Trayer Lee Harvey
6/30/2022
001341276

Data Structures and Algorithms II
NHP2 TASK 1: WGUPS ROUTING PROGRAM
Write Up

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

The program involves a sequence of loading packages onto trucks, making adjacency lists of their destinations, then using that list to find an optimal route for the truck via a routing function. This routing function is powered by a **Nearest Neighbor Algorithm;** selecting which remaining location is closest to our current location, before navigating to said location and running the algorithm again, forming a list of neighbors which can be used as a list of stops in a more optimal order.

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

1. Explain the algorithm’s logic using pseudocode.

```
Sort(list of addresses, list of adjacencies from CSV)
    # Find closest to start (hub) first
    min distance = 100 (any large number)
    for each address in list of addressess
        distance = distance between address and ‘Hub’
        if distance < min distance
            # update
            min key = address
            min distance = distance
    # add to sorted list as first destination, remove from old
    sorted addressess.append(min key)
    list of addressess.remove(min key)

    # then for each new destination added to the sorted list
    for each address in sorted addresses
        # find nearest neighbor, add them to list too (if new)
        min distance = 100
        for each address in list of addresses
            if distance between address and current location < min distance
                # check to make sure it’s a new entry and not already on the list
                for other entry in sorted list
                    if address = entry
                        is not new
                    else
                        min distance = distance
        # add to sorted list as first destination, remove from old
        sorted addressess.append(min key)
        list of addressess.remove(min key)
```

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

IDE: PyCharm 2022.1.2
Python: 3.9
Hardware: Windows 10 Desktop Computer

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

See comments in program. Entire Program complexity is discussed in main.py at the top in the comments

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

My program grabs values from the CSV files it is given. The underlying algorithm for pathing is not reliant on a certain length of list (number of packages), so it can scale as demand does. At the moment, the loading phase is done manually, as this makes sense for the relatively small number of packages going through the area. But as demand scales up, the loading will become the bottleneck until it's deemed worth automating

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

The software is efficient because it focuses on the important part of the delivery, the next nearest neighbor, and doesn't waste resources processing for anything but. It's easy to maintain because of it's modular code structure (functions on their own pages) and easy to read comments

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

The strength of a hash map comes from it's quick lookup ability (a time complexity of O(1)) with even a worst case scenario being a O(n) search through all items. Space is used well in a hash map due to the resize feature being called whenever there are too many elements. However no hash is perfect, and the relatively simple hash feature used in this program can lead to collisions, causing the size to be larger than necessary in some cases

C. Write an original program to deliver all the packages, meeting all requirements, using the attached supporting documents "Salt Lake City Downtown Map," "WGUPS Distance Table," and the "WGUPS Package File."

Attached

1. Create an identifying comment within the first line of a file named "main.py" that includes your first name, last name, and student ID.

See main.py

2. Include comments in your code to explain the process and the flow of the program.

See included files

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.

Hash Map / Hash Table

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

Hash Tables are used in a few places to keep track of relationships between data. Trucks contain lists of hashes, unreadable without the list of packages on that truck, of all the packages and package info on that truck. These hashes are stored with the key being equal to that package id, and the value being a list of all the attributes of that package. That way only someone with the approved data can access our other data points

E. Develop a hash table, without using any additional libraries or classes, that has an insertion function that takes the following components as input and inserts the components into the hash table:

- package ID number
- delivery address
- delivery deadline
- delivery city
- delivery zip code
- package weight
- delivery status (e.g., delivered, en route)

See load.py where the package info is formed and hashmap.py for the add (insertion) feature

F. Develop a look-up function that takes the following components as input and returns the corresponding data elements:

- package ID number
- delivery address
- delivery deadline
- delivery city
- delivery zip code
- package weight
- delivery status (i.e., “at the hub,” “en route,” or “delivered”), including the delivery time

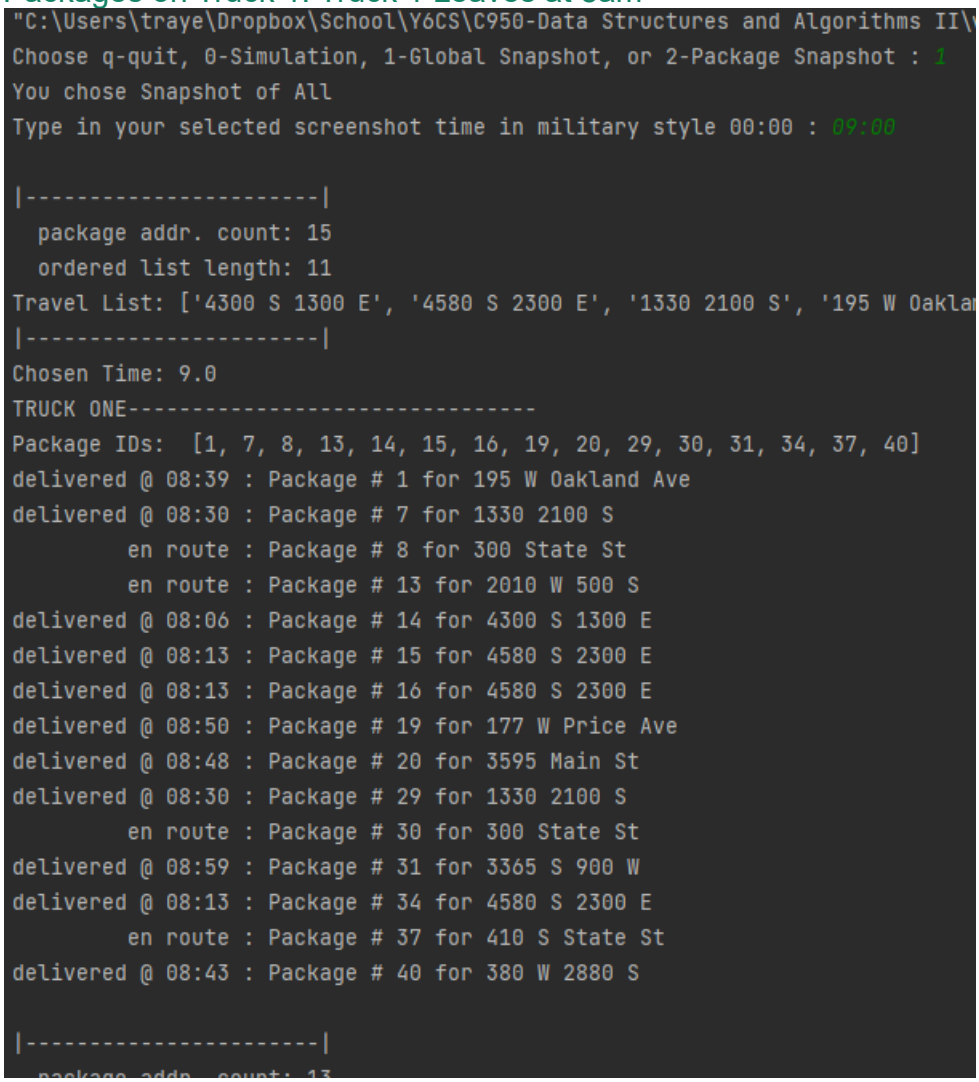
See hashmap.py “get”, also see load.py where the package info is formed and hashmap.py for the add (insertion) feature

G. Provide an interface for the user to view the status and info (as listed in part F) of any package at any time, and the total mileage traveled by all trucks. (The delivery status should report the package as at the hub, en route, or delivered. Delivery status must include the time.)

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

The program prints the status of all packages by truck, too big to fit in one screenshot. Can be verified in program if necessary

Time: 09:00
Packages on Truck 1: Truck 1 Leaves at 8am



```
"C:\Users\traye\Dropbox\School\Y6CS\C950-Data Structures and Algorithms II\
Choose q-quit, 0-Simulation, 1-Global Snapshot, or 2-Package Snapshot : 1
You chose Snapshot of All
Type in your selected screenshot time in military style 00:00 : 09:00

|----------------------|
  package addr. count: 15
  ordered list length: 11
Travel List: ['4300 S 1300 E', '4580 S 2300 E', '1330 2100 S', '195 W Oakla
|----------------------|
Chosen Time: 9.0
TRUCK ONE-------------------------------
Package IDs:  [1, 7, 8, 13, 14, 15, 16, 19, 20, 29, 30, 31, 34, 37, 40]
delivered @ 08:39 : Package # 1 for 195 W Oakland Ave
delivered @ 08:30 : Package # 7 for 1330 2100 S
         en route : Package # 8 for 300 State St
         en route : Package # 13 for 2010 W 500 S
delivered @ 08:06 : Package # 14 for 4300 S 1300 E
delivered @ 08:13 : Package # 15 for 4580 S 2300 E
delivered @ 08:13 : Package # 16 for 4580 S 2300 E
delivered @ 08:50 : Package # 19 for 177 W Price Ave
delivered @ 08:48 : Package # 20 for 3595 Main St
delivered @ 08:30 : Package # 29 for 1330 2100 S
         en route : Package # 30 for 300 State St
delivered @ 08:59 : Package # 31 for 3365 S 900 W
delivered @ 08:13 : Package # 34 for 4580 S 2300 E
         en route : Package # 37 for 410 S State St
delivered @ 08:43 : Package # 40 for 380 W 2880 S

|----------------------|
```

Packages on Truck 2: Leaves at 9:06

```
|----------------------|
  package addr. count: 13
  ordered list length: 11
Travel List: ['5383 South 900 East #104', '2530 S 500 E', '2835 Main St', '33
|----------------------|
Chosen Time: 9.0
TRUCK TWO-------------------------------
            at hub : Package # 2 for 2530 S 500 E
            at hub : Package # 3 for 233 Canyon Rd
            at hub : Package # 6 for 3060 Lester St
            at hub : Package # 12 for 3575 W Valley Central Station bus Loop
            at hub : Package # 18 for 1488 4800 S
            at hub : Package # 25 for 5383 South 900 East #104
            at hub : Package # 27 for 1060 Dalton Ave S
            at hub : Package # 28 for 2835 Main St
            at hub : Package # 32 for 3365 S 900 W
            at hub : Package # 33 for 2530 S 500 E
            at hub : Package # 35 for 1060 Dalton Ave S
            at hub : Package # 36 for 2300 Parkway Blvd
            at hub : Package # 38 for 410 S State St

|----------------------|
```

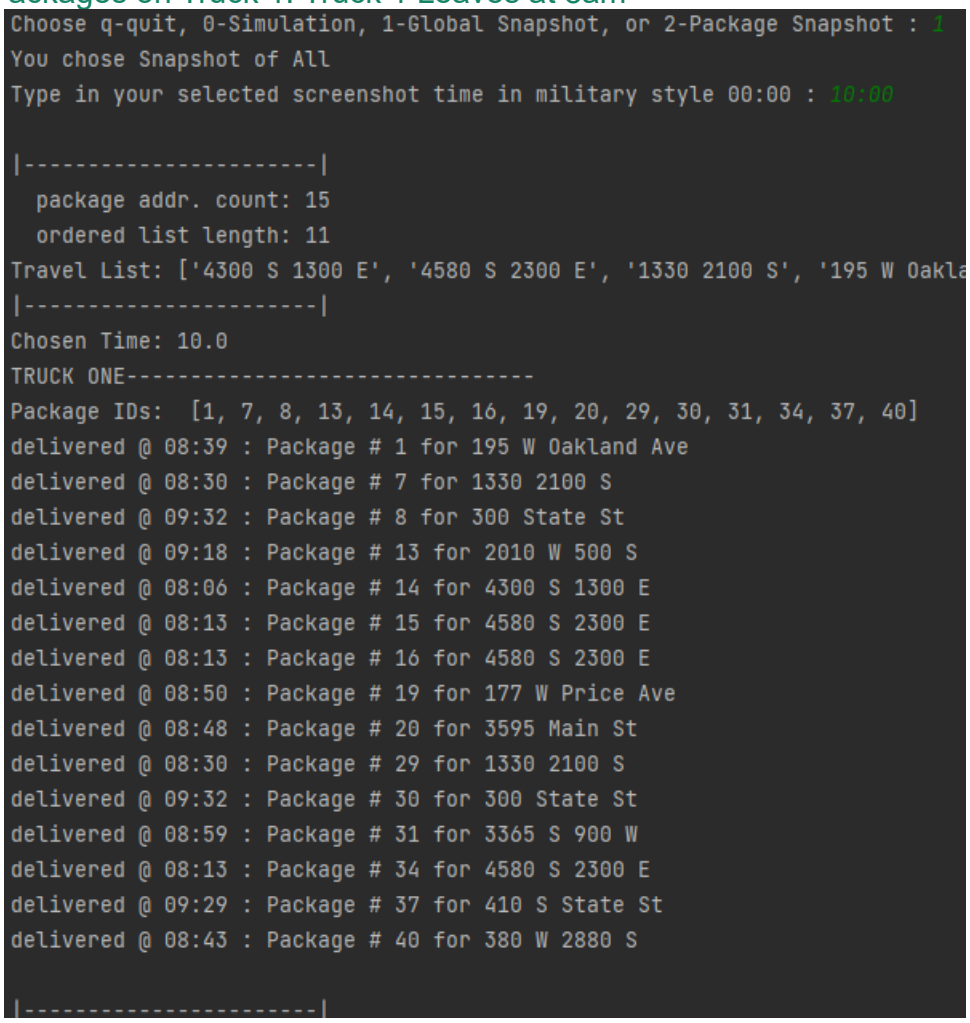
Packages on Truck 3: Leaves at 10:21

```
|----------------------|
  package addr. count: 12
  ordered list length: 11
Travel List: ['5383 South 900 East #104', '3595 Main St', '380 W 2880 S',
|----------------------|
Chosen Time: 9.0
TRUCK THREE-------------------------------
            at hub : Package # 4 for 380 W 2880 S
            at hub : Package # 5 for 410 S State St
            at hub : Package # 9 for 410 S State St
            at hub : Package # 10 for 600 E 900 South
            at hub : Package # 11 for 2600 Taylorsville Blvd
            at hub : Package # 17 for 3148 S 1100 W
            at hub : Package # 21 for 3595 Main St
            at hub : Package # 22 for 6351 South 900 East
            at hub : Package # 23 for 5100 South 2700 West
            at hub : Package # 24 for 5025 State St
            at hub : Package # 26 for 5383 South 900 East #104
            at hub : Package # 39 for 2010 W 500 S
Choose q-quit, 0-Simulation, 1-Global Snapshot, or 2-Package Snapshot :
```

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

Time: 09:00

Packages on Truck 1: Truck 1 Leaves at 8am



```
Choose q-quit, 0-Simulation, 1-Global Snapshot, or 2-Package Snapshot : 1
You chose Snapshot of All
Type in your selected screenshot time in military style 00:00 : 10:00

|----------------------|
  package addr. count: 15
  ordered list length: 11
Travel List: ['4300 S 1300 E', '4580 S 2300 E', '1330 2100 S', '195 W Oakla
|----------------------|
Chosen Time: 10.0
TRUCK ONE--------------------------------
Package IDs:  [1, 7, 8, 13, 14, 15, 16, 19, 20, 29, 30, 31, 34, 37, 40]
delivered @ 08:39 : Package # 1 for 195 W Oakland Ave
delivered @ 08:30 : Package # 7 for 1330 2100 S
delivered @ 09:32 : Package # 8 for 300 State St
delivered @ 09:18 : Package # 13 for 2010 W 500 S
delivered @ 08:06 : Package # 14 for 4300 S 1300 E
delivered @ 08:13 : Package # 15 for 4580 S 2300 E
delivered @ 08:13 : Package # 16 for 4580 S 2300 E
delivered @ 08:50 : Package # 19 for 177 W Price Ave
delivered @ 08:48 : Package # 20 for 3595 Main St
delivered @ 08:30 : Package # 29 for 1330 2100 S
delivered @ 09:32 : Package # 30 for 300 State St
delivered @ 08:59 : Package # 31 for 3365 S 900 W
delivered @ 08:13 : Package # 34 for 4580 S 2300 E
delivered @ 09:29 : Package # 37 for 410 S State St
delivered @ 08:43 : Package # 40 for 380 W 2880 S

|----------------------|
```

Packages on Truck 2: Leaves at 9:06

```
|----------------------|
  package addr. count: 13
  ordered list length: 11
Travel List: ['5383 South 900 East #104', '2530 S 500 E', '2835 Main St', '3
|----------------------|
Chosen Time: 10.0
TRUCK TWO-------------------------------
Package IDs:  [2, 3, 6, 12, 18, 25, 27, 28, 32, 33, 35, 36, 38]
delivered @ 09:15 : Package # 2 for 2530 S 500 E
         en route : Package # 3 for 233 Canyon Rd
delivered @ 09:34 : Package # 6 for 3060 Lester St
         en route : Package # 12 for 3575 W Valley Central Station bus Loop
         en route : Package # 18 for 1488 4800 S
delivered @ 09:06 : Package # 25 for 5383 South 900 East #104
delivered @ 09:48 : Package # 27 for 1060 Dalton Ave S
delivered @ 09:19 : Package # 28 for 2835 Main St
delivered @ 09:29 : Package # 32 for 3365 S 900 W
delivered @ 09:15 : Package # 33 for 2530 S 500 E
delivered @ 09:48 : Package # 35 for 1060 Dalton Ave S
delivered @ 09:39 : Package # 36 for 2300 Parkway Blvd
         en route : Package # 38 for 410 S State St

|----------------------|
```

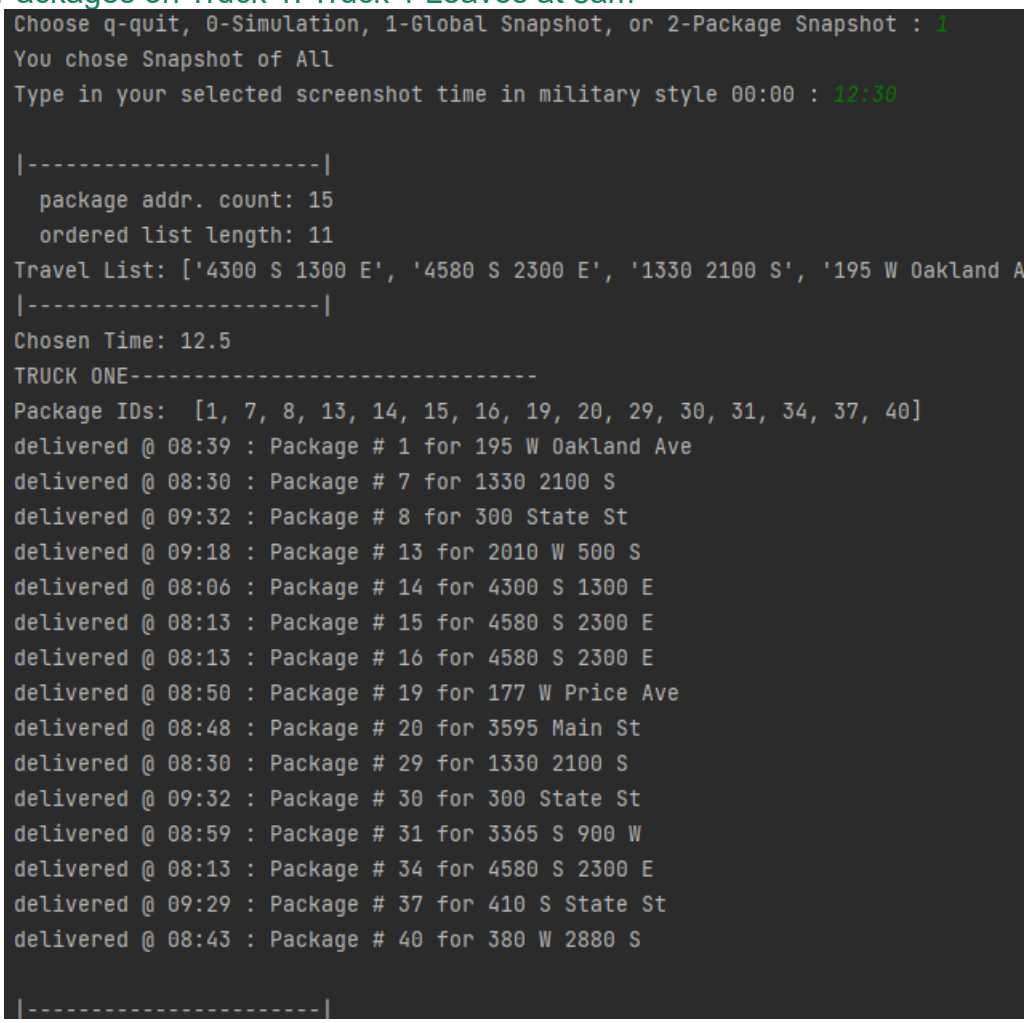
Packages on Truck 3: Leaves at 10:21

```
|----------------------|
  package addr. count: 12
  ordered list length: 11
Travel List: ['5383 South 900 East #104', '3595 Main St', '380 W 2880 S', '3
|----------------------|
Chosen Time: 10.0
TRUCK THREE-----------------------------
            at hub : Package # 4 for 380 W 2880 S
            at hub : Package # 5 for 410 S State St
            at hub : Package # 9 for 410 S State St
            at hub : Package # 10 for 600 E 900 South
            at hub : Package # 11 for 2600 Taylorsville Blvd
            at hub : Package # 17 for 3148 S 1100 W
            at hub : Package # 21 for 3595 Main St
            at hub : Package # 22 for 6351 South 900 East
            at hub : Package # 23 for 5100 South 2700 West
            at hub : Package # 24 for 5025 State St
            at hub : Package # 26 for 5383 South 900 East #104
            at hub : Package # 39 for 2010 W 500 S
Choose q-quit, 0-Simulation, 1-Global Snapshot, or 2-Package Snapshot :
```

3. Provide screenshots to show the status of all packages at a time between 12:03 p.m. and 1:12 p.m.

Time: 12:30

Packages on Truck 1: Truck 1 Leaves at 8am



```
Choose q-quit, 0-Simulation, 1-Global Snapshot, or 2-Package Snapshot : 1
You chose Snapshot of All
Type in your selected screenshot time in military style 00:00 : 12:30

|----------------------|
  package addr. count: 15
  ordered list length: 11
Travel List: ['4300 S 1300 E', '4580 S 2300 E', '1330 2100 S', '195 W Oakland A
|----------------------|
Chosen Time: 12.5
TRUCK ONE--------------------------------
Package IDs:  [1, 7, 8, 13, 14, 15, 16, 19, 20, 29, 30, 31, 34, 37, 40]
delivered @ 08:39 : Package # 1 for 195 W Oakland Ave
delivered @ 08:30 : Package # 7 for 1330 2100 S
delivered @ 09:32 : Package # 8 for 300 State St
delivered @ 09:18 : Package # 13 for 2010 W 500 S
delivered @ 08:06 : Package # 14 for 4300 S 1300 E
delivered @ 08:13 : Package # 15 for 4580 S 2300 E
delivered @ 08:13 : Package # 16 for 4580 S 2300 E
delivered @ 08:50 : Package # 19 for 177 W Price Ave
delivered @ 08:48 : Package # 20 for 3595 Main St
delivered @ 08:30 : Package # 29 for 1330 2100 S
delivered @ 09:32 : Package # 30 for 300 State St
delivered @ 08:59 : Package # 31 for 3365 S 900 W
delivered @ 08:13 : Package # 34 for 4580 S 2300 E
delivered @ 09:29 : Package # 37 for 410 S State St
delivered @ 08:43 : Package # 40 for 380 W 2880 S

|----------------------|
```

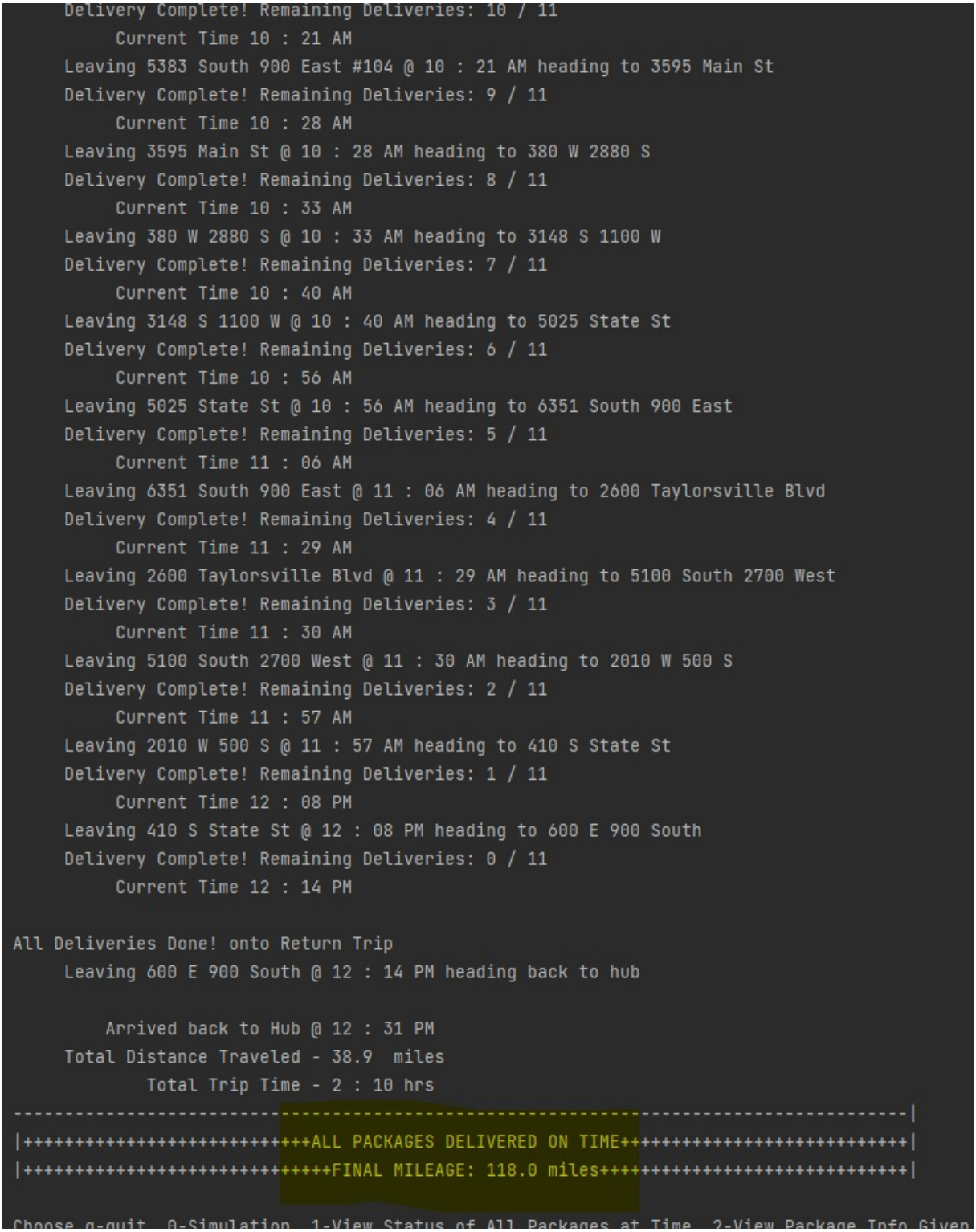
Packages on Truck 2: Leaves at 9:06

```
delivered @ 08:45 : Package # 40 for 380 W 2880 S

|----------------------|
  package addr. count: 13
  ordered list length: 11
Travel List: ['5383 South 900 East #104', '2530 S 500 E', '2835 Main St', '336
|----------------------|
Chosen Time: 12.5
TRUCK TWO-------------------------------
Package IDs:  [2, 3, 6, 12, 18, 25, 27, 28, 32, 33, 35, 36, 38]
delivered @ 09:15 : Package # 2 for 2530 S 500 E
delivered @ 10:08 : Package # 3 for 233 Canyon Rd
delivered @ 09:34 : Package # 6 for 3060 Lester St
delivered @ 10:32 : Package # 12 for 3575 W Valley Central Station bus Loop
delivered @ 10:56 : Package # 18 for 1488 4800 S
delivered @ 09:06 : Package # 25 for 5383 South 900 East #104
delivered @ 09:48 : Package # 27 for 1060 Dalton Ave S
delivered @ 09:19 : Package # 28 for 2835 Main St
delivered @ 09:29 : Package # 32 for 3365 S 900 W
delivered @ 09:15 : Package # 33 for 2530 S 500 E
delivered @ 09:48 : Package # 35 for 1060 Dalton Ave S
delivered @ 09:39 : Package # 36 for 2300 Parkway Blvd
delivered @ 10:04 : Package # 38 for 410 S State St

|----------------------|
```

Packages on Truck 3: Leaves at 10:21

```
delivered @ 10:04 : Package # 38 for 410 S State St

|----------------------|
  package addr. count: 12
  ordered list length: 11
Travel List: ['5383 South 900 East #104', '3595 Main St', '380 W 2880 S', '31
|----------------------|
Chosen Time: 12.5
TRUCK THREE-----------------------------
Package IDs:  [4, 5, 9, 10, 11, 17, 21, 22, 23, 24, 26, 39]
delivered @ 10:33 : Package # 4 for 380 W 2880 S
delivered @ 12:08 : Package # 5 for 410 S State St
delivered @ 12:08 : Package # 9 for 410 S State St
delivered @ 12:14 : Package # 10 for 600 E 900 South
delivered @ 11:29 : Package # 11 for 2600 Taylorsville Blvd
delivered @ 10:40 : Package # 17 for 3148 S 1100 W
delivered @ 10:28 : Package # 21 for 3595 Main St
delivered @ 11:06 : Package # 22 for 6351 South 900 East
delivered @ 11:30 : Package # 23 for 5100 South 2700 West
delivered @ 10:56 : Package # 24 for 5025 State St
delivered @ 10:21 : Package # 26 for 5383 South 900 East #104
delivered @ 11:57 : Package # 39 for 2010 W 500 S
Choose q-quit, 0-Simulation, 1-Global Snapshot, or 2-Package Snapshot :
```

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.

Nearest Neighbor makes no assumptions about data, whether it be in the number of packages, addresses, or times, which helps it accommodate the ever-evolving needs of the WGUPS. Using nearest neighbor provides a balance between speed (with a time complexity of O(n^2)) and efficiency, with the overall route being more effective than delivering packages in a package order, and the program running more quickly than a brute force algorithm.

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

Viewing a simulation in the program (screenshot section H) will show a breakdown of the trucks, what packages they have, and where they go, adding up the mileage and keeping track of the times. From that simulation we can see that the miles remain under 140 when all trucks are taken into account, and that all 40 packaged have been delivered, and to where, and what time the trucks return to the hub. **This simulation indicates that the current algorithm satisfies all delivery constraints while keeping the total mileage under 140 miles, as indicated in the scenario**

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

Greedy Algorithm, Genetic Algorithm

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

The greedy algorithm works by ordering the shortest pairs of points and then connecting all of them to form a path. A genetic algorithm runs simulations and swaps random pieces of information in them each time creating longer and shorter routes, the shortersroutes are then used to inform another generation of the process. Good for huge amounts of addresses, not as valuable for 40 packages

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

At the moment, the trucks are loaded, and then the routing algorithm is run based on just the packages for that truck individually. If I were to do this project again, I would run the algorithm on all the packages together, and then load packages based on distance from each other. That way two trucks aren't driving in the same location when one truck should be carrying all the packages for one spot

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 hashmap I wrote has an **add (insert) function** that can take any values, including the ones listed, and a **get (lookup) function** that can retrieve package information when a key is given in the form of the desired package id

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

As more packages are added to the hashmap, their information can be referenced directly in an O(1) fashion due to the resizing feature. In order to iterate over all packages in order to compare hashes to a known id, the time is O(n) and increases as n does of course.

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

Whenever the data starts to have numerous collisions, the map resizes, growing as more packages are added to keep lookup time low. As a result, **a large amount of packages would result in a large data structure space**

c. 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 algorithm in its current state only runs once per truck, so they're largely independent of the performance of the algorithm. As more cities are added, our CSV graph holding their relational info must be updated. The lookup time on this graph will remain n(1) as it grows, but the size increases by O(n^2) due to every address referencing every other in the graph.

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

Linked List, Graph

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

A linked list is a linear data structure where each package would reference the next until they'd all be chained together. A graph is an abstract, non-linear data structure where all the packages are represented by nodes, with their relationships (such as two packages with the same destination, or being on the same truck) indicated by edges

L. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

From Class Resources:

YouTube. (2016). Python: Creating a Hashmap using Lists. YouTube. Retrieved June 20, 2022, from https://www.youtube.com/watch?v=9HFbhPscPU0.