WGUPS Package Delivery Program Documentation

A. loadTruck is a greedy algorithm used to load the packages onto the trucks.

B1. loadTruck is my self-adjusting algorithm.

loadTruck(truckList, truckNum)

index = 0 # used to ensure no more than 16 packages is loaded at a time

planeArrival = 9:05 AM

if packageTable is no empty:

if truckNum == 2:

for list in packageTable:

for i in list:

updateStatus(packageTable, truckNum, ‘Out for Delivery’)

truckList.insert(i[0], i[1] )# package number and contents

packageTable.remove(i[0]) # remove from packageTable

index = index + 1 # increment index

for list in packageTable:

for i in list:

if i[0] = 13, 14, 15, 16, 19, or 20 # packages must ship together:

updateStatus(packageTable, truckNum, ‘Out for Delivery’)

truckList.insert(i[0], i[1] )# package number and contents

packageTable.remove(i[0]) # remove from packageTable

index = index + 1 # increment index

for list in packageTable:

for i in list:

if deadline != ‘EOD’ or I[0] != 6 or 25:

updateStatus(packageTable, truckNum, ‘Out for Delivery’)

truckList.insert(i[0], i[1] )# package number and contents

packageTable.remove(i[0]) # remove from packageTable

index = index + 1 # increment index

if index == 16:

return # Exit method

for list in packageTable:

keyList = [] # Create empty list to keep indexes the same so nothing is skipped

for i in list:

if package notes contains ‘Delayed on flight’:

if truckNum == 1 and truck1Time >= planeArrival:

updateStatus(packageTable, truckNum, ‘Out for Delivery’)

truckList.insert(i[0], i[1] )# package number and contents

packageTable.remove(i[0]) # remove from packageTable

index = index + 1 # increment index

else if truckNum == 2 and truck2Time >= planeArrival:

updateStatus(packageTable, truckNum, ‘Out for Delivery’)

truckList.insert(i[0], i[1] )# package number and contents

packageTable.remove(i[0]) # remove from packageTable

index = index + 1 # increment index

if index == 16:

break # Exit inner loop

for i in keyList: # Remove packages from packageTable

packageTable.remove(i)

if index == 16:

return # Exit method

for list in packageTable:

keyList = [] # Create empty list to keep indexes the same so nothing is skipped

for i in list:

if package notes does not contain ‘Delayed on flight’:

updateStatus(packageTable, truckNum, ‘Out for Delivery’)

truckList.insert(i[0], i[1] )# package number and contents

packageTable.remove(i[0]) # remove from packageTable

index = index + 1 # increment index

if index == 16:

break # Exit inner loop

for i in keyList:

packageTable.remove(i) # Remove packages from packageTable

if index == 16:

return # Exit the method

B2. Software: PyCharm 2021.2.2 (Community Edition) using Python version 3.7.2

Hardware: Processor - Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz

RAM - 8.00 GB

Hard Drive – 225 GB SSD

B3. loadTruck – O(N^2)

deliverPackages – O(N^2)

userInterface – O(N^2)

loadPackageData – O(N)

loadDistanceData – O(N)

distanceBetween – O(N)

minDistanceFrom – O(N^2)

The entire program – O(N^2)

B4. Adapting to a growing number of packages would not be an issue. Adding some more commands for loading trucks and changing small amounts of code in loadTruck, such as having a truck return to the HUB for more packages, would be easy to change.

B5. The software is efficient and easy to maintain since if the number of packages were to increase or decrease, minimal amounts of coding would need to be added, if any at all. Loading packages onto the trucks is limited to 16 packages maximum to a truck at a time, regardless of how many total packages there are to deliver. Due to this fact, loading packages is efficient since the workload would never increase at any one time. Also, if the number of addresses were to increase or decrease, there are no steps that would need to be taken in order to compensate for the change.

B6. A strength of using hash tables is that you can easily reference a specific item without having to search through an entire list. Another strength is there are no indexes. You can add or remove items from the hash table, but you can still very easily, and quickly, reference the specific item that you need. A weakness of the hash table is code to add or remove objects from the hash table is more complex than simply appending an item to the end of the list.

C. WGUPS Distance Table is imported into the list distanceData. WGUPS Package File is imported into the hash table packageTable. All delivery requirements and constraints have been met.

C1. A comment has been put in the first line of main.py that includes my name and my student ID.

C2. Comments in the code have been added to explain what each method does and how it works.

D. Hash tables are used in association with the loadTruck method.

D1. The data structure of the hash tables uses the package IDs for keys, and the rest of the package information as the value. loadTruck goes through the package data and add packages to the trucks by using the keys. Data is stored in the hash table using the package ID. The package ID is modulo of 10 to determine where in the hash table the data is to be store. The package ID serves as the key, and the package information serves as the value.

E. Hash table packageTable is created and takes the insertion of the package ID, deadline, address, city, states, zip code, weight, and delivery status from the ‘WGUPS Package File’ CSV file. No additional libraries or classes are used for insertion.

F. The user interface takes user input to search for package ID, deadline, address, city, zip code, weight, and delivery status.

G. An interface has been provided for the user.

G1. Please reference ‘Package Status Screenshot – 925 AM’ in the program’s root folder.

G2. Please reference ‘Package Status Screenshot – 1006 AM’ in the program’s root folder.

G3. Please reference ‘Package Status Screenshot – 1230PM’ in the program’s root folder.

H. Please reference ‘Program Status with Mileage’ in the program’s root folder.

I1. The strengths of the loadTruck method are that it compensates for requirements on how to load the packages, such as some packages can only ship on truck 2 or some packages are delayed on a flight, and the method prioritizes priority packages ahead of end of day shipments.

I2. The algorithm meets all of the stated requirements.

I3. Dijkstra’s algorithm, Nearest Neighbor algorithm

I3a. Dijkstra’s algorithm creates a graph to reference the shortest path when delivering packages, whereas the algorithm used does not create a graph and instead searches through the packages to find the next closest address. Nearest Neighbor algorithm uses cartesian coordinates to find the nearest neighbor. This helps to make the algorithm fast and efficient. The algorithm used loops through the packages and finds the closest one to deliver next, whereas Nearest Neighbor will use the predetermined coordinates to determine which is closest.

J. I would have like to have come in with a more streamlined version of loadTruck as well as deliverPackages I feel that they could be improved upon with loading according to restrictions, and delivering priority packages in a better manner.

K1. The hash table meets all of the requirements.

K1a. Looking up packages would take longer if the number of packages increased. Using the hash table helps to quickly narrow down where a certain package is located in the hash table, but after that, a list will need to be searched through to find the correct package.

K1b. Similarly to the previous answer, space usage would increase should the number of packages be increased. While the hash table has only 10 slots, within each slot is a list that could contain many packages. Naturally, as the number of packages increases and the lists become longer, more space will be required to store the data.

K1c. Adding trucks would help out the lookup times and space usage. The more trucks that are loaded and delivered, the more quickly packages will be searched through since the amount would be decreasing. Adding cities to the addresses would not affect the lookup times or space usage.

K2. Dictionaries and nested lists

K2a. Dictionaries differ from hash tables as every key needs to be unique making a lot of keys, whereas in hash tables, multiple values can reference the same key, and then contained within a list inside of that key. Nested lists differ from hash tables in that there is no key value relationship. In order to find a package, the list would need to be searched through in order to find the correct information.

L. References have been listed for ChaingingHashTable class and Package class.

M. Professional communication has been used throughout the program.