C950 WGUPS Algorithm Overview

Jared-Michael Torres

ID #000961916

WGU Email: jtor196@wgu.edu

10/05/2022

C950 Data Structures and Algorithms II

# Introduction

The following information provided will describe the algorithm and approach for a method of solving the traveling salesman problem within a set of confined parameters. Throughout the article you learn of the algorithms used as well as their weakness’ and strengths.

# A. Algorithm Identification

The main algorithm used in the Parcel Service program was a greedy approach to sorting and loading all three trucks. It was then followed by a delivery algorithm using the nearest neighbor method to travel to the next closest destination first. The algorithms do not go back and correct previous decisions. After a decision is made that will remain the choice.

# B1. Logic Comments

The first section of code filters the 40 packages onto the corresponding trucks to meet the delivery requirements.

Sort Packages to Truck1  
1.1 If deadline != and spec. instruct is not delayed on flight   
 assign to truck 1  
1.2 if package count < 16 and deliveryStatus is At the Hub and truckPackageAddress == PackageListAddress  
 assign to truck 1  
 if count == 16:  
 break  
   
Sort Packages to Truck2  
2.1 if delivery status == At the Hub and Spec. Instruct != empty  
 assign to truck 2  
2.2 if package count < 16 and deliveryStatus is At the Hub and truckPackageAddress == PackageListAddress  
 assign to truck 2  
 if count == 16:  
 break  
  
Sort Packages to Truck 3  
3.1 if delivery status == At the Hub  
 assign to truck 3  
 if package count == 16  
 break

The second section delivers the packages and tracks the delivery times.

Deliver Packages(departure time, truck)  
1.1 for x in truckPackages   
 if current location and truckPackage[x] location == shortest distance  
 deliver package  
 assign time delivered (distance traveled \* 60)/18 = minutes traveled  
 remove package from truck  
 departure time += minutes traveled

# B2. Development Environment

Processor: Intel(R) Core(TM) i7-5820K CPU @ 3.30GHz

Installed Ram: 32GB

System type 64-bit operating system, x64-based processor

Windows 10 Home

Version: 21H2

Pycharm Community Edition IDE 2022.2.1

Python version 3.10

# B3. Space-Time and Big-O

main.py time complexity  
def main() = O(n)  
main.py = O(n)  
  
chainingHashTable.py  
\_\_init\_\_() = O(n)  
insert() = O(n)  
search() = O(n)  
remove() = O(n)  
chainingHashtable.py = O(n)+O(n)+O(n)+O(n) = O(n)  
  
packageDelivery.py  
main() = O(n)  
loadPackageData() = O(n)  
loadDistanceData() = O(n^2)  
loadAddressData() = O(n^2)  
loadTruck1() = O(n^2)  
loadTruck2() = O(n^2)  
loadTruck3() = O(n^2)  
deliverPackages = O(n^2)  
timeStatus() = O(n)  
packageDelivery.py = O(n) + O(n) + O(n^2) + O(n^2) + O(n^2) + O(n^2) + O(n^2) + O(n^2) + O(n) = O(n^2)  
  
Package.py  
\_\_init\_\_() = O(n)  
Package.py = O(n)  
  
Truck.py  
\_\_init\_\_() = O(n)  
Truck.py = O(n)  
  
Parcel service project = O(n) + O(n) + O(n^2) + O(n) + O(n) = O(n^2)

The entire parcel service project runs in O(n^2) time complexity

# B4. Scalability and Adaptability

This program will easily be able to be modified for scalability, however the adaptability of the program is a concern. The program is dynamic in that it will be able to handle an increase in addresses or increase in the number of packages and be able to sort them if the csv files are formatted the same way every time. More trucks can be added if needed as well. The chaining hash table is also dynamic in the size it can grow and would not be a concern.

# B5. Software Efficiency and Maintainability

Object-oriented programming structure was used in the development of Parcel Service program.

All the major sections of the code are commented, and the Big O notation given for each individual function or class. The comments are clear and concise and a new developer working with the code will easily be able to determine what each section of code does and where changes can be made to scale up or scale down the program.

# B6. Self-Adjusting Data Structures

The chaining hash table used in the parcel service program has no limit to the number of packages that can be added. You can also scale up and down the amount of buckets used as needed for larger or smaller scales. The linked lists created in each bucket can grow infinitely (essentially). Searching, removing and inserting are all simple processes keeping the data structure operating in a O(n) space time complexity. A downside of the hast table is that the linked lists grow larger, and if you don’t include enough buckets in the data structure your linked lists can get so large that searching no longer will be efficient as it will take time to loop through the entire linked list in the bucket. You can minimize this by making the size of the array larger to increase the modulo number used to determine which bucket an entry goes in therefore shortening the lists for larger sets of data.

# C. Original Code

Verified

# C1. Identification Information

# C2. Process and Flow Comments

Process and Flow comments were added to all functions and classes.

# D. Data Structure

A chaining hash table was used. It can accept an unlimited number of packages without collision. The code does not need to be altered to accommodate more packages, however if the packages increase to a large number, the number of buckets may be adjusted to optimize search times. All package variable will be stored as a package object with the key, address, city, state, zipcode, deadline, weight, special instructions, of the package all being saved and stored. The packages will have 3 other variables assigned for sorting and tracking of delivery. A location ID was added for the corresponding location to search the distances between 2 locations and a delivery time and departure time.

# D1. Explanation of Data Structure

The data structure takes an object, variable, string, or anything you wish to store in it and a key to store the data in the correct bucket. My data structure has an array size of 10 and therefore has 10 buckets for storing. The bucket the data is stored in is determined by taking the modulo of the key and adding it to a linked list inside the bucket. If your key is the number 10, and your array size is 10, the modulo of 10 mod 10 will be 0 and the package with the key of 10 will be stored in bucket 0 of the array which has buckets 0-9.

# E. Hash Table

The hash table stores all given components of the packages with the package ID as the key. A package key and a package object are stored in the corresponding bucket associated with its key.

# F. Look-Up Function

The hash table has a search function that accepts the key of the packages and looks it up by finding the correct bucket and then searching the linked list within the bucket for the corresponding key.

# G. Interface

# 

# G1. First Status Check

# 

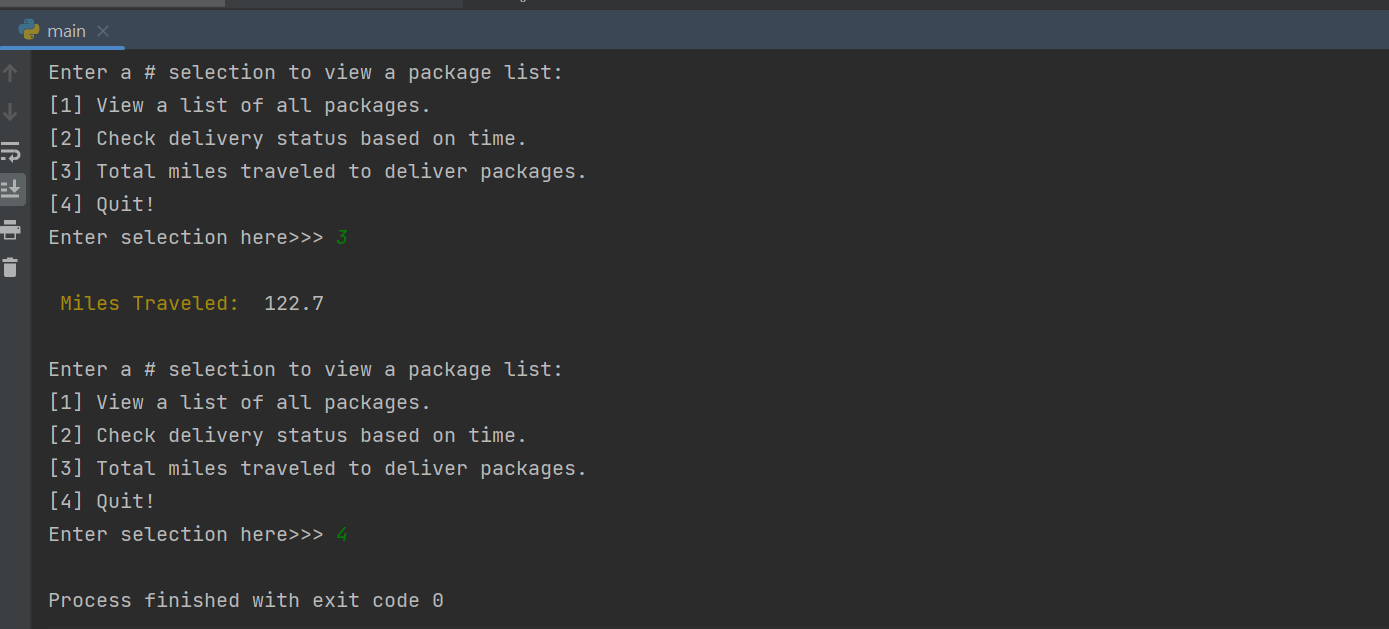
# G2. Second Status Check

# 

# G3. Third Status Check

# 

# H. Screenshots of Code Execution

Miles traveled execution w/ program quit

# List of all packages executing

# 

# The execution of packages based on delivery times can be seen on the status check code snips.

# I1. Strengths of Chosen Algorithm

The strengths of the algorithm I chose are that it is simple implement and understand. It runs in a time complexity of O(n^2) and will always run in that time complexity. It will also be easy to adapt into other uses.

The algorithm accurately tracks all packages and location distances between packages while being delivered. The packages all deliver in 122.7 miles which is under the requirements of 144 miles.

The algorithm can also make decisions and properly sort the packages into the correct truck based on the deadline, address, and special instructions. If any package data were to change the algorithm will adjust accordingly.

# I2. Verification of Algorithm

The algorithm meets all requirements.

The trucks deliver all packages in 122.7 miles

All the packages were delivered in the appropriate time frame

All packages were delivered according to their delivery specifications.

All of these can be checked using the UI built into the program. You can pull a list of the packages to be delivered as well as a list of the packages at a specific time and their location. You will also be able to print out the total miles traveled with one of the selections in the UI and then quit the program.

# I3. Other possible Algorithms

1. Dijkstra’s Algorithm
2. Genetic Algorithm

# I3A. Algorithm Differences

1. Dijkstra’s algorithm differs from the nearest neighbor algorithm in that it is a self adjusting algorithm that changes its previous selection based on total weight of the travel distance. In this case it would be based on total miles traveled. In other words Dijkstra would check all possible paths.
2. The Genetic Algorithm is a metaheuristic algorithm which means that it isn’t dependent on the problem and can be used for a wide range of optimization, unlike the heuristic algorithms which are customized for the current problem. A metaheuristic algorithm is more optimal than a greedy algorithm. (Bani-Hani, 2020)

# J. Different Approach

If I had to build this program again, I would approach loading the trucks with packages differently. If the number of trucks increases or the number of packages with more deadlines or special instructions the current method would not be efficient. Sorting the packages into separate hash tables based on special requirements and having each truck set to a limit to what types of packages from each separate hast table to optimize its travel time and guarantee deadlines are met still. I would also try to implement the Dijkstra algorithm to see if the total miles traveled would be further optimized.

# K1. Verification of Data Structure

The chaining hash table implemented in the project does meet all requirements. All packages delivered in 122.7 miles. All special instructions and deadline requirements were met.

# K1A. Efficiency

The chaining hash table is efficient in its linear search through the lists in the bucket for this program. The hash table runs in O(n) time complexity and the size of the data list is so small that the 10 assigned buckets in the hash table make it so each bucket only has 4 items to search through. If the search starts getting slower, we can easily increase the size of the hash table to reduce search times.

# K1B. Overhead

As previously noted, the chaining hash table is easily adjusted for larger data sets. If the linked lists within each bucket gets too large, you can just increase the number of buckets to reduce the number of items in each bucket to linear search through.

# K1C. Implications

Adding trucks or cities would not affect the look-up time or space used. The location ID’s I assigned to each package makes distance look-up happen in a time complexity of O(1) between 2 destinations. The trucks just store a list of packages, an assigned departure time and a returned to the hub time. Each truck being limited to a maximum of 16 packages will not increase search times when determining the nearest neighbor as no matter how many trucks there are, they will always have only 16 or fewer packages to search through.

# K2. Other Data Structures

1. Linked List
2. Weighted Graph

# K2a. Data Structure Differences

1. A linked list is exactly as it sounds. It is a list of all data linked together. A linked list would have worked in this type of project due to the small amount of data being imported. However, this isn’t a scalable data structure for larger sets of data. The search times would increase drastically the more data you add to the list. A search would potentially go through all 40 packages before finding the correct package, vs searching the chaining hash table it only needs to search through 4 packages per bucket based on the key provided.
2. A weighted graph could have been used in conjunction with Dijkstra’s algorithm to optimize distance traveled. The weighted graph would assign weights (distances) between each vertex(location) and Dijkstra’s would find the path with the lowest combined weight to complete. This would help optimize the distance traveled by the trucks.

# M. Professional Communication

Text goes here

# L. Sources - Works Cited

Dr. Cemal Tepe – “Lets Go Hashing” webinar -Retrieved October 5th, 2022, from WGU C950 Webinars

Bani-Hani, D. (2020). Genetic Algorithm (GA): A Simple and Intuitive Guide*.*

Retrieved from <https://towardsdatascience.com/genetic-algorithm-a-simple-and-intuitive-guide-51c04cc1f9ed>