**Nearest Neighbor Algorithm Overview**

*B1. Explain the algorithm using pseudocode.*

**Nearest Neighbor Algorithm**

**delivery(truck)**

**create an empty list called undeliveredIds**

**while undeliveredIds still has packages**

**set initial value of nextAddressDistance to 9999**

**set nextPackage to None**

**for each ID in undeliveredIds**

**if the distance between the trucks address and the package’s address is <= nextAddressDistance,**

**set nextAddressDistance to that distance**

**set nextPackage to the package.**

**Add nextPackage ID to the truck’s package ID list**

**Remove NextPackage from undeliveredIds**

**Add nextAddressDistance to the truck’s miles property**

**Set the truck’s address to nextPackage’s address**

**Add the duration of nextAddressDistance / 18 to the truck’s time property**

**Set the next package’s packageDeliveryTime to the truck’s time**

**Set the next package’s packageDepartureTime to the truck’s departed time.**

B2. *Describe the programming environment used to create the Python application.*

This project was created using PyCharm 2022.1.2 (Professional Edition) on macOS Ventura. The screenshots were taken using native macOS tools and the program was run in the terminal using an application called Hyper (version 3.4.1).

*B3. Evaluate the space-time complexity of each major segment of the program, and the entire program, using big-O notation.*

The time complexity of this algorithm is O(N2). This is because for each package in the undelivered list, the algorithm loops through the entire undelivered list again to find the package with the shortest distance to the truck’s current location.

Other than the main algorithm listed on the first page, here are the other main methods of the program.

**Methods:**

ReadCSVs.getPackages() :: goes through all packages in csv file and creates a package object.

ReadCSVs.getDistances ( row, col ) :: returns value found in row, col place in distances csv.

ReadCSVs.getAddressIndex(address) :: finds address in csv file and returns the ID of the location

**Running Times:**

ReadCSVs.getPackages() : O(N)

ReadCSVs.getDistances(row, col) : O (1)

ReadCSVs.getAddressIndex(address) : O(n)

Adding all of the methods, we would get a time complexity of O(N2 + N + N + N + N + 1) when we consider reading the CSV files as well as all other methods. But we generally would refer to this program as having a time complexity of O(N2).

*B4. Explain the capability of your solution to scale and adapt to a growing number of packages.*

My program can easily scale to accept more packages. The packages object is created directly from the data provided. Therefore, this program is already ready to accept new packages. If we are to schedule new trucks, there will need to be additional work done. The data required for each package may also need to be updated. As of now, the trucks are created manually based on the notes that give us information on which packages need to be delivered together.

*B5. Discuss why the software is efficient and easy to maintain.*

This program is efficient and easy to maintain because it was built with object-oriented principals. Because the two major aspects of this program have their own models, it is easy to follow the code, and to expand the capabilities in needed. If in future versions more data is needed for each truck, we can easily add more properties.

The use of hash tables leads to a very efficient piece of software. Hash tables allow you to access data in constant time (O(n)) on average. The worst-case scenario would be O(n).

*B6. Discuss the strengths and weaknesses of the self-adjusting data structures.*

This program uses a hash table for the packages. The main strength of using a hash table is its fast access time. Accessing a specific package will happen in constant time on average. At worst, the lookup time will be O(n). Hash tables are more space and time efficient than binary trees and other data structures.

The disadvantages of using hash tables are potential collisions. A collision occurs when the hash function generates the same index for more than one key. For large datasets, this can become unavoidable. Another downside is that it is more complex to implement than other structures, which take very few lines of code to create. Hash tables also do not maintain any order of elements, which would make retrieving elements in a specific order difficult without additional helpers.

*D1. Explain how your data structure accounts for the relationship between the data points you are storing.*

In the MakeHashTable class, we initialize an empty array. When we insert a package into the hash table, we do so by passing the init method a key, and then the package itself. This allows up to easily search and remove packages by passing in the key to the remove and search methods. We use the package ID as the key.

*I. Describe at least two strengths of the algorithm used in the solution.*

A. A benefit of the nearest neighbor algorithm is that is simple to implement and understand. We simply check each package destination against every other package destination and find the closest one. Then we move to the next package. Another benefit is that it can perform quickly with small datasets like we have in the solution.

C. Two other algorithm we could use instead are the greedy algorithm and Dijkstra’s algorithm. The greedy algorithm uses a different method for determining the route. The greedy algorithm traverses a graph and makes decisions based on what is the favorable outcome at that time. It is very fast, but it does not always provide optimal results.

Dijkstra’s algorithm works by assuming that each address will keep track of its distance from another address and that we keep a record of each address that we have and have not visited. As we continue to visit each address, our records are updated. We create edges and vertices to keep track of the distance between addresses. It is a faster algorithm than nearest neighbor, but storage is less efficient.

*J. Describe what you would do differently, other than the two algorithms identified in I3, if you did this project again.*

If I were to do this project again, I think I would look for a way to make the project more automated. As it is now, the trucks are loaded manually with package IDs. If I did it again, I would look for a way to automatically create each truck without needing to manually specify package IDs.

*K1.*

*1a. Explain how the time needed to complete the look-up function is affected by changes in the number of packages to be delivered.*

Because we are using hash tables as our data structure, the look-up function will be unaffected by the number of packages. The look-up function of a hash table’s time complexity is O(1).

*1b. Explain how the data structure space usage is affected by changes in the number of packages to be delivered.*

The space complexity of a hash table is O(n). This means that the space will increase evenly as the number of items in the hash table increase.

*K2. Identify two other data structures that could meet the same requirements in the scenario.*

Two other data structures that would work in this scenario would be graphs and arrays.

*K2a. Describe how each data structure is different from the data structure used in the solution.*

An array is different because it will use less space but will take more time for searching. An array stores each item right after one another, while a hash table stores items in many different places without relation to any other item. A hash table offers much faster look up at O(1) while an array’s look-up is O(n)

A graph differs from a hash table in many ways. In a graph there will be many nodes and edges that connect the nodes. They are used to model complex networks like transportation networks and social networks. A hash table is used to map keys to items and does not store any extra type of data. Graphs allow for traversal and pathfinding, while hash tables offer quick lookup based on keys.