Samantha Millhart

Student ID 004640948

Data Structures and Algorithms II – C950

1. I chose to use the Greedy Algorithm for my program. The program looks only for the next closest address in the list at the moment.
2. 1. Pseudo Code explanation:

Create timeArrays for each truck

Set start time for each truck

Convert time from string to timedelta

Create check Time function

New time = truck miles / 18 mph

Convert new time in to timedelta

Import Distance File

Set column title to row number

Create distance look up function

Find distance between starting point and location array using distance dictionary repeats

Import Package File

Set column title to column number

Create package dictionary based on package ID

Set package ID with matching address, city, state, zip, deadline, mass, notes, and delivery status

Create package array

Create array for packages with special notes

Create array to calculate the number of packages with special notes

Create distance array

Create distance dictionary

Set starting point as the hub

totalPackages = truckOne + truckTwo + truckThree

While truckOne < 16

Run look up distance from starting point for next location

truckOne = truckOne + 1

Set starting point = next location

totalPackages = truckOne + truckTwo + truckThree

While truckTwo < 16

Run look up distance from starting point for next location

truckTwo = truckTwo+ 1

Set starting point = next location

totalPackages = truckOne + truckTwo + truckThree

While truckThree < 16 or totalPackages < 41

Run look up distance from starting point for next location

truckThree = truckThree+ 1

Set starting point = next location

Print(‘welcome’ + total miles traveled)

User input ‘1’ for all packages

User input ‘2’ to search

Get user input for time

If user input is 1

Check user time

If start time >= user time then status is ‘at hub’

If start time < user time < time delivered then status is ‘on route’

If start time < user time > time delivered then status is ‘delivered’

Print all packages

If user input is 2

Get user package id input

Check user time

If start time >= user time then status is ‘at hub’

If start time < user time < time delivered then status is ‘on route’

If start time < user time > time delivered then status is ‘delivered’

Print package where package ID == user package ID input

2. PyCharm Python 3.9 pip 21.1.2

3. Big O

Big O citation in line in code.

Total = O(N^2)

4. With any changes to the number of packages the program could easily be changed to simply add 7 more packages to truck three. If there are additional packages from that it is simply a matter of running another While Loop to reload truck one, once it has returned to the hub, and running the distance look up for the last point to bring the truck two driver back to take it. This would make big O = O(N^2)

5. This software is efficient because it can quickly find best options for delivery and runs smoothly. It’s easy to maintain, because it is a simple couple of changes to increase or decrease package numbers, truck capacities, distances in miles, or number of trucks.

6. Hash tables are efficient at looking up data along with storing and deleting it. One downside of hash tables is that it cannot have null values.

C – H) in program

D) 1. The hash table is used to store the package information. When a packages is assigned to a truck number or the delivery status has changed this will update. It also allows for packages to be looked up by their package ID.

1. 1. The greedy algorithm is a good option for this case, because it efficiently gets the packages out and quickly with a simple algorithm. This also allows easy changes to add more packages.

3. One other algorithm would be nearest neighbor algorithm. For this we would have looked at the map overall and planned a route looking at all the options instead of just the next one. Another algorithm would be dijkstra algorithm which would find the shortest paths among the nodes.

J) If doing this project again I would start with focusing more on the packages than the locations distances. I spent a lot of time wrapping my head around the distances and how to make it work and I think had I started with the trucks and packages it would have come easier to me.

K) 1. a) Time would lower if less packages were being delivered and higher if more packages were being delivered, because the function would need to iterate either less or more times.

1. b) With an increase to the number of packages to be delivered there is an increase to the size of the hash table causing increased to the space usage.

1. c) Since the cities are and trucks are lists the space will grow linearly but the lookup time will stay the same.

2. One other data structure that could have been used is a graph. Using a graph would have allowed similar packages to be grouped together and then I could have traversed the graph until the length of 16 was reached. A second option would be a binary search tree. The benefit here would be presorting packages based on attributes which would all them to be quickly accessed through a tree.