Core Algorithm Overview

**Section 1: Programming/Coding**

**A. Identify the algorithm that will be used to create a program to deliver the packages and meets *all* requirements specified in the scenario.**

*I will be using a simple greedy algorithm that favors to the node with the lowest distance.*

**B. Write a core algorithm overview, using the sample given, in which you do the following:**

1. Comment using pseudocode to show the logic of the algorithm applied to this software solution.

*Step 1: Create a list of nodes from the list of addresses given.*

*Step 2: Create a list of edges based on the distances given.*

*Step 3: After loading the package onto the truck, sort the packages based on the distance from the current node at each delivery point.*

*Step 4: Choose the closest node to you and deliver all packages there.*

*Step 5: Repeat until completed.*

2. Apply programming models to the scenario.

TODO.

3. Evaluate space-time complexity using Big O notation throughout the coding and for the entire program.

TODO.

4. Discuss the ability of your solution to adapt to a changing market and to scalability.

*This solution can handle a solid number of packages but when we reach a high amount of packages, there will be slowdown based on how we traverse the package and how much data we’re sorting.*

5. Discuss the efficiency and maintainability of the software.

*The program is efficient enough to handle small to medium use cases. Further testing is required to achieve efficiency with large datasets. The maintainability is admittedly fragile since the number of packages loaded into each truck is hard-coded.*

6. Discuss the self-adjusting data structures chosen and their strengths and weaknesses based on the scenario.

*The strength of the queues are how easy they are to manage via push/pop operations. The weakness of the queues come from the lack of a lookup table to quickly find a single item inside. Additionally, iterating over the queue can be tricky depending on the data structures contained inside.*

C. Write an original code to solve and to meet the requirements of lowest mileage usage and having *all* packages delivered on time.

1. Create a comment within the first line of your code that includes your first name, last name, and student ID.

2. Include comments at *each* block of code to explain the process and flow of the coding.

**D. Identify a data structure that can be used with your chosen algorithm to store the package data.**

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

*I chose a priority queue combined with a graph and greedy algorithm to determine which packages to deliver and when to deliver them. The relationship to the data points are based on how far away each package needs to go and when it needs to be there.*

**E. Develop a hash table, without using any additional libraries or classes, with 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, in 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

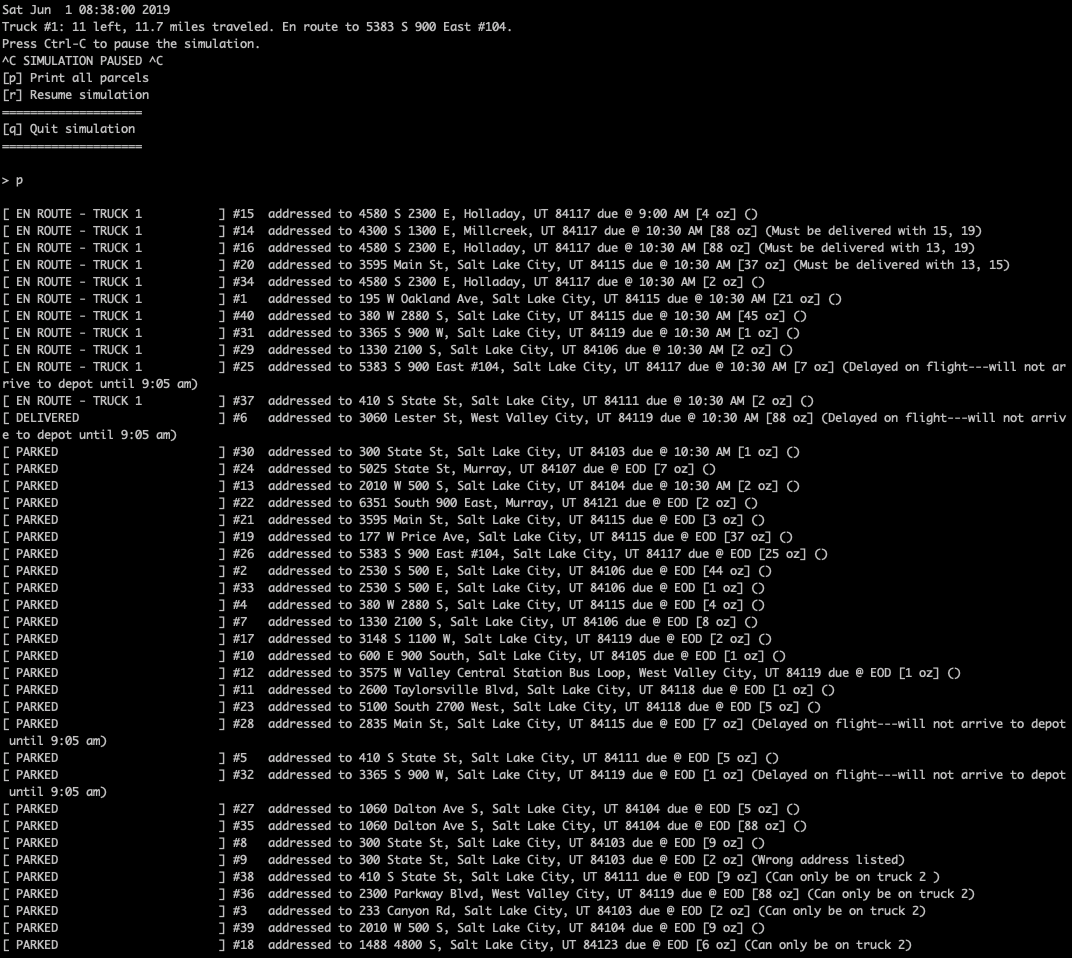
• delivery zip code

• package weight

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

**G. Provide an interface for the insert and look-up functions to view the status of any package at any time. This function should return all information about eachpackage, including delivery status.**

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



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

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



H. Run your code and provide screenshots to capture the complete execution of your code.



**Section 2: Annotations**

**I. Justify your choice of algorithm by doing the following:**

1. Describe *at least*  two strengths of the algorithm you chose.

*\* The code for a greedy algorithm is simple to implement.*

*\* Depending on the implementation and datasets provided, it can be very quick.*

2. Verify that the algorithm you chose meets *all* the criteria and requirements given in the scenario.

3. Identify two other algorithms that could be used and would have met the criteria and requirements given in the scenario.

*Dijkstra’s Algorithm could be used to shorten the distance traveled even further. This would require adding weights to each part of the graph and determining the route to take before leaving the Hub.*

*Miller-Tucker-Zemlin could also be used but it would require the cities to only be accessible via one other city. You could ignore all edges over a certain threshold and try using it that way.*

a. Describe how *each* algorithm identified in part I3 is different from the algorithm you chose to use in the solution.

**J. Describe what you would do differently if you did this project again.**

*I would implement Dijkstra’s instead of a naïve greedy algorithm.*

**K. Justify your choice of data structure by doing the following:**

1. Verify that the data structure you chose meets all the criteria and requirements given in the scenario.

a. Describe the efficiency of the data structure chosen.

*A queue is usually O(N) when traversing it and O(1) for push and pop operations.*

b. Explain the expected overhead when linking to the next data item.

*This is also O(1) because there is no linking involved. We always pull from the top of the queue.*

c. Describe the implications of when more package data is added to the system or other changes in scale occur.

*Sorting and traversing becomes slower as the dataset grows but the push/pop operations should remain constant.*

2. Identify two other data structures that can meet the same criteria and requirements given in the scenario.

*I could use a linked list and construct the package order based on the initial list.*

*I could also use a stack which would be the most analogous to my current implementation.*

a. Describe how *each* data structure identified in part K2 is different from the data structure you chose to use 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.**