Christopher Brown
Student #000968168
6/7/2020

Greedy Algorithm Overview

Stated Problem:

This project's goal is to find an efficient solution to delivering a set of packages to their destinations within a reasonable number of miles. There are 40 packages currently in the list but the program should be built so that more packages can be delivered in the future at different locations. To solve the problem an algorithm should be identified that will be utilized to deliver the packages. The program will be able to output delivery status at any given time for any given package. The program will also display the final miles traveled and time that the last package was delivered.

Algorithm Overview:

The basic code for the Greedy Algorithm used in the project is created as follows:

1. The starting location for the truck is identified "At Hub" to ensure it is in a location where packages can be loaded.

2. The closest address to the current location is found by comparing the distance from the current location to the location of all of the available packages.

3. The closest address is then appended to the back of the list of loaded packages on that truck and 1 is subtracted from the trucks capacity.

4. The current address is then updated to the address of the package that was just loaded and the process repeated while there is still room on the truck or until all packages are loaded.

5. To deliver the packages are simply popped off of the front of the list and the truck moves to the next location and repeats the process.

The worst case time complexity for this algorithm is O(n^2) and the best case is O(1). The best case scenario is unlikely because the method would have to be called when the truck is not at the hub or the list of packages would have to be empty when the method is called.

Greedy Algorithm

Below is a pseudo code example of the algorithm. (Please note that there are many constraints that complicated the example requiring extra steps that are not documented in the pseudo code. This is to show a basic example of what the algorithm does)

```
available_packages = [1, 2, 3, 4…n]
packages_on_truck = [ ]

def order_and_load(available_packages, packages_on_truck)
        current_address = At Hub
        capacity = 16   # (This is the trucks capacity)

         while requests is not 0 and available packages is not empty:

                package_to_load = available_packages.sort(closest package from the current location)
                packages_on_truck = packages_on_truck.append(package_to_load)
                available packages.remove(package_to_load)
                capacity -= 1
                current_address = package_to_load address
         end
   return packages_on_truck
end


def deliver(packages_on_truck)
        current_address = At Hub
        while packages_on_truck is not empty:
                package_to_deliver = packages_on_truck.pop(0)
                distance = get_distance(current_address, package_to_deliver)
                total_distance += distance
                current_address = package_to_deliver.address
        end
   return total_distance
end
```