**NHP3 – NHP3 TASK 1**

**Ayomide Adedeji**

**Data Structures and Algorithms II – C950**

**Western Governors University**

**Student #007440467**

1. **Identify a named self-adjusting algorithm (e.g., nearest neighbor algorithm, greedy algorithm) that could be used to create your program to deliver the packages.**

**Answer:** The Nearest Neighbor Algorithm is an algorithm that could be used in my program to deliver the packages. This algorithm selects the nearest delivery location to the current position as the next stop for the delivery vehicle. This step is repeated until all delivery locations have been visited.

1. **Identify a self-adjusting data structure, such as a hash table, that could be used with the algorithm identified in part A to store the package data.**

Answer: The data structure used to store package data is the Chaining Hash table

**B1. Explain how your data structure accounts for the relationship between the data components you are storing**.

**Answer:** The hash table used for this project is influenced by the chaining hash table structure. It implements an insertion function that uses a key-value pair to store and retrieve pieces of information. The hash table for this project has an initial size of forty. The key represents one of its forty ‘buckets’, the value is the object that will be stored in said bucket. For example, to insert package p into hashtable myPackage\_hash\_table.insert(p.ID, p). This way, the unique package id (p.ID) indicates which bucket the item is stored in.

The hash table accounts for the relationship between the stored data points first by using the ID

number. This ID acts as the key that can be used to retrieve the entire package object being stored or one of its specific attributes, such as its deadline. Information about any package can be accessed using this model. Thus, the hash table accounts for all relationships between the

data it is storing.

1. **1. Explain the algorithm’s logic using pseudocode.**

**Steps:**

* Read the csv file containing distance, address, and package information.
* Create empty address list and hash table.
* Load address data into address list.
* Load package data into a hash table.
* Define function for distance between two addresses and address retrieve function.
* Create truck object.
* Load packages into trucks and sort them using the nearest neighbor algorithm.

**2.  Describe the programming environment you will use to create the Python application, including both the software and hardware you will use.**

**Answer:**

Pycharm Community Version: Version: 3.5

Python Version: Python 3.12

Hardware: Windows 11

The programming environment that I used for this application is Pycharm Community Edititon. It is a popular integrated development environment (IDE) specifically designed for python development. The Community Edition of PyCharm is free and open source, making it accessible to developers of all levels. While it lacks some advanced features available in the Professional Edition, it still offers a comprehensive set of tools for Python development.

**3.  Evaluate the space-time complexity of each major segment of the program and the entire program using big-O notation.**

**Answer:**

1. Reading CSV Files:

* Space Complexity: O(n)

The space complexity is linear because all CSV data is stored in lists (Csv\_distance, Csv\_address, Csv\_package) and each list stores all elements from the corresponding CSV file.

* Time Complexity: O(n)

Similarly, the time complexity is linear because each CSV file is read once, and the time taken to read each CSV file depends on the number of rows in the file.

1. Loading Package Data into Hash Table:

* Space Complexity: O(n)

The space complexity is linear because each package's data is stored in the hash table (myPackage\_hash\_table), which can grow linearly with the number of packages.

* Time Complexity: O(n)

The time complexity is linear because each package's data is processed once from the CSV file, and each insertion operation into the hash table takes constant time on average.

1. Finding Distance Between Addresses:

* Space Complexity: O(1)

The space complexity is constant because no additional space is allocated based on the input size.

* Time Complexity: O(1)

The time complexity is constant because the distance between two addresses is retrieved directly from pre-loaded data (Csv\_distance) using indexing.

1. Delivering Packages and Nearest Neighbor Algorithm:

* Space Complexity: O(n)

The space complexity is linear because it involves maintaining lists of packages (not\_delivered) and the number of packages processed linearly affects the space.

* Time Complexity: O(n^2)

The time complexity is quadratic because for each package, the algorithm iterates over all remaining packages to find the nearest neighbor, resulting in nested loops. This algorithm's time complexity can be optimized using more efficient algorithms like the Travelling Salesman Problem (TSP) solver.

1. User Interface (UI) and Package Status Checking:

* Space Complexity: O(1)

The space complexity is constant because no additional space is allocated based on the input size.

* Time Complexity: O(1)

The time complexity is constant because the UI and package status checking involve user interaction and processing a fixed number of inputs, which do not depend on the input size.

Overall Space Complexity: O(n)

The overall space complexity is primarily determined by the number of packages and addresses processed, as well as the size of pre-loaded data structures.

Overall Time Complexity: O(n^2)

The overall time complexity is dominated by the delivering packages segment, which has a quadratic time complexity due to the nested loops in the nearest neighbor algorithm.

**4. Explain the capability of your solution to scale and adapt to a growing number of packages**.

**Answer:** With this solution, as the number of packages and delivery points increases, the complexity of the routing problem grows exponentially. The 2-opt algorithm may become less scalable for very large numbers of packages or delivery points due to manually loading of the trucks.

5**.  Discuss why the software design would be efficient and easy to maintain.**

Answer: This program is efficient and easy to maintain because of its:

1. Simplicity: Its straightforward approach to iteratively improving routes through local search makes it easy for developers to implement and maintain.
2. Modularity: It operates independently of other components, allowing developers to maintain and update the optimization logic without affecting the rest of the system.
3. Real-Time Updates: This real-time updating capability ensures that the routes remain optimized and responsive to evolving circumstances, improving overall efficiency and customer satisfaction.
4. Ease of Maintenance: Developers can make updates or improvements to the algorithm without significant overhead, ensuring that the optimization logic remains effective and up to date with evolving requirements or changes in the delivery package system.

**6. Describe both the strengths and weaknesses of the self-adjusting data structure (e.g., the hash table).**

**Strengths of the hash table:**

1. Fast Retrieval: Hash tables provide constant-time average-case complexity for lookup, insertion, and deletion operations. This makes them highly efficient for storing and retrieving data, especially when the number of elements in the table is large.
2. Flexible Key-Value Storage: Hash tables allow the storage of key-value pairs, providing a flexible data structure for various applications. This versatility makes them suitable for implementing associative arrays, symbol tables, caches, and other data structures.
3. Space Efficiency: Hash tables typically use space efficiently, especially when the load factor (ratio of elements to buckets) is kept low. They dynamically resize themselves to maintain a suitable load factor, minimizing wasted space while accommodating a growing number of elements.

**Weaknesses of the hash table:**

1. Hash Function Dependency: The performance of hash tables heavily relies on the quality of the hash function used to map keys to bucket indices. A poor hash function can lead to hash collisions, which degrade performance and may require additional collision resolution techniques.
2. Collision Handling Overhead: Hash collisions occur when two distinct keys map to the same bucket. Resolving collisions adds overhead, either through chaining (storing collided elements in linked lists or other data structures) or open addressing (probing for alternative bucket locations). This overhead can impact performance and memory usage, especially in high-load scenarios.
3. Unordered Storage: Hash tables do not maintain any inherent order among elements. While this is suitable for many applications, it can be a limitation when ordered traversal or sorting of elements is required.
4. Space Complexity: While hash tables are generally space-efficient, they can still consume significant memory, especially when the load factor is high or when dealing with a large number of collisions. Resizing the table to maintain a low load factor can also incur additional memory overhead.

7. Justify the choice of a key for efficient delivery management from the following components:

1. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.