·         [MY WGU](http://my.wgu.edu/)

* Uri Williams Easter

Ask for Help Print

NHP2 — NHP2 TASK 1: WGUPS ROUTINGz PROGRAM

**DATA STRUCTURES AND ALGORITHMS II — C950**

**PRFA — NHP2**

TASK OVERVIEWSUBMISSIONSEVALUATION REPORT

COMPETENCIES

**4048.5.1** : **Non-Linear Data**

The graduate creates software applications that incorporate non-linear data structures for efficient and maintainable software.

**4048.5.2** : **Hashing Algorithms and Structures**

The graduate writes code using hashing techniques within an application to perform searching operations.

**4048.5.3** : **Dictionaries and Sets**

The graduate incorporates dictionaries and sets in order to organize data into key-value pairs.

**4048.5.4** : **Self-Adjusting Data Structures**

The graduate evaluates the space and time complexity of self-adjusting data structures using big-O notation to improve the performance of applications.

**4048.5.5** : **Self-Adjusting Heuristics**

The graduate writes code using self-adjusting heuristics to improve the performance of applications.

**4048.5.6** : **NP-Completeness and Turing Machines**

The graduate evaluates computational complexity theories in order to apply models to specific scenarios.

INTRODUCTION

For this assessment, you will apply the algorithms and data structures you studied in this course to solve a real programming problem. You will also implement an algorithm to route delivery trucks that will allow you to meet all delivery constraints while traveling under 140 miles. You will then describe and justify the decisions you made while creating this program.

The skills you showcase in your completed project may be useful in responding to technical interview questions for future employment. This project may also be added to your portfolio to show to future employers.

SCENARIO

The Western Governors University Parcel Service (WGUPS) needs to determine an efficient route and delivery distribution for their Daily Local Deliveries (DLD) because packages are not currently being consistently delivered by their promised deadline. The Salt Lake City DLD route has three trucks, two drivers, and an average of 40 packages to deliver each day. Each package has specific criteria and delivery requirements.

Your task is to determine an algorithm, write code, and present a solution where all 40 packages (listed in the attached “WGUPS Package File”) will be delivered on time while meeting each package’s requirements and keeping the combined total distance traveled under 140 miles for both trucks. The specific delivery locations are shown on the attached “Salt Lake City Downtown Map,” and distances to each location are given in the attached “WGUPS Distance Table.” The intent is to use the program for this specific location and also for many other cities in each state where WGU has a presence*.* As such, you will need to include detailed comments to make your code easy to follow and to justify the decisions you made while writing your scripts.

Keep in mind that the supervisor should be able to see, at assigned points, the progress of each truck and its packages by any of the variables listed in the “WGUPS Package File,” including what has been delivered and at what time the delivery occurred.

ASSUMPTIONS

•   Each truck can carry a maximum of 16 packages, and the ID number of each package is unique.

•   The trucks travel at an average speed of 18 miles per hour and have an infinite amount of gas with no need to stop.

•   There are no collisions.

•   Three trucks and two drivers are available for deliveries. Each driver stays with the same truck as long as that truck is in service.

•   Drivers leave the hub no earlier than 8:00 a.m., with the truck loaded, and can return to the hub for packages if needed.

•   The delivery and loading times are instantaneous, i.e., no time passes while at a delivery or when moving packages to a truck at the hub (that time is factored into the calculation of the average speed of the trucks).

•   There is up to one special note associated with a package.

•   The delivery address for package #9, *Third District Juvenile Court*, is wrong and will be corrected at 10:20 a.m. WGUPS is aware that the address is incorrect and will be updated at 10:20 a.m. However, WGUPS does not know the correct address (410 S State St., Salt Lake City, UT 84111) until 10:20 a.m.

•   The distances provided in the WGUPS Distance Table are equal regardless of the direction traveled.

•   The day ends when all 40 packages have been delivered.

REQUIREMENTS

*Your submission must be your original work. No more than a combined total of 30% of the submission and no more than a 10% match to any one individual source can be directly quoted or closely paraphrased from sources, even if cited correctly. The similarity report that is provided when you submit your task can be used as a guide.*  
  
*You must use the rubric to direct the creation of your submission because it provides detailed criteria that will be used to evaluate your work. Each requirement below may be evaluated by more than one rubric aspect. The rubric aspect titles may contain hyperlinks to relevant portions of the course.*

*Tasks may****not****be submitted as cloud links, such as links to Google Docs, Google Slides, OneDrive, etc., unless specified in the task requirements. All other submissions must be file types that are uploaded and submitted as attachments (e.g., .docx, .pdf, .ppt).*

A.  Identify a named self-adjusting algorithm (e.g., “Nearest Neighbor algorithm,” “Greedy algorithm”) that you used to create your program to deliver the packages.  
What is the algorithm chosen?

The Nearest Neighbor Algorithm was elected in this program as the solution for this traveling salesman problem. Nearest Neighbor finds the minimum distance between a grouping of points and maps to the closest proximal location using the smallest distance identified. This algorithm was one of the best choices given this problem since it serves the purpose of mapping an efficient path in terms of distance. This suggests that it will find one of the most optimized paths for delivering all packages quickly, and it will most likely stay within the limit for mileage for the day.

B.  Write an overview of your program, in which you do the following:

1.  Explain the algorithm’s logic using pseudocode.

*Note: You may refer to the attached “Sample Core Algorithm Overview” to complete part B1. –email*

Truck\_list = [list of package ids]

Truck\_current\_location = 0

Minimum\_distance = some big number

next\_package = None

While truck\_list: //will keep going while truck\_list is not empty

For pid in truck\_list:

          Get package from hash table using pid

          Current\_package\_location = address\_dict[package.address]

          Distance = distance\_list[truck\_current\_location][current\_package\_location]

          If distance < minimum\_distance:

                    Reset minimum\_distance and next\_package to current distance and package

//queues the package to be delivered that’s closest to the current truck location

truck.time += datetime.timedelta(hours=next\_address / 18)

print(str(truck.truck\_name) + " TIME: " + str(truck.time) + ", DISTANCE: " + str(truck.tot\_miles) + "\n" +

              str(next\_pkg) + "\n")

Move the truck

Pop off the id of the package that was just delivered from truck\_list

2.  Describe the programming environment you used to create the Python application.

# Pycharm 2023.2.2,

# Python 3.10.7

3.  Evaluate the space-time complexity of each major segment of the program, and the entire program, using big-O notation.

<https://www.geeksforgeeks.org/how-to-analyse-loops-for-complexity-analysis-of-algorithms/>

4.  Explain the capability of your solution to scale and adapt to a growing number of packages.

For a growing number of packages, we could first increase the number of trucks to deliver more packages, and if they have a different maximum, we could reset the value for the attribute of pkg\_max to a higher number to deliver more packages. Further actions we could take are implementing an add package function to allow users to add packages and increasing the hashindex (after clearing the hashmap) relative to the number of growing packages.

One of the difficult things to account for scaling up is that more addresses means we would need to switch from adjacency matrix as that isn't maintainable when you have large numbers of addresses (greater than fifty or one hundred).We could keep writing data to a csv file, but in reality, for industry we may want to connect to a database like SQL using sqllite library in python or MongoDB using a pymongo driver, depending on the  app design and preference

If the trucks were allowed to go faster, similar to average speed in real life, that may also hasten the number of deliveries.

speed of truck constraints  
<https://www.up.com/customers/track-record/tr081319-truck-pros-cons.htm#:~:text=Trucks%20travel%20at%20an%20average,and%20cost%2Deffective%20shipping%20solution>.

chance of collisions with hashmap  
<https://www.geeksforgeeks.org/hash-map-in-python/>

need an algorithm to predict which truck should load which packages. Nearest neighbor predicts nearest locations to each other, but given the complexity of constraints in this problem, for massive amounts of packages, and existing package limits, a more advanced prediction tool may be needed to allot a number of packages to the correct trucks.  
Inconveniences  
I knew the inconveniences because they were project requisites, in real life people most likely will not know many of the inconveniences faced ahead of time, nor will packages be delivered instantaneously and therefore delays in predicted delivery times will be slower.

5.  Discuss why the software is efficient and easy to maintain.

<https://realpython.com/python-pep8/#naming-styles>

The time complexity of the entire program O(n^2) by worst case scenario. Other functions within the program function at O(1), or O(n) time, which suggests that on the whole this program is efficient especially since it does not run in O(n!) time or O(2^n).

Further, I commented extensively on each major block of code, followed naming conventions, and employed functional programming practices, which means if a new developer needed to understand or revise my code, they should in theory be able to do so with ease. Lastly, but not least, I also pasted sources that helped me derive some of the code solutions to the traveling salesman problem. The sources give insight to conclusions about the efficiency of the how the code operates, and the logic that has been adapted to the expectations of this program.

6.  Discuss the strengths and weaknesses of the self-adjusting data structures (e.g., the hash table).  
One of the strengths of the HashMap is the O(1) time complexity for object access time.

<https://www.geeksforgeeks.org/hash-map-in-python/>

Another one of the benefits of using the HashMap data structure is that values map sequentially from the order set by the hash index.

<https://www.geeksforgeeks.org/hash-map-in-python/>

One of the weaknesses I observed is that it is difficult to get the size of all objects within the HashMap for scaling without a list or for loop since a HashMap does not "maintain the same order of items in a collection". There can be many values per index, and there appears to be no way of getting the count of items other than directly calling the number of items as a parameter from the lookup function, or through iteration rather then using len() or size() functions to measure the amount of values in the HashMap.

<https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.baeldung.com/java-arraylist-vs-linkedlist-vs-hashmap%23:~:text%3DUsing%2520HashMap%2520makes%2520sense%2520only,of%2520items%2520in%2520a%2520collection.&ved=2ahUKEwip0KzRiOuBAxWCkmoFHdA5DbAQFnoECBYQBQ&usg=AOvVaw1utJidxRmU0va23A2E8lsf>

Another one of the weaknesses of HashMap is that a greater number of keys increases the chances of collision. When there are fairly limited amount of keys, operation stays at O(1) time. As the keys increase, and potentially the hash index, it is possible for the HashMap to start processing closer to O(n) time if the key-value ratio begins to get overloaded.  
<https://stackoverflow.com/questions/6924852/what-are-the-disadvantages-to-hashmaps>

<https://www.geeksforgeeks.org/hash-map-in-python/>

C.  Write an original program to deliver *all* the packages, meeting *all* requirements, using the attached supporting documents “Salt Lake City Downtown Map,” “WGUPS Distance Table,” and the “WGUPS Package File.”

1.  Create an identifying comment within the first line of a file named “main.py” that includes your first name, last name, and student ID.

2.  Include comments in your code to explain the process and the flow of the program.

--object instantiation from existing files (like csv), classes, and libraries (time e.g.)

--major algorithms that use the objects to generate data important to the requirements

--major UI functions that enable the user to evaluate the output of the major algorithms in a variety of contexts.

D.  Identify a self-adjusting data structure, such as a hash table, that can be used with the algorithm identified in part A to store the package data.

The data structure used for this program to facilitate the access needs of the nearest neighbor algorithm a chaining hash table/ hash map which is a form hash map. The book for this course defines a hash table as "..a data structure that stores unordered items by mapping (or hashing) each item to a location in an array (or vector)…the modulo (remainder) operator can be used to map[to].. a bucket index from the item's key".  
  
The chaining hash map differs from the hash map data structure in that it has a method of mapping a list of values to a bucket for insertion and search from the key, sometimes referred to as the hash index. Still, as the Chaining HashMap is a form of HashMap it still shares the benefits of O(1) time complexity which is faster than the O(n) time complexity of linear search, and the reduced space complexity of the Chaining HashMap means that when a bucket is identified less time is executed per search for fewer items as compared to having to search an entire list for an item in a linear search.

Explain how your data structure accounts for the relationship between the data points you are storing.

In the HashMap a list of packages is designated to a bucket using a modulus of ten on the package id to ascertain their location. In the nearest neighbor algorithm, we use the attribute of the package id from the Truck’s list of package id’s known as package\_load. This enables us to search for a package object within the HashMap using package id, and add it to a list of package objects.

   Then while the list of package objects is greater than zero, we use the calc\_distance function to compare the address indices of the truck and packages to the nearest possible address, then we assign that address as the next address.  
The HashMap here plays a pivotal role in allowing the program to go from taking the id of the package, to looking up the package object so the addresses can be compared using the address\_index and calc\_distance function.

E.  Develop a hash table, without using *any* additional libraries or classes, that has an insertion function that takes the following components as input and inserts the components into the hash table:

•   package ID number

•   delivery address

•   delivery deadline

•   delivery city

•   delivery zip code

•   package weight

•   delivery status (e.g., delivered, en route)

F.  Develop a look-up function that takes the following components as input and returns the corresponding data elements:

•   package ID number

•   delivery address

•   delivery deadline

•   delivery city

•   delivery zip code

•   package weight

•   delivery status (i.e., “at the hub,” “en route,” or “delivered”), including the delivery time

*Note: Your function should output all data elements for the package ID number.*

G.  Provide an interface for the user to view the status and info (as listed in part F) of *any* package at *any* time, and the total mileage traveled by *all* trucks. (The delivery status should report the package as *at the hub*, *en route*, or *delivered*. Delivery status *must* include the time.)

1.  Provide screenshots to show the status of *all* packages at a time between 8:35 a.m. and 9:25 a.m.

2.  Provide screenshots to show the status of *all* packages at a time between 9:35 a.m. and 10:25 a.m.

3.  Provide screenshots to show the status of *all* packages at a time between 12:03 p.m. and 1:12 p.m.

H.  Provide a screenshot or screenshots showing successful completion of the code, free from runtime errors or warnings, that includes the total mileage traveled by *all* trucks.

I.  Justify the core algorithm you identified in part A and used in the solution by doing the following:

1.  Describe *at least***two** strengths of the algorithm used in the solution.

Nearest Neighbor  
No training step (knn does a training step)  
<https://quantdare.com/10-reasons-for-loving-nearest-neighbors-algorithm/>

Point pattern analysis model to find closest points nearest each other in space.  
<https://epgp.inflibnet.ac.in/epgpdata/uploads/epgp_content/S000017GE/P001787/M031066/ET/1527502623NNA_text(Final(1.pdf>

2.  Verify that the algorithm used in the solution meets *all* requirements in the scenario.

Provide the total combined miles traveled by all trucks. It must be less than 140.

● State that all packages were delivered on time.

● State that all packages were delivered according to their delivery specifications.

● Describe how all the above points are verifiable through the user interface

•   Each truck can carry a maximum of 16 packages, and the ID number of each package is unique. The truck can carry a up to 16 packages since it’s representation of the truck object has its attribute of pkg\_max is set to sixteen. The pkg\_max attribute maintains the maximum number of packages until manually altered. To maintain the uniqueness of id, each id is entered individually into one of two lists per truck object in accordance with a planned route associated with the list the id is sorted into.

•   The trucks travel at an average speed of 18 miles per hour and have an infinite amount of gas with no need to stop.

To ensure that the trucks travel at an average speed of eighteen, the truck objects have the value of avg. mph set to eighteen once the objects are instantiated. Representing a limitless supply of gas originated from not setting any variable for gas supply. Expressing no need for the trucks to stop in this program was the result of not developing any conditional logic that emulates driving rules which would require the trucks to stop. Not developing any conditional logic that emulates driving rules also negates the ability for collisions.

•   Three trucks and two drivers are available for deliveries. Each driver stays with the same truck as long as that truck is in service.  
Assuming there is one driver per truck, and that each driver stays with the same track as long as it is in service, the minimum requirement in this instance is simply that each driver has a truck. Therefore only two truck objects were instantiated in this program and their combined total distance 124.7 miles meets the requirement of being under 140 miles. Between both sets of delivery routes executed by the trucks all 40 packages were delivered.

•   Drivers leave the hub no earlier than 8:00 a.m., with the truck loaded, and can return to the hub for packages if needed.

•   The delivery and loading times are instantaneous, i.e., no time passes while at a delivery or when moving packages to a truck at the hub (that time is factored into the calculation of the average speed of the trucks).

•   There is up to one special note associated with a package.

•   The delivery address for package #9, *Third District Juvenile Court*, is wrong and will be corrected at 10:20 a.m. WGUPS is aware that the address is incorrect and will be updated at 10:20 a.m. However, WGUPS does not know the correct address (410 S State St., Salt Lake City, UT 84111) until 10:20 a.m.

•   The distances provided in the WGUPS Distance Table are equal regardless of the direction traveled.

Answer with function processes and screenshots

3.  Identify **two** other named algorithms, different from the algorithm implemented in the solution, that would meet the requirements in the scenario.

a.  Describe how *each* algorithm identified in part I3 is different from the algorithm used in the solution.

Christofides Algorithm

<https://www.youtube.com/watch?v=ayIsRZAGyi4>, <https://www.youtube.com/watch?v=GiDsjIBOVoA&t=748s>

Genetic algorithm

J.  Describe what you would do differently, other than the two algorithms identified in I3, if you did this project again.  
  
  
If this project were to be an industry facing solution to scale for over 4000 or more, there are in fact a few changes I would make. The first is that I would add functions for the user to interact with where they could insert packages into the hashmap, or delete as needed if possible. My observations with big companies such as Fedex or Amazon is not that they merely load 4000 entries of data at a time, but rather that any person who needs their services can submit a shipment order from wherever they are. Therefore, with an add button, that could help scale this project beyond its current capacity, and likewise for the delete, so users could undo unintentional orders. Further, the greater the amount of orders, I would try to dispatch more trucks. I would also see if there’s an api I could pull from to continue the adjacency matrix for more locations, or see if another geolocation api could remedy the goal of working with shortest distances.  
Given a chance to do this again, I would also do more research into how to implement a Tkinter gui with this project. I was most of the way there, but I got stuck on how to display more rows into one of my treeviews.  
Finally, for the intended scale, I would set up a database. Most likely in SQL for value binding in queries and search.

K.  Justify the data structure you identified in part D by doing the following:

1.  Verify that the data structure used in the solution meets *all* requirements in the scenario.  
  
To verify all attributes of the packages and statuses that pertain to them, the user can select option eight in the user interface and verify the accuracy of the information as they please by selecting option eight after selecting any of the options from option four to option seven. Selecting options two or three after any of the options from four to seven also demonstrates how quickly the hash map can retrieve current information on the statuses of all and any package of choosing.  
A user could also check the intended results of the program by selecting option one “Check Full Delivery Cycle”. Option one models the intended use of the program by executing the call to each option of option four to option seven once, which represents each truck traveling both routes to deliver all packages according to their delivery requirements in timely fashion. The “Check Full Delivery Cycle” also calls option eight to report that the truck’s total mileage is 124.69 which is still beneath the 140 miles requirement.

a.  Explain how the time needed to complete the look-up function is affected by changes in the number of packages to be delivered.  
not affected timewise, but space-wise, collisions can occur if one is not careful,  
and as discussed earlier, if the key-to-value ratio is overloaded then the time may begin to operate closer to O(n) time complexity rather than O(1).

--ids from packages  
--more items to search per index which slows search time, since searching through the value list is more like a linear sort  
<https://stackoverflow.com/questions/9849633/hash-table-vs-linear-list>  
<https://stackoverflow.com/questions/6924852/what-are-the-disadvantages-to-hashmaps>

b.  Explain how the data structure space usage is affected by changes in the number of packages to be delivered.  
more buckets per hash\_index

c.  Describe how changes to the number of trucks or the number of cities would affect the look-up time and the space usage of the data structure.  
--number of truck to package ratio could speed up the lookup time  
--greater number of cities could slow down lookup time in comparison of address index for calc distance. Less cities with greater distances could increase the distances the trucks travel, so the efficiency of the lookup function is determined by distance as well as number of cities.

2.  Identify **two** other data structures that could meet the same requirements in the scenario.

--dictionary

(dictionary is to list as hashmap is to linked list)  
(both come from map and implement key value stores, but hashmap uses index)

<https://www.reddit.com/r/learnpython/comments/s8oz0f/hash_maps_vs_dictionaries/>  
<https://stackoverflow.com/questions/2061222/what-is-the-true-difference-between-a-dictionary-and-a-hash-table>  
--list

a.  Describe how *each* data structure identified in part K2 is different from the data structure used in the solution.

--chaining hashmap can store multiple objects per index, no known length f(x)  
--dictionary maps key to value pairs, so it can be a little more difficult extracting object properties from the value when loaded from csv  
--list can store list of attributes per index, but has a greater number of indices to traverse as opposed to values per index.

L.  Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

M.  Demonstrate professional communication in the content and presentation of your submission.

A, D, I,K, J. L

F, B,H

**File Restrictions**

File name may contain only letters, numbers, spaces, and these symbols: ! - \_ . \* ' ( )  
File size limit: 200 MB  
File types allowed: doc, docx, rtf, xls, xlsx, ppt, pptx, odt, pdf, txt, qt, mov, mpg, avi, mp3, wav, mp4, wma, flv, asf, mpeg, wmv, m4v, svg, tif, tiff, jpeg, jpg, gif, png, zip, rar, tar, 7z