This file serves to satisfy the writing section of the assignment.

There are comments throughout the code that help explain logic, and attempt to calculate Big O

Screenshots of the program running are included as .png files.

A:

This program uses a greedy approach type of algorithm to find the nearest package and deliver it

B1: (This is pseudo code)

The packages are manually loaded into the trucks to simplify the sorting. truck.packages = [1,2,3,4] where the numbers in the array are package ids

The packages are then looped through one by one, and related to the addresses and distances data which is then stored in an array

for package in packages:

findNextNearestDelivery(package)

DeliverPackage(package)

what goes on inside of findNextNearestDelivery(package)

is a recursive algorithm that loops through the data in distances.csv and packages.csv and calls itself recursively.

The package is then marked as delivered and removed from the truck upon reaching the next destination

This process is repeated until there are no more packages on the truck.

B2:

written using visual studio code 1.52

python v3

windows 11

B3:

There comments in the code that attempt to evaluate the time complexity for most of the intensive methods

B4:

This solution is not very scalable or adaptable due to manually loading the trucks, and maybe some other hard coded data such as

the start time of the first two trucks. Other than that most of the code is dynamic and could be adapted to be more scalable with some refactoring

B5:

I do not believe this software is 'very' efficient; though I believe with some refactoring it could be made to be more so.

I tried to make it easy to contain by making the packages and trucks into actual objects rather than trying ot keep track of tuples.

The program also attemps to be modular by separating concerns into small methods.

B6:

The hashing method used to insert items into the the hash table is not perfect

and it causes a lot of collisions in buckets which causes the underlying linear

storage to grow much larger than it needs to be.

It however allows for an average insertion and lookup time-complexity of O(1).

The table also allows for easy O(n) iteration over all items

D1:

The revision sent back :

The submission states that a hash table was used in the program. A detailed explanation of the data structure, including how it organizes and relates the stored data, cannot be located.

In computing, a hash table (hash map) is a data structure that implements an associative array abstract data type, a structure that can map keys to values. A hash table uses a hash function to compute an index, also called a hash code, into an array of buckets or slots, from which the desired value can be found. During lookup, the key is hashed and the resulting hash indicates where the corresponding value is stored.

Ideally, the hash function will assign each key to a unique bucket, but most hash table designs employ an imperfect hash function, which might cause hash collisions where the hash function generates the same index for more than one key. Such collisions are typically accommodated in some way.

In a well-dimensioned hash table, the average cost (number of instructions) for each lookup is independent of the number of elements stored in the table. Many hash table designs also allow arbitrary insertions and deletions of key–value pairs, at (amortized[2]) constant average cost per operation.[3][4]

In many situations, hash tables turn out to be on average more efficient than search trees or any other table lookup structure. For this reason, they are widely used in many kinds of computer software, particularly for associative arrays, database indexing, caches, and sets.

-Wikipedia (hashtables)

D1 continued:

My hash table is very simple and does not have a delete method as I feel the project did not require that functionality.

There is a simple hash function used to smear the elements across the array and chaining is implemented to prevent collisions.

The key is stored as the package id itself, and then the value is a DeliveryPackage object which can be found in DeliveryPackage.py

E: DeliveryPackage class has a property called 'transits' which stores the time the package is delivered

I1:

One of the major strengths of the algorithm I used to find the nearest location for the next package to be delievered is its simplicity to understand the logic. It calculates the distances of all of the location from the current location, and delivers the package, and then repeats in a recursive fashion. Another strength of this algorithm is that it is easy to code...

I initially thought to try a more iterative approach without using recursion, but this seemed easier for me to wrap my head around.

I3:

Dijkstra algorithm, A\* search algorithm

J:

If I were to write this program again I would start with very abstract methods and think more 'object oriented'.

I got hung up on small details too soon, and that hindered me from thinking about the big picture.

K1a:

The hash table has an insert and lookup function that work correctly and without

any errors.

K1b:

The hash table will continue to grow was the number of packages grow.

K1c:

The cities and distances between them are held in a graph. The lookup time there

will remain O(1) as the number of cities grows. The space grows at O(n^2) as

each city references the other.

K2a: Linked list, priority queue

A linked list's average lookup is O(n)

its stored elements need to be iterated over in order to find the desired value.

If appending to the end of the list, insertion is O(n)

A priority queue could help simplify the delivery of packages by assigning a

priority to the package before putting it in the queue. This could be done two

ways:

O(1) retrieval and O(n) insertion if elements are sorted at insertion time

O(n) retrieval and O(1) insertion if elements are not sorted at insertion time

The assignment rubric:

The Western Governors University Parcel Service (WGUPS) needs to determine the best route and delivery distribution

for their Daily Local Deliveries (DLD) because packages are not currently being consistently delivered by their

promised deadline. The Salt Lake City DLD route has three trucks, two drivers, and an average of 40 packages to

deliver each day; each package has specific criteria and delivery requirements.

Your task is to determine the best algorithm, write code, and present a solution where all 40 packages, listed in

the attached “WGUPS Package File,” will be delivered on time with the least number of miles added to the combined

mileage total of all trucks. The specific delivery locations are shown on the attached “Salt Lake City Downtown Map”

and distances to each location are given in the attached “WGUPS Distance Table.”

While you work on this assessment, take into consideration the specific delivery time expected for each package and the

possibility that the delivery requirements—including the expected delivery time—can be changed by management at any

time and at any point along the chosen route. In addition, you should keep in mind that the supervisor should be able

to see, at assigned points, the progress of each truck and its packages by any of the variables listed in

the “WGUPS Package File,” including what has been delivered and what time the delivery occurred.

The intent is to use this solution (program) for this specific location and to use the same program in many cities in each

state where WGU has a presence. As such, you will need to include detailed comments, following the industry-standard Python

style guide, to make your code easy to read and to justify the decisions you made while writing your program.

Assumptions:

1. Each truck can carry a maximum of 16 packages.

2. Trucks travel at an average speed of 18 miles per hour.

3. Trucks have a “infinite amount of gas” with no need to stop.

4. Each driver stays with the same truck as long as that truck is in service.

5. Drivers leave the hub at 8:00 a.m., with the truck loaded, and can return to the hub for packages if needed.

The day ends when all 40 packages have been delivered.

6. Delivery time is instantaneous, i.e., no time passes while at a delivery

(that time is factored into the average speed of the trucks).

7. There is up to one special note for each package.

8. The wrong delivery address for package #9, Third District Juvenile Court, will be corrected

at 10:20 a.m. The correct address is 410 S State St., Salt Lake City, UT 84111.

9. The package ID is unique; there are no collisions.

10. No further assumptions exist or are allowed.