Erik Cohen
Student #000915169

Core Algorithm Overview

Stated Problem:

The purpose of this project is to create an algorithm using the Python programming language to develop an effective and efficient solution for the traveling salesman problem (TSP) which Wikipedia defines as the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" [1] .

This assignment adds additional requirements of:

- Package delivery deadlines
- Packages which must be on the same truck
- Packages which must be on a specific truck
- Truckload limitation
- Truck speed constraint
- 2 Trucks can be operational at one time.
- One package has the wrong address assigned and must be fixed after 10:20 am

Program Blueprint and Architecture:

Classes:

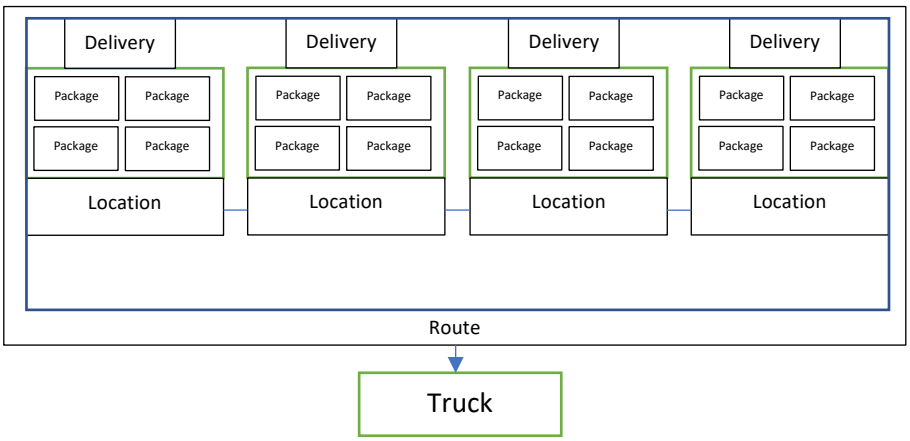
- csv_import: imports all the data from the CSV's provided
- Config: Contains all global variables for the application
- Truck: represents a single truck that follows a route.
- Route: Represents an ordered collection of deliveries and locations
- Delivery: Represents a collection of Packages which are delivered to the same location
- Location: a single point in a Route which has a name, address, city, state, zip.
- Package: The smallest unit to deliver which is imported from the packages.csv file.
- CLI: The Command-line Interface which presents the user with options and reports
- Debug: functions which print out reports for Trucks, Deliveries, Routes, Packages
- Distance: A class that calculates the distances of routes and between two deliveries/locations.
- HashTable: A custom hash map class that is used instead of the python dictionary.

Algorithm Overview:

The proposed solution for this assignment uses a **brute force** method that tried to insert a delivery into each index within the route. Determining the miles which would be added to the route at a particular index. The method will return the route and the index which would cause the route to increase in the least number of miles possible which adhering to the deadlines of the packages within a delivery.

[1] Traveling Salesman Problem. (2020) wikipedia.org
https://en.wikipedia.org/wiki/Travelling_salesman_problem

Relationship between data points stored



Deliveries are composed of Packages grouped by location.
Routes is a list of Deliveries with a sorted list of Locations to follow through the route.
Each Truck is assigned a Route.

The logic of the Applied Algorithm

The **distribute_deliveries_to_trucks** and the **added_distance** function within the route class perform this procedure with the following pseudocode.

distribute_deliveries_to_trucks (Big O(n^2))**:**

> **first assigns any deliveries which must be assigned to truck 2.**
> **Retrieve the remaining deliveries with unassigned trucks.**
> **Define the time during when a particular truck will depart the depot (starting point)**
> **Loop through each Delivery and call the added_distance method for each truck**
> **Calculate the best truck and best route index for a given delivery.**
> **Check the delivery ETA is before the deadline and return the Route index or None.**
> **Assign the delivery to the truck with the least added distance**.

Repeat this loop until all deliveries are assigned to a truck.

Handling Constraints:

1. **Package delivery deadlines**
   The **added_distance** function will check to make sure all packages arrive at its' destination before the deadline when returning a route and index
   If no options are found to meet these criteria None will be returned.
2. **Packages which must be on a specific truck and Packages which must be on the same truck**
   The **distribute_deliveries_to_trucks** method first assigns any deliveries which have packages with assignment constraints
3. **Truck speed constraint**
   The Truck class has a property of **max_speed** which is used to calculate how many miles were traveling during a specified time frame.

4. **Truckload limitation**
   The Truck class has a **will_fit** method which calculates to see if a truck has enough space for delivery before doing any additional assertions and calculations.
5. **Two Trucks at a time.**
   The main program loop will check to see if a truck has completed its' route and it is past 10:20 am.
   It will then dispatch Truck 3 and start its route.
6. **Handling the wrong address constraint:**
   Keep Package 9 unassigned since the address is incorrect, assign it to Truck 3 when the correct address is available (10:20 am)

Adaptability to a changing market
This application has some flexibility in adjusting to more packages up to the point that another truck is needed.

Scalability
If the user were to need additional trucks, another truck could be easily added and dispatched.
CSV Files can be easily loaded with more data.

Efficiency and maintainability of the software.
The application is split into logical files that individually handle each aspect of the structure of the application. Each method is commented to be able to easily follow the lifecycle of the application.
To make changes and make sure the results are as expected the future developer can rely on the Debug Class to output a printed table of all the data structures within the application at any given time using the CLI

Implications of adding additional cities.

Since the HashTable data structure has a complexity of O(1). Adding an additional address in the same or different cities to the data structure would increase the HashTable by one. The added address would contain a nested HashTable with all the distances from the other addresses. The newly added address would have to be added with the distance to all the other addresses in the HashTable.
In total the implications of adding an additional address to the application would be:
(n – 1) [*adding the address to all the existing values*] +
1 [*The newly added Address*] +
(n – 1) [*The nested HashMap of the newly added Address with the distances of all the other keys*] =
2n – 1 [*The Increase in memory by adding one address to the application*]

Increasing the number of trucks would require a refactor of how trucks are handled. Currently Trucks are defined explicitly, each action the truck performs (dispatch, return to base, loading) is called on each truck. This was done to meet the requirement of handling the loading of packages to specific trucks, controlling the time the truck arrives and departs. Any new truck added would have to be called and managed in the same way. To make the handling of trucks more scalable, trucks would have to be defined more implicitly, adding packages, dispatching and returning when business rules are met. A way to achieve this would be to create another module which would be the dispatcher and control the coming and going of all trucks.

Increasing the number of packages would be the simplest as long as no new locations are added. Since the Package class is just an item within the Delivery Class in the application. Having a new Package would just mean the assignment of one more package to a delivery and all other aspects would remain the same.

Other Data Structures

Array – A simple array of arrays could be used. In doing the application would be required to loop through the complete array when needing to access an item. Which would increase the brute force complexity by N

Graph – By using a graph representation of all the locations the memory usage of the application would decrease substantially. As by representing the locations as a node the application could use the edges to define the distance between the two nodes. Doing so would remove the need of having a nested data structure. A Graph implementation will make performing a nearest neighbor algorithm search much simpler since the edges are already defined, and a search would no longer be necessary.

Self-adjusting data structures chosen

The Hash Table Class implements a simple hash key taking into account the sum of the Unicode code point value for each character in the key's string modulo into the hash Table's size[2].
The default value for the size of the HashTable Class is 128 but it can be adjusted.
Magic Methods have been implemented to make interaction with the HashTable class simpler to iterate with and check if it contains a value.

The adapted Hash Table data structure has a complexity of Big O(1) when accessed during run time.

Explanation of Data Structure

The HashTable is used to store a nested HashTable in which the key is a unique location within the application. By being able to access the address HashTable the nested HashTable has all Addresses as keys and distances between the two addresses.
This implementation allows us to quickly retrieve the value of a distance between two addresses and check for the shortest route.

Strengths of the chosen algorithm

1. Having a Brute Force algorithm makes sure all options are taken into account and evaluated to select the most efficient route.
2. Having a simple algorithm such as Brute force for a small sample of packages and possible routes makes the program simpler to test.

Different Approach

[2] Lysecky, R., & Vahid, F. (2018). C950: Data Structures and Algorithms Ii. zyBook.

This program is not scalable to many packages, as is **Dijkstra's[3] algorithms** and **2-opt**
If we were to implement Dijkstra's algorithms we would use a binary heap between the calculated nodes this would provide an improvement as the application doesn't have to run the calculation more than once. This way we can continue to create a weighted graph of the nodes and determine the best route. This approach is more performant and scalable than the brute force algorithm used.

Another approach would be to use the **2-opt** which we can use most of what we currently have by switching into the nearest neighbor comparison. Instead of trying all possible combinations of a route, we can retrieve the nearest address to the current address within the nested HashTable we already have implemented. This approach would not compare all possible combinations but will be much more performant and scalable than the brute force algorithm used.

Another approach I would have taken is changing the architecture of how the Trucks are defined and dispatched. Currently, every truck is explicitly created and defined.
Looping through the array and not calling each truck explicitly would make the app more flexible and scalable for when more packages and trucks are added.

Efficiency
The HashTable has a complexity of O(1) compared to an O(N) complexity that a Queue would have.
By using a HashTable data structure we can quickly address the value of two distances from the nested HashTable and compare all possible outcomes of the route.

Overhead
The current HashTable can be adjusted, but any adjustment would cause all the items internally to rearrange. This would be computationally expensive as the Hashing algorithm would have to run again on each item to decide where it should be placed.

The memory limit would be dependent on the limit of random access memory available on the machine running the program.

The program currently does not use any bandwidth as all the items are stored in memory.

[3] Nilsson, C. (2003). Heuristics for the traveling salesman problem. Linkoping University.

Screenshots of Code Execution: CLI Options presented to the user.

```
-- Menu --
Enter the Letter for the report you would like to generate:
enter a number of minutes you would like to simulate has passed e.g 15, 60 (1 hr), 120 (2 hr).
    P Packages
    D Deliveries
    R Route
    T trucks
    Q return to main menu
enter an option: p
```

First Status Check 9:30 am

```
_____________________________PACKAGES_____________________________
id | Address                                | Truck      | Deadline |  Notes               | Time Delivere
1  | 195 W Oakland Ave                      | 1          | 10:30 AM | N/A                  | 08:22:00
2  | 2530 S 500 E                           | 3          | 17:00 PM | N/A                  | undelivered
3  | 233 Canyon Rd                          | 2          | 17:00 PM | truck 2              | 08:54:20
4  | 380 W 2880 S                           | 1          | 17:00 PM | N/A                  | 08:25:40
5  | 410 S State St                         | 2          | 17:00 PM | N/A                  | undelivered
6  | 3060 Lester St                         | 1          | 10:30 AM | depart 9:05 am       | 08:41:20
7  | 1330 2100 S                            | 1          | 17:00 PM | N/A                  | 08:12:40
8  | 300 State St                           | 1          | 17:00 PM | N/A                  | 09:10:40
9  | 300 State St                           | unassigned | 17:00 PM | Wrong address listed | undelivered
10 | 600 E 900 South                        | 3          | 17:00 PM | N/A                  | undelivered
11 | 2600 Taylorsville Blvd                 | 3          | 17:00 PM | N/A                  | undelivered
12 | 3575 W Valley Central Station bus Loop | 3          | 17:00 PM | N/A                  | undelivered
13 | 2010 W 500 S                           | 1          | 10:30 AM | N/A                  | 08:56:40
14 | 4300 S 1300 E                          | 2          | 10:30 AM | together 15;19       | 08:26:00
15 | 4580 S 2300 E                          | 2          | 09:00 AM | N/A                  | 08:19:20
16 | 4580 S 2300 E                          | 2          | 10:30 AM | together 13;19       | 08:19:20
17 | 3148 S 1100 W                          | 3          | 17:00 PM | N/A                  | undelivered
18 | 1488 4800 S                            | 2          | 17:00 PM | truck 2              | undelivered
19 | 177 W Price Ave                        | 3          | 17:00 PM | N/A                  | undelivered
20 | 3595 Main St                           | 1          | 10:30 AM | together 13;15       | 08:31:00
21 | 3595 Main St                           | 1          | 17:00 PM | N/A                  | 08:31:00
22 | 6351 South 900 East                    | 3          | 17:00 PM | N/A                  | undelivered
23 | 5100 South 2700 West                   | 3          | 17:00 PM | N/A                  | undelivered
24 | 5025 State St                          | 3          | 17:00 PM | N/A                  | undelivered
25 | 5383 South 900 East #104               | 2          | 10:30 AM | depart 9:05 am       | 08:08:00
26 | 5383 South 900 East #104               | 2          | 17:00 PM | N/A                  | 08:08:00
27 | 1060 Dalton Ave S                      | 1          | 17:00 PM | N/A                  | 08:51:20
28 | 2835 Main St                           | 3          | 17:00 PM | depart 9:05 am       | undelivered
29 | 1330 2100 S                            | 1          | 10:30 AM | N/A                  | 08:12:40
30 | 300 State St                           | 1          | 10:30 AM | N/A                  | 09:10:40
31 | 3365 S 900 W                           | 1          | 10:30 AM | N/A                  | 08:36:20
32 | 3365 S 900 W                           | 1          | 17:00 PM | depart 9:05 am       | 08:36:20
33 | 2530 S 500 E                           | 3          | 17:00 PM | N/A                  | undelivered
34 | 4580 S 2300 E                          | 2          | 10:30 AM | N/A                  | 08:19:20
35 | 1060 Dalton Ave S                      | 1          | 17:00 PM | N/A                  | 08:51:20
36 | 2300 Parkway Blvd                      | 2          | 17:00 PM | truck 2              | undelivered
37 | 410 S State St                         | 2          | 10:30 AM | N/A                  | undelivered
38 | 410 S State St                         | 2          | 17:00 PM | truck 2              | undelivered
39 | 2010 W 500 S                           | 1          | 17:00 PM | N/A                  | 08:56:40
40 | 380 W 2880 S                           | 1          | 10:30 AM | N/A                  | 08:25:40
```

Second Status Check 10:20 am

_____PACKAGES_____

id	Address	Truck	Deadline	Notes	Time Delivered
1	195 W Oakland Ave	1	10:30 AM	N/A	08:22:00
2	2530 S 500 E	3	17:00 PM	N/A	undelivered
3	233 Canyon Rd	2	17:00 PM	truck 2	08:54:20
4	380 W 2880 S	1	17:00 PM	N/A	08:25:40
5	410 S State St	2	17:00 PM	N/A	undelivered
6	3060 Lester St	1	10:30 AM	depart 9:05 am	08:41:20
7	1330 2100 S	1	17:00 PM	N/A	08:12:40
8	300 State St	1	17:00 PM	N/A	09:10:40
9	300 State St	unassigned	17:00 PM	Wrong address listed	undelivered
10	600 E 900 South	3	17:00 PM	N/A	undelivered
11	2600 Taylorsville Blvd	3	17:00 PM	N/A	undelivered
12	3575 W Valley Central Station bus Loop	3	17:00 PM	N/A	undelivered
13	2010 W 500 S	1	10:30 AM	N/A	08:56:40
14	4300 S 1300 E	2	10:30 AM	together 15;19	08:26:00
15	4580 S 2300 E	2	09:00 AM	N/A	08:19:20
16	4580 S 2300 E	2	10:30 AM	together 13;19	08:19:20
17	3148 S 1100 W	3	17:00 PM	N/A	undelivered
18	1488 4800 S	2	17:00 PM	truck 2	09:31:20
19	177 W Price Ave	3	17:00 PM	N/A	undelivered
20	3595 Main St	1	10:30 AM	together 13;15	08:31:00
21	3595 Main St	1	17:00 PM	N/A	08:31:00
22	6351 South 900 East	3	17:00 PM	N/A	undelivered
23	5100 South 2700 West	3	17:00 PM	N/A	undelivered
24	5025 State St	3	17:00 PM	N/A	undelivered
25	5383 South 900 East #104	2	10:30 AM	depart 9:05 am	08:08:00
26	5383 South 900 East #104	2	17:00 PM	N/A	08:08:00
27	1060 Dalton Ave S	1	17:00 PM	N/A	08:51:20
28	2835 Main St	3	17:00 PM	depart 9:05 am	undelivered
29	1330 2100 S	1	10:30 AM	N/A	08:12:40
30	300 State St	1	10:30 AM	N/A	09:10:40
31	3365 S 900 W	1	10:30 AM	N/A	08:36:20
32	3365 S 900 W	1	17:00 PM	depart 9:05 am	08:36:20
33	2530 S 500 E	3	17:00 PM	N/A	undelivered
34	4580 S 2300 E	2	10:30 AM	N/A	08:19:20
35	1060 Dalton Ave S	1	17:00 PM	N/A	08:51:20
36	2300 Parkway Blvd	2	17:00 PM	truck 2	09:44:40
37	410 S State St	2	10:30 AM	N/A	undelivered
38	410 S State St	2	17:00 PM	truck 2	undelivered
39	2010 W 500 S	1	17:00 PM	N/A	08:56:40
40	380 W 2880 S	1	10:30 AM	N/A	08:25:40

The third Status Check 1:00 pm

PACKAGES

id	Address	Truck	Deadline	Notes	Time Delivered
1	195 W Oakland Ave	1	10:30 AM	N/A	08:22:00
2	2530 S 500 E	3	17:00 PM	N/A	undelivered
3	233 Canyon Rd	2	17:00 PM	truck 2	08:54:20
4	380 W 2880 S	1	17:00 PM	N/A	08:25:40
5	410 S State St	2	17:00 PM	N/A	10:07:00
6	3060 Lester St	1	10:30 AM	depart 9:05 am	08:41:20
7	1330 2100 S	1	17:00 PM	N/A	08:12:40
8	300 State St	1	17:00 PM	N/A	09:10:40
9	410 S State St	3	17:00 PM	Wrong address listed	10:50:20
10	600 E 900 South	3	17:00 PM	N/A	undelivered
11	2600 Taylorsville Blvd	3	17:00 PM	N/A	undelivered
12	3575 W Valley Central Station bus Loop	3	17:00 PM	N/A	11:57:20
13	2010 W 500 S	1	10:30 AM	N/A	08:56:40
14	4300 S 1300 E	2	10:30 AM	together 15;19	08:26:00
15	4580 S 2300 E	2	09:00 AM	N/A	08:19:20
16	4580 S 2300 E	2	10:30 AM	together 13;19	08:19:20
17	3148 S 1100 W	3	17:00 PM	N/A	undelivered
18	1488 4800 S	2	17:00 PM	truck 2	09:31:20
19	177 W Price Ave	3	17:00 PM	N/A	11:52:40
20	3595 Main St	1	10:30 AM	together 13;15	08:31:00
21	3595 Main St	1	17:00 PM	N/A	08:31:00
22	6351 South 900 East	3	17:00 PM	N/A	11:34:00
23	5100 South 2700 West	3	17:00 PM	N/A	undelivered
24	5025 State St	3	17:00 PM	N/A	11:44:20
25	5383 South 900 East #104	2	10:30 AM	depart 9:05 am	08:08:00
26	5383 South 900 East #104	2	17:00 PM	N/A	08:08:00
27	1060 Dalton Ave S	1	17:00 PM	N/A	08:51:20
28	2835 Main St	3	17:00 PM	depart 9:05 am	undelivered
29	1330 2100 S	1	10:30 AM	N/A	08:12:40
30	300 State St	1	10:30 AM	N/A	09:10:40
31	3365 S 900 W	1	10:30 AM	N/A	08:36:20
32	3365 S 900 W	1	17:00 PM	depart 9:05 am	08:36:20
33	2530 S 500 E	3	17:00 PM	N/A	undelivered
34	4580 S 2300 E	2	10:30 AM	N/A	08:19:20
35	1060 Dalton Ave S	1	17:00 PM	N/A	08:51:20
36	2300 Parkway Blvd	2	17:00 PM	truck 2	09:44:40
37	410 S State St	2	10:30 AM	N/A	10:07:00
38	410 S State St	2	17:00 PM	truck 2	10:07:00
39	2010 W 500 S	1	17:00 PM	N/A	08:56:40
40	380 W 2880 S	1	10:30 AM	N/A	08:25:40

Delivery Chart of which truck each delivery was assigned

DELIVERIES

id	Address	Truck	packages	deadline	ETA
1	1060 Dalton Ave S	1	27,35	17:00:00	08:51:20
2	1330 2100 S	1	7,29	10:30:00	08:12:40
3	1488 4800 S	2	18	17:00:00	09:31:20
4	177 W Price Ave	3	19	17:00:00	11:52:40
5	195 W Oakland Ave	1	1	10:30:00	08:22:00
6	2010 W 500 S	1	13,39	10:30:00	08:56:40
7	2300 Parkway Blvd	2	36	17:00:00	09:44:40
8	233 Canyon Rd	2	3	17:00:00	08:54:20
9	2530 S 500 E	3	2,33	17:00:00	12:28:40
10	2600 Taylorsville Blvd	3	11	17:00:00	12:58:20
11	2835 Main St	3	28	17:00:00	12:32:20
12	300 State St	1	8,30	10:30:00	09:10:40
13	3060 Lester St	1	6	10:30:00	08:41:20
14	3148 S 1100 W	3	17	17:00:00	12:41:00
15	3365 S 900 W	1	31,32	10:30:00	08:36:20
16	3575 W Valley Central Station bus Loop	3	12	17:00:00	11:57:20
17	3595 Main St	1	20,21	10:30:00	08:31:00
18	380 W 2880 S	1	4,40	10:30:00	08:25:40
19	410 S State St	2	5,37,38	10:30:00	10:07:00
20	4300 S 1300 E	2	14	10:30:00	08:26:00
21	4580 S 2300 E	2	15,16,34	09:00:00	08:19:20
22	5025 State St	3	24	17:00:00	11:44:20
23	5100 South 2700 West	3	23	17:00:00	12:57:00
24	5383 South 900 East #104	2	25,26	10:30:00	08:08:00
25	600 E 900 South	3	10	17:00:00	12:18:00
26	6351 South 900 East	3	22	17:00:00	11:34:00
27	410 S State St	3	9	17:00:00	10:50:20

Route table listing the locations each specified truck will go through throughout the day.

Routes

```
Route assigned to truck 1
 |- 1330 2100 S
 |- 195 W Oakland Ave
 |- 380 W 2880 S
 |- 3595 Main St
 |- 3365 S 900 W
 |- 3060 Lester St
 |- 1060 Dalton Ave S
 |- 2010 W 500 S
 |- 300 State St
Route assigned to truck 2
 |- 5383 South 900 East #104
 |- 4580 S 2300 E
 |- 4300 S 1300 E
 |- 233 Canyon Rd
 |- 1488 4800 S
 |- 2300 Parkway Blvd
 |- 410 S State St
Route assigned to truck 3
 |- 410 S State St
 |- 6351 South 900 East
 |- 5025 State St
 |- 177 W Price Ave
 |- 3575 W Valley Central Station bus Loop
 |- 600 E 900 South
 |- 2530 S 500 E
 |- 2835 Main St
 |- 3148 S 1100 W
 |- 5100 South 2700 West
 |- 2600 Taylorsville Blvd
```

The truck table notifying the current location of the truck packages onboard and the miles it will travel during the assigned route.

```
_____________________________________________________________________________
_____________________________Trucks__________________________________________
id | Current Location                    | Deliveries | Packages | Miles|
1  | 4001 South 700 East                 | 9          | 16       | 21.2 |
2  | 4001 South 700 East                 | 7          | 12       | 44.6 |
3  | 4001 South 700 East                 | 11         | 12       | 44.9 |
_____________________________________________________Total Miles: 110.7
_____________________________________________________________________________
```