1. Identify a named self-adjusting algorithm (e.g., “Nearest Neighbor algorithm,” “Greedy algorithm”) that you used to create your program to deliver the packages.

I used a nearest neighbor algorithm, it first finds the starting point from the set of packages on the truck, and then from there moves to the nearest point after.

B. Write an overview of your program, in which you do the following:

1. Explain the algorithm’s logic using pseudocode.

While delivered[15] == 0

Lowest = 50000

For i of trucks

Check each point if its at a lower distance then the variable lowest

After finding the lowest distance, go to that location and mark the package delivered in the list

2.Describe the programming environment you used to create the Python application.

PyCharm 2020.3.5 (Professional Edition)

Build #PY-203.7717.81, built on March 25, 2021

Licensed to PyCharm Evaluator

Expiration date: April 25, 2021

Runtime version: 11.0.10+8-b1145.96 amd64

VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.

Windows 10 10.0

GC: ParNew, ConcurrentMarkSweep

Memory: 1981M

Cores: 8

3.Evaluate the space-time complexity of each major segment of the program, and the entire program, using big-O notation.

Space-time complexities are found in inline pydocs.

4.Explain the capability of your solution to scale and adapt to a growing number of packages.

My solution can scale and adapt to a growing number of packages if another algorithm is added that pre sorts the packages instead of manually sorting them.

5.Discuss why the software is efficient and easy to maintain.

The software is easy efficient and easy to maintain because it does the job properly, and is commented properly.

6. Discuss the strengths and weaknesses of the self-adjusting data structures (e.g., the hash table).

The hashing method has some collisions from the hash key method used, however for a majority of the look up, it results in a O(1) time complexity.

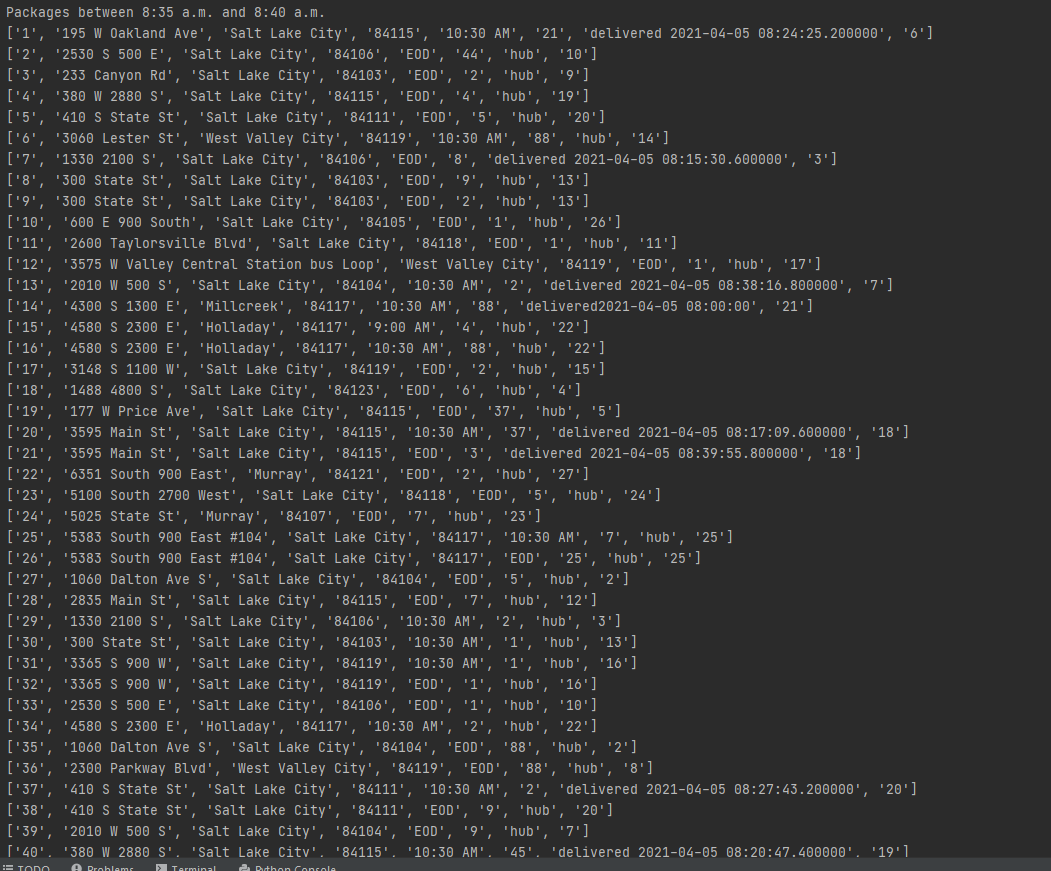
D. Identify a self-adjusting data structure, such as a hash table, that can be used with the algorithm identified in part A to store the package data.

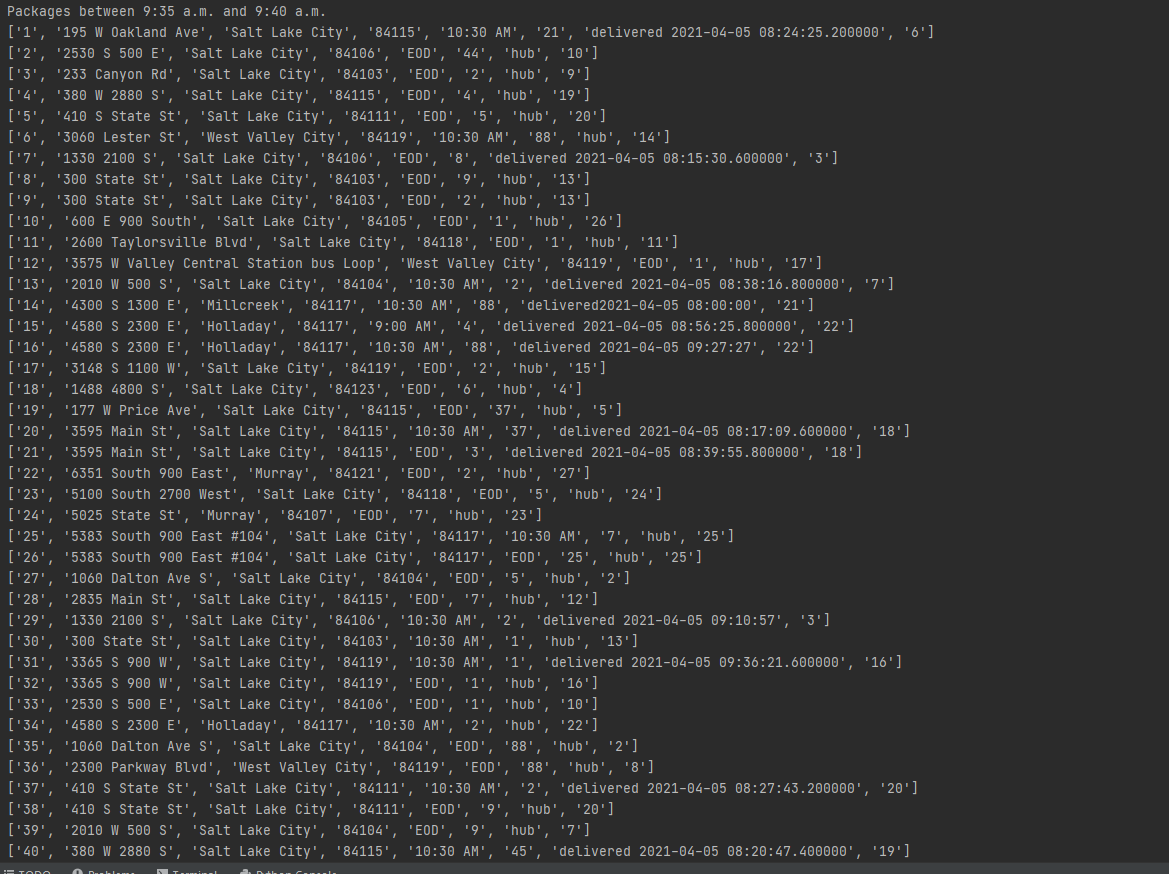
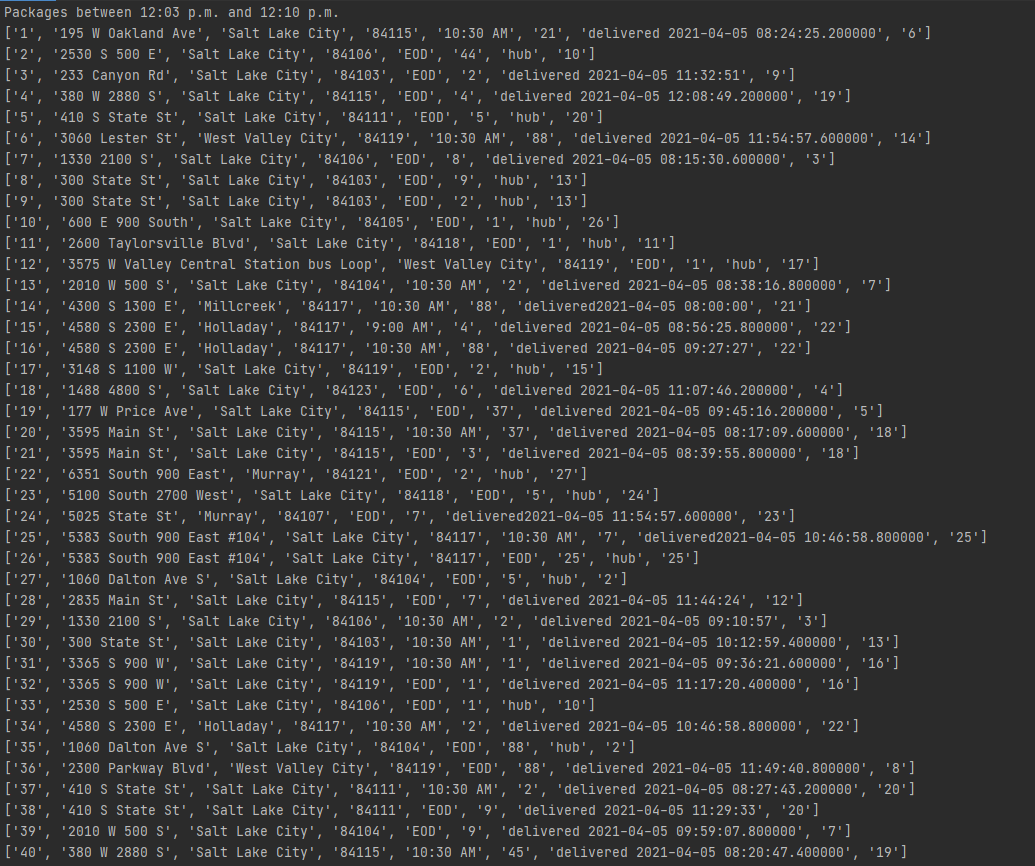
A hash table is used.

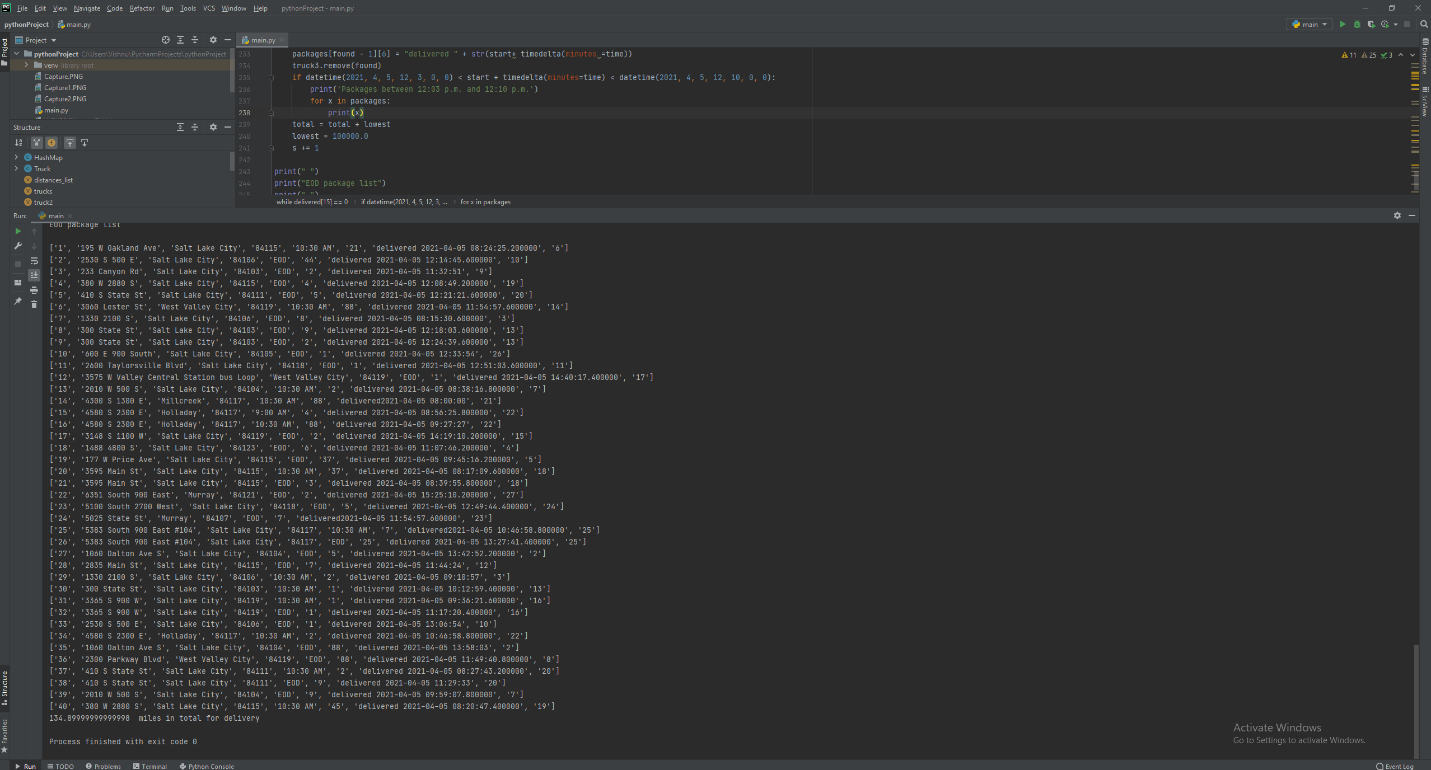
1. Explain how your data structure accounts for the relationship between the data points you are storing.

The hash table stores the package data, and helps match the destination to the destination cvs file.

1. Provide screenshots to show the status of all packages at a time between 8:35 a.m. and 9:25 a.m.



1. Provide screenshots to show the status of all packages at a time between 9:35 a.m. and 10:25 a.m. 
2. Provide screenshots to show the status of all packages at a time between 12:03 p.m. and 1:12 p.m.

H. Provide a screenshot or screenshots showing successful completion of the code, free from runtime errors or warnings, that includes the total mileage traveled by all trucks

I. Justify the core algorithm you identified in part A and used in the solution by doing the following:

1. Describe at least two strengths of the algorithm used in the solution.

The algorithm can scale and adapt if an algorithm is added to the packages added to the truck.

The algorithm finds the solution required with the requirements given

1. Verify that the algorithm used in the solution meets all requirements in the scenario.

The algorithm does

1. Identify two other named algorithms, different from the algorithm implemented in the solution, that would meet the requirements in the scenario.

Dijkstra algorithm and farthest insertion

A . Describe how each algorithm identified in part I3 is different from the algorithm used in the solution.

Farthest algorithm starts off by finding the farthest distance node available It then repeatedly finds the city not already in the tour that is furthest from any city in the tour, and places it between whichever two cities would cause the resulting tour to be the shortest possible.

Dijkstra algorithm assigns initial values to the nodes and from there finds the most optimal route.

J. Describe what you would do differently, other than the two algorithms identified in I3, if you did this project again.

I would not manually load the trucks, and would instead creating a sorting algorithm for them.

K. Justify the data structure you identified in part D by doing the following:

1. Verify that the data structure used in the solution meets all requirements in the scenario.

The hash table has an insert and lookup feature that works.

* 1. Explain how the time needed to complete the look-up function is affected by changes in the number of packages to be delivered.

The hash map itself is 0(1) which is affected the number of packages delivered, so a change in packages is an increase in look up time, which is still considered constant.

* 1. Explain how the data structure space usage is affected by changes in the number of packages to be delivered.

More collisions will occur.

* 1. Describe how changes to the number of trucks or the number of cities would affect the look-up time and the space usage of the data structure.

The number of trucks shouldn’t affect look-up time as those are not looked up in the program, simply used as a vehicle for the next trip.

An increase in cities should still keep the look up time of the data structure O(1)

2. Identify two other data structures that could meet the same requirements in the scenario.

A priority queue, and a linked list.

1. Describe how each data structure identified in part K2 is different from the data structure used in the solution.

A priority queue would help sort the packages according to the priority given to them, for instance their distance to the previous package placed in the queue.

A linked list is a linear data storage structure with a look up of O(N) as opposed to O(1) of a hash map