C950 Task-1 WGUPS Algorithm Overview

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

Sochandaling Teng

ID # 010309126

WGU Email: steng5@wgu.edu

Date: 11/22/2023

C950 Data Structures and Algorithms II

# Introduction

The purpose of this project is to create a routing program for the delivery and distribution of WGUPS. The program is created within given delivery requirements and constraints by determining a self-adjusting data structure and to optimize the routing solution for WGUPS algorithm using Python programming language. It also provides an intuitive interface to users in order to view the delivery status of packages and the distance traveled by all trucks of WGUPS.

# A. Algorithm Identification

The nearest neighbor algorithm is being used to create my program for delivering packages by starting at the hub and selecting the nearest location to deliver based on criteria and the delivery deadline of the packages.

# B. Data Structure Identification

The hash table with chaining is being used in my program to store package data by using package ID as the key.

# B1. Explanation of Data Structure

Hash table is an efficient data structure for inserting, deleting, and retrieving data components. It stores data as key-value pairs. The hash table uses a unique package ID as the key and stores those keys in buckets, with each bucket can store multiple items. With the hash table, all data component of the package is associated with its unique key (package ID). Hence, it uses the key to find an index where it is stored, so its corresponding package details can be accessed and retrieved quickly.

# C1. Algorithm’s Logic

**CREATE hash table to store package data**

**LOAD package data from CSV file into hash table**

**INSERT packages into hash table using packageID as the key**

**LOAD all trucks with packages manually**

**CREATE ToBeDeliveredList**

**FOR package in each truck:**

**SEARCH the package in hash table**

**STORE the package in ToBeDeliveredList**

**ENDFOR**

**CLEAR the package from each truck**

**NEAREST NEIGHBOR ALGORITHM:**

**WHILE ToBeDeliveredList > 0:**

**FOR package in ToBeDeliveredList:**

**FIND nearest address to deliver nextPackage**

**PRIORITIZE package 6 & 25**

**ENDFOR**

**AFTER nextPackage delivered:**

**REMOVE package from ToBeDeliveredList**

**UPDATE mileage traveled**

**UPDATE truck’s current location**

**UPDATE time to deliver packages (Time = Mileage / 18)**

**UPDATE package’s delivery time when truck arrives at address**

**UPDATE package’s departure time when truck leaves the hub**

**TRACK which truck delivering the package**

**ENDWHILE**

**CALL truck 1 and truck 2 to leave the hub**

**CALL truck 3 to leave the hub after one of the trucks delivered all loaded packages**

# C2. Development Environment

The development environment that is used to create my program consists of:

* Hardware: 2019 MacBook Air (1.6 GHz Dual-Core Intel Core i5) with operating system running MacOS Ventura 13.1.
* Software: Python 3.9, PyCharm 2023.2.5 (Community Edition)

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

The space-time complexity of each major segment of my program is:

* The hash table has a complexity of O(1) as hashing functions map a single key to a single value.
* The function to find the distance between two addresses has a complexity of O(n) as it iterates over a distance list linearly one at a time.
* The nearest neighbor algorithm has a complexity of O(n2) as it has a nested loop to iterate over packages to be delivered on each truck in order to loop through a n-size of distance list to find the minimum distance between two addresses.
* The function to print a single package has a complexity of O(1) as it searches the hash table using the key and returns a corresponding value.
* The function to print all packages has a complexity of O(n) as it iterates over the entire package list linearly one at a time to print all packages.

Overall, the space-time complexity of the entire program is O(n2).

# C4. Scalability and Adaptability

The capability of my solution is scalable to adapt to a growing number of packages. By using the nearest neighbor algorithm, the program can accommodate more packages at the same speed by using the package class to ensure that all packages have all the required information and its criteria and using the truck class to add more trucks to deliver more packages. Hence, the algorithm runs independently from the number of packages growing, so any significant change will not reduce the performance of the program.

# C5. Software Efficiency and Maintainability

The software design would be efficient because it utilizes the data structure and algorithm being used in the solution in order to determine the efficient route to deliver all packages within the requirements with the minimum distance traveled. It is easy to maintain as it has a pseudocode to write the code more efficiently with classes and modules, combined with the self-adjusting data structure and algorithm. Therefore, it allows quick access to any data component, easy to change to any part and pinpoints any error within the code. Overall, the software design can minimize resources used and maximize results and performance of the program.

# C6. Self-Adjusting Data Structures

The strengths of the hash table are:

* Fast access to data components
* Efficient modification of data components
* Constant time complexity
* Handle collisions using chaining which each bucket stores multiple items (Lysecky & Vahid, 2020).

The weaknesses of the hash table are:

* Inefficient if there are too many collisions, especially for a large set of keys
* Has limited capacity and can be easily filled up
* Difficult to maintain elements in a specific order.

# C7. Data Key

The key for efficient delivery management is the package ID. Each package ID is unique and constant while other attributes such as address, delivery deadline or delivery status are not unique and can be changed over time. By using package ID as the key, it can map directly to its corresponding value of data components in order to retrieve the correct package information.

# D. Sources

Lysecky, R., & Vahid, F. (2020, August). *C950: Data Structures and Algorithms II*. zyBooks. Retrieved November 24, 2023, from <https://learn.zybooks.com/zybook/WGUC950Template2023/chapter/6/section/2>