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)

Andrew Hilleary

ID #011315946

WGU Email: ahil783@wgu.edu

4/5/2024

C950 Data Structures and Algorithms II

# Hash Table

# A screen shot of a computer program Description automatically generated

# B. Look-Up Functions

# A screen shot of a computer program Description automatically generated

# C. Original Code

A screen shot of a computer program

Description automatically generated

# C1. Identification Information

# A screen shot of a computer program Description automatically generated

# C2. Process and Flow Comments

A screen shot of a computer program

Description automatically generated

# D. Interface

A computer screen with many white and blue text

Description automatically generated

# D1. First Status Check

A screen shot of a computer screen

Description automatically generated

# D2. Second Status Check

A screen shot of a computer screen

Description automatically generated

# D3. Third Status Check

A screen shot of a computer screen

Description automatically generated

# E. Screenshot of Code Execution

A black screen with white text

Description automatically generated

# F1. Strengths of the Chosen Algorithm

The biggest strengths for this algorithm are its efficiency to run using a hash map, and that it allows for prioritization of packages. The ability to effectively use a hash map speeds up this algorithm significantly due to the search speed to pull out the package IDs. The list used to delay packages after a certain time can be easily swapped out for use in other special circumstances that a package may need.

# F2. Verification of Algorithm

Each truck has less than 17 packages and the ID of each package is unique. The trucks’ speed is used to calculate the time using the distance from each location. The min function is used to allow whichever driver of truck 1 or 3 gets back first to send the next out on truck 2. Each special note has been attended to. Package 9 is corrected after 10:20 and is delivered to the right address.

# F3. Other Possible Algorithms

While this algorithm was based on it, the nearest neighbor algorithm would work for this case as well. It would be able to spot the nearest neighbor and travel to that location.

Another algorithm that would work for this would be implementing a shortest path algorithm. This way, you could Identify a path for the trucks to take depending on a weighted graph given with the distances of each route as the weights.

# F3a. Algorithm Differences

This algorithm is different from a nearest neighbor algorithm because it contains constraints due to needs from some packages. That makes this less efficient that a NN algorithm, but it accomplished the goal better than it would naturally.

The shortest path algorithm differs in that it requires a weighted graph as an input and would have to be altered to have a running list of visited locations on top of the visited locations local to a single point.

# G. Different Approach

If I were to do this project using one of the alternatives, I would use the nearest neighbor algorithm. I would change it so it would run in a more pure and proven form and work the constraints around it rather than hard coded inside of it. I would also design a more user-friendly interface with more UI options and a prettier front-end than the current terminal window.

# H. Verification of Data Structure

The chained hash table used fits perfectly into the use of this solution. Each record holding the packages works great to keep on the details grouped to similar items, but still kept separate (Lysecky & Vahid, 2018). The look-up times being constant makes this an obvious choice.

# H1. Other Data Structures

Other data structures that could be used would be an array or a binary search tree, or BST.

# H1a. Data Structure Differences

An array would be used similarly to the hash table in that it would store lists within it, but it would be less efficient as each insertion and deletion would cause slower results. It would also not be dynamically sized so it would require more maintenance.

A BST could be used for more efficient insertion and deletion than an array, but it would still be less efficient than a hash table while having a logarithmic operations case. BST would be self-balancing, but would require a high memory load to resolve collisions while re-balancing.

# I. Sources

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

Retrieved April 5, 2024, from https://learn.zybooks.com/zybook/WGUC950Template2023/chapter/6/section/2

# J. Professional Communication