CSCI203 A2 Code Report

Ethan Dakin | SN: 8209194

Code Description

A brief analysis of my code is as follows:

My code first takes the input of two variables, the filename, and the number of tellers. Then, it calls the simulation function, which initializes a queue and an array of Tellers, then reads the file line by line, and appropriately either allocates a record (Customer) to a teller, or into the queue, depending on the state of the teller(s). After the main reading is done, the queue is dequeued until it is empty, and then it prints out the simulation details. Here is the pseudocode of my program.

```
initialise
   input(filename)
   input(n_tellers)

   MAX_SIZE = 100;

   queue
   tellers
   customer

   currentTime, totalWaitTime, totalServiceTime, totalCustomers, maxQueueLength,
queueLengthSum, timeSteps = 0;

   while (read arrival, service, priority into customer)
      if arrival is 0 and service is 0 then
         break
      fi
```

```
currentTime = arrival

for (i = 0; i < n_tellers; i++) do
   if tellers[i] is busy and tellers[i] is finished then
      tellers[i].finish()

      if (queue is not empty) then
         next = queue.dequeue()
         tellers[i].serve(currentTime, next.service)
         totalWaitTime += currentTime - next.arrival
      fi
   fi
rof

customerServed = false

for (i = 0; i < n_tellers; i++) do
   if not tellers[i] is busy then
      tellers[i].serve(currentTime, service)
      customerServed = true
      break
   fi
rof

if not customerServed then
   queue.enqueue(customer)
fi
```

```
    currentQueueLength = queue.size()
    queueLengthSum += currentQueueLength
    timeSteps++
    totalCustomers++
    totalServiceTime += service
  elihw

  while (queue is not empty)
    currentTime++

    for (int i = 0; i < n_tellers; i++) do
      if tellers[i] is busy and tellers[i] is finished then
        tellers[i] finish service(currentTime)

        if (queue is not empty) then
          next = queue.dequeue()
          tellers[i].serve(currentTime, next.service)
          totalWaitTime += currentTime - next.arrival
        fi
      fi
    rof

  elihw

  print out simulation information
esilaitini
```

Complexity Analysis

The complexity analysis of my code is as follows:

Primarily, the main loop is O(n), as this is the size of the file, however, I use a few algorithms to sort my queue, and there is several loops inside the main loop. In order to properly assign the correct position in the queue, I must first loop through the entire queue to find the proper position (due to time and importance of the customer), therefore that is O(n^2), however, as the MAX_SIZE is only 100, complexity of the overall program is big O(n^2).

Data Structures

In my code, I use two main data structures. I use records for the customer information, and I use a queue for the positioning of the requests. I opted out of using a min-heap priority queue, as it overcomplicated the task that was at hand. Furthermore, I use custom classes for the Teller object, which helped to optimize my code further.

Output

```
PS D:\Projects\CSCI203\A2> g++ ./*.cpp -o main;./main.exe
Please enter the number of tellers: 1
Please enter the name of the input file: a2-sample.txt

Simulation Inputs
Number of tellers: 1
Name of input file: a2-sample.txt


Simulation Statistics
Customers Served by Each Teller
Teller [0]: 100
Total Time of Simulation: 1415.480
Average Service Time per Customer: 12.7983
Average Waiting Time per Customer: 460.7699
Maximum Length of the Queue: 64
Average Length of the Queue: 460.76993
Average Idle Rate of Each Teller
Teller [0]: 0.0019349
PS D:\Projects\CSCI203\A2>
```

```
PS D:\Projects\CSCI203\A2> g++ ./*.cpp -o main;./main.exe
Please enter the number of tellers: 2
Please enter the name of the input file: a2-sample.txt

Simulation Inputs
Number of tellers: 2
Name of input file: a2-sample.txt


Simulation Statistics
Customers Served by Each Teller
Teller [0]: 49
Teller [1]: 51
Total Time of Simulation: 747.480
Average Service Time per Customer: 12.7983
Average Waiting Time per Customer: 121.8134
Maximum Length of the Queue: 34
Average Length of the Queue: 121.81345
Average Idle Rate of Each Teller
Teller [0]: 0.0036640
Teller [1]: 0.0048676
```

```
PS D:\Projects\CSCI203\A2> g++ ./*.cpp -o main;./main.exe
Please enter the number of tellers: 4
Please enter the name of the input file: a2-sample.txt

Simulation Inputs
Number of tellers: 4
Name of input file: a2-sample.txt


Simulation Statistics
Customers Served by Each Teller
Teller [0]: 34
Teller [1]: 26
Teller [2]: 25
Teller [3]: 15
Total Time of Simulation: 509.480
Average Service Time per Customer: 12.7983
Average Waiting Time per Customer: 0.8093
Maximum Length of the Queue: 1
Average Length of the Queue: 0.80933
Average Idle Rate of Each Teller
Teller [0]: 0.0053757
Teller [1]: 0.1157488
Teller [2]: 0.2539359
Teller [3]: 0.5092445
```

Output Discussion:

As you can see through the outputs of my program, the customers are distributed evenly between the tellers, and it drastically reduced the time of the simulation as the number of tellers increased.

Furthermore, the idle rate of the tellers increased as there were more tellers, as the number of things in the queue was less than in prior simulations. Overall, the simulation worked as expected, and provided ample output, which highlights the efficacy of the algorithm.