C950 Task-1 WGUPS Algorithm Overview

Task 1: WGUPS Routing Program Planning

Michael Castro

ID # 001084095

WGU Email: mcas175@wgu.edu

November 22, 2023

C950 Data Structures and Algorithms II

Introduction

The delivery program focuses on optimizing package delivery efficiency through the use of a self-adjusting Greedy Algorithm and hash tables. This combination enables real-time route optimization and efficient information management. Also, a user-friendly GUI is used to provide users with a rich, intuitive interface. The project's approach ensures constant-time access, effective collision handling, and alignment with the Greedy Algorithm's requirements, making it a robust solution for efficient package delivery.

A. Algorithm Identification

Delivering packages while meeting specific requirements poses a significant challenge, one that can be effectively addressed by employing a self-adjusting algorithm to optimize a delivery route. Given these challenges, the self-adjusting Greedy Algorithm could be used to optimize package delivery. The algorithm works by prioritizing package deliveries by selecting the most advantageous routes at the moment and adjusting dynamically as it progresses, making it an effective choice for addressing delivery challenges.

B. Data Structure Identification

Complimenting the use of the Greedy Algorithm, a hash table is a self-adjusting data structure that can be employed to effectively address the challenges of package delivery.

B1. Explanation of Data Structure

By using hash table data as the self-adjusting structure, data components such as package ID, address, and delivery details, are effectively managed. Using unique package IDs as keys, a hash table allows for instant access to any package's information, facilitating key functions such as status updates and searches. The choice to use a hash table is driven by efficiency, offering constant-time access and updates, effective collision handling, and seamless alignment with the Greedy Algorithm's requirements for fast data access. Overall, the hash table adeptly supports the relationships and attributes necessary for an effective package delivery system.

C1. Algorithm's Logic

```
function deliverNextPackage( ):                    //Greedy algorithm for routing next package

        SET currentLocation to WGU Location                    // Initial location
        SET currentTime TO 08:00                               // Initial time
        SET totalMileage TO 0                                  // Initial total miles traveled

        CREATE empty LIST packageDeliveryList

        ASSIGN packages TO packageDeliveryList         //each element is a tuple with attributes:
                                                        //packageID, deadline, distance,
                                                       // urgencyFactor, urgencyValue

        WHILE packageDeliveryList COUNT > 0

            SET nextPackage TO none

            FOR each package in packageDeliveryList

                SET distance TO distance from current location TO package   //Using distance table
                SET urgencyValue TO urgencyFactor * distance                //urgencyFactor based on the
deadline

            ENDFOR

                SORT packageDeliveryList BY urgencyValue ASCENDING

                SET nextPackage TO packageDeliveryList at INDEX 0

                //Deliver nextPackage
                SET distanceTraveled TO nextPackage.distance          // Miles traveled

                SET timeTraveled TO nextPackage.distance / 18 MPH     // Time traveled

                UPDATE currentLocation TO nextPackage.Location        // Move to new location

                UPDATE totalMileage TO totalMileage + distanceTraveled   // Track total miles traveled

                UPDATE currentTime TO currentTime + timeTraveled      // Update time

                UPDATE nextPackage status TO Delivered                // Update package status

                SET nextPackage delivery.time TO currentTime          // Log time of delivery

                REMOVE nextPackage from packageDeliverList

        ENDWHILE
```

C2. Development Environment

The following hardware and software will be used to create this program:

Hardware Environment

- **Computer Model:** Surface Laptop Studio
- **Operating System:** Windows 11 Windows 11 Home Edition
- **Monitor:** The laptop's built-in monitor
- **Input Device:** Laptops built-in keyboard and touchpad.

Software Environment:

- **IDE:** For writing Python code, I'll be using PyCharm 2023.2.3 Community Edition
- **Python Version:** I will use Python version 3.9.2rc1 for this project.
- **GUI Framework:** PyQt 5.15.2 Designer
- **Operating System:** Windows 11 Home Edition

C3. Space and Time complexity using Big-O notation

Hash Tables

The space complexity for building the hash tables is O(n), as it requires storing each unique package. The time complexity is O(n), due to reading and inserting each package into the hash table, a process that scales linearly with the number of packages. Both searching and updating nodes in the hash table are highly efficient operations with space and time complexity of O(1).

Loading Package Algorithm

The package loading in the delivery system involves three steps:

1. Manually assigning packages with special requirements Time which has a time complexity of O(k) and a space complexity of O(n).

Loading Package Algorithm (continued)

2. Loading packages with a delivery deadline not End of Day which has a time complexity of O(n) and space complexity of O(1)
3. And then manually loading any remaining packages marked for end of day which has a time complexity of O(n) and space complexity of O(1)

Overall, this process has a linear time complexity of O(n) and a constant space complexity of O(1). This approach is useful as it allows the addressing of specific special requirements, creates an 'express delivery truck' to focus on express delivery packages (non-EOD), and allows for efficient handling of remaining packages.

Delivery Algorithm

In the delivery algorithm, the time complexity is influenced by two factors: the O(n) operation for calculating distances and urgency values for each package, and the O(n log n) complexity for sorting the packageDeliveryList. However, as this sorting is conducted within a loop iterating over each package, the overall time complexity escalates to O(n^2 log n). Regarding space complexity, the algorithm predominantly relies on the packageDeliveryList, which requires an O(n) space allocation. This approach allows for a more complex delivery algorithm while striking a balance between processing time and memory usage.

User Interface

This program uses PyQt-based GUI to display package delivery and manage deliveries. The space complexity is O(n), growing linearly with the number of packages handled by the package hash table. Static UI elements maintain a minimal constant space impact at (O(1)). The GUI excels in time efficiency, particularly in updating and searching package data, using the hash table's O(1). This design ensures a richer, responsive, and user-friendly experience, balancing memory usage and operational speed for effective package management.

C4. Scalability and Adaptability

The program's scalability and adaptability are rooted in the self-adjusting nature of the Greedy Algorithm, which dynamically optimizes package delivery routes. This algorithm efficiently handles varying package volumes and delivery constraints, making it well-suited for scaling to accommodate a growing number of packages. The use of hash tables to manage package data also lends the program's ability toe scale and adapt to a growing number of packages. This utilization of hash tables ensures swift access to package information, facilitating key functions and contributing to the program's ability to scale seamlessly as the number of packages grows. However, it's important to note that scalability may face challenges when dealing with a significant influx of packages with unique requirements or tight deadlines, potentially impacting optimization. Despite this, the modular program design allows for future enhancements to address scalability issues, ensuring it can adapt to evolving package delivery demands.

C5. Software Efficiency and Maintainability

The program's software design ensures efficiency by maintaining polynomial time complexity, meeting the requirement for scalability. The program's design emphasizes constant-time operations for package management, addressing efficiency concerns. Additionally, it prioritizes maintainability through modularity, code clarity, and thorough documentation, making it easy to understand, repair, and enhance by developers other than the author. This combination of efficiency and maintainability makes the program an effective solution for package delivery optimization while accommodating growth in the number of packages.

C6. Self-Adjusting Data Structures

The main advantage of using a hash table for this project lies in the data structure efficiency in data operations. As noted by Lysecky and Vahid (2018), a hash table allows for constant-time searching, insertion, and removal (O(1)). This is exceptionally fast compared to other data structures like lists (O(N)) or binary search (O(log N)). This efficiency ensures rapid data access and manipulation, making it an ideal choice for optimizing package data management and ensuring the program's responsiveness. However, it's important to note that using a hash table has a weakness in the form of increased storage requirements. This is due to the

need for empty buckets or unused slots, potentially leading to inefficient memory usage, particularly when dealing with a large range of possible keys

C7. Data Key

This project will employ the package ID as the key used for the hash table. The use of the package ID is the only data component that makes sense to use due to its unique nature as an identifier. Conversely, attributes such as delivery deadline, delivery city, delivery zip code, package weight, and delivery status lack uniqueness and can be associated with multiple packages. Utilizing these non-unique attributes as keys would introduce the potential for errors and hinder the efficiency of package management. By leveraging a unique identifier, the program mitigates the likelihood of collisions and thereby ensures the streamlined and effective operation of the program.

D. Sources

Lysecky, R., & Vahid, F. (2018, June). *C950: Data Structures and Algorithms II*. zyBooks.

Retrieved March 22, 2021, from https://learn.zybooks.com/zybook/WGUC950AY20182019/