Elene Karangozishvili and Trang Le

Professor Xia

CS 150-02: Data Structures and Algorithms

December 07, 2018

Logistics Management Project Report

Introduction

The premise of the project is implementation of schedule optimization for a coffee shop delivery based on given routes and allowed capacity. The goal is to minimize the total distance traveled between the cities by all trucks, excluding the distance traveled within each city (Xia, 1).

Optimally solving the problem in general is challenging (Xia), so an alternative approach, most likely an algorithm, should be utilized as the solution.

Approach

A program was designed to read given input files, simulate the logistics of the delivery process, and write an output file of the results.

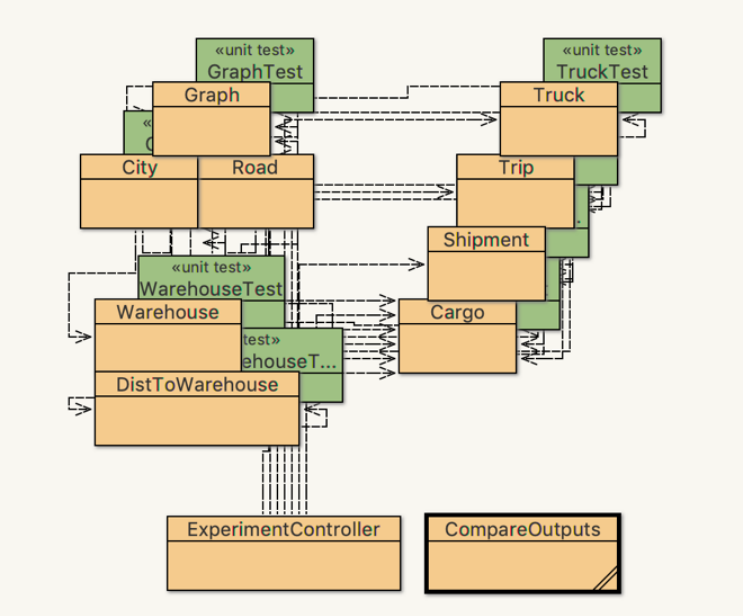

Figure 1. Class diagram of the program

Object-oriented programming was the principle behind the design of the project. For simulation of the cities and roads, we implemented undirected graph (Xia) as the data structure, where cities were vertices and roads were edges. The adaptation of class `Vertex` to class `City` was modified to include a variable linking to a warehouse. Classes `Cargo`, `Shipment`, `Trip`, `Truck`, and `Warehouse` were designed as simulations of objects involved in the delivery process: class `Cargo` kept track of the weight and ID of each cargo; class `Shipment` had an `ArrayList<Cargo>` (*ArrayList (Java Platform SE 8)*) of all the cargos delivered to the same warehouse within one trip; class `Trip` had an `ArrayList<Shipment>` of all shipments in the same trip; class Truck had an `ArrayList<Trip>`, the location, and the maximum capacity; class `Warehouse` contained a link to a city, a `PriorityQueue<Cargo>` (*PriorityQueue (Java Platform SE 7)*), since cargos are loaded onto trucks in ascending order of weight, and `ArrayList<DistToWarehouse>`, which kept track of the shortest distances to other warehouses. Lastly, class `ExperimentController` class read given input files, processed the simulation, and wrote the results to an output file.

In terms of theoretical approach, Dijkstra algorithm (Xia) was used to find the shortest paths from each vertex on undirected weighted graph because all the edges (length of roads connecting cities) had positive values. Furthermore, optimal solving of the problem is hard, so the “greedy” approach using Dijkstra was chosen to find an approximate solution.

Methods

The program was run on two given sets of data, one small and one large. For each data set, an output file was written with class `PrintWriter` (*PrintWriter (Java Platform SE 7)*) to record the trucks, the trips, the cargos and the distance covered in each trip as processed by the program.

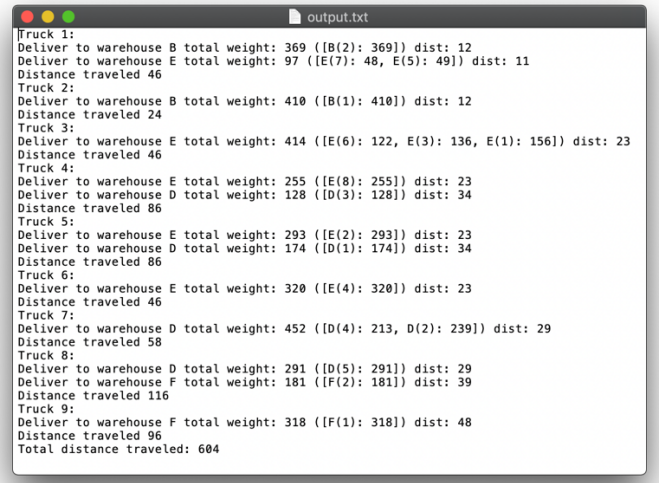

Figure 2. The output file written by the program based on the small data set

Apart from the main program that ran the experiments, JUnit test classes were also created for the purpose of testing self-written methods (Figure 3, 4, 5).

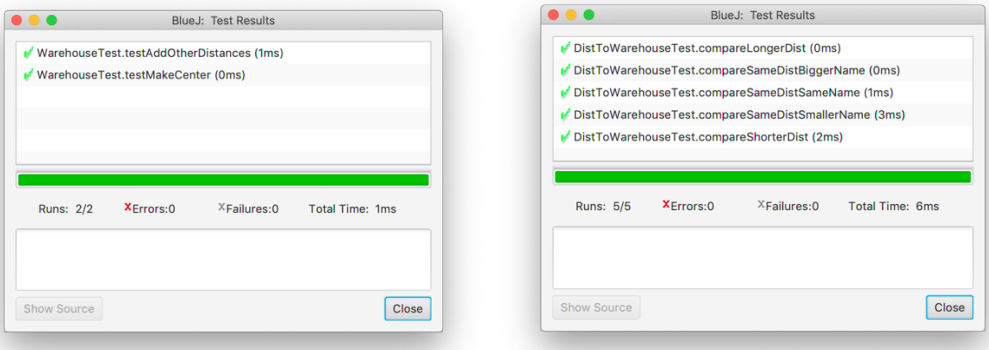

Figure 3. Unit tests for classes `Warehouse` and `DistToWarehouse`

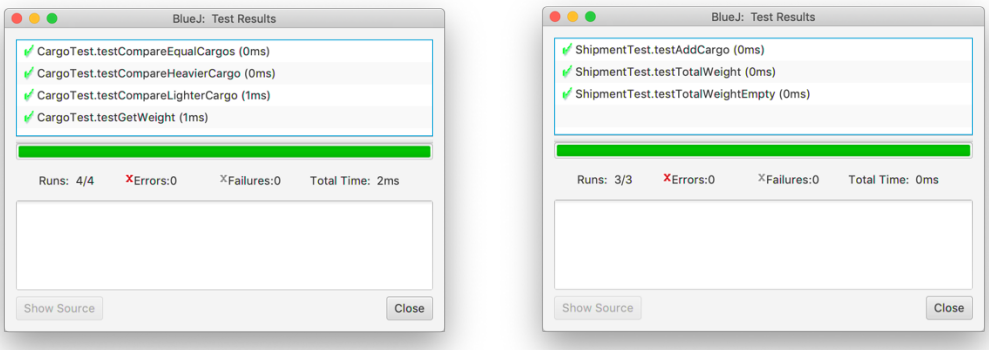

Figure 4. Unit tests for classes `Cargo` and `Shipment`

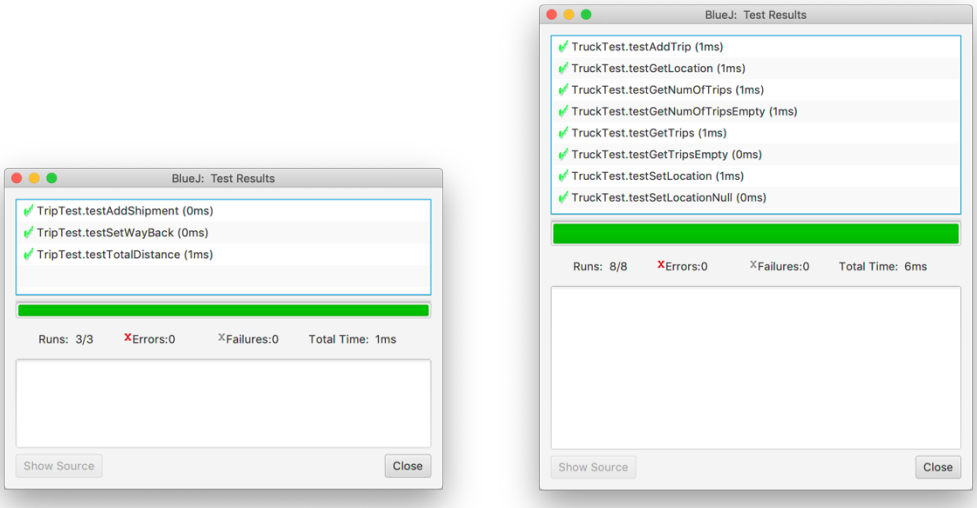

Figure 5. Unit tests for classes `Trip` and `Truck`

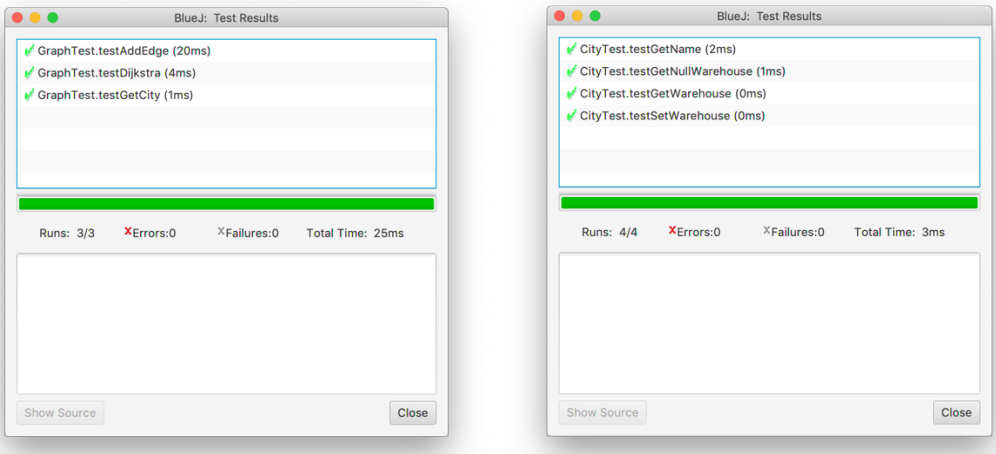

Figure 6. Unit tests for classes `Graph` and `City`

For class `Graph` and `City`, unit tests were written to test generation of graph elements as well as the accuracy of Dijkstra algorithm (Figure 6). Class `Road` only had a constructor, so no unit testing was necessary.

Data and Analysis

The final tally of total distances traveled calculated by Dijkstra was 604 and 8846, respectively, for the small input set and the large input set.

From the terminal, `diff` command was run to compare the program's generated output file with the provided sample output. Since the command line returned no difference, it can be seen that the content of the two files matched up completely (Figure 7).

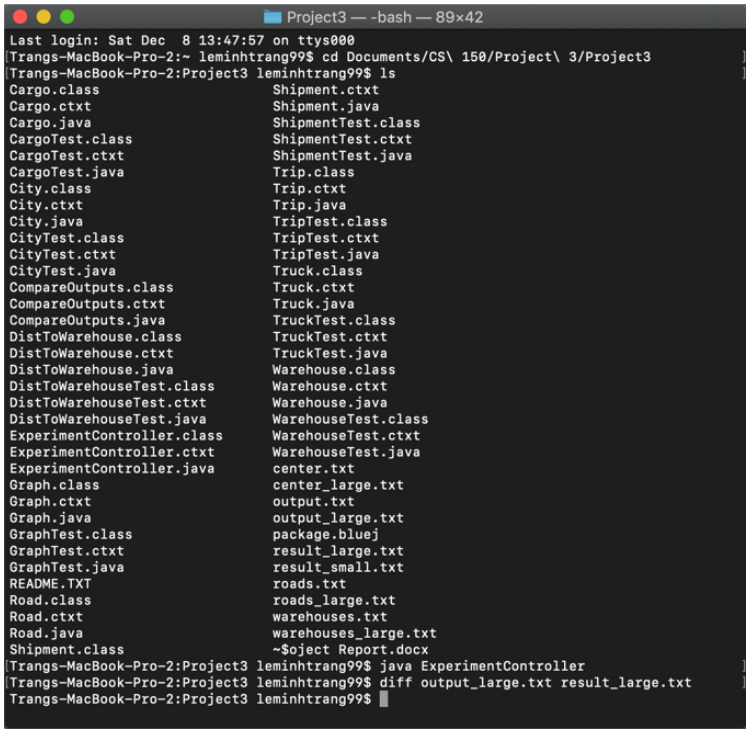

Figure 7. Running the `diff` command from terminal to compare the output and the sample

The run time of Djikstra algorithm is $O(|E|\log|V|)$ on average.

Conclusion

Even though the routes created by Dijkstra might not have been the absolute most optimal and ideal way to execute the delivery, the algorithm still provided a relative efficient approach to solving the problem. Furthermore, the average run time of Dijkstra is $O(|E|\log|V|)$ if all vertices are reachable from the starting vertex. Even when the graph is dense, the performance is $O(|V|^2)$, which is acceptable because that means Djikstra runs in linear time on the number of edges ($|E| = |V|^2$) (Xia).

References

ArrayList (Java Platform SE 8).

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html. Accessed 20 Nov. 2018.

PrintWriter (Java Platform SE 7).

https://docs.oracle.com/javase/7/docs/api/java/io/PrintWriter.html. Accessed 14 Oct. 2018.

PriorityQueue (Java Platform SE 7).

https://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html. Accessed 14 Oct. 2018.

Xia, Ge. *Graph.Java*.

https://moodle.lafayette.edu/pluginfile.php/437735/mod_resource/content/2/Graph.java. Accessed 8 Dec. 2018.

Xia, Ge. CS 150: Project III. 2018.

Xia, Ge. CS 150 Lecture. November 28, 2018.