

Assignment Cover Letter

(Group Assignment)

|  |  |
| --- | --- |
| **Student Names:**  Christopher Samuel Tendi  Jason Jeremy Wijadi  Monique Senjaya | 2440065030  2440077301  2440061285 |
| **Course Code:** COMP 6571 | **Course Name:**  Data Structures and Algorithms |
| **Class:** L2BC | **Name of Lecturer:**  Maria Seraphina Astriani |
| **Major:** Computer Science |  |
| **Title of Assignment:**  Queuing Simulation |  |
| **Type of Assignment:** Final Project |  |
| **Due Date:**         28 - 06 - 2021 | **Submission Date:** 28 - 06 - 2021 |

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

# Plagiarism/Cheating

BINUS International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

# Declaration of Originality

By signing this assignment, I understand, accept and consent to BINUS International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

Christopher Samuel Tendi

Jason Jeremy Wijadi

Monique Senjaya

# Introduction

**Problem Description**

When the lecturer first introduced the DSA final project to the class and students have formed their respective groups, my team members including myself decided to immediately research on what we want to base our project on. After researching for quite some time, we found a problem that may allow us to implement an algorithm that we have studied in the course. We noticed that stores in Jakarta implement different kinds of queuing systems. For example, stores such as SOGO implement a multi-POS, multi queue system, on the other hand, stores like UNIQLO implement a multi-POS, single-queue system. We were genuinely curious as why these 2 multi national department stores would have different queuing systems, is one more efficient than the other or it’s just a mere gimmick? Hence, we decided to dive deeper into this issue in our project.

**Research Questions**

In order to make the task of creating the simulation much easier for us, we decided to formulate 2 research questions, which is essentially what we want to find out from the program.

These are:

* Why implement different queuing systems?
* Which is more efficient and effective?

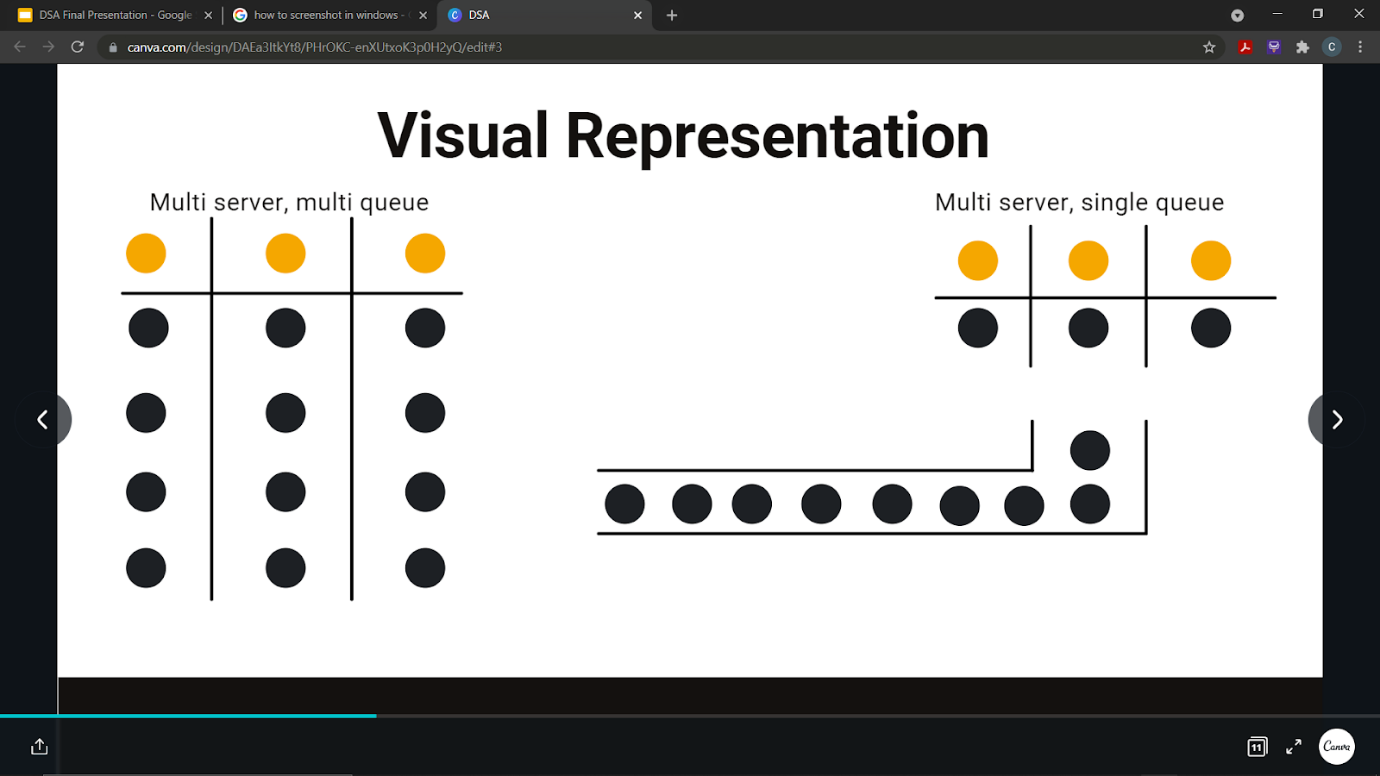


Figure 1. Visual Representation of the 2 queues (left – SOGO, right - UNIQLO)

**Program Specifications**

The program specifications are as follows:

* Software and Libraries used:
  + IDE: Visual Studio Code
* Input:
  + For both single and multi-queue:
    - Start of service time – start time of processing each new customer
    - Range of service time – the range of time needed to process a new customer
    - Arrival time – interval between each new customer
    - Simulation time (s) – how long will the simulation take place
* Output:
  + Total number of customers processed
  + CSV file for the output data

All the code, documentation and video evidences of this project can be found under this GitHub repository: <https://github.com/digaji/Queuing-Simulation>

# Solution Implementation

**Single vs Multi-queue**

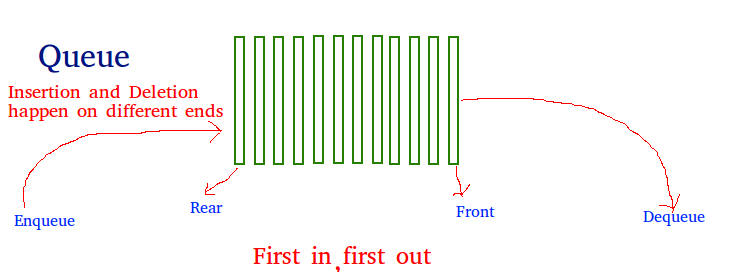
Since the problem we are trying to solve involves real-life queues, thus my team and I decided to program a simulation comparing the efficiency of a single and multi-queue. But before that, let’s look into the theory behind these 2 queues. A queue data structure refers to a set of ordered data in which each item, when added, is placed (enqueue) in the back of the queue. The items that need to get out (dequeue) of the queue does so in the front of the queue, thus we can say that queues implement a FIFO or first in first out rule.

Figure 2. representation of a queue data structure (source: https://www.geeksforgeeks.org/queue-data-structure/)

The difference between single and multi-queue is right there in its name. Single queue funnels the items into a single line, whereas multi-queue funnels the items into multiple queues. And initially, during the making of this simulation, my team and I hypothesized that single queue will be faster in the long term, which means that as more and more customers arrive, single queue will be able to process these customers much faster than multi-queue does. This is proven from a paper written by S Vijay Prasad, B Mahaboob, Ranadheer Donthi titled “A Comparative Study Between Multi Queue Multi Server And Single Queue Multi Server Queuing System”. This research article has proved the superiority of SQMS over the MQMSM and this is shown by using the principle of finite mathematical induction. Besides the corresponding.

**Program Manual**

As for the program we decided to use the queue data structure to store the number of customers. Each customer has their own variables (the input) that will be randomly generated by the simulation. Finally, the results will be displayed in a graph allowing us to compare single vs multi-queue.

When the program is run, the terminal will ask you to fill the necessary input, which were mentioned previously in the program specifications along with the expected output.

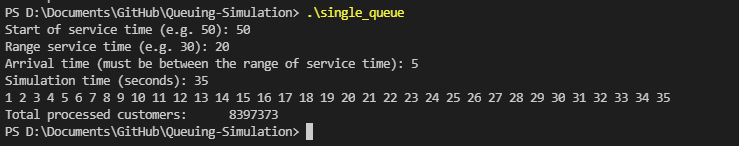


Figure 3. Terminal Input and Output for Single Queue

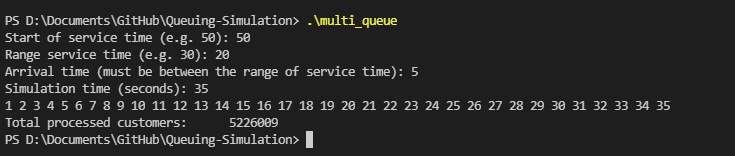


Figure 4. Terminal Input and Output for Multi Queue

**Simulation Results**

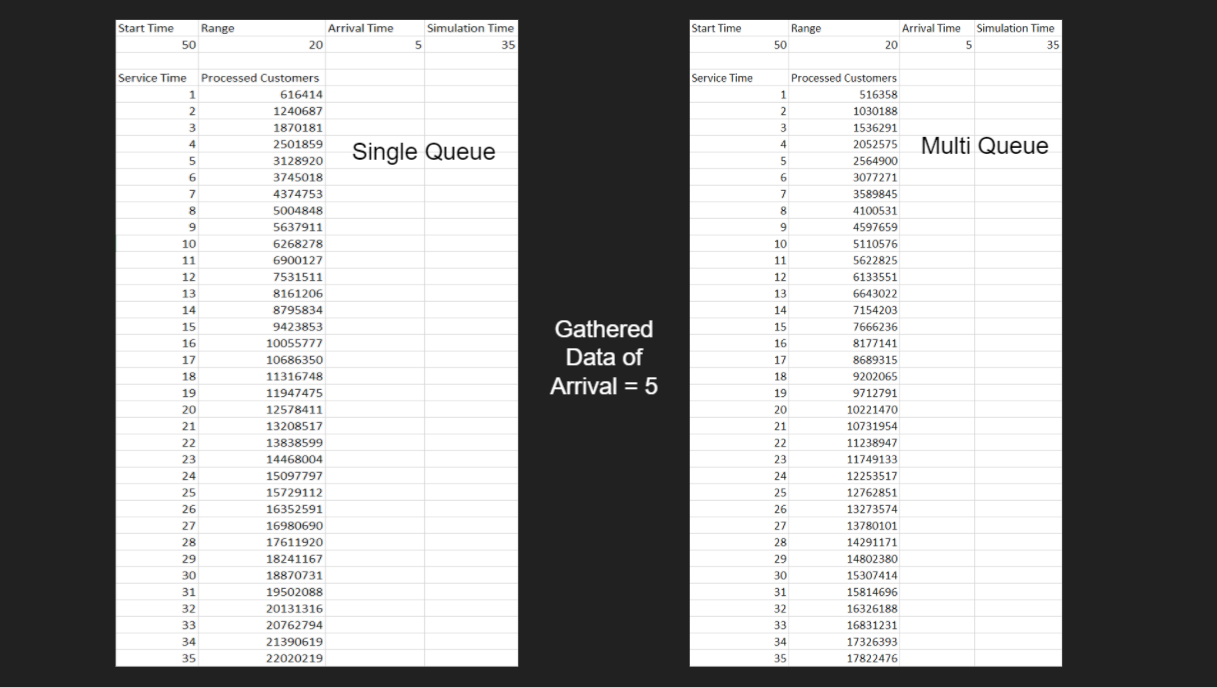


Figure 5. Results of Simulation for arrival time = 5 (interval time between each new customer)

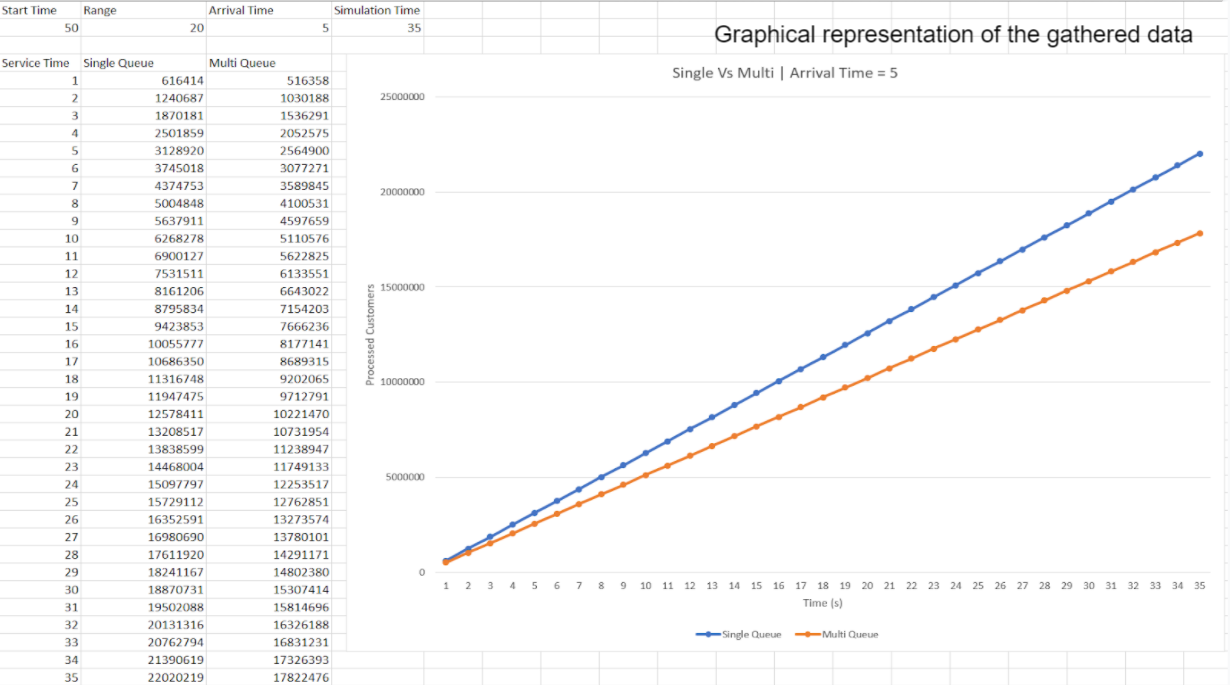
****

Figure 6. graphical representation of the results for arrival time = 5

Figures 6 and 7 above are the results and graphical representations of the simulation when the inputted arrival time = 5, with start time of 50, range of 20 and simulation time of 35. Based on these results, and what is evident from the graph, when the arrival time of the customers is 5, a single queue is seen to be much more efficient as it is able to process many more customers compared to the multi-queue. After the data has been output in the csv, we can combine the results of the two queues and make a visual representation like shown above. The result shows that the line is linear, which means that the amount of customers being processed does not increase exponentially.

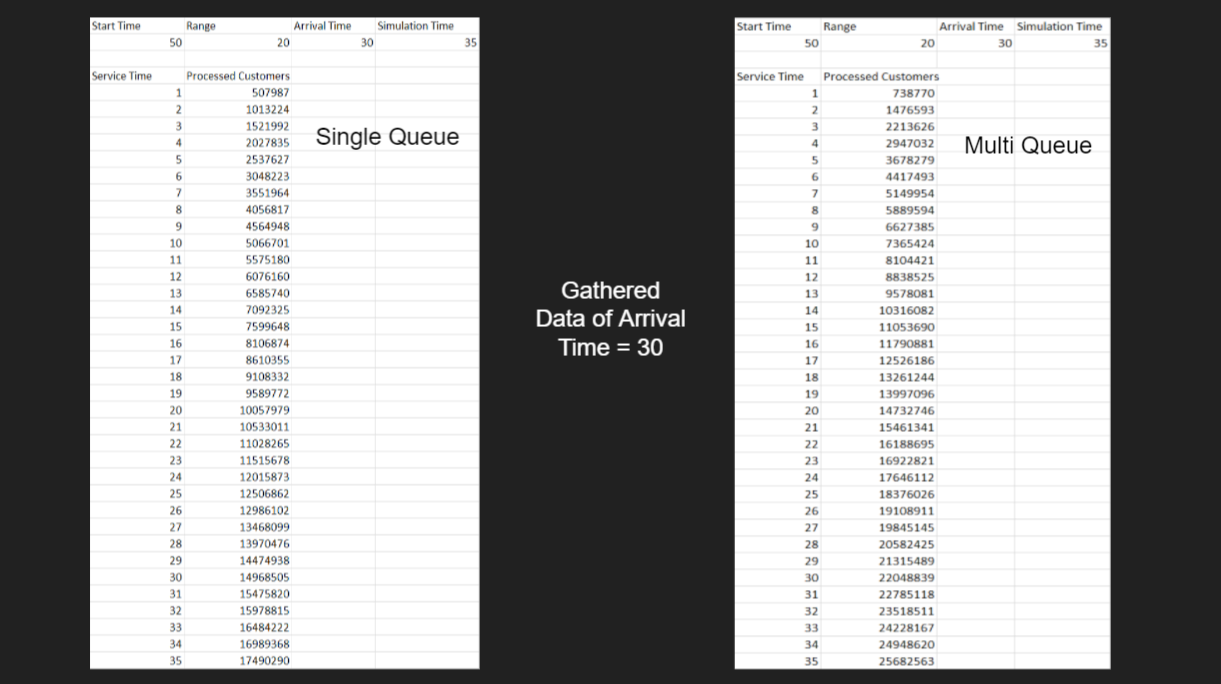


Figure 7. results of simulation for arrival time = 30s

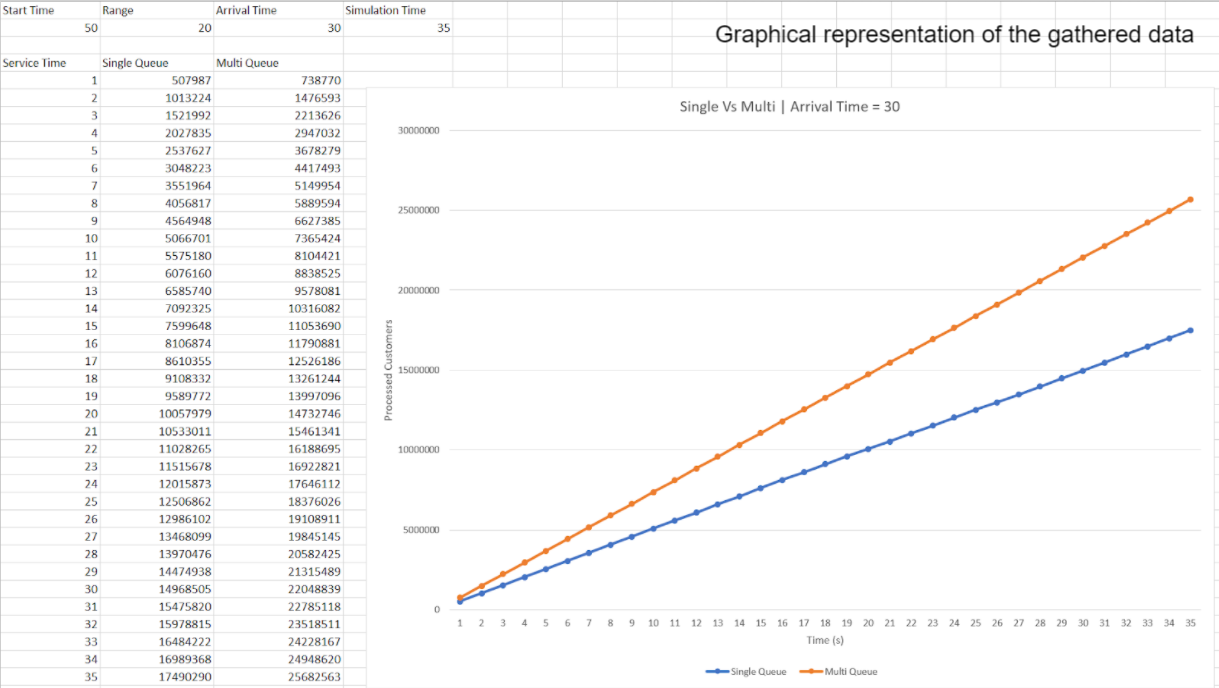


Figure 8. Graphical representation of the results for arrival time = 30s

Instead of just testing with the same input over and over again, we figured that we should change the numbers up a bit and see if the output is the same. However, when trying with a larger arrival time input, the conclusion that is derived from the graph is different from the initial hypothesis. Figures 7 and 8 above show the results and the graphical representation of the simulation when the arrival time = 30s. Based on the figures above, it is evident that when the interval between each new customer is large, multi-queue is said to be more efficient, which makes sense as with that large gap between customers, there is no way that there will be a “line” in a multi-queue system.

# Conclusion

Although we initially thought that a single queue will be much more efficient in the long run, this statement is proven to be not entirely true, evident from the results we attained when the arrival time for the customers is set to 5. However, overall single queue can be said to be faster than multi-queue, if and only if, the queues are extremely busy with customers coming in every short interval. On the contrary, when the queue is not crowded, single queue is proven to be slower than multi-queue, evident from the results we attained when the arrival time for the customers is set to 30. Moreover, we also found out that another factor that is involved as to why stores use different queuing systems also depends on how many payment sections are there in the store. For example, SOGO seems to have multiple payment sections in one single floor, which allows them to utilise a multi-queue. On the other hand, UNIQLO only has 1 payment section for an entire floor, hence why they have to utilize a multi-POS, single queue system.