WGU C950 DSA2

NHP2 – NHP2 TASK1: WGUPS ROUTING PROGRAM

## Identify a named self-adjusting algorithm

I used a **genetic algorithm** to determine a near ideal route by randomly guess a subset of the total permutations of routes and mix the best routes (shortest) together and repeated this for a set number of times. The more iterations of this the shorter the route, but the longer it will take to run.

## Write an over of your program

1. **Explain the algorithm’s logic using pseudocode.**

**Note: to make the application reproducible, a numpy.random.seed of 42 is used in Genetic.py**

For each truck

Create a list of package ids to be loaded in a list.

Convert the list of package ids to the index of the address in distance matrix.

Randomly select 25 permutations of the route

Loop for N iterations:

Evaluate the route by taking a list of routes and calculating the total distance of the route, but the route will get a 100-mile penalty if any package is delivered after its deadline.

The shortest route will be added to the best list and to the parent list and a list of vector probabilities will be returned and will randomly be picked to add to the parent list until there are 4 routes.

Once there are 4 routes in the parent list, a mutation may be done (probably of 90%) by swapping the first part of one route with the second part of another route without repeating a stop, then possibility a swapping any of the two stops may occur (probability of 90%).

Loop

The best route will be returned along with the distance of the route.

Next truck

1. **Describe the programming environment you used to create the application.**
   1. The application was developed with Python 3.11 with a virtual environment of pipenv in VS Code.
2. **Evaluate the space-time complexity of each major segment of the program and the entire program using big-O notation.**
   1. Main.py
   2. Helper.py
   3. Genetic.py
   4. Truck.py
   5. HashTable.py
      1. Time complexity: inserts are O(1) and lookups will be O(n) once the list is full.
      2. Space complexity: O(n)
   6. Package.py
   7. ExcelToCSV.py
3. **Explain the capability of your solution to scale and adapt to a growing number of packages.**
   1. The program can scale on the time complexity with the genetic algorithm by trading the number of iterations for efficiency. The data matrix will grow space complexity at O(n^2), but this is not bad for small to medium sized areas.
4. Discuss why the software is efficient and easy to maintain.
   1. The software produces a pretty good route distance that will be more optimal than nearest neighbor. With just 100 iterations most of the routes, if the seed of 42 is not used, will be under 95 miles. The software does not determine what packages go in each truck, but packages with the same address are grouped in the same truck then there will be fewer stops. The maintainability of the software is ok, since there is type hinting and some tests that can be used to make sure that the program can be tested.
5. Discuss the strengths and weaknesses of the self-adjusting data structures (e.g., the hash table).
   1. A hash table is a good data structure for insert and lookups with O(1) being the best possible time complexity and O(n) for the worst for lookups. If the table is under sized or the size of a number that has many divisors, then a weakness will be that a collision. The simple way that I handled collisions was to create a list of items at that hash, but with a size of a prime number there will be minimal collisions.

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.
   1. The hash table is used to hold all the packages that are available to the genetic algorithm. It is used to convert the package ids to address indexes and back to package ids. This is used to generate the route and then to quickly access each package object and fill in the delivery times and what truck was used to deliver the package.

## 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

*Note: Your function should output all data elements for the package ID number.*