

## Data Structures and Algorithms II Report

### Strengths of the Chosen Algorithm

---

My chosen algorithm for implementing this program is the nearest neighbor algorithm, which uses a greedy approach.

The strengths identified in this algorithm are routing efficiency, self-adjusting nature, and Ease of Scaling/implementation.

- **Routing Efficiency**

The algorithm always chooses the closest delivery address from the current location. It takes a simple approach as it does not need to know or evaluate every possible route beforehand. It is only concerned with the current address and the following address. This helps minimize travel distance and truck mileage. In my program, this results in faster routing decisions and avoids unnecessary calculations.

- **Self-Adjusting**

The nearest neighbor algorithm is self-adjusting as it recalculates the next best step based on the current location. In my implementation, the algorithm updates the truck's location after each delivery, adds the mileage, and then finds the next closest address from that new point. It does this until all packages are delivered.

- **Ease of Scaling/Implementation**

The simple step-by-step logic makes it easy to implement and scale. It can easily handle a larger number of packages without significant changes.

## Verification of Algorithm

---

The nearest neighbor implementation I used for my solutions meets all the requirements as follows:

- It delivers all 40 packages based on their delivery deadline while meeting their specific requirements
  - For example, my solution updates Package 9's incorrect address at 10:20 am. This is shown in the queried results screenshots in part D. Before 10:20, package 9 displays the wrong address, and any queries at or after 10:20 am show the corrected address.
- The algorithm determines an efficient route, as shown by the total mileage traveled by all trucks, which is 122.10 miles.
- The algorithm starts at each truck's assigned start time, which is determined in the `run_delivery()` method to account for truck loading, delayed packages, and driver constraints. This ensures that routing logic operates within the requirements.

## Other Possible Algorithms

---

Two other algorithms that would meet all the requirements of this task are:

- Brute Force Algorithm
- Branch and Bound Algorithm

Their comparison with the next neighbor is detailed on the following page.

## Algorithm Differences

---

- **Brute Force Algorithm vs. Nearest Neighbor Algorithm**

The brute force algorithm would calculate and compare the total distance for all possible combinations of the addresses to be visited and return the shortest total distance(Singh, 2024). While brute force would give the guaranteed shortest distance, it also increases time complexity. This differs from the nearest neighbor algorithm, which only looks for the closest address at each step and does not explore all possible routes. This makes the nearest neighbor faster and ideal for real-world scenarios, like the WGUPS package delivery system, whereas the large set of addresses would be difficult to manage with brute force.

- **Branch and Bound Algorithm vs. Nearest Neighbor Algorithm**

This algorithm also explores different possible combinations of addresses to find the shortest total distance. However, it abandons branches that would lead to worse results. It does this by calculating the total distance of one complete delivery route and saving that value. As it explores other possible routes, it compares the partial distance of each branch to the saved value. The algorithm stops exploring that branch if the partial path exceeds the saved best distance. It eventually returns the best total distance. (Kuo, 2024)

While this algorithm would give the best total distance, it is different from the nearest neighbor, as the nearest neighbor doesn't explore partial paths of all possible combinations and does not backtrack once a decision is made. The branch and bound is not ideal for large data sets like the WGUPS package delivery system with deadline constraints, as it would be challenging to meet the constraints and find the shortest total distance.

Brute force and Branch and Bound algorithms are guaranteed to find the optimal result(shortest distance). However, both algorithms are inefficient and have a time complexity of  $O(n!)$ , which is the worst efficiency in Big O. In comparison, the nearest neighbor has a time complexity of  $O(n^2)$ , making it significantly more efficient than the other two's  $O(n!)$  complexity.

## Different Approach

---

If I were to do this project again, I would make the following modifications:

- **Improve how trucks are loaded:**

My current approach uses a `run_delivery()` method to manually load trucks to meet each package's specific requirements. An improvement would be to write a method that automatically parses through each delivery constraint, such as deadlines, truck specifications, delayed packages, package 9 address correction, and uses a priority measure to determine the loading order for the available trucks.

- **Use a different collision handling strategy for the hash table.**

I would also consider using open addressing with linear probing to handle collisions. My current implementation uses separate chaining, which stores key-value pairs in a list at each bucket. With linear probing, each bucket would store a single key-value pair, and if two keys map to the same index, instead of adding it to a list, it searches for the next available slot in the hash table. However, this would require additional logic to resize and rehash the table when it becomes too full.

## Verification of Data Structure

---

The data structure used in this project is a custom hash table. This is implemented with separate chaining for collision handling. This hash table meets the requirements for the WGUPS Package delivery service.

- The hash table is sized to hold 59 buckets, thus giving it the general recommended load factor of less than 0.7. This ensures that it handles all 40 packages and minimizes collisions that would lead to lower efficiency.
- The hash table allows  $O(1)$  time insertion in the best and average cases and  $O(N)$  time insertion in the worst case. This is done by the `insert()` function, which takes the package ID as key and the package object as value to store in the hash table. To handle collisions, this insert function checks if a bucket is empty and adds a list for separate chaining. Afterwards, it appends the key-value pair to that list, making the packages accessible for the nearest package algorithm and other helper methods.

- The hash table allows lookup using the `lookup()` function, which takes the package ID as key and uses that to calculate the bucket index using the custom hash function in  $O(1)$  time, since each bucket holds a list for collision handling, it then loops through each item in the list, and returns the package object that matches the key value passed. This lookup function is essential as it is used throughout the program to access the hash table and retrieve and update package objects.

## Other Data Structures

---

Two other data structures that would meet the requirement of this task are:

- Linked Lists
- Binary Search Tree

## Data Structure Differences

---

A linked list implementation would hold package objects at each node and connect them by pointers. Searching for and inserting package objects would require looping through the list until the correct package is found, which results in  $O(n)$  time complexity. Compared to a hash table, it does not allow fast lookups by key and fast insertion. While a linked list would meet all the requirements to implement the routing algorithm, it would not be as efficient as a hash table.

A binary search tree organizes data in sorted order by key, which allows for faster searching and insertion than a linked list if the tree is balanced, with an average time complexity of  $O(\log n)$ . However, balancing the tree would require more logic and setup, which is harder to maintain, especially if future implementation requires packages to be deleted. In this case, it would need a separate balancing logic, and if this is not handled correctly, it could result in slower lookups. Even when balanced, BST is generally slower than a hash table.

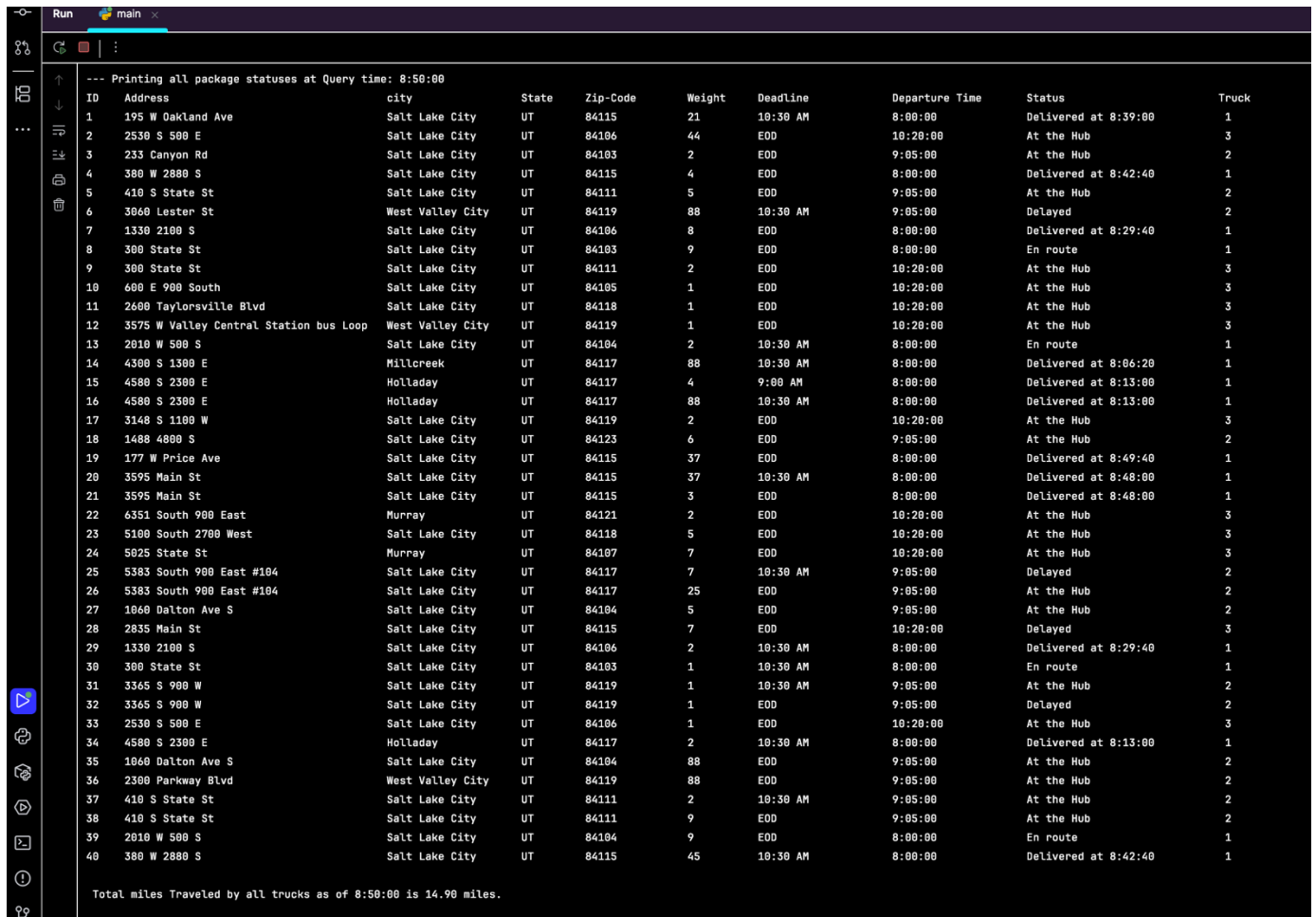
## Status Checks:

Three status checks were done to display the delivery status of packages at 3 separate times.

### First Status Check

Query Time: 08:50 am

The first checks shows that that the constraint of the delayed packages is met. The 4 delayed packages[6,25,28,32] all show as delayed. This is because the package will not be available to leave the HUB until 9:05. Package 9's delivery address also shows the old address[300 S State St] and not the address that will be updated at 10:20.



ID	Address	city	State	Zip-Code	Weight	Deadline	Departure Time	Status	Truck
1	195 W Oakland Ave	Salt Lake City	UT	84115	21	10:30 AM	8:00:00	Delivered at 8:39:00	1
2	2530 S 500 E	Salt Lake City	UT	84106	44	EOD	10:20:00	At the Hub	3
3	233 Canyon Rd	Salt Lake City	UT	84103	2	EOD	9:05:00	At the Hub	2
4	380 W 2880 S	Salt Lake City	UT	84115	4	EOD	8:00:00	Delivered at 8:42:40	1
5	410 S State St	Salt Lake City	UT	84111	5	EOD	9:05:00	At the Hub	2
6	3060 Lester St	West Valley City	UT	84119	88	10:30 AM	9:05:00	Delayed	2
7	1330 2100 S	Salt Lake City	UT	84106	8	EOD	8:00:00	Delivered at 8:29:40	1
8	300 State St	Salt Lake City	UT	84103	9	EOD	8:00:00	En route	1
9	300 State St	Salt Lake City	UT	84111	2	EOD	10:20:00	At the Hub	3
10	600 E 900 South	Salt Lake City	UT	84105	1	EOD	10:20:00	At the Hub	3
11	2600 Taylorsville Blvd	Salt Lake City	UT	84118	1	EOD	10:20:00	At the Hub	3
12	3575 W Valley Central Station bus Loop	West Valley City	UT	84119	1	EOD	10:20:00	At the Hub	3
13	2010 W 500 S	Salt Lake City	UT	84104	2	10:30 AM	8:00:00	En route	1
14	4300 S 1300 E	Millcreek	UT	84117	88	10:30 AM	8:00:00	Delivered at 8:06:20	1
15	4580 S 2300 E	Holladay	UT	84117	4	9:00 AM	8:00:00	Delivered at 8:13:00	1
16	4580 S 2300 E	Holladay	UT	84117	88	10:30 AM	8:00:00	Delivered at 8:13:00	1
17	3140 S 1100 W	Salt Lake City	UT	84119	2	EOD	10:20:00	At the Hub	3
18	1480 4800 S	Salt Lake City	UT	84123	6	EOD	9:05:00	At the Hub	2
19	177 W Price Ave	Salt Lake City	UT	84115	37	EOD	8:00:00	Delivered at 8:49:40	1
20	3595 Main St	Salt Lake City	UT	84115	37	10:30 AM	8:00:00	Delivered at 8:48:00	1
21	3595 Main St	Salt Lake City	UT	84115	3	EOD	8:00:00	Delivered at 8:48:00	1
22	6351 South 900 East	Murray	UT	84121	2	EOD	10:20:00	At the Hub	3
23	5100 South 2700 West	Salt Lake City	UT	84118	5	EOD	10:20:00	At the Hub	3
24	5025 State St	Murray	UT	84107	7	EOD	10:20:00	At the Hub	3
25	5383 South 900 East #104	Salt Lake City	UT	84117	7	10:30 AM	9:05:00	Delayed	2
26	5383 South 900 East #104	Salt Lake City	UT	84117	25	EOD	9:05:00	At the Hub	2
27	1060 Dalton Ave S	Salt Lake City	UT	84104	5	EOD	9:05:00	At the Hub	2
28	2835 Main St	Salt Lake City	UT	84115	7	EOD	10:20:00	Delayed	3
29	1330 2100 S	Salt Lake City	UT	84106	2	10:30 AM	8:00:00	Delivered at 8:29:40	1
30	300 State St	Salt Lake City	UT	84103	1	10:30 AM	8:00:00	En route	1
31	3365 S 900 W	Salt Lake City	UT	84119	1	10:30 AM	9:05:00	At the Hub	2
32	3365 S 900 W	Salt Lake City	UT	84119	1	EOD	9:05:00	Delayed	2
33	2530 S 500 E	Salt Lake City	UT	84106	1	EOD	10:20:00	At the Hub	3
34	4580 S 2300 E	Holladay	UT	84117	2	10:30 AM	8:00:00	Delivered at 8:13:00	1
35	1060 Dalton Ave S	Salt Lake City	UT	84104	88	EOD	9:05:00	At the Hub	2
36	2300 Parkway Blvd	West Valley City	UT	84119	88	EOD	9:05:00	At the Hub	2
37	410 S State St	Salt Lake City	UT	84111	2	10:30 AM	9:05:00	At the Hub	2
38	410 S State St	Salt Lake City	UT	84111	9	EOD	9:05:00	At the Hub	2
39	2010 W 500 S	Salt Lake City	UT	84104	9	EOD	8:00:00	En route	1
40	380 W 2880 S	Salt Lake City	UT	84115	45	10:30 AM	8:00:00	Delivered at 8:42:40	1

Total miles Traveled by all trucks as of 8:50:00 is 14.90 miles.

## Second Status Check

Query Time: 10:10 am

The second status check done at 10:10 shows package 9's delivery address still hasn't been updated.

--- Printing all package statuses at Query time: 10:10:00									
ID	Address	city	State	Zip-Code	Weight	DeadLine	Departure Time	Status	Truck
1	195 W Oakland Ave	Salt Lake City	UT	84115	21	10:30 AM	8:00:00	Delivered at 8:39:00	1
2	2530 S 500 E	Salt Lake City	UT	84106	44	EOD	10:20:00	At the Hub	3
3	233 Canyon Rd	Salt Lake City	UT	84103	2	EOD	9:05:00	Delivered at 10:00:20	2
4	380 W 2880 S	Salt Lake City	UT	84115	4	EOD	8:00:00	Delivered at 8:42:40	1
5	410 S State St	Salt Lake City	UT	84111	5	EOD	9:05:00	En route	2
6	3060 Lester St	West Valley City	UT	84119	88	10:30 AM	9:05:00	Delivered at 9:37:40	2
7	1330 2100 S	Salt Lake City	UT	84106	8	EOD	8:00:00	Delivered at 8:29:40	1
8	300 State St	Salt Lake City	UT	84103	9	EOD	8:00:00	Delivered at 9:14:40	1
9	300 State St	Salt Lake City	UT	84111	2	EOD	10:20:00	At the Hub	3
10	600 E 900 South	Salt Lake City	UT	84105	1	EOD	10:20:00	At the Hub	3
11	2600 Taylorsville Blvd	Salt Lake City	UT	84118	1	EOD	10:20:00	At the Hub	3
12	3575 W Valley Central Station bus Loop	West Valley City	UT	84119	1	EOD	10:20:00	At the Hub	3
13	2010 W 500 S	Salt Lake City	UT	84104	2	10:30 AM	8:00:00	Delivered at 9:28:40	1
14	4300 S 1300 E	Millcreek	UT	84117	88	10:30 AM	8:00:00	Delivered at 8:06:20	1
15	4580 S 2300 E	Holladay	UT	84117	4	9:00 AM	8:00:00	Delivered at 8:13:00	1
16	4580 S 2300 E	Holladay	UT	84117	88	10:30 AM	8:00:00	Delivered at 8:13:00	1
17	3140 S 1100 W	Salt Lake City	UT	84119	2	EOD	10:20:00	At the Hub	3
18	1488 4800 S	Salt Lake City	UT	84123	6	EOD	9:05:00	En route	2
19	177 W Price Ave	Salt Lake City	UT	84115	37	EOD	8:00:00	Delivered at 8:49:40	1
20	3595 Main St	Salt Lake City	UT	84115	37	10:30 AM	8:00:00	Delivered at 8:48:00	1
21	3595 Main St	Salt Lake City	UT	84115	3	EOD	8:00:00	Delivered at 8:48:00	1
22	6351 South 900 East	Murray	UT	84121	2	EOD	10:20:00	At the Hub	3
23	5100 South 2700 West	Salt Lake City	UT	84118	5	EOD	10:20:00	At the Hub	3
24	5025 State St	Murray	UT	84107	7	EOD	10:20:00	At the Hub	3
25	5383 South 900 East #104	Salt Lake City	UT	84117	7	10:30 AM	9:05:00	Delivered at 9:13:00	2
26	5383 South 900 East #104	Salt Lake City	UT	84117	25	EOD	9:05:00	Delivered at 9:13:00	2
27	1060 Dalton Ave S	Salt Lake City	UT	84104	5	EOD	9:05:00	Delivered at 9:52:20	2
28	2035 Main St	Salt Lake City	UT	84115	7	EOD	10:20:00	Delayed	3
29	1330 2100 S	Salt Lake City	UT	84106	2	10:30 AM	8:00:00	Delivered at 8:29:40	1
30	300 State St	Salt Lake City	UT	84103	1	10:30 AM	8:00:00	Delivered at 9:14:40	1
31	3365 S 900 W	Salt Lake City	UT	84119	1	10:30 AM	9:05:00	Delivered at 9:32:40	2
32	3365 S 900 W	Salt Lake City	UT	84119	1	EOD	9:05:00	Delivered at 9:32:40	2
33	2530 S 500 E	Salt Lake City	UT	84106	1	EOD	10:20:00	At the Hub	3
34	4580 S 2300 E	Holladay	UT	84117	2	10:30 AM	8:00:00	Delivered at 8:13:00	1
35	1060 Dalton Ave S	Salt Lake City	UT	84104	88	EOD	9:05:00	Delivered at 9:52:20	2
36	2300 Parkway Blvd	West Valley City	UT	84119	88	EOD	9:05:00	Delivered at 9:43:00	2
37	410 S State St	Salt Lake City	UT	84111	2	10:30 AM	9:05:00	En route	2
38	410 S State St	Salt Lake City	UT	84111	9	EOD	9:05:00	En route	2
39	2010 W 500 S	Salt Lake City	UT	84104	9	EOD	8:00:00	Delivered at 9:28:40	1
40	380 W 2880 S	Salt Lake City	UT	84115	45	10:30 AM	8:00:00	Delivered at 8:42:40	1
Total miles Traveled by all trucks as of 10:10:00 is 56.50 miles.									

## Third Status Check

Query Time: 12:55 pm

The third status check shows package 9's address has been updated and all packages have successfully been delivered while meeting all delivery constraints.



## Truck Summary Report

---

Here is a screenshot of the truck summary report. It shows each truck's start time, when it returned to the hub, mileage, and the total mileage traveled by all trucks.

```
=====Simulation Complete=====
All packages delivered. You can now query status reports
Note: Please use 24-hour format for time entries E.g 10:20, 13:50, 20:00

Please choose a report to view: Enter numbers only

1. View status report for all packages
2. View status report for a specific package at a specific time
3. View Truck Summary including total mileage
4. Enter 4 to exit the system

Enter your choice:

3

=====Truck Summary Report=====

Truck 1 started at: 8:00:00 and returned to hub at : 10:05:00
Truck 1 Miles: 37.50

Truck 2 started at: 9:05:00 and returned to hub at : 11:23:40
Truck 2 Miles: 41.60

Truck 3 started at: 10:20:00 and returned to hub at : 12:43:20
Truck 3 Miles: 43.00

Total miles traveled by all trucks is 122.10
```

## Sources

---

Barrett, S. (n.d.). *Python data structures & algorithms + LEETCODE exercises*. Udemy

<https://www.udemy.com/course/data-structures-algorithms-python>

Bhagavan. (2023, March 26). *Solving the traveling salesman problem in Python using the nearest neighbor algorithm*. Medium

<https://medium.com/@suryabhagavanchakkapalli/solving-the-traveling-salesman-problem-in-python-using-the-nearest-neighbor-algorithm-48fcf8db289a>

Kuo, M. (2024, April 18). *Algorithms for the travelling salesman problem*. Routific

<https://www.routific.com/blog/travelling-salesman-problem>

Singh, R. (2024, March 15). *Best algorithms for the traveling salesman problem*. NextBillion.ai

<https://nextbillion.ai/post/algorithms-for-the-traveling-salesman-problem>