C950 Performance Assessment

Greg Henry

Student ID: 001130918

Revisions: The required revisions are at the top of this document and are written in blue

A: Algorithm Overview

1. Guiding Principles
   1. I will determine which packages need to be delivered first based on required delivery times
   2. I will deliver packages to the same address at the same time and will wait if a package for the same address is delayed
   3. I will organize delivery routes by zip code to be efficient
2. Truck One
   1. First, I will load package 15 onto the truck since it has the earliest required delivery time of 9:00 am
   2. Second, I will load other packages in the same zip code as package 15 (zip code 84117) to be as efficient as possible. However, I will not load package 26 since it is located at the same address as package 25 and package 25 has been delayed on the plane.
   3. Third, I will load packages from zip codes 84121 and 84107 since these zip codes are close to 84117. I want to organize delivery by zip codes to be as efficient as possible.
   4. Fourth, I will consider the requirement that packages 13, 14, 15, 16, 19, and 20 must all be on the same truck. I already have packages 14, 15, and 16 on the truck, so I will load 13, 19 and 20. Since the zip code for packages 19 and 20 (zip code 84115) is closer than the zip code for package 13 (zip code 84104), I will first load packages 19 and 20 and other packages from zip code 84115 and then load package 13 and other packages from zip code 84104.
   5. Truck One is now fully loaded with 16 packages and will deliver the packages in the order that they were loaded onto the truck.
3. Truck Two
   1. For truck two, I will focus on delivering a few packages that have the 10:30 deadline as well as others that are close by. I want to make sure that I return around 9:05 to focus on delivering the packages that have been delayed on the plane.
   2. First, I will load packages from zip code 84111 that include package 37 that must be delivered by 10:30 as well as package 38 that is required to be on truck two.
   3. Second, I will load packages from zip code 84103 since it is close to zip code 84111 and includes package 3 which can only be on truck two. I will not load package 9 since it has the wrong address.
   4. Third, I will load packages from zip codes 84105 and 84106 since they are somewhat on the way back to WGU and include package 29 that has a 10:30 delivery deadline.
   5. Fourth, I will return to WGU for the packages that were delayed on the airplane; I want to make sure those packages with a 10:30 delivery deadline (packages 6 and 25) are delivered in time.
   6. Fifth, I will load the packages from zip code 84117 (including package 25) since it is close and then load packages from zip codes 84118, 84119 (including package 6) and 84123. I added packages from 84118 before 84119 because it is on the way to 84119 and that still leaves enough time for package 6 to be delivered by 10:30. I waited for zip code 84123 until last because there wouldn’t have been enough time otherwise for package 6 to be delivered on time.
4. Truck One
   1. After truck one has finished delivering its 16 packages, it will return to WGU and load any packages that are left except for packages that are required to be on truck two. This will include package 9 that now has an updated address.
5. Truck Two
   1. After truck two has finished its deliveries for those packages that are required to be delivered by 10:30, it returns to WGU and delivers any packages that are required to be on truck two

B1: Logic Comments

My logic for this project is based on the following three principles: 1) I will determine which packages need to be delivered first based on required delivery times; 2) I will deliver packages to the same address at the same time and will wait if a package for the same address is delayed; and 3) I will organize delivery routes by zip code to be efficient. In the algorithm, I decide what order to load the packages onto the trucks based on these principles. I always want to make sure I deliver packages on time but also be efficient in delivering packages with the same address together and send trucks to certain zip codes based on proximity and on the direction the truck is headed.

B2: Space-Time

Each block of code uses a for-loop to iterate through the array and select those packages that match certain criteria. Each for-loop has an operating time of O(n) so the time complexity is O(n). If a block of code has six for-loops, then the operating time is O(6n). However, since O(6n) is linear, it resolves to O(n). Each block of code in the program resolves to O(n). The overall operating time of the program is O(29n + C), which is linear and resolves to O(n).

The array has 40 Package objects so the space used does not change. Memory is allocated for the array of 40 Package objects and additional memory is not needed.

B4: Adaptability

My approach to this project does not scale or grow very well. I sifted through the requirements and created an approach based on my guiding principles—1) I will determine which packages need to be delivered first based on required delivery times; 2) I will deliver packages to the same address at the same time and will wait if a package for the same address is delayed; and 3) I will organize delivery routes by zip code to be efficient. However, if a new set of 40 packages were provided, the code would not be able to deliver those packages efficiently. The packages would get delivered but the code would have to be altered to do it efficiently and make sure that all deadlines and requirements are met.

If I was given a situation where 4,000 packages needed to be delivered, my code would not be very efficient. Once again, I would need to alter the order of the loading and the order of the decisions to meet all of the deadlines and requirements.

B5: Software Efficiency and Maintainability

The software is efficient because I thought through an efficient solution based on my guiding principles—1) I will determine which packages need to be delivered first based on required delivery times; 2) I will deliver packages to the same address at the same time and will wait if a package for the same address is delayed; and 3) I will organize delivery routes by zip code to be efficient. However, it would not be efficient for a new set of 40 packages would different requirements.

The code is pretty maintainable in the sense of being modular. I have created a Package class with a constructor and functions that can be used for any situation. I have also created functions to determine distances between addresses, convert from miles to time, insert a Package into the hash table, and look-up a Package from the hash table. The decision-making part of the code is broken down into loading the trucks, delivering the packages, and printing the status of the packages at certain times. These blocks of code are modular but would need to be updated when given new information and requirements.

B6: Self-adjusting data structures

As mentioned previously, the time complexity for each for-loop to iterate through my array data structure is O(n). The only searches I do in this particular project are searches for packages with specific criteria using a for-loop. So, the overall impact to running time is O(n) for each for-loop, which is linear. If I had 4,000 packages, the running time would increase but would still be linear with a running time of O(n).

K1a: Efficiency

I used an array of Package objects for this project. As mentioned previously, using a for-loop to iterate through my array is O(n). I did searches to find packages with specific criteria to load onto the trucks. An array was efficient to find these packages, and load them onto the trucks.

K1b: Overhead

As mentioned previously, the computational time is O(n) when using for-loops to iterate through an array to find packages with specific criteria. The program allocates enough memory for an array of 40 Package objects. The bandwidth is sufficient for the searches done and the complexity of this project.

L: Sources

zyBooks learning materials from this course, C950

Hash Table help: <http://blog.chapagain.com.np/hash-table-implementation-in-python-data-structures-algorithms/>

Stated Problem: Deliver 40 packages on time and efficiently

Algorithm Overview:

1. Determine which packages need to be delivered first

2. Deliver packages to the same address at the same time (wait if a package is delayed)

3. Organize delivery routes by zip code to be efficient

Operating time: O(n) for the blocks of code to determine which packages to load. Several for loops are used along with if statements to iterate through the array of Packages and choose the packages to be loaded into each truck.

Ability to adapt: My solution would work for other situations but is specifically efficient with the requirements and constraints of this problem. It could be scaled fairly easily.

Efficiency: My code is efficient as all the packages are delivered by 10:55 and in 103.2 miles.

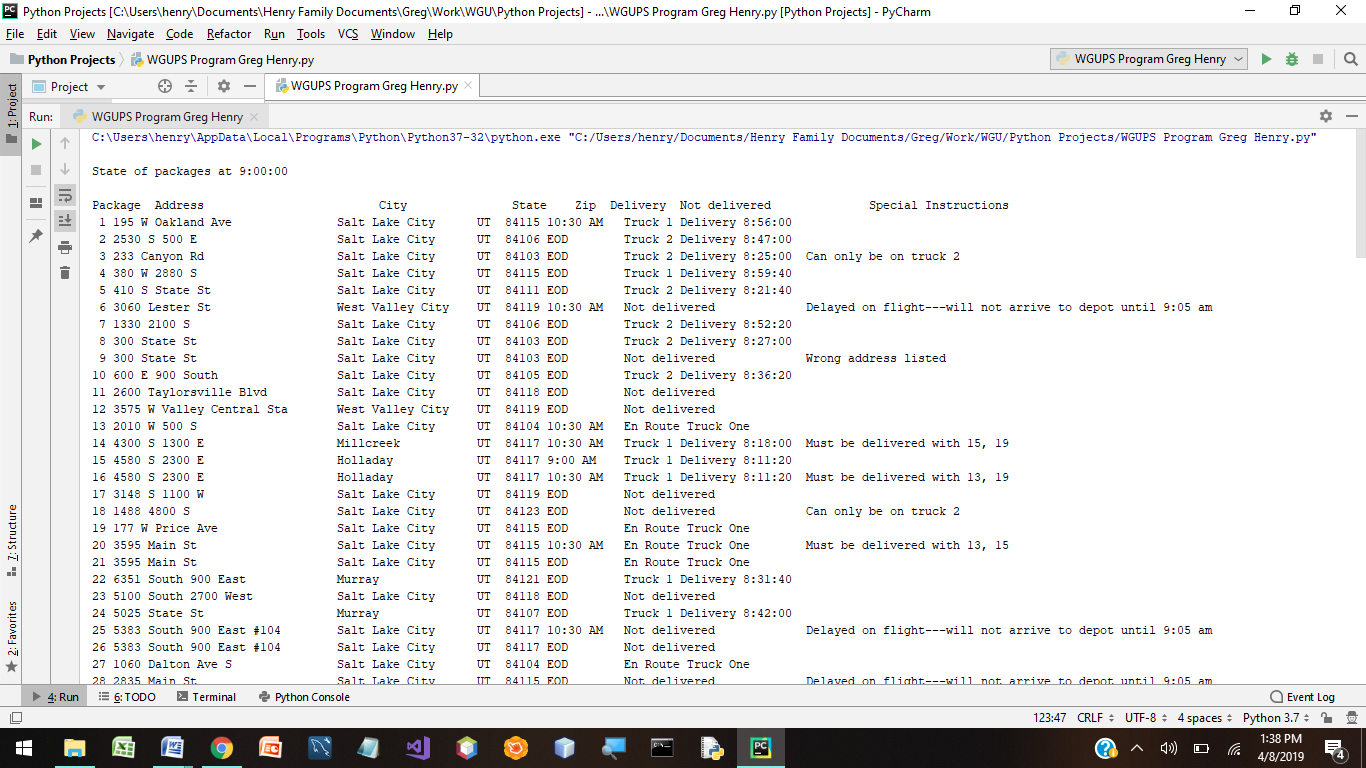
Self-adjusting data structure: I used an array to store my Package objects. Arrays are easy to use and adjust as needed. The weakness would be the need to iterate through all the packages to find the ones to load on the trucks.

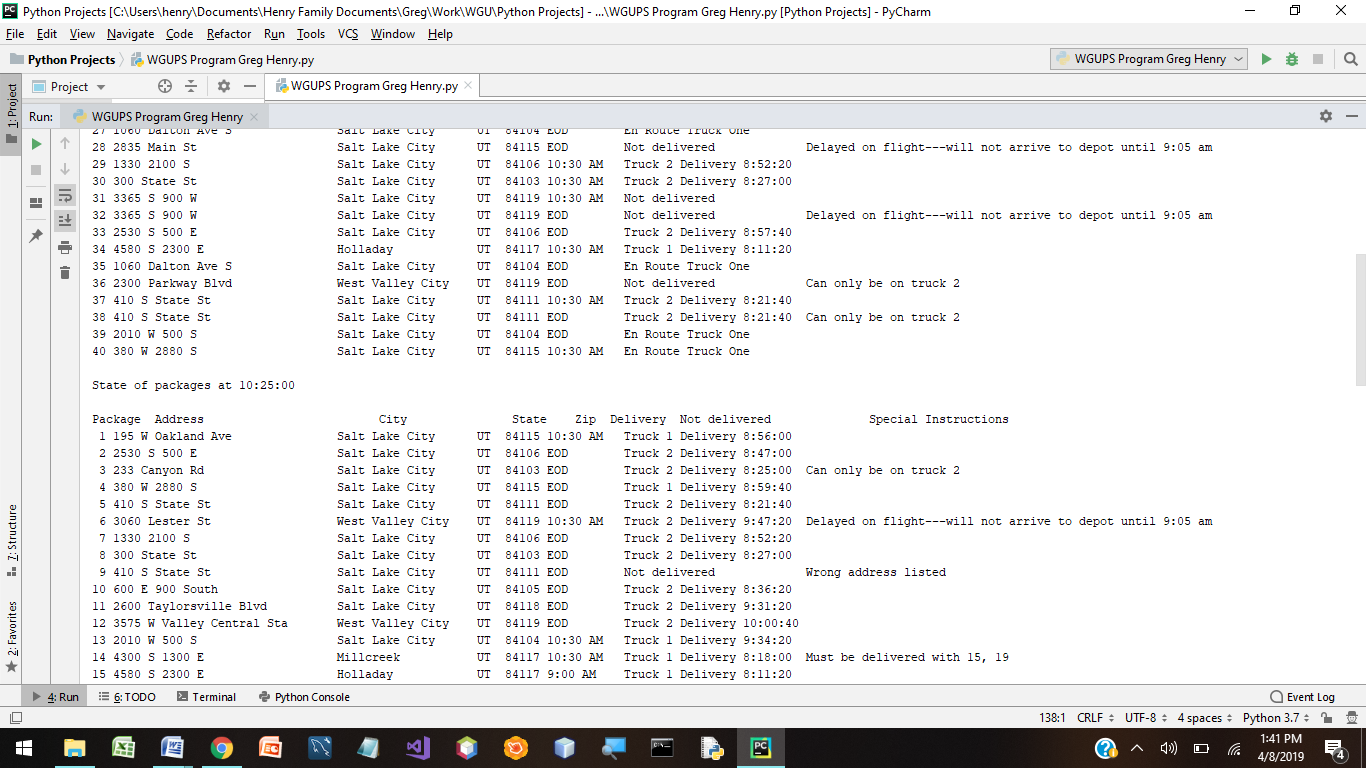
Hash Table with Insertion Function: Lines 72-80 in code

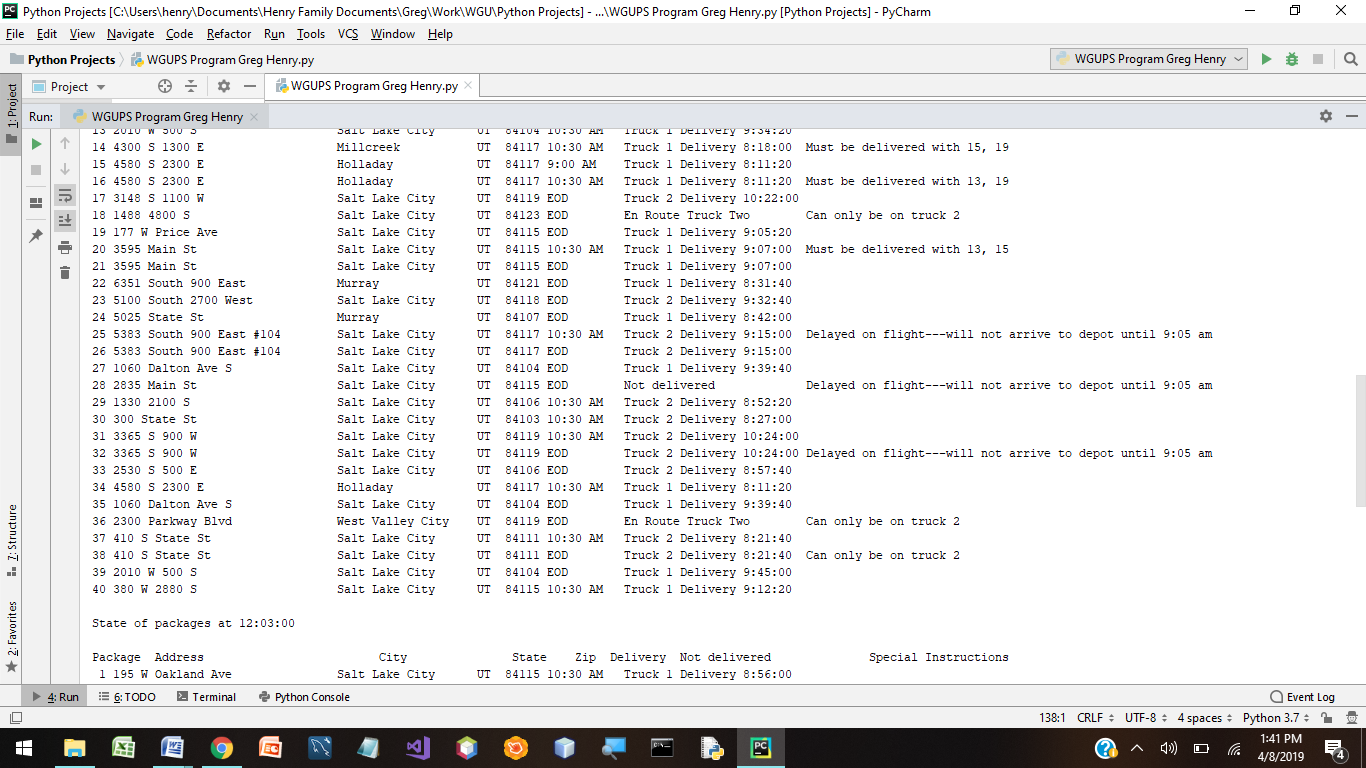
Look-up Function: Lines 82-83 in code

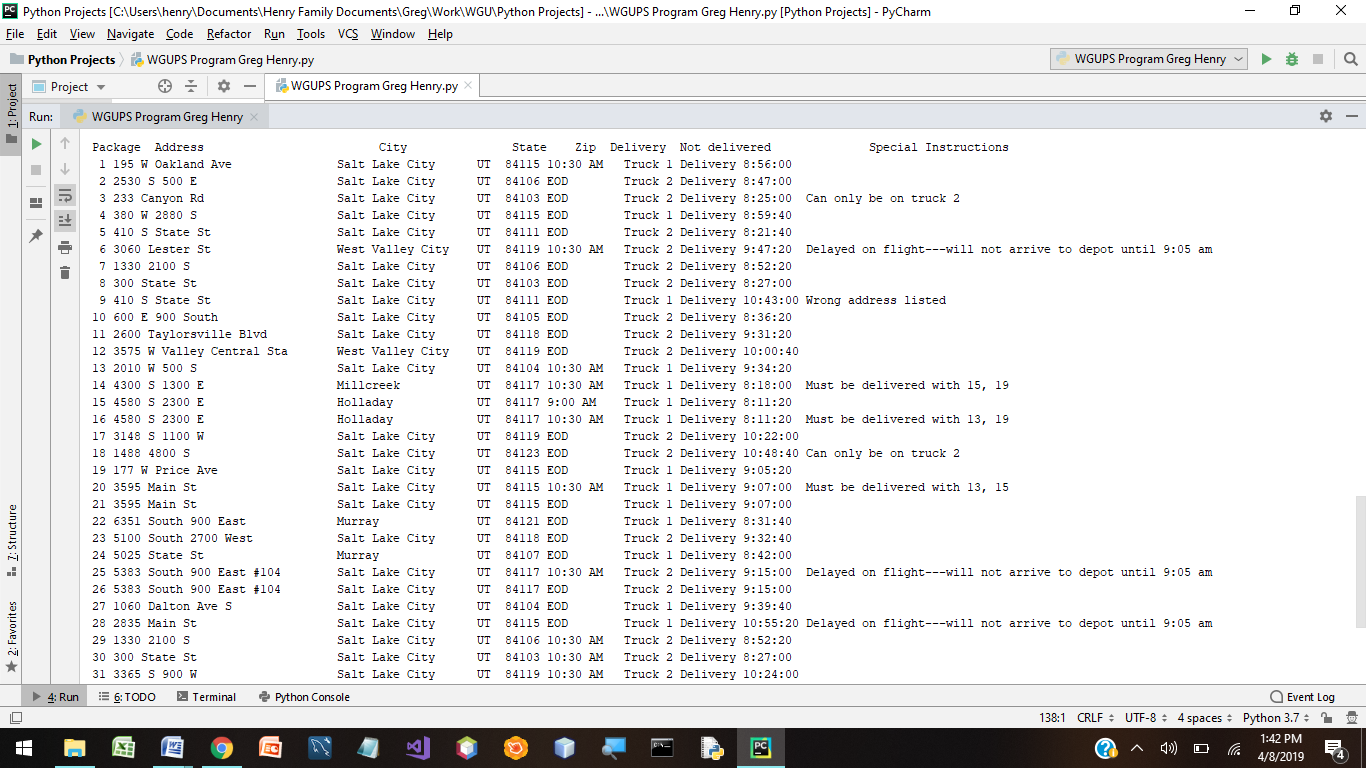
Interface to show package info:

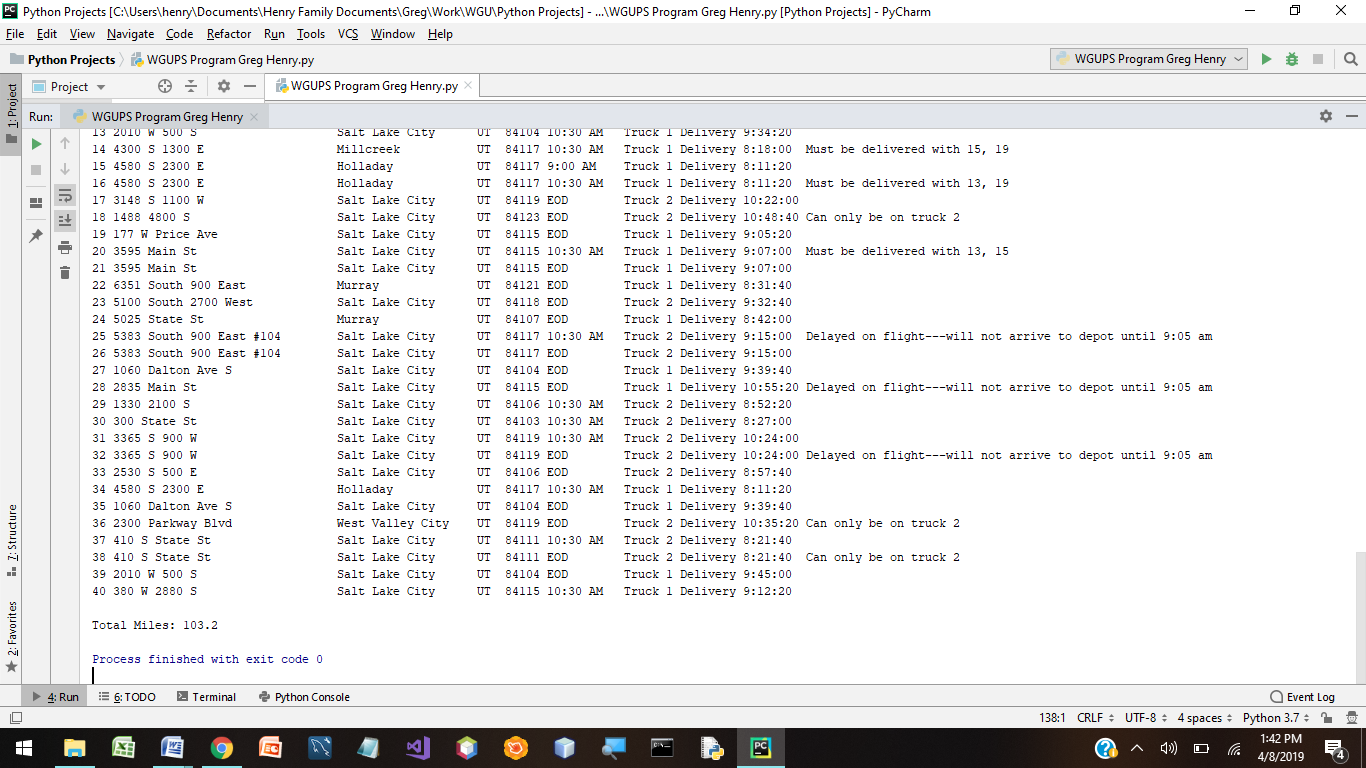
Screenshots at 9:00, 10:25 and 12:53











**Two strengths: My algorithm is strong because I focus on delivering packages to the same address at the same time and I also focus on delivering packages to the same zip code.**

Two other algorithms:

1- I could have sent truck one with all the priority packages without taking packages in the same zip code as well. This would have been different from my approach.

2 – I could have calculated the next closest package and used a heap to prioritize the packages that way. This would have been different from mine because I didn’t worry about it being the exact closest. That would have required more calculations and searches. I just put packages of the same zip code together and delivered them in the order they were loaded.

What I would do differently next time: Next time, I think I would use the third truck as well. I would assign the delayed packages to the third truck along with any packages of the same address. Then, I would load truck two with packages that had to be carried on that truck along with others of the same zip codes. Then, I would load truck one with the remaining packages and start delivering. Each truck would be prioritized to the packages that had the time deadline.