C950 WGUPS Algorithm Overview

Paul Overfelt III
ID #009278142
WGU Email: poverfe@wgu.edu
Date 8/19/2022

C950 Data Structures and Algorithms II

A. Algorithm Identification

The algorithm used within the project is a Greedy Algorithm, which just looks at all the packages left on a truck and delivers the next closest one. This is done in 2 stages. The packages are loaded on the truck manually, then we loop through the tucks package list to find the next closest delivery. After every delivery a running distance total for the truck is calculated and the time is tracked for when each package is delivered.

B1. Logic Comments

Create Empty HashMap

Read Package Data objects into HashMap

Read Distance table

Read Address List

Create a Address List Dictionary(Address:Address ID)

Create truck objects

Load Truck 1

While packages are still on truck deliver package method

if the truck is empty return to hub method and set truck 3 departing time to the same as truck 1 arival time

Load Truck 2

While packages are still on truck deliver package method

if truck is empty return to hub method

Load Truck 3

While packages are still on truck deliver packages

if truck is empty return to hub method

Print Delivery Times and Distances

Main Menu to search for package

1 Look up single package, ask for time and package id

2 Show all packages, ask for time

3 Exit

Deliver_package(truck)

Call find next shortest delivery to find next package to deliver

Updates locations of truck, distance driven, time of package delivery

find_next_shortest_delivery(truck)

Loops through package list to find the shortest distance in the packages list on the passed in truck object and returns the package object

B2. Development Environment

This program was built on Pop_OS! 22.04 in Pycharm 2022.2.1 on python 3.9. Computer is a Dell XPS 15 7590 i7-9750H and 32GB of ram

B3. Space-Time and Big-O

Name	Time-Complexity
Main.py	O(n)
read_package_data	O(n)
read_distance_data	O(1)
read_address_data	O(1)
distance_between	O(1)
find_next_shortest_delivery	O(n)
create_address_dic	O(n)
load_truck	O(n)
deliver_package	O(1)
return_to_hub	O(1)
Package.py	O(n)
package_delivery_status_at_time	O(n)
Truck.py	O(n)
load_package	O(1)
package_delivered	O(1)
hash_table.py	O(n^2)
_get_hash	O(n)
Add	O(n)

Get	O(n)
Delete	O(n)
Keys	O(n)
return_all_iems	O(n^2)

B4. Scalability and Adaptability

This application underlying data structure is very scalable, as the hash map will adapt its size to fit as many packages as passed into it.

B5. Software Efficiency and Maintainability

The efficiency of the code at worst case is O(n^2), but that is only for if you need to retrieve all items in the hash map. The delivery algo runs in O(n) time so I would say its very Efficient. The efficiency of the delivery times and distance mostly relies on careful ordering of packages onto trucks.

B6. Self-Adjusting Data Structures

A HashMap was used as the self adjusting data structure

Strengths	Weaknesses
Speed in which to access items	Does not allow null or empty values
Insertion time complexity = O(1) on new keys	Linearly slower as the load factor increases
Can grow as large as memory space allows	

D. Data Structure

I used a chaining hash table with linear probing for retrieval as the self-adjusting data structure

D1. Explanation of Data Structure

The chaining has table can perform insertion and deletion. The insertion function also performs the update action if provided with the same object. The chaining hash table does perform slower then a standard hashing but avoids the issue of hash collisions. The find_next_shortest_delivery uses the unique package ID value of the package to retrieve objects from the hash table.

The table has 3 basic functions. Add, get, delete. Add adds or updates a item in the table. Get returns and item based on the key provided, and delete deletes a item based on the key provided.

G1. First Status Check

main

Enter a time (HH:MM:SS) in 24h format: 9:00:00

Package ID	Package Address	Deadline	Delivery Time	Status
10	600 E 900 South	EOD	10:28:40	AT HUB
11	2600 Taylorsville Blvd	EOD	12:12:20	AT HUB
20	3595 Main St	10:30 AM	08:29:40	DELIVERED
12	3575 W Valley Central Station bus Loop	EOD	10:08:00	AT HUB
21	3595 Main St	EOD	08:29:40	DELIVERED
30	300 State St	10:30 AM	09:05:40	EN ROUTE
13	2010 W 500 S	10:30 AM	09:19:40	EN ROUTE
22	6351 South 900 East	EOD	11:49:40	AT HUB
31	3365 S 900 W	10:30 AM	09:40:00	AT HUB
40	380 W 2880 S	10:30 AM	08:35:00	DELIVERED
14	4300 S 1300 E	10:30 AM	08:06:20	DELIVERED
23	5100 South 2700 West	EOD	12:13:40	AT HUB
32	3365 S 900 W	EOD	09:40:00	AT HUB
15	4580 S 2300 E	9:00 AM	08:13:00	DELIVERED
24	5025 State St	EOD	11:39:20	AT HUB
33	2530 S 500 E	EOD	11:25:00	AT HUB
16	4580 S 2300 E	10:30 AM	08:13:00	DELIVERED
25	5383 South 900 East #104	10:30 AM	09:13:00	AT HUB
34	4580 S 2300 E	10:30 AM	08:13:00	DELIVERED
17	3148 S 1100 W	EOD	11:12:40	AT HUB
26	5383 South 900 East #104	EOD	09:13:00	AT HUB
35	1060 Dalton Ave S	EOD	10:57:20	AT HUB
18	1488 4800 S	EOD	10:03:40	AT HUB
27	1060 Dalton Ave S	EOD	10:57:20	AT HUB
36	2300 Parkway Blvd	EOD	09:50:20	AT HUB
19	177 W Price Ave	EOD	10:03:20	AT HUB
28	2835 Main St	EOD	09:30:20	AT HUB
37	410 S State St	10:30 AM	09:02:20	EN ROUTE
29	1330 2100 S	10:30 AM	08:48:00	DELIVERED
38	410 S State St	EOD	10:39:00	AT HUB
39	2010 W 500 S	EOD	10:52:00	AT HUB
1	195 W Oakland Ave	10:30 AM	08:38:40	DELIVERED
2	2530 S 500 E	EOD	11:25:00	AT HUB
3	233 Canyon Rd	EOD	10:42:20	AT HUB
4	380 W 2880 S	EOD	08:35:00	DELIVERED
5	410 S State St	EOD	09:02:20	EN ROUTE
6	3060 Lester St	10:30 AM	09:45:00	AT HUB
7	1330 2100 S	EOD	08:48:00	DELIVERED
8	300 State St	EOD	09:05:40	EN ROUTE
9	300 State St	EOD	10:38:00	AT HUB

G2. Second Status Check

main

Enter a time (HH:MM:SS) in 24H format: 10:00:00

Package ID	Package Address	Deadline	Delivery Time	Status
10	600 E 900 South	EOD	10:28:40	EN ROUTE
11	2600 Taylorsville Blvd	EOD	12:12:20	EN ROUTE
20	3595 Main St	10:30 AM	08:29:40	DELIVERED
12	3575 W Valley Central Station bus Loop	EOD	10:08:00	EN ROUTE
21	3595 Main St	EOD	08:29:40	DELIVERED
30	300 State St	10:30 AM	09:05:40	DELIVERED
13	2010 W 500 S	10:30 AM	09:19:40	DELIVERED
22	6351 South 900 East	EOD	11:49:40	EN ROUTE
31	3365 S 900 W	10:30 AM	09:40:00	DELIVERED
40	380 W 2880 S	10:30 AM	08:35:00	DELIVERED
14	4300 S 1300 E	10:30 AM	08:06:20	DELIVERED
23	5100 South 2700 West	EOD	12:13:40	EN ROUTE
32	3365 S 900 W	EOD	09:40:00	DELIVERED
15	4580 S 2300 E	9:00 AM	08:13:00	DELIVERED
24	5025 State St	EOD	11:39:20	EN ROUTE
33	2530 S 500 E	EOD	11:25:00	EN ROUTE
16	4580 S 2300 E	10:30 AM	08:13:00	DELIVERED
25	5383 South 900 East #104	10:30 AM	09:13:00	DELIVERED
34	4580 S 2300 E	10:30 AM	08:13:00	DELIVERED
17	3148 S 1100 W	EOD	11:12:40	EN ROUTE
26	5383 South 900 East #104	EOD	09:13:00	DELIVERED
35	1060 Dalton Ave S	EOD	10:57:20	EN ROUTE
18	1488 4800 S	EOD	10:03:40	EN ROUTE
27	1060 Dalton Ave S	EOD	10:57:20	EN ROUTE
36	2300 Parkway Blvd	EOD	09:50:20	DELIVERED
19	177 W Price Ave	EOD	10:03:20	EN ROUTE
28	2835 Main St	EOD	09:30:20	DELIVERED
37	410 S State St	10:30 AM	09:02:20	DELIVERED
29	1330 2100 S	10:30 AM	08:48:00	DELIVERED
38	410 S State St	EOD	10:39:00	EN ROUTE
39	2010 W 500 S	EOD	10:52:00	EN ROUTE
1	195 W Oakland Ave	10:30 AM	08:38:40	DELIVERED
2	2530 S 500 E	EOD	11:25:00	EN ROUTE
3	233 Canyon Rd	EOD	10:42:20	EN ROUTE
4	380 W 2880 S	EOD	08:35:00	DELIVERED
5	410 S State St	EOD	09:02:20	DELIVERED
6	3060 Lester St	10:30 AM	09:45:00	DELIVERED
7	1330 2100 S	EOD	08:48:00	DELIVERED
8	300 State St	EOD	09:05:40	DELIVERED
9	300 State St	EOD	10:38:00	EN ROUTE

G3. Third Status Check

main

Enter a time (HH:MM:SS) in 24H format: 12:30:00

Package ID	Package Address	Deadline	Delivery Time	Status
10	600 E 900 South	EOD	10:28:40	DELIVERED
11	2600 Taylorsville Blvd	EOD	12:12:20	DELIVERED
20	3595 Main St	10:30 AM	08:29:40	DELIVERED
12	3575 W Valley Central Station bus Loop	EOD	10:08:00	DELIVERED
21	3595 Main St	EOD	08:29:40	DELIVERED
30	300 State St	10:30 AM	09:05:40	DELIVERED
13	2010 W 500 S	10:30 AM	09:19:40	DELIVERED
22	6351 South 900 East	EOD	11:49:40	DELIVERED
31	3365 S 900 W	10:30 AM	09:40:00	DELIVERED
40	380 W 2880 S	10:30 AM	08:35:00	DELIVERED
14	4300 S 1300 E	10:30 AM	08:06:20	DELIVERED
23	5100 South 2700 West	EOD	12:13:40	DELIVERED
32	3365 S 900 W	EOD	09:40:00	DELIVERED
15	4580 S 2300 E	9:00 AM	08:13:00	DELIVERED
24	5025 State St	EOD	11:39:20	DELIVERED
33	2530 S 500 E	EOD	11:25:00	DELIVERED
16	4580 S 2300 E	10:30 AM	08:13:00	DELIVERED
25	5383 South 900 East #104	10:30 AM	09:13:00	DELIVERED
34	4580 S 2300 E	10:30 AM	08:13:00	DELIVERED
17	3148 S 1100 W	EOD	11:12:40	DELIVERED
26	5383 South 900 East #104	EOD	09:13:00	DELIVERED
35	1060 Dalton Ave S	EOD	10:57:20	DELIVERED
18	1488 4800 S	EOD	10:03:40	DELIVERED
27	1060 Dalton Ave S	EOD	10:57:20	DELIVERED
36	2300 Parkway Blvd	EOD	09:50:20	DELIVERED
19	177 W Price Ave	EOD	10:03:20	DELIVERED
28	2835 Main St	EOD	09:30:20	DELIVERED
37	410 S State St	10:30 AM	09:02:20	DELIVERED
29	1330 2100 S	10:30 AM	08:48:00	DELIVERED
38	410 S State St	EOD	10:39:00	DELIVERED
39	2010 W 500 S	EOD	10:52:00	DELIVERED
1	195 W Oakland Ave	10:30 AM	08:38:40	DELIVERED
2	2530 S 500 E	EOD	11:25:00	DELIVERED
3	233 Canyon Rd	EOD	10:42:20	DELIVERED
4	380 W 2880 S	EOD	08:35:00	DELIVERED
5	410 S State St	EOD	09:02:20	DELIVERED
6	3060 Lester St	10:30 AM	09:45:00	DELIVERED
7	1330 2100 S	EOD	08:48:00	DELIVERED
8	300 State St	EOD	09:05:40	DELIVERED
9	300 State St	EOD	10:38:00	DELIVERED

H. Screenshots of Code Execution

```
main
/home/avatre0/PycharmProjects/DSA2/venv/bin/python /home/avatre0/PycharmProjects/DSA2/main.py
WGUPS Package System
Delivering Packages
Truck 1 has finished its route and returned to the hub at 09:56:00
Truck 1 drove 23.90 miles
Truck 2 has finished its route and returned to the hub at 11:07:40
Truck 2 drove 29.20 miles
Truck 3 has finished its route and returned to the hub at 12:35:00
Truck 3 drove 41.30 miles
Total Distance Driven is: 94.40 miles
-------------------------------------------------
Please select from the options below:
1. Get Status for single package
2. Get Status of all Packages at a time
3. Quit
Enter Your Selection:
```

I1. Strengths of Chosen Algorithm

Accuracy: This algorithm delivers the packages on time and under millage. Delivering the packages in 94.40 miles.

Efficiency: The greedy algorithm is O(n) and thus is very efficient, and easy to implement

I3. Other possible Algorithms

Other possible approaches include:

Brute force algorithm.

Dijkstra's algorithm

I3A. Algorithm Differences

	Brute Force	Dijkstra	Greedy
Time Complexity	O(n*m)	O(n^2)	O(n)
Advantages	Finds all routes	Low relative complexity	Easy to implement and low complexity

J. Different Approach

If I were to do this project again I would change the following,

Package assignment. As it is now, in order for the packages to be delivers on time and with the special requirements (only arriving at 9:05, only on truck 2, being delivered with other packages etc), requires the packages to be pre-sorted by hand onto the tucks. I would like to develop some auto sorting algo possibly based on zip code. Also this would require the package objects to have flags for the special requirements.

K1. Verification of Data Structure

The chaining hash table, meets all requirements. It delivered the packages in a total of 94.40 miles. All packages were delivered before deadlines. The table was also able to quickly lookup data inside it.

K1A. Efficiency

The worst case for lookup on the hash table is O(n). This table uses linear probing to find the packages. As packages are added the time to perform the lookup will increase.

K1B. Overhead

As packages are added to the hash table, the likelihood of hash collisions increases, increasing the size of the table. This also affects lookup time as the table will not only need to find the key in the hash, but then find the object in the linked list at the key. To combat this we can increase the size of the table at creation.

K1C. Implications

Changing the number of trucks will not increase the lookup time or space requirement of the hash table. The table does not store data based on trucks. Like wise changing he number of cities shouldn't make any difference in the lookup process or time-space complexities.

The map only contains package object data, unless we add more packages there should be no changes to the lookup time or time-space complexities.

K2. Other Data Structures

Two other data structures that could be used are

1) Weighted Graph

2) Stack

K2a. Data Structure Differences

Stack: A stack could be created with the order in which packages are to be delivered. You push the item on the stack then pop them off to get the next package to be delivered. This would be more space efficient, but would require pre-loading of the trucks and finding the best path between them before pushing them onto the stack

Weighted Graph: Nodes would represent package delivery addresses. This would eliminate the need to find the next efficient route as it is built into the traversal. The graph would likely be more space-time efficient.