Ali Sultan Sikandar
4/19/2020
Project 2 Report

Introduction

The project analyses the operations within a coffee shop. Customers arrive at the shop and depending on the available cashiers, they either get assigned to a cashier or just wait in the line. Note that the waiting line cannot be more than 8 times the number of cashiers due to size constraints. Any customer that arrives and sees that the waiting line is more than its limit is automatically turned away and hence is regarded as **overflown**. Our goal is to analyze various factors related to the shop's performance like the **average wait time** of the customers that waited, the **day's profi**t, and the **maximum time** a customer waited. After these analysis, we find out the number of cashiers that must be employed in order to gain **maximum profit**.

Approach

Various data structures are used to achieve our goal. The main data structure which helps fulfill the operations is the **PriorityQueue[4]**. This **PriorityQueue** stores in itself the **events** that take place according to their time. The events are either the arrival of a customer to the coffee shop or its departure after it has been served. To represent these events we make an **Event** class which basically helps us identify the event based on their type and the time they take place.

The information (**the arrival time**) of all the **Customers** that arrive at the coffee shop on a single day is stored in the input **txt** file. These are all read and stored inside a **Java Api Linked List [3]** in the **CollectData** class. Later, when the **simulation** is being started, all of these are used to initialise and set up the **PriorityQueue [4]** with **arrival events**.

To represent the waiting line of the **customers,** we use Java's inbuilt first in first out **queue.** This helps us simulate a real life queue because the **Customer** who enters the **Queue** first is also the customer who gets served first. The **waiting times** of the customers is the difference in time between **an arrival event** and the **next departure event**. We store the waiting times of the **customers** in an array list **[5]** so that we can perform easy calculations on them to find the average and maximum waiting times.

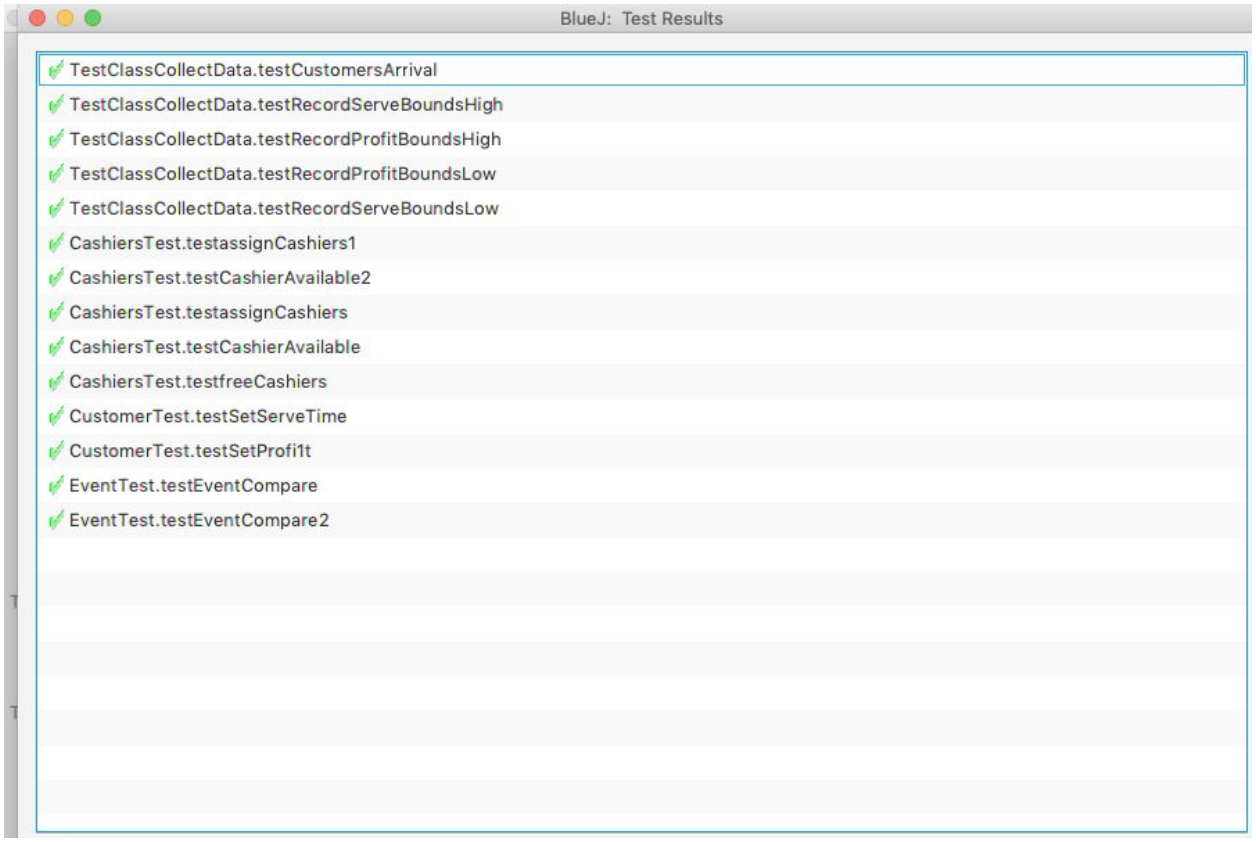
Every **Customer** has some properties : these include their arrival time, the **Profit** of serving them and **ServeTime** which is the time it takes for their service. These **Profit** and **ServeTime** values are randomised between a certain range to emulate the coffee shop operation on any random given day. These two values are assigned by the cashiers in the **Cashiers** class. We make a seperate class **RandomGenerator** which helps us obtain random **integer [1]** values between 2 d.p or 0d.p to our convenience which we can assign accordingly to the **Customers** in the **Cashiers** class.

Methods

The simulation which emulates the day's operation of the coffee shop is performed through the approach of the **event driven simulation.[2]**

Various methods are used to achieve our goals in this lab. The whole operation of the lab can however be carried out using one method that integrates all the others: **obtainInfo(int cashiers, seed)**. This method, when carried out in the main class of the project, yields a **String** which contains information about a day's operation of the coffee shop. The file **input.txt** contains the arrival times of all the **customers** that arrive at the coffee shop. While initially the total number of customers that arrive at the coffee shop is 1053, note that the customers who arrive after **9:00 pm (75600 seconds),** are never serviced nor overturned. To obtain consistent and reliable results, we run the **obtainInfo** method 5 times in the **main class** and accordingly derive the average values of different shop properties.

All the methods which do not rely on using inbuilt **Random** class are unit tested for their correctness. The results of these unit tests are as follows:



For unit testing, however, we use a much shorter input txt file named **input-1.txt.**

Data & Analysis

Different properties of the coffee shop are compared against the number of cashiers employed. Given below is the data obtained by running the **obtainInfo** method for different numbers of cashiers from 1 to 15.

```
OverFlow: 668 AverageWaitTime: 1139 MaxWaitTime: 2264 Revenue: 2800 Day's Profit: 2500 TotalServed: 360
OverFlow: 405 AverageWaitTime: 1175 MaxWaitTime: 2722 Revenue: 4861 Day's Profit: 4261 TotalServed: 623
OverFlow: 235 AverageWaitTime: 1130 MaxWaitTime: 2593 Revenue: 6196 Day's Profit: 5296 TotalServed: 793
OverFlow: 131 AverageWaitTime: 988 MaxWaitTime: 2943 Revenue: 7020 Day's Profit: 5820 TotalServed: 897
OverFlow: 70 AverageWaitTime: 758 MaxWaitTime: 2585 Revenue: 7507 Day's Profit: 6007 TotalServed: 958
OverFlow: 38 AverageWaitTime: 560 MaxWaitTime: 2556 Revenue: 7764 Day's Profit: 5964 TotalServed: 990
OverFlow: 21 AverageWaitTime: 395 MaxWaitTime: 2443 Revenue: 7901 Day's Profit: 5801 TotalServed: 1007
OverFlow: 12 AverageWaitTime: 340 MaxWaitTime: 2618 Revenue: 7973 Day's Profit: 5573 TotalServed: 1016
OverFlow: 6 AverageWaitTime: 204 MaxWaitTime: 1248 Revenue: 8022 Day's Profit: 5322 TotalServed: 1022
OverFlow: 3 AverageWaitTime: 168 MaxWaitTime: 870 Revenue: 8045 Day's Profit: 5045 TotalServed: 1025
OverFlow: 1 AverageWaitTime: 143 MaxWaitTime: 939 Revenue: 8060 Day's Profit: 4760 TotalServed: 1027
OverFlow: 0 AverageWaitTime: 82 MaxWaitTime: 708 Revenue: 8070 Day's Profit: 4470 TotalServed: 1028
OverFlow: 0 AverageWaitTime: 0 MaxWaitTime: 0 Revenue: 8075 Day's Profit: 4175 TotalServed: 1029
OverFlow: 0 AverageWaitTime: 0 MaxWaitTime: 0 Revenue: 8075 Day's Profit: 3875 TotalServed: 1029
OverFlow: 0 AverageWaitTime: 0 MaxWaitTime: 0 Revenue: 8075 Day's Profit: 3575 TotalServed: 1029
```

```
Can only enter input while your programming is running
```

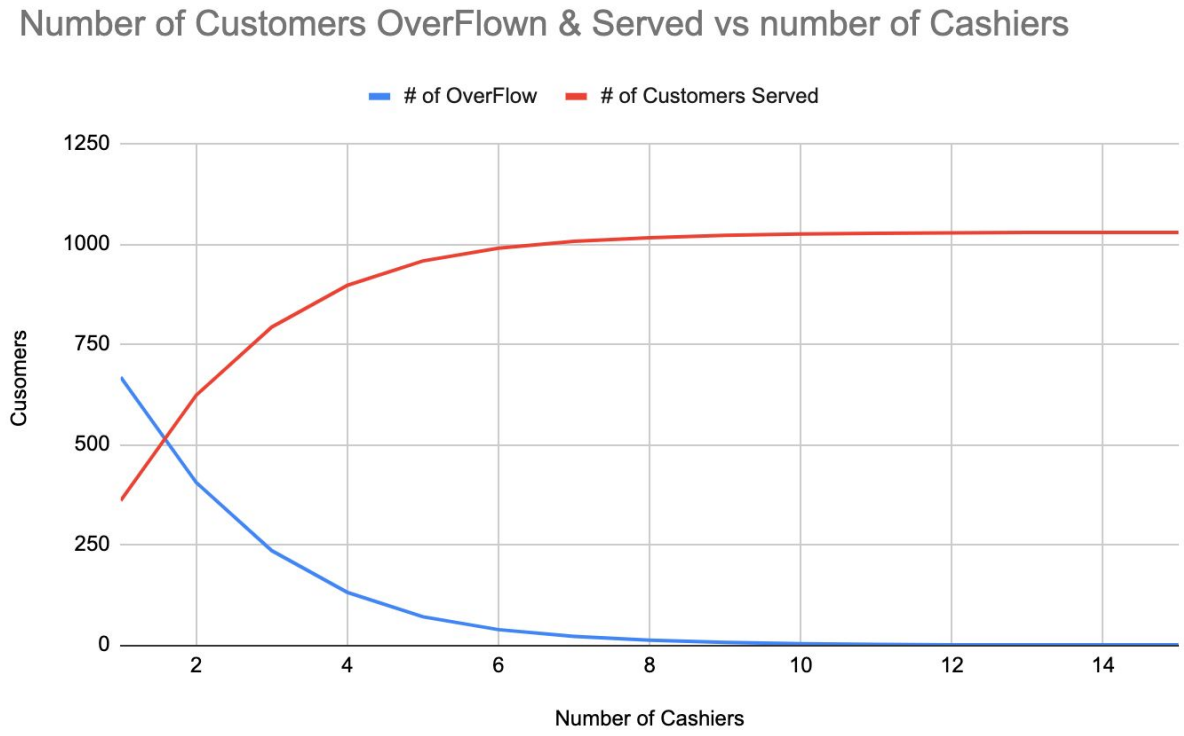


FIG.1 Customers Served and OverFlown

As expected, with employment of more customers, the number of customers that get overturned because the waiting line (**Queue length > 8* number of cashiers**)

decreases following somewhat of a quadratic decline. For the given input text file, we come to know that 0 **customers are overturned** when the customers employed are either **12 or more than 12.** On the other hand, this means that the number of customers served increases as the number of cashiers increases. That is why employing **12 cashiers** ensures that no customers are overturned and all of them are **served** except the ones who arrive after the closing of the coffee shop.

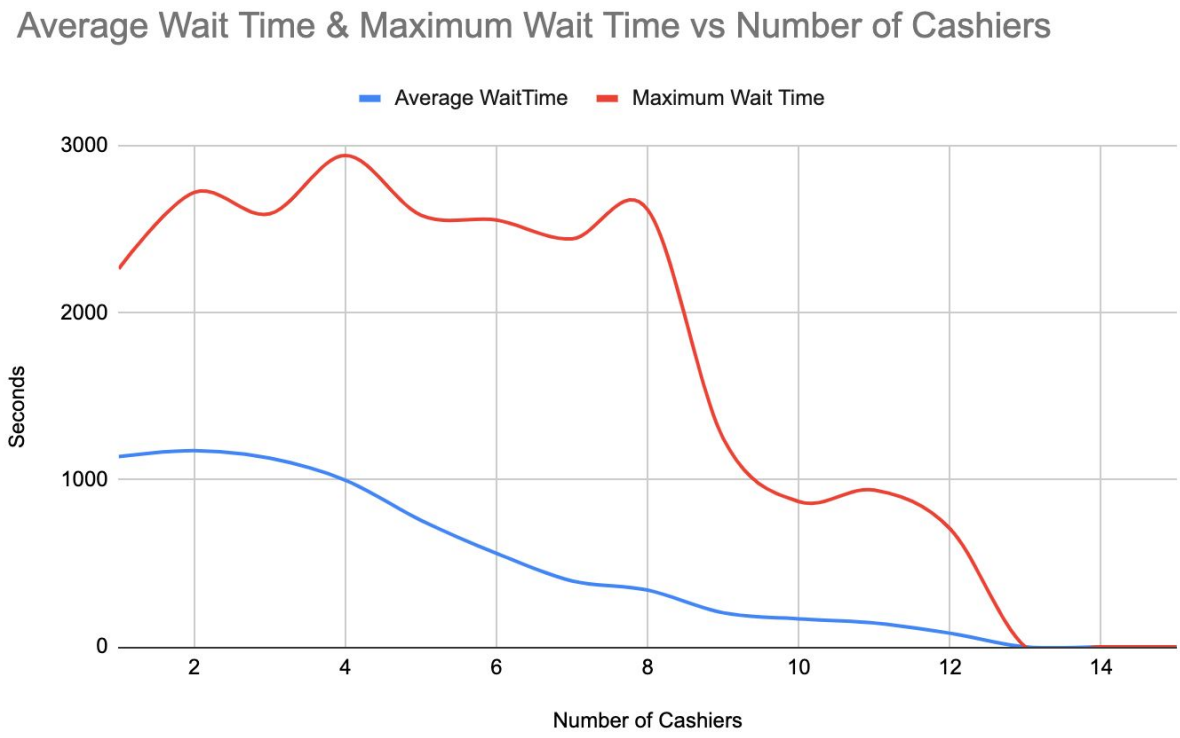


FIG.2 Wait-time analysis

While the general trend is that the average wait time a customer waits in the **queue** decreases for more number of customers, the maximum wait time of a customer gives somewhat of a varying behavior. This can be attributed to the fact that for a relatively small number of cashiers (**1-3**), the waiting line is not big enough to accomodate the customers who arrive, and therefore the overflow rate is high. Therefore even those **customers** who are not overflown wait for a significant amount of time. The **maximum waiting time** for the given input file is **2943 seconds** which occurs when the number of cashiers employed is 4 . Regardless, we see that the **maximum waiting time** of the **customers** do increase with the increase of **cashiers** after **6.** This indicated that the waiting line is always able to accomodate the arriving customers who then wait for relatively less time as more and more **cashiers** are available to service them.

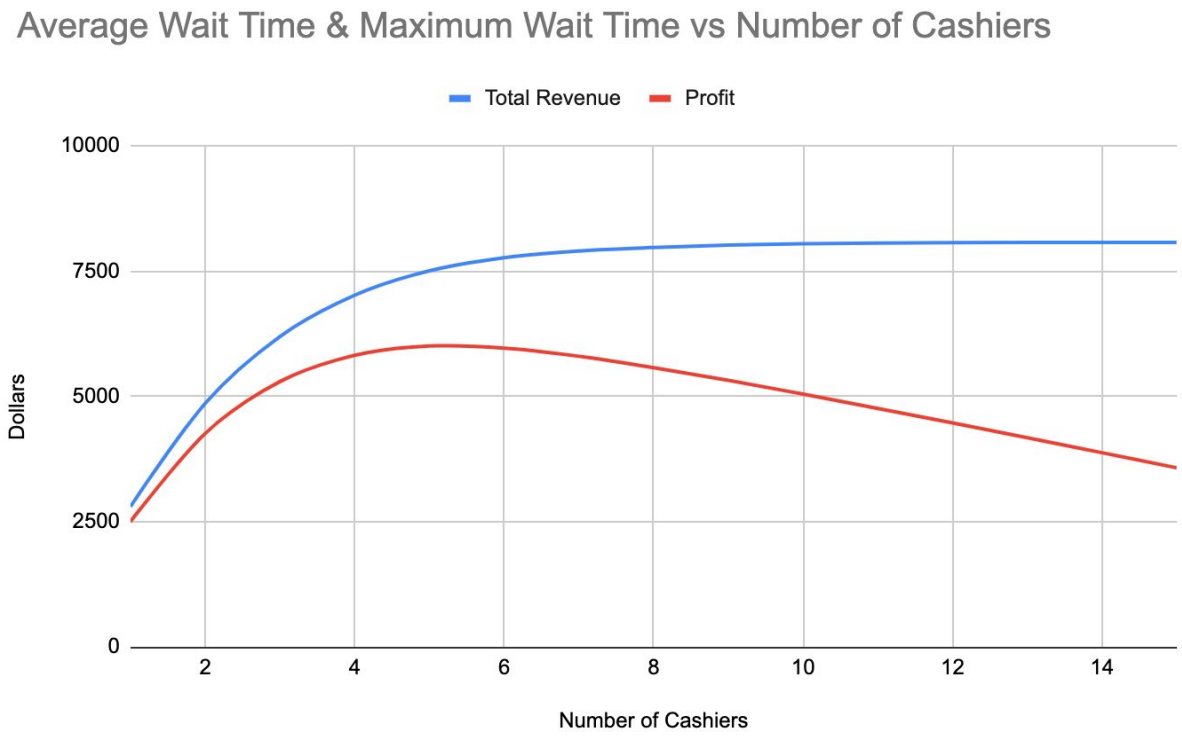


FIG 3. Total Revenue & Profit vs Number of Cashiers

With more customers being served due to more number of cashiers, the revenue generated by the coffee shop subsequently increases. The maximum revenue, as expected, is **8070 dollars,** for when all the customers who arrive within the working hours of the coffee shop and all of them are served. However, after **5 cashiers**, as we can see from the graph that the **net profit** collected within one day begins to decrease as the cost of employing the **cashiers** increases too.

Conclusion

After running the analyses for the given input file, we come to two conclusions. First, it is clear that if gaining **maximum profit** is the goal of the coffee shop, then no more or less than **5 cashiers** must be employed. This is the optimal number of cashiers that earns the maximum profit (**6007 dollars**) after subtracting the **total cashiers cost** from **the total revenue.** This is also relatively favourable to the customers because the **average waiting time** is **758 seconds (compared to a maximum of 1175 seconds average waiting time)** and only **70 of the 1053 customers are overturned** (**6.5 %**).

However, if increasing the popularity of the shop is the only goal, it should be imperative that all the customers that arrive at the shop are served and none of them is overturned. This is achieved by employing at least **12** cashiers because it is then then all the customers who arrive during the working hours are served (**1028 of the 1053 who arrive at the coffee shop**).This is very favourable for the customers as service is guaranteed upon arrival within the working hours. The profit earned, however, is relatively low : **4470 dollars.**

References

1. Random. (2018, October 6). Retrieved April 20, 2020, from https://docs.oracle.com/javase/7/docs/api/java/lang/System.html
2. Weiss, M. A. (2010). *Data structures & problem solving using Java*. Boston, MA: Pearson.
3. Class LinkedList<E>. (2018, October 6). Retrieved April 20, 2020, from https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html
4. Class PriorityQueue<E>. (2018, October 6). Retrieved April 20, 2020, from https://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html
5. Class ArrayList<E> (2018, October 6). Retrieved April 20, 2020, from https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html
6. Scanner (2018, October 6). Retrieved April 20, 2020, from https://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html