1. ALGORITHM IDENTIFICATION

The algorithm that I created was the nearest neighbor algorithm. With this algorithm I created the “inOrder()” method, where I started at the hub and looked for the closest address (address A), then looked for an address that was closest to address A., and so on.

1. DATA STRUCTURE IDENTIFICATION  
   I developed a HashTable class which it is like the Dictionary data structure. My HashTable stores a collection of key-value pairs (package ID with Package). This allows for quick access to the packages.  
   B1. The relationship between the key and the value is that the key is the ID of the package, and the value is the package. When you create an instance of my HashTable class, it will automatically create an empty list of lists (this will act as a bucket). There are 10 lists. Any package with a package ID that has the same unit number will be added to the same list. For example: 5, 15, 25 and 35 will be in list index 5 (bucket 5)
2. For this project I decided to put all the packages that are required to be in the same truck in truck one, the packages that are delayed, in truck two, and the package that we have the incorrect address in truck three. I assigned each truck with a number (1,2,3). Then with the method “assingPacks()” we will assigned the remaining packages to truck 1 or truck 2 based on closest distance. With this method we will see which package from the remaining packages has the closest distance with the packages from truck 1 or truck 2. If the package is closest to one of the packages from truck 1, we will add the package to truck 1. Otherwise, package gets added to truck 2. This is repeated until both truck 1 and truck 2 have 16 packages. The packages that were not assigned are added to truck 3.   
   After all packages are assigned to a truck. I review truck 3 to see if there are any packages that need to be delivered at a specific time. If so, we will add those packages to either truck 1 or truck 2 (based on closest distance) and the additional package(s) from truck one or two that does not have a specific delivery time, gets added to truck three.   
   Now that all the packages are assigned to the correct truck, we will put the packages from each truck in order with the “inOrder()” method. With this method we will start with finding the package closest to the hub which will be the first package that will be delivered. Second package will be the closest distance from the first package, and so on. This will let us know in what order the packages will be delivered.  
   Once the packages are in order for each truck, we will find the packages that have a specific delivery time (for truck 1 and truck 2)  
   Truck one will start at 8:00am. Truck 2 since it has delayed packages, will start at 9:05am. And truck 3, because it has the wrong address package, will start at 10:20am.   
   The “getRoute” method verifies that the packages that were placed in order will be delivered on time. This method obtains the distance from point A to B and how long it takes to get to point B. Starting with the first package on the truck list, we want to find out the distance between the hub and the first package, and how long it will take to get to the first package. With that delivery time, we verify if we will have sufficient time for the packages that need to be dropped at a specific time. if we have sufficient time, then the package can stay in the order it was placed. Otherwise, we will do the calculation with the first deadline package and will be placed as the first package to be delivered. Each package will have their delivery time within the method’s calculations.  
   The packages are assigned with a truck number and start time when the truck started the deliveries. Truck 3 will start when either truck 1 or truck 2 are done delivering the packages.

1 –   
For this Nearest Neighbor algorithm. We want to find the next closest package   
since we start at the hub, we would want to find the closest package to the Hub  
we will provide the list of packages from a truck.  
variables:  
minDist – we want to find the minimum distance from the HUB  
packList – list of packages from a truck   
we will do a for loop with all the packages in the list to see which one is closest to the HUB  
once we have the closest package to the HUB, that package will be the first package that will be delivered.

**for i in range(0, len(packList)):**

**dist = distance2Hub(packList[i])**

**if dist < minDist:**

**minDist = dist**

**pack = packList[i]**

Once we get the package nearest to the Hub we will find a package nearest to that package

orderedPackages = [pack] (this will store all the packages in the order they will be delivered)

availablePackages -> a list that stores all 16 packages minus the package found above.

**for k in range(0, 15):**

**minDist = number larger than all distances**

**for i in range(0, len(availablePackages)):**

**pack = orderedPackages[-1]** (this is last package that was added to orderedPackages)

**dist = distanceBtwnPackagess(pack, availablePackages[i])**

**if dist < minDist**

**minDist = dist**

**closestPack = availablePackages[i]**

**orderedPackages.append(closestPack)**

**availablePackages.remove(closestPack)**

orderedPackages now has all 16 packages in order.

2 –

For my project I used jupyter lab. I was able to create notebooks at first and then transfer all of my coding into a python file. Jupyter lab is a web based interactive development.

Operating system – Windows 11 version 22H2

Jupyter Lab - Version 4.0.5

Python - Version 3.11.4

3 -

**Line# Time Complexity Space Complexity**  
HashTable class O(1) O(n)

availablePackages() O(n^2) O(n^2)

availableCombinedPackages() O(n) O(n)

getCombinedPackages() O(n) O(n)

distanceBtwnPackagess() O(n) O(n)

whichTruck() O(n^2) O(n^2)

assignPackages() O(n) O(n)

getRestrictedPackages() O(n) O(n)

closestPackage() O(n^2) O(n^2)

addRestrictedPackagesInTruck() O(n) O(n)

furthestPackFromHubNoRestrictions() O(n^2) O(n^2)

switchPackages() O(n) O(n)

distance2Hub() O(n) O(n)

closestPackage2Hub() O(n) O(n)

getClosestPackageWithinList() O(n) O(n)

inOrder() O(n^2) O(n^2)

getDeadlineList() O(n^2) O(n^2)

miles2Min() 1 1

isEnoughTime() O(n) O(n)

getRoute() O(n) O(n)

getRoute2() O(n) O(n)

setTruck() O(n) O(n)

setStartTime() O(n) O(n)

getTotalDistance() O(n) O(n)

**ENTIRE PROGRAM** O(n^2) O(n^2)

4 -

If the number of packages increases for each truck, my algorithm is adaptable. I created a Truck class where it takes the max number of packages. It can be changed from 16 to any number below or above. My code/methods will adapt to the new max number.

5 -

My software is efficient and easy to maintain because I created classes for different objects, and I commented on all the methods to help understand what they do.

6 -

My HashTable advantage compared to a regular list of packages is that the time complexity would be better, the searching is faster, and it takes up less space. A disadvantage would be that if there is a larger quantity of packages, hash table becomes inefficient and collisions happen

7 -

In this case, the key was the package ID. There is reason why I made that the key it is because the package ID is a unique number to the package which made it efficient to get the specific package we are looking for.