**WGUPS ROUTING PROGRAM OVERVIEW**

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.

**I utilized the Nearest Neighbor algorithm (IBM, n.d.) to create my 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.

**Initialize Truck**

**Initialize Packages**

**While there are still Packages in Truck**

**Initialize Shortest Distance with high number**

**For each Package left in Truck**

**If Shortest Distance is greater than distance between Truck's Current Address and Iterated Package's Address**

**Then Shortest Distance equals Distance between Truck's Current Address and Iterated Package's Address and Next Package equals Iterated Package**

**Deliver Packages**

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

**My programming environment:**

* **Visual Studio Code v1.73.1 as my IDE**
* **Python 3.10.8 as my programming language**
* **macOS Ventura 13.0.1 as my operating system**
* **64 GB 2667 MHz DDR4 as my memory**
* **2.4 GHz 8-core Intel Core i9 as my processor**
* [**MacBook Pro 16-inch, 2019**](https://support.apple.com/kb/SP809?locale=en_US)

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

**See source code comments.**

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

**The capability of the solution to scale and adapt is limited due to some assumptions:**

* **Package manifest of 40 packages**
* **Delivery Hub will not change from Western Governor University's address**
* **Trucks are manually loaded with packages to accommodate for special notes and late deliveries which were anticipated ahead of time with the package manifest and performance assessment task instructions, source code will need to change if the package manifest changes e.g. truck departure times**
* **The overall space-time complexity of the program will increase O(n^2) with each package that is added.**

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

**If the package manifest, delivery times, and originating delivery point does not change, there is limited effort required to maintain the software outside of possible software/hardware deprecation, and additionally due to the overall space-time complexity of the program being O(n^2) (due to the nested for loop within a while loop) the program's operational time will only grow quadratically in proportion to the number of packages in the manifest.**

6. Discuss the strengths and weaknesses of the self-adjusting data structures (e.g., the hash table).

**A primary strength of a hash table data structure is its efficiency, as the hash function maps the data to a specific location in the table, the data operations can be done in constant time O(1) on average (Aljuboori, 2021).**

**A primary weakness of using a hash table is the requirement of having a good hash function that prevents hash collisions, a scenario where more than one data element is mapped to the same location in the table, which can lead to loss of data integrity and data operation performance (Aljuboori, 2021).**

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.

**See source code, Main.py.**

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

**See source code.**

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.

**Hash table data structure.**

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

**The hash table accounts for the relationship between the data points via the package ID as the key and the package object as the value.**

**The hash table is a data structure that stores each data element in an array format, where each index in the array is a "bucket" (Aljuboori, 2021). Each of these "buckets" can technically hold multiple key-value pairs, and each key-value pair is stored in a linked list . The hash table uses a hashing algorithm is a slightly modified algorithm borrowed from a YouTube video that is recommended in the C950 course tips section of this course, which helps to determine the index of the array to store the key-pair, using the modulo operator, taking the key and dividing it by the size of the array minus one, returning the remainder as the key hash value, the index (James, 2016). The remainder is used as the index of the array to operate on a key-value pair, supporting "add", "delete", and "get" functions. This hash table class implementation also has a print utility function, which prints the key-value pairs in the hash table.**

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:

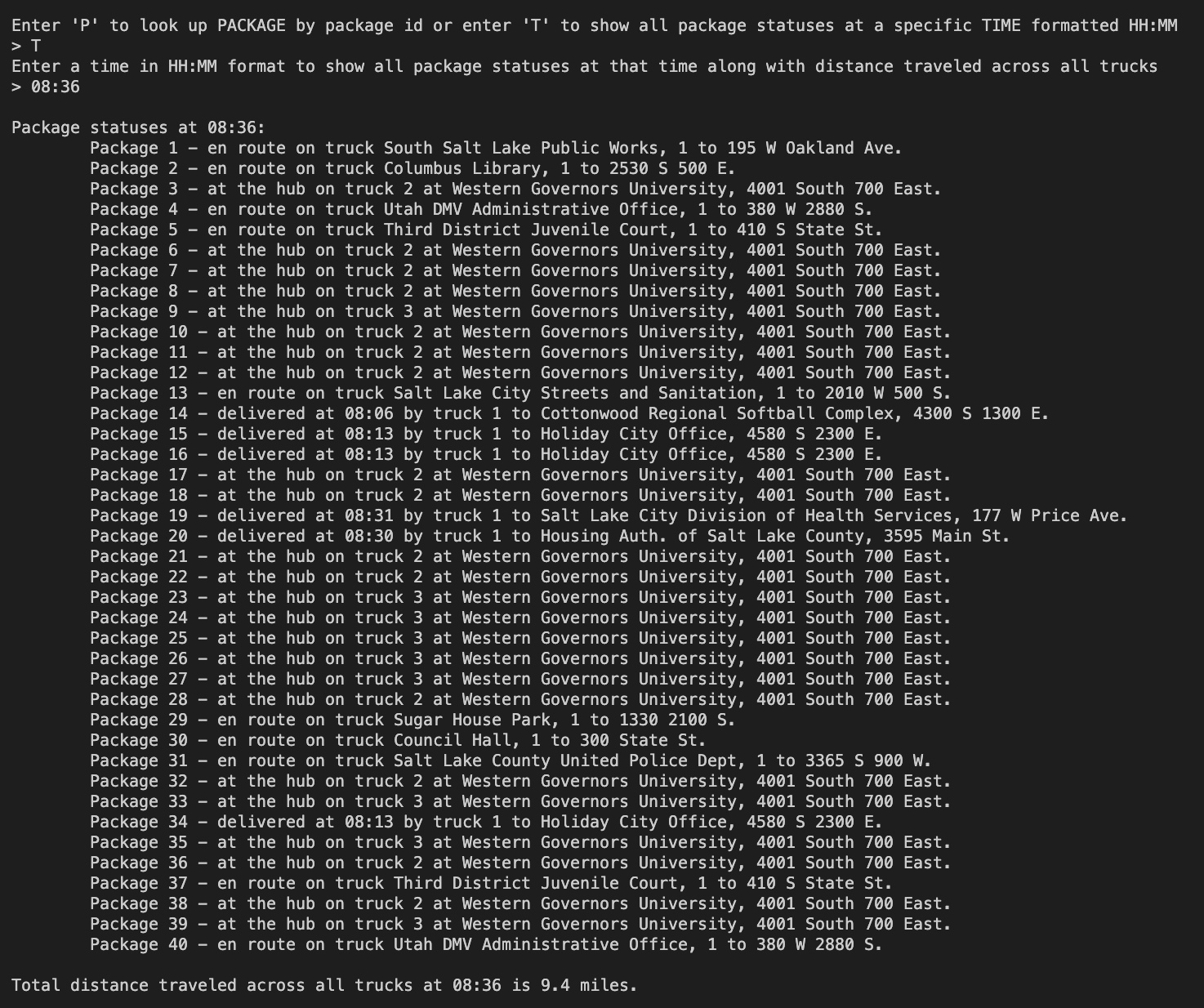
**See source code.**

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

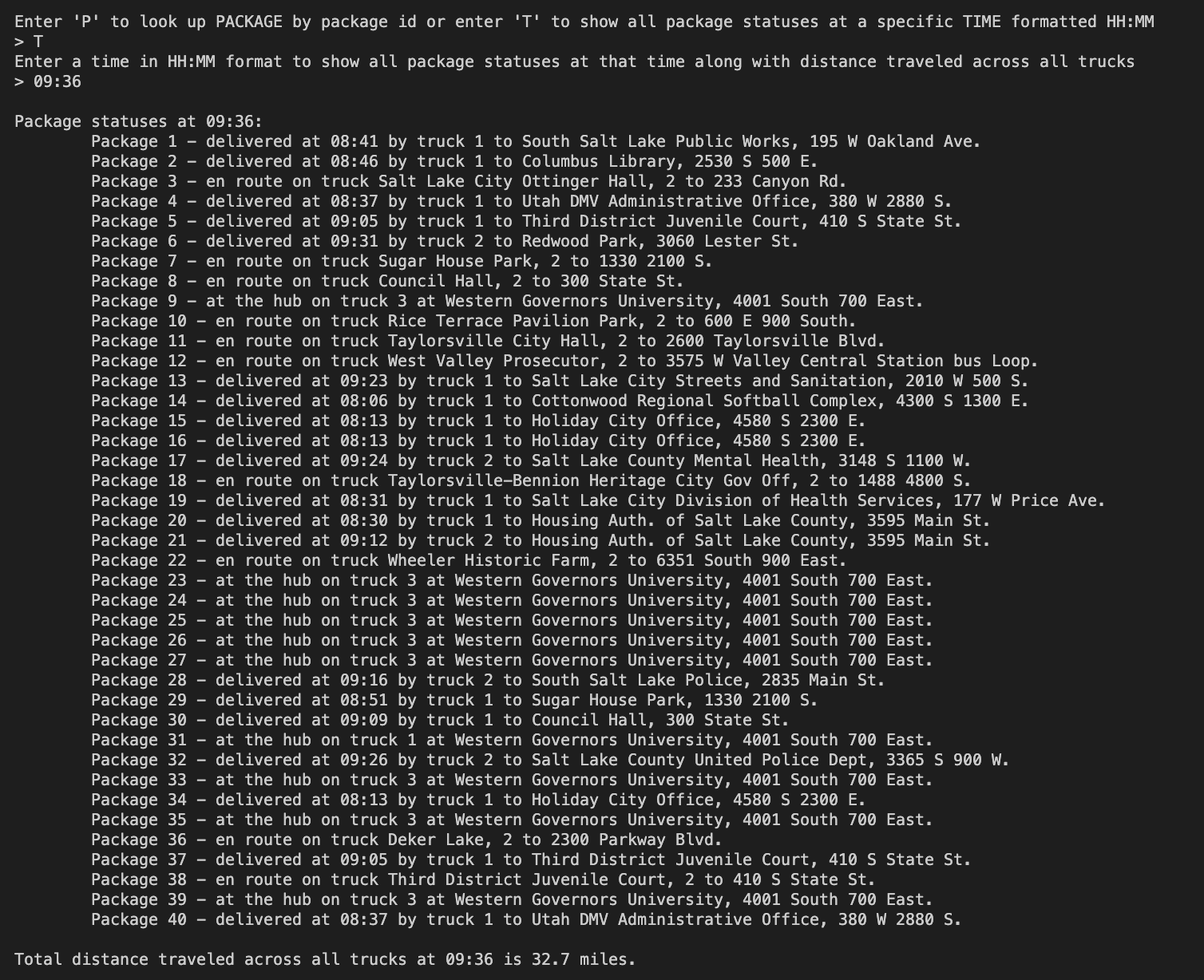
**See source code.**

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.)

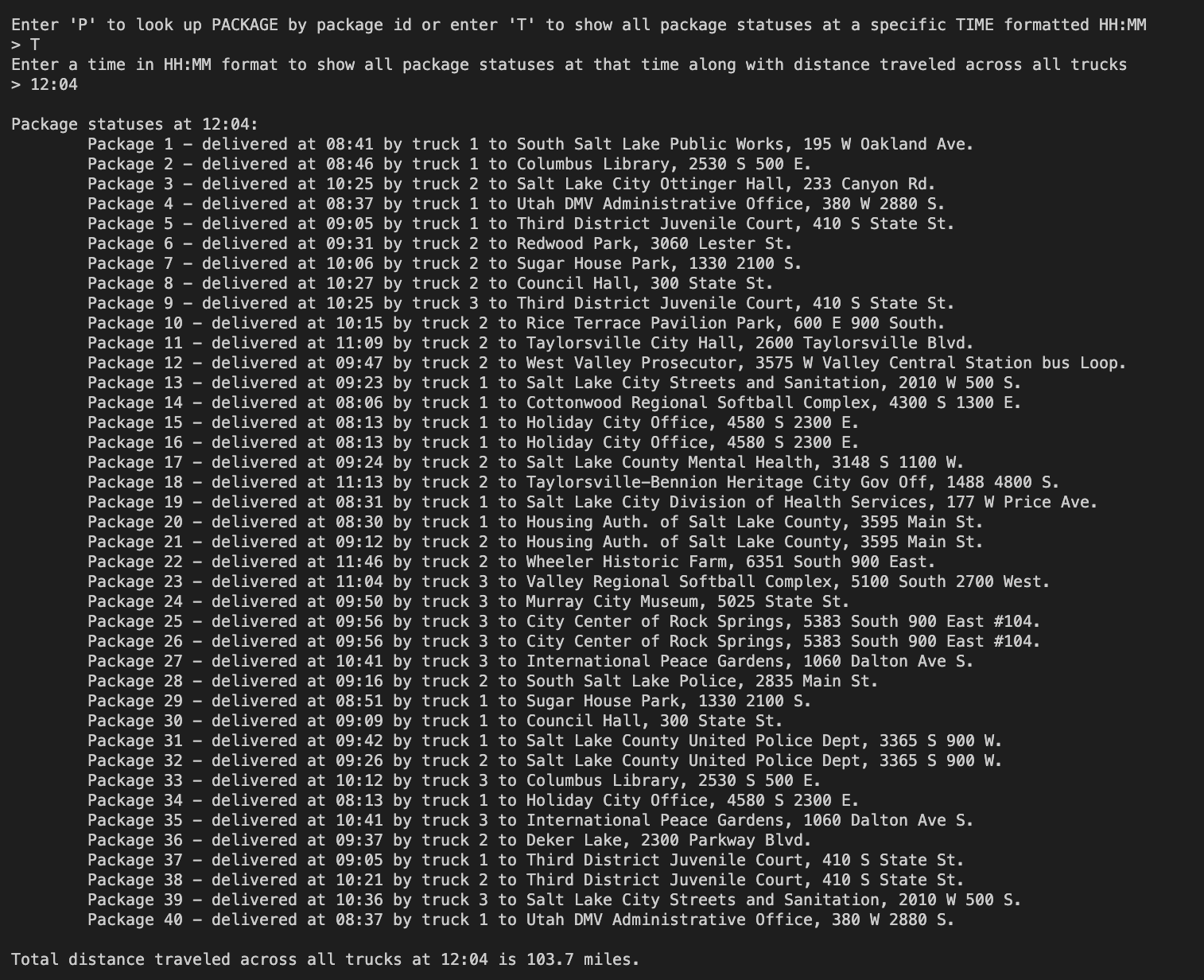
1. Provide screenshots to show the status of all packages at a time between 8:35 a.m. and 9:25 a.m.



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.

**The Nearest Neighbor algorithm is straightforward and easily to implement and it is also flexible, allowing it to be applied to a variety of problems, making no prior assumptions about the dataset (IBM, 2021).**

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

**Yes, I've verified that the algorithm meets all requirements in the performance assessment task scenario.**

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

**B\* algorithm and Dijkstra's algorithm (Hans, 1979; Navone, 2020).**

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

**Both B\* (or B Star) and Dijkstra's algorithms are search algorithm's used to find the shortest-path algorithms that are used to find the shortest path between two points in a graph, whereas the Nearest Neighbor algorithm, while simple, intuitive, and flexible, is generally used for classification and regression tasks to make predictions (Hans, 1979; Navone, 2020; IBM, 2021).**

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

**Spend more time optimizing the code for less manual configuration (e.g. manual truck loading), which is based on the performance task's specific constraints, and attempt at obtaining a much lower delivery mileage across all trucks.**

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.

**The hash table functionality works as expected without errors.**

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

**The implemented hash table has a space-time complexity of O(n) for lookup due to a for loop in the hash table look-up.**

b. Explain how the data structure space usage is affected by changes in the number of packages to be delivered.

**A hash table typically has a space complexity of O(n), meaning the space will grow with the number of packages in the manifest for delivery (Aljuboori, 2021).**

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.

**The trucks are referenced manually and called to deliver packages explicitly in the source code, so the space complexity will be linear however there is effectively no time complexity in play.**

**The cities and distances are stored in a two-dimensional array, so effectively the space complexity would be O(n\*m) and the time complexity using linear search would be O(1) (Leung, 2019).**

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

**Linked List, Binary Search Tree.**

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

* **Hash Table is a non-linear data structure, so the packages would subsequently be non-linearly ordered; Insert, delete, look-up operations are generally in constant time, O(1) (Aljuboori, 2021).**
* **Linked List is a linear data structure, so the packages would subsequently be linearly ordered; Operations are in linear time, and generally O(n) in the worst case (Aljuboori, 2021).**
* **Binary Search Tree is a hierarchical data structure, so the packages would subsequently be linearly ordered; Operations are in logarithmic time, and generally O(log n) in the worst case (Aljuboori, 2021).**

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

**Joe James. (2016, January 16). *Python: Creating a HASHMAP using Lists* [Video]. YouTube. https://www.youtube.com/watch?v=9HFbhPscPU0**

**IBM. (n.d.).  *What is the K-nearest neighbors algorithm?***

**K-Nearest Neighbors Algorithm. https://www.ibm.com/topics/knn.**

**Aljuboori, S. (2021, March 9). *CISP 403: Data Structures.* Folsom Lake College. Open Education Resource (OER) LibreTexts Project. https://eng.libretexts.org/Courses/Folsom\_Lake\_College/CISP\_430%3A\_Data\_Structures\_(Aljuboori)**

**Navone, C. E. (2020, September 28). *Dijkstra's Shortest Path Algorithm - A Detailed and Visual Introduction*. freeCodeCamp. https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/**

**Hans, B. (1979, May). The B\* tree search algorithm: A best-first proof procedure. *Artificial Intelligence*, 12(1), 23-40. https://doi.org/10.1016/0004-3702(79)90003-1.**

**Leung, Y. L. (2019, March 18). *The complexity of simple algorithms and data structures in JS.* freeCodeCamp. https://www.freecodecamp.org/news/the-complexity-of-simple-algorithms-and-data-structures-in-javascript-11e25b29de1e/**

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

**See submission.**