# C950 Task-1 WGUPS Algorithm Overview

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

Alireza Barzegar

ID #010057350

WGU Email: abarzeg@wgu.edu

5/28/2024

C950 Data Structures and Algorithms II

## A. Algorithm Identification

A greedy algorithm will be used to create a program for delivering packages.

## B. Data Structure Identification

The program will use a hash table as the data structure for storing package data.

## B1. Explanation of Data Structure

a package class will be created and each bucket in the hash table will hold a package object that contains all the necessary information about a specific package as fields. The package ID will be used as the key for determining which bucket to store the object in. Searching for packages will be done frequently in the program and a hash table provides great searching speed.

## C1. Algorithm's Logic

```
INITIALIZE truck1 and truck2 objects
SET truck.current_location to hub for both trucks
SET  truck.packages to empty lists for both trucks
SET truck.total_miles to 0 for both trucks
INITIALIZE the hash table all_packages to hold all the 40 package objects
INITIALIZE sorted 2D list, distance_list, to hold a list of distances to all
other destinations from each one
INITIALIZE total variable to 0
WHILE total < 40:
        IF truck1.packages is empty AND truck2.packages is empty:
                SET truck.location to hub for both trucks
```

```
            IF truck1.total_miles > truck2.total_miles:
                ADD packages to truck2.packages up to 16
                ADD remaining packages to truck1.packages up to 16
                INCREMENT total by the number of added packages
            ENDIF
            ELSE:
                ADD packages to truck1.packages up to 16
                ADD remaining packages to truck2.packages up to 16
                INCREMENT total by the number of added packages
            ENDELSE
        ENDIF

        SORT truck.packages based on distance from truck.current_location
        ASCENDING using distance_list

        FOR each truck1 and truck2 :
            SET truck.current_location to the next package's
            package.destination on the list
            REMOVE package from truck.packages
            INCREMENT total by 1
        ENDFOR
    ENDWHILE
```

## C2. Development Environment

PyCharm IDE will be used for developing the program. The hardware specifications of the computer used for development will include an AMD Ryzen 9 CPU which has 8 cores,  and 16 GB of RAM.

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

The main data structure used in the program is a hash table, which has a space complexity of O(n). Searching, insertion, and deletion operations will be done on the hash table throughout the program and they all have a time complexity of O(n). Another major component of the program is a greedy algorithm used to determine the delivery route for each truck. The while loop in the algorithm has a time complexity equivalent to the number of packages to be delivered, so it's O(40) which simplifies to O(1). Each truck can only hold up to 16 packages so the sorting done inside the while loop will have a time complexity of O(16log16) which is O(1). The time complexity of the for loop inside is also O(1) because there are only 2 trucks so the for loop will always consist of 2 iterations. Therefore the time complexity will be O(1) if the number of packages to deliver always remains 40. If the number of packages to deliver each day differs, it's O(n) because of the while loop. The space complexity of the greedy algorithm will be O($n^2$) because of the 2d array used for distances.

## C4. Scalability and Adaptability

As the number of packages grows, the greedy algorithm's time complexity will be O(n) as explained in C3. O(n) is efficient so the algorithm will be scalable time-wise. On the other hand, a space complexity of O($n^2$) will not be scalable because as the number of packages increases, the resources required for storing distances increase drastically. To counter this, a more space-efficient data structure should be considered for storing distances instead of the 2d array.

## C5. Software Efficiency and Maintainability

When determining the overall efficiency of the software design, all major components of the program should be considered. These components include the greedy algorithm, the hash table for storing package data, and a simple frontend for inputting package information and displaying the route for each truck. The time complexities of the algorithm, hash table operations, and the front end are all o(n), which is an efficient time complexity. Many parts of the software can be easily written as modular functions, such as a function to handle sorting inside the algorithm or a function to handle user input validation. This, along with clear comments on code, makes it easier for developers to maintain the software by allowing them to narrow down code repairs and optimizations to specific functions.

## C6. Self-Adjusting Data Structures

The hash table can provide great search speeds for finding a specific package. While the worst-case time complexity of searching a hash table is O(n), searching a well-distributed hash table usually is the average-case time complexity, which is O(1). insertion and deletion in hash tables are also very efficient as only one bucket (array element) needs to be updated instead of shifting all elements. while these qualities make hash tables a great choice in many cases, there are also some disadvantages. collisions can greatly slow down operations on hash tables and hash functions that can greatly reduce collisions can be complex to implement. hash tables also do not store elements in any specific order, which makes them a bad choice in programs where the order of stored elements matters.

## C7. Data Key

package IDs will be used as keys for efficient delivery management. They are unique so each key will be associated with exactly 1 package ID, which makes them a good key.