C950 WGUPS Algorithm Overview

Name: Mya Thomas
ID: 010507144
Email: mth1759@wgu.edu
Date: April 26, 2023

Introduction:

For this assessment, I was asked to implement an algorithm to route delivery trucks that will allow you to meet all delivery constraints while traveling under 140 miles. The program starts with 40 parcels that must be delivered around the Salt Lake City area by a specified deadline. Parcels have constraints or delays that must be accounted for before they can be successfully delivered. The program also contains a simple user interface that allows users to view all parcels at a specific time or view a single parcel by its unique id number.

A1. Algorithm Identification:

This program utilizes the nearest-neighbor algorithm to optimize the delivery route of each delivery truck. This algorithm starts by initializing all parcel addresses as 0, the delivery trucks start at the 'WGUPS' hub which is considered the starting point. From that point the algorithm checks for the closes address near the hub. Once it finds the next new address, it makes that new address the starting point and continues to repeat until all parcels have been delivered.

B1. Logic Comments:

The nearest neighbor algorithm that was implemented in my project, checks the manually loaded parcels with each other to determine which delivery address is the closes to the starting position. Afterward, the parcel that was previously the closes address becomes the new starting position. This pattern continues until all the parcels have been successfully delivered, leaving the delivery trucks empty.

1. Create a hash table (insert/update, search, remove)

```
CreateHashTable:
init(initial_capacity=40
insert(self, key, item)
search(self, key)
remove(self, key)
```

2. Load CSV data into a hash

```
read_parcel_csv(csv_file, hash_table)
def printHashTable(hash_table)
read_address_csv(csv_file, address_info)
read_distance_csv(csv_file, distance_info)
```

3. Initialize "WGUPS Hub' as starting address
4. Find the shortest distance from the hub to the next address

```
addressDistances_BetweenTwoPoints(
       addressA,
       addressB,
       address_info,
        distance_info)
   Return distance

ShortestDistance(
       address_now,
       parcel_list,
       hash_table,
       address_csv_info,
       distance_csv_info)
   shortestDistance = 1000
   next_stop = ''
   next_parcel_id = 0
   for parcel_id in parcel_list:
       parcel = parcel_id
       address2 = parcel
       distance = AddressDistances_BetweenTwoPoints(
           address_now,
           address2,
           address_csv_info,
           distance_csv_info)
       if distance == 0:
           shortestDistance = distance
           # Next Address
           next_stop = address2
           # Next Parcel
           next_parcel_id = parcel.parcel_id
           return next_stop, next_parcel_id, shortestDistance
       elif distance < shortestDistance:
           # Shortest distance
           shortestDistance = distance
           # Next Address
           next_stop = address2
           # Next Parcel
           next_parcel_id = parcel.parcel_id

   return next_stop, next_parcel_id, shortestDistance
```

5. Delivery Parcel, set the current address as the new starting position and remove the parcel from the list

```
Parcel_not_delivered.remove(parcel_id)
```

6. Repeat steps 4 and 5 until all parcels have been delivered

```
# Continues to deliver parcels until
# The 'Parcel_not_delivered' list is empty (0)
while len(Parcel_not_delivered) > 0:
```

B2. Development Environment:
This program was written using:
PyCharm 2021.3.3 (Community Edition)
Build #PC-213.7172.26, built on March 16, 2022
Runtime version: 11.0.14.1+1-b1751.46 x86_64
VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.
macOS 13.0.1
GC: G1 Young Generation, G1 Old Generation
Memory: 2048M
Cores: 12
The program contains all the appropriate files to run. If any files are missing or misplaced the program will not run. This program also runs locally on a machine and does not require the internet.

B3. Space-Time and Big-O:
A breakdown of the space-time complexity by each file in the program

File: main.py
Methods:

- def AddressDistance_BetweenTwoPoints	Time Complexity = O(1)
- def ShortestDistance	Time Complexity = O(N)
- def ParcelsToTruck	Time Complexity = O(1)
- def DeliverParcels	Time Complexity = O(N^2)
- def UI	Time Complexity = O(log n)
- def main	Time Complexity = O(N^2)

Total: (O(1) * O(1)) + O(N) + (O(N^2) * O(N^2)) + O(log n) = **O(N^2)**

File: HashTable.py
Methods:

- def __init__	Time Complexity = O(N)
- def insert	Time Complexity = O(N)
- def search	Time Complexity = O(N)
- def remove	Time Complexity = O(N)

Total: O(N) * O(N) * O(N) * O(N) = **O(N)**

File: Parcel.py
Method:

- def __init__	Time Complexity = O(1)
- def __str__	Time Complexity = O(1)
- def ReturnParcelPosition	Time Complexity = O(1)

Total: O(1) * O(1) * O(1) = **O(1)**

File: DeliveryTruckClass.py
Method:

- def __init__	Time Complexity = O(1)
- def __str__	Time Complexity = O(1)

Total: O(1) * O(1) = **O(1)**

File: Read_csvFile.py
Method:

- def read_parcel_csv	Time Complexity = O(N)
- def printHashTable	Time Complexity = O(N)
- def read_address_csv	Time Complexity = O(N)
- def read_distance_csv	Time Complexity = O(N)

Total: O(N) * O(N) * O(N) * O(N) = **O(N)**

Total run time:
O(N^2) + (O(N) * O(N)) + (O(1) * O(1)) = **O(N^2)**

B4. Scalability and Adaptability:
With a chaining hash table, items are able to be located faster in the table. Chaining allows the hash table to be scaled up or down depending on how many items are inserted or removed from the table.

B5. Software Efficiency and Maintainability:
The program's source code is organized to allow for easy readability and future adjustments. Each distinct component has a header tag and comments to help separate the code into sections for debugging purposes.

B6. Self-Adjusting Data Structures:
The chaining hash table allows for the continued growth of the table. By increasing the number of linked lists, the length of each individual list will decrease. Although, the more linked list in a hash table the longer the search time for an item will be.

C. Original Code:
When the program is executed:

```
- Mya Thomas
- ID: 010507144
- C950 Data Structures and Algorithms II
--------------------------------------------------------------------------------------------------
--------------------------------- WGUPS PARCEL DELIVERY TRACKING ---------------------------------

* Total Delivery Truck Mileage: 110.5 miles
Truck Mileage Breakdown:
 - Delivery Truck 1: 37.1 miles
 - Delivery Truck 2: 39.7 miles
 - Delivery Truck 3: 33.7 miles
----------------------------------------------
```

```
* Please Select From The Following Options Below
----------------------------------------------

* (Enter) 1: To View A Specific Parcel [1-40]

* (Enter) 2: To View All Parcels By Time

* (Enter) 3: Exit Program

-----------------------------------------------------------------------------------------------
```

When the program starts, the user is able to see the total mileage of all three delivery trucks as well as the individual mileage of each truck. The next prompt asks the user to enter either a '1' to view a specific parcel with a unique Id or a '2' to view all parcels at a specific time. Lastly, the user can enter '3' to quit the program entirely.

C. Original Code (Continued):
If the user enters '1' output:

```
-----------------------------------------------------------------------------------------------
1
Enter a Parcel ID number 1-40
23
 - Parcel ID:          23
 - Parcel Address:      5100 South 2700 West
 - Parcel City:         Salt Lake City
 - Parcel State:        UT
 - Postal Code:          84118

 - Deliver By: EOD,  Parcel Weight: 5.00,  Parcel Status: Delivered at 9:30:40

-----------------------------------------------------------------------------------------------
```

The user is prompted to enter a number between 1-40 (representing a unique Id for a parcel).

If the user enters '2' output:

```
-----------------------------------------------------------------------------------------------
2
Enter a time: (HH:MM:SS)
09:35:00
----------------------------------------------

Viewing All Parcels at time:  9:35:00
----------------------------------------------
```

```
[ Parcel Id: 1  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:39:59 ** ]
[ Parcel Id: 2  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:34:59 ** ]
[ Parcel Id: 3  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 4  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 5  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:04:18 ** ]
[ Parcel Id: 6  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 7  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:29:39 ** ]
[ Parcel Id: 8  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 9  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 10  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:10:18 ** ]
[ Parcel Id: 11  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:29:20 ** ]
[ Parcel Id: 12  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 13  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:27:18 ** ]
[ Parcel Id: 14  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:06:20 ** ]
[ Parcel Id: 15  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:12:59 ** ]
[ Parcel Id: 16  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:12:59 ** ]
[ Parcel Id: 17  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 18  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:32:40 ** ]
[ Parcel Id: 19  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:46:19 ** ]
[ Parcel Id: 20  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:47:58 ** ]
[ Parcel Id: 21  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 22  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 23  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:30:40 ** ]
[ Parcel Id: 24  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 25  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:13:00 ** ]
[ Parcel Id: 26  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 27  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 28  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 29  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:29:39 ** ]
[ Parcel Id: 30  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 31  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 32  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 33  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:34:59 ** ]
[ Parcel Id: 34  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:12:59 ** ]
[ Parcel Id: 35  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 36  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 37  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:04:18 ** ]
[ Parcel Id: 38  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 39  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:27:18 ** ]
[ Parcel Id: 40  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]

-----------------------------------------------------------------------------------------------
```

The user will be prompted to enter a time in the format (HH:MM:SS), once a time is entered and the user hits ‘return’ a list of parcel IDs, departure time, and status will appear for the user to view. This list will show what parcels have been delivered, delayed, and are out for delivery at that time entered by the user.

C1. Identification Information:
This information is also located at the top of each Python file included in the project, it also prints out when the program executes.
- Mya Thomas
- ID: 010507144
- C950 Data Structures and Algorithms II

C2. Process and Flow Comments:
Comments have been added to all classes, objects, and methods to allow for easier comprehension and organization.

D. Data Structure:
The data structure used to store parcel information was a chaining hash table. Each parcel’s information is stored in a list. That list is then stored inside the chaining hash table array. The parcel’s unique id is used as the key for the hash table. Each parcel has a unique id, so that helps mitigate collisions inside the hash table. With a chaining hash table, the size can grow or shrink automatically depending on whether an item is entered or removed.

D1. Explanation of Data Structure:
The hash table implemented in the program contains the following functions, search, remove, update, and insert.

def search:

```
# —-- SEARCH HASH TABLE --------------------------------------

# Searches for item in hash table
# Time Complexity -> O(N)
def search(self, key):
   bucket = hash(key) % len(self.table)
   bucket_list = self.table[bucket]

   for kv in bucket_list:
       if kv[0] == key:
           return kv[1]

   # Return 'None' if not found
   return None
```

def remove:

```
# --- REMOVE ITEM FROM HASH TABLE
--------------------------------------------------------------------------------
----------------

# Remove item from hash table
# Time Complexity -> O(N)
def remove(self, key):
   bucket = hash(key) % len(self.table)
   bucket_list = self.table[bucket]

   # If key is found, then remove item from
   # hash table
   for kv in bucket_list:
       if kv[0] == key:
           bucket_list.remove([kv[0], kv[1]])
           return print('Parcel ', key, ' has been delete from table')
   # If no item to remove is found return 'None'
   return None
```

D1. Explanation of Data Structure (Continued):

def insert/update:

```
# --- INSERT/UPDATE INTO HASH TABLE
--------------------------------------------------------------------------------
----------------

# Inserts a new item into hash table
# Time Complexity -> O(N)
def insert(self, key, item):
   # Get the list where the inserted item will reside
   bucket = hash(key) % len(self.table)
   bucket_list = self.table[bucket]

   # Time Complexity -> O(N)
   # Updates key if it already exists
   for kv in bucket_list:

       # return (key value)
       if kv[0] == key:
           kv[1] = item
           return True
   # If no key found, new item will be added
   # to the end of the list
   key_value = [key, item]
   bucket_list.append(key_value)
   return True
```

E. Hash Table:
Insert/update, search, and remove functions of the hash table are above in D1, and the search (look-up) function is also below in F.

```
# Set hash table size to 40
# Time Complexity -> O(N)
def __init__(self, initial_capacity=40):
   # Empty List
   self.table = []

   for i in range(initial_capacity):
       self.table.append([])
```

F. Look-Up Function:
def search:

```
# —-- SEARCH HASH TABLE --------------------------------------

# Searches for item in hash table
# Time Complexity -> O(N)
def search(self, key):
   bucket = hash(key) % len(self.table)
   bucket_list = self.table[bucket]

   for kv in bucket_list:
       if kv[0] == key:
           return kv[1]

   # Return 'None' if not found
   return None
```

G.Interface:
When the program is executed:

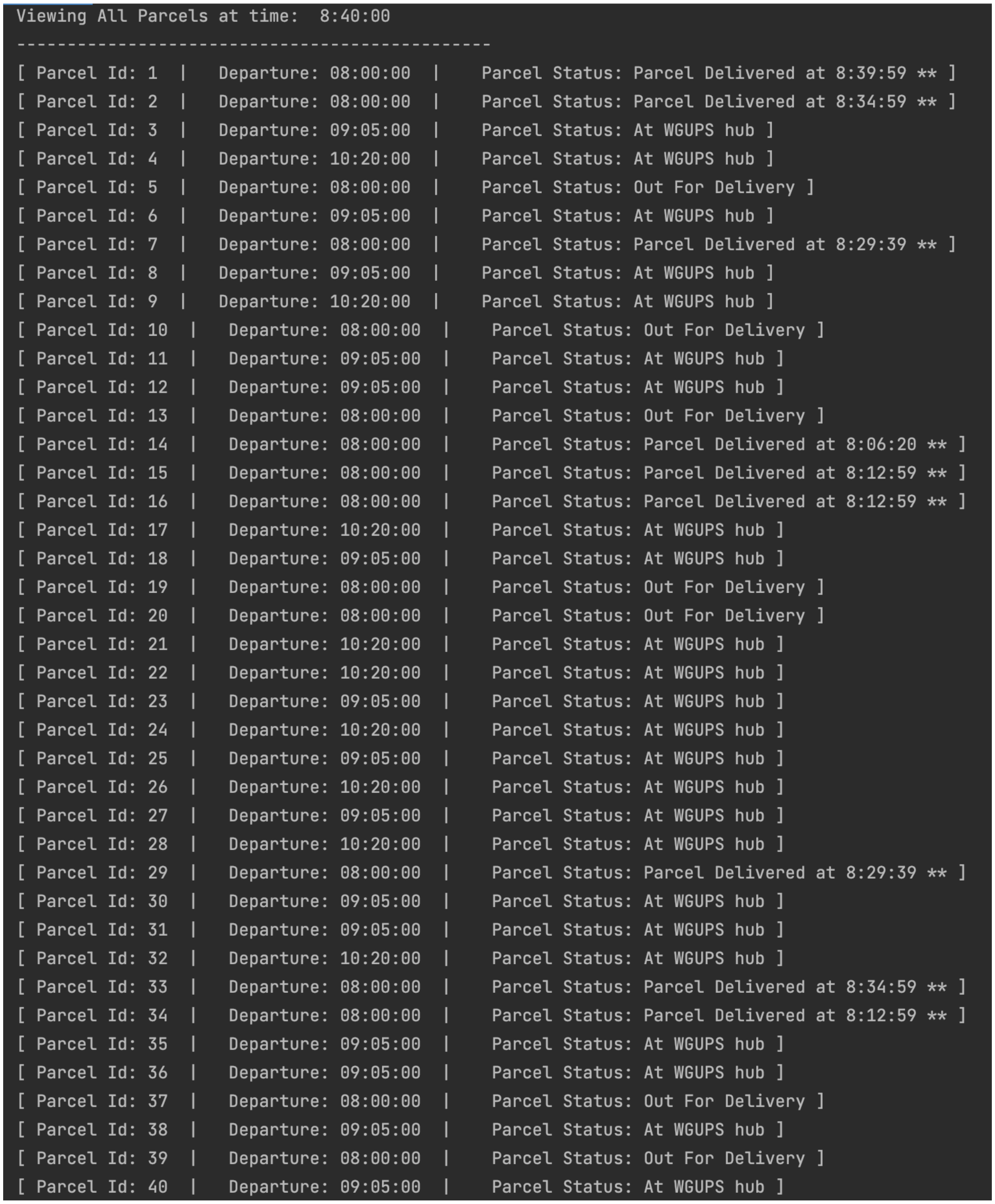
- Mya Thomas
- ID: 010507144
- C950 Data Structures and Algorithms II

--------------------------------- WGUPS PARCEL DELIVERY TRACKING ---------------------------------

```
* Total Delivery Truck Mileage: 110.5 miles
Truck Mileage Breakdown:
 - Delivery Truck 1: 37.1 miles
 - Delivery Truck 2: 39.7 miles
 - Delivery Truck 3: 33.7 miles
---------------------------------------------

* Please Select From The Following Options Below
---------------------------------------------

* (Enter) 1: To View A Specific Parcel [1-40]

* (Enter) 2: To View All Parcels By Time

* (Enter) 3: Exit Program

-----------------------------------------------------------------------------------------------
```
(CONTINUE BELOW)

G1. First Status Check:

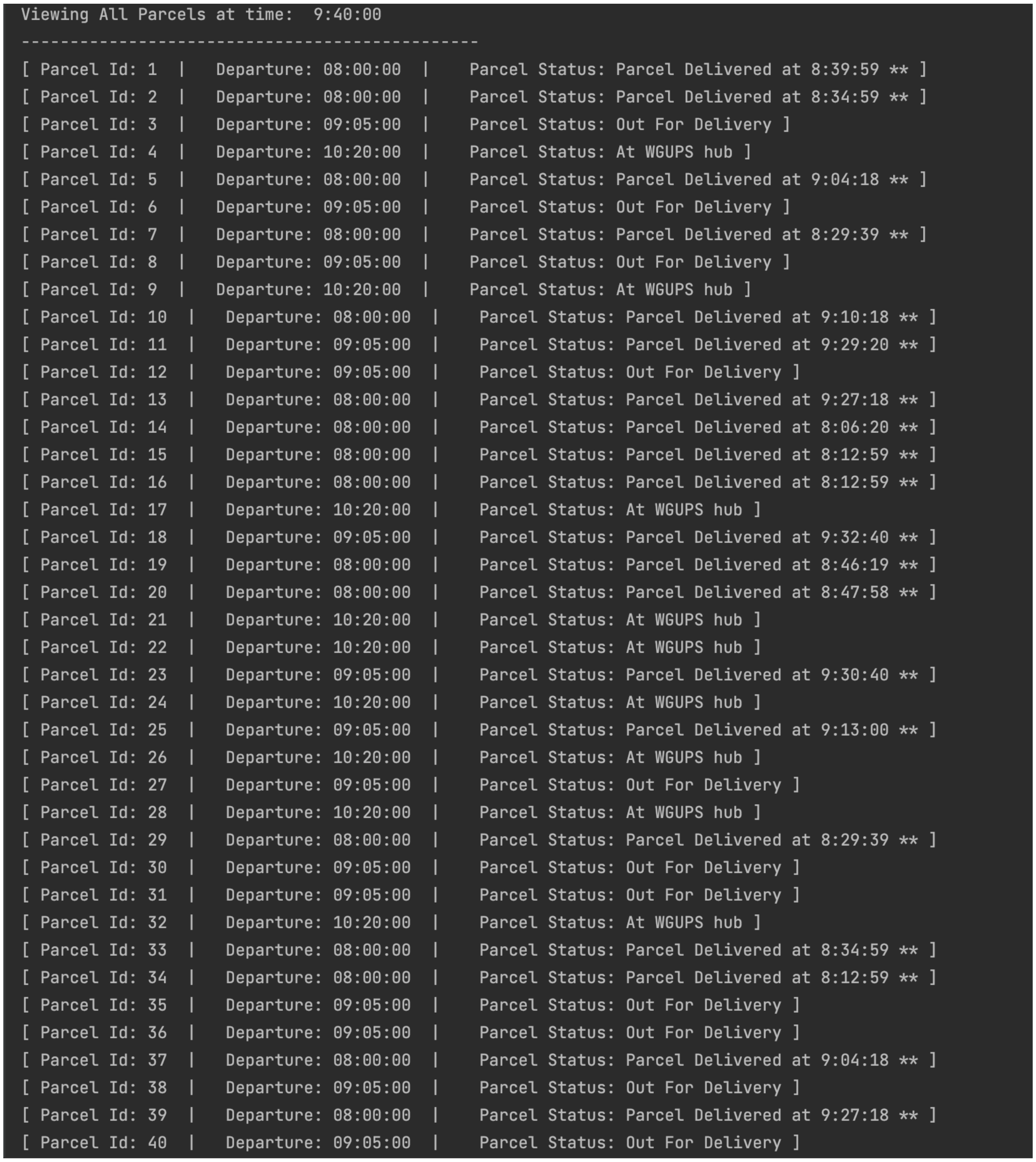
- Screenshot of all parcels: **08:40:00 AM**



```
Viewing All Parcels at time:  8:40:00
----------------------------------------------
[ Parcel Id: 1  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:39:59 ** ]
[ Parcel Id: 2  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:34:59 ** ]
[ Parcel Id: 3  |   Departure: 09:05:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 4  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 5  |   Departure: 08:00:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 6  |   Departure: 09:05:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 7  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:29:39 ** ]
[ Parcel Id: 8  |   Departure: 09:05:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 9  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 10  |   Departure: 08:00:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 11  |   Departure: 09:05:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 12  |   Departure: 09:05:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 13  |   Departure: 08:00:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 14  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:06:20 ** ]
[ Parcel Id: 15  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:12:59 ** ]
[ Parcel Id: 16  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:12:59 ** ]
[ Parcel Id: 17  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 18  |   Departure: 09:05:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 19  |   Departure: 08:00:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 20  |   Departure: 08:00:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 21  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 22  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 23  |   Departure: 09:05:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 24  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 25  |   Departure: 09:05:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 26  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 27  |   Departure: 09:05:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 28  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 29  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:29:39 ** ]
[ Parcel Id: 30  |   Departure: 09:05:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 31  |   Departure: 09:05:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 32  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 33  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:34:59 ** ]
[ Parcel Id: 34  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:12:59 ** ]
[ Parcel Id: 35  |   Departure: 09:05:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 36  |   Departure: 09:05:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 37  |   Departure: 08:00:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 38  |   Departure: 09:05:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 39  |   Departure: 08:00:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 40  |   Departure: 09:05:00  |    Parcel Status: At WGUPS hub ]
```

G2. Second Status Check:

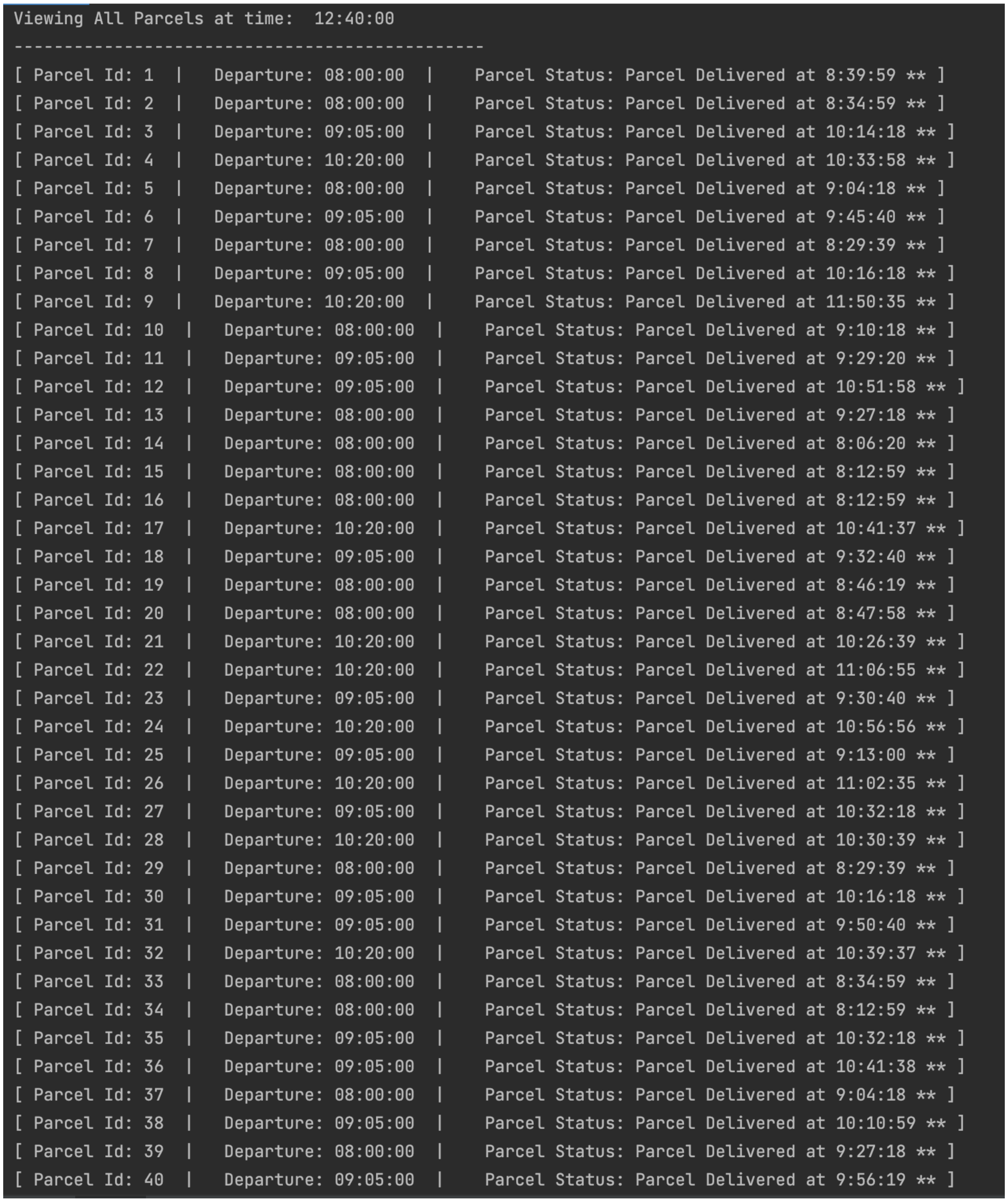
- Screenshot of all parcels: **09:40:00 AM**



```
Viewing All Parcels at time:  9:40:00
-----------------------------------------------
[ Parcel Id: 1  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:39:59 ** ]
[ Parcel Id: 2  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:34:59 ** ]
[ Parcel Id: 3  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 4  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 5  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:04:18 ** ]
[ Parcel Id: 6  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 7  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:29:39 ** ]
[ Parcel Id: 8  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 9  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 10  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:10:18 ** ]
[ Parcel Id: 11  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:29:20 ** ]
[ Parcel Id: 12  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 13  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:27:18 ** ]
[ Parcel Id: 14  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:06:20 ** ]
[ Parcel Id: 15  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:12:59 ** ]
[ Parcel Id: 16  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:12:59 ** ]
[ Parcel Id: 17  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 18  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:32:40 ** ]
[ Parcel Id: 19  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:46:19 ** ]
[ Parcel Id: 20  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:47:58 ** ]
[ Parcel Id: 21  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 22  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 23  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:30:40 ** ]
[ Parcel Id: 24  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 25  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:13:00 ** ]
[ Parcel Id: 26  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 27  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 28  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 29  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:29:39 ** ]
[ Parcel Id: 30  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 31  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 32  |   Departure: 10:20:00  |    Parcel Status: At WGUPS hub ]
[ Parcel Id: 33  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:34:59 ** ]
[ Parcel Id: 34  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:12:59 ** ]
[ Parcel Id: 35  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 36  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 37  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:04:18 ** ]
[ Parcel Id: 38  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
[ Parcel Id: 39  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:27:18 ** ]
[ Parcel Id: 40  |   Departure: 09:05:00  |    Parcel Status: Out For Delivery ]
```

G3. Third Status Check:

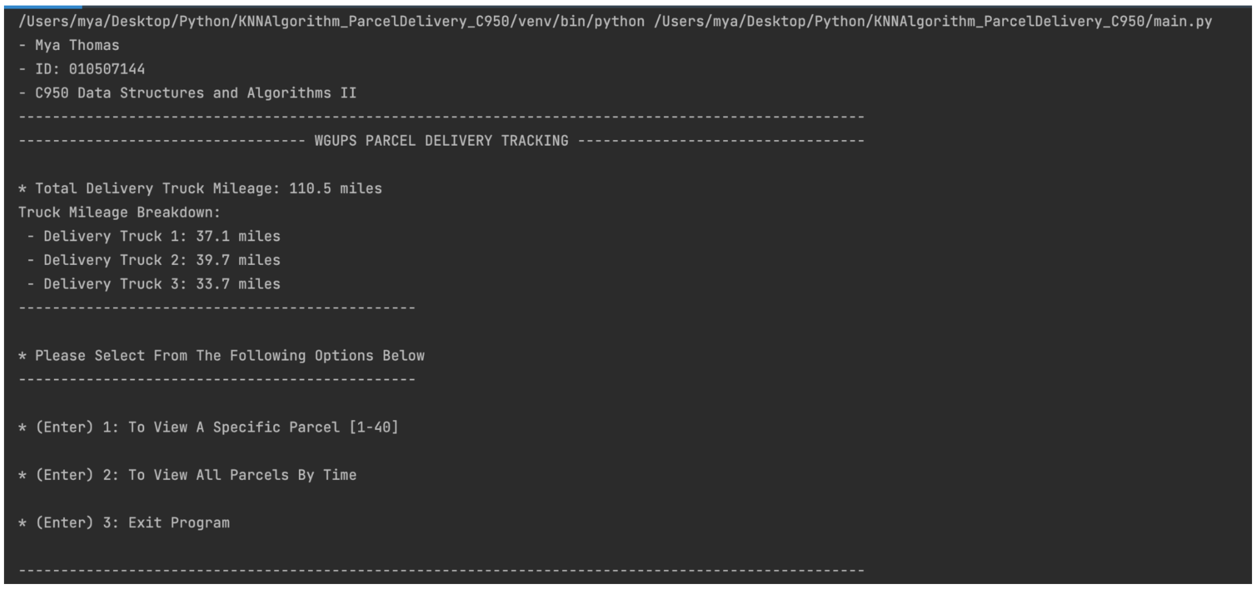
- Screenshot of all parcels: **12:40:00 AM**



```
Viewing All Parcels at time:  12:40:00
------------------------------------------------
[ Parcel Id: 1  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:39:59 ** ]
[ Parcel Id: 2  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:34:59 ** ]
[ Parcel Id: 3  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 10:14:18 ** ]
[ Parcel Id: 4  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 10:33:58 ** ]
[ Parcel Id: 5  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:04:18 ** ]
[ Parcel Id: 6  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:45:40 ** ]
[ Parcel Id: 7  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:29:39 ** ]
[ Parcel Id: 8  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 10:16:18 ** ]
[ Parcel Id: 9  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 11:50:35 ** ]
[ Parcel Id: 10  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:10:18 ** ]
[ Parcel Id: 11  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:29:20 ** ]
[ Parcel Id: 12  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 10:51:58 ** ]
[ Parcel Id: 13  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:27:18 ** ]
[ Parcel Id: 14  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:06:20 ** ]
[ Parcel Id: 15  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:12:59 ** ]
[ Parcel Id: 16  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:12:59 ** ]
[ Parcel Id: 17  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 10:41:37 ** ]
[ Parcel Id: 18  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:32:40 ** ]
[ Parcel Id: 19  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:46:19 ** ]
[ Parcel Id: 20  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:47:58 ** ]
[ Parcel Id: 21  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 10:26:39 ** ]
[ Parcel Id: 22  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 11:06:55 ** ]
[ Parcel Id: 23  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:30:40 ** ]
[ Parcel Id: 24  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 10:56:56 ** ]
[ Parcel Id: 25  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:13:00 ** ]
[ Parcel Id: 26  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 11:02:35 ** ]
[ Parcel Id: 27  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 10:32:18 ** ]
[ Parcel Id: 28  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 10:30:39 ** ]
[ Parcel Id: 29  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:29:39 ** ]
[ Parcel Id: 30  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 10:16:18 ** ]
[ Parcel Id: 31  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:50:40 ** ]
[ Parcel Id: 32  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 10:39:37 ** ]
[ Parcel Id: 33  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:34:59 ** ]
[ Parcel Id: 34  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:12:59 ** ]
[ Parcel Id: 35  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 10:32:18 ** ]
[ Parcel Id: 36  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 10:41:38 ** ]
[ Parcel Id: 37  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:04:18 ** ]
[ Parcel Id: 38  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 10:10:59 ** ]
[ Parcel Id: 39  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:27:18 ** ]
[ Parcel Id: 40  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:56:19 ** ]
```

- **All parcels have been successfully delivered**

H. Screenshot of Code Execution:



```
/Users/mya/Desktop/Python/KNNAlgorithm_ParcelDelivery_C950/venv/bin/python /Users/mya/Desktop/Python/KNNAlgorithm_ParcelDelivery_C950/main.py
- Mya Thomas
- ID: 010507144
- C950 Data Structures and Algorithms II
----------------------------------------------------------------------------------------------------
-------------------------------- WGUPS PARCEL DELIVERY TRACKING ---------------------------------

* Total Delivery Truck Mileage: 110.5 miles
Truck Mileage Breakdown:
 - Delivery Truck 1: 37.1 miles
 - Delivery Truck 2: 39.7 miles
 - Delivery Truck 3: 33.7 miles
-----------------------------------------------

* Please Select From The Following Options Below
-----------------------------------------------

* (Enter) 1: To View A Specific Parcel [1-40]

* (Enter) 2: To View All Parcels By Time

* (Enter) 3: Exit Program

----------------------------------------------------------------------------------------------------
```

I1. Strengths of Chosen Algorithm:

The nearest neighbor algorithm implemented in this program is easy to understand, it also is recommended for non-linear data. The disadvantage of using the nearest neighbor algorithm is that this algorithm is not recommended to be used with a very large value of data. This algorithm also can require a large amount of memory storage.

I2. Verification of Algorithm:

The algorithm can be viewed in the above screenshots (G1- G3) or by executing the program.

I3. Other Possible Algorithms:

The Dijkstra Algorithm or the Breadth First Algorithm could have been used to get a similar result to the implementation of the nearest neighbor algorithm. Dijkstra implements a graph with values and vertices to find the shortest path. BFS starts with a starting vertex and then divides into visited and unvisited vertices, instead of using a chaining hash table.

I3A. Algorithm Differences

Difference between Dijkstra Algorithm and Nearest Neighbor Algorithm:

The Nearest Neighbor Algorithm is considered a classification algorithm while Dijkstra's algorithm is considered a graph search algorithm.

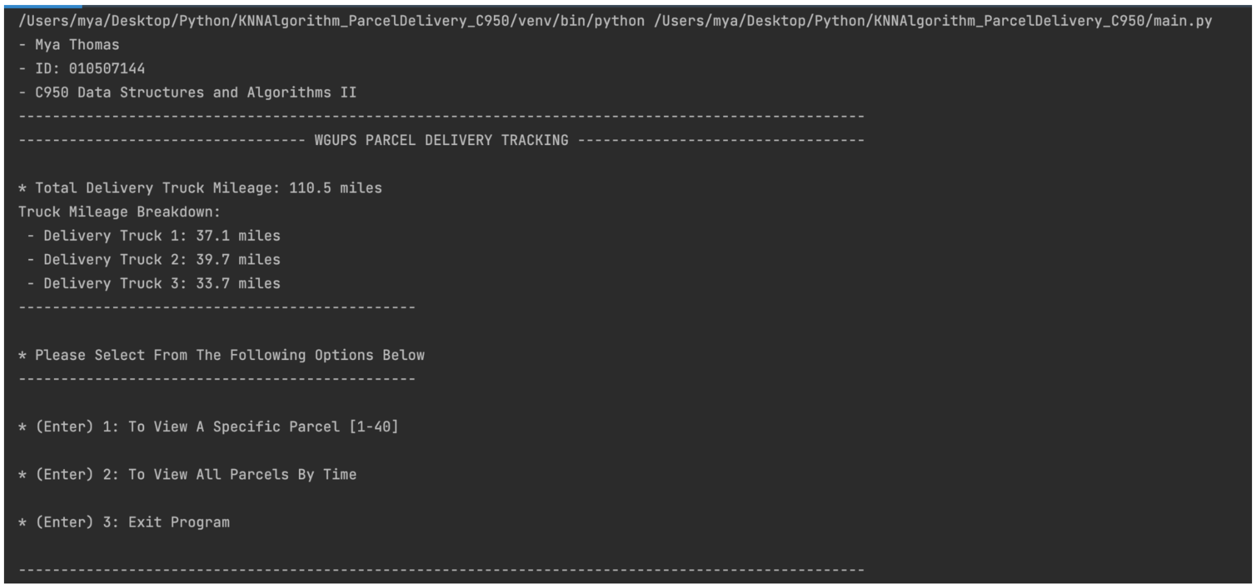
The nearest neighbor calculates the distance between items and located the closest one. Dijkstra usually uses a graph to locate the shortest path between points.

J. Different Approach:

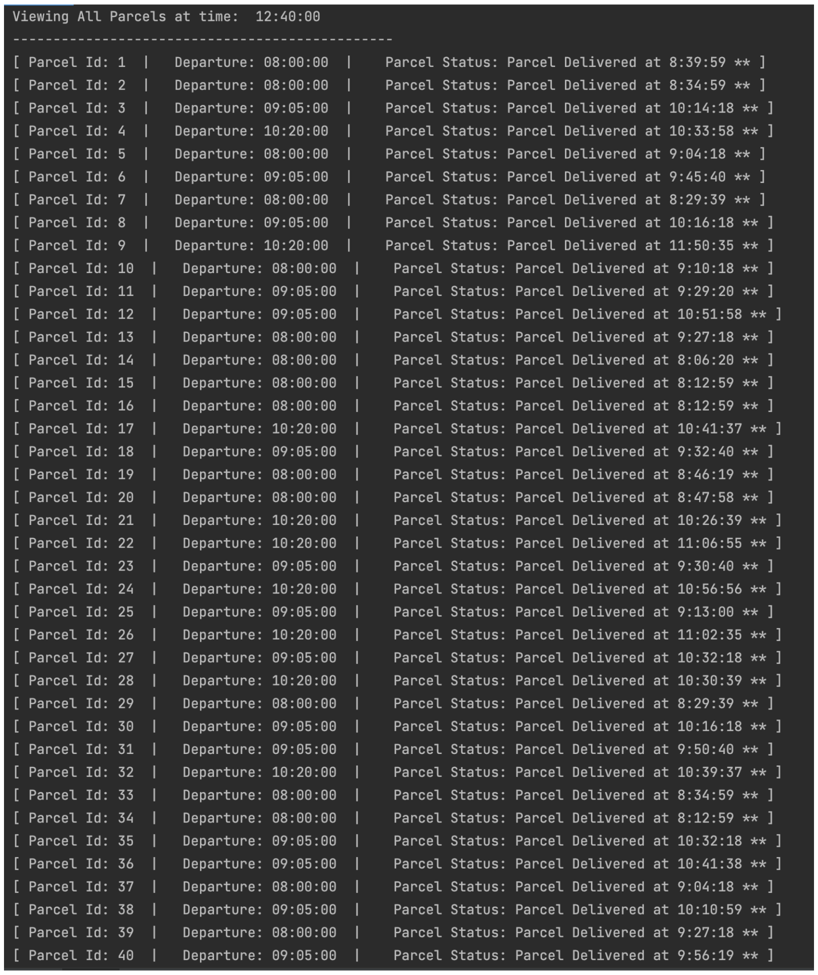
A different approach I would take if I restarted this project would be probably creating a more engaging and interesting user interface. Maybe adding more colors and imagery to the program through a GUI. I also would want to find a fast route for parcel delivery and play around with different algorithms to see the results and how they differ. Also, using the SQL database to store, adjust, or delete parcel and delivery truck information would be another way to manage the data associated with the project.

K1. Verification of Data Structures:

The above sections and screenshots show how the hash table implemented in the program delivers all parcels under 140 total miles.



```
/Users/mya/Desktop/Python/KNNAlgorithm_ParcelDelivery_C950/venv/bin/python /Users/mya/Desktop/Python/KNNAlgorithm_ParcelDelivery_C950/main.py
- Mya Thomas
- ID: 010507144
- C950 Data Structures and Algorithms II
----------------------------------------------------------------------------------------------------
---------------------------------- WGUPS PARCEL DELIVERY TRACKING ----------------------------------

* Total Delivery Truck Mileage: 110.5 miles
Truck Mileage Breakdown:
 - Delivery Truck 1: 37.1 miles
 - Delivery Truck 2: 39.7 miles
 - Delivery Truck 3: 33.7 miles
-----------------------------------------------

* Please Select From The Following Options Below
-----------------------------------------------

* (Enter) 1: To View A Specific Parcel [1-40]

* (Enter) 2: To View All Parcels By Time

* (Enter) 3: Exit Program

----------------------------------------------------------------------------------------------------
```



```
Viewing All Parcels at time:  12:40:00
---------------------------------------------
[ Parcel Id: 1  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:39:59 ** ]
[ Parcel Id: 2  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:34:59 ** ]
[ Parcel Id: 3  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 10:14:18 ** ]
[ Parcel Id: 4  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 10:33:58 ** ]
[ Parcel Id: 5  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:04:18 ** ]
[ Parcel Id: 6  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:45:40 ** ]
[ Parcel Id: 7  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:29:39 ** ]
[ Parcel Id: 8  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 10:16:18 ** ]
[ Parcel Id: 9  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 11:50:35 ** ]
[ Parcel Id: 10  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:10:18 ** ]
[ Parcel Id: 11  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:29:20 ** ]
[ Parcel Id: 12  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 10:51:58 ** ]
[ Parcel Id: 13  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:27:18 ** ]
[ Parcel Id: 14  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:06:20 ** ]
[ Parcel Id: 15  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:12:59 ** ]
[ Parcel Id: 16  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:12:59 ** ]
[ Parcel Id: 17  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 10:41:37 ** ]
[ Parcel Id: 18  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:32:40 ** ]
[ Parcel Id: 19  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:46:19 ** ]
[ Parcel Id: 20  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:47:58 ** ]
[ Parcel Id: 21  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 10:26:39 ** ]
[ Parcel Id: 22  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 11:06:55 ** ]
[ Parcel Id: 23  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:30:40 ** ]
[ Parcel Id: 24  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 10:56:56 ** ]
[ Parcel Id: 25  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:13:00 ** ]
[ Parcel Id: 26  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 11:02:35 ** ]
[ Parcel Id: 27  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 10:32:18 ** ]
[ Parcel Id: 28  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 10:30:39 ** ]
[ Parcel Id: 29  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:29:39 ** ]
[ Parcel Id: 30  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 10:16:18 ** ]
[ Parcel Id: 31  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:50:40 ** ]
[ Parcel Id: 32  |   Departure: 10:20:00  |    Parcel Status: Parcel Delivered at 10:39:37 ** ]
[ Parcel Id: 33  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:34:59 ** ]
[ Parcel Id: 34  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 8:12:59 ** ]
[ Parcel Id: 35  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 10:32:18 ** ]
[ Parcel Id: 36  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 10:41:38 ** ]
[ Parcel Id: 37  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:04:18 ** ]
[ Parcel Id: 38  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 10:10:59 ** ]
[ Parcel Id: 39  |   Departure: 08:00:00  |    Parcel Status: Parcel Delivered at 9:27:18 ** ]
[ Parcel Id: 40  |   Departure: 09:05:00  |    Parcel Status: Parcel Delivered at 9:56:19 ** ]
```

-

K1A.Efficiency:
The hash table implemented in the program is efficient for this purpose. As there are only three trucks and 40 parcels, it is not a vast and massive amount of data to handle. The program run time will scale linearly as more parcels are loaded into the program.

K1B. Overhead:
The program run time will increase linearly as the size of the data increases in size as well. More memory would have to be used as the hash table grows in size.

K1C.Implications:
Using the provided CSV files, helped with implementing the project. The address CSV contained the different addresses that the delivery trucks had to visit. The distance CSV provided the distances between locations and the parcel CSV contain the information regarding the parcels. It was important to know which parcels where delay, or which parcels needed to be delivered first (due to deadlines) or together with other parcels.

K2.Other Data Structures:
Binary search trees, stacks, and queues could also be used in the program to meet all requirements

K2A. Data Structures Differences:
Difference between the following below:

Hash Table:
- The hash table is a data structure that stores information in a similar manner. The data is stored within a list inside an array of lists. Each data value is associated with a unique index value. Hash tables allow for very fast access to data that is stored within the table.

Queue:
- The queue data structure is a linear data structure that has an open start and end. Queues perform the FIFO (first in, first out) operation. Where data that is inserted in first is accessed before data that is recently entered in. (Similar to humans standing in line at a register).

Stack:
- The stack data structure is also a linear data structure. Stacks perform the LIFO (Last in, first out) operation, where the data is stacked on top of each other, the recently entered data will be the first to be accessed. (Similar to Pringle chips in a tube can)

Binary Search Tree:
- The binary search tree is a collection of nodes. Each node has a key and an associated value. The search operation starts with the root node, then if the root data is less than the key value, the left subtree will be searched, else, the right subtree will be searched.

L.Sources:

- *Zybooks*, https://learn.zybooks.com/zybook/WGUC950AY20182019. https://learn.zybooks.com/zybook/WGUC950AY20182019

- “Find Shortest Paths from Source to All Vertices Using Dijkstra's Algorithm.” *GeeksforGeeks*, GeeksforGeeks, 28 Mar. 2023, https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/.