C950 Task-1 WGUPS Algorithm Overview

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

5/21/2024

C950 Data Structures and Algorithms II

Introduction

This performance task requires me to write a WGUPS Routing Program that needs to find “an efficient route and delivery distribution” to ensure that packages are delivered on time. There are three trucks, two drivers, and about 40 packages to be delivered each day, some with delivery deadlines. Some packages have constraints on them such as the time that they will arrive at the hub, specific trucks that they must be on, delivery deadlines, or specific packages that must be with them.

In the following sections, I will detail how I plan to approach the execution of the program.

A. Algorithm Identification

The algorithm I plan on using is the nearest neighbor algorithm. It will provide an approximate solution that should meet the project requirements.

B. Data Structure Identification

I plan on using a hash table to store the package data for this project.

B1. Explanation of Data Structure

The hash table will use the package ID numbers as the key. The values will include all the package information (address, city, state, zip, delivery deadline, weight, delivery time, and special notes). This will allow for a quick lookup of package details.

C1. Algorithm’s Logic

For loading the package, I will put all packages due by a specified delivery time on truck 1. Truck 2 will get all required packages loaded on it. Truck 3 will get the corrected package. The other packages will be loaded randomly along with other specified packages as needed.

After loading packages into trucks as a list:

1. Initialize total mileage equaling 0.
2. Initialize time for truck 1 starting at 6 am.
3. Initialize time for truck 2 starting at 9:05.
4. Initialize time for truck 3 starting at time returned to hub for truck 1.
5. For a package in the truck list, loop through the distance dictionary from the current location to find the nearest next package address.
6. Enter the package into a truck route list.
7. Increment mileage for each truck. Increment delivery time.
8. Add delivery time to package hash table.
9. Repeat steps 5 – 8 until all packages are delivered.
10. Check total mileage is under 140.

C2. Development Environment

I will be using PyCharm 2024.1.1 Community Edition and a MacBook running Python 3.9.6 for creating the application.

C3. Space and Time complexity using Big-O notation

Loading Packages:

The space complexity for creating lists to load the trucks should be O(n) for each list. The time complexity for doing this should be O(n) as well.

Nearest Neighbor Algorithm:

The space complexity of the nearest neighbor algorithm is O(n) since the space only depends on the number of packages. The time complexity of the algorithm is O(n^2) since the program needs to iterate through the list of packages to find the next nearest address.

Hash Table:

The space complexity of the hash table is O(n) since it depends on the number of packages. The time complexity is O(1).

C4. Scalability and Adaptability

If the number of packages increases, the nearest neighbor algorithm will scale. As the number of packages increases to larger numbers, it will become slower to run. The solution is also adaptable to changes in the number and requirements of packages.

C5. Software Efficiency and Maintainability

The software will be written in a way that is efficient and takes into consideration reasonable time and space complexity. I plan on using lists, hash tables, and the nearest neighbor algorithm. With enough comments, it should be easily maintainable.

C6. Self-Adjusting Data Structures

The self-adjusting data structure's strengths include a faster lookup time. They can also be used for efficient storage of large amounts of data. Self-adjusting data structures can also be resized dynamically. Some weaknesses include the fact that they are unordered and could potentially experience collisions.

C7. Data Key

I will choose the package ID as the key since it is a unique number. This will help avoid collisions.

D. Sources

Lysecky, R., & Vahid, F. (2022, August). *C950: Data Structures and Algorithms II*. zyBooks.

Retrieved May 21, 2024, from https://learn.zybooks.com/zybook/WGUC950Template2023

E. Professional Communication

Ran through https://www.grammarly.com/