C950 Task-1 WGUPS Algorithm Overview

(Task-1: The planning phase of the WGUPS Routing Program)

Michael Rufo

ID #010825128

WGU Email: mrufo2@wgu.edu

06/21/2025

C950 Data Structures and Algorithms II

# A. Algorithm Identification

# We can use a nearest neighbor algorithm to identify the next nearest delivery node to the previous node. Sorting algorithms will also be used to prioritize deliveries and for identifying shortest/longest routes.

# B. Data Structure Identification

A hash table would be an efficient data structure to store and retrieve package details.

# B1. Explanation of Data Structure

A hash table key could be used to uniquely identify a specific passage and the properties of the package, such as address, delivery status, weight, etc. can be stored as values. A second hash table can also be used efficiently pack trucks where multiple deliveries are going to the same address by utilizing a nested hash with an address to list of IDs relationship.

# C1. Algorithm’s Logic

The algorithm would start with creating package objects encapsulating the package details specifically address and unique handling.

Then the packages would be sorted by distance from with a duplicative index to associate discrete packages with the same address as another.

Then the packages would be distributed into trucks based on the delivery node and it’s nearness to the previously loaded delivery node. This would iterate until a truck is “full” based on aggregate miles traveled between nodes and root (the hub) or the number of packages on the truck, whichever comes first. A pre-lookup function can be utilized to evaluate the next-nearest node in the queue to see how many 0-distance neighbors (other packages with the same address) it shares so loading and travel time between nodes can be optimized by avoiding multiple deliveries to the same node.

Pseudo-Code:

Create Trucks (This creates a table of {Truck[Packages]} objects)

Instantiate a set of n trucks where each truck is an array of package objects.

Sort Packages (Storing the value of i indicates which packages can be loaded together.)

Instantiate an array of size j where j == length of PackageFile[]

For each package in PackageFile set i = 0

If the address of element in PackageFile at index i + 1 is equal to the address of the current package: i++

Loop until address !=

Store value of i.

Skip i iterations then set i back to 0.

Break when i > j

Create Packages (Instantiate package object details and add them to loading queue)

Instantiate package objects

For each package in PackageFile:

package = Package(populate package details)

append value of i

Return package

enqueue package

Load Packages (dequeue packages into trucks until queue is empty)

Let x = 0

For each package in queue let current index in queue = k

Load first package onto truck stack then for each subsequent package:

If package at index k has a value of i > 0:

dequeue packages k:k+i onto truck

Else check value of i at index k + 1:

If i > 0 then dequeue package at index k+1 else dequeue k, set x- -

Resume next iteration at index k – x

Repeat until queue is empty.

# C2. Development Environment

The program will be developed using a local git repository with VS Code and a virtual python environment. The repository will also contain a bash file for starting the environment, managing dependencies, and running the program. All of which will be written into a docker-compose.yaml file and uploaded to docker-hub to deploy as a container on AWS. The local development environment will use a Windows machine running 96GB of RAM and a 16 core CPU. The Python version in the local environment will be 3.12.

# C3. Space and Time complexity using Big-O notation

Time complexity for the algorithm will be O(n) in the worst case where the function is bounded on the size of the n due to requiring a constant number of iterations over the set of packages. Where one loop over n elements is required to sort array into groups of same-node packages. Then an additional loop over n elements to load them onto trucks. Because the algorithm efficiently groups the nodes prior to loading creates an O(n) and Ω (n) to sort. Then an additional iteration of <= n items to load the sorted packages creates an O(2n) or just O(n) on scale.

Space complexity will be similar due to the requirement of 2 hash tables where the first is indexed on address to create the groups of “sorted” packages and the second is indexed on package ID to distribute into trucks after enqueueing them into the loading queue.

Overall time complexity will scale with the size of the package set primarily at O(n).

The space complexity will also scale with the size of the package set and also be at O(n)

# C4. Scalability and Adaptability

Because the algorithm scales at O(n) complexity the algorithm will scale at a linear rate for sorting and queueing packages, but deliveries will optimize the route of each truck by preferencing grouped address, then nearest neighbors, queued on furthest distance from the hub.

# C5. Software Efficiency and Maintainability

The program utilizes git for version control and docker-hub for deployment on AWS containers. The local repository can be pushed to a remote depository on a bitbucket or github repository.

# C6. Self-Adjusting Data Structures

The hash-table is efficient for this use case because our algorithm relies on accessing values associated with a unique attribute of a package. This retrieval and storage method allows flexibility and lookup while sacrificing space efficiency.

The weakness of the hash-table is the potential for collision from inserting keys to existing locations. Thereby having to resolve the collision via methods like chaining which impacts the efficiency of insertion operations.

# C7. Data Key

Package ID is the ideal delivery key because it uniquely identifies each package where as other package attributes have potential duplicates.

# C8. Sources

Lysecky, R., & Vahid, F. (2018, June). *C950: Data Structures and Algorithms II*. zyBooks. Ch 2.3, Ch 6.1, Ch 6.2

Retrieved July 6, 2025, from <https://learn.zybooks.com/zybook/WGUC950AY20182019/>