Travis Bittner

C950 – Task 1

1. *Identify a named self-adjusting algorithm (e.g., nearest neighbor algorithm, greedy algorithm) that could be used to create your program to deliver the packages.*

A self-adjusting algorithm that I would use to create my program is the Nearest Neighbor Algorithm. With this algorithm I can adapt to constraints and delays such as those required in the assignment.

1. *Identify a self-adjusting data structure, such as a hash table, that could be used with the algorithm identified in part A to store the package data. Explain how your data structure accounts for the relationship between the data components you are storing.*

A self-adjusting data structure that can be used with my program is a hash table. The relationship would be each key-value pair in the hash table will correspond to package using its ID. The key for each will be the package\_id, and the value will be the entire Package object.

1. *Write an overview of your program in which you do the following:*
2. *Explain the algorithm’s logic using pseudocode.*

For each truck

Start at the hub graph node/Set current hub

While the truck has less than 16 packages and there are unvisited nodes

Find the next nearest unvisited hub

Add all packages from that hub to the current truck

Mark node as visited

Set the current hub as the next hub

1. *Describe the programming environment you will use to create the Python*

*application, including both the software and hardware you will use.*

AMD Ryzen 7700x 8 cores @ 5.5GHz  
32Gb 5200MHz DDR5

Windows 11 Home 10.0.26100  
  
JetBrains PyCharm 2024.3 Professional Edition

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

The complexity of the entire program O(n^2) being a nearest neighbor algorithm with a nested while loop in a for loop. It loops through each truck and then loops through each node until the truck is full.  
  
Each major segment will be O(n). Linear with single for loops dominating the majority of program.  
  
The hash table is O(n)

Populating the hash table is O(n)

Populating the graph is O(n)

The nearest neighbor algorithm and populating the trucks is O(n^2)

Displaying the data such as loading times and delivery times will be O(n)

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

Currently, my solution will be able to scale linearly with the growing number of packages. As it traverses the graph, if it fills all of the trucks there will still be unvisited nodes that need deliveries. The implementation for this would need to be done but the algorithm and nature of the graph would allow for this. This is because all of the packages are “loaded” into the truck before departures. The graph is traversed and each truck is loaded until it is full, getting every package for each hub it visits and marking it visited. The order that the packages are added to the truck is the order that they will be delivered starting from the hub.

1. *Discuss why the software design would be efficient and easy to maintain.*

My software design would be easy to maintain due to modularity and unit testing. Every object is a module with unit testing to accompany to ensure viability. Creating new methods or manipulating the data of the object

1. *Describe both the strengths and weaknesses of the self-adjusting data structure.*

Some of strengths of the self-adjusting data structure are scalability, performance, and efficiency in some cases. The scalability is obvious, as the data structure fills it will re-balance and create more space for more objects. This also ties into the efficiency of the data structure as balancing creates more efficient operations. Performance comes in the form of element reordering, as elements are more frequently accessed, they are moved accordingly. This creates improved access times of these elements and overall performance.

Some of the weaknesses of the self-adjusting data structure are unpredictable performance, complexity, and overhead. The unpredictable performance can come adjustments that create different, or worse, worst-case scenarios. The complexity stems from the nature of the structure dynamicity. Things that have more moving parts are more complex. While overhead isn’t a huge concern with modern day computing, it can be in certain applications. The extra operations needed by the self-adjusting structure increase runtime, thus increasing the computational overhead of the program.

1. *Justify the choice of a key for efficient delivery management from the following components.*The choice of key would be the package ID due to its uniqueness. While being one-of-a-kind, it’s also immutable and it will never change. To further prove the point, they are the shortest attribute of the choices given.