Stated Problem:

WGUPS must get 40 packages delivered across Salt Lake City. The company has 2 drivers and 3 trucks, however there are complications to be considered. Each truck can only have a total of 16 packages, but weight does not come into consideration. Some packages won’t get to the hub until 9:05 AM as they were delayed. Some of the packages have delivery deadlines attached and others can only be on a specific truck, or they must be delivered with other packages. The student Is to figure out a way of optimizing the delivery system using the 2 drivers with 3 trucks and reduce miles as much as possible.

Algorithm Overview:

A: The greedy algorithm

B: (1)

1: Get parameters of the trucks list, truck number and current location. Starting at the hub

2: Current location is the compared to the other locations of the other packages on the truck to determine the lowest value.

3: The truck then moves to the next location and once there removes the value from the list.

4: Value is attached to an optimal list, then recalls the algorithm for the next package in the list.

Space-time complexity of the chosen algorithm is worst case scenario is O(n^2), best case is O(1)

Pseudocode:

1)

Greedy Algorithm:

1: Receive the following parameters

Truck\_distance\_list – a list of packages on the truck

Truck\_number – variable that determines which truck is currently delivering

Current\_location – where the truck is currently, starts at the hub

2: Create a way to break out of the algorithm

If(truck\_distance\_list) is empty

Return

3: Enter the first recursive loop to find lowest value if truck list is not empty:

Set lowest value to 20

If length of truck\_distance\_list is not 0

For index of truck\_distance\_list

If get\_distance of current\_location and current\_location + 1

New\_distance <= lowest\_value

Lowest\_value = new distance

4: Enter second recursive loop to use lowest value to get optimized travel

For index of truck\_distance\_list – trucks working list

If get\_distance of current location and the next in the index is the lowest value

get\_truck(truck\_number).append– add package to optimized list

get\_truck\_distance(truck\_number).append - add distance truck travelled to the distance list

remove package from working list and move truck to next location

recall algorithm until no more packages

B: 2)

Software:

IDE : Pycharm 2022.3.1

V: Python v 3.11.1

Hardware:

Intel® Core™ i9-10900KF CPU @ 3.7GHZ

64GB RAM

Windows 10 Home

64-bit operating system, x64-based processor

3) Space-time Complexity: O(N^2) – Worst Case, complexity listed throughout code

LaddGi

4) The program that I wrote does not have restrictions on truck capacity, weight limit, time or distance. It could potentially work for any number of packages within reason. Very few things would have to be changed in the program for it to account for more packages. Even more destination or address could easily be implemented, but eventually the algorithm will not be efficient enough for the loaded files. The more files or packages that are loaded into the program will increase the length of the algorithm. The algorithm uses a double for loop, so it is N \* N runtime complexity, the more N you add to either loop will increase the runtime by that many more packages. Currently the program runs a search through 26 address and 40 packages, worst case for the algorithm is O(N^2) which is dependent on the size of N.

5) The program is efficient because it helps determine the shortest route for a delivery of a list of packages. It is easy to maintain because the parameters are easy to adjust for different instances and variations to the read files. More trucks or packages could be added with ease as well as more delivery start times. Such values could even be read in as input from the user. Potentially even the read file that populates the lists could be user based.

6) A strength of my chosen algorithm is that it is easily adaptable to a growing set of data. It efficiently will calculate the shortest distance for any group of packages that are fed to it. With minor changes the algorithm could read in another set of location names and distances to add those to the already existing locations, and potentially cross reference the 2 data files. A weakness in using a hash table is that there will eventually be collisions and the probability of collisions increases as the data pool increases. Eventually the hash table will be filled up and it will become more inefficient as more collisions occur. Hash tables do not allow null values, this was at first a problem for me as I was not assigning a value to my arrival time.

C: Program attached

D: Hash Table: By using a hash table I was able to easily compare the address of the packages that I stored in the table with the list of address where deliveries could be made. By looking up and comparing these data points, I was able to assign a location number to each package which in turn was used as a column/row in the distance table. Package ID was used in the lookup function for users. Using a hash table made both functions very simple. By extension with not much effort search parameters could be added for any of the package information stored. Such as address, mass or exodus times.

I:

1: First strength of my chosen algorithm is that it is adaptable to increasingly larger data sets. The size of the data input into the algorithm does not change the big O notation, O(N^2) is as complex as the program gets. This algorithm can preform better than other algorithms for computing the shortest distance, but it may not be the best.

2: Screenshots show that the algorithm works

3:

Nearest neighbor would be a good choice for the algorithm to solve this problem as it uses prediction for grouping of data points. This method is another type of greedy algorithm as it grabs or moves to the closest node, where the greedy algorithm I used looks for the shortest distance total. Nearest neighbor will move to the closest neighbor and may not always be the shortest route.

Dijsktra is another algorithm that could be used to calculate the shortest distance, it plots the distances between points by using a graph. Uses graph theory and is a type of greedy algorithm that uses weighted graphs with positive weights.

J: What I would do differently if I had more time or had to take a new approach at this would be to possibly try a different or multiple algorithms and see which one would work best. I would have also preferred to get the range(len(packages)) to work properly, but I constantly got errors I got tired of tracking down. In the end I ended up using the basic (1, 41) length set for timestamp. This value can be easily changed to accommodate more packages.

K:

1: Using a hash table meets the requirements, it allows for retrieval of a package data in constant time by using the key or package id in this program. The use of a hash table allowed me to allow for the user to do a search for a specific package based on id. Hash tables are worst case O(N) and best case is O(1). By adding more packages, it would increase the look up time in a linear sense, but not run time. If you doubled the number of packages, it would take twice as long to get to the end of the list. The length of the search is based on the number N. Hash tables are good for these situations because they are expected to remain constant regardless of the amount of data store within.

2: One possible data structure that could also be used in this scenario are linked lists. You could store each package into a list and link it to the next, this would also work with the greedy algorithm, and when it sorts by distance it would create a new list starting with the first, shortest distance in the register and link the other packages by the next closest destination. This is very similar to a hash table where the information is store in buckets rather than linked lists. Each linked list points to the next one in the chain, whereas there is an index of the has table.

A second type of data structure is an array which are like a hash table except for lookup time. In an array data structure, you have to loop over all the items before you find what you are looking for whereas in a hash table you can compare against the index. In a hash table you store each value or object using a key and in an array you do not.

L: No sources to cite