WGUPS Routing Program Planning

WGU Student ID	012201560

A. Identify an Algorithm

For the WGUPS Routing Program, I will be using the Nearest Neighbor algorithm. This greedy heuristic algorithm is well-suited for solving vehicle routing problems efficiently.

B. Identify a Data Structure

A hash table would be an appropriate self-adjusting data structure to use with the Nearest Neighbor algorithm for storing package data (Bian et al., 2020).

1. Explanation

The hash table can store key-value pairs where the key is the package ID, and the value is a Package object containing all relevant package information. This allows for O(1) average time complexity for insertions, deletions, and lookups of package data (Bian et al., 2020). The hash table can dynamically resize to accommodate a growing number of packages.

C. Program Overview

1. Nearest Neighbor Algorithm Pseudocode

```
function nearest_neighbor(packages, start_location):
    unvisited = set of all package delivery locations
    current_location = start_location
    route = empty list

    while unvisited is not empty:
        nearest = find closest unvisited location to current_location
        add nearest to route
        remove nearest from unvisited
        current_location = nearest

    return route
```

2. Programming Environment

- **Software:** Python 3.13, VS Code IDE
- **Hardware:** Standard desktop/laptop computer, with at least 8GB RAM and a multi-core CPU.

3a. Time Complexity

- **Hash Table Operations:** O(1) average case for insertions, deletions, and lookups
- **Nearest Neighbor Algorithm:** O(n^2) time complexity, where n is the number of delivery locations.
- **Overall Program:** O(1 + n^2), therefore, O(n^2) time complexity, dominated by the Nearest Neighbor algorithm.

3b. Space Complexity

- **Hash Table:** O(n) space, where n is the number of packages
- **Nearest Neighbor Algorithm:** O(n) space for storing the route and unvisited locations
- **Overall Program:** O(n) space complexity

3b. Space-Time Analysis

The space complexity of the program is primarily determined by the data structures used to store package information and the algorithm's working memory. The hash table storing package data will require O(n) space, where n is the number of packages. The Nearest Neighbor algorithm uses additional space for maintaining the list of unvisited locations and the current route, both of which are O(n) in the worst case.

While the time complexity is quadratic due to the Nearest Neighbor algorithm, the space complexity remains linear, making the program memory-efficient even as the number of packages increases. However, the quadratic time complexity may become a performance bottleneck for large datasets.

4. Scalability

This solution can scale to handle a growing number of packages by utilizing the dynamic resizing capability of the hash table. However, the quadratic time complexity of the Nearest Neighbor algorithm may become a bottleneck for exceptionally large numbers of packages.

5. Maintenance and Efficiency

- Modular design with separate classes for Packages, Trucks, and the Routing algorithm.
- Use of object-oriented programming for easy maintenance and extensibility.
- Well commented code to explain logic and design decisions.

6. Hash Table Strengths and Weaknesses

- **Strengths:** O(1) average time complexity for operations. Able to be dynamically resized.
- **Weaknesses:** Potential for collisions. May have a higher memory overhead compared to other data structures.

7. Key for Efficient Delivery Mangement

The package ID should be used as the key for the hash table. It is unique for each package, allowing for efficient lookups and updates of package information.

D. References

1. Bian, Z.; Huang, X.; Wang, W.; Yu, C.; Boncz, P. SAHA: A String Adaptive Hash Table for Analytical Databases. Applied Sciences 2020, 10, 1915. https://www.mdpi.com/2076-3417/10/6/1915