Agathe Benichou
CS 150
Professor Liew
Due: December 18th 2015

Project 3 Final Report

1 Introduction

To conclude the wonderful class that has been CS150, Project 3 extends the past two projects which have all been about the Eastons Farmer Market. In Project 3, customers of the Farmers Market can now have their goods delivered to them instead of having to pick it up at the Farmers Market. This way, the vendors do not have to wait each week to find out how many customers want to purchase their goods. The vendors can now have scheduled deliveries to customers to keep their incomes and list of customers stable. Given to us are three .txt files that contain the profit per unit of each good, the locations of each vendor's warehouses, and an interconnection graph that shows the connecting roads between cities and the distance between the cities. Each vendor has different warehouse locations within the cities. The goal of this project was to find the shortest tours that a vendor has to deliver his goods to customers. [1]Similar to the well known Vehicle Routing Problem, the shortest tour should take into account all the different warehouse locations for the vendor and from each warehouse, find the shortest tour to cities close to that one warehouse location.[2] This avoids having to go to all cities in one tour, which will not make the tour very short. However, I chose to do a simplification of the project which is that for each vendor, only have one warehouse location. This makes this Project more similar to the Traveling Salesman Problem because it has one starting point and it must go to all the cities it needs to go to from that one starting point and then return back to the starting point.[3] Some assumptions I made are that there is only one warehouse location for each vendor and that if the cost of going to a city is greater than the revenue you will make delivering your goods in the city, it is not worth traversing to that city. Although there is major room for improvement within my algorithm, it does find the shortest path from city to city using Dijkstra's Algorithm and it does return the guaranteed shortest path which is specific to my program setup. I hypothesize that the time complexity for my algorithm will be fast for a small amount of data points in a graph and it will be slower for a large amount of data points in a graph because my algorithm mainly uses Dijkstra's algorithm. Dijkstra's algorithm takes O(E log V) time for sparse graphs and $O(V^2)$ time for dense graphs where E is the number of Edges and V is the number of cities.[4] This will change with the

---

[1] Liew, Chun Wai. "CS150: Project 3." (n.d.): n. pag. *Moodle*. Lafayette College, 22 Nov. 2015. Web. 22 Nov. 2015.

[2] "Vehicle Routing Problem." *Vehicle Routing Problem*. NEO, 18 Jan. 2013. Web. 18 Dec. 2015.

[3] Weisstein, Eric W. "Traveling Salesman Problem." From *MathWorld*--A Wolfram Web Resource.

[4] Weiss, Mark Allen. *Data Structures & Algorithm Analysis in Java*. Reading, MA: Addison-Wesley, 1999. Page 567.

way I am implementing Dijkstra's algorithm in my code which is explained in the Analysis section of this report.

2 Approach

A majority of my classes and data structures are from Project 2. Starting with the classes from Project 2:

- The Needs class is used to create Need objects that are randomly assigned to Customers to create their list of needs.
- In the GenerateListOfNeeds class, Need objects are selected using a Gaussian distribution and are stored in an Array List to create a list of needs.
- The Gaussian Class contains the code for generating a number given a mean and a variance number. This class is called every time a Customer is created and needs a list of needs.
- The NeedContainer class contains a Linked List of all six of the Need objects possible for Customers to have on their list. The NeedContainer class is frequently used to help fill other larger data structures.
- The Customer class is used to create Customer objects that will be assigned an ID number, a list of needs and a city that the Customer resides in.
- The CustomerContainer class stores the Customer objects into a Linked List. The CustomerContainer class is used to fill the larger data structures that are used to in my shortest tour algorithm. The CustomerContainer class calls the CustomerGenerator class 10 times and which creates about 200 customers each time.
- The CustomerGenerator class generates about 2000 customers in total. This class calls on the GenerateListOfNeeds class to generate a list of needs for each customer object created. The CustomerGenerator class also calls on the CityContainer class.
- The CityContainer class is a Linked List of all cities in the connectivity graph to randomly assign Customer objects to cities in the graph. is filled with the citymap.txt file provided to us by the instructor which contains a list of all cities and the connecting roads and weights of the roads between the cities. Every city that is read from the txt file is created into a City object and stored in the CityContainer class. The CityContainer is used to fill larger data structures used in the algorithm for the shortest tour.
- The City class is used to create City objects that are given an integer ID number, this number is the number read from the citymap.txt file.
- There are three .txt files that are read and whose information is stored inside class and containers. One of those files is the citymap.txt file uses the GraphGeneratorContainer class to create a connectivity graph. The GraphGeneratorContainer class calls the UnDirectedGraph class to add cities and to add roads with a distance between them.
- The UnDirectedGraph class contains the methods to create a graph which is stored as a Hash Map whose keys are integers and whose values are UnDirectedGraphNode objects. The UnDirectedGraphNode and UnDirectedEdge inner class are within UnDirectedGraph Node and they are used to create the graph. The UnDirectedGraphNode inner class

represents the cities and the UnDirectedEdge inner class represents the roads between the cities with distances between them. Dijkstra's algorithm is within the UnDirectedGraph class and the algorithm uses the Path class to find the shortest path. and it returns an Array List of all the nodes (each node is a city and each city contains a reference ID number, the Array List is a list of integers) within the calculated shortest path.

- The second file is the goods.txt file which contains the profit per unit for each good. This information is stored in the ProfitPerUnitContainer class which uses a Hash Map whose keys are Strings of all the goods and whose values are the profits read from the goods.txt file. This container is used to calculate the total revenue a vendor makes at a city selling his goods.
- The third file is the locations.txt file which contains all the locations of all the warehouses within the connectivity map for all the goods. This file is read and the information is stored in the LocationsContainer class which consists of a Hash Map whose keys are all the goods that vendors sell and whose values are Linked Lists of warehouse locations.
- The ItemCityCustomers class contains two of the large data structures that play a big part in finding the shortest tour and which are dependent on the smaller containers listed above. One of the data structures is a Hash Map that stores all of the goods as its keys and its values are Linked Lists of cities who have at least one Customer object who have that good on their list. Each good has a different order set of cities whose customers have at least one of those goods in their list. However, since there are 2000 customers, the goods have many similar cities since there are only so many cities for the customers to be placed into. This data structure is used in the shortest tour calculation because it acts as a list of cities that the vendor has to go to to deliver all of his goods to all of the customers who have that good on their list. The second data structure is a Hash Map whose keys are all of the cities in the CityContainer class and whose values are Linked Lists of Customer objects who are in the city. This data structure is used in calculating the total revenue that the vendor is going to make by delivering to those cities.
- The FindShortestTour class is the class that contains my algorithm and it calculates the revenue, cost and profit that a vendor has made during the tour. The algorithm works as follows: for each good, it retrieves the list of cities who have at least one customer with that good (from the first data structure in the ItemCityCustomers class) and it retrieves the first warehouse location on the list for that good in the LocationsContainer class. It calculates the shortest path using Dijkstra's algorithm from the UnDirectedGraph class from the warehouse to the first city on the list of cities who have at least one customer with that good. Then it calculates the shortest path from the first city on that list to the second city on that list. It adds up the Array List of city references that is being returned on each shortest path calculation. It does this until there are no more cities in the list of cities for that good, in which case it calculates the shortest path from the last city back to the warehouse. It returns a full Array List of the city references of all the shortest paths calculated city to city and back to the warehouse. There is no order to the way the cities are chosen to calculate shortest path, it just depends on which is the next city in the list. The revenue is calculated by multiplying the amount of customers who have that good on their list (by going through every city on the list of cities who have a customer who has

that good and for each city going through each customer and seeing if that customer contains that good, using both data structures in the ItemCityCustomers class) by the profit per unit of that good(from the data structure in the ProfitPerUnitContainer class). The cost is calculated by multiplying the cost per unit distance (which is passed through the program) by the edge weight of traversing each city.

The top level class is the InputMethods class which takes in the args parameters and runs the InputMethods class. The InputMethods class is what controls the entire program. After the args parameter is passed in, it does two things. First, the class reads all the files and stores them in the appropriate contains. All of the files get read and the containers get filled in the ReadFile class which uses the Scanner class to read from the files. Then, the class fills the larger data structures by running the CustomerContainer class which gets filled using the CustomerGenerator class. When the larger data structures are filled, the FindShortestTour class runs. All of the data structures used in the FindShortestTour class are passed through the parameter for this initialization. A complete overview of all the classes and their connections is shown in Figure 1.

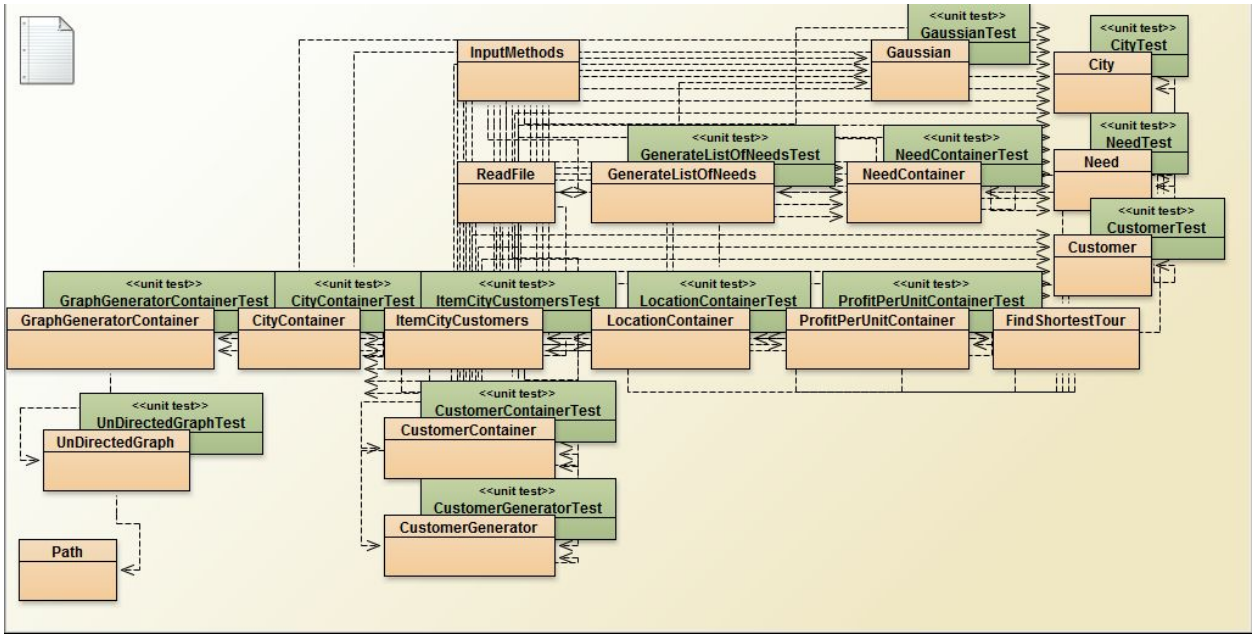

Figure 1 - Overview of all the classes in the Project and their connections.

3 Methods

There are two things that can be measured in this experiment. This first is the time is takes to find the shortest tour of all goods, it should take more time with more cities and more connections. The second is the profit made by the vendor for each type of good. We were given 11 data sets that contain 11 different number of cities and city connections.

To measure the time is takes to calculate the shortest tour of all goods, I created 4 graphs. The first 3 graphs plots the time taken vs the cost per distance unit for data sets 1, 5 and 11. This is to show how the time taken to find the shortest path varies for an increasing cost per distance unit that starts at 0.15 and ends at 5. To plot these points, I ran my program 5 times for each cost per distance unit, took the average of the 5 and plotted the average. The fourth graph plots the time taken vs each data set with the cost per distance unit kept steady at 2.236 which is the average price for gas in the state of Pennsylvania according to gasbuddy.com.[5] To plot these points, I ran my program 5 times for each data set, took the average of the 5 and plotted the average.

To measure the profit made by the vendor for each type of good, I created 6 graphs. Each graph measured the profit for one good using the information from data set 1. All 6 graphs plot the profit that vendor for that good made on that shortest tour vs the cost per distance unit that starts at 0.15 and ends at 5. To plot the points, I ran my program 5 times for each cost per distance unit, took the average of the 5 and plotted the average to the graph.

4 Data and Analysis

The following plots are to measure the time is takes to find the shortest tour.

For Data Set 1 including cost per distance unit:

Figure 2 shows the time taken to find the shortest tour for all the vendors in Data Set 1. The time it takes stays pretty consistent at about 400 milliseconds throughout the increase in cost per distance unit. Theoretically, the cost per distance unit should have nothing to do with calculating the shortest tour since it is just a multiplication at the end while finding the cost of the vendor. However, it clearly has some effects on the time taken since the times vary for finding the shortest tour. Overall, Figure 2 shows a consistency in the times it took my algorithm to find the shortest path. With Data Set 1 only having 100 cities, it does not take that long of a time for my algorithm to find a shortest tour.

[5] "USA and Canada Current Average Gas Prices By City/State/Province - GasBuddy.com." *USA and Canada Current Average Gas Prices By City/State/Province - GasBuddy.com*. N.p., n.d. Web. 12 Dec. 2015.

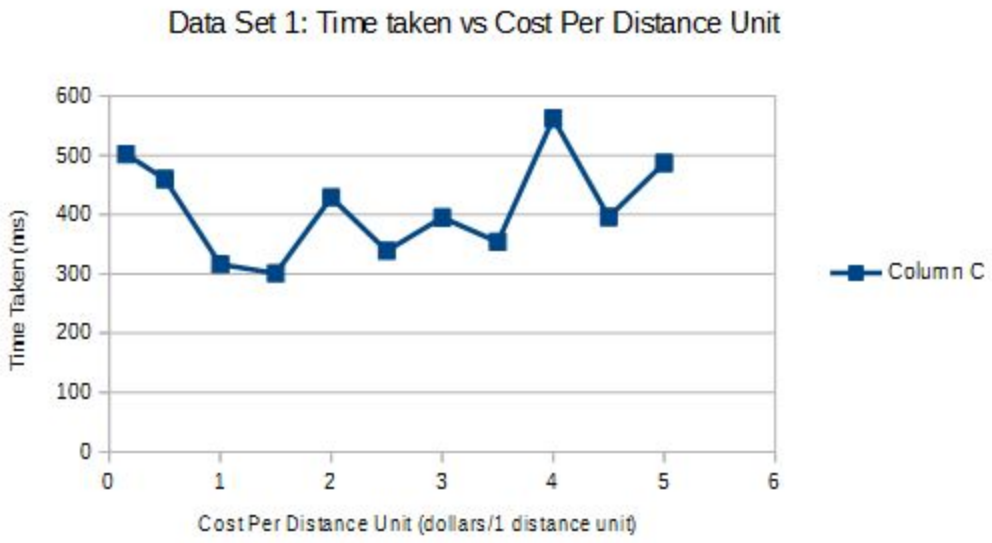

Figure 2- The Time Taken for Data Set 1 to find the Shortest Tour with Increasing Cost Per Distance Unit

For Data Set 5 including cost per distance unit:

Figure 3 shows the time taken to find the shortest tour for all the vendors in Data Set 5. The time it takes stays pretty consistent at about 1000 milliseconds throughout the increase in cost per distance unit. Data Set 5 has more than 100 cities that it is connecting and therefore it takes longer to find the shortest tour than Data Set 1 does. This Data Set is much more consistent than Data Set 1 which has more variations. Dijkstra's algorithm still works pretty well on this Data Set because it has less than 1000 nodes which, according to my graph results from Lab 9, is when Dijkstra's algorithm really slows down because the graph is so dense. However, for now the graph is still relatively sparse.

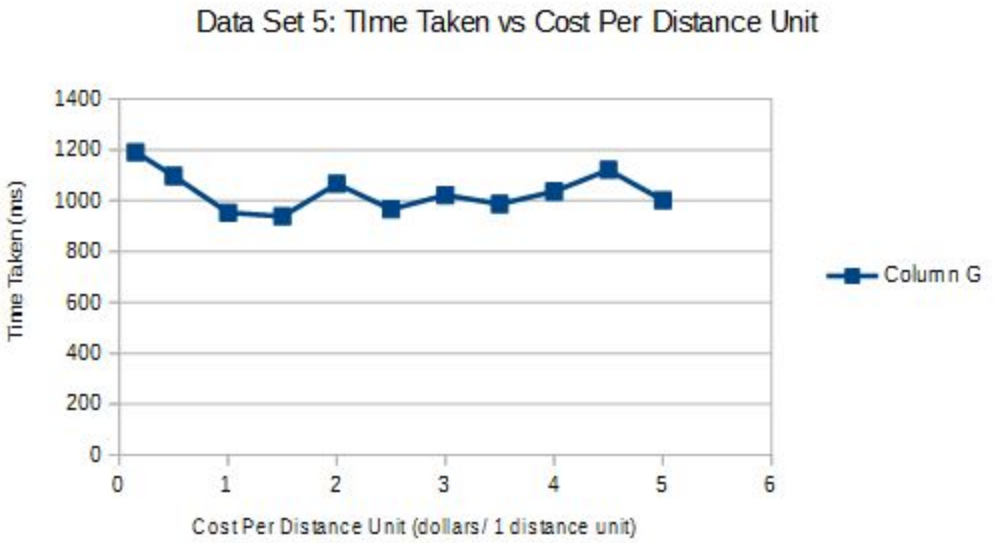

Figure 3- The Time Taken for Data Set 5 to find the Shortest Tour with Increasing Cost Per Distance Unit

For Data Set 11 including cost per distance unit:

Figure 4 shows the time taken to find the shortest tour for all the vendors in Data Set 11. The time it takes stays pretty consistent at about 1100 milliseconds throughout the increase in cost per distance unit. Data Set 11 has more than 150 cities that it is connecting and therefore it takes longer to find the shortest tour than Data Set 1 and Data Set 5 do. This Data Set is has more variations than any of the previous Data Sets. With a higher number of cities comes a higher number of edges and therefore each node has more options for their next node choice to be. This is why Dijkstra's algorithm is less efficient for dense graphs, it takes more time looking at all the options and considering all shortest tours.

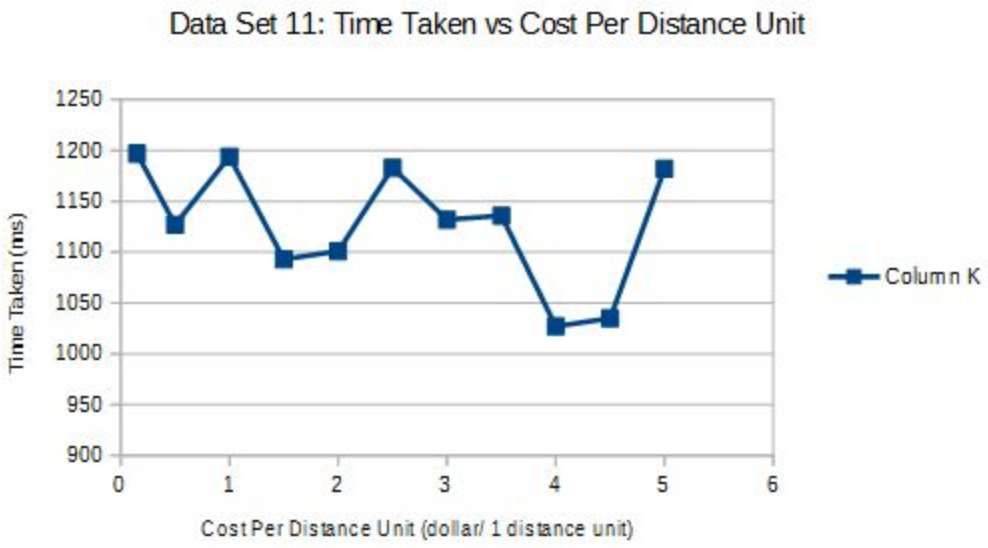

Figure 4- The Time Taken for Data Set 11 to find the Shortest Tour with Increasing Cost Per Distance Unit

For all Data Sets including cost per distance unit:

Figure 5 shows the time it takes the algorithm for each Data Set to find its shortest tour. It is is clear that the time increases as the number of cities (nodes) in the graph increases. In the beginning, the increase in time is significant. It increases about 200 milliseconds on each new data set for the first two data sets. After the first four data sets, the time it takes for the algorithm to calculate the shortest path remains pretty constant at about 1100-1200 milliseconds. I think this is because the higher data sets do not vary that much in amount of cities it has compared to the lower data sets.

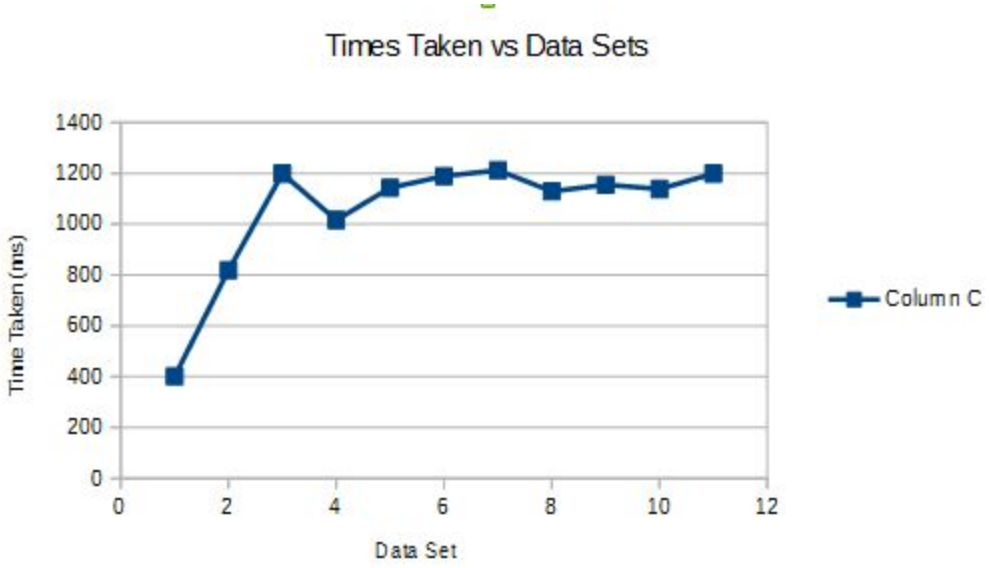

Figure 5 - The time is takes for all Data Sets to find the shortest tour for all vendors

My algorithm uses Dijkstra's algorithm to calculate the shortest tour which has a time complexity of O(E log V) for sparse graphs and O(V^2) for dense graphs. Dijkstra's is being run inside a for loop that goes through the list of cities who have at least one customer who has that item on their list. This adds O(N) to each of the complexities because I am doing the same thing on each index of the Linked List. Therefore the complexity for my shortest tour algorithm is O(E*N log V) for sparse graphs and O(N*V^2) for dense graphs. I found in my Lab 9 lab report which analyzed the performance of Dijkstra's algorithm, that it worked pretty fast with graphs that has fewer than 1000 nodes (sparse graphs) and it took much longer with graphs that had more than 1000 nodes (dense graphs). These are not very good time complexities for my shortest tour algorithm. However, the graphs we were given in the Data Sets never surpassed 200 cities and about 5000 edges. This meets the requirements for sparse graphs which still work at an okay time with my algorithm. I was not able to show plots for how my algorithm works on dense graphs because none of the Data Sets contained that many cities.

The following plots are to measure the profit each vendor made on their shortest tour.

For data set 1, the profit per unit is as follows: 10 dollars for one unit of beverages, 15 dollars for one unit of fruits, 40 dollars for one unit of meat, 25 dollars for one unit of baked goods, 20 dollars for one unit of vegetables and 10 dollars for one unit of dairy. There are about 100 cities in data set 1. My hypothesis is that the profit the vendor makes will decrease as the cost per unit distance increases which makes sense because if you think about reality, when gas prices go, the vendors spend more money on gas delivering the items than they make. Remember that the gaussian distribution of items from Project 1 still apply to this Project. Each good has a percentage of customers who want to purchase that good from the farmers

market. The goods from highest demand to lowest demand are as follows: vegetables, dairy, meat, fruits, baked goods and then beverages. [6]

For the beverage vendor:

Figure 6 shows the graph that plots the profit made by the baked goods vendor during their shortest tour delivering items vs the cost per distance unit. This graph shows that at the cost per distance unit increases from 0.15 cents to 5 dollars, the profit decreases linearly. The vendor makes 10 dollars for one unit of baked goods which is pretty low compared to the other goods. Figure 6 shows that the vendor starts losing profit when the cost per distance unit reaches about 1.00 dollars. This is a really low cost per distance unit to start losing profit for. However, this is because beverages is the least demanded good out of all the 6 goods. When you combine the low desire for beverages with its price per unit which is 10 dollars, then it is expected that the vendor will start losing profit at a relatively low cost per distance unit.

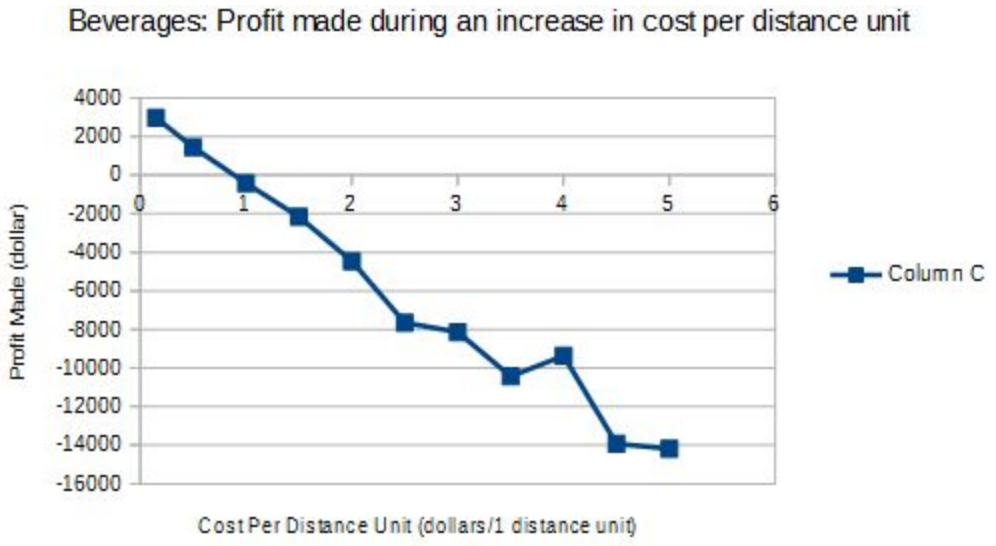

Figure 6: The graph for the profit made by the fruit vendor for an increasing cost per distance unit.

For the fruit vendor:

Figure 7 shows the graph that plots the profit made by the fruit vendor during their shortest tour delivering items vs the cost per distance unit. This graph shows that at the cost per distance unit increases from 0.15 cents to 5 dollars, the profit decreases linearly. The vendor makes 15 dollars for one unit of fruit which is a little below average compared to the other goods. The vendor starts losing profit as about 4 dollars for the cost per distance unit. Fruit is third to last on the more demanded goods for Customers to want so losing profit is expected when the cost per distance unit gets so high.

[6] Liew, Chun Wai. "CS150: Project 1." (n.d.): n. pag. *Moodle*. Lafayette College, 18 Oct. 2015. Web. 18 Oct. 2015.

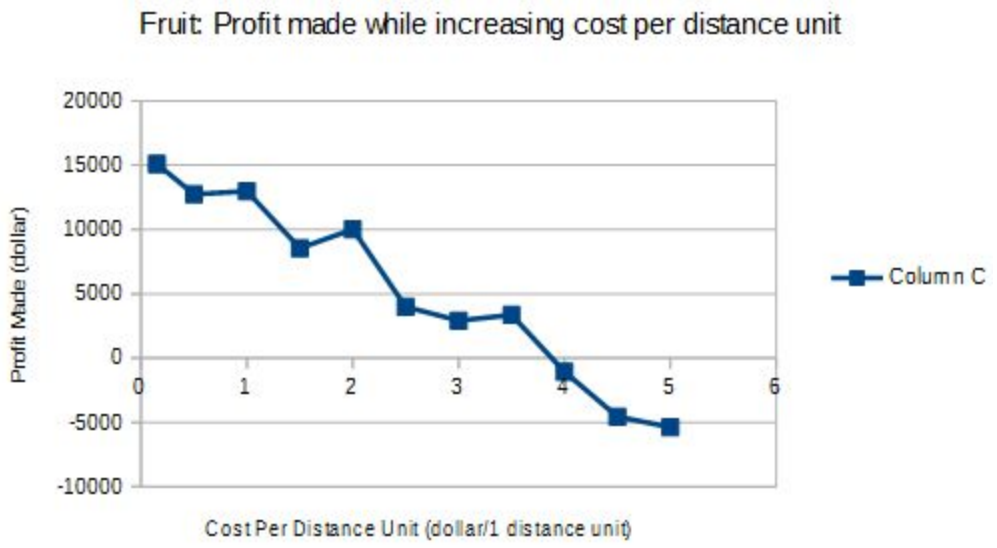

Figure 7: The graph for the profit made by the fruit vendor for an increasing cost per distance unit.

For the meat vendor:

Figure 8 shows the graph that plots the profit made by the meat vendor during their shortest tour delivering items vs the cost per distance unit. This graph shows that at the cost per distance unit increases from 0.15 cents to 5 dollars, the profit decreases linearly. The vendor makes 40 dollars for one unit of meat which is very high compared to the other goods. Figure 8 shows that the vendor never loses profit throughout the increase in the cost per distance unit. This is because not only is the cost of one unit of meat 40 dollars and the highest of all the other goods in this data set but also because meat is the more desired good at the farmers market. The meat vendor should almost never lose profit and Figure 8 shows that the vendor has a good amount of increase to go before they start losing profit.

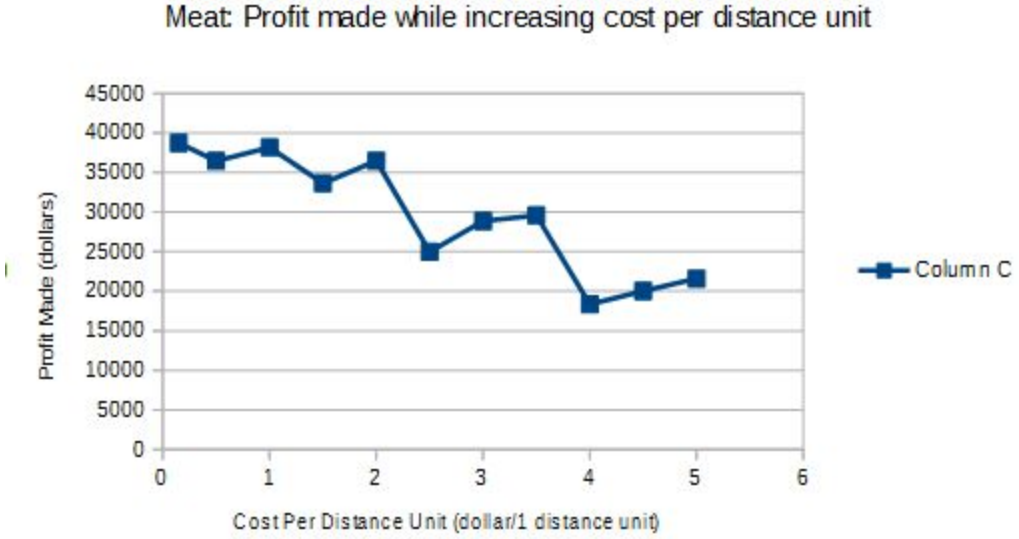

Figure 8: The graph for the profit made by the meat vendor for an increasing cost per distance unit.

For the baked good vendor:

Figure 9 shows the graph that plots the profit made by the baked goods vendor during their shortest tour delivering items vs the cost per distance unit. This graph shows that at the cost per distance unit increases from 0.15 cents to 5 dollars, the profit decreases linearly. The vendor makes 25 dollars for one unit of baked goods which is above average compared to the other goods. The vendor only loses profit at the last cost per distance unit of 5.00 dollars. Other than that, the vendor always made some profit which is pretty good considering that it is second to last on the most desired list for goods at the farmers market.

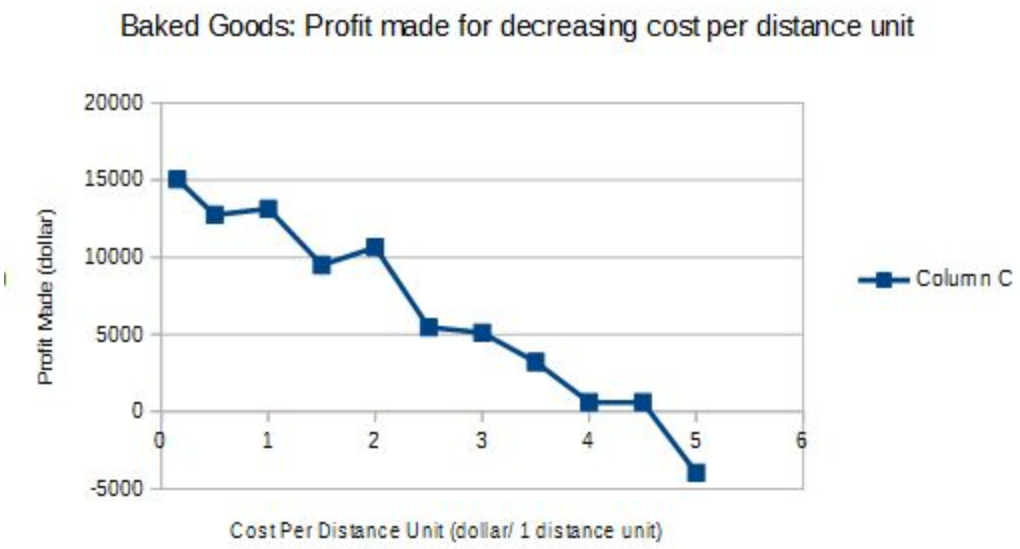

Figure 9: The graph for profit made for the baked goods vendor for an increasing cost per distance unit.

For the vegetable vendor:

Figure 10 shows the graph that plots the profit made by the vegetable vendor during their shortest tour delivering items vs the cost per distance unit. This graph shows that at the cost per distance unit increases from 0.15 cents to 5 dollars, the profit decreases linearly. The vendor makes 20 dollars for one unit of fruit which is a above average compared to the other goods. The vendor does not lose profit throughout the increase in cost per unit distance. This is because the profit of one unit of vegetable is high enough for the vendor to always be making a profit of at least 10,000 dollars and vegetable is the most desired item out of all other items at the farmers market. The vegetable vendor has some increase of cost per distance unit to go before they start losing profit. As this high of a demand for vegetables, the vendor could increase their price on one unit of vegetables and make a lot of money.

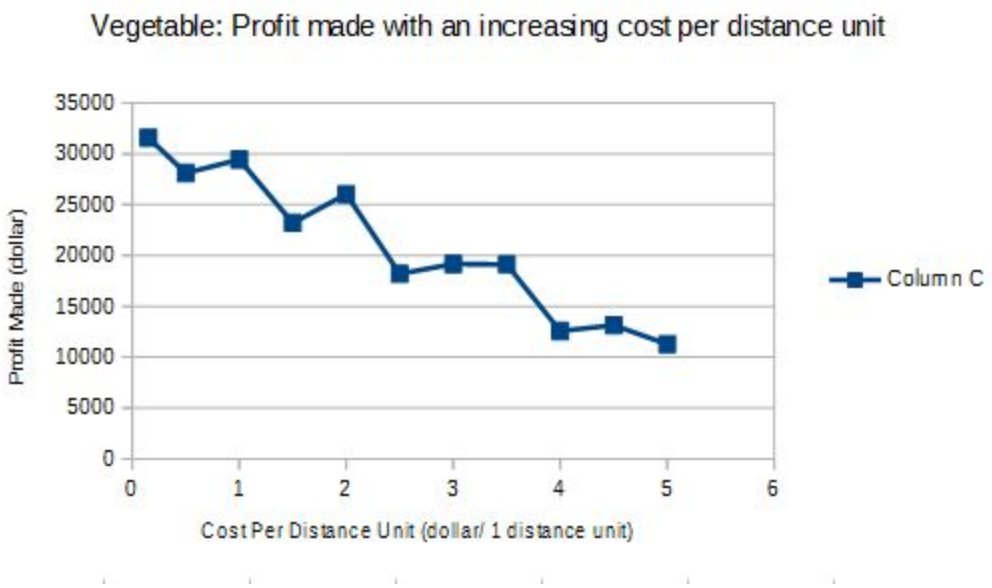

Figure 10: The graph for profit made for the vegetable vendor for an increasing cost per distance unit.

For the dairy vendor:

Figure 11 shows the graph that plots the profit made by the dairy vendor during their shortest tour delivering items vs the cost per distance unit. This graph shows that at the cost per distance unit increases from 0.15 cents to 5 dollars, the profit decreases linearly. The vendor makes 10 dollars for one unit of dairy which is pretty low compared to the other goods. Figure 11 shows that the vendor starts losing profit when the cost per distance unit reaches about 2.5 dollars which is the average price for gas in most states according to gasbuddy.com. The dairy is the second most desired item at the farmer's market but that does not make up for the price it is being sold at.

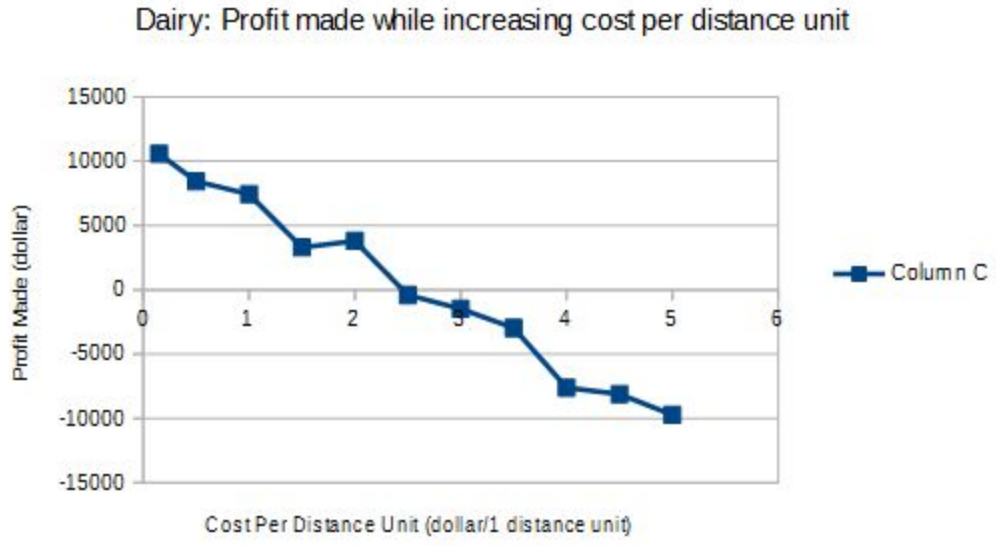

Figure 11: The graph for profit made for the dairy vendor for an increasing cost per distance unit.

All of the profits for the vendors decreased at the cost per distance unit increased which matches up with my hypothesis. The amount of profit the vendors make and the speed at which they lose profit when the cost per distance unit increases depends heavily on the price the vendor is selling the good at for one item and the desirability of that good at the farmers market.

5 Conclusion

Overall, I think that this is the best piece of code I have ever written. Not the shortest tour algorithm itself but the entire code is very well-structured for object oriented programming. However, there is a lot of room for improvement for my shortest tour algorithm. There is no other to which the algorithm chooses the next city to find the shortest path to. That could be fixed by starting at one city and for every city that there is left in the city container, calculating the shortest path to all of those and the shortest path that is the chosen is the shortest of all the shortest paths. This will return a much shorter shortest tour than my shortest tour. My shortest tour is the shortest tour for the way I am processing my cities and passing them into Dijkstra's algorithm. Ideally, all the warehouses will be used in finding the shortest tours. All the warehouses a good has will have their own shortest tours so that not one truck from a warehouse is going to all the cities. The cities can be split up by which cities are closest to them and together all the trucks from all the warehouses cover all the cities that need to be covered for that good. My time complexity for the algorithm was not as good as I expected it to be in my conclusion. Dijkstra's algorithm has time complexities of O(E log V) for sparse graphs and O(V^2) for dense graphs. My algorithm adds another variable to those complexities which makes the time complexities to be O(E*N log V) for sparse graphs and O(N*V^2) for dense graphs.

6 References

"ArrayList (Java Platform SE 8)." *ArrayList (Java Platform SE 7)*. N.p., n.d. Web. 12 Dec. 2015.

"HashMap (Java Platform SE 8)." *HashMap (Java Platform SE 7)*. N.p., n.d. Web. 12 Dec. 2015.

Liew, Chun Wai. "CS150: Project 1." (n.d.): n. pag. *Moodle*. Lafayette College, 18 Oct. 2015. Web. 18 Oct. 2015.
https://moodle.lafayette.edu/pluginfile.php/189193/mod_resource/content/5/p1.pdf

Liew, Chun Wai. "CS150: Project 3." (n.d.): n. pag. *Moodle*. Lafayette College, 22 Nov. 2015. Web. 22 Nov. 2015.
https://moodle.lafayette.edu/pluginfile.php/198324/mod_resource/content/1/p3.pdf

"LinkedList (Java Platform SE 8)." *LinkedList (Java Platform SE 7)*. N.p., n.d. Web. 12 Dec. 2015.

"USA and Canada Current Average Gas Prices By City/State/Province - GasBuddy.com." *USA and Canada Current Average Gas Prices By City/State/Province - GasBuddy.com*. N.p., n.d. Web. 12 Dec. 2015.

Weiss, Mark Allen. *Data Structures & Algorithm Analysis in Java*. Reading, MA: Addison-Wesley, 1999.

Weisstein, Eric W. "Traveling Salesman Problem." From *MathWorld*--A Wolfram Web Resource. http://mathworld.wolfram.com/TravelingSalesmanProblem.html

"Vehicle Routing Problem." *Vehicle Routing Problem*. NEO, 18 Jan. 2013. Web. 18 Dec. 2015.

7 People Talked To

Amrit
Andrew
Nax
Josh
Kevin
Jack
Thomas
Yahan