NHP2 — NHP2 TASK 1: WGUPS ROUTING PROGRAM

DATA STRUCTURES AND ALGORITHMS II — C950

PRFA — NHP2

TASK OVERVIEW | SUBMISSIONS | EVALUATION 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. An originality report is provided when you submit your task that 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.

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.

   2. Describe the programming environment you used to create the Python application.
   3. Evaluate the space-time complexity of each major segment of the program, and the entire program, using big-O notation.
   4. Explain the capability of your solution to scale and adapt to a growing number of packages.
   5. Discuss why the software is efficient and easy to maintain.
   6. Discuss the strengths and weaknesses of the self-adjusting data structures (e.g., the hash table).

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.

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

   Note: Use only *appropriate built-in data structures, except dictionaries. You must design, write, implement, and debug* all *code that you turn in for this assessment. Code downloaded from the Internet or acquired from another student or* any *other source may not be submitted and will result in automatic failure of this assessment.*

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

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

J. Describe what you would do differently, other than the two algorithms identified in I3, if you did this project again.

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.
      a. Explain how the time needed to complete the look-up function is affected by changes in the number of packages to be delivered.
      b. Explain how the data structure space usage is affected by changes in the number of packages to be delivered.
      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.

2. Identify **two** other data structures that could meet the same requirements in the scenario.
   a. Describe how *each* data structure identified in part K2 is different from the data structure used in the solution.

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.

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

RUBRIC

A: ALGORITHM IDENTIFICATION

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A named algorithm is not identified.	The identified named self-adjusting algorithm does not perform the task of delivering the packages or does not perform the task.	The identified named self-adjusting algorithm used to create the program performs the task of delivering *all* packages.

B1: LOGIC COMMENTS

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
Pseudocode is not provided.	The submission provides pseudocode but does not explain the algorithm's logic, or it contains inaccuracies.	The submission explains the algorithm's logic using pseudocode, and the algorithm's logic contains *no* inaccuracies.

B2: DEVELOPMENT ENVIRONMENT

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A description is not provided.	The submission does not accurately describe the programming environment used to create the Python application. Or it includes either the software or hardware used in creating the program, but not *both*.	The submission accurately describes the programming environment used to create the Python application, including *both* the software and hardware used to create the program.

B3: SPACE-TIME AND BIG-O

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
An evaluation of space-time complexity is not provided.	The evaluation shows the space-time complexity of either major segments	The evaluation accurately shows the space-time complexity of *each* major

	of the program or the entire program, but not *both*, using big-O notation. Or the evaluation addresses space or time, but not *both*. Or the evaluation contains inaccuracies.	segment of the program, and the entire program, using big-O notation.

B4: SCALABILITY AND ADAPTABILITY

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The capability of the solution to scale and adapt to a changing market is not explained.	The capability of the solution to scale and adapt is explained, but the explanation is illogical, or it does not include the algorithm's ability to scale to the number of packages. Or the explanation contains inaccuracies.	The capability of the solution to scale and adapt is logically and accurately explained, including the algorithm's ability to scale to the number of packages.

B5: SOFTWARE EFFICIENCY AND MAINTAINABILITY

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
Neither the efficiency nor the maintainability of the software is discussed.	The discussion addresses either the efficiency or the maintainability of the software, but not *both*. Or the discussion contains inaccuracies.	The discussion accurately addresses why the software is efficient and easy to maintain.

B6: SELF-ADJUSTING DATA STRUCTURES

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
Neither the strengths nor the weaknesses of the self-adjusting data structures are discussed.	The discussion addresses either the strengths or the weaknesses of the self-adjusting data structures, but not *both*. Or the discussion contains inaccuracies.	The discussion accurately addresses *both* the strengths and weaknesses of the self-adjusting data structures and the hash table.

C: ORIGINAL CODE

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
Code is not provided. Or the code is not functional.	The code is original but does not run without errors or warnings, or the code is not original.	The code is original and runs without errors or warnings.

C1: IDENTIFICATION INFORMATION

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT

An identifying comment is not provided.	The identifying comment is missing from the first line of a file named "main.py," the file is not named "main.py," or the comment is missing the candidate's first name, last name, or student ID, or 1 or more of these elements is incorrect.	An identifying comment is located within the first line of a file named "main.py" that includes the candidate's correct first name, last name, and student ID.

C2: PROCESS AND FLOW COMMENTS

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
Comments to explain the process and the flow are not provided.	Comments are provided separately from the code, or the comments do not adequately or accurately explain the process and the flow of the program.	Comments are provided within the code that adequately and accurately explain the process and the flow of the program.

D: DATA STRUCTURE

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The submission does not identify a data structure.	The submission identifies a data structure, but the data structure is not self-adjusting, or it cannot store the package data. Or it does not perform well with the algorithm in part A.	The submission identifies a self-adjusting data structure that performs well with the algorithm in part A and can store the package data.

D1: EXPLANATION OF DATA STRUCTURE

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The submission does not explain the data structure.	The submission explains the data structure but does not explain the relationship between the data points to be stored. Or the explanation contains inaccuracies.	The submission accurately explains the data structure and how that data structure accounts for the relationship between the data points to be stored.

E: HASH TABLE

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A hash table is not provided.	The hash table does not have an insertion function or has an insertion function that includes additional libraries or classes or that does not account for *all* of the given components. Or the table contains errors.	The hash table has an insertion function, without using *any* additional libraries or classes, that is free from errors and includes, as input, *all* of the given components.

F: LOOK-UP FUNCTION

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A look-up function is not provided.	The look-up function does not include *all* of the given data elements, or it does not complete searches or return the listed data. Or the look-up function does not complete or completes with run-time errors.	The look-up function includes *all* of the given data elements, and it completes searches and returns the listed data. The look-up function completes without run-time errors.

G:INTERFACE

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
An interface is not provided.	The interface provides an intuitive means for either determining the total mileage traveled by *all* trucks or for accessing package delivery status as required, but not *both*. Or the status check does not include the time.	The interface provides an intuitive means of *both* determining the total mileage traveled by *all* trucks and for accessing package delivery status as required and includes the time of the status check.

G1:FIRST STATUS CHECK

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A screenshot is not provided.	The screenshots show an incomplete list of the packages that are loaded on *each* truck, or they do not show the status of *each* package at a time between 8:35 a.m. and 9:25 a.m.	The screenshots show a list of *all* packages that are loaded on *each* truck and the status of *each* package at a time between 8:35 a.m. and 9:25 a.m.

G2:SECOND STATUS CHECK

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A screenshot is not provided.	The screenshots show an incomplete list of the packages that are loaded on *each* truck, or they do not show the status of *each* package at a time between 9:35 a.m. and 10:25 a.m.	The screenshots show a list of *all* packages that are loaded on *each* truck and the status of *each* package at a time between 9:35 a.m. and 10:25 a.m.

G3:THIRD STATUS CHECK

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A screenshot is not provided.	The screenshots show an incomplete list of the packages that are loaded on *each* truck, or they do not show the status of *each* package at a time between 12:03 p.m. and 1:12 p.m.	The screenshots show a list of *all* packages that are loaded on *each* truck and the status of *each* package at a time between 12:03 p.m. and 1:12 p.m.

H:SCREENSHOTS OF CODE EXECUTION

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
Screenshots are not provided.	The screenshots capture an incomplete execution of the code, or the screenshot does not capture the total delivery time.	The screenshots capture a complete execution of the code and include the total delivery mileage.

I1: STRENGTHS OF THE CHOSEN ALGORITHM

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A description is not provided.	*At least* 2 strengths of the algorithm identified in part A are described, but 1 or *both* of the strengths are inaccurate or do not relate to the scenario.	*At least* 2 strengths specific to the algorithm identified in part A are accurately described, and *both* strengths apply to the scenario.

I2: VERIFICATION OF ALGORITHM

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The submission does not verify the algorithm used in the solution.	The submission does not verify the algorithm used in the solution meets *all* requirements. The verification is missing the total miles added to *all* trucks, or the total combined delivery distance is more than 140 miles, or it does not state that *all* packages were delivered on time.	The submission verifies the algorithm used in the solution meets *all* requirements, and the verification includes the total miles added to *all* trucks, and the total combined delivery distance is less than 140 miles, and it states that *all* packages were delivered on time.

I3: OTHER POSSIBLE ALGORITHMS

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The submission does not identify 2 other algorithms.	The submission identifies 2 algorithms different from the one used in the solution, but 1 or *both* algorithms do not meet the requirements in the scenario.	The submission identifies 2 algorithms different from the one used in the solution, and *both* algorithms meet the requirements in the scenario.

I3A: ALGORITHM DIFFERENCES

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The submission does not describe the differences between algorithms.	The submission does not describe attributes for *each* algorithm identified in part I3, or the description does not compare these attributes to the attributes of the algorithm used in the solution. Or the description contains inaccuracies.	The submission accurately describes attributes of *each* algorithm identified in part I3, and it accurately compares these attributes to the attributes of the algorithm used in the solution.

J:DIFFERENT APPROACH

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A description is not provided.	The description identifies *at least* 1 aspect that would be done differently if the project were attempted again, but it does not describe what modifications would be made.	The description identifies *at least* 1 aspect that would be done differently if the project were attempted again, and it includes details of the modifications that would be made.

K1:VERIFICATION OF DATA STRUCTURE

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The submission does not verify the data structure used in the solution.	The submission does not verify the data structure used in the solution meets *all* requirements. The verification is missing the total miles added to *all* trucks, or the total combined delivery distance is more than 140 miles, or *all* packages are not delivered on time, or the hash table with look-up function is missing, or the reporting needed is inaccurate or inefficient.	The submission verifies the data structure used in the solution meets *all* requirements, and the verification includes the total miles added to *all* trucks, the total combined delivery distance is less than 140 total miles, *all* packages are delivered on time, the hash table with look-up function is present, and the reporting needed is accurate and efficient.

K1A:EFFICIENCY

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
An explanation is not provided.	The explanation addresses how changes in the number of packages affect the time needed to complete the look-up function, but the explanation contains inaccuracies.	The explanation accurately addresses how changes in the number of packages directly affect the time needed to complete the look-up function.

K1B:OVERHEAD

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
An explanation is not provided.	The explanation addresses how changes in the number of packages affect the data structure space usage, but the explanation contains inaccuracies.	The explanation accurately addresses how changes in the number of packages directly affect the data structure space usage.

K1C:IMPLICATIONS

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A description is not provided.	The description addresses how changes to the number of trucks or	The description accurately addresses how changes to the number of trucks

	the number of cities would affect look-up time or space usage, but not *both*. Or the description contains inaccuracies.	or the number of cities would affect look-up time and space usage.

K2:OTHER DATA STRUCTURES

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The submission does not identify other data structures.	The submission identifies only 1 data structure other than the one used in part D, or the submission identifies 2 data structures, but 1 or *both* of the data structures do not meet the requirements in the scenario.	The submission identifies 2 data structures other than the one used in part D that could meet the requirements in the scenario.

K2A:DATA STRUCTURE DIFFERENCES

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The submission does not describe the differences between data structures.	The submission does not describe attributes for *each* data structure identified in part K2, or the description fails to compare these attributes to the attributes of the data structure used in the solution. Or the description contains inaccuracies.	The submission accurately describes attributes of *each* data structure identified in part K2, and it accurately compares these attributes to the attributes of the data structure used in the solution.

L:SOURCES

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The submission does not include both in-text citations and a reference list for sources that are quoted, paraphrased, or summarized.	The submission includes in-text citations for sources that are quoted, paraphrased, or summarized and a reference list; however, the citations or reference list is incomplete or inaccurate.	The submission includes in-text citations for sources that are properly quoted, paraphrased, or summarized and a reference list that accurately identifies the author, date, title, and source location as available.

M:PROFESSIONAL COMMUNICATION

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
Content is unstructured, is disjointed, or contains pervasive errors in mechanics, usage, or grammar. Vocabulary or tone is unprofessional or distracts from the topic.	Content is poorly organized, is difficult to follow, or contains errors in mechanics, usage, or grammar that cause confusion. Terminology is misused or ineffective.	Content reflects attention to detail, is organized, and focuses on the main ideas as prescribed in the task or chosen by the candidate. Terminology is pertinent, is used correctly, and effectively conveys the intended meaning. Mechanics, usage, and grammar promote accurate interpretation and understanding.

SUPPORTING DOCUMENTS

Sample Core Algorithm Overview.docx

SLC downtown map.docx

WGUPS Distance Table.xlsx

WGUPS Package File.xlsx