C950 Task 1 Writeup

Xavier Loera Flores
ID: 011037676
xloeraf@wgu.edu
C950 Data Structures and Algorithms II

Algorithm Overview

Stated Problem

The Western Governors University Parcel Service (WGUPS) needs to determine an efficient route and delivery distribution for their daily local deliveries (DLD) because packages are not currently being consistently delivered by their promised deadline. The Salt Lake City DLD route has three trucks, two drivers, and an average of 40 packages to deliver each day. Each package has specific criteria and delivery requirements that are listed in the attached “WGUPS Package File.”

Your task is to determine an algorithm, write code, and present a solution where all 40 packages will be delivered on time while meeting each package’s requirements and keeping the combined total distance traveled under 140 miles for two of the trucks. The specific delivery locations are shown on the attached “Salt Lake City Downtown Map,” and distances to each location are given in the attached “WGUPS Distance Table.” The intent is to use the program for this specific location and also for many other cities in each state where WGU has a presence. As such, you will need to include detailed comments to make your code easy to follow and to justify the decisions you made while writing your scripts.

The supervisor should be able to see, at assigned points, the progress of each truck and its packages by any of the variables listed in the “WGUPS Package File,” including what has been delivered and at what time the delivery occurred.

Assumptions

- Each truck can carry a maximum of 16 packages, and the ID number of each package is unique.
- The trucks travel at an average speed of 18 miles per hour and have an infinite amount of gas with no need to stop.
- There are no collisions.

- Three trucks and two drivers are available for deliveries. Each driver stays with the same truck as long as that truck is in service.
- Drivers leave the hub no earlier than 8:00 a.m., with the truck loaded, and can return to the hub for packages if needed.
- The delivery and loading times are instantaneous (i.e., no time passes while at a delivery or when moving packages to a truck at the hub). This time is factored into the calculation of the average speed of the trucks.
- There is up to one special note associated with a package.
- The delivery address for package #9, Third District Juvenile Court, is wrong and will be corrected at 10:20 a.m. WGUPS is aware that the address is incorrect and will be updated at 10:20 a.m. However, WGUPS does not know the correct address (410 S. State St., Salt Lake City, UT 84111) until 10:20 a.m.
- The distances provided in the “WGUPS Distance Table” are equal regardless of the direction traveled.
- The day ends when all 40 packages have been delivered.

A. Algorithm Overview

Our solution to solve the WGUPS Delivery Routing Problem is to develop a delivery routing program that utilizes a greedy algorithm that uses a nearest neighbor algorithm to find an efficient route and delivery distribution.

B. Data Structure And Relationship To Components

The package delivery program utilizes a hash table to store the delivery package data and a weighted adjacency matrix, or distance matrix to store the distances between addresses.The hash table data structure contains all the raw package data contained in a package object. The package object contains the package ID, delivery address, deadline, city, zip code, and weight. Alongside the package object, the hash table stores a delivery package object which contains the delivery status, delivery time if delivered, and address object. The distance matrix matrix stores the distances between the addresses. It can look up the distance between two addresses by using the address ID for the row and column index in constant time. The program will update the delivery package object in the hash table after determining which package to deliver first based on the closest address in the distance matrix.

C. Overview

The greedy nearest neighbor algorithm is a viable solution to the delivery routing program because it can efficiently find an optimal route and delivery distribution. The nearest neighbor algorithm is a greedy algorithm that finds the shortest path between two points. The nearest neighbor algorithm is a greedy algorithm because it makes the locally optimal choice at each step.

1. Algorithm Pseudocode

function delivery_routing_system(time):

- initialize the trucks
- initialize the current time
- Run delivery_algorithm for the truck1 and truck2
- Compare the truck1 and truck2 return times
- If truck1 returns first:
    - Run delivery_algorithm for truck3 with the return time of truck1
- Else:
    - Run delivery_algorithm for truck3 with the return time of truck2
- print the system

function load_packages:

- Get the list of packages
- For each package in the list:
    - Create a delivery package from the package
    - Get the index of the delivery package
    - If the package is delayed or the index is 9:
        - Load the package onto truck 3
    - Else if the package is for truck 2 or the package is bundled or the index is 19:
        - Load the package onto truck 2
    - Else if the package has a deadline:
        - Load the package onto truck 1
    - Else:
        - If the load of truck 2 is less than the load of truck 3:
            - Load the package onto truck 2
        - Else:
            - Load the package onto truck 3

function delivery_algorithm(truck, time):

- Attempt to depart with the truck

- Identify the next package to deliver
- While the time is None or the current time is less than the truck's time and there is a next package:
    - Deliver the package
    - Identify the next package to deliver
- Return the truck to the hub
- Return the truck's time

function identify_next_package(truck):

- initialize a flag for wrong address package as False
- Get the list of undelivered packages from the truck
- Get the current address ID of the truck
- initialize lists for undelivered packages addresses and priority packages addresses
- For each package in the undelivered packages:
    - Get the package from the hash table
    - Get the address ID of the package
    - If the package ID is 9, set the wrong address package flag to True
    - Else if the package has a deadline, add the address ID to the priority packages addresses list
    - Else, add the address ID to the undelivered packages addresses list
- Get the closest address from the undelivered packages addresses to the current address
- If there are priority packages, get the closest address from the priority packages addresses to the current address

2. Programming Environment

The programming environment features Python version 3.8.9 which is the ninth maintenance release of Python 3.8 released on April 2, 2021.

The programming environment ran within MacOS Monterey Version 12.0.1 on a 16 inch MacBook Pro with an Apple M1 Pro processor and 32GB of unified memory.

3. Space Time Complexity

main.py

Method	Line Number	Time Complexity	Space Complexity
delivery_routing_system	12	O(n^2log(n)	O(n)
load_packages	24	O(n)	O(n)
delivery_algorithm	42	O(n^2log(n))	O(n)
identify_next_package	52	log(n)	(n)
interface_loop	79	O(n^2log(n))	(n)
main	103	O(n^2log(n))	O(n)

HashTable.py

Method	Line Number	Time Complexity	Space Complexity
Node.__init__	3	O(1)	O(1)
HashTable.__init__	11	O(n)	O(n)
hash_key	19	O(1)	O(1)
insert	24	O(n)	O(1)
update	37	O(n)	O(1)

get	48	O(n)	O(1)

DistanceMatrix.py

Method	Line Number	Time Complexity	Space Complexity
read_distance	4	O(n)	O(n)
get_distance_between_addresses	24	O(1)	O(1)
get_closest_address	31	O(n)	O(1)

System.py

Method	Line Number	Time Complexity	Space Complexity
create_system	17	O(n^2)	O(n)
print_system	31	O(n^2)	O(n)

Truck.py

Method	Line Number	Time Complexity	Space Complexity
__init__	20	O(1)	O(1)
load_package	40	O(1)	O(1)

str	48	O(1)	O(1)
set_depart_time	54	O(1)	O(1)
attempt_depart	59	O(1)	O(1)
get_undelivered_packages	66	O(n)	O(n)
get_undelivered_packages_addresses	75	O(n)	O(n)
deliver_package	82	O(1)	O(1)
return_to_hub	96	O(1)	O(1)
travel_for_time_in_min	105	O(1)	O(1)
set_address_id	111	O(1)	O(1)
get_priority_packages	115	O(n)	O(n)

DeliveryPackage.py

Method	**Line Number**	**Time Complexity**	**Space Complexity**
__init__	24	O(1)	O(1)
print	35	O(1)	O(1)
get_id	60	O(1)	O(1)

set_status_to_delivered	64	O(1)	O(1)
set_status_to_enroute	68	O(1)	O(1)
set_delivered_time	72	O(1)	O(1)
set_truck_id	76	O(1)	O(1)

Package.py

Method	Line Number	Time Complexity	Space Complexity
read_packages	4	O(n)	O(n)
raw_packages_to_packages	14	O(n)	O(n)
__init__	22	O(1)	O(1)
has_deadline	38	O(1)	O(1)
has_notes	42	O(1)	O(1)
is_delayed	46	O(1)	O(1)
is_truck_2	50	O(1)	O(1)
has_wrong_address	54	O(1)	O(1)
is_bundled	58	O(1)	O(1)

Address.py

Method	Line Number	Time Complexity	Space Complexity
read_addresses	4	O(n)	O(n)
get_address_list	14	O(n)	O(n)
__init__	26	O(1)	O(1)
get_address_by_id	36	O(n)	O(1)
get_address_by_street	45	O(n)	O(1)

Utils.py

Method	Line Number	Time Complexity	Space Complexity
__init__	7	O(1)	O(1)
read	11	O(n)	O(n)
convert_time_str_to_time	20	O(1)	O(1)
convert_time_str_to_time_extended	26	O(1)	O(1)
print_red	31	O(1)	O(1)
print_green	35	O(1)	O(1)

print_blue	40	O(1)	O(1)
print_yellow	45	O(1)	O(1)
get_distance_traveled	49	O(1)	O(1)
get_time_traveled	53	O(1)	O(1)

4. Solution scalability & adaptability

The solution as a whole has a time complexity of O(n^2log(n)) as well as a space complexity of O(n). As the number of packages increases, the time complexity for the entire program should also increase at a rate of n^2log(n) while the space will increase at a rate of n. The program isn't too adaptable to an increased amount of packages since the three trucks may not have enough time to deliver all the packages on time. The program isn't automatically written to adapt to the addition of more trucks or drivers since the 3 trucks and 2 drivers are hard programmed into the the delivery routing system function.

5. Software Design Efficiency & Maintainability

The software was designed in a way that is efficient and maintainable. The software is efficient because it utilizes a greedy algorithm that finds the shortest path between two points. The software is maintainable because it is written in a way that is easy to understand and can be easily modified to meet future requirements. The delivery routing system features many functions that are modular and can be easily modified to meet future requirements. The delivery routing system also features many comments that explain the purpose of each function and the logic behind the code.

6. Hash Table Strengths & Weaknesses

Strengths:

- Hash tables are very efficient for lookups since they can happen in constant time O(1).
- Hash tables are also very efficient for insertions and deletions since they only need to change to update one array index for each operation as opposed to a

linear data structure like an array or linked list which would need to update multiple indexes.
- Hash tables are very flexible and can be used to store any type of data structure.
- Hash tables feature collision resolution which handles the cases for multiple key-values hashing to the same index.

Weaknesses:

- Hash tables can be inefficient when there are a large number of collisions.
- Hash tables do not allow null values
- Hash tables have a limited capacity and will need to be resized or utilize linked nodes if the number of elements exceeds the capacity.
- Hash tables are unordered which do not allow for sorting and make it difficult to retrieve the data in a specific order.

7. Key Justification

The delivery routing system will utilize the package's package ID as the key to access the package's data. The package ID is a unique identifier for each package and is a very efficient way to access the package's data from the hash table.

Resources

I did not use any external sources to complete this writeup.