WGUPS Routing Program Overview

By: Alexandra Beno

ID: 011095454

Class: C950

A: Algorithm Identification

The Algorithm used in the WGUPS Routing Program created by Alexandra Beno is the nearest neighbor algorithm. It is a self-adjusting, greedy algorithm that is dynamically changing based on its input. In this program, the input is a list of package IDs associated with packages that are being transported on a truck. The truck must deliver these packages to a variety of locations, so the nearest neighbor algorithm is being employed to find an efficient path.

This scenario, where the trucks must deliver a variety of packages to a variety of locations, is an example of the traveling salesman problem. In the problem, the salesman is trying to find a path through a variety of cities to sell his goods. In this situation, the trucks are like the salesmen and the delivery locations are like the cities. The nearest neighbor algorithm is self-adjusting because it dynamically evaluates the packages on the truck while it is at a given location to find the shortest path to another package drop off. This quickly produces a short path through the locations. A variety of algorithms could be employed to solve this problem, but the nearest neighbor algorithm can provide a solution that satisfies the requirements efficiently.

B: Data Structure Identification

The self-adjusting data structure used in the WGUPS Routing Program created by Alexandra Beno is the chaining hash table. It is a data structure that focuses on efficiently storing and retrieving a variety of elements. It uses a hash function to determine where to place an element with a given key on a list of lists. The chaining hash table is self-adjusting because if it finds that it already has an element at a given location on its lists, it can append to the inner list at that spot. This way, the chaining hash table can grow dynamically as it gains elements beyond its original size.

Applied in this example of the traveling salesman problem, the chaining hash table is used to store a variety of data imported from CSV files, such as package or distance data.

Package data is particularly important in this project, as there are 40 packages with many unique attributes that must be efficiently stored and retrieved. By using the package ID as a key in the hash function, the chaining hash table is able to store all 40 packages in its list of lists and retrieve them for a variety of operations.

B1: Explanation of Data Structure

In this program, the chaining hash table starts with a list of 10 lists. For the package hash table, it needs to store 40 packages. This calls for the inner lists of the hash table to be shared amongst the packages. When a collision occurs, and two values need to be stored in the same slot, a normal hash table would just overwrite the information in that slot. By using key-value pairs the chaining hash table will instead append the new data to the end of the inner list. This way no information is lost, and it can all be efficiently stored and accessed.

For this situation, the chaining hash table is used to store packages with the package ID as the key, and the package as the value. The data of the 40 packages can be spread out amongst the 10 inner lists for later access. When a package ID is used to search the hash table, it will iterate through the inner list associated with that ID's hash, and return the package data if it is found. This way the data for a single package is all stored as one object and can be easily accessed with the package ID as a key.

C1: Algorithm's Logic

The following is an explanation of the nearest neighbor algorithm's logic in this scenario using pseudo-code:

```
1. given a Truck (T) with a List (L) of associated packages and a starting location
2. closest distance -> 9999
3. closest package -> nothing
4. for each Package (P) in L:
       • distance = the distance between the location of the truck and the package
         delivery location
       o if distance < closest distance:
             ▪ closest distance -> distance
             ▪ closest package -> P
5. move T to the package delivery location of closest package
```

6. deliver the package and remove it from L
7. if there are still packages on the truck:
    - go to step 2
8. Return the truck home

By the end of this algorithm, the Truck will have moved through each of the delivery locations and dropped off all of its packages before finally returning to the WGU HUB.

C2: Development Environment

The program was developed using Python version 3.13. It was written in PyCharm version 2025.1.2. No external Python libraries are required, as the program uses only what comes with Python by default. Though it may function in other Python environments, it has not been tested in them. It was developed on a local machine, a Dell XPS 8700 running Microsoft Windows 10 Pro version 10.0.19045 build 19045. The computer is using an Intel Core i7-4790 CPU and an AMD Radeon RX 480 graphics card.

C3: Space-Time Complexity Using Big-O Notation

There are two major segments of this program that need to be measured for their time-space complexity. The first is the nearest neighbor algorithm. As it is a greedy algorithm, it is interested in finding a solution in a reasonable amount of time. As such, it does not spend forever trying to find the perfectly optimal route, just one that gets you there through a relatively short path. The nearest neighbor algorithm has a Time Complexity of O(N^2), as it has to iterate through the list each time it runs. It has a Space Complexity of O(N) as it only has to store each data point once.

The other major segment of this program is the chaining hash table. Due to its chaining properties, it is more interested in accurately preserving data than maintaining the same complexity in every situation. In cases where the inner list that was hashed to is empty, it performs well as all it has to do is add the new element to the list. However, in cases where many elements hash to the same inner list, the chaining hash table will slow down considerably. In the best cases, the chaining hash table has a Time Complexity of O(1). In the worst cases, it can be O(N). It has a Space Complexity of O(N) as it only has to store each data point once.

Of these two major segments, the nearest neighbor algorithm is much more complex, and as a result the program inherits its O(N^2) Time Complexity. The program also inherits its Space Complexity of O(N) from the pieces that make it up.

C4: Scalability and Adaptability

With its O(N^2) complexity, the program can quickly scale beyond memory and time limits if the inputs increase enough. It will take longer and longer for the program to store and retrieve packages from the package hash table, and though this can be temporarily alleviated with a greater number of internal lists, doing so will also increase the load on the memory. The nearest neighbor algorithm will be especially costly in terms of time, as it will have to iterate through the entire list of packages associated with the truck at each stop. The more packages the trucks are transporting, the longer the program will have to take in order to find an efficient route. If attempting to expand the WGUPS system, it would be wiser to instead change the algorithm used to store packages or implement multiple iterations of the system.

C5: Software Efficiency and Maintainability

The program is efficient as it runs in O(N^2) complexity thanks to the nearest neighbor algorithm. It can quickly provide a path between a variety of package delivery locations, allowing for the packages to be delivered by the WGUPS system on time and in accordance with special notes.

The program is maintainable as it contains a variety of comments that help document what the code does and how it functions. Each method or block of code has a comment explaining what is happening, and many important loops and if statements have comments throughout that help to elaborate. Different objects are stored in different class files, allowing future developers to make changes to individual parts of the program without altering the rest.

C6: Self-Adjusting Data Structures

The chaining hash table, as a self-adjusting data structure, has a few notable strengths and weaknesses. One strength is that it does not have a maximum size limit, and therefore can store a theoretically infinite amount of data so long as memory is infinite. Another strength is that multiple values whose keys hash to the same slot will not overwrite each other like in a normal hash table, instead both will become part of an inner list so all data is retained.

One weakness of the chaining hash table is that its memory cost can quickly increase as the number of items added to the table increases. No maximum on the amount of items stored can mean that the chaining hash table could quickly balloon in size as its inner lists grow in length. Another weakness is that, in the worst case, if many elements hash to the same location, the complexity of the data structure can grow to O(N). This could happen regardless of the number of items or their size in memory if their keys happen to line up correctly with the hashing function.

C7: Data Key

The packages come with a variety of attributes, including delivery address, delivery deadline, delivery city, delivery zip code, package ID, package weight, and delivery status. The key chosen for the chaining hash table used for packages in this program was the package ID. This is because the package ID is a unique identifier for each package that is not repeated on any others. Things like the deadline, address, city, zip code, weight, and status could all end up mapping to multiple different packages, while each package ID maps only to one package. Using the package ID as the key allows you to retrieve package data with one line of code, whereas using any of the other fields could result in more coding required and even data potentially being overwritten.

D: Sources

- The Packages and Addresses CSV files were provided for the project by Robert Ferdinand, program mentor at WGU via WGU connect on March 29th, 2025. Accessed 6/18/25. Citing source: https://wguconnect.wgu.edu/hub/wgu-connect/groups/c950-data-structures-and-algorithms-ii/discussion/question/168133/CSV-Data-Loading-Address-Information

- The Distance CSV file is a cleaned version of the WGUPS Distance Table.xlsx file provided for the project via the task page by Western Governor's University. Date of publication unknown. Accessed 6/16/25. Citing source: https://tasks.wgu.edu/student/011095454/course/30860017/task/4042/overview#:~:text=downtown%20map.docx-,WGUPS%20Distance%20Table,-.xlsx