Danielle Ajala Cruz

001179605

C950

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

For this solution I used a greedy algorithm.

**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.**

The greedy algorithm in this solution is used to find the shortest path from different addresses in Salty Lake City. The algorithm is implemented to help WGUPS to determine the best route to deliver its daily delivery packages.

The greedy algorithm works as follows:

START

Parameters:

**package\_addresses\_list** // list of packages addresses index in a truck to be delivery

**distances\_adjacency\_list** // an adjacency list that holds the distance value among all the addresses

Initialization:

**initialize start\_location to 0** // to set the start location to the hub

**initialize distance\_accumulator to 0**

**initialize shortest\_distance to 1000**

**initialize path\_distance\_value as an empty list** // this list will hold the value of the distance of the

shortest path

**initialize path\_addresses as an empty list** // this list will hold the address indexes of the shortest

path

**WHILE package\_addresses\_list length is greater than 0**

**{**

**FOR each line in distances\_adjacence\_list** // each line represents one address

**{**

**IF the value of distances\_adjacence\_list [current line][start\_location] is between 0 and**

**shortest\_distance**

// note: the first brackets represents a line and the second brackets represents a column

**{**

// Checks if the address with the shortest distance is the destination of one or more

package in the truck

**IF the address index of the current line of distances\_adjacence\_list is also in**

**package\_addresses\_list**

**{**

**set shortest\_distance to distances\_adjacence\_list [current line][start\_location]**

**set address\_index to current line index**

**}**

**}**

**}**

**remove address\_index from package\_addresses\_list** // to avoid the same address to be visited more

than once

**set distance\_accumulator to distance\_accumulator + shortest\_distance** // to get the total travel

distance

**set start\_location to address\_index** // to start the search for the next closer address from the last

address visited

**set shortest\_distance to 1000**

**append shortest\_distance to path\_distance list** // it will be used to calculate the delivery time

**append address\_index to path\_addresses\_list** // it will be used to calculate the delivery time

**}**

// after the last package delivered, the truck has to come back to the hub

**IF the length of package\_addresses\_list = 0**

**{**

**set minimum\_distance to distances\_adjacence\_list [0][start\_point]**

**set distance\_accumulator to distance\_accumulator + minimum\_distance**

**}**

END

**2.  Apply programming models to the scenario.**

The programming model for the scenario was developed to run in just the current host and local machine. The software is written in Python language, version 3.7, and was developed in the PyCharm IDE. Due to the fact that the entire application is executed in just the local machine, no communication protocol nor target host environment was needed. To have access to the data demanded, the application pulls information from CSV files located in the project folder on the local machine. To do that, no network and/or server connection is required, it only made use of the csvreader function from the inbuilt CSV Python library. In addition, the application has no interaction semantics to control connect, data exchange, and disconnect sequences, because once again, the application is only being hosted in the local machine.

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

**hashTable.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Line number** | **Space complexity** | **Time complexity** |
| \_\_init\_\_ | 8 | O(1) | O(1) |
| \_my\_hash | 19 | O(1) | O(1) |
| add | 25 | O(1) | O(1) |
| search | 36 | O(N) | O(n) |
| **Total** |  | **1N +3 = O(N)** | **1N +3 = O(N)** |

**package.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Line number** | **Space complexity** | **Time complexity** |
| \_\_init\_\_ | 6 | O(1) | O(1) |
| **Total** |  | **O(1)** | **O(1)** |

**readPackages.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Line number** | **Space complexity** | **Time complexity** |
| None | 9 | O(N) | O(N) |
| None | 19 | O(N) | O(N) |
| get\_hash\_table | 36 | O(1) | O(1) |
| **Total** |  | **2N + 1 = O(N)** | **2N + 1 = O(N)** |

**loadTruck.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Line number** | **Space complexity** | **Time complexity** |
| load\_trucks | 17 | O(1) | O(1) |
| get\_address\_vertices | 40 | O(N) | O(N) |
| get\_vertices\_truck1 | 58 | O(N) | O(N) |
| get\_vertices\_truck2 | 74 | O(N) | O(N) |
| get\_vertices\_truck3 | 87 | O(N) | O(N) |
| get\_truck\_one\_packages | 100 | O(1) | O(1) |
| get\_truck\_two\_packages | 106 | O(1) | O(1) |
| get\_truck\_three\_packages | 112 | O(1) | O(1) |
| get\_all\_packages | 118 | O(1) | O(1) |
| **Total** |  | **4N+5 = O(N)** | **4N+5 = O(N)** |

**distances.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Line number** | **Space complexity** | **Time complexity** |
| None | 9 | O(N) | O(N) |
| find\_short\_path | 18 | O(N^2) | O(N^2) |
| get\_shortPath | 69 | O(1) | O(1) |
| **Total** |  | **N^2+N+1 = O(N^2)** | **N^2+N+1 = O(N^2)** |

**timer.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Line number** | **Space complexity** | **Time complexity** |
| time\_per\_mile | 9 | O(1) | O(1) |
| set\_time | 26 | O(N^2) | O(N^2) |
| **Total** |  | **N^2+1 = O(N^2)** | **N^2+1 = O(N^2)** |

**main.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Line number** | **Space complexity** | **Time complexity** |
| truck\_out\_for\_delivery | 16 | O(N^2) | O(N^2) |
| None | 37 | O(1) | O(1) |
| None | 65 | O(N) | O(N) |
| None | 85 | O(N) | O(N) |
| None | 100 | O(N) | O(N) |
| None | 105 | O(1) | O(1) |
| **Total** |  | **N^2+3N+2= O(N^2)** | **N^2+3N+2= O(N^2)** |

**lookup.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Line number** | **Space complexity** | **Time complexity** |
| lookUp\_id | 10 | O(N) | O(N) |
| lookUp\_time | 29 | O(N) | O(N) |
| next\_action | 73 | O(1) | O(1) |
| **Total** |  | **2N+1 = O(N)** | **2N+1 = O(N)** |

**Total**

|  |  |  |
| --- | --- | --- |
| **Section** | **Space complexity** | **Time complexity** |
| hashTable.py | O(N) | O(N) |
| packages.py | O(1) | O(1) |
| readPackages.py | O(N) | O(N) |
| loadTruck.py | O(N) | O(N) |
| distances.py | O(N^2) | O(N^2) |
| timer.py | O(N^2) | O(N^2) |
| main.py | O(N^2) | O(N^2) |
| lookUp.py | O(N) | O(N) |
| **Total** | **3N^2+4N+1= O(N^2)** | **3N^2+4N+1= O(N^2)** |