Data Structures and Algorithms II

A.) A greedy algorithm is used to optimize each truck’s list of packages for deliver. This algorithm is located in OptimizeDeliver.py line 81.

B1.)

The Greedy algorithm takes a truck\_Package\_List as a parameter.

set current\_LocationID = 0 (Location of the Hub)

set truck\_list\_length = number of packages in the trucks list

set optimize\_Package\_List = empty list

for length in range (0, truck\_list\_length)

set shortest\_Distance = 20.00 (value will change upon first iteration)

set shortest\_DistanceID = 0

set iteration = 0

for item in truck\_Package\_List:

item\_Address\_ID = Gets package’s Address ID from the item list

distance = get distance from current location to first item in truck\_Package\_List

if (distance of current item is less than current shortest\_Distance)

shortest\_Distance = distance

shortest\_DistanceID = item in list

shortest\_Distance\_Index = loop iteration

iteration = iteration + 1

set current\_Location = to shortest Distance location

Remove shortest distance item from the truck\_Package\_List

Add shortest distance item to Optimize Package List

B2.) PyCharm 2020.3.2 used to Develop Program

B3.)

|  |  |  |  |
| --- | --- | --- | --- |
| DeliveryTrucks.py |  |  |  |
| Method | Line | Space | Time |
| None | 48 | O(N) | O(N) |
| Overall |  | O(N) | O(N) |

|  |  |  |  |
| --- | --- | --- | --- |
| HashTable.py |  |  |  |
| Method | Line | Space | Time |
| \_\_init\_\_ | 5 | O(1) | O(1) |
| add | 9 | O(1) | O(1) |
| get | 13 | O(1) | O(1) |
| update | 18 | O(1) | O(1) |
| Overall |  | O(1) | O(1) |

|  |  |  |  |
| --- | --- | --- | --- |
| ImportCVS.py |  |  |  |
| Method | Line | Space | Time |
| None | 12 | O(N) | O(N) |
| None | 51 | O(N) | O(N) |
| None | 59 | O(N^2) | O(N^2) |
| Overall |  | O(N^2) | O(N^2) |

|  |  |  |  |
| --- | --- | --- | --- |
| OptimizeDeliver.py |  |  |  |
| Method | Line | Space | Time |
| optimze\_List | 87 | O(N^2) | O(N^2) |
| optimze\_List | 93 | O(N) | O(N) |
| deliver\_Packages | 127 | O(N) | O(N) |
| Overall |  | O(N^2) | O(N^2) |

|  |  |  |  |
| --- | --- | --- | --- |
| SearchPackages.py |  |  |  |
| Method | Line | Space | Time |
| search\_List | 11 | O(N) | O(N) |
| search\_List | 16 | O(N) | O(N) |
| search\_Options | 44 | O(1) | O(1) |
| Overall |  | O(N) | O(N) |

Overall Complexity of program:

O(N) + O(1) + O(N^2) + O(N^2) + O(N) = O(N^2)

B4.) The core algorithm is designed to scale and adapt to the number or packages and trucks. Adding additional items to a trucks list will not significantly increase the route optimization. After each item is added to the optimized list, it is removed from the items needed to be checked and thus reduces the number of iterations by 1. Adding addition trucks will not affect the core algorithm as it optimized 1 truck at a time. Adding addition locations could affect the program’s run time. In ImportCVS.py in 59, the program assigns each item with a locations ID. The run time is O(N^2). This step would grow exponentially with the addition of new locations.

B5.) The software is efficient because its time complexity is O(N^2). The core algorithm optimizes packages in an efficient manner and the delivery complexity of each list of packages is O(N).

This software was developed to be easy to maintain. The code is appropriately commented. Each method and class have meaningful names that explain their function. The code is broken down so that it is easy to follow and understand.

B6.) The data structure used in this program are lists stored in a hash table. This data structure is efficient at data retrieval with its constant time complexity of O(1). I was able to pull items from the Hash table and access different elements of the list very quickly. This allowed for very quick searching when looking for a package by its ID number.

A drawback I found in using this data structure was when searching for packages, I found myself iterating over the entire list and checking each item to see if it was what the user was looking for. This creating a time complexity of O(N). With a list of 40 items, that is not a big deal. However, if there we 10,000 packages, it would take much longer to look at every item every time when searching.

D1.) The data structure used in this project is a Hash Table. The hash table is used by the core algorithm to look up items and add them to the optimized list. A Hash table is an important data structure for this algorithm because it allows for a lookup time of O(1) when finding and adding an item to the Optimized list.

I1.) One strength of this algorithm is its only parameter is a list of packages. This is an advantage because it will optimize any number of packages. If the company decides to put 20 packages on a truck, no adjustments will need to be made to the algorithm to account for that.

Another strength of this algorithm is it sorts faster after each item. After the closest package to the trucks current location is found, it is removed from the list of packages needing optimizing. This allows for the algorithm to do one less iteration after each package is added to the optimized list.

I2.) The algorithm used meet all requirements in this scenario. The algorithm uses a greedy approach to optimize a truck’s packages for delivery. The final distance is under 140 miles. No more than 16 packages are loaded onto a single truck. Each package is delivered, marked with a timestamp upon delivery and marked delivered in the Hash table.

I3.) Heuristic algorithm: With this approach, the algorithm would determine delivery locations closest to the hub and find which packages needed to be loaded onto the truck until a maximum of 16 packages were on a truck. The algorithm would then look for the next closest locations not yet visited and load packages that need to be delivered to those locations. The process would repeat until every location was visited.

Dynamic Algorithm: With this approach, the algorithm would break the package list into smaller groups. It would then organize those based on distance from the hub. These smaller groups could then be combined to larger groups and organized until the full package list is combine organized on distance from the hub. This organized list could then be split between the trucks and the packages could be delivered.

J.) If I were to do this project again, I would take out the need of assigning each address and address ID. While working on the project, it was easier for me to understand the delivery of item if each address had and ID. I built a two-dimensional array with all the distances from one location to another and looked at each delivery item as a points on a graph. For example if the current location was address ID 7 and the next location was address ID 4 then the point (7,4) would hold the distance between these 2 points. While this worked, it created several extra loops that could be avoided if I were to use a different method to get distance between points and save a lot of run time.

K1.)

1. Looking up a package by its ID is not affect by number of packages as its time is O(1). Looking up packages by any other criteria is increased because the lookup function iterates over every item O(N) to see if it meets the user’s criteria.
2. Each package added to the overall packages list adds another bucket to the Hash Table. The table’s space is O(N).
3. Changing the number of trucks would not affect the lookup time of packages. Adding more cities would also not affect the lookup time or space of the hash table.

K2.) Another data structure I could have used would have been a binary search tree. This would allow me to presort the packages based on a specific attribute. Searching the binary tree would be significantly faster.

I could have also used a graph. The advantage to using a graph is the ability to group similar packages together in adjacent vertices. The graph could then be traversed until the is reach the length of 16. It could load those packages onto a truck.

L.) Resource used for project.

Learn.zybooks.com. (n.d.). *zyBooks*. [online] Available at: https://learn.zybooks.com/zybook/WGUC950AY20182019