
* Refactoring code for modularity and ease of maintenance.

Interface (Rubric Requirement D)

A command-line interface (CLI) was implemented for:

* Viewing package statuses at specific times.
* Displaying total mileage for all trucks.
* Simulating the delivery process.

Sources

* Lysecky, R., & Vahid, F. (2018, June). C950: Data Structures and Algorithms II. zyBooks. Retrieved from <https://learn.zybooks.com/zybook/WGUC950AY20182019/>
* <https://www.bigocheatsheet.com>
* Madakor, J. (2021, Oct). WGU Data Structures and Algorithms 2 - WGU C950 (A Step by Step Guide). YouTube. Retrieved January 2, 2025, from <https://www.youtube.com/watch?v=lHeVDmgpKy4>

**Note:** Screenshots of code, hash table, and interface will be provided in a separate folder. The Python project folder, screenshots, and this write-up will be included in the submission. Screenshots will have descriptive names to help.