C950 Task-2 WGUPS Write-Up

(Task-2: The implementation phase of the WGUPS Routing Program).

(Zip your source code and upload it with this file)

Alexander Fair

ID # 001574781

WGU Email: afair26@wgu.edu

3/5/2024

C950 Data Structures and Algorithms II

# A. Hash Table

A screenshot of a computer program

Description automatically generatedA computer screen shot of a bucket code

Description automatically generated

# B. Look-Up Functions

*formatpackagedetails* methodA screen shot of a computer program

Description automatically generated

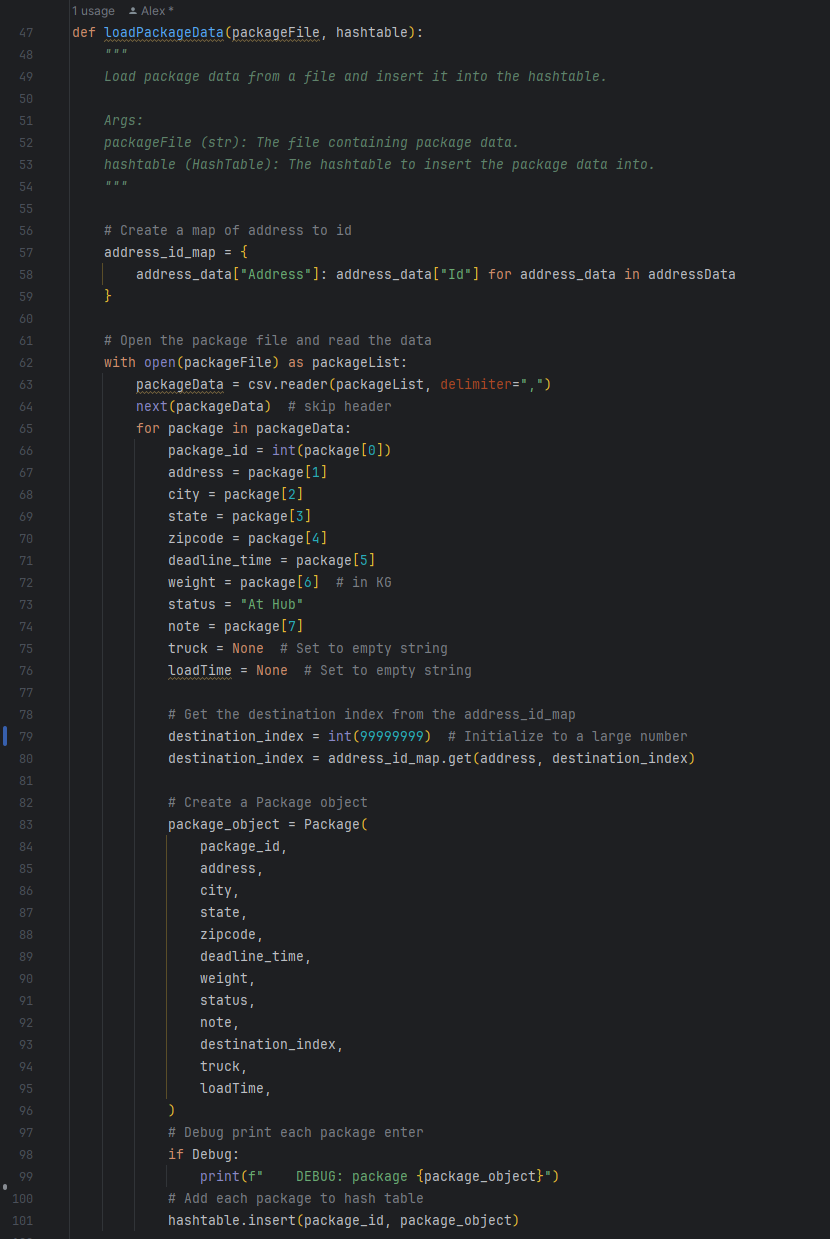
*reportDeliveryTime* methodA screen shot of a computer program

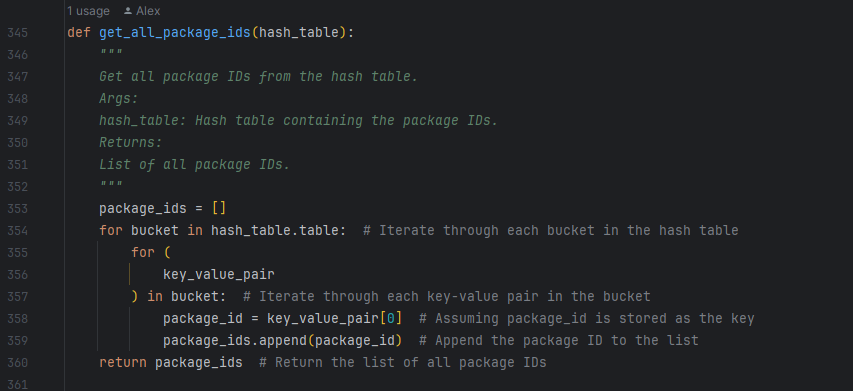
Description automatically generated

packageReport method A screen shot of a computer program

Description automatically generated

# C. Original Code

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screen shot of a computer program

Description automatically generated

# C1. Identification Information

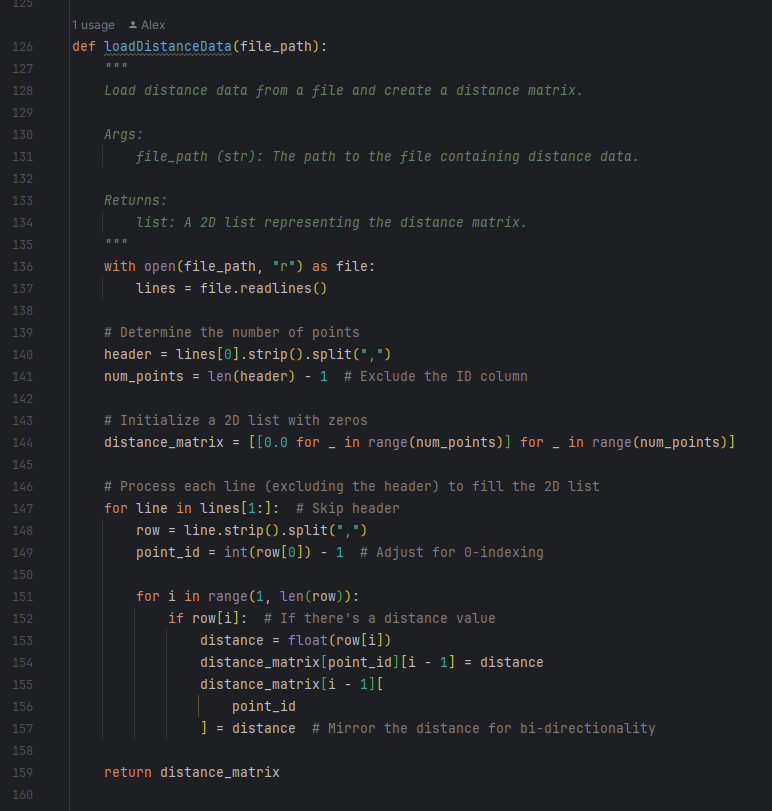
A screen shot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generatedA screen shot of a computer program

Description automatically generatedA computer screen shot of a program

Description automatically generatedA screen shot of a computer program

Description automatically generatedA computer screen with text

Description automatically generatedA screenshot of a computer program

Description automatically generatedA computer screen shot of a program

Description automatically generatedA computer screen shot of a program

Description automatically generatedA screen shot of a computer code

Description automatically generatedA screenshot of a computer program

Description automatically generatedA computer screen shot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screen shot of a computer program

Description automatically generated

# D. Interface

A screenshot of a computer

Description automatically generated

# D1. First Status Check

All Packages Check #1 @ 09:01:34


# D2. Second Status Check

All Packages Check #2 @ 10:02:49


# D3. Third Status Check

All Packages Check #3 @ 12:38:12


# E. Screenshot of Code Execution

A screenshot of a computer program

Description automatically generated

# F1. Strengths of the Chosen Algorithm

In my project, I implemented the Nearest Neighbor algorithm, drawn to its simplicity and straightforward nature. This choice significantly simplified the initial development phase, providing an intuitive framework that was easy to grasp and apply. The theoretical advantage of minimal preprocessing requirements stood out, especially considering scenarios involving frequently changing data, such as theoretical package routing in logistics. This aspect of the algorithm suggested it would be highly adaptable and efficient in environments where data updates are common, making it a potentially valuable tool for dynamic applications. The algorithm's straightforward implementation and the prospect of easily accommodating changes highlighted its suitability for projects that theorize on rapid data variability, underscoring its practicality for simulations or models requiring quick adaptation and straightforward logic.

# F2. Verify Algorithm

The algorithm used in my code meets all the requirements of the scenario. It routes and delivers (on time) the 40 packages on three trucks with only two drivers, resulting in a total combined mileage of under 140 miles. It uses the provided files (after some cleaning) and can also be reused for other package data, so it also can be used in other cities.

# F3. Other Possible Algorithms

I found two other possible algorithms: Dijkstra's (discussed in the code repository) and Greedy Best-First search.

# F3a. Algorithm Differences

Dijkstra's algorithm and Greedy Best-First Search differ primarily from the Nearest Neighbor algorithm in their application domains and operational mechanics. Dijkstra's algorithm is designed to find the shortest path in a graph from a single source to all other nodes, focusing on minimizing the cumulative distance without using heuristics. It guarantees the shortest path by systematically exploring all possible paths. In contrast, Greedy Best-First Search uses a heuristic to prioritize exploration toward the goal, focusing on reducing the estimated distance to the goal rather than ensuring the shortest path, which makes it faster but not consistently accurate.

# G. Different Approach

I would have implemented a weighting system for the packages based on the delivery time and some way to parse the notes so that the package assignment could be adequately automated. I also would have modified the package status method to use a dictionary instead of the elif code that it currently uses. I would create a constants.py file that would hold the constant variables so they are all in 1 place; this could make adding to the user interface so that the user could modify, say, the speed of the trucks or add new CSV files to the interface for it to run with instead of having them set.

# H. Verification of Data Structure

The hash table provides constant time complexity O(1) for average search insert and delete operations. It is crucial to quickly access package details based on their IDs during the routing process. Hash tables also have well-defined methods for handling collisions, ensuring that even in the cases of key conflicts, package data remains accessible and distinct. Hash tables also allow for flexible handling of package data, including addresses, delivery deadlines, delivery status, and weight.

# H1. Other Data Structures

Other data structures, such as trees (KD-trees or Ball Trees) or graphs, could have been utilized with Dijkstra's algorith.

# H1a. Data Structure Differences

Hash tables organize similar data points into "buckets" based on a mathematical formula. This method is akin to categorizing objects by similarity into containers, accelerating the search process because you only need to examine a limited number of containers instead of the entire collection. This approach is particularly beneficial when dealing with a vast quantity of data and when an approximate nearest neighbor is sufficient for the task, trading off a small amount of precision for speed.

Like KD-Trees and Ball Trees, trees systematically divide data into smaller segments based on their attributes, like organizing items into a hierarchical structure. This method is comparable to creating a directory tree that guides you to the item you are searching for, following specific criteria at each level. Trees are especially effective for datasets that are not overly complex or excessively large, providing a highly accurate method for finding the closest match.

Conversely, graphs create a network of data points by connecting them with edges based on their similarity. They are like a web of connections where each node (or data point) is linked to others it resembles; the density of these connections can highlight clusters of similar items. Searching for the nearest neighbor involves traversing this network, starting from a given node and exploring connected paths to find closely related nodes. Graphs excel in environments where relationships between data points are integral, offering a nuanced way to navigate through and identify clusters of similarity within large datasets.

# I. Sources

R. Fair, personal communication, February 20, 2024 (https://www.linkedin.com/in/rob-fair-262730a/)

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

Retrieved February 10, 2024, from <https://learn.zybooks.com/zybook/WGUC950AY20182019/>

# J. Professional Communication