Assignment 2 - Dragos Pop

Finding the shortest queue

The first step in implementing the grocery shop was building a function that returns the shortest queue. This was done by measuring the queue size of each open server and comparing the identified values. Consequently, by keeping track of the smallest line after each comparison, the index of the shortest queue could be discovered.

Registry opening

Secondly, I developed an algorithm able to detect if a new checkout should be opened, by assessing the situation at each registry. In the beginning, the number of open counters was counted and compared with the number of stands where a customer can be served (5) since the first one must always be lower than the second in order to open a new server. If this is the case, another registry should be open when there are 3 persons at all already open counters. Keeping into account that every new client will always choose the smallest queue, it can be noticed that all the queues are going to grow steadily(see Fig. 1). In other words, the customers will be uniformly spread at each registry. Hence, a new server should be opened when the size of the shortest line is equal to the open limit given by the shop.

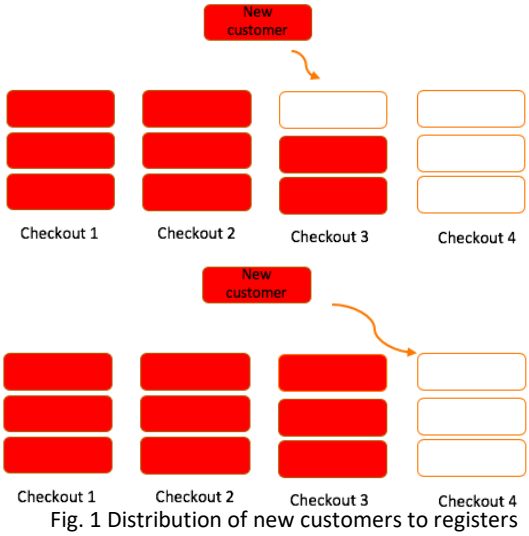

Fig. 1 Distribution of new customers to registers

Registry closing

The next part approached closing the checkouts when necessary. According to the assignment description, registries should be closed when all the queues, except one, are empty. Considering that the person who is being served is still part of the line, the previous condition is equivalent to the situation when all servers, but one, are idle. Thus, transposing this into code implied counting the number of open and idle servers. If the first figure is higher with one than the second, the registries that are both open and idle will be closed by calling the closeRegistry() function.

Handling arrivals

At the fourth step, a method that controls what happens with the customer from arrival to serving was designed. The program starts by generating a new customer, which also contains his\hers simulated arrival time. Next, assuming that a new checkout can be open instantly, whenever a buyer comes in, it is tested through Question2() if a new server should open to help the person. Later, the smallest queue is found with the help of the function Question1() and the client is directed to it. In the end, if the checkout is busy, it means that there is a queue of at least one person (the one that is served at that time, because he/she also waits) and the shopper is seated at the end of the respective line. Otherwise, the client can proceed to be served immediately.

Completing the service

Lastly, I programmed the function responsible for completing the service of a customer, namely serviceCompleted(). This was easily transmuted into code using the completed() method, which simulates the completion interval and calculates the total serving time. As the latter is very helpful in drawing the results, it was added to the tally together with the waiting time of the respective client. Afterward, the buyer is removed from the queue and the checkout continues assisting the next in line. In the case when there are no customers to be served at that registry, the program evaluates the situation to see if it is possible to shut down the idle servers.

Interpreting the results

Commuting the previous-mentioned into the given template led to the implementation of the grocery shop is SSJ. In addition, the code was written in such a way that permits the user to input values for the arrival rate (lambda) and the service rate (mu). Therefore, it can be observed that if the lambda is five times bigger than mu, the servers occupancy average approaches 99%, as the registries are busy most of the time trying to handle the arriving customers. On the other hand, if the two rates are close to each other, the proportion of time the servers are busy sharply decrease and approaches 0 as mu surpasses lambda. Regarding the average waiting time per client, it rises as the arrival rate exceeds the service rate. This makes sense given that the faster the serving speed is, the longer the waiting time drop.