**Core Algorithm Overview**

**Stated Problem:**

The purpose of this project is to create an algorithm using Python 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.

**Programming Environment**

IDE – PyCharm Community Edition 2020.3.3

Python Interpreter – Python 3.9

OS – Windows 10

RAM: 8GB

Environment Variables: C:\Python39\Scripts, C:Python39\, %PyCharm Community Edition%, D:\Program Files\JetBrains\PyCharm Community Edition 2020.3.3\bin

**Please note: The program is designed to run in the Windows command line on a 1080p monitor. The GUI may look/behave strange if you run the program in the IDE terminal or on a Mac/Linux.**

**Algorithm Overview**

The routing program is created in the following manner:

1. Create a graph to hold vertices representing each delivery stop.
2. Create a package object for each package provided in the sample file
3. Create a hub and vertex object for each delivery address provided in the sample file.
4. Add the vertices to the graph and connect each vertex using weighted undirected edges.
5. Load each truck using the nearest neighbor algorithm while sorting packages with special notes once they have been discovered. Once a package is loaded, the associated vertex is added to a truck’s available travel path.
6. Once each truck is active, it will go to the closest vertex in its available travel path and deliver the package and remove that vertex from its list of available paths.

**Major Segments**

* **Startup**
  + Initializes the Graphical User Interface (GUI) in the form of a command-line interface (CLI).
    - **Space-Time Complexity - O(1)**
  + Creates a Graph data structure to represent our delivery locations. Loads the information located in the package and distance files as package, truck, hub, and vertex objects.
    - **Space-Time Complexity – O(n^2)**
  + Load the trucks with packages utilizing the nearest neighbor algorithm.
    - **Space-Time Complexity – O(n)**
  + Begin a loop that will cause a truck that is active to deliver a package at the current vertex and then travel to the closest vertex in its available vertex path.
    - **Space-Time Complexity – O(n)**
  + After each loop, update the GUI to reflect new events, package status, and truck details.
    - **Space-time Complexity: O(n)**

Operation time would depend mainly on the number of vertices to be created. The program indicates that there are no directional limitations on the distance, and as a result, each vertex is linked with all other vertices using weighted edges. During creation, we must use two for loops to obtain the distance for each entry in the distance file resulting in the program’s worst-case space-time complexity of O(n^2).

**Ability to Scale:**

The program’s ability to scale would be limited up to 48 packages. Due to the requirements of the program, we only have three trucks with the capability of storing 16 packages each. The loading of the trucks only loads one truck at a time until they have 16 packages or until there are no more vertices left to discover. If the package count is greater than 48, truck count is greater than 3, or driver count changes. The loading and special\_event functions would have to be recreated to accommodate the changes.

**Maintainability**

The program is efficient due to it utilizing a nearest neighbor algorithm to determine the next package to load and the next destination to deliver a package to.

The program is easy to maintain due to it being very modular with comments at each major action. Each major function reads like a set of instructions with each action listed as a separate function/method. This allows the major function to be adjusted and understood easily.

**Strengths and Weaknesses of the Self-Adjusting Data Structures:**