C950 WGUPS Algorithm Overview

Zarif M Quamrul

ID #010319619

WGU Email: zquamr1@wgu.edu

Date: September 22nd, 2022

C950 Data Structures and Algorithms II

The purpose of this project is to find an efficient route to the daily task of delivering the packages to their designated addresses using an algorithm to pick an optimal route. The intent is to have only two out of three trucks loaded with packages at a time to deliver and to achieve to finish delivering all packages while keeping the total mileage under 140 miles. The algorithm implemented for this project is called the “Nearest neighbor” algorithm. It repeatedly calculates the shortest distance to a destination from the current position until all the packages have been iterated through.

Pseudocode:

Function organize\_and\_ship(Truck object):

For every ID in Truck object:  
 Search through the package hash table with the ID and  
 Add the package to the to\_be\_delivered list

Clear the package list in the truck object for a new organized list for optimal delivery

While(to\_be\_delivered is not empty):

For every package in to\_be\_delivered:

Find the closest destination by comparing the current position  
 and the destination distance of each package in the truck  
 and then travel to the closest destination while updating the truck  
 object data and package data to keep track of mileage and deliver

The project was developed using PyCharm v2022.2.1 as its primary IDE and using Python 3.10.6. The local machine used had a Ryzen 2600X processor and 16GB RAM. The program results in an average time complexity of O(n2) and a space complexity of O(n). Space-time complexities for major parts of the program are provided in in-line comments of the Pydocs. This program, as simple as it may be, is scalable and adaptable as adding more packages to the trucks to its maximum amount or adding more trucks would incur no penalties on the program or the algorithm.

Using a very efficient data structure as a hash table makes the program fetch data at a constant time which makes the program capable of handling a large number of inputs without major costs to resources. The program implements a list of lists to store data. The use of a hash table allows for mapping of package ids to the associate package data, which is then added to the appropriate bucket in the hash table which is selected via a hashing algorithm, allowing for quick retrieval of all relevant data using just an identifier. It is also easily maintainable due to its simplicity and use of Object-Oriented Programming. The data structure used for this project is a hash table or hash map. This is an extremely effective and efficient data structure when implemented correctly which results in an average of constant time for insertion, lookup and removal operations while handling all overhead with the chaining property of the structure. This is a very scalable and adaptable data structure which allows for addition of packages in the future will expand the spatial complexity linearly but have no effect on retrieval on information via the lookup operation. On the other hand, adding more trucks is also very much possible as the “organize and ship” method would only have to be called which will incur minimal penalties on the spatial complexity but virtually no effect on the lookup operation. Adding more cities to the delivery list will also effect the spatial complexity as more key-value pairs would be added to the hash table but without any effect to the speed and efficiency of the lookup operation. The only downside of using a hash table is the possibility of collisions in data which can be avoided if the hash table is implemented with an appropriate hashing function. Looking back, other data structures that would fit the requirements of this project are graphs and binary search trees. Using graphs would allow me to place similar packages together as adjacent vertices allowing me to traverse through the graph and pick optimal destinations for the trucks. Binary Search Trees on the other hand would allow me to have the packages presorted.

The chosen algorithm for this project is the “Nearest neighbor” algorithm which is a simple program which I think is its biggest strength. The simple implementation allows for the program to achieve its goal of delivering all packages by the end of the day under the required miles of 140. The algorithm further allows for future scalability with very minimal, if not any, adjustments to the program since adding more destinations to the program would incur no penalties on the functionality of the algorithm.

Looking back, some other possible algorithms that would have been able to accomplish the goals of this project are Dijkstra’s and the Greedy Algorithm. Using Dijkstra’s would result in a much more optimal route for the delivery of the packages and might make it a more appropriate fit for the project. The greedy algorithm on the other hand, would provide a much faster calculation of the route for delivery but at the cost of being optimal.

If I could start over with the project, I would implement a function to optimally load the packages into the trucks to presort them and find out which package should go into which truck, making the project much more scalable and allowing for all related functions to work much more efficiently.

Works Cited:

C950 Supplemental Resources by Western Governors University

Retrieved September 29, 2022, from https://westerngovernorsuniversity-my.sharepoint.com/personal/cemal\_tepe\_wgu\_edu/\_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fcemal%5Ftepe%5Fwgu%5Fedu%2FDocuments%2FMyDocs%2FC950%2FWebinar%2FW%2D3%5FChainingHashTable%5FzyBooks%5FKey%2DValue%5FCSV%5FGreedy%5FDijkstra%2Epy&parent=%2Fpersonal%2Fcemal%5Ftepe%5Fwgu%5Fedu%2FDocuments%2FMyDocs%2FC950%2FWebinar&ga=1