A: ALGORITHM SELECTION

*Identify the name of the algorithm used to create a program to deliver the packages and meet all requirements specified in the scenario.*

I have chosen an algorithm that uses a generated adjacency matrix to search for the shortest path using a greedy algorithm. The adjacency matrix is assumed to be ordered the same up-down as left-right. It chooses the direction that the truck needs to travel based if the destination is positioned *before* or *after* where the current location of the truck is in the matrix. E.g. if the current location is 5th in the list, and destination is 12th in the list, the algorithm will look to the right until the destination has been reached.

Once direction has been decided, the algorithm will find the shortest distance in that direction. It will repeatedly search for the shortest distance until the destination has been reach, all while adding all these distances together. It will then finally compare the combined distances path to what the direct path would be. It will then return whichever one is the shortest. Oddly enough, with this method there are only a few instances where the direct path *isn’t* the smaller path.

In essence, the software determines a greedy algorithm route of shortest distances and decides if that route is shorter than a direct route. It is self-adjusting because it does not matter the amount of locations. It will select the shortest path to deliver a package regardless of the size of the location matrix, or the amount of packages.

This algorithm is sufficient because the total miles traveled is 128.3. The packages that have notes are loaded hard coded into arrays, but the algorithm will still work regardless of the number of items in each array. The program could be adjusted to accommodate these packages with special requirements. Also, the algorithm is not adjusted to find the absolute shortest path that it can take. It could be made to check all available paths, but it would require a lot of programming and a lot of time complexity. Finally, the algorithm isn’t made to take all packages considered and find the shortest route possible out of every combination. While this could be done, it would require a large time complexity.

B1: LOGIC COMMENTS

*The comments accurately explain the logic applied to the solution.*

Comments are presented throughout the code. Each comment explains what procedure each chunk of code is accomplishing step-by-step. It is written so that a stranger who has no idea what the program is will understand the goal and the process. The comments also have a space-time complexity Big O notation for each larger chunk.

B2: APPLICATION OF PROGRAMMING MODELS

*Provide a description of the communications protocol used to exchange data, the target host environment used to host the application server application, and the interaction semantics defined by the application to control connect, data exchange, and disconnect sequences.*

The program is written in Python 3. There is no communication protocol or a host server according to the requirements of the project. All mandatory components of the program are located on the same machine in the same root file. The location list and the package list are located inside of the Resources folder within the root of the program.

The lists are stored in a CSV file (one that is formatted differently from the given lists, but still contains the exact same data.) The lists are read in through the openFile method and returned as a dictionary. The returned dictionary is then used to make a list of the data collected.

The method of opening a file and reading the information is based on an example that I found from Reading CSV files In Python (pythonprogramming.net, 2020). It uses a similar opening and closing method, but with the information being passed into a dictionary.

B3: SPACE-TIME AND BIG-O

*The evaluation shows the space-time complexity using Big-O notation for each major block of code and the entire program.*

The space-time complexity of each large chunk of code has been identified throughout the program in the comments. If the block of code calls another method that will change the complexity, it is also noted why the complexity is larger than what it looks like from a glance.

B4: ADAPTABILITY

*The discussion includes the chosen algorithm’s ability to handle a growing number of packages (scalability).*

The algorithm can control as many packages as needed. The ability to deliver packages is unaffected by number of packages or locations. The hash table is the only part of the algorithm that changes in size according to the number of entries in a list of packages. The hash table uses a bucket chaining method. It will create 10 ^ int(log N buckets) for N number of packages. This ensures that the max number of packages and collisions in each hash bucket never exceeds 9. Also every bucket inside of the hash table will be used at least once.

Each bucket not exceeding 9 entries is beneficial because it will have O(1) search time of finding a bucket, and then have to search through no more than 9 entries before the searched package is found. It will only take 1 step to select the bucket, then a max of 9 steps to find the entry. So a search will take a max of 10 total steps.

Another benefit to the algorithm is that the bucket is an array that is appended each time a package is added to it. The time complexity of appending the array is O(1) because it does not require the packages to be in order to function.

As for the cons of this approach, though the search time will never take more than 10 steps total, the amount of space needed will grow by a 10 fold every time 10 ^ int(log N) increases. This is problematic because there is a potential the hash table will hold one entry per bucket while there are 10 spaces allocated in each bucket. 90% of the space allocated will be unused and can grow quite dramatically. While it does not take significantly more space between 10 and 100 packages (99 spaces vs. 999 spaces), when N number of packages grows large the amount of space needed does grow significantly. E.g. from 99,999 to 100,000 packages, there needs to be allocated 99,999 spaces to 999,999 spaces – just for one more package. Of which 899,999 spaces will be allocated with no information in them.

In exchange for a quick hash search, the hash table sacrifices space complexity and potentially could use a massive amount of space once the number of packages grows.

B5: SOFTWARE EFFICIENCY AND MAINTAINABILITY

*The discussion addresses how the software’s efficiency and why it is efficient and easy to maintain.*

The Big-O time complexity of the program never exceeds N^3, or a polynomial time. The simulation is ran, which will repeat either until a specified time or until all packages are delivered. While the simulation is being conducted, the truck will determine the next route after the current package is delivered. This process takes O(N^2) because the routing method find the shortest route of N locations with N packages.

The algorithm is easy to maintain because each process if broken into separate methods and is explained thoroughly with comments throughout the code. A stranger to the program could look at the code and easily understand the flow and reasoning.

B6: SELF-ADJUSTING DATA STRUCTURES

*The discussion of the self-adjusting data structure(s) includes the ability of the data structure(s) to adapt when data is inserted, removed, or accessed; and how that adaptation affects running time.*

Data is initially inserted into the hash table from the CSV file. After the initialization of hash table, there are functions to insert a new package using it is packageID for the hash function. There is an update function, which finds the package and updates the address values. This function is used to update the incorrect address for package number 9. And there is a delete function to remove any package, though not used in the program.

C: ORIGINAL CODE

*The original code runs properly and delivers all packages within their requirements while adding the least number to the combined mileage total of all trucks in less than 145 miles.*

The program functions properly and correctly. The simulation can be ran until any time, or until they are done. The total distance that they travel is 128.3 miles. The screenshots show the status of the packages and truck locations at 9:00am, 10:00am, and 1:00pm. There are no errors.

C1: IDENTIFICATION INFORMATION

*The initial comment is located within the first line of code and includes the candidate’s first name, last name, and student ID.*

Identified in the main.py where the program is initialized and ran, and in the mainMenu as well.

C2: PROCESS AND FLOW COMMENTS

*Include comments at each major block of code explaining the process and flow of the code.*

The entire program has explanations of each process. The beginning of each file gives a small explanation of what its functions are used for, and each big block explains what is happening. If there is a call to another function, it is also explained for maintainability.

D: DATA STRUCTURE

*Identify a data structure written by the student using only primitive data structures, lists, tuples, or sets, used by your program to store and retrieve package data.*

The data structure that I have chosen is my hash table, which is used to store the packages. The packages are read in and stored based on package ID, and can be inserted, deleted, or updated by package ID as well.

The information is read in from the CSV file into a dictionary, and then is put into the hash table that is made of arrays. The packages are stored as a tuple, with package ID as its key and the original information stored as the second value. (‘1’, [‘1’, ‘WGU Hub’, ‘Address’…]). This tuple is stored into an array as the bucket. This bucket can contain multiple package tuples. Finally, the hash table is made up of these buckets. It could be called “an array of array of tuples”, therefore using only primitive data and not using a dictionary of any sort.

In the hash table, I defined an iterator and next methods to be able to use for loops. I also used a str method in the hash table and other places to allow for a printing format when an item in a table was called to be printed. A Guide to Pythons Magic Methods (Kettler, 2012) was a huge help to understanding how these methods work and how to write them.

D1: EXPLANATION OF DATA STRUCTURE

*The submission accurately explains the data structure and how that data structure accounts for the relationship between the data points to be stored.*

The hash table is used to quickly search for a package using its ID. Rather than searching through every single package stored, it finds the bucket in which the package is stored instantly and then will only have to search through a small number of packages. This allows for a much shorter time complexity.

E: HASH TABLE

*The hash table has an insertion function that includes, as input, all of the given components.*

The insert function allows for a package to be inserted into the hash table by passing in the package information that is formatted the same as the CSV file. The hash function will automatically add it to the table based on it’s package ID, and all of the package information is stored.

F: LOOK-UP FUNCTION

*The look-up function includes all of the given data elements, completes searches via package ID, returns the data corresponding to the provided ID including the package’ status (at the hub, en route, or delivery time).*

The data from the hash table is initially used to make package objects for each package at start of the simulation. These packages are copied exactly without altering the package information from the hash table. These package objects will provide a packages status. The searches are complete at the beginning of the simulation and are searched throughout the program as well.

G: INTERFACE

*Provide an interface for the user to view the status and info of any package at any time.*

The status of any package can be viewed at any time via the main menu option #4. The user selects the time, and it will show all package statuses at that instant. The function to search for only one individual package at a certain time is not coded, but the ability to find the status of any package is available at any time.

G1-G3: 1st, 2nd, and 3rd status checks.

*Provide screenshots showing the info (outlined in part F) and statuses at a time between:*

* *8:35 a.m. and 9:25 a.m.*
* *9:35 a.m. and 10:25 a.m.*
* *12:03 p.m. and 1:12 p.m.*

A screen shot for 9:00am, 10:00am, and 1:00pm are all in the screenshot folder at the root of the program.

H: SCREENSHOTS OF CODE EXECUTION

*Provide a screenshot or screenshots showing successful completion of the code free from runtime errors or warnings.*

The screen shot of a successful screenshot is also located in the screenshot folder at the root of the program. However, all of the screenshots prove that the program is error-free.

I1: STRENGTHS OF THE CHOSEN ALGORITHM

*The description includes at least two specific strengths of the chosen algorithm as they apply to the scenario.*

The strengths of my package delivering algorithm will search for the nearest package that is deliverable from the current location of the truck. Rather than trying to find all possible routes from the beginning, deciding which package to deliver after the current one is delivered is much quicker. The algorithm also uses a greedy algorithm to find the shortest path by finding the sum of shortest distances from location to location until the destination is found. It compares the sum of shortest distances to the direct path and will choose whichever path is shorter. This allows for the algorithm to compare two short paths without taking much time.

I2: VERIFICATION OF ALGORITHM

*The verification includes the total miles added to all trucks, and it states that all packages were delivered on time.*

The distance traveled of each truck is printed with the package statuses. The trucks mileages, current location, and destination are all supplied at any given time. Also, the last lines printed state the total miles driven at that time. The max miles driven is 128.3.

I3: OTHER POSSIBLE ALGORITHMS

*The submission identifies two other algorithms that could meet the requirements of the scenario.*

One possible algorithm could use a binary search tree to search multiple routes to find the shortest distance.

Another algorithm could decide a route before the trucks even deploy.

I3A: ALGORITHM DIFFERENCES

*The description includes attributes of each algorithm identified in part I3 and how the identified attributes compare to the attributes of the algorithm used in the solution.*

The binary search for multiple routes for each package is like the algorithm I chose, where the truck decides which package to deliver next based on current location and packages left. But my algorithm only compares two distances using a greedy algorithm. The binary search could search many more paths that are possibly shorter than the two that my algorithm could find.

The algorithm to decide the routes before the trucks leave could optimize the shortest path possible for each truck. It could find distances between combinations of two packages and use that to determine a faster path. This could potentially be a significantly shorter distance traveled than my algorithm, but at the cost of a lot of time complexity. Comparing the distance between a pair of *every* package would take a long time.

J: DIFFERENT APPROACH

*The description includes at least one aspect of the process that the candidate would do differently and includes how the candidate would modify the process.*

I would go back and change out the greedy path vs. direct path algorithm. While it is quick and optimizes the milage below 145, it is not even close to the optimized solution. I think I would incorporate at least a few more paths between each package to ensure that the distance is close to the shortest possible.

K1: VERIFICATION OF DATA STRUCTURE

*The verification shows all the criteria have been met: the least number of total miles added to all trucks, all packages were delivered on time, the hash table with look-up function is present, and the reporting needed is accurate and efficient.*

The result menu after running the full day shows that the trucks complete all deliveries in 128.3 miles. All packages are delivered on time, and the results show when the packages were delivered as well. The hash table has a search, insert, and delete function. All data reported can be easily found through the run terminal.

K1A: EFFICIENCY

*The description of the efficiency of the data structure (hash table) used in the solution includes what type of data is being used and how that data is being used.*

The hash table makes the program able to search for packages and their information faster and able to look up information based on package ID. All information of the package can be accessed from the hash table. The program uses the hash table to look up the address of the packages, as well as their deadlines and special notes.

K1B: OVERHEAD

*The explanation of the data structure (hash table) includes the computational time, memory, and bandwidth aspects when handling data in this program.*

There is no use in bandwidth because all data is located on the local machine and not through a database. The computational time is Big-O (N), because it is dependent on the number of packages being stored in the hash table. But the look up function will never exceed more than 10 searches, so it will still have a very quick search time.

K1C: IMPLICATIONS

*In regards to the data structure (hash table), describe the implications when more packages are added to the system or other changes in scale occur.*

Adding more packages to the hash table will not interfere with the ability to store the information, as long as the package ID remains unique. The search function still will not ever take more than ten searches. The only downfall of my chosen style of hash table is that the number of buckets can start to take up a lot of space, and all of that space may not always

K2: OTHER DATA STRUCTURES

*The submission identifies two data structures other than the one used in the solution that meets the criteria and requirements in the scenario.*

A hash table that uses a linear probing hash function could have been used. Or one with a double hash function as well.

K2A: DATA STRUCTURES DIFFERENCES

*The description includes the attributes of each data structure identified in part K2 and compares these attributes to the attributes of the data structure used in the solution.*

A linear probing hash function uses an equation that would take a part of the package information and turn it into a key. If two different packages had matching keys, the linear probing would move to the next space of the hash table until an empty space is found that had not been used before.

I originally tried this by using the packages address, converting each char into the ASCII number of each, adding up the value of each, and then using a modulo to shorten the amount of spaces used. This proved problematic because there were many collusions. So I moved on to the double hash function method.

The double hash function method works the same way as the linear, except when there is a collision then a different hash function is applied to the key until they are no collisions.

When I tried to use the double hash function, I still attempted to add the chars of the address together and apply a modulo. Once there was a collision, my second hash function was to square the number and select the middle two digits of that number to use as a new key. There still were many collisions. I tried various functions, even a quadratic hash function, and still had a hard time coming up with a hash function that had under 5 collisions. I decided after a while that I would use the bucket chaining hash method with a scaling amount of buckets.

L: SOURCES

*The submission includes in-text citations for sources that are properly quoted, paraphrased, or summarized and a reference list that accurately identifies the author, date, title, and source location as available.*

Where I researched and got information from has been cited, and all references are located on the “References” page.

M: PROFESSIONAL COMMUNICATION

*The content reflects an attention to detail, is organized, and focuses on the main ideas as prescribed in the task or chosen by the candidate. Terminology is pertinent, is used correctly, and effectively conveys the intended meaning. Mechanics, usage, and grammar promote accurate interpretation and understanding.*

I used as professional communication to the best of my ability. I haven’t even used any emojis anywhere.

References

Kettler, R. (2012). A Guide to Python's Magic Methods. Retrieved November 10, 2020, from https://rszalski.github.io/magicmethods/

Reading CSV files in Python. (n.d.). Retrieved November 10, 2020, from https://pythonprogramming.net/reading-csv-files-python-3/