C950 WGUPS Algorithm Overview

Kyle Christian

ID #000959459

WGU Email: kchr134@wgu.edu

10/03/2022

C950 Data Structures and Algorithms II

# Introduction

This project required that 40 packages be delivered to 26 different locations with certain requirements as to which trucks the packages could be on and when they needed to be delivered. I used a hash table to store the package data and used the nearest neighbor algorithm to achieve a moderately efficient system in which to deliver the packages and meet all the requirements.

# A. Algorithm Identification

The algorithm used in this program is the nearest neighbor algorithm.

# B1. Logic Comments

1. Set Current Location to Hub – Location 0 in the address table
2. Check the distance between current location and the address on each package on the truck
3. Update shortest distance
4. Take the location that has the shortest distance and update the current address to it
5. Remove package from truck
6. Update package status in hash table to delivered
7. Repeat steps 2-6 until truck is empty.

# B2. Development Environment

The development environment used to create this program was PyCharm Community

Edition 2022

Hardware: 12th gen Intel® Core™ i7-12600H, 2.7GHz

16G RAM

Windows 11

# B3. Space-Time and Big-O

The program runs in O(n^2).

Time complexities are found in the comments of the program with the highest time complexity being O(n^2) causing the program as a hole to run at worst, O(n^2)

# B4. Scalability and Adaptability

This program has the ability to take a different number of packages or add more trucks if necessary. It can adapt to the user’s needs without making huge changes to the code.

# B5. Software Efficiency and Maintainability

The program runs in polynomial time making it efficient. The program contains many comments that explain exactly what is happening, making it easy for the software to be maintained by others that aren’t familiar with the program.

# B6. Self-Adjusting Data Structures

The hash tables strengths are its ability to retrieve and insert with such a low time complexity.

The hash tables weakness is that when it contains more data, the chances for collisions go up.

The nearest neighbor algorithm’s strength is its ability to find a decently efficient way to solve the TSP problem with an easy learning curve

The nearest neighbor algorithm weakness is that with a larger dataset it wouldn’t work as well.

# C. Original Code

My code is original and runs without error

# C1. Identification Information

First line of code contains my identity

# C2. Process and Flow Comments

My comments are well written and explain thoroughly, what is happening in the program so that an average person may understand

# D. Data Structure

The data structure used was a chaining hash table

# D1. Explanation of Data Structure

A hash is a value that has a fixed length, and it is generated using a mathematical formula. In data indexing, hash values are used because they have fixed length size regardless of the values that were used to generate them. It makes hash values to occupy minimal space compared to other values of varying lengths.

A hash function employs a mathematical algorithm to convert the key into a hash. A collision occurs when a hash function produces the same hash value for more than one key.

The chaining hash table was chosen due to its efficiency in searching, inserting and retrieving information. My hash table stores the package information and can pull any package information using the key or package ID. Because it is a chaining hash table, it won’t have any issues with collision.

# E. Hash Table

Contains an insert function free from any use of dictionaries

# F. Look-Up Function

Looks up all data for a given package

# G. Interface

The interface used is a simple text user interface where the user is able to enter a number that corresponds to a menu option.

# G1. First Status Check

Text

Description automatically generated

# G2. Second Status Check

Text

Description automatically generated

# G3. Third Status Check

Text

Description automatically generated

# H. Screenshots of Code Execution

Graphical user interface, text

Description automatically generated

# I1. Strengths of Chosen Algorithm

The nearest neighbor algorithm is intuitive

Easy to implement and understand.

Can constantly evolve with new data

# I2. Verification of Algorithm

The algorithm successfully delivers all the packages by their required delivery times, meeting all the requirements. All 40 packages are delivered to the 26 locations VIA 3 trucks with a total of 117 miles, putting it under the maximum of 140.

# I3. Other possible Algorithms

The Dijkstra Algorithm is one algorithm I could have chosen. Another is the Christofides Algorithm

# I3A. Algorithm Differences

A huge difference between the Dijkstra algorithm and the Nearest Neighbor Algorithm is the fact that the Dijkstra Algorithm runs at near linear time where as the KNN (Nearest Neighbor) algorithm runs at O(n). Another difference is that the Dijkstra Algorithm goes back and relaxes the vertices as the algorithm finds shorter distances to that vertex. The KNN (Nearest Neighbor) algorithm find the shortest distances out of the neighbors and visits the closest one, then repeats. The Christofides Algorithm is said to achieve at worst 1.5 times the most optimal solution. The Christofides requires that a graph is created where the nearest neighbor does not, making it a less complicated solution. The Christofides Algo runs at O(n^3) time which is much slower than the KNN ((Nearest Neighbor).

# J. Different Approach

If I were going to try a different approach when coming at the same problem, I would create a graph and use the Dijkstra Algorithm in order to achieve a more efficient mileage and lower my total miles. I would also do a bit more research and figure out ways to lower my time complexity. I would also reorganize my load truck functions and optimize them to lower the total miles.

# K1. Verification of Data Structure

# K1A. Efficiency

The look-up function is unaltered by the number of packages.

# K1B. Overhead

Changing the number of packages would not affect the time complexity due to how the program is written.

# K1C. Implications

Changing the number of trucks would not change the time complexity at all, due to how the program is written. It would change the total miles, however.

# K2. Other Data Structures

An alternative data structure to use would be an Array. Another option would be a linked list.

# K2a. Data Structure Differences

The difference between the hash table (which I chose) and an array is the time it takes to search for an item. To find an item in an array, the program needs to iterate over each item in the array. With a hash table, it goes straight to the item. The difference between the linked list and the hash table is that the linked list requires pointers which adds extra space. Pointers are not used in python which made a linked list not the way to go.

# M. Professional Communication

Information is portrayed professionally

# L. Sources - Works Cited

Hash table referenced from - *C950 supplemental resources let’s go hashing my.wgu.edu* 10/04/22