C950: DATA STRUCTES & ALGORITHM II

Lois Vernon Pua
WESTERN GOVERNORS UNIVERSITY

A. Algorithm:

The self-adjusting algorithm used in the project is a variation of the Greedy Algorithm known as "Nearest Neighbor" algorithm. The self-adjust algorithm is called "nearest_distance" inside the program. To begin, the algorithm checks if a given truck is empty, this determines if the algorithm can exist and eventually continue. On the other hand, when a certain truck is not empty or loaded at some point, the algorithm will then loop through an iteration with the packages remaining and retrieve the distances within the truck's current location (this is beginning from the "HUB") to the next nearest delivery location. Then, the values that was returned is stored into another value variable and the package associated with the delivery is also stored to a temporary variable. The truck's location will be updated to the new location where it recently delivered the most recent package. From the requirements that the truck will run in a constant velocity of 18 mi/hr without stopping, the algorithm should continue as long as the total miles given remains greater than all the miles possible. If the case is false, the current package will be removed from the truck's current inventory and the status will change to "Delivered".

Complexity of the Algorithm:

- Looping through an iteration with the packages: **O(n)**
- Dictionary lookup: **O**(1)
- Accessing the hash table: O(n)
- Removing a package from the list: **O(n)**
- Overall space-time complexity: **O**(**n**)

Once the truck completely delivers all the packages, the last destination is set as a new "HUB", and the total distance traveled as miles is also updated.

Although the overall space-time complexity is O(n), this doesn't entail the final space-time complexity of the said algorithm. This is only when running through the iteration of algorithm one time. When another delivery is "called" this will increase to a space-time complexity of $O(n^2)$.

B1. Take a random argument(in this case, "truck_package"):

```
Lowest Distance(truck_package) {
```

 $minimum_distance = 1000000$ (this is an arbitrary large number greater than all possible distances)

```
package_remove == null (acts as a placeholder)
```

```
for package in truck_package \{ \rightarrow O(n) \}
      if minimum_distance > the current distance from the current address to next
   package address \rightarrow O(1)
   desitance of truck = package address
   minimum_distance = new distance (new distance that is shortest)
   package_remove=package (package for next delivery)
      }
   }
   current location of truck = truck destination (updated for running through the
   next loop)
   total miles of truck =+ minimum distance (total miles of the delivered packages)
   if total miles of truck > miles of truck from user's input{
   return \rightarrow (The given constraint should be consistently applied (18 miles per
   hour), this will ensure the algorithm will continue and stop otherwise if
   constraint is not followed)
   package removed status = leave time + time elapsed
   remove "package removed status" from truck packages
   if truck packages == 0(or empty) {
   new trucks destination = WGUPSHUB
   total miles = truck total + distance to WGUPSHUB

    ** Overall distance to the hub is summed as total miles when all

         deliveries are done. **
      }
   }
B2. Programming Environment used:
      IDE: PyCharm 2021.2.3 Community Edition
      Python Version: Python 3.10
B3. Complexities:
```

```
distance.py → Looping= O(n)

→ Dictionary look up = O(1)

→ HashTable access = O(n)

→ Removing packages = O(n)

→ Overall = O(n)

truck.py → truck_delivery = f(delivery) * f(deliver_time) = O(1) + O(n²) = O(n²)

→ Overall = O(n²)

main.py → main() = f(delivery) * f(deliver_time) = O(1) + O(n²) = O(n²)

→ print_pacakge: O(n)

→ print_table: O(n)
```

B4. Capability scale of program:

The algorithm is made for smaller values of "n". From the assessment requirements of 40 n, the overall complexity value will be $O(n^2)$. Although, when n starts to grow to large amounts, the algorithm will lose efficiency. Thus, the scalability of the program is made for smaller n values and larger n values will require a different algorithm.

B5.

As stated from the previous section, the algorithm is most efficient when n is small. But the class "truck" inside the program maintains the increased maintainability of the algorithm. It contains all the given information about the status of each individual trucks, this allows for easier organization and readability. Furthermore, the efficiency will also be higher when compared to other multiple self-adjusting algorithms, creating hash tables and dictionaries. Thus, when n starts to grow exponentially, the efficiency of the algorithm will eventually go down.

B6.

Inside the program, a hash table is strictly implemented, with this, comes with advantages and disadvantages. The main advantage is the efficiency of the hash table. Hash tables average lookup time is constant, which yields a complexity of O(1). Hash tables also helps in organizing data with the use of hash functions. The main disadvantage of the hash table is that it has to be implemented correctly. If the buckets are appended the wrong way, collisions will exists which in turn will increase the complexity of the overall algorithm.

D1. Data structures:

The Hash table located inside the program has a purpose of mapping data points with related unique IDs in incrementing order. The hash function will make it easier it terms of choosing data points. Therefore "key % 10" is used to fill the buckets with certain elements. Yielding in a constant time complexity of O(1) no matter how many elements that will fill the buckets. The hash table is also efficient in changing the statuses of packages delivered.

I1. Strengths of the entire algorithm:

The version of greedy algorithm used in the program will depend on the implementation of looking up dictionaries for addresses with the smaller n values. The dictionaries inside the program, mainly "destinations" and "distance" allows very efficient retrieval between the distance of two points. Therefore, the algorithm scales with the number of n given at a certain point.

I2. Algorithm requirements:

The algorithm implemented in this project meets all the requirements. The total distance traveled by all truck is 123.5 miles (with rounding) after all the trips are done, which is less than the maximum allowed distance of 140 miles. All the packages meet the delivery deadlines as well.

I3. & I3 A. Other algorithms:

Other algorithm that would meet the requirements would be a Topological sorting algorithm and Dijkstra's algorithm. The Topological sorting algorithm behaves the same with depth first search (**Acta Informatica 6, 171–185 (1976)**). It visit each nodes until the desired outcome is met, yielding a time complexity of O(n + m). It visits the next neighbor node, which is like a greedy algorithm, but it visits till the last node is visited. On the other hand, the Dijkstra's algorithm will calculate the shortest path from a node given and compares to every other node ("**Section 24.3: Dijkstra's algorithm"**. **Introduction to Algorithms (Second ed.)**. **MIT Press and McGraw–Hill. pp. 595–601.)** It differs from the implemented algorithm since it also visit and calculates the shortest distance closest and not the entire paths.

J. Other data structures approach:

One specific different approach will be the use of a different style of calculating the nodes. The calculation of shortest distance rather than looping and calling through the iteration is one key change to be made for better efficiency and scalability.

K.1.

The self-implementation of Hash tables inside the program sufficiently meets the project's requirements. It can work with the algorithm created in terms of retrieving values with the created hash key. It's also capable of inserting new values with unique IDs.

K1.A & K1.B

The hash function inside the program's hash table class "key % 10" was used because of the number of packages given, which is 40. The buckets created is 10, therefore, If the number of packages increases, the complexity will also worsen. To make it less worse and more efficient, a hash function that adapts to the changes can be implemented.

K1. C

When a change of number in trucks eventually occurs, this would not affect the hash table inside the program. Primarily when the hash table only stores packages with information given and is not interdependent with the amount of trucks. Although it is possible that the total distance might change but the hash table is dependent on the number of packages given.

K2. A.

Other data structures I would have used will be the implementation of graphs and trees. Specifically, binary search trees. Binary search trees would allow the packages to be presorted based on specific attributes, and can be quickly accessed by the tree. The graph is somewhat like trees, but graph will allow the packages to be group with adjacent vertices. This will be traversed until the maximum transversal limit of 16 is reached. Both can also adapt to scalability of the given n values.

Screenshots:

1. Provide screenshots to show the status of all packages at a time between 8:35 a.m. and 9:25 a.m.

8:50 AM:

```
Select from the following options:
1. To view by Package ID
2. To view all packages
3. To exit the program
Please enter a time in the format of HH:MM(Ex: 08:30 or 15:10).
PACKAGES STATUS AS OF: 08:50:00
Total Truck Distance Traveled: 15.0 miles
 Package ID: 10 || Package Address: 600 E 900 South || City: Salt Lake City || Package Zip: 84105 || Package Deadline: EOD || Package Weight: 1 || Package Status: AT THE HUB
 Package ID: 20 || Package Address: 3595 Main St || City: Salt Lake City || Package Zip: 84115 || Package Deadline: 10:30 AM || Package Weight: 37 || Package Status: Delivered At: 2021-11-21 08:29:40
 Package ID: 40 || Package Address: 380 W 2880 S || City: Salt Lake City || Package Zip: 84115 || Package Deadline: 10:30 AM || Package Weight: 45 || Package Status: Delivered At: 2021-11-21 08:40:40
 Package ID: 1 || Package Address: 195 W Oakland Ave || City: Salt Lake City || Package Zip: 84115 || Package Deadline: 10:30 AM || Package Weight: 21 || Package Status: Delivered At: 2021-11-21 08:44:20
 Package ID: 11 || Package Address: 2600 Taylorsville Blvd || City: Salt Lake City || Package Zip: 84118 || Package Deadline: EOD || Package Weight: 1 || Package Status: AT THE HUB
 Package ID: 21 || Package Address: 3595 Main St || City: Salt Lake City || Package Zip: 84115 || Package Deadline: EOD || Package Weight: 3 || Package Status: AT THE HUB
 Package ID: 31 || Package Address: 3365 S 980 W || City: Salt Lake City || Package Zip: 84119 || Package Deadline: 10:30 AM || Package Weight: 1 || Package Status: Delivered At: 2021-11-21 08:35:00
 Package ID: 2 || Package Address: 2530 S 500 E || City: Salt Lake City || Package Zip: 84106 || Package Deadline: EOD || Package Weight: 44 || Package Status: AT THE HUE
 Package ID: 32 || Package Address: 3365 S 900 W || City: Salt Lake City || Package Zip: 84119 || Package Deadline: EOD || Package Weight: 1 || Package Status: AT THE HUB
 Package ID: 4 || Package Address: 380 W 2880 S || City: Salt Lake City || Package Zip: 84115 || Package Deadline: EOD || Package Weight: 4 || Package Status: AT THE HUB
 Package ID: 36 || Package Address: 2300 Parkway Blvd || City: West Valley City || Package Zip: 84119 || Package Deadline: EDD || Package Weight: 88 || Package Status: AT THE HUB
 Package ID: 17 || Package Address: 3148 S 1100 W || City: Salt Lake City || Package Zip: 84119 || Package Deadline: EOD || Package Weight: 2 || Package Status: AT THE HUB
 Package ID: 37 || Package Address: 410 S State St || City: Salt Lake City || Package Zip: 84111 || Package Deadline: 10:30 AM || Package Weight: 2 || Package Status: Truck currently in Transit
 Package ID: 8 || Package Address: 300 State St || City: Salt Lake City || Package Zip: 84103 || Package Deadline: EOD || Package Weight: 9 || Package Status: AT THE HUB
 Package ID: 18 || Package Address: 1488 4800 S || City: Salt Lake City || Package Zip: 84123 || Package Deadline: EOD || Package Weight: 6 || Package Status: AT THE HUB
 Package ID: 28 || Package Address: 2835 Main St || City: Salt Lake City || Package Zip: 84115 || Package Deadline: EDD || Package Weight: 7 || Package Status: AT THE HUB
 Package ID: 38 || Package Address: 410 S State St || City: Salt Lake City || Package Zip: 84111 || Package Deadline: EOD || Package Weight: 9 || Package Status: AT THE HUB
 Package ID: 9 || Package Address: 300 State St || City: Salt Lake City || Package Zip: 84103 || Package Deadline: EDD || Package Weight: 2 || Package Status: AT THE HUB
 Package ID: 19 || Package Address: 177 W Price Ave || City: Salt Lake City || Package Zip: 84115 || Package Deadline: EOD || Package Weight: 37 || Package Status: AT THE HUB
 Package ID: 29 || Package Address: 1330 2180 S || City: Salt Lake City || Package Zip: 84186 || Package Deadline: 18:30 AM || Package Weight: 2 || Package Status: Truck currently in Transit
 Package ID: 39 || Package Address: 2010 W 500 S || City: Salt Lake City || Package Zip: 84104 || Package Deadline: EOD || Package Weight: 9 || Package Status: AT THE HUB
```

2. Provide screenshots to show the status of all packages at a time between 9:35 a.m. and 10:25 a.m.

10:00 AM:

****WGU C950 Delivery Program****

```
Select from the following options:
1. To view by Package ID
2. To view all packages
 . To exit the program
Please enter a time in the format of HH:MM(Ex: 08:30 or 15:10)
 Package ID: 31 || Package Address: 3365 S 900 W || City: Salt Lake City || Package Zip: 84119 || Package Deadline: 10:30 AM || Package Weight: 1 || Package Status: Delivered At: 2021-11-21 08:35:00
 Package ID: 2 || Package Address: 2530 S 500 E || City: Salt Lake City || Package Zip: 84106 || Package Deadline: EOD || Package Weight: 44 || Package Status: AT THE HUB
 Package ID: 23 || Package Address: 5100 South 2700 West || City: Salt Lake City || Package Zip: 84118 || Package Deadline: EOD || Package Weight: 5 || Package Status: AT THE HUB
 Package ID: 25 || Package Address: 5383 South 900 East #104 || City: Salt Lake City || Package Zip: 84117 || Package Deadline: 10:30 AM || Package Weight: 7 || Package Status: Delivered At: 2021-11-21 09:08:00
 Package ID: 6 || Package Address: 3060 Lester St || City: West Valley City || Package Zip: 84119 || Package Deadline: 10:30 AM || Package Weight: 88 || Package Status: Delivered At: 2021-11-21 09:40:00
             36 || Package Address: 2300 Parkway Blvd || City: West Valley City || Package Zip: 84119 || Package Deadline: EDD || Package Weight: 88 || Package Status: Delivered At: 2021-11-21 09:45:20
 Package ID: 37 || Package Address: 410 S State St || City: Salt Lake City || Package Zip: 84111 || Package Deadline: 10:30 AM || Package Weight: 2 || Package Status: Delivered At: 2021-11-21 09:13:40
 Package ID: 28 || Package Address: 2835 Main St || City: Salt Lake City || Package Zip: 84115 || Package Deadline: EDD || Package Weight: 7 || Package Status: Delivered At: 2021-11-21 09:25:20
 Package ID: 38 || Package Address: 410 S State St || City: Salt Lake City || Package Zip: 84111 || Package Deadline: EOD || Package Weight: 9 || Package Status: Truck currently in Transit
 Package ID: 19 || Package Address: 177 W Price Ave || City: Salt Lake City || Package Zip: 84115 || Package Deadline: EOD || Package Weight: 37 || Package Status: AT THE HUB
 Package ID: 39 || Package Address: 2018 W 500 S || City: Salt Lake City || Package Zip: 84104 || Package Deadline: EDD || Package Weight: 9 || Package Status: AT THE HUE
```

3. Provide screenshots to show the status of *all* packages at a time between 12:03 p.m. and 1:12 p.m.

1:00 PM:

All Packages Delivered:

```
Color Foliation of State Processing States (Color Processing States Stat
```

L. Sources:

Learn.zybooks.com. (n.d.). zyBooks. [online] Available at: https://learn.zybooks.com/zybook/wGUC950AY2018201
9

[Accessed 5 Nov. 2021].

Tarjan, Robert E. "Edge-disjoint spanning trees and depth-first search"

1976

Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001).

Introduction to Algorithms (Second ed.). MIT