C950 Task-1 WGUPS Algorithm Overview

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

Luke Forte

ID #010398679

WGU Email: lfort11@wgu.edu

2/#/25

C950 Data Structures and Algorithms II

Introduction

This task requires creating an algorithm to assist in delivering packages by determining an effective route and delivery distribution. The overarching goal is to ensure that all 40 packages get delivered on time by 2 trucks, keeping the total distance traveled under 140 miles. To accomplish this, I will be using a combination of nearest-neighbor and greedy algorithms.

A. Algorithm Identification

The algorithm used to load the truck will be a greedy algorithm. It will load packages on the truck until no more can fit. Then, it will choose the next package to load and deliver based on its proximity to the hub (for the first delivery) or from the previously delivered package (for all subsequent deliveries).

B. Data Structure Identification

I will be using a priority queue data structure to store the packages that assign a higher priority to the packages that are closest to the delivery address of the last package or the hub for the first package, as well as the highest delivery priority, for truck loading purposes. I will use a standard queue for the trucks as the packages should be sorted by proximity to each other from loading in the hub.

B1. Explanation of Data Structure

The priority queue will consider the highest delivery priority packages as the highest priority thus assigning them to the top of the queue for the next delivery. This works to enable to functionality of the nearest neighbor algorithm enabling efficient delivery of closest proximal packages first.

The regular queue will preserve the decisions from loading and keep the packages proximal to one another.

C1. Algorithm's Logic

Step 1. Initialize the trucks, packages, and drivers

Step 2. Initialize hub package priority sorting packages by highest priority in regard to delivery priority.

Step 3. Load trucks with priority packages, while checking for special cases switching between trucks on each package selected, the first truck will pick a random package, the second takes the high-priority package farthest from the first package to initialize. Then truck 1 picks the next package that is high priority and closest to the last package it selected.

Step 4. Deliver packages.

Step 5. Return to hub.

Repeat steps 3-5 until no packages are left at the hub.

Step 1

Initialize truck

Initialize package

Initialize drivers

Step 2

Sort packages by priority then by distance from the central hub.

```
While packages at hub
    Step 3
        Truck 1 pick a package close to the hub of high priority
        Truck 2 pick a package farthest from the package truck 1 picked
            While truck 1 not full
                Truck 1 pick a package closest to the last picked package of highest priority in hub queue
            While truck 2 not full
                Truck 2 pick a package closest to last picked package of highest priority in hub queue
    Step 4
        While active trucks not empty
            Deliver package
    Step 5
        Return to hub.
```

C2. Development Environment

Operating system: Windows 11 home

Python: v3.12.8

IDE: visual studio code 1.96.4

Processor: Intel(R) Core(TM) i5-9600K CPU @ 3.70GHz   3.70 GHz

C3. Space and Time complexity using Big-O notation

Time complexity for priority queue for both hub

Insertion: O(log n)

Remove: O(1)

Search: O(n)

Access: O(n)

Space complexity for priority queue for both hub

O(n)

Time complexity for queue on truck

Insertion: O(1)

Remove: O(1)

Search: O(n)

Access: O(n)

Space complexity for queue on truck

O(n)

Overall Time Complexity

O(log n)

C4. Scalability and Adaptability

This project is scalable because it does not assume a fixed number of packages, trucks or divers, but lays out an efficient strategy to aggregate priority and proximity to other packages as a means of keeping high efficiency thus allowing it to grow as large as needed until the algorithms themselves would take too long to maintain. Specifically, the sorting algorithm for the hub.

C5. Software Efficiency and Maintainability

This code is maintainable due to the comments to increase readability and static typing to reduce type errors. The software is efficient because it prioritizes packages near previous packages cutting down on unnecessary travel to previously visited locations.

C6. Self-Adjusting Data Structures

The priority queue is beneficial because it has a very fast dequeue operation O(1). Beyond the initial sorting of the queue the program will be able to operate fast and efficiently. The main shortcoming is the initial sort. This will grow proportionally to the amount of packages O(n) still operating fast for a small number of packages but will not keep up with a large quantity. The queue is effective because it does not need to be sorted and boasts fast dequeue times similar to the priority queue O(1). But will not hold any specific sorting after it inherits from the main hub priority queue.

C7. Data Key

delivery address – this key would not suffice because it could be repeated thus making this not unique

delivery deadline – this will not suffice because it will be repeated among many different packages making it hard to distinguish

delivery city - this will not suffice because it will be repeated among many different packages making it hard to distinguish delivery zip code - this will not suffice because it will be repeated among many different packages making it hard to distinguish

package ID – this will be the primary key because it is unique to each package and will not repeat, while not ever being null.

package weight - this will not suffice because it will be repeated among many different packages making it hard to distinguish

delivery status- this will not suffice because it will be repeated among many different packages making it hard to distinguish

delivery time- this will not suffice because it will be repeated among many different packages making it hard to distinguish

D. Sources

Lysecky, R., & Vahid, F. (2018, June). *C950: Data Structures and Algorithms II*. zyBooks.

Rowell, Eric. “Big-O Algorithm Complexity Cheat Sheet (Know Thy Complexities!).” *Bigocheatsheet.com*, 2019, www.bigocheatsheet.com/.

www.naukri.com. “Code 360 by Coding Ninjas.” *Naukri.com*, 2025, www.naukri.com/code360/library/priority-queue-in-cpp. Accessed 4 Feb. 2025.

E. Professional Communication

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