C950 WGUPS Algorithm Overview

Christina LaPane
ID #008207171
WGU Email: clapane@wgu.edu

C950 Data Structures and Algorithms II

Introduction

This project is to create an algorithm for the WGUPS Salt Lake City route to determine the best route for their deliveries. Currently, deadlines are not consistently being met. This route delivers an average of 40 packages a day using two trucks and three drivers.

Assumptions:

Each truck can carry a maximum amount of 16 packages

Trucks average speed is 18mph

Trucks have an unlimited amount of gas

There are no collisions between trucks

Trucks are loaded by 8:00 AM and can leave the hub no earlier

End of day is when all 40 packages are delivered

No time passes while at delivery location.

There is up to one special note for each package

Wrong delivery address for package 9 – known address at 10:20 AM

Package ID is unique

No further assumptions exist or are allowed.

Package Restrictions:

Package 3, 18, 36, 38 must all be on truck 2

Package 13, 15, 19 must be delivered together by 10:30 AM

Package 13 must be delivered by 9:00 AM

Packages 28 and 32 won't arrive to hub until 9:05

Packages 6 and 25 won't arrive to hub until 9:05 and must be delivered by 10:30 AM

Package 1, 29, 30, 31, 37 and 40 must be delivered by 10:30 AM

Package 9 address will update to correct one at 10:20 AM

Rest of packages don't have any restrictions or delivery deadlines

A. Algorithm Identification

The WGUPS utilizes the greedy algorithm to solve the delivery problem and is done in these steps:

1. Creating the best route
2. Creates distance graph
3. The greedy algorithm
4. A delivery simulation

Creating the best route:

Using the assumptions and restrictions, packages are sorted and loaded onto the three trucks. The packages that have a delivery deadline of 9:00 or 10:30 AM, but have no special notes are loaded onto truck 1. The packages that aren't arriving to the hub until 9:05 AM and have notes that say they can only be on truck 2 are loaded onto truck 2. The rest of the packages that don't have any deadlines or restrictions are loaded onto truck 3.

The packages delivery street address is extracted and put into the individual trucks route, which starts and ends with the hubs address.

Create distance graph:

The distance between all 40 delivery addresses are loaded and read into a list. A vertex is created which represent an address. Another vertex is created which is the next address. An edge weight is created which is the distance between the two vertices. A list is created which will hold the addresses and their edge weights. These will be loaded into the greedy algorithm

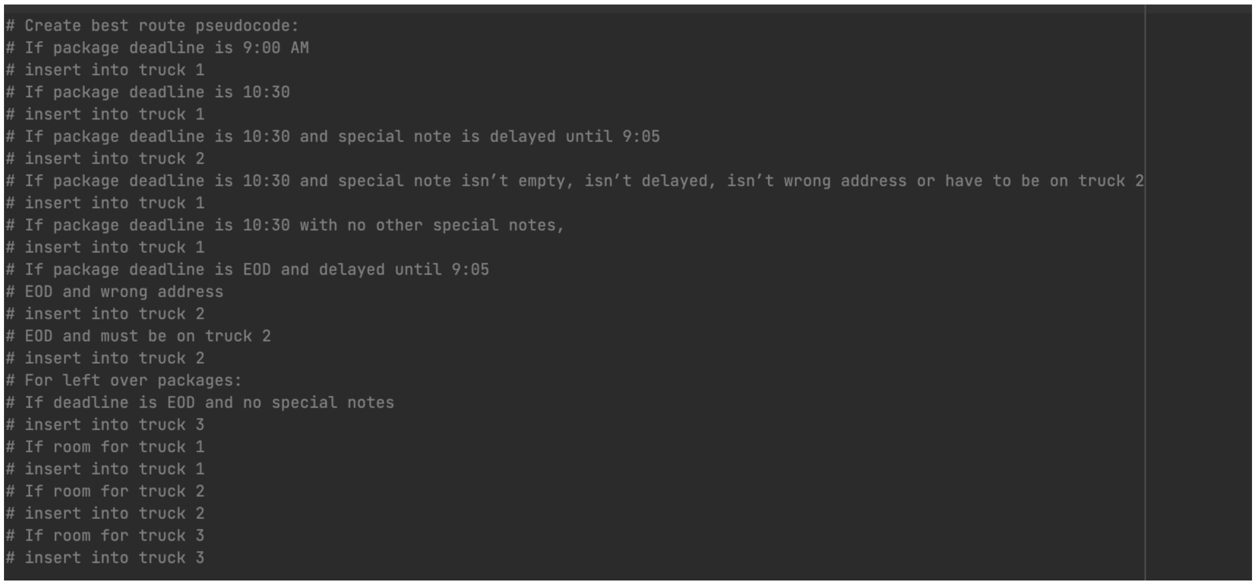
The Greedy Algorithm:

Now that each truck has their own route, Truck 1/2/3 Route, those routes are individually loaded into the greedy algorithm. The algorithm uses the trucks current location, which is the hubs address, and compares it to all the delivery addresses in their route to find the closest one - delivery location 1. The distance between the two addresses are stored as edge weights. The current location then becomes delivery location 1 and is compared to the other delivery addresses in the route to find the closest one – delivery location 2. The current location is then set to delivery location 2 and it keeps doing that until there are no more delivery addresses left in the route.

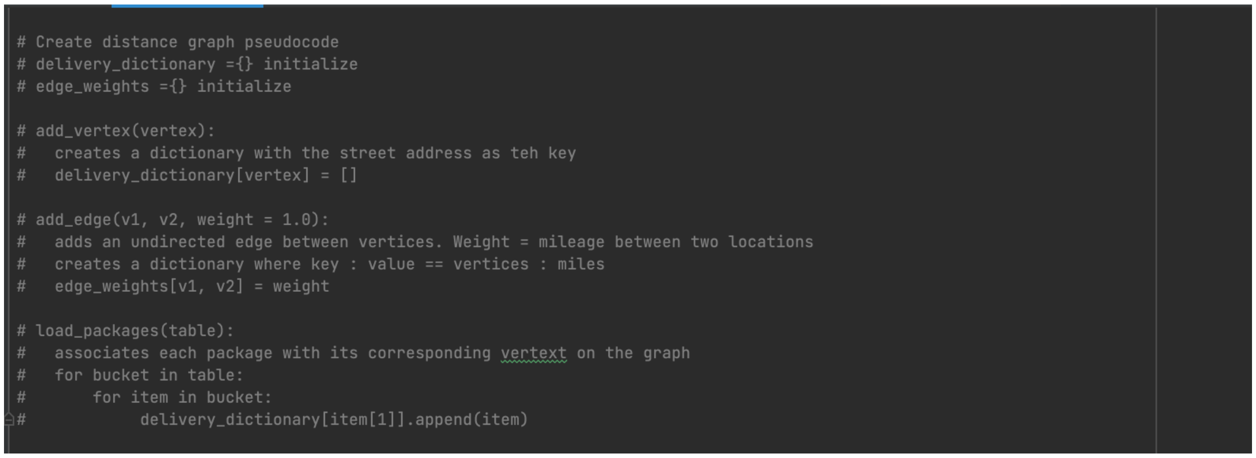
A delivery simulation:

Now that the greedy algorithm has created the shortest route for each truck, it is put into the delivery simulation where the route is tested against the time, to ensure delivery deadlines and that mileage is less than 140 miles. Each truck route is set to a start time – truck 1 leaves at 8:00 AM, truck 2 leaves at 9:05 because that is when the delayed packages are arriving to the hub, and then truck 3 leaves when truck 1 returns to the hub. The simulation goes through the route created at 18 mph. The time between each address is determined by the edge weights (distance) divided by 18 mph and multiplied by 60 minutes. These minutes will be added to the current time. As the program loops through the route, the current time and total edge weights are updated once a delivery is met.

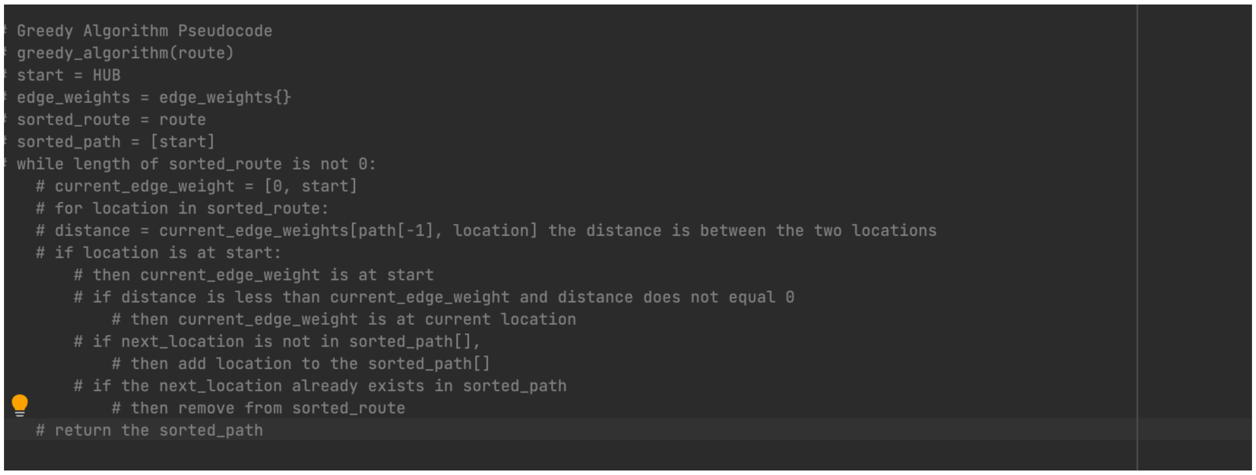
B1. Logic Comments



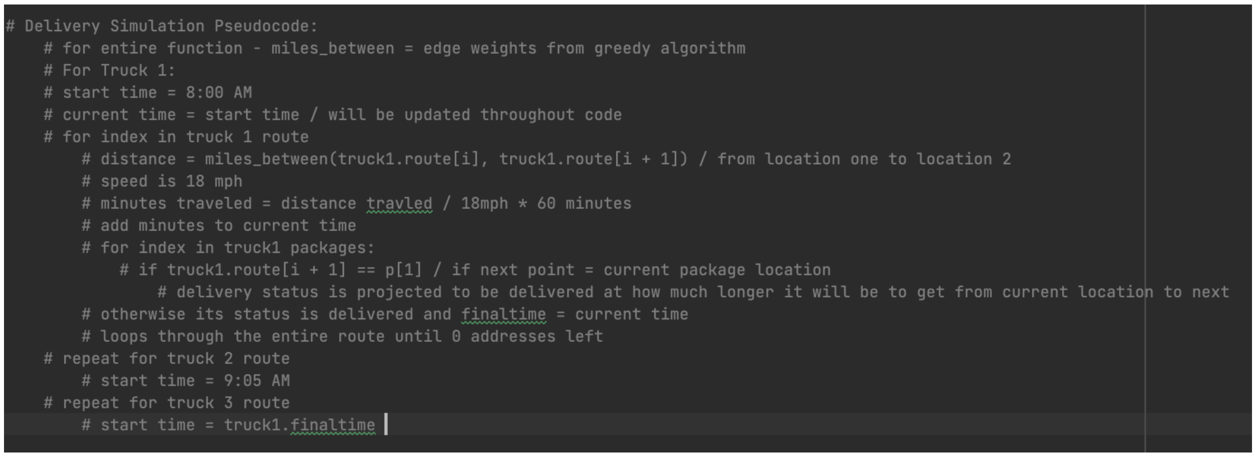
```
# Create best route pseudocode:
# If package deadline is 9:00 AM
# insert into truck 1
# If package deadline is 10:30
# insert into truck 1
# If package deadline is 10:30 and special note is delayed until 9:05
# insert into truck 2
# If package deadline is 10:30 and special note isn't empty, isn't delayed, isn't wrong address or have to be on truck 2
# insert into truck 1
# If package deadline is 10:30 with no other special notes,
# insert into truck 1
# If package deadline is EOD and delayed until 9:05
# EOD and wrong address
# insert into truck 2
# EOD and must be on truck 2
# insert into truck 2
# For left over packages:
# If deadline is EOD and no special notes
# insert into truck 3
# If room for truck 1
# insert into truck 1
# If room for truck 2
# insert into truck 2
# If room for truck 3
# insert into truck 3
```



```
# Create distance graph pseudocode
# delivery_dictionary ={} initialize
# edge_weights ={} initialize

# add_vertex(vertex):
#   creates a dictionary with the street address as teh key
#   delivery_dictionary[vertex] = []

# add_edge(v1, v2, weight = 1.0):
#   adds an undirected edge between vertices. Weight = mileage between two locations
#   creates a dictionary where key : value == vertices : miles
#   edge_weights[v1, v2] = weight

# load_packages(table):
#   associates each package with its corresponding vertext on the graph
#   for bucket in table:
#       for item in bucket:
#           delivery_dictionary[item[1]].append(item)
```



```
# Greedy Algorithm Pseudocode
# greedy_algorithm(route)
# start = HUB
# edge_weights = edge_weights{}
# sorted_route = route
# sorted_path = [start]
# while length of sorted_route is not 0:
    # current_edge_weight = [0, start]
    # for location in sorted_route:
    # distance = current_edge_weights[path[-1], location] the distance is between the two locations
    # if location is at start:
        # then current_edge_weight is at start
        # if distance is less than current_edge_weight and distance does not equal 0
            # then current_edge_weight is at current location
        # if next_location is not in sorted_path[],
            # then add location to the sorted_path[]
        # if the next_location already exists in sorted_path
            # then remove from sorted_route
    # return the sorted_path
```



```
# Delivery Simulation Pseudocode:
    # for entire function - miles_between = edge weights from greedy algorithm
    # For Truck 1:
    # start time = 8:00 AM
    # current time = start time / will be updated throughout code
    # for index in truck 1 route
        # distance = miles_between(truck1.route[i], truck1.route[i + 1]) / from location one to location 2
        # speed is 18 mph
        # minutes traveled = distance travled / 18mph * 60 minutes
        # add minutes to current time
        # for index in truck1 packages:
            # if truck1.route[i + 1] == p[1] / if next point = current package location
                # delivery status is projected to be delivered at how much longer it will be to get from current location to next
        # otherwise its status is delivered and finaltime = current time
        # loops through the entire route until 0 addresses left
    # repeat for truck 2 route
        # start time = 9:05 AM
    # repeat for truck 3 route
        # start time = truck1.finaltime
```

B2. Development Environment

PyCharm 2021.3.2 the community edition.

VM : OpenJDK

Language : Python 3

WGUPS is implemented as a command-line interface that executes locally on the users computer.

B3. Space-Time and Big-O

graph.py	Space Complexity	Time Complexity
add_vertex	O(N)	O(N)
add_edge	O(N)	O(N)
load_packages	O(N^2)	O(N^2)
read_distance_csv(filename)	O(N^2)	O(N^2)
get_graph(filename)	O(N^2)	O(N^2)

greedy.py	Space Complexity	Time Complexity
algorithm(route)	O(log N)	O(log N)

hash.py	Space Complexity	Time Complexity
insert(self, key, package)	O(N)	O(N)
search(self, key)	O(N)	O(N)
remove(self, key)	O(N)	O(N)
get_package_data(file)	O(N)	O(N)

main.py	Space Complexity	Time Complexity
ui	O(N)	O(N)

mytime.py	Space Complexity	Time Complexity
__init__(self, *args)	O(N)	O(N)
add_minutes(self, minutes)	O(N)	O(N)
__eq__(self, value)	O(N)	O(N)
__ne__(self, value)	O(N)	O(N)
__lt__(self, value)	O(N)	O(N)
__le__(self, value)	O(N)	O(N)
__gt__(self, value)	O(N)	O(N)
__ge__(self, value)	O(N)	O(N)
__str__(self)	O(N)	O(N)

package.py	Space Complexity	Time Complexity
none	O(N)	O(N)

truck.py	Space Complexity	Time Complexity
insert(self, package)	O(N)	O(N)
remove(self, package)	O(N)	O(N)
start(self, current)	O(N)	O(N)
current(self, current)	O(N)	O(N)
final(self, final)	O(N)	O(N)
create_best_route	O(N^2)	O(N^2)
traveled(route)	O(N)	O(N)
total_miles	O(N)	O(N)
deliver	O(N^2)	O(N^2)
first_round	O(N^2)	O(N^2)
secondRound	O(N^2)	O(N^2)
thirdRound	O(N^2)	O(N^2)
allTimes	O(N^2)	O(N^2)
search_id(package_id)	O(N^2)	O(N^2)

B4. Scalability and Adaptability

The core function of the program is designed to create a route for any number of packages, any number of routes, and trucks without causing the runtime to be affected too much. The algorithm's time complexity is O(log N) which is the best run time. As far as loading the data and creating the route, the time complexities are O(N^2), which the more data the slower the runtime will be.

The one way that the program is not adaptable is the way the packages are loaded into the trucks. The trucks are loaded by specific deadlines and special notes, which would not work the same for another round of packages. Overall, this program would not be suitable for anything outside of these 40 packages.

B5. Software Efficiency and Maintainability

Overall, the program is efficient with the overall time complexity being O(N^2). It does not do well with increasing the data, but it works well and efficiently with the current data. It's easy to understand what each method is doing because everything is well documented and if there is an issue with a method someone could easily refer to another method, because they are all somewhat similar.

B6. Self-Adjusting Data Structures

The primary data structure used throughout the program was the linear hash table used to store and utilize the data, throughout the entire program. The hash table is self-adjusting because it will expand its storage with the more data that is loaded into it. The strength of the hash table is that information can be found through a key, which was the package ID in this program. The negative side to this is the effect that it will have on the runtime. If there is not an initial capacity, the table will constantly expand and rehash itself which can be expensive.

A graph was also created to self-adjust the data from the hash table. The graph data's strength is that the data can be inserted and removed quickly and update the path to take, without much effect on runtime. The downside is that the more data, the slower the runtime is.

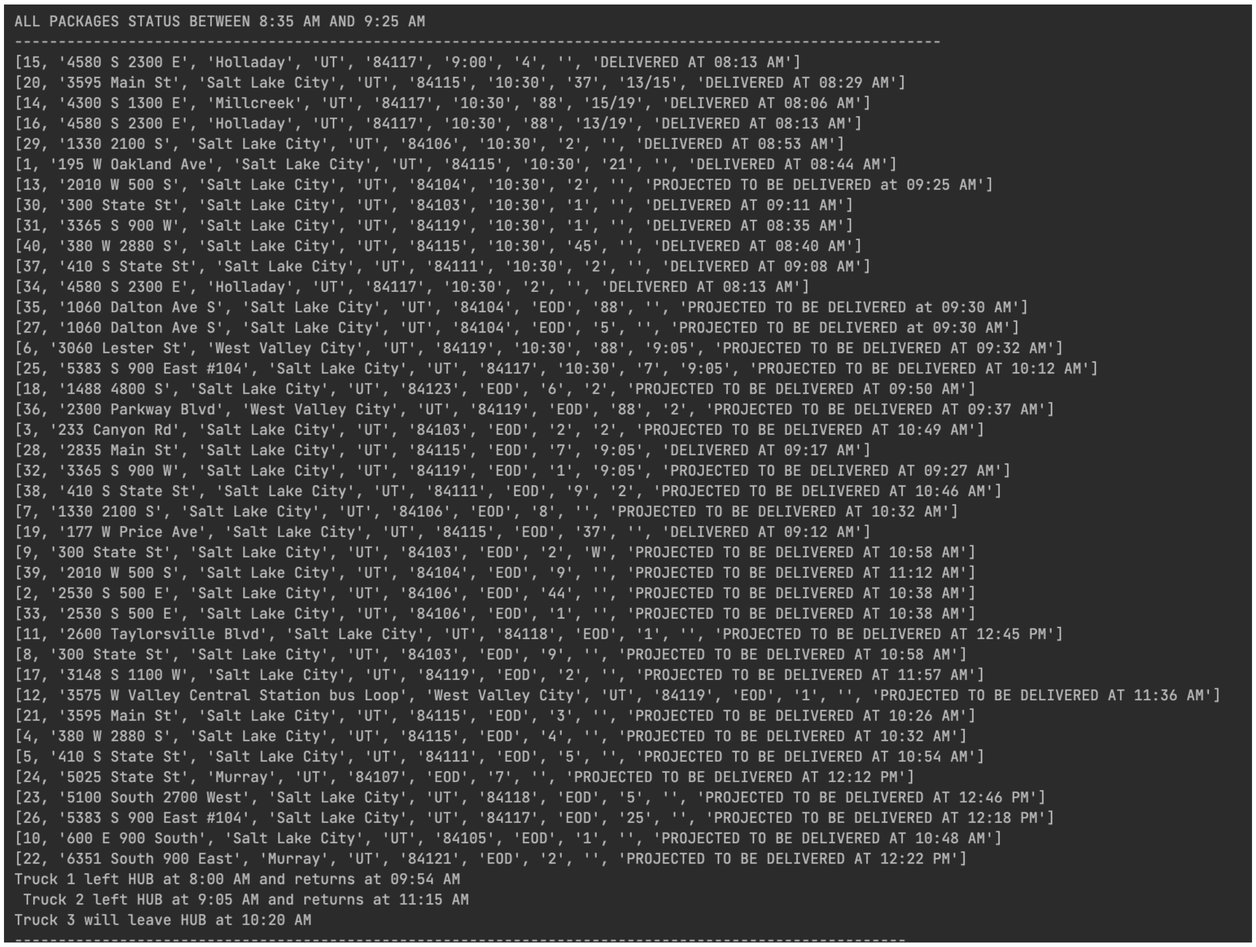
D. Data Structure

As mentioned in B6, the self-adjusting structure that holds the package information is the chaining hash table. The information is searched, inserted, and removed by referencing the key, which is the package ID in this program. The hash table runs at a time complexity of O(N) so no matter how much data is being entered, the runtime is fairly low.

D1. Explanation of Data Structure

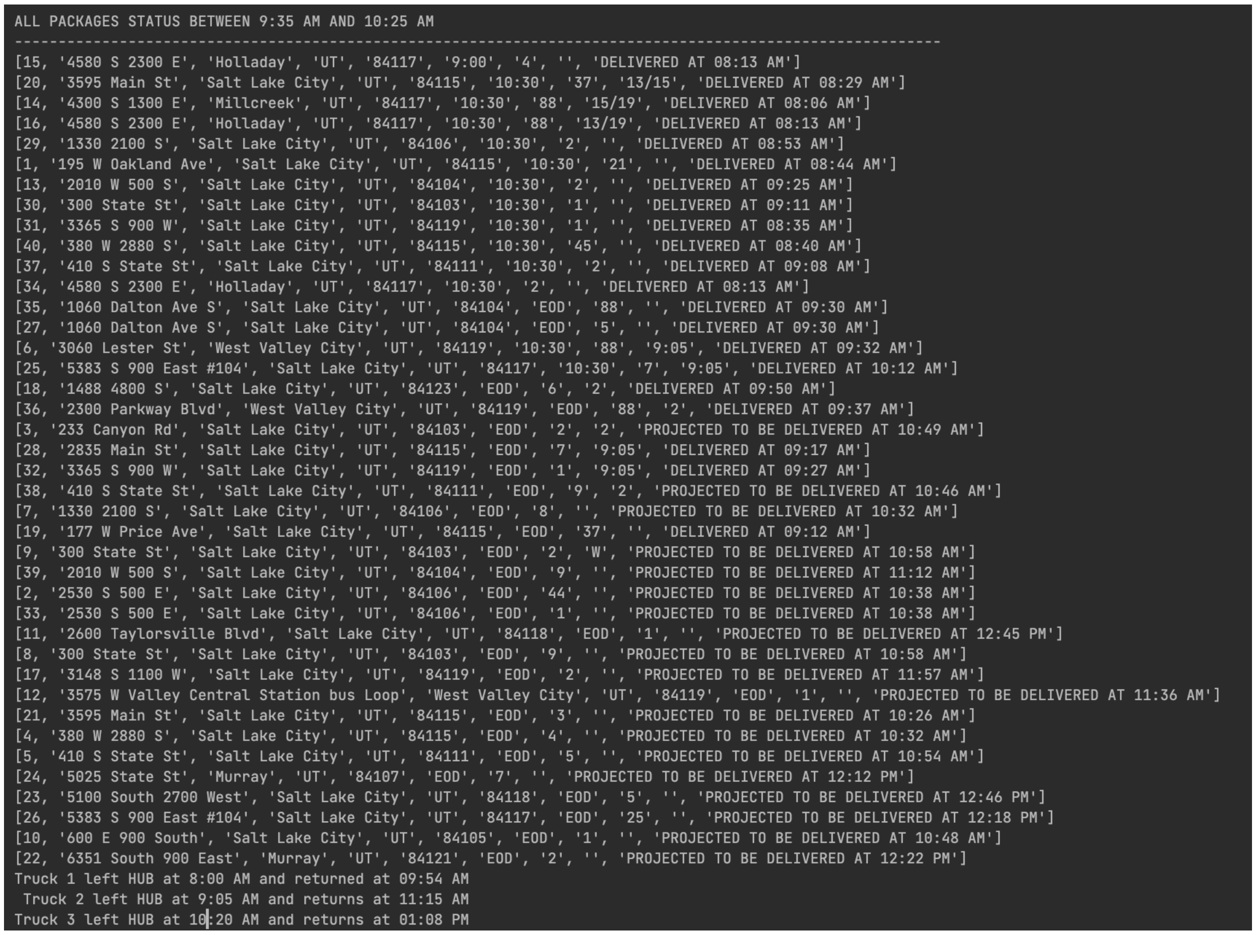
The hash table starts off by reading the package csv and skips the first row, because that's the header. It then goes row by row and stores the information and uses index 0, the package ID, as the key. The key will be used throughout the program to gather all package information. The package hash table will be cross referenced with the distance data to figure out what addresses are closest together.

G1. First Status Check



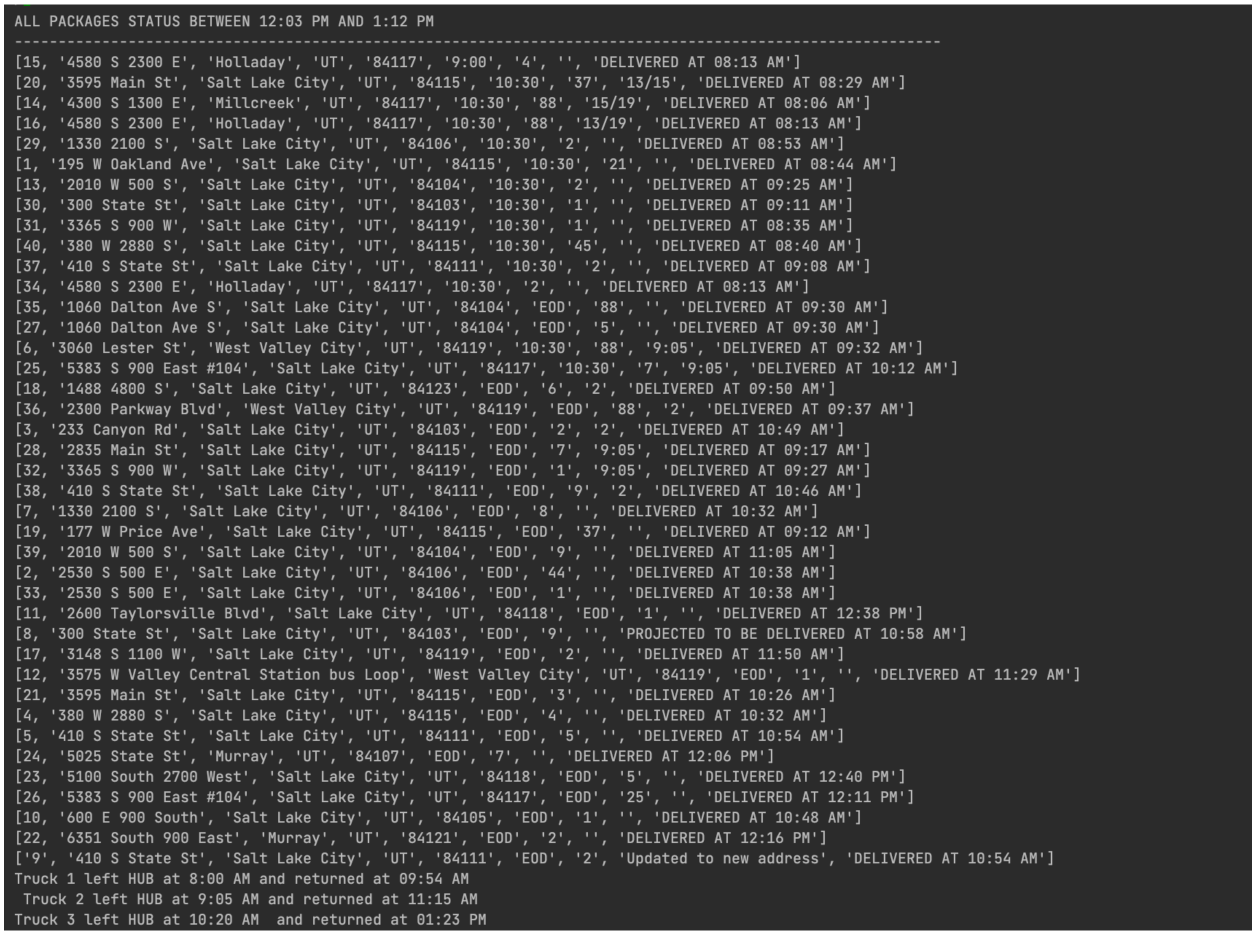
```
ALL PACKAGES STATUS BETWEEN 8:35 AM AND 9:25 AM
-----------------------------------------------------------------------------------------------------------
[15, '4580 S 2300 E', 'Holladay', 'UT', '84117', '9:00', '4', '', 'DELIVERED AT 08:13 AM']
[20, '3595 Main St', 'Salt Lake City', 'UT', '84115', '10:30', '37', '13/15', 'DELIVERED AT 08:29 AM']
[14, '4300 S 1300 E', 'Millcreek', 'UT', '84117', '10:30', '88', '15/19', 'DELIVERED AT 08:06 AM']
[16, '4580 S 2300 E', 'Holladay', 'UT', '84117', '10:30', '88', '13/19', 'DELIVERED AT 08:13 AM']
[29, '1330 2100 S', 'Salt Lake City', 'UT', '84106', '10:30', '2', '', 'DELIVERED AT 08:53 AM']
[1, '195 W Oakland Ave', 'Salt Lake City', 'UT', '84115', '10:30', '21', '', 'DELIVERED AT 08:44 AM']
[13, '2010 W 500 S', 'Salt Lake City', 'UT', '84104', '10:30', '2', '', 'PROJECTED TO BE DELIVERED at 09:25 AM']
[30, '300 State St', 'Salt Lake City', 'UT', '84103', '10:30', '1', '', 'DELIVERED AT 09:11 AM']
[31, '3365 S 900 W', 'Salt Lake City', 'UT', '84119', '10:30', '1', '', 'DELIVERED AT 08:35 AM']
[40, '380 W 2880 S', 'Salt Lake City', 'UT', '84115', '10:30', '45', '', 'DELIVERED AT 08:40 AM']
[37, '410 S State St', 'Salt Lake City', 'UT', '84111', '10:30', '2', '', 'DELIVERED AT 09:08 AM']
[34, '4580 S 2300 E', 'Holladay', 'UT', '84117', '10:30', '2', '', 'DELIVERED AT 08:13 AM']
[35, '1060 Dalton Ave S', 'Salt Lake City', 'UT', '84104', 'EOD', '88', '', 'PROJECTED TO BE DELIVERED at 09:30 AM']
[27, '1060 Dalton Ave S', 'Salt Lake City', 'UT', '84104', 'EOD', '5', '', 'PROJECTED TO BE DELIVERED at 09:30 AM']
[6, '3060 Lester St', 'West Valley City', 'UT', '84119', '10:30', '88', '9:05', 'PROJECTED TO BE DELIVERED AT 09:32 AM']
[25, '5383 S 900 East #104', 'Salt Lake City', 'UT', '84117', '10:30', '7', '9:05', 'PROJECTED TO BE DELIVERED AT 10:12 AM']
[18, '1488 4800 S', 'Salt Lake City', 'UT', '84123', 'EOD', '6', '2', 'PROJECTED TO BE DELIVERED AT 09:50 AM']
[36, '2300 Parkway Blvd', 'West Valley City', 'UT', '84119', 'EOD', '88', '2', 'PROJECTED TO BE DELIVERED AT 09:37 AM']
[3, '233 Canyon Rd', 'Salt Lake City', 'UT', '84103', 'EOD', '2', '2', 'PROJECTED TO BE DELIVERED AT 10:49 AM']
[28, '2835 Main St', 'Salt Lake City', 'UT', '84115', 'EOD', '7', '9:05', 'DELIVERED AT 09:17 AM']
[32, '3365 S 900 W', 'Salt Lake City', 'UT', '84119', 'EOD', '1', '9:05', 'PROJECTED TO BE DELIVERED AT 09:27 AM']
[38, '410 S State St', 'Salt Lake City', 'UT', '84111', 'EOD', '9', '2', 'PROJECTED TO BE DELIVERED AT 10:46 AM']
[7, '1330 2100 S', 'Salt Lake City', 'UT', '84106', 'EOD', '8', '', 'PROJECTED TO BE DELIVERED AT 10:32 AM']
[19, '177 W Price Ave', 'Salt Lake City', 'UT', '84115', 'EOD', '37', '', 'DELIVERED AT 09:12 AM']
[9, '300 State St', 'Salt Lake City', 'UT', '84103', 'EOD', '2', 'W', 'PROJECTED TO BE DELIVERED AT 10:58 AM']
[39, '2010 W 500 S', 'Salt Lake City', 'UT', '84104', 'EOD', '9', '', 'PROJECTED TO BE DELIVERED AT 11:12 AM']
[2, '2530 S 500 E', 'Salt Lake City', 'UT', '84106', 'EOD', '44', '', 'PROJECTED TO BE DELIVERED AT 10:38 AM']
[33, '2530 S 500 E', 'Salt Lake City', 'UT', '84106', 'EOD', '1', '', 'PROJECTED TO BE DELIVERED AT 10:38 AM']
[11, '2600 Taylorsville Blvd', 'Salt Lake City', 'UT', '84118', 'EOD', '1', '', 'PROJECTED TO BE DELIVERED AT 12:45 PM']
[8, '300 State St', 'Salt Lake City', 'UT', '84103', 'EOD', '9', '', 'PROJECTED TO BE DELIVERED AT 10:58 AM']
[17, '3148 S 1100 W', 'Salt Lake City', 'UT', '84119', 'EOD', '2', '', 'PROJECTED TO BE DELIVERED AT 11:57 AM']
[12, '3575 W Valley Central Station bus Loop', 'West Valley City', 'UT', '84119', 'EOD', '1', '', 'PROJECTED TO BE DELIVERED AT 11:36 AM']
[21, '3595 Main St', 'Salt Lake City', 'UT', '84115', 'EOD', '3', '', 'PROJECTED TO BE DELIVERED AT 10:26 AM']
[4, '380 W 2880 S', 'Salt Lake City', 'UT', '84115', 'EOD', '4', '', 'PROJECTED TO BE DELIVERED AT 10:32 AM']
[5, '410 S State St', 'Salt Lake City', 'UT', '84111', 'EOD', '5', '', 'PROJECTED TO BE DELIVERED AT 10:54 AM']
[24, '5025 State St', 'Murray', 'UT', '84107', 'EOD', '7', '', 'PROJECTED TO BE DELIVERED AT 12:12 PM']
[23, '5100 South 2700 West', 'Salt Lake City', 'UT', '84118', 'EOD', '5', '', 'PROJECTED TO BE DELIVERED AT 12:46 PM']
[26, '5383 S 900 East #104', 'Salt Lake City', 'UT', '84117', 'EOD', '25', '', 'PROJECTED TO BE DELIVERED AT 12:18 PM']
[10, '600 E 900 South', 'Salt Lake City', 'UT', '84105', 'EOD', '1', '', 'PROJECTED TO BE DELIVERED AT 10:48 AM']
[22, '6351 South 900 East', 'Murray', 'UT', '84121', 'EOD', '2', '', 'PROJECTED TO BE DELIVERED AT 12:22 PM']
Truck 1 left HUB at 8:00 AM and returns at 09:54 AM
 Truck 2 left HUB at 9:05 AM and returns at 11:15 AM
Truck 3 will leave HUB at 10:20 AM
-------------------------------------------------------------------------------------------------------
```

G2. Second Status Check



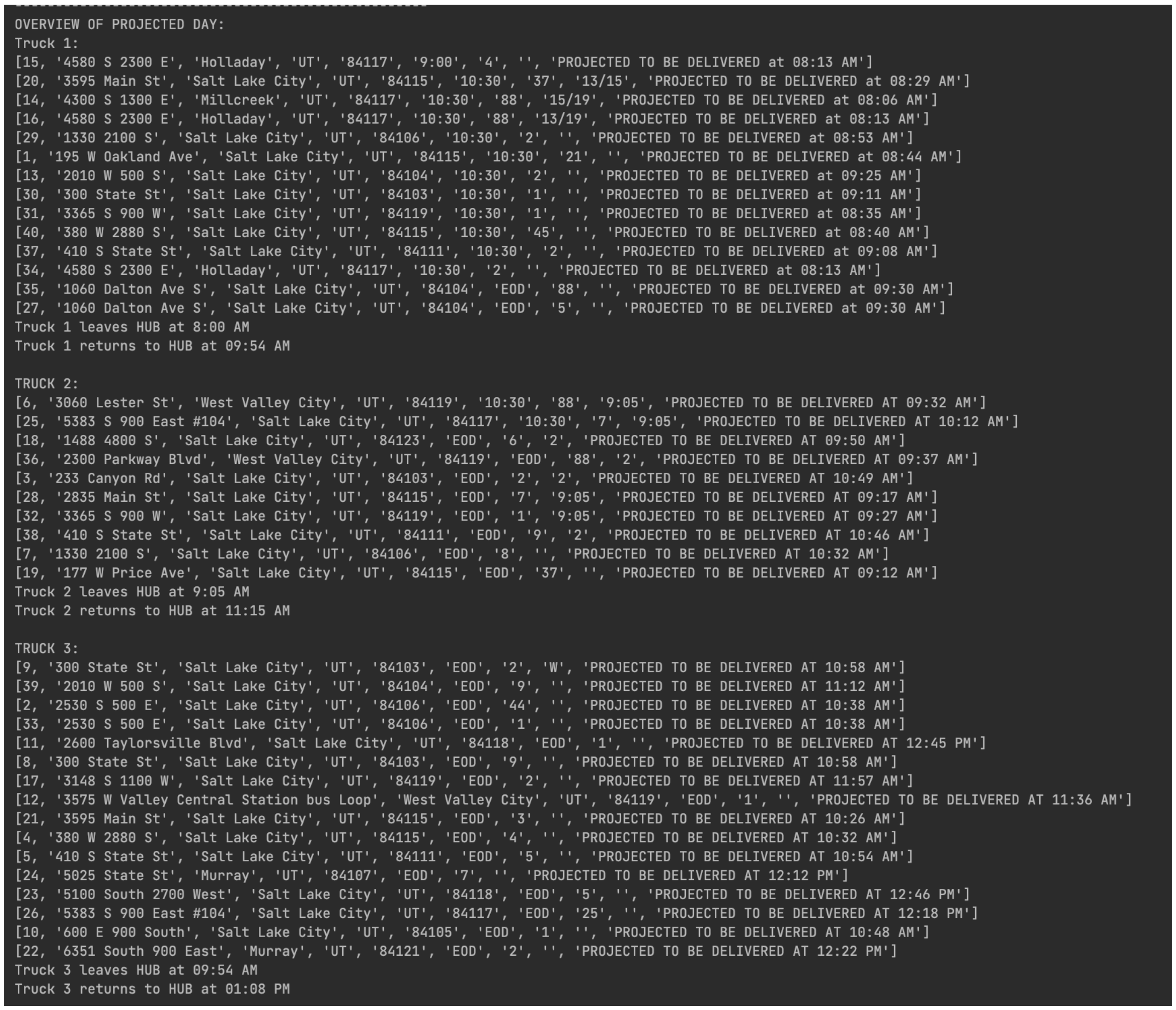
```
ALL PACKAGES STATUS BETWEEN 9:35 AM AND 10:25 AM
-------------------------------------------------------------------------------------------------
[15, '4580 S 2300 E', 'Holladay', 'UT', '84117', '9:00', '4', '', 'DELIVERED AT 08:13 AM']
[20, '3595 Main St', 'Salt Lake City', 'UT', '84115', '10:30', '37', '13/15', 'DELIVERED AT 08:29 AM']
[14, '4300 S 1300 E', 'Millcreek', 'UT', '84117', '10:30', '88', '15/19', 'DELIVERED AT 08:06 AM']
[16, '4580 S 2300 E', 'Holladay', 'UT', '84117', '10:30', '88', '13/19', 'DELIVERED AT 08:13 AM']
[29, '1330 2100 S', 'Salt Lake City', 'UT', '84106', '10:30', '2', '', 'DELIVERED AT 08:53 AM']
[1, '195 W Oakland Ave', 'Salt Lake City', 'UT', '84115', '10:30', '21', '', 'DELIVERED AT 08:44 AM']
[13, '2010 W 500 S', 'Salt Lake City', 'UT', '84104', '10:30', '2', '', 'DELIVERED AT 09:25 AM']
[30, '300 State St', 'Salt Lake City', 'UT', '84103', '10:30', '1', '', 'DELIVERED AT 09:11 AM']
[31, '3365 S 900 W', 'Salt Lake City', 'UT', '84119', '10:30', '1', '', 'DELIVERED AT 08:35 AM']
[40, '380 W 2880 S', 'Salt Lake City', 'UT', '84115', '10:30', '45', '', 'DELIVERED AT 08:40 AM']
[37, '410 S State St', 'Salt Lake City', 'UT', '84111', '10:30', '2', '', 'DELIVERED AT 09:08 AM']
[34, '4580 S 2300 E', 'Holladay', 'UT', '84117', '10:30', '2', '', 'DELIVERED AT 08:13 AM']
[35, '1060 Dalton Ave S', 'Salt Lake City', 'UT', '84104', 'EOD', '88', '', 'DELIVERED AT 09:30 AM']
[27, '1060 Dalton Ave S', 'Salt Lake City', 'UT', '84104', 'EOD', '5', '', 'DELIVERED AT 09:30 AM']
[6, '3060 Lester St', 'West Valley City', 'UT', '84119', '10:30', '88', '9:05', 'DELIVERED AT 09:32 AM']
[25, '5383 S 900 East #104', 'Salt Lake City', 'UT', '84117', '10:30', '7', '9:05', 'DELIVERED AT 10:12 AM']
[18, '1488 4800 S', 'Salt Lake City', 'UT', '84123', 'EOD', '6', '2', 'DELIVERED AT 09:50 AM']
[36, '2300 Parkway Blvd', 'West Valley City', 'UT', '84119', 'EOD', '88', '2', 'DELIVERED AT 09:37 AM']
[3, '233 Canyon Rd', 'Salt Lake City', 'UT', '84103', 'EOD', '2', '2', 'PROJECTED TO BE DELIVERED AT 10:49 AM']
[28, '2835 Main St', 'Salt Lake City', 'UT', '84115', 'EOD', '7', '9:05', 'DELIVERED AT 09:17 AM']
[32, '3365 S 900 W', 'Salt Lake City', 'UT', '84119', 'EOD', '1', '9:05', 'DELIVERED AT 09:27 AM']
[38, '410 S State St', 'Salt Lake City', 'UT', '84111', 'EOD', '9', '2', 'PROJECTED TO BE DELIVERED AT 10:46 AM']
[7, '1330 2100 S', 'Salt Lake City', 'UT', '84106', 'EOD', '8', '', 'PROJECTED TO BE DELIVERED AT 10:32 AM']
[19, '177 W Price Ave', 'Salt Lake City', 'UT', '84115', 'EOD', '37', '', 'DELIVERED AT 09:12 AM']
[9, '300 State St', 'Salt Lake City', 'UT', '84103', 'EOD', '2', 'W', 'PROJECTED TO BE DELIVERED AT 10:58 AM']
[39, '2010 W 500 S', 'Salt Lake City', 'UT', '84104', 'EOD', '9', '', 'PROJECTED TO BE DELIVERED AT 11:12 AM']
[2, '2530 S 500 E', 'Salt Lake City', 'UT', '84106', 'EOD', '44', '', 'PROJECTED TO BE DELIVERED AT 10:38 AM']
[33, '2530 S 500 E', 'Salt Lake City', 'UT', '84106', 'EOD', '1', '', 'PROJECTED TO BE DELIVERED AT 10:38 AM']
[11, '2600 Taylorsville Blvd', 'Salt Lake City', 'UT', '84118', 'EOD', '1', '', 'PROJECTED TO BE DELIVERED AT 12:45 PM']
[8, '300 State St', 'Salt Lake City', 'UT', '84103', 'EOD', '9', '', 'PROJECTED TO BE DELIVERED AT 10:58 AM']
[17, '3148 S 1100 W', 'Salt Lake City', 'UT', '84119', 'EOD', '2', '', 'PROJECTED TO BE DELIVERED AT 11:57 AM']
[12, '3575 W Valley Central Station bus Loop', 'West Valley City', 'UT', '84119', 'EOD', '1', '', 'PROJECTED TO BE DELIVERED AT 11:36 AM']
[21, '3595 Main St', 'Salt Lake City', 'UT', '84115', 'EOD', '3', '', 'PROJECTED TO BE DELIVERED AT 10:26 AM']
[4, '380 W 2880 S', 'Salt Lake City', 'UT', '84115', 'EOD', '4', '', 'PROJECTED TO BE DELIVERED AT 10:32 AM']
[5, '410 S State St', 'Salt Lake City', 'UT', '84111', 'EOD', '5', '', 'PROJECTED TO BE DELIVERED AT 10:54 AM']
[24, '5025 State St', 'Murray', 'UT', '84107', 'EOD', '7', '', 'PROJECTED TO BE DELIVERED AT 12:12 PM']
[23, '5100 South 2700 West', 'Salt Lake City', 'UT', '84118', 'EOD', '5', '', 'PROJECTED TO BE DELIVERED AT 12:46 PM']
[26, '5383 S 900 East #104', 'Salt Lake City', 'UT', '84117', 'EOD', '25', '', 'PROJECTED TO BE DELIVERED AT 12:18 PM']
[10, '600 E 900 South', 'Salt Lake City', 'UT', '84105', 'EOD', '1', '', 'PROJECTED TO BE DELIVERED AT 10:48 AM']
[22, '6351 South 900 East', 'Murray', 'UT', '84121', 'EOD', '2', '', 'PROJECTED TO BE DELIVERED AT 12:22 PM']
Truck 1 left HUB at 8:00 AM and returned at 09:54 AM
 Truck 2 left HUB at 9:05 AM and returns at 11:15 AM
Truck 3 left HUB at 10:20 AM and returns at 01:08 PM
```

G3. Third Status Check

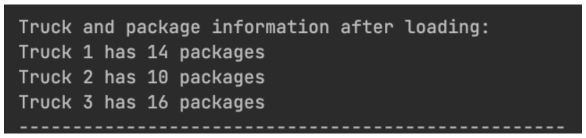


```
ALL PACKAGES STATUS BETWEEN 12:03 PM AND 1:12 PM
----------------------------------------------------------------------------------------------------------
[15, '4580 S 2300 E', 'Holladay', 'UT', '84117', '9:00', '4', '', 'DELIVERED AT 08:13 AM']
[20, '3595 Main St', 'Salt Lake City', 'UT', '84115', '10:30', '37', '13/15', 'DELIVERED AT 08:29 AM']
[14, '4300 S 1300 E', 'Millcreek', 'UT', '84117', '10:30', '88', '15/19', 'DELIVERED AT 08:06 AM']
[16, '4580 S 2300 E', 'Holladay', 'UT', '84117', '10:30', '88', '13/19', 'DELIVERED AT 08:13 AM']
[29, '1330 2100 S', 'Salt Lake City', 'UT', '84106', '10:30', '2', '', 'DELIVERED AT 08:53 AM']
[1, '195 W Oakland Ave', 'Salt Lake City', 'UT', '84115', '10:30', '21', '', 'DELIVERED AT 08:44 AM']
[13, '2010 W 500 S', 'Salt Lake City', 'UT', '84104', '10:30', '2', '', 'DELIVERED AT 09:25 AM']
[30, '300 State St', 'Salt Lake City', 'UT', '84103', '10:30', '1', '', 'DELIVERED AT 09:11 AM']
[31, '3365 S 900 W', 'Salt Lake City', 'UT', '84119', '10:30', '1', '', 'DELIVERED AT 08:35 AM']
[40, '380 W 2880 S', 'Salt Lake City', 'UT', '84115', '10:30', '45', '', 'DELIVERED AT 08:40 AM']
[37, '410 S State St', 'Salt Lake City', 'UT', '84111', '10:30', '2', '', 'DELIVERED AT 09:08 AM']
[34, '4580 S 2300 E', 'Holladay', 'UT', '84117', '10:30', '2', '', 'DELIVERED AT 08:13 AM']
[35, '1060 Dalton Ave S', 'Salt Lake City', 'UT', '84104', 'EOD', '88', '', 'DELIVERED AT 09:30 AM']
[27, '1060 Dalton Ave S', 'Salt Lake City', 'UT', '84104', 'EOD', '5', '', 'DELIVERED AT 09:30 AM']
[6, '3060 Lester St', 'West Valley City', 'UT', '84119', '10:30', '88', '9:05', 'DELIVERED AT 09:32 AM']
[25, '5383 S 900 East #104', 'Salt Lake City', 'UT', '84117', '10:30', '7', '9:05', 'DELIVERED AT 10:12 AM']
[18, '1488 4800 S', 'Salt Lake City', 'UT', '84123', 'EOD', '6', '2', 'DELIVERED AT 09:50 AM']
[36, '2300 Parkway Blvd', 'West Valley City', 'UT', '84119', 'EOD', '88', '2', 'DELIVERED AT 09:37 AM']
[3, '233 Canyon Rd', 'Salt Lake City', 'UT', '84103', 'EOD', '2', '2', 'DELIVERED AT 10:49 AM']
[28, '2835 Main St', 'Salt Lake City', 'UT', '84115', 'EOD', '7', '9:05', 'DELIVERED AT 09:17 AM']
[32, '3365 S 900 W', 'Salt Lake City', 'UT', '84119', 'EOD', '1', '9:05', 'DELIVERED AT 09:27 AM']
[38, '410 S State St', 'Salt Lake City', 'UT', '84111', 'EOD', '9', '2', 'DELIVERED AT 10:46 AM']
[7, '1330 2100 S', 'Salt Lake City', 'UT', '84106', 'EOD', '8', '', 'DELIVERED AT 10:32 AM']
[19, '177 W Price Ave', 'Salt Lake City', 'UT', '84115', 'EOD', '37', '', 'DELIVERED AT 09:12 AM']
[39, '2010 W 500 S', 'Salt Lake City', 'UT', '84104', 'EOD', '9', '', 'DELIVERED AT 11:05 AM']
[2, '2530 S 500 E', 'Salt Lake City', 'UT', '84106', 'EOD', '44', '', 'DELIVERED AT 10:38 AM']
[33, '2530 S 500 E', 'Salt Lake City', 'UT', '84106', 'EOD', '1', '', 'DELIVERED AT 10:38 AM']
[11, '2600 Taylorsville Blvd', 'Salt Lake City', 'UT', '84118', 'EOD', '1', '', 'DELIVERED AT 12:38 PM']
[8, '300 State St', 'Salt Lake City', 'UT', '84103', 'EOD', '9', '', 'PROJECTED TO BE DELIVERED AT 10:58 AM']
[17, '3148 S 1100 W', 'Salt Lake City', 'UT', '84119', 'EOD', '2', '', 'DELIVERED AT 11:50 AM']
[12, '3575 W Valley Central Station bus Loop', 'West Valley City', 'UT', '84119', 'EOD', '1', '', 'DELIVERED AT 11:29 AM']
[21, '3595 Main St', 'Salt Lake City', 'UT', '84115', 'EOD', '3', '', 'DELIVERED AT 10:26 AM']
[4, '380 W 2880 S', 'Salt Lake City', 'UT', '84115', 'EOD', '4', '', 'DELIVERED AT 10:32 AM']
[5, '410 S State St', 'Salt Lake City', 'UT', '84111', 'EOD', '5', '', 'DELIVERED AT 10:54 AM']
[24, '5025 State St', 'Murray', 'UT', '84107', 'EOD', '7', '', 'DELIVERED AT 12:06 PM']
[23, '5100 South 2700 West', 'Salt Lake City', 'UT', '84118', 'EOD', '5', '', 'DELIVERED AT 12:40 PM']
[26, '5383 S 900 East #104', 'Salt Lake City', 'UT', '84117', 'EOD', '25', '', 'DELIVERED AT 12:11 PM']
[10, '600 E 900 South', 'Salt Lake City', 'UT', '84105', 'EOD', '1', '', 'DELIVERED AT 10:48 AM']
[22, '6351 South 900 East', 'Murray', 'UT', '84121', 'EOD', '2', '', 'DELIVERED AT 12:16 PM']
['9', '410 S State St', 'Salt Lake City', 'UT', '84111', 'EOD', '2', 'Updated to new address', 'DELIVERED AT 10:54 AM']
Truck 1 left HUB at 8:00 AM and returned at 09:54 AM
 Truck 2 left HUB at 9:05 AM and returned at 11:15 AM
Truck 3 left HUB at 10:20 AM  and returned at 01:23 PM
```

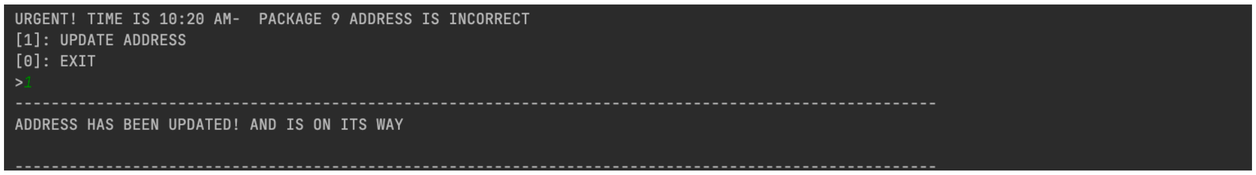
H. Screenshots of Code Execution



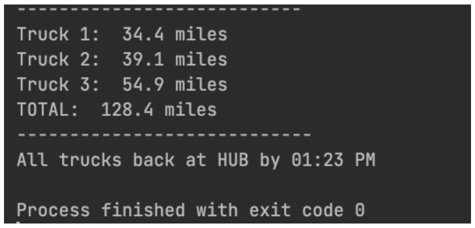
```
------------------------------------------------------------
OVERVIEW OF PROJECTED DAY:
Truck 1:
[15, '4580 S 2300 E', 'Holladay', 'UT', '84117', '9:00', '4', '', 'PROJECTED TO BE DELIVERED at 08:13 AM']
[20, '3595 Main St', 'Salt Lake City', 'UT', '84115', '10:30', '37', '13/15', 'PROJECTED TO BE DELIVERED at 08:29 AM']
[14, '4300 S 1300 E', 'Millcreek', 'UT', '84117', '10:30', '88', '15/19', 'PROJECTED TO BE DELIVERED at 08:06 AM']
[16, '4580 S 2300 E', 'Holladay', 'UT', '84117', '10:30', '88', '13/19', 'PROJECTED TO BE DELIVERED at 08:13 AM']
[29, '1330 2100 S', 'Salt Lake City', 'UT', '84106', '10:30', '2', '', 'PROJECTED TO BE DELIVERED at 08:53 AM']
[1, '195 W Oakland Ave', 'Salt Lake City', 'UT', '84115', '10:30', '21', '', 'PROJECTED TO BE DELIVERED at 08:44 AM']
[13, '2010 W 500 S', 'Salt Lake City', 'UT', '84104', '10:30', '2', '', 'PROJECTED TO BE DELIVERED at 09:25 AM']
[30, '300 State St', 'Salt Lake City', 'UT', '84103', '10:30', '1', '', 'PROJECTED TO BE DELIVERED at 09:11 AM']
[31, '3365 S 900 W', 'Salt Lake City', 'UT', '84119', '10:30', '1', '', 'PROJECTED TO BE DELIVERED at 08:35 AM']
[40, '380 W 2880 S', 'Salt Lake City', 'UT', '84115', '10:30', '45', '', 'PROJECTED TO BE DELIVERED at 08:40 AM']
[37, '410 S State St', 'Salt Lake City', 'UT', '84111', '10:30', '2', '', 'PROJECTED TO BE DELIVERED at 09:08 AM']
[34, '4580 S 2300 E', 'Holladay', 'UT', '84117', '10:30', '2', '', 'PROJECTED TO BE DELIVERED at 08:13 AM']
[35, '1060 Dalton Ave S', 'Salt Lake City', 'UT', '84104', 'EOD', '88', '', 'PROJECTED TO BE DELIVERED at 09:30 AM']
[27, '1060 Dalton Ave S', 'Salt Lake City', 'UT', '84104', 'EOD', '5', '', 'PROJECTED TO BE DELIVERED at 09:30 AM']
Truck 1 leaves HUB at 8:00 AM
Truck 1 returns to HUB at 09:54 AM

TRUCK 2:
[6, '3060 Lester St', 'West Valley City', 'UT', '84119', '10:30', '88', '9:05', 'PROJECTED TO BE DELIVERED AT 09:32 AM']
[25, '5383 S 900 East #104', 'Salt Lake City', 'UT', '84117', '10:30', '7', '9:05', 'PROJECTED TO BE DELIVERED AT 10:12 AM']
[18, '1488 4800 S', 'Salt Lake City', 'UT', '84123', 'EOD', '6', '2', 'PROJECTED TO BE DELIVERED AT 09:50 AM']
[36, '2300 Parkway Blvd', 'West Valley City', 'UT', '84119', 'EOD', '88', '2', 'PROJECTED TO BE DELIVERED AT 09:37 AM']
[3, '233 Canyon Rd', 'Salt Lake City', 'UT', '84103', 'EOD', '2', '2', 'PROJECTED TO BE DELIVERED AT 10:49 AM']
[28, '2835 Main St', 'Salt Lake City', 'UT', '84115', 'EOD', '7', '9:05', 'PROJECTED TO BE DELIVERED AT 09:17 AM']
[32, '3365 S 900 W', 'Salt Lake City', 'UT', '84119', 'EOD', '1', '9:05', 'PROJECTED TO BE DELIVERED AT 09:27 AM']
[38, '410 S State St', 'Salt Lake City', 'UT', '84111', 'EOD', '9', '2', 'PROJECTED TO BE DELIVERED AT 10:46 AM']
[7, '1330 2100 S', 'Salt Lake City', 'UT', '84106', 'EOD', '8', '', 'PROJECTED TO BE DELIVERED AT 10:32 AM']
[19, '177 W Price Ave', 'Salt Lake City', 'UT', '84115', 'EOD', '37', '', 'PROJECTED TO BE DELIVERED AT 09:12 AM']
Truck 2 leaves HUB at 9:05 AM
Truck 2 returns to HUB at 11:15 AM

TRUCK 3:
[9, '300 State St', 'Salt Lake City', 'UT', '84103', 'EOD', '2', 'W', 'PROJECTED TO BE DELIVERED AT 10:58 AM']
[39, '2010 W 500 S', 'Salt Lake City', 'UT', '84104', 'EOD', '9', '', 'PROJECTED TO BE DELIVERED AT 11:12 AM']
[2, '2530 S 500 E', 'Salt Lake City', 'UT', '84106', 'EOD', '44', '', 'PROJECTED TO BE DELIVERED AT 10:38 AM']
[33, '2530 S 500 E', 'Salt Lake City', 'UT', '84106', 'EOD', '1', '', 'PROJECTED TO BE DELIVERED AT 10:38 AM']
[11, '2600 Taylorsville Blvd', 'Salt Lake City', 'UT', '84118', 'EOD', '1', '', 'PROJECTED TO BE DELIVERED AT 12:45 PM']
[8, '300 State St', 'Salt Lake City', 'UT', '84103', 'EOD', '9', '', 'PROJECTED TO BE DELIVERED AT 10:58 AM']
[17, '3148 S 1100 W', 'Salt Lake City', 'UT', '84119', 'EOD', '2', '', 'PROJECTED TO BE DELIVERED AT 11:57 AM']
[12, '3575 W Valley Central Station bus Loop', 'West Valley City', 'UT', '84119', 'EOD', '1', '', 'PROJECTED TO BE DELIVERED AT 11:36 AM']
[21, '3595 Main St', 'Salt Lake City', 'UT', '84115', 'EOD', '3', '', 'PROJECTED TO BE DELIVERED AT 10:26 AM']
[4, '380 W 2880 S', 'Salt Lake City', 'UT', '84115', 'EOD', '4', '', 'PROJECTED TO BE DELIVERED AT 10:32 AM']
[5, '410 S State St', 'Salt Lake City', 'UT', '84111', 'EOD', '5', '', 'PROJECTED TO BE DELIVERED AT 10:54 AM']
[24, '5025 State St', 'Murray', 'UT', '84107', 'EOD', '7', '', 'PROJECTED TO BE DELIVERED AT 12:12 PM']
[23, '5100 South 2700 West', 'Salt Lake City', 'UT', '84118', 'EOD', '5', '', 'PROJECTED TO BE DELIVERED AT 12:46 PM']
[26, '5383 S 900 East #104', 'Salt Lake City', 'UT', '84117', 'EOD', '25', '', 'PROJECTED TO BE DELIVERED AT 12:18 PM']
[10, '600 E 900 South', 'Salt Lake City', 'UT', '84105', 'EOD', '1', '', 'PROJECTED TO BE DELIVERED AT 10:48 AM']
[22, '6351 South 900 East', 'Murray', 'UT', '84121', 'EOD', '2', '', 'PROJECTED TO BE DELIVERED AT 12:22 PM']
Truck 3 leaves HUB at 09:54 AM
Truck 3 returns to HUB at 01:08 PM
```



```
Truck and package information after loading:
Truck 1 has 14 packages
Truck 2 has 10 packages
Truck 3 has 16 packages
--------------------------------------------------
```



```
URGENT! TIME IS 10:20 AM-  PACKAGE 9 ADDRESS IS INCORRECT
[1]: UPDATE ADDRESS
[0]: EXIT
>1
----------------------------------------------------------------------------------------------------
ADDRESS HAS BEEN UPDATED! AND IS ON ITS WAY

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



```
--------------------------
Truck 1:  34.4 miles
Truck 2:  39.1 miles
Truck 3:  54.9 miles
TOTAL:  128.4 miles
---------------------------
All trucks back at HUB by 01:23 PM

Process finished with exit code 0
```

I1. Strengths of Chosen Algorithm

A strength of the chosen algorithm for solving the WGUPS problem is that no matter how many times you run the program, the path will always be the same. The path being the same isn't the case for some other algorithms. Another strength is the accuracy of the algorithm. All truck and package restrictions are met, and the route that was created delivers all packages by their deadline under 140 miles.

I2. Verification of Algorithm

The screen shots in G and H show that the algorithm was successful.

I3. Other possible Algorithms

Plenty of other algorithms could be used to solve the WGUPS problem. One of those algorithms is the self-adjusting heuristic, which is somewhat like the greedy algorithm.

Another algorithm that could solve the WGUPS problem is the dynamic programming approach. This solves problems by breaking it into smaller subproblems.

I3A. Algorithm Differences

The self-adjusting heuristic approach is like the greedy algorithm in the way that it loops through the data and finds the shortest path from its current location to the next. The main difference is that heuristic algorithms don't always give the same result when running the simulation repeatedly, unlike the greedy algorithm where the result is always the same. Unlike the greedy algorithm, the heuristic algorithms are designed to scale as the problems increase in complexity, where the greedy algorithm doesn't scale well.

Using dynamic programming can be beneficial in the fact that because it breaks down into subproblems, there is a chance for a shorter path. That's because the dynamic programming doesn't just look for the shortest path from the current location to the next, it stores all distances from one location to the next. You could then, used these stored paths to see if maybe going to one location, makes for an easier route. This differs from the greedy algorithm because it does not store paths, it just stores one big route.

J. Different Approach

One thing that I would do differently if I were to repeat this project is the sorting of the packages onto the truck. As of now, it only works for these 40 packages and cannot be used for another day. I think the only way to change that would be to change how the packages are entered into the system, which I had no control over. Or, instead of reading the deadline as a string, I could have used the myTime.py to convert it to a time and then sort it that way.

K1A. Efficiency, K1B. Overhead, and K1C. Implications

Mentioned earlier in B6, the hash table is self-adjusting. It will expand its capacity to rehash its contents until reaching the capacity.

The hash table is efficient because all values are referenced by using the key, package ID. This means that all information can quickly be found just by knowing the ID.

The downside to the hash table is if there is not an initial capacity, the table will constantly expand and rehash itself which can be expensive. If there is an initial capacity and more data is added into them, then rehashing still needs to be done, which is also expensive.

Loading more packages and more trucks will have the same effects.

K2. Other Data Structures

One data structure that could have been used to represent package destinations is a stack data structure.

Another data structure that could have been used is the binary search tree.

K2a. Data Structure Differences

Using the stack data structure, the program could easily "pop" successive destinations to the top of the list and calculate the distance between the previous popped location. This would repeat until the list was empty. The difference between this approach and mine, is that the stack is only one direction, meaning there would be less memory used, but it wouldn't have been the best route.

Using the binary search tree (BST) would be beneficial in the fact that I could have pre-sorted the packages based on attribute and quickly access them through a tree.

L. Sources - Works Cited

Lysecky, R., & Vahid, F. (2018, June). *C950: Data Structures and Algorithms II*. zyBooks.

Retrieved March 22, 2021, from

https://learn.zybooks.com/zybook/WGUC950AY20182019/