Richard Phan

CS 150 Data Structures and Algorithms

Professor Frank

March 15, 2019

Project 1 Report

**Introduction**

The scenario that I was presented with was to consult a coffee shop owner about how many cashiers to hire in order to maximize daily profit. To help with my consultation, I was given the average profit made per customer, the cost of hiring a cashier for a day, the average amount of time it takes for a cashier to serve a customer, as well as a list of sample arrival times of customers. I was also given the structure of the shop where all customers who arrive between six in the morning and ten at night stand in a single line and the customer at the front of the line is served once there is an open cashier. However, if the number of people in line is greater than or equal to eight times the number of cashiers, any additional incoming customer is turned away as an overflow. Specifically, my task is to minimize the overflow rate while also minimizing the number of cashiers so that the coffee shop owner can make the most money possible. I conducted experiments by using the given information to create a computer simulation of the coffee shop and varying the number of cashiers in each experiment.

**Approach**

In my simulation, there are the classes CoffeeShopSim, Event, and Customer. My simulation is run by an event driven simulation, which means that the simulation progresses through the occurrence of events. Each event occurs in the order of time, which makes the PriorityQueue data structure useful because it will process events in order of whichever event occurs most recently (Weiss 275). Specifically, in this simulation, there is the event where a customer arrives, and an event when the customer leaves the shop. These two events are defined in the event class and are represented as 1 for arrival and 2 for departure. The event class also contains information about the time the event occurs and the customer the event is referring to. In addition, the event class implements the Comparable interface because it makes it possible for the time in each event to be compared and later ordered in the PriorityQueue (Oracle “Comparable Interface”). The Customer class contains information about the customer’s arrival time and represents a customer who comes into the shop. There are also many instances of encapsulation in the Customer and Event classes so that the data fields of these two classes can be used in the simulation class.

The CoffeeShopSim class has instances of the other two classes and other essential elements and data structures to run the simulation. This class uses the Scanner class to read information from an input file and uses that information along with the classes’ methods to run the simulation. As previously mentioned, PriorityQueue is a vital data structure in this type of simulation and it is declared and initialized in the CoffeeShopSim class. There is also the Queue interface of type Customer which is initialized by ArrayDeque. The Queue is a necessary and efficient data structure for this simulation because it can simulate customers waiting in line (GeeksforGeeks “Queue Interface in Java”). In a Queue data structure, elements enter through the tail of the Queue and leave through the head of a Queue, which is much like how a line works in a grocery store. A double ended queue is used to initialize the Queue because it is more efficient than using a LinkedList (Oracle “Class ArrayDeque”). An additional data structure that I used is the ArrayList and I used multiple ArrayLists to store lists of arrival times, the wait times of each customers, and a list of all customers who will enter the shop. ArrayLists are useful because adding an element to the end of the list and accessing an element at a given index is quick. There are many instances throughout the simulation where the program needs to access an element in an ArrayList and add elements to the end of a list.

**Methods**

To accurately determine the number of cashiers to optimize profit, I had to run different experiments of my simulation. The variable that changes between my experiments is the number of cashiers because this variable is what determines how long customers have to wait in line and how long that line will be. The number of cashiers directly impacts the overflow rate of the shop and therefore also impacts the profit. I have set up the simulation so that the number of cashiers is inputted as a command line argument because it will be easier to change the number of cashiers between experiments. The different values that are calculated in each experiment include the total profit, total cost to hire cashiers, net profit, average wait time of a customer in line, maximum wait time of a customer in line, and the overflow rate. I ultimately decided to conduct seven different experiments with the first experiment having only one cashier and the last experiment having seven. I chose seven experiments because if all customers were to be served, the maximum amount of profit the shop can gain is $2108. Since the daily wage of a cashier is $300, having eight cashiers would mean that the shop loses more money than it gains. After determining how many experiments I will run, I began the experiment and recorded all of my data after each experiment.

**Data and Analysis**

Once all of the experiments have concluded, I inserted all of my data into a table and plotted the total profit, total cost, and net profit with respect to the number of cashiers in the shop. As I was analyzing Figure 1, I noticed that the table depicts that when there are four or more cashiers in the shop, the total profit is maximized at $2108. There is also an overflow rate of 0% for these cases, which makes sense because if the shop made $2108, that means all of the customers were served. Since all of the customers were served in the cases with four to seven cashiers, adding more than four cashiers would progressively lessen the net profit. This means that the ideal number of cashiers that the coffee shop owner should hire is four or less. One way to determine the highest net profit is to use Figure 2. Figure 2 has the net profit of each experiment plotted, so to find the maximum net profit, I just have to find the highest point on the net profit curve. The graph showed that the coffee shop owner would make the most money by hiring three cashiers, which would yield a net profit of $1134. When analyzing the data for hiring three cashiers in Figure 1, I noticed that hiring three cashiers has a low overflow rate of 3.51%. Having less than three cashiers significantly increases the overflow rate and as a result, the shop would lose out on a lot of profit. From all of my data and observations obtained from my experiments, I can conclude that to make as much profit possible, the coffee shop owner should hire three cashiers.



*Figure 1*

Figure 1 is a table containing all of the data collected with respect to how many cashiers there are in the shop



*Figure 2*

Figure 2 shows the total profit, net profit, and total cost of paying a cashier per day according to how many cashiers are in the shop.

**Conclusion**

Upon completing the project, I learned many things including the implementations and concepts of different data structures, as well as the concept of an event driven simulation. I specifically learned about why these data structures are used and in what scenarios they are most efficient. For example, I discovered that the reason why the Event class needed to implement the Comparable interface was so that the times in the Event class could be compared with each other. This would allow the PriorityQueue to order the events with priority over time. I had also used the Queue data structure for the first time, and I found it interesting how a Queue can perfectly simulate a single file line.

The part of the project that I struggled from the most was calculating the correct departure times for each departure event. This is a crucial part for the simulation to work and I had to think of the simulation step by step to finally understand the correct time I should assign each departure event. Despite the struggle I experienced while doing this project, this coffee shop simulation project helped me thoroughly learn about essential data structures.

**References**

GeeksforGeeks, GeeksforGeeks. “Queue Interface In Java.” *GeeksforGeeks*, 18 Dec. 2018, <http://www.geeksforgeeks.org/queue-interface-java/>

Oracle, Oracle. “Class ArrayDeque.” *ArrayDeque (Java Platform SE 7 )*, 6 Oct. 2018, <http://docs.oracle.com/javase/7/docs/api/java/util/ArrayDeque.html.>

Oracle, Oracle. “Comparable Interface.” *Comparable (Java Platform SE 8 )*, 9 Oct. 2018, <http://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html.>

Weiss, Mark A. *Data Structures and Problem Solving in Java Fourth Edition.* Boston: Pearson Education, Inc. 2010. Print.