Data Structures and Algorithms Two

Christian Oxner 009554482

A)
Algorithm Used: Nearest Neighbor

The first part of the algorithm removes all packages from the truck and stores them in a temporary list. Originally, I tried to implement the algorithm without doing this but kept getting an index out of bounds error.

The program uses the nearest neighbor algorithm, and it is implemented using a two-pointer technique that I learned while doing a data structures and algorithms course on LeetCode. The two pointers start at the first package on the truck and last package on the truck. The code will run until the last point index is less than the first pointer index. I create a temporary variable for the first and last package on the truck. From these packages we calculate the distance from the current address to the address of the package. If the distance of the first package is less than we set temporary variable min package equal to the first package. We set the distance of the min package also equal to a temporary variable and then decrease the pointer of at the index of the last package by 1. If the index of the min package is found with the last package index the same process occurs with assignment of min package and min package distance, but instead of decreasing the index of the last package we increase the index of the first package.

Once the min package is found a few other things occur. The trucks address is set to the address of the min package. The distance to deliver the package is added to the trucks total distance traveled. The algorithm then removes the package from the list it was stored in. It adds time to the truck based on the distance traveled to deliver the package divided by the average speed given in assignment instruction. Lastly it sets the delivery time of the package to the truck time.

Pseudo Code:

```
Function fn(truck):
     Declare empty array to store packages from truck
     For package on truck add it to array storing package from truck
     Add the trucks departure time to the package
     While there are packages in array from truck
     left = 0
     right = truck.length  -  1
     while left <= right
          first package = packages on truck index[left]
          last package = packages on truck index[right]
```

```
    for address in the address csv file
        if truck current address = address
            truck index = address index
        elif first package address = address
            first package index = address index
        elif last package address = address
            last package index = index
    start distance = distance from truck current address to package found with left index
    if start distance is blank
        start distance = distance from left index to truck current address
    convert start distance to a float
    last distance = distance from truck current address to package found with right index
    if last distance is blank
        last distance = distance from tight index to truck current address
    convert last distance to a float
    if start distance < last distance
        min package = first package
        min package distance = start distance
        right–
    else
        min package = last package
        min package distance = last distance
        left++
truck current address = min package address
truck distance traveled += min package distance
remove package from array storing truck packages
truck time += package distance / truck avg speed
min package delivery time = truck time
```

B)

The program I used to develop this program was PyCharm 2022.3.1 (Professional Edition). It was my first time using PyCharm, but it was very similar to IntelliJ which I used for software 1 and 2. I enjoyed the environment it helped me correct small mistakes I would make when typing and solving errors in the code was simple. Python version 3.9.12 was used to create and run the project. Project developed on a Mac OS version Ventura 13.1

C) The space-time complexity for reading in the packages into a hash map is O(n) it will run for as many packages are in the package's csv file. Moving down to when trucks are instantiated we load the packages manually into each truck giving us a space-time complexity of O(1). Moving to the algorithm the initial step of loading the packages from the truck into a separate array has a space-time complexity of O(n). The next part of the algorithm is done with two while loops and a for loop to deliver the packages giving a space-time complexity of O(n^3). In the main part of the program the space-time complexity is 0(n) if the user enters T as each package is looped through to check its delivery status. If the user enters a P we search the package hash map until we find the package we are looping for. The worst-case space- time for this section will be O(n).

D)
The algorithm is not the best in terms of growing and scaling. Each new package that is added to the csv must be manually loaded to the truck. The manual addition of packages those results in a constant run time for loading packages instead of another algorithm needing to be involved. Every time a new package is added the O(n^2) time complexity will increase more than if we were able to have an algorithm that ran with a 0(n log n) or below complexity.

E)
The software is easy to maintain and efficient because of its simplicity. Manual loading of trucks is a simple process but can also result in packages being missed if not careful. The way main is laid out the workflow of the program and what is happening is easy to follow. The code within the algorithm is easy to understand. The main method to is very straightforward on what user input is accepted and what is not. Failure to enter accepted input results in the closure of the program.

F)
One weakness of the Hash Table is that the class functions run on a O(n) run time. If we were to search for an element of the Hash table if it were the last element, we would have to check every element to find the one we wanted. The advantage to hash tables for our program is that packages all having a unique id there is no need to handle collisions. All packages will be given a unique value in the hash table and be easily identifiable.

D1)
The data structure hash map uses accounts for the relationship between the data by creating a key value for each package being stored within it. The packages records are stored in buckets formed by using the hash key. The hash key is calculated by calling the built in python hash function. A bucket is equal to a hash function called on the key values, and the remainder returned when divided by the length of the table. So the data structure accounts for relationships by storing the data in a bucket. That bucket is chosen by calling the hash key function on the data key which in this case is the ID of the package. (Garg, P 2023)

Sources

Garg, P. (n.d.). *Basics of hash tables tutorials & notes: Data structures*. HackerEarth. Retrieved February 16, 2023, from https://www.hackerearth.com/practice/data-structures/hash-tables/basics-of-hash-tables/tutorial/