Bank Queuing Algorithm

ASSIGNMENT 2

Sami Sheikh | Algorithms and Data Structures |17/09/2023

Solution Description

MAIN PROGRAM:

Input filename

Open file with input filename

If file not opened successfully:

Print error and exit

Input number of tellers for simulation

Create bankSimulation object with user number of tellers

WHILE data in file:

Read arrivalTime, serviceTime and priority from file.

If arrivalTime and serviceTime are 0:

Break the loop (EOD)

Create a new customer with read details

Add the customer to the bankSimulation

END MAIN PROGRAM

CLASS Queue:

Constructor:

Initialize an empty queue

Functions:

Size(): Returns the number of customers in the queue

isFull(): Checks if the queue is full

isEmpty(): Checks if the queue is empty

enQueue(customer): Adds the customer into the queue

deQueue(): Removes and returns the customer into the front of the queue

peek(): Views the customer at the front of the queue without removal

CLASS Simulation:

Constructor:

Initialize simulation with given number of tellers and other related metrics

Destructor:

Clean up resources

Functions:

runSimulation(): Run bank simulation

assignmentCustomerToTeller(customer): Assign a customer to a free teller

displayStatistics(): Display collected statistics

tellerCompletion(tellerIndex): Handle teller becoming idle

enqueueCustomer(customer): Add customer to the queue

anyTellerActive(): Check if there's any active teller

CLASS Customer:

Constructor:

Initialize customer with arrival time, service time and priority.

Class Teller:

Constructor:

Initialize teller as idle

Functions:

isIdle(): Check if teller is idle

serve(customer, currentSimulationTime): Serve a customer

becomeIdle(): Make teller idle

END CLASS DEFINITIONS

Complexity Analysis

For this program, the complexity comes from 5 major components of the algorithm:

1. Reading Customer Data - $O(n)$
2. Queue Operations - $O(n)$
3. Customer Assignment to Teller - $O(m)$
4. Simulation Running - $O(n \times m)$
5. Teller active checks - $O(m)$

READING CUSTOMER DATA

Due to the presence of n characters that are present in the file, the time complexity of this procedure is $O(n)$

QUEUE OPERATIONS

For these operations, there are three main procedures: **enQueue, deQueue** and **Peek**.

enQueue in its worst-case situation would require that every customer would need to be shifted to make space for new customers based on their arrival and priority. Making this procedure $O(n)$.

deQueue in its procedure just removes the front element of the queue, making it's time complexity of $O(1)$.

Peek operations checks the front element of the queue, giving its complexity of $O(1)$

CUSTOMER ASSIGNMENT TO TELLER

This procedures' worst case scenario is if it is necessary to check all 'm' elements to see which teller is idle, it has a time complexity of $O(m)$ where m is the number of tellers.

SIMULATION RUNNING

In this procedure, for every customer in total 'n', may have to check all tellers in 'm' which creates a time complexity of $O(n \times m)$

TELLER ACTIVE CHECKS

This procedure requires going through all the 'm' tellers, giving it a complexity of $O(m)$.

OVERALL TIME COMPLEXITY

For the entire simulation, the most dominant term is:

$$O(n \times m)$$

SPACE COMPLEXITY

In this algorithm, I am using arrays to help store tellers and idle times. This gives me a space complexity of $O(m)$ along with another array to represent the queue of customers which gives a complexity of $O(n)$.

Giving us the overall space complexity:

$$O(n+m)$$

Data structures used and reasons

ARRAYS

Arrays were used to store the variables: 'tellers', 'tellerIdleTime' and 'customerServedByTeller'.

The reason for this was because arrays provide a constant time access to their individual elements since I would often need to access details of every teller by their index. Which makes arrays a suitable choice for this data structure.

DYNAMIC ARRAY

The use of queue in the Queue class for storing Customer objects. I have used this structure because it can act as a dynamic array if there are more customers that need to be added into the program. However, in my program as per the constraints I have left it restricted to 100 customers hence it behaves like a normal array in this assignment.

CLASS

The use of classes such as "Customer", "Teller", "Queue" and "Simulation" to help encapsulate the necessary attributes, helper methods/functions and to use as a blueprint for multiple objects and organization of related data and methods into a single unit.

PRIMITIVE DATA TYPES

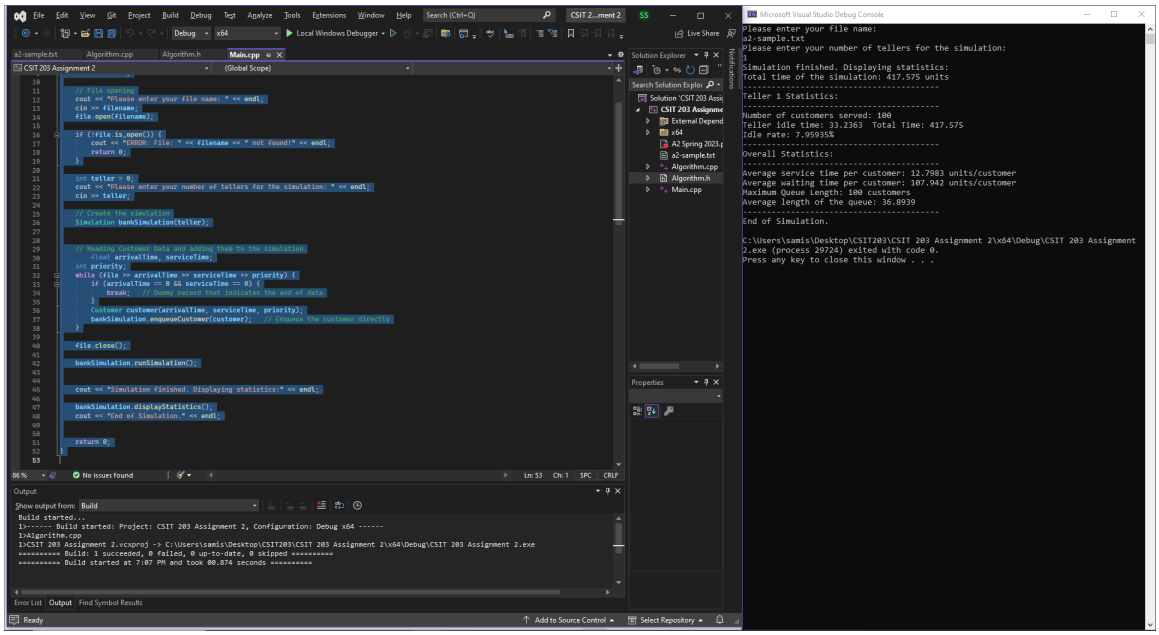
Integers, floats and bool were used to store variable data such as number of tellers, total service time along with other variables.

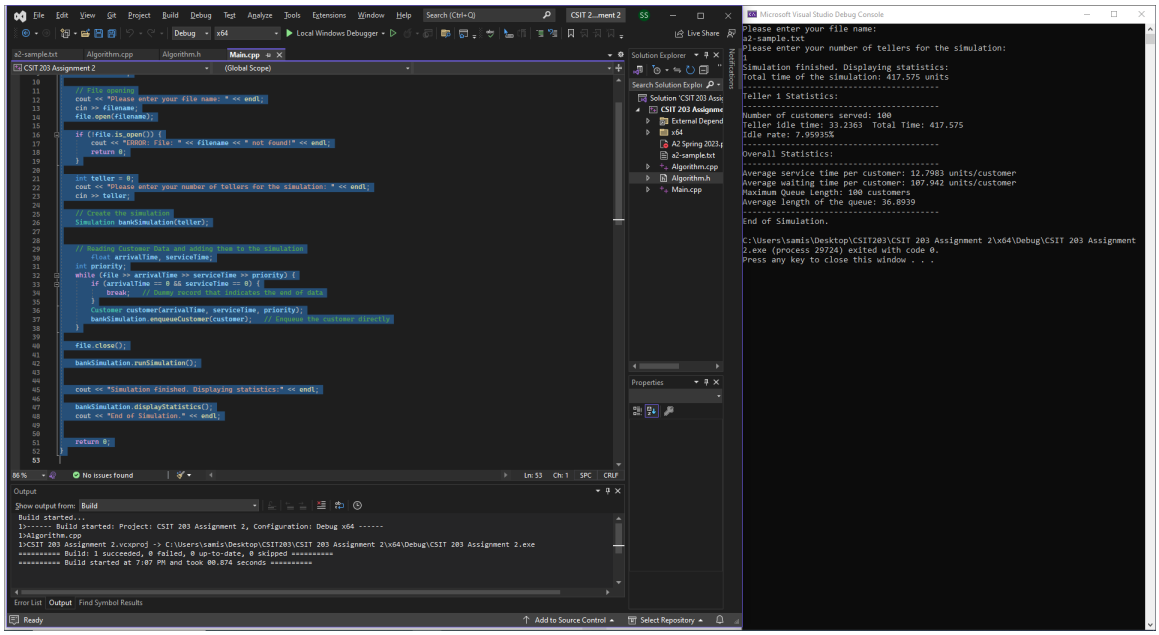
REASONS FOR CHOOSING DATASTRUCTURES

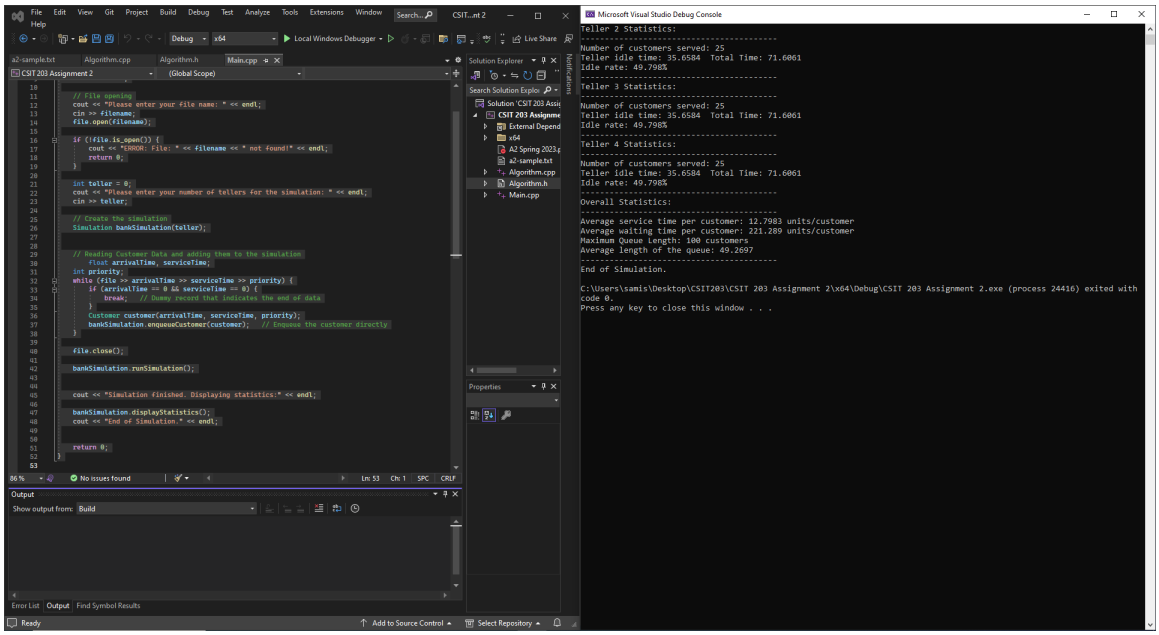
These data structures were used as per their benefits. Arrays provided me with fast access for updates and querying of individual tellers as I needed to access them frequently. Constant time access was necessary in this case.

In the case of queue data structures, the FIFO (First in, First out) structure was appropriate for the simulation as it was close to simulating a real-life queue in a bank. The structure's array-based implementation was simple to implement and as the max-size was defined, it provided a clear and structured way to model the simulation to its real life counter-part.

Compilation Snapshot and Execution of "a2-sample.txt"




Please enter your file name:
a2-sample.txt
Please enter your number of tellers for the simulation:
1
Simulation finished. Displaying statistics:
Total time of the simulation: 417.575 units
Teller 1 Statistics:
Number of customers served: 100
Teller Idle time: 33.2363 Total Time: 417.575
Idle rate: 7.95935%
Overall Statistics:
Average service time per customer: 12.7983 units/customer
Average waiting time per customer: 107.942 units/customer
Maximum Queue Length: 100 customers
Average length of the queue: 36.8939
End of Simulation.
C:\Users\samis\Desktop\CSIT203\CSIT 203 Assignment 2\x64\Debug\CSIT 203 Assignment 2.exe (process 20724) exited with code 0.
Press any key to close this window . . .


Teller 2 Statistics:
Number of customers served: 25
Teller Idle time: 35.6584 Total Time: 71.6061
Idle rate: 49.798%
Teller 3 Statistics:
Number of customers served: 25
Teller Idle time: 35.6584 Total Time: 71.6061
Idle rate: 49.798%
Teller 4 Statistics:
Number of customers served: 25
Teller Idle time: 35.6584 Total Time: 71.6061
Idle rate: 49.798%
Overall Statistics:
Average service time per customer: 12.7983 units/customer
Average waiting time per customer: 221.289 units/customer
Maximum Queue Length: 100 customers
Average length of the queue: 49.2697
End of Simulation.
C:\Users\samis\Desktop\CSIT203\CSIT 203 Assignment 2\x64\Debug\CSIT 203 Assignment 2.exe (process 24416) exited with code 0.
Press any key to close this window . . .

Impact of number of tellers on services efficiency

1. **1 Teller Performance**
    a. Total Simulation Time: 417.575 units of time
    b. Idle Rate: 7.59535%
    c. Average waiting time per customer: 107.942 units/customer
    d. Average length of the queue: 36.8939

2. **2 Teller Performance**
    a. Total Simulation Time: 155.722 units
    b. Idle Rate: Teller 1: 11.6704%
    c. Teller 2: 22.8987%
    d. Average waiting time per customer: 193.082 units/customer
    e. Average length of the queue: 43.9039

3. **4 Teller Performance**
    a. Total Simulation Time: 71.6061 units
    b. Idle rate: Teller 1 - 4.60479%
    c. Teller 2, 3, 4 - 49.798%
    d. Average waiting time per customer: 221.289 units/customer
    e. Average length of the queue: 49.2697

TOTAL SIMULATION TIME

The change in the total simulation time illustrates that with the increase in the number of tellers, the total time it takes to serve all customers decrease significantly, which is important as with more tellers, customers are processed faster.

IDLE RATE

In terms of the idle rate, a single teller is relatively low because of their constant engagement with the customer and small breaks in between. In the case of two tellers, the first teller's idle rate is still low while the second teller has a higher idle rate of 23%. Which means that while the workload is divided between both, the distribution is not completely even.

Lastly, with four tellers, the first teller has the least idle time whereas the other 3 tellers have a more significantly higher idle rate of 49%. This shows that while more tellers can

decrease the total simulation time, it is not the most optimal in terms of resource utilization as most tellers are underutilized

AVERAGE WAITING TIME PER CUSTOMER

As more tellers are involved, the average waiting time is increased, however, the results can be such due to the prioritization or serving of customers who arrived earlier than others. As there are more tellers, more customers are serviced simultaneously which leads to longer waits for other arrived customers.

AVERAGE LENGTH OF THE QUEUE

As more tellers are present, the average length of the queue increases as more customers are waiting in the queue on average throughout the simulation.

CONCLUSION

While increasing the total number of tellers has shown to decrease the total time taken to serve all customers. It has not always improved customer experience in terms of waiting time. The presence of too many tellers can cause under utilization as shown by higher idle rates for tellers when there are 4 or more.

As such, for optimal efficiency and customer satisfaction in this system, it is necessary to find the right balance in the necessary number of present tellers.

Note: For compilation of the program, please utilize the following command:

*"**g++ *.cpp -o output**"*

*"**./output**"*

**.cpp is used to compile all present cpp files that are present in one folder, please navigate towards the folder containing the program, and run the above commands.*