Core Algorithm

1. Identify the algorithm that will be used to create a program to deliver the packages and meets all  requirements specified in the scenario.

The algorithm I use is mainly a greedy algorithm since I choose the option that is optimal at specific points of time. This is evident since each truck has a list of packages and they drive to the package that has the least distance to their current location. I say that this is the algorithm mostly used since I manually sort some packages to make sure it fits the special delivery requirements. However, after I manually sort it, however, the truck still automatically sorts it and goes to the closest package to it.

# This node class helps with sorting the distances of the packages in an array.

# An adjacency matrix could have also been used instead of the hash map of packages

# A weighted graph could also have been made based on the distances between each packages but this would take a lot of calculations since there are 40 packages

# If there are no delivery requirements, a more efficient algorithm would be to recalculate distances every time a package is picked up

B.  **Write a core algorithm overview, using the sample given, in which you do the following:**

1.  **Comment using pseudocode to show the logic of the algorithm applied to this software solution.**

I created my own data structure to store the packages as, which I named Node, in the file. This node file includes the package id, package location, and package distance to whatever it is set. I use the hashmap to pull the information of each packages. Before the trucks start delivering, an array of these package objects are created with the distances set from the hub. But after every delivery, the distances of each node is recalculated to make sure the the truck always goes to the closest package.

O(1)

arrayOfPackages = []

O(n) to create and fill this hash

hashMapOfAllPackages = {…}

O(1) to create a single node object

PackageObjects = Node(“8”, “300 State St”, distanceFromHub)

O(n) as it iterates through the number of packages, O(1) for appending it to an array

for eachNumber in range(n)

arrayOfPackages.append(str(hashMapOfAllPackages.get(eachNumber)))

O(n^2) since it is nested. O(n) for iterating through the array of packages and O(n) for finding the distance on the csv which is done by iterating through the columns and rows.

for eachPackage in arrayOfPackages

eachPackage.distance = distance(fromHub, toLocation)

O(n^2logn) since it iterating through the array of packages is O(n) and sorting the entire array is O(nlogn). Since the two is nested, the complexity comes to O(n^2logn)

Before every delivery:

Sort(arrayOfPackages.distances)

Go to first on the list of arrayOfPackages

2.  Apply programming models to the scenario.

In this, the csv file is loaded into the program and loaded into a hash table with the information from the csv file.

4.  Discuss the ability of your solution to adapt to a changing market and to scalability.

As noted, the program’s whole complexity is O(n^2logn) since iterates through n items and then sorts them after. No matter the increase in the amount of packages, this complexity will always stay the same since n is based on packages that is on the csv file. The complexity is also not terrible as some algorithms have cubic, exponential, and factorial time complexities which will deeply hurt the scability of the software, however, the one provided here is in quadratic-log time.

5.  Discuss the efficiency and maintainability of the software.

The program’s efficiency is O(n^2logn) based on the fact that there are n items to be added to an array, and the array is then sorted with nlogn efficiency. The program is easy to maintain since if you simply add information to the csv file, it will be automatically be added to the packages list. The only thing that is difficult is the special delivery requirements which might need some manual sort but most of the work is done with the provided algorithm used.

6.  Discuss the self-adjusting data structures chosen and their strengths and weaknesses based on the scenario.

The total efficiency of the program is O(n^2log(n)) since the most that I use is a loop of the packages remaining and sorting every time that I drop of the package to see which package is the next closest. This is based on the fact that the sort algorithm in python is O(nlog(n)) and going through the array of objects is O(n) and they are nested so it is O(n^2log(n)).

D.  Identify a data structure that can be used with your chosen algorithm to store the package data.

1.  Explain how your data structure includes the relationship between the data points you are storing.

I had a hash table to hold all the package information. This includes package id, address, zip, city, state, delivery status, delivery deadline time delivered, and time picked up. A hash table is really important since it allows you to store a lot of information in each key/value pair and lets you look it up with an average search complexity of O(1). This is important since I use the hash table to fill up the package object with the necessary information such as location before creating an array of it which is then sorted for my algorithm.

Apply programming models to the scenario.