C950 Task-1 WGUPS Algorithm Overview

(Task-1: The planning phase of the WGUPS Routing Program)

Jennifer Murphy

ID #001532028

WGU Email: jmur326@wgu.edu

February 25, 2024

Version II

C950 Data Structures and Algorithms II

# Introduction

The routing program for this project needs to store package data in a data structure that allows for insertions and searching of data, calculates the distance between packages addresses using the information provided in CSV files and finding the best route to deliver the packages with under 140 miles total. The program will contain a user interface that will allow the user to enter a time and retrieve updated package data with multiple data points for that specific time. The total mileage used by the trucks to deliver the packages also needs to be displayed. The program will be written using Python.

# A. Algorithm Identification

I chose to use the nearest neighbor algorithm to deliver all packages efficiently.

# B. Data Structure Identification

I used a chaining hash table to store my data and work with the nearest neighbor algorithm to deliver all packages within the specified mileage and time constraints.

# B1. Explanation of Data Structure

The chaining hash table is scalable and able to handle data collisions. The table is initialized with a capacity of 50 to provide plenty of space for the number of packages that were going to be utilized for this project. The insert method uses a hash function to identify an index for the key-value pairs inserted. This specified bucket at that index holds the key-value pairs that map to its location. The chaining hash table prevents collisions by creating a list for each bucket. Additional key-value pairs that are mapped to occupied indexes are appended to the end of the list. The remove method works oppositely of the insert method. It calculates the bucket using the has function, searches the bucket list in the corresponding index and removes the key value pair if found.

The chaining hash table also includes a search method that takes the key as a parameter. The hash functions use the key value to locate the index and search the list at the specified index for a matching key value. If one is found, it returns the value. This method allows the program to easily search and return requested key value pairs.

# C1. Algorithm’s Logic

Nearest Neighbor

not\_delivered = [empty]

for packages in truck

iterate through packages in hash table with status = not delivered

add to not\_delivered list

while not\_delivered > 0:

for packages in not\_delivered:

distance to package = calculate distance between truck current address and package address

IF package distance <= closest address distance

Set package address to closest address distance

Add closest address package to truck

Remove closest address package from not\_delivered

Truck mileage += closest address

Truck.time += closest address/18 mph

Package.delivery\_time = truck.time

# C2. Development Environment

The programming environment used is an HP Spectre x360 laptop running Windows 11 Pro. The software used to develop the program consists of PyCharm using Python version 3.11. The program relies on local files to run.

# C3. Space and Time complexity using Big-O Notation

The dominant component of the program is the nearest neighbor algorithm with its space complexity of O(n) and time complexity of O(n^2). The nearest neighbor algorithm has this complexity due to key value pairs that need to be accessed being stored in the hash table and the nested loops in which the algorithm compares the address of one package to all other addresses of the packages loaded on the truck. The space and time complexity of the chaining hash table could vary. The space complexity is O(n) because the data is stored in buckets with lists. Due to the low volume of key value pairs being stored for this program, the overall time complexity is O(1). If the data grew substantially, it would increase to O(n) due to the increased load in each bucket. The function that finds the address in the CSV file has a time complexity of O(n) because the function iterates over each row in the data set and a space complexity of O(n). The amount of time it takes to find and return addresses and store the addresses increases linearly with the size of the data set being searched. The function to find the distance between two address from the matrix of data in the CSV file has a complexity of O(1) because it is simply retrieving information, but no iteration is involved, so the time to retrieve information remains constant. This function uses the distance matrix and has a space complexity of O(n^2). The space complexity is O(n) and time complexity is O(n^2) of the overall program.

# C4. Scalability and Adaptability

The program is very easily scaled based on the volume of packages. The chaining hash table’s performance remains relatively constant despite the volume of package data stored. It is easily expanded by adding buckets and altering the hash function. The nearest neighbor algorithm would also be able to handle an increase in package volume without sacrificing efficiency. The time complexity would not greatly increase with a package volume increase.

The program has classes for the hash table, truck, and packages. This allows changes to be made easily and simple additions of more complex functionalities. Dynamic updates are provided of packages status that allows monitoring and adjustments from users.

# C5. Software Efficiency and Maintainability

The software coding is minimal and concise making maintenance easy. The classes are clearly defined and separated into their own files to make an editing straightforward. The number of functions is also low, making maintaining and editing easy. There are many explanatory comments throughout the program to aid in understanding the flow and relationships between classes and functions. The use of python also increases the ease of maintenance and efficiency. Object oriented programming is easily understood, while maintaining peak performance.

# C6. Self-Adjusting Data Structures

The chaining hash table I chose has pros and cons. The chaining component of the hash table is its greatest strength. It can handle any collision of data with the same index by creating a list. The chaining hash table can hold a lot of data and the searches on moderate amounts of data are relatively quick. This data structure does have weaknesses that would be prominent with large volumes of data. Worst case searching can be long since the keys are not stored in order. Each key must be searched and compared to find the correct one. Overall, the chaining hash table works very well with the nearest neighbor algorithm to be an effective data structure for this program.

# C7. Data Key

The data key used in the program is the package ID. This was the only truly unique qualifier for the packages. Many packages share the same address, city, state, zip code, and deadlines. The notes could be set to make each package different, but for future usability, it would make sense to make that an editable field, so this would become mutable. The key needed to be immutable, and the package ID fits that category. There is also an infinite amount of unique number combinations that would allow for scalability while retaining uniqueness.

# D. Sources

Not Applicable

# E. Professional Communication

Nothing to write. Run your document through <https://www.grammarly.com/>