C950 Task-1 WGUPS Algorithm Overview

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

Jacob Petrakovitz

ID #010616446

jpetrak@wgu.edu

6/1/2024

C950 Data Structures and Algorithms II

# Introduction

# The Western Governors University Parcel Service (WGUPS), needs to develop an efficient routing and delivery system for local deliveries. The goal is to create a system that ensures on-time deliveries adhere to a schedule, and package specifications and maintains a total distance of lower than 140 miles. The intent is to begin using the software locally and begin implementing it in other cities.

# A. Algorithm Identification

Nearest Neighbor Algorithm

This algorithm works as follows (very simplistically): First, it has a list of the initial points of locations, you choose a starting point, measure the distance for each location inside the list of points you initialize, and then compare each location measuring the distances, then it will select the point that closest as your *nearest neighbor*. Once it has its new location, it loops through the distances of each point thereafter and continues the cycle, until all locations have been visited.

# B. Data Structure Identification

Hash Table

# B1. Explanation of Data Structure

Hash tables have key-value pairs that make insertion, deletion, and data retrieval very fast and efficient, O(1) to be specific. The hash table uses a hash function from the key, to find the index where the data is stored inside an array. While each key-value pair is stored in an element also called a “bucket”, each bucket typically can contain multiple pairs, in case of collisions.

# C1. Algorithm’s Logic

Nearest Neighbor Algorithm

Function NNAlgo(truck)

# Initialize list of unvisited packages and current in-process route for truck

Initialize hashPackages = empty list

Initialize current\_route = null

# Call packages to be loaded onto hashPackages list

Call addPackages(hashpackages, truck)

# Initialize a set of unvisited addresses onto truck

# Initialize current\_location to starting point

current\_location = starting\_point

# Use a while loop to loop through each package in hash table, then in conditionals select which truck has space

While unvisited\_address =! Empty or 0

# Initialize the next possible address variable to null and a minimum distance to a large number

next\_possible\_address = null

min\_distance = 9999

# Initialize a for each loop for unvisited addresses to find next nearest address

For each address inside unvisited\_addresses

Distance = find distance between current\_location and package addresses

If distance < min\_distance

Min\_distance = distance

Next\_address = address

End if statement

End for loop

Append next address to current\_route

Remove next address from unvisited\_address

Set current\_location to next\_address

End While loop

Append starting point to next address

End function

# C2. Development Environment

Custom PC: Intel i7 – 10700k || 32GB DDR4 || Win 10 Pro

IDE: IntelliJ Pycharm 2024.1.1 || Python version 3.10.9

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

Beginning with the hash table, it has a runtime complexity of O(1) for individual operations, including insertion, deletion, and search. As, it uses a hash function with a key to direct it specifically to the element inside of the stored array. The space complexity, however, has an O(n) as the values are directly stored inside an array.

For loading the CSV files to the hash table, it will have both a runtime and space complexity of O(n). Where n is the rows, it takes linear time to read each row and store it in memory.

A method for loading the trucks will be needed, that will have a space and time complexity of O(n) where n is the packages needed to be stored in the trucks.

Now the algorithm Nearest Neighbor has a runtime of O(n^2) due to the nested loop and space complexity of O(n^2) as well, this will need to calculate the distances between all pairs of addresses to find the “nearest neighbor”.

Lastly, the print functions. The print function that prints by each package ID will have a time and space complexity of O(1), as it only needs to search the hash table for the individual key-value pair. But a print-all function would have a time and space complexity of O(n) as it would need to go through an entire list of n packages.

Overall, the program will have an O(n^2) space and runtime complexity.

# C4. Scalability and Adaptability

Using the nearest neighbor algorithm with a hash table will demonstrate good scalability and adaptability due to the algorithm being able to assess each key from the hash table at each current location it arrives at. With the hash table, it will be able to scale as it gets to larger cities and it can continue to add key-value pairs to every package as it increases in quantity. Other algorithms may have more optimized runtime and space complexities but the nearest neighbor algorithm is simple and can be adapted much easier than others.

Either way, the current data structure and algorithm will be able to meet and exceed WGUPS expectations and demands.

# C5. Software Efficiency and Maintainability

The nearest neighbor algorithm is not only simplistic but locally optimal. Meaning, it is designed to make a series of choices that locally make the most sense at its current position, it is also typically easy to implement, scale, and maintain. Hash tables are efficient in insertion, deletion, and data retrieval by instant access to the data via the key provided. It runs at constant time O(1) for those operations. Also, because of being constant time for those operations, it handles large data sets very well as it scales. As hash tables grow, programmers may also implement a load factor for automatic resizing of the hash table, also known as rehashing.

With all that being said, hash tables are very versatile while generally being simplistic. With this algorithm and data structure, the software should generally be easy to maintain, scale, and keep efficiency high.

# C6. Self-Adjusting Data Structures

Below I will discuss the pros and cons of using a self-adjusting data structure, in our case, a hash table.

A few of the pros of using a hash table would include, fast access to data whether that is inserting, deleting, or simply retrieving data. Those operations have an average runtime of O(1) constant time. Hash tables are also dynamic, where they can be resized. That makes them memory efficient, as they don’t need to be oversized for scalability. While being dynamic they can also implement a load factor which would resize the hash table automatically once it is needed. Lastly, they are quite simplistic as they are easy to implement and maintain.

On the other hand, these data structures do have downsides to them. They need to have proper collision handling to be able to maintain uniqueness. They can cause a lot of space overhead due to the type of hash functions used. While these data structures are dynamic in nature, the flip side of the coin is also that rehashing/resizing can be costly.

While there are pros and cons for every data structure you may use, you should always weigh the options and choose whichever data structure makes the most sense for your problem.

# C7. Data Key

As the data key, I will be using and generating a “package ID” for each individual package. That will make sure there is uniqueness within each package and there is no ambiguity between IDs. It will also allow constant time O(1) data retrieval for each package allowing for quick update between priorities.

# D. Sources

Mathispower4u (2013, Sept. 12). Graph Theory: Nearest Neighbor Algorithm (NNA)

Retrieved June 1, 2024 from

<https://www.youtube.com/watch?v=zPgsNsOfxQ8>

Garg, Prateek (n.d). Basics of Hash Tables

Retrieved June 1, 2024 from

<https://www.hackerearth.com/practice/data-structures/hash-tables/basics-of-hash-tables/tutorial/>

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

Retrieved June 1, 2024, from <https://learn.zybooks.com/zybook/WGUC950AY20182019/>

# E. Professional Communication

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