C950 Task 2 Writeup

Xavier Loera Flores
ID: 011037676
xloeraf@wgu.edu
C950 Data Structures and Algorithms II

Algorithm Justification

The main core of the algorithm comes in two parts. The first part is the package loading into the trucks and the second part is the delivering of the packages using a nearest neighbor greedy algorithm.

Packages loaded into the trucks in a way that would allow the first truck to deliver high priority packages first. The second truck would then deliver the truck2 and bundled packages. The third truck would then deliver the delayed or wrong address packages. The remaining packages are distributed between truck 2 and truck 3 since the truck 1 driver needs to return to the hub to drive truck 3.

The package delivery part of the algorithm is a greedy algorithm. The algorithm will prioritize packages that have a deadline as its first working set and then move onto packages that don't have a deadline. The algorithm will find the closest package to the current location of the truck and deliver it. The algorithm will then find the next closest package and deliver it. This process will continue until all packages are delivered. The algorithm also takes into account the special package that needs to be delivered last since the delivery address needs to be corrected.

Algorithm Strengths

Strength 1

Greedy algorithms are simple to implement and understand. They make locally optimal choices that can create a result that is sufficiently efficient. The delivery routing program uses a greedy algorithm that optimizes for loading the priority packages across all trucks to be delivered on time and then for finding the next closest package and then delivering it.

Strength 2

Greedy algorithms are great for finding a suitable solution quickly rather than spending resources finding the best most optimal solution. The delivery routing program uses a nearest greedy

algorithm that optimizes for finding the next closest package and then delivering it rather than finding the quickest route to deliver all packages considering all possible routes.

Algorithm Verification

The algorithm was verified by running the program and checking the results against the requirements of the project. For each package, the program returns the delivery status, time if the package has been delivered, address, deadline, city, zip code, and weight. It also returns information about each truck including load, mileage, and return time if the truck has returned. The program was finally verified by running the program and checking the results against the CSV files.

Alternative Algorithm Solutions

Alternative Algorithm 1

The A* algorithm would be an excellent alternative to the greedy algorithm we used in our program. The A* algorithm is a path finding algorithm that uses a heuristic to find the best path to a goal. The A* algorithm differs from our greedy algorithm since its goal is to find the best path to deliver all packages.

Alternative Algorithm 2

Utilizing simulated annealing would be another alternative to the greedy algorithm we used in our program. Simulated annealing approximates the global optimum of a given function to find the best path by generating random potential solutions and then either accepting or rejecting. The simulated annealing algorithm differs from our greedy algorithm since its goal is to find the best path to deliver all packages.

Modifications To The Program Solution

I would make the following modifications to the program solution:

Firstly, I would rather have the system take in a parameter indicating which algorithm would be implemented. This would allow the system to compare the results of different algorithms. Rather than calling the delivery_algorithm(truck, time=None) function, I would call delivery_algorithm(truck, time=None, algorithm=nearest_neighbor) function to deliver the

packages. The algorithm parameter would be used to determine which algorithm to use. The algorithm parameter would default to the greedy algorithm if no algorithm is specified.

Next, I would also make the actual delivery algorithm more generic rather than have it specialized to our solution. The data that is passed into the delivery algorithm should be ready to use with a generic implementation of the algorithm. The same would go for other algorithms in the solution. The current implementation of the algorithm specifically calls methods from our truck and package classes. I would change the implementation to use generic data structures that can be used with any algorithm.

Lastly, I would implement a GUI to make the experience of using the program more intuitive. Our program presents all the results in the console. The GUI would allow the user to select the algorithm to use and then run the program. The GUI would also allow the user to view the results of the program.

Data Structure Verification

The delivery routing program uses a hash table data structure to store the packages. The hash table data structure is used to store the packages since it allows for constant time lookup of the packages. The program uses a weighted adjacency matrix or distance matrix to store the distances between different addresses in the program since it allows for constant time lookup of the distances between addresses. Both the hash table and distance matrix data structures were verified by running the program and checking the results against the requirements of the project. The hash table returns a delivery package object which contains the delivery status, delivery time if delivered, address object, and package object(which contains raw package data such as address, deadline, city, zip code, and weight).

Alternative Potential Data Structures

Data Structure 1

Instead of the adjacency matrix, the program could use a weighted adjacency list. The adjacency list would save on space at the cost of query time. Rather than storing vertices a V x V grid O(V^2), the adjacency list would store a list with vertices and edges of space O(V + E). The adjacency list would be a better choice if the program needed to save on space. However, query time would be slower since the program would need to iterate through the list to find the distance between two addresses with a speed of O(|V|) rather than having the constant O(1) lookup time of the adjacency matrix.

Data Structure 2

Alternatively to the hash table used to lookup packages, we can utilize a binary search tree. The binary search tree would avoid the need for collision resolution at the cost of query time. However, query time would be slower since the program would need to traverse the tree to find the package with a speed of O(log n) rather than having the constant O(1) lookup time of the hash table.

Sources

I did not use any external sources to complete this writeup.