# Algorithm Overview

# Data Structures and Algorithms II – C950

Brandon Smith

A: Algorithm Identification

The self-adjusting algorithm used to deliver all packages is called packageDelivery and is located in Main.py file.

B1: Logic Comments

packageDelivery(self)

if truck(number=self)

for package in truck

initialize enroute start time

List of undeliveredpackages = self

Initialize shortestPath = infinite

While undeliveredpackages is not empty

#First pass of determining shortest path

if undeliveredpackages size == self. Size

for each package in undeliveredpackages

if currentaddress to package address < shortestPath

update shortestpath

package address = next delivery address

#Determining second and up to last path

else: for each package in undeliveredpackages

if currentaddress to package address < shortestPath

update shortestpath

package address = next delivery address

remove package from undeliveredpackages

update currentaddress

update package deliveryStatus = delivered

calculate traveltime, travelMileage

update global totalMileage of all trucks

set shortestPath to infinite

if self = truck(#)

update truck(#) travelTime

updated truck(#) travelMileage

B2: Development environment

I used PyCharm 2021.2.1 (Community Edition) IDE with Python 3.9 and windows operating system on laptop hardware.

B3: Space-Time and Big-O

Noted in comments before each section of code. The entire program runs in N\*N time complexity and N space complexity.

B4: Scalability and Adaptability

The algorithm has the ability to scale and adapt to the number of packages, trucks, and addresses. The hash table class can be scaled and adaptive to more or less packages. In Hashtable class the beginning capacity argument can be changed to allow more or less packages stored efficiently. The number of trucks can be adjusted by adding more truck list objects and putting them through package delivery function. If more addresses are needed, the program can read-in more addresses and store them in larger 2D array. The overall time of program would run longer due to an increase of trucks and packages.

B5: Software Efficiency and Maintainability

The software runs in a short period of time and is consistent. It is maintainable due to the fact each section is labeled and easy to read.

B6: Self-Adjusting Data Structures

The hashtable correctly stores multiple packages and handles collisions using chaining. Weaknesses would be if hashtable becomes uneven or ununiform distribution and could possibly increase operating times.

C: Original Code

All code is original and error free.

C1: Identification Information

Information in first line of main.py.

C2: Process and Flow comments

Comments are posted above each code section.

D: Data Structure

The creation of Hashtable class is used to store the package data with functions to manipulate data.

D1: Explanation of Data Structure

The Hashtable class is used to store the package data. The packages are chained together by using Node class to handle collisions in each bucket. The hash function is set to use 10 buckets to store the objects. It can be modified if number of packages grows or shrinks in the code for the hash function.

E: Hash Table

Hashtable has insert function associated to class.

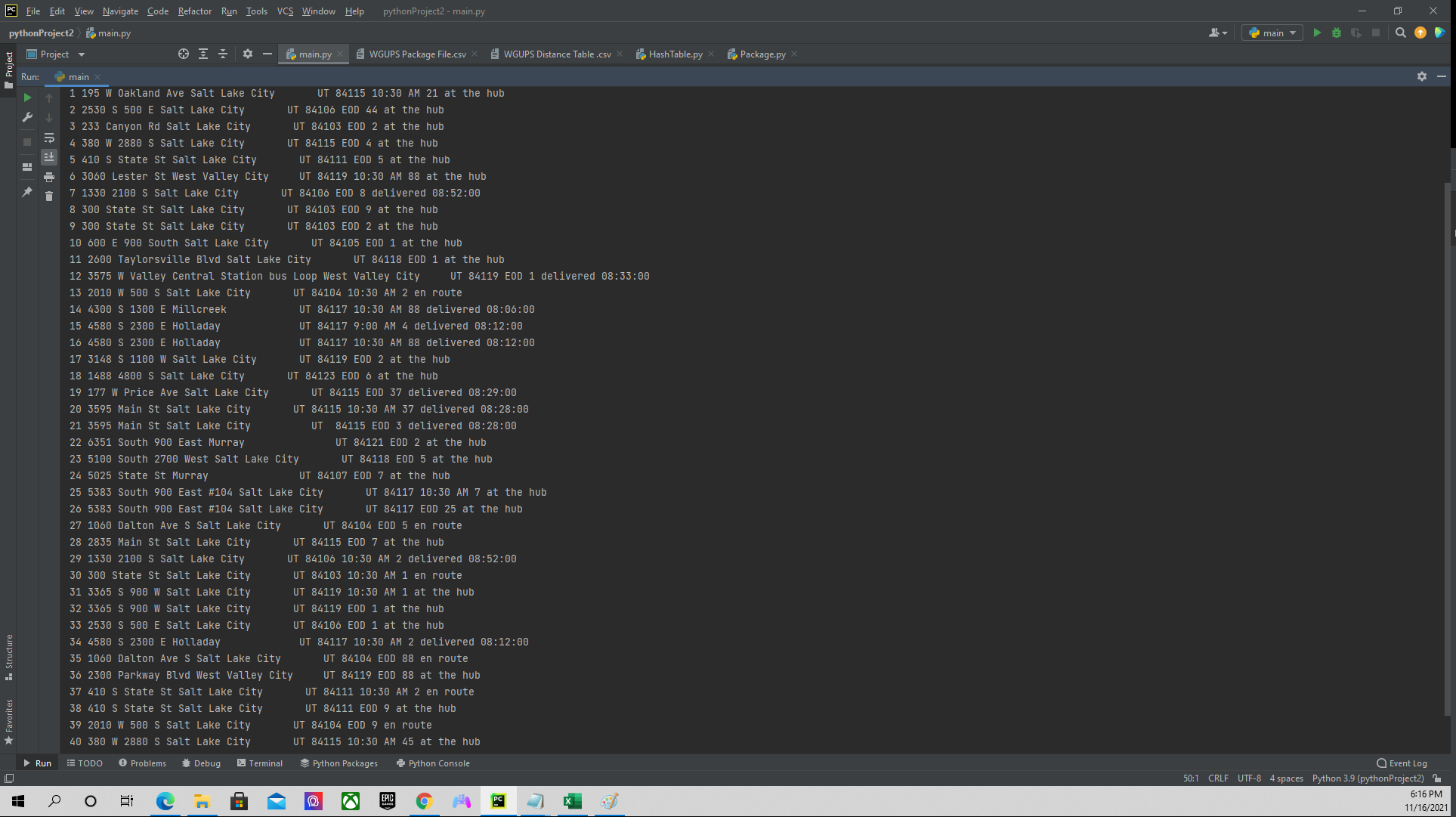
F: Lookup Function

Hashtable has find function to lookup objects.

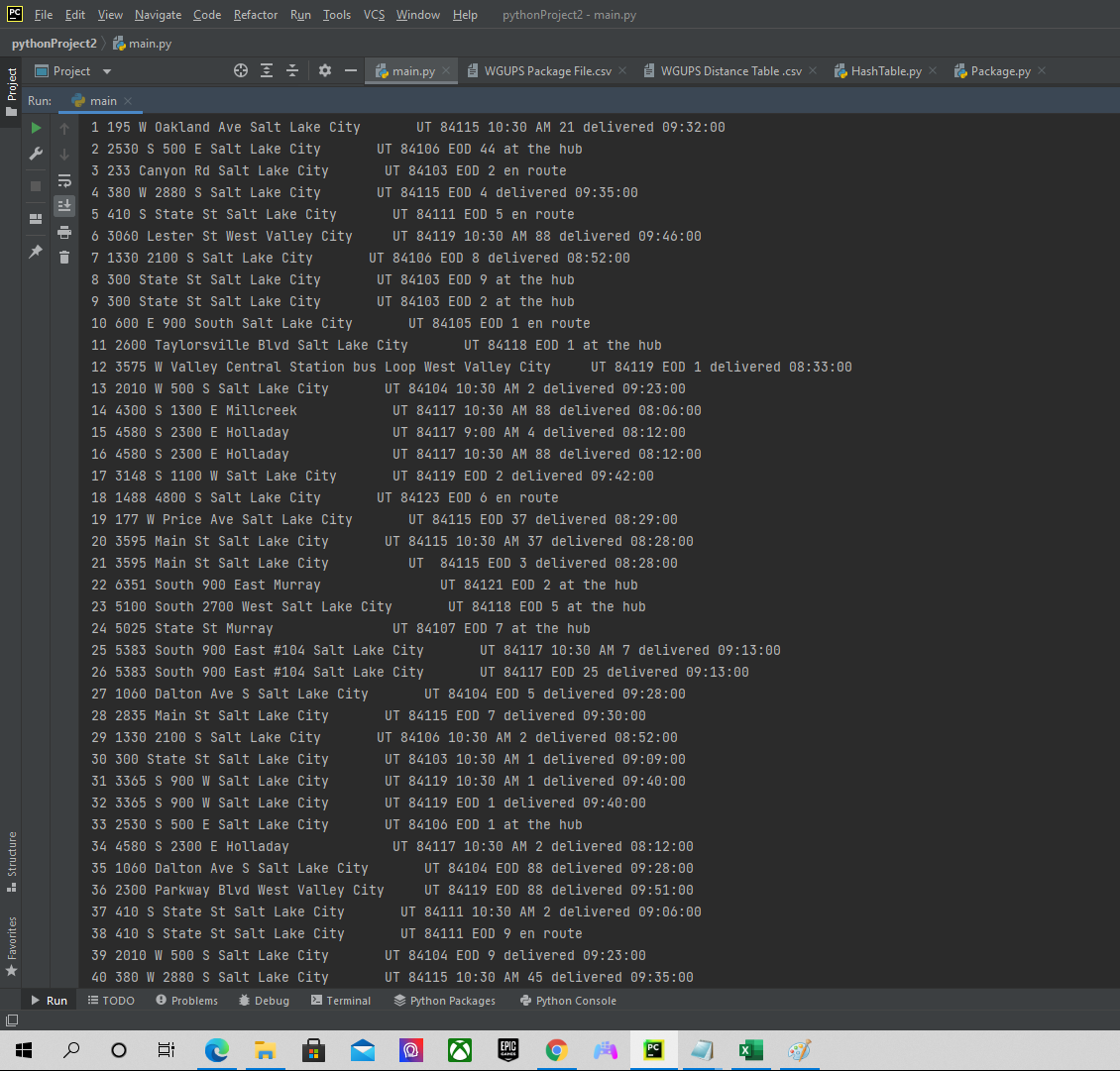
G: Interface

Console interface is provided for user interface.

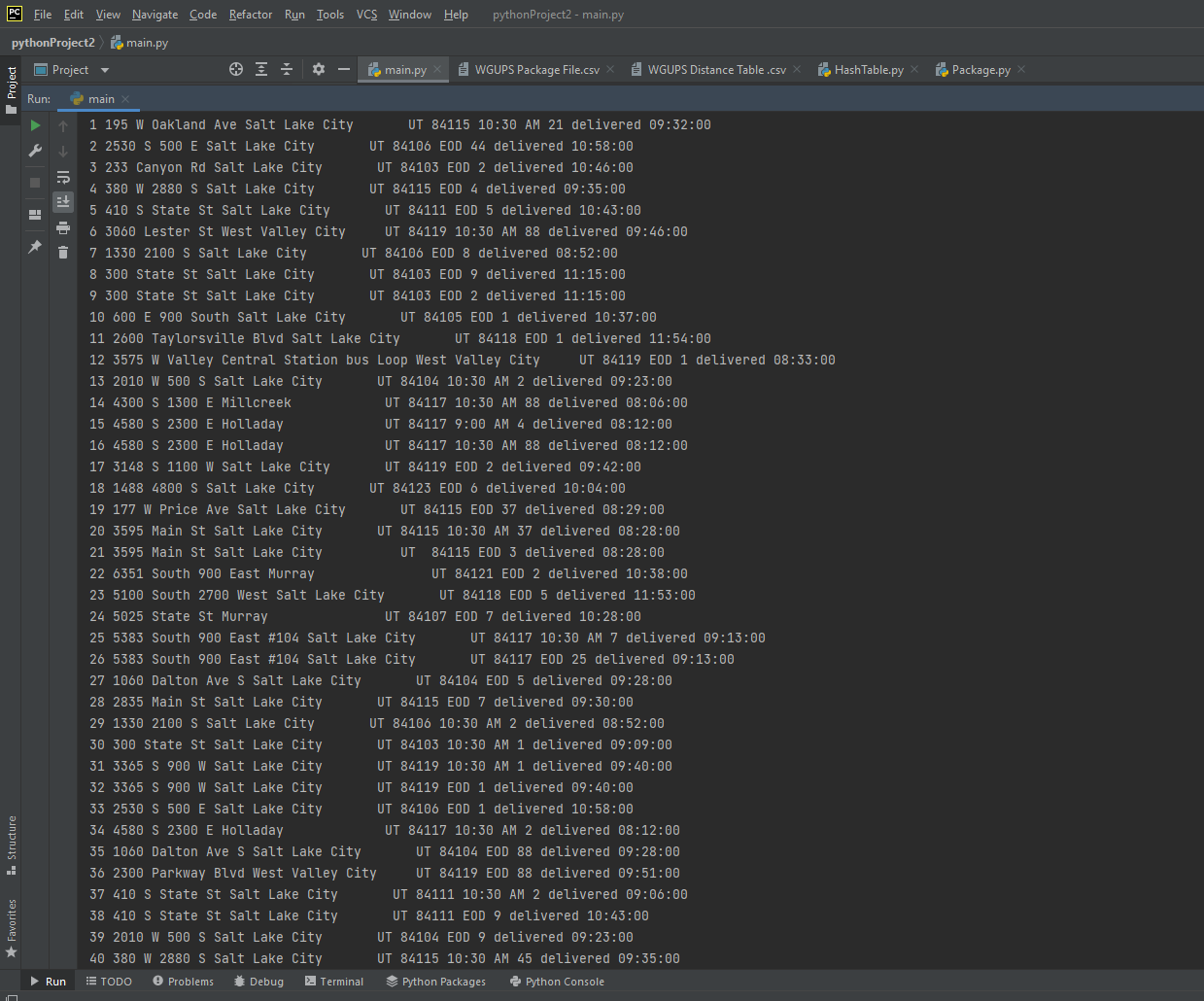
G1: First Status Check @9:00



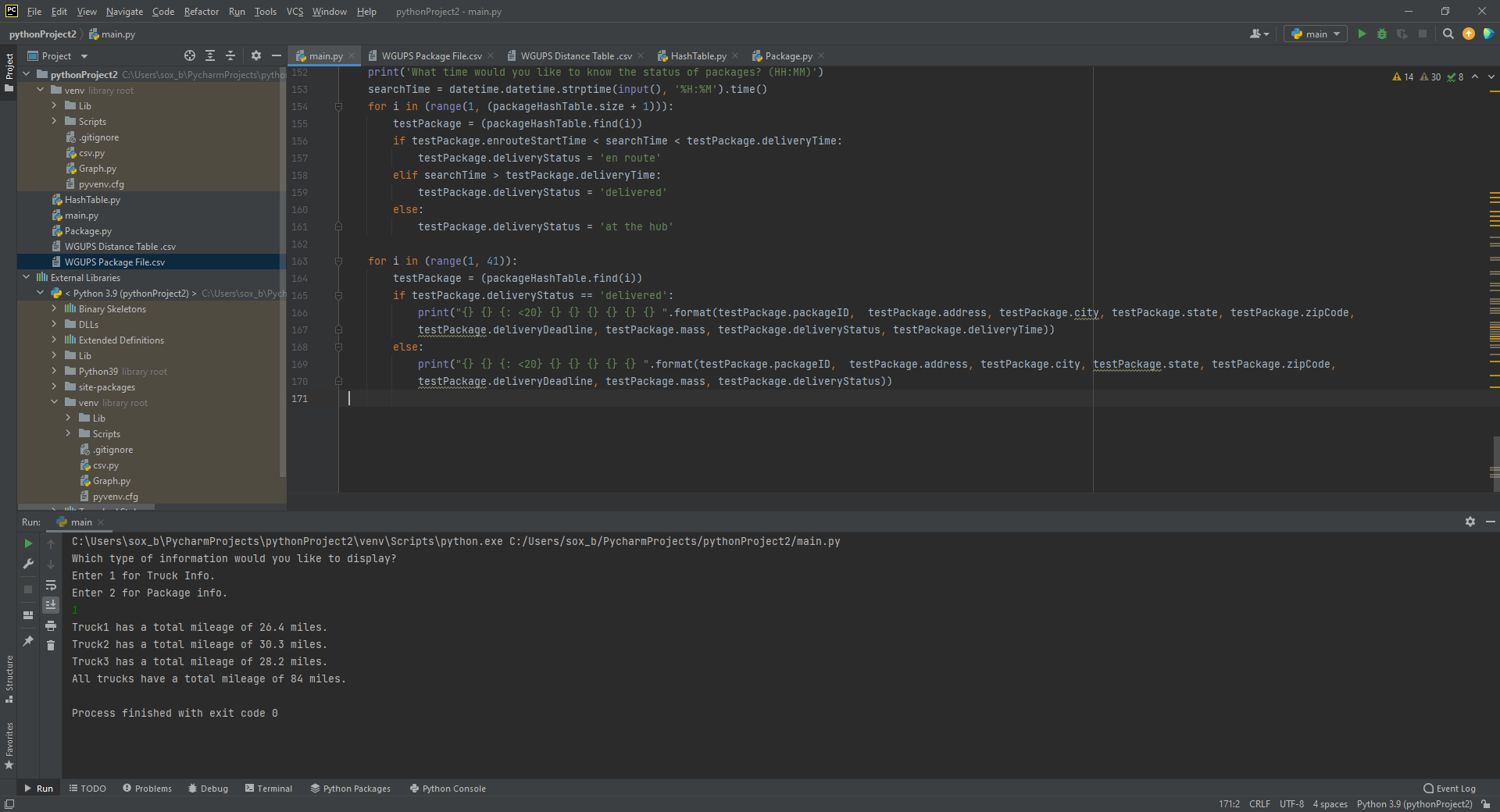
G2: Second Status Check @10:00



G3: Third Status Check @12:45



**H:SCREENSHOTS OF CODE EXECUTION**



I2: Verification of Algorithm

The algorithm delivers all packages on time and is under 140 miles total.

I3: Other Possible Algorithms

Other possible algorithms include Dijkstra's algorithm and Prym’s algorithm.

I3A: Algorithm Differences

Dijkstra's algorithm is used to find the shortest distance between all points in a graph. Dijkstra's would work best if the starting point was not the ending point as was not the case in the scenario. Prym’s algorithm works if all vertices are connected. It would work as well but require more space as you have to update each Node in the graph with a weight compared to algorithm used.

J: Different Approach

I might would change the data structure to store address into a graph structure for navigating through the algorithm.

K1: Verification of Data Structure

Data structure meets all requirements of scenario. It includes total miles of all trucks, with packages delivered on time, with hash table lookup function, and all reports are generated through console interface.

K1A: Efficiency

The number of packages could increase the lookup time if packages are not consecutive numbers. It could turn into linear lookup time if hashtable becomes not uniformly distributed. The capacity would need to be increased and hash function modulo operator increased to counteract the nonuniformity.

K1B: Overhead

If the number of packages changes the space would need to be increased accordingly.

K1C: Implications

If the number of cities increases the lookup time would still run-in same time complexity and increase storage complexity.

K2: Other Data Structures

Possible other data structures include dictionaries and 2-D arrays instead of hashtable data structure.

K2A: Data Structure Differences

Dictionaries use key- value pairs to lookup information. Dictionaries are similar to hashtable but does not use the hash function when accessing information to manipulate. 2-D arrays are basically a list of lists that store information. 2-D arrays also do not use a hash function when accessing information to manipulate.

L: Sources

No sources were used for project.

M: Professional Communication

Content is organized and straight forward with appropriate comments to ensure readability and maintainability.