C950 Task 1 Algorithm Overview

Planning Phase for WGUPS

Robert Wright
ID: 010924506
rwri326@wgu.edu
January 10, 2024

C950 Data Structures and Algorithm II

A. :ALGORITHM IDENTIFICATION

Stated Problem:

The program's goal was to create an algorithm that could efficiently deliver 40 packages to various addresses in Salt Lake City, Utah. The algorithm needs to be able to deliver the packages by end of day (EOD), choosing the most efficient route. Certain constraints are added that need to implemented in the algorithm Such as:

- *Each truck carries max of 16 packages, package ID number is unique.*
- *The trucks speed is 18 miles per hour no stopping.*
- *No collisions.*
- *Three trucks and two drivers.*
- *Drivers leave the hub at 8:00 a.m., can return to hub if needed.*
- *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.*

** *https://tasks.wgu.edu/student/010924506/course/30860017/task/4041/overview*

A.Algorithm Overview & C1. Logic

The algorithm used in the program utilizes the nearest neighbor algorithm. The algorithm is divided into three different functions. By dividing the logic into different functions each is given a specific role that work together to implement the delivery strategy in the most efficient way. Each section of the algorithm is explained below:

1. find_closest_package Function:

This function is composed of the main logic of the nearest neighbor algorithm. The overall purpose is to find the package that will require the truck to travel the least distance based on it's current location. Variables **closest_package** and **shortest_distance** tracks the closest package and distance from the truck. The function will then loop through each package in **un_delivered**. After that the distance needs to be calculated. This is determined as followed:

- Determines the trucks current location
- Then, converts address into an index
- Distance is then calculated from the current location to delivery location.

After finding closest package **shortest_distance** is the updated with the new distance, **closest_package** is set to the new current package.

Pseudocode for Closest Package:

```
function find_closest_package(truck, undelivered_packages,
distance_table)
    closest_package = None
    shortest_distance = infinity

    for package in undelivered_packages
        current_location = truck.current_location
        destination =
address_to_index(package.delivery_address)
        distance = calculate_distance(current_location,
destination, distance_table)

        if distance < shortest_distance
            shortest_distance = distance
            closest_package = package

    return closest_package, shortest_distance
```

Pseudocode terms:

'truck' object that includes current location.

'undelivered_packages' contains list of undelivered package object.

'distance table' stores distance between different location

'address_to_index' function matches address with distance.

'calculate_distance' function distance between truck and remaining addresses.

2. update_truck_after_delivery Function:

This function is utilized by the algorithm to update the delivery trucks after each completed delivery. The function itself isn't an algorithm. But updates the status and logistical details. The function accounts for time it takes to get to each delivery by calculating **shortest_distance / SPEED_LIMIT** . The delivery time is then updates to match delivery trucks time. While keeping track of when package was delivered.

Pseudocode for Delivery Truck Update.

```
function update_truck_after_delivery(truck, package, shortest_distance)

     truck.current_location = address_to_index(package.delivery_address)
     truck.total_mileage += shortest_distance
     if truck.available_at_time is not None
        truck.available_at_time = add_hours_to_time(truck.available_at_time,
     shortest_distance / SPEED_LIMIT)
     else
        truck.available_at_time = add_hours_to_time(current_time(),
     shortest_distance / SPEED_LIMIT)
     package.delivery_time = truck.time
```

Pseudocode terms:

'truck' object for current location, total mileage, and available time.

'package' package that was just delivered.

'shortest_distance' distance the truck traveled to deliver the package.

'address_to_index' function to convert address to location.

'add_hours_to_time' function adds a hours to a given time, used to calculate the new available time of the truck after delivery.

'current_time()' function that returns current time, used when truck's available time is not set.

3. Delivering Package Function:

This function uses the last functions of the algorithm that controls the process of delivering the packages. It also incorporates it's own use of a greedy neighbor algorithm. The algorithm starts by gathering list of undelivered packages. The **package_hash_table** is used to access the information needed from a hash table. It then checks from missing packages to ensure no packages were missed. The delivery process utilizes a while loop that will continue to loop until all packages are delivered. Then find_closest_package algorithm finds closest package's based on truck's location and proceeds to location. Departure time and status are updated continuously. The package is then removed from both truck and package list.

Pseudocode:

```
function delivering_packages(truck, package_hash_table,
reader_distance_list)
    undelivered_packages = create a list to store packages
    for each packageID in truck.packages
        package = lookup packageID in package_hash_table
        if package is not found
            print an error message indicating missing package
        else
            add package to undelivered_packages

    while undelivered_packages is not empty
        closest_package, shortest_distance =
find_closest_package(truck, undelivered_packages,
reader_distance_list)

        if closest_package is not found
            continue to the next iteration of the loop

        if truck.available_at_time exists
            set closest_package.departure_time to
truck.available_at_time
            update status of closest_package based on
truck.available_at_time

        update truck state after delivery, passing truck,
closest_package, and shortest_distance
        set closest_package.delivery_time to truck's current
time
        update status of closest_package based on truck's
current time

        remove closest_package.id from truck.packages
        remove closest_package from undelivered_packages

    return some form of delivery completion status or report
```

Pseudocode Terms:

’Truck’ is the object representing the delivery truck.

‘package_hash_table’ is a data structure that allows quick access to package information using package IDs.
‘reader_distance_list’ might be a data structure or a list containing distance-related information used in the ‘find_closest_package’ function.
‘Undelivered’ starts off as a empty list before initializing.

B1-2 Identification and Explanation of Data Structure

The data structure that was used in the program was a hash table. Since I’m using the nearest neighbor algorithm. The hash table doesn’t assist but can efficiently handle the data while the nearest neighbor determines the shortest route. The hash table key allowed me to make sure the relevant information in each csv matched correctly. The insert allows new data to be added and determines the bucket location. The lookup operation was utilized to make sure information was being calculated correctly throughout the main script. The hash table was useful in how it retrieves the data with an average time complexity of O(1). The hash table would allow us to handle a much large database vs a linear search which is only useful with small ones.

C2. Development Environment

This delivery routing system program was written in the Python 3.11 using Pycharm 2023.3.2. To test the efficiency of the program a command line interface was created. The “homepage” of the user interface (UI) created will have selection for checking the status of all packages or a single package based on the time inputted by the user. It will also show the total mileage of each delivery truck and total combined of all trucks. Once the user selects an option the user then inputs a time. A (Package ID, Address, Delivered Time, or the current status if the package has not been delivered) will then be displayed. The program was completed using MACos on a MacBook M1.

C3. Space Time Complexity with Big-O

1. **find_closest_pacakage:** *Space= O(1), Time=O(N)*
   - Outer Loop "`for package in undelivered`" iterates over each undelivered package "n" times. Space Complexity= not counted, Time Complexity= O(N).
   - **`address_to_index`** based on size input. Space Complexity = O(1) Time Complexity = O(A).
   - **`distance_table`** based on input size. Space Complexity = O(1), Time Complexity = O(D).

Overall Space Complexity = O(1). Time Complexity = O(N * (A + D)) = O(N)

2. **delivering_packages:** *Space = O(N), Time O(N^2)*
   - **`undelivered_packages`** iterates over created list for. Space Complexity = O(N) Time Complexity = O(N).
   - **If none in undelivered_packages** scans over list. Space Complexity = O(1), Time Complexity = O(N).
   - While loop iterates for all undelivered_packages. Space Complexity = O(1), Time Complexity O(N).
     - A. find_closest_package inside while loop has Space Complexity = O(1). Time Complexity = O(N * M).

Overall Space Complexity = O(N), Time Complexity = O(N +(N*M) = O(N^2).

The two nearest neighbor(greedy) algorithm's combined operations would have an average case of O(n^2) with a worst case of O(n^3). Since the list created decreases in size with each package delivered. The overall would average around O(n^2) for the algorithm.

C4 & C5 SCALABILITY AND ADAPTABILITY

Since the hash table is being used to access the data. The allows for specific information to be gathered quickly such as: destination, status, priority. The algorithm. The algorithm allows the usage of more functions that help show more information related to specific data. For example: showing the average time it takes for each truck to get to it's destination. The hash table and nearest neighbor algorithm would be able to handle larger datasets if more packages were added to the system. Due to nearest neighbor algorithm being easy to use with a Time Complexity of O(n) and O(n^2). You could add more algorithm functions to allow more analysis on the data and better optimization. The scripts have comment sections over each code block with function names that describe it's purpose. This allows for other programmers to adapt and change the program due to specific needs. Such as delivery route constraints that apply to different cities.

C6. Strength & Weakness of Hash Tables.

The self adjusting aspect of the hash table allows for better optimization. It allows it to adjust to improve the access time. Other forms of self adjusting data structures would be a binary search tree. Hash Table can adjust to to prevent potential collisions. Hash table relocates frequently used elements to improve faster access.
If the hash table data were to grow resizing can cause performance issues. The choice of using dynamic resizing could be more complex, if the program would benefit better from just a static resizing like an array.

C7. Data Key

The delivery routing system uses the package id as the key. Package id must remain unique to prevent delivery to wrong address. So it makes it an obvious choice. This allows easy retrieval and access to the package data. The delivering_address algorithm uses the package id key for easy lookup. Address could also be useful for grouping packages going to the same address, which could improve route optimization. But addresses are not unique to individual packages. So the key would need to handle multiple packages, potentially causing collision issues.

D. Sources, in-text citations and references.

Adamczyk, J. (2022, March 30). k nearest neighbors computational complexity - Towards Data Science. *Medium*. https://towardsdatascience.com/k-nearest-neighbors-computational-complexity-502d2c440d5

zyBooks: Figure 7.8.2: Hash table using chaining.

SV File Reading and Writing. (n.d.-b). Python Documentation. https://docs.python.org/3/library/csv.html

How To Convert a String to a datetime or time Object in Python. DigitalOcean. https://www.digitalocean.com/community/tutorials/python-string-to-datetime-strptime

Python datetime.timedelta function. https://www.geeksforgeeks.org/python-datetime-timedelta-function/

K Nearest neighbours — Introduction to machine learning algorithms. Medium. https://towardsdatascience.com/k-nearest-neighbours-introduction-to-machine-learning-algorithms-18e7ce3d802a

Shovic, J. C., & Simpson, A. (2019). *Python All-in-One for dummies*. https://openlibrary.org/books/OL27759609M/Python_All-in-One_For_Dummies. Working with External Libraries.