Grocery Store Queue

Overview

In this project, we try to apply queueing theory through the lens of simulation. We have a grocery store with multiple cashiers, and we try to maximize the profit for the grocery store. We assume that wait time is negatively correlated with customer satisfaction. The longer the wait time, the fewer customers we will be served due to customers’ dropouts. Hence, this project aims to understand the relationships between the number of cashiers, wait time, and queue lengths. We aim to find a balance between customer satisfaction by wait time and hiring cost (numbers of cashiers) to ensure business efficiency.

Models in this project simulate customers' arrival in multiple queues served by the cashiers with an extended service from the grocery store manager included. The unit we will use in the project will be minutes because the customer arrival intervals are in minutes.

Model Assumptions

- The time between the next customers joining the queue follows an exponential distribution with a rate parameter $\lambda = 1$ customer per minute.

- Customers enter the shortest queue when they decide to join the queue.
- There is a 5% chance that a customer will need extended service from the store manager.
- Grocery store operating hours are 9 am -6 pm. After the closing time, all remaining customers are served, and no other customers are allowed to enter the queue.
- The service rate (cashier/server) follows a normal distribution with a mean of 3 min and a standard deviation of 1 min per customer.
- The service rate (manager) follows a normal distribution with a mean of 4 min and a standard deviation of 1 min per customer.
- Only one manager and they can only take one customer at the same time.
- It's a first-in-first-serve (FIFO) priority queue.
- The queue is an infinite queue.
- Customers don't leave the queue before they are served.
- Serving time follows a normal distribution and does not depend on the number of items held by the customer.

Factors to change

Target variable to optimize

- numbers of cashiers (independent)

Observed variable

- Average wait time (extended service included): The duration from customers joining the queue to customers finished being served by regular servers or potentially by the manager. We use this variable to understand a customer's overall experience in the queue.
- Average wait time (extended service excluded): The duration from customers joining the queue to customers finished being served by regular servers. We use this variable to understand the wait time before going to the manager. It allows us to immediately identify the core problem of the long wait time contributed by the regular servers or the manager.
- Maximum queue length: The maximum queue length for queues. We use it to estimate the total number of customers in the store to reach the capacity of the grocery store.
- Average queue length: The average queue length for queues.

Theoretical Analysis

This queue follows the M/G/C structure.

- M: The Markov chain. We assume that events are independent and the next stage of the model is only based on the previous state. We will assume the wait time between customer arrival follows an exponential distribution.
- G: A general distribution with mean and variance for the service time.
- C: The total number of servers.

Since there isn't a simple formula to calculate M/G/C, we use C = 1 to compare simulation and theoretical results. Since we want to apply the M/G/1 formula here, we also assume manager extended service takes no time (mean = 0, std = 0) and assigns the arrival rate as 1. Average service time as 0.1 (mean = 0.1, std = 0) to make sure two results are comparable.

For theoretical analysis to be valid, we must be aware of its academic boundaries. For M/G/1 wait time formula, we need to ensure the utilization rate ρ obeys the following formula.

$$0 \leq \rho = \frac{\lambda}{\mu} < 1$$

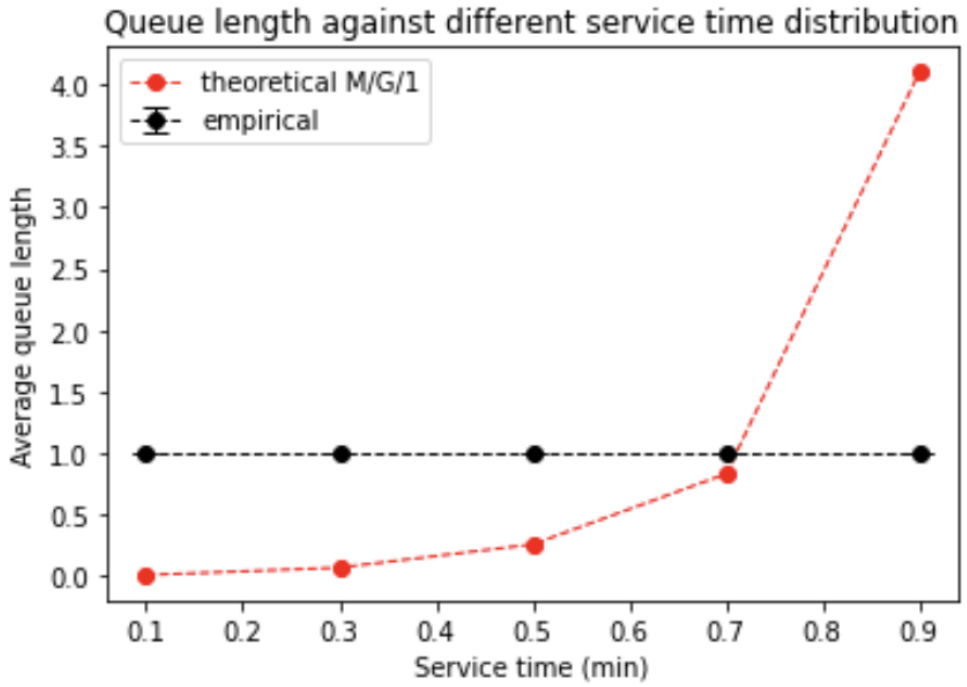
This indicates that when our customers arrive faster the servers serving speed, our calculation doesn’t hold anymore.

The average queue length for M/G/1 is

$$\frac{\rho^2}{2(1-\rho)}(1+\frac{\sigma^2}{\tau^2})$$

where τ is the mean of the service time distribution and σ^2 is the variance of the service time distribution.

If we observe the plot for average queue length, we will see that simulation results for queue remains 1, while the theoretical results peak when the utilization rate is closer to 1. It indicates a mismatch between theoretical and simulation result. However, intuitively speaking, the simulation result seems to be more reasonable. Since servers serve faster than customers arrival, all the customers get immediately serve after they arrival. If we always count queue length after a customer arrives. The average queue length will always be 1 in our scenario.



Note: Though theoretical analysis provides us estimation for our result, it might not be valid due to some theoretical constraints and assumption violations. You can view the limitation sections to learn more about it.

Simulation Results

The simulation result is based on 400 experiments lasting from 9 hours (9 am - 6 pm). Note that it is sufficient because we obtain similar results when we run 200 experiments.

If we compare the result of running 100 experiments and 400 experiments, we will notice the average wait time decrease significantly when we have 400 experiments. It is because the arrival rate follows an exponential distribution; therefore, the arrival time interval is an approximate Poisson distribution skewed to the right. (It is not an exact Poisson distribution because it is continuous.) If we execute enough sampling, the mean arrival interval will be larger, so the wait time will be shorter.

Our four variables of interest here are average wait time (extended service included), average wait time (extended service excluded), average queue length, and maximum queue length. We include the mean and the 95% confidence interval for our error plot, with our number of servers ranging from 1 to 10. The results are shown below.

Wait time

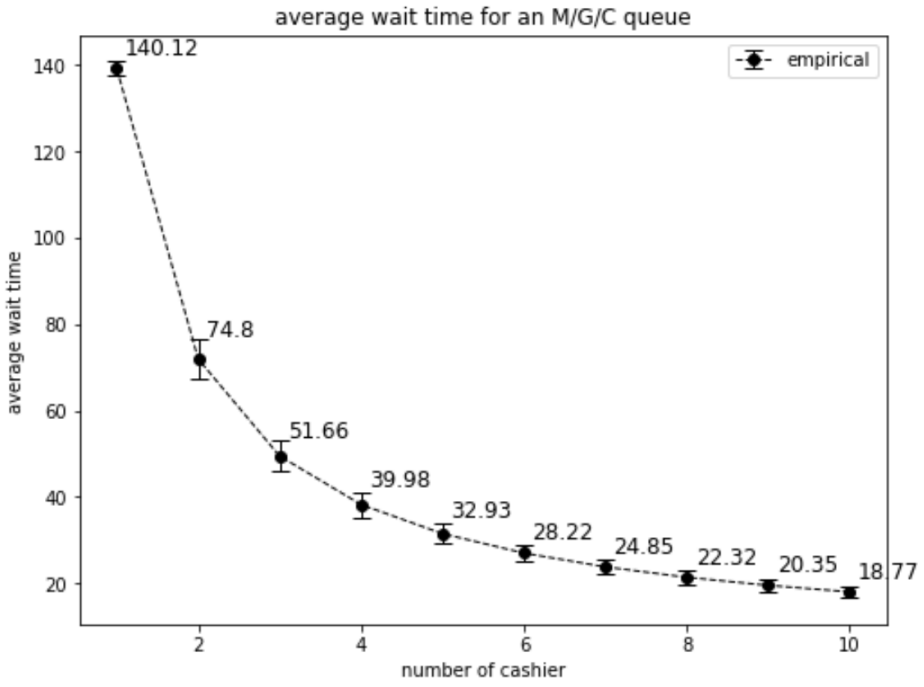


Figure 3. 95% confidence interval for average waiting time versus number of cashiers for 400 experiments

From the graph, we can see that the average wait time is pretty long, up to 18.77 min of waiting, even though there are ten servers! In this case, we will suggest the grocery store hire more cashiers to decrease the customer wait time; however, if the grocery store cannot afford more cashiers. The most economical way is to hire around four cashiers. This is because the elbow point (the point when the wait time drops the most) is about 4. To simplify, we can see that we save customers 140.12 - 39.98 = 100.14 min of wait time when we have four cashiers than having 1. Yet, we only save 39.98 - 18.77 = 21.21 min of wait time when hiring six more cashiers after having 4. This decision is based on the assumption that customer dissatisfaction grows linearly with wait time. In reality, customer dissatisfaction might grow exponentially after a specific wait time threshold.

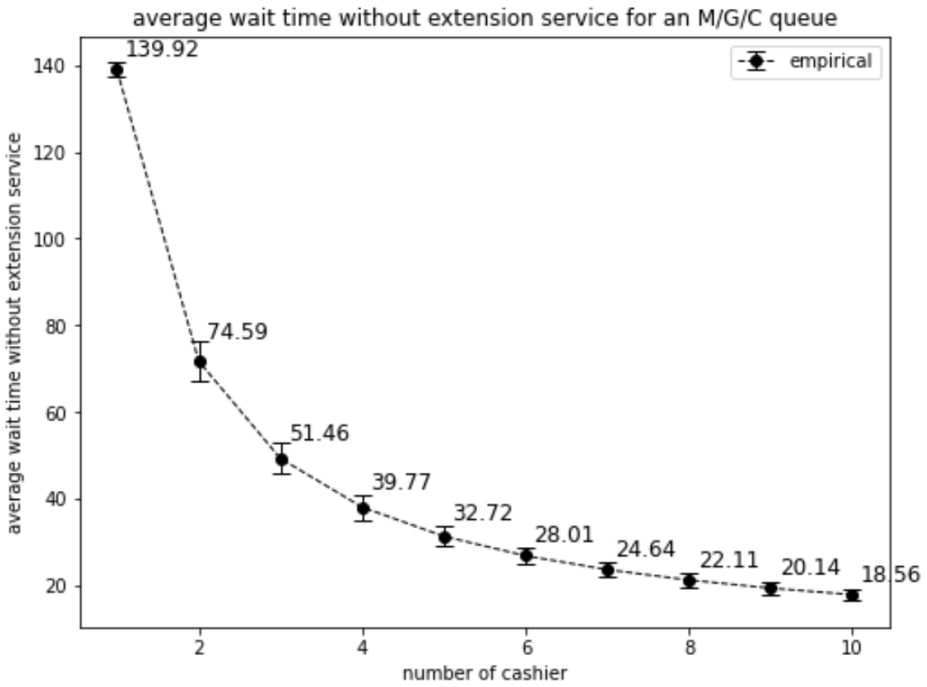


Figure 4. 95% confidence interval for average waiting time without extension service versus number of cashiers for 400 experiments

Suppose we wonder if manager service contributes significantly to the average wait time. We can compare Figure 3. and Figure 4. Figure 4. excluded the manager service time in the average wait time calculation. The graph shows both figures have similar wait times, showcasing that manager service isn't the bottleneck for waiting.

Queue length

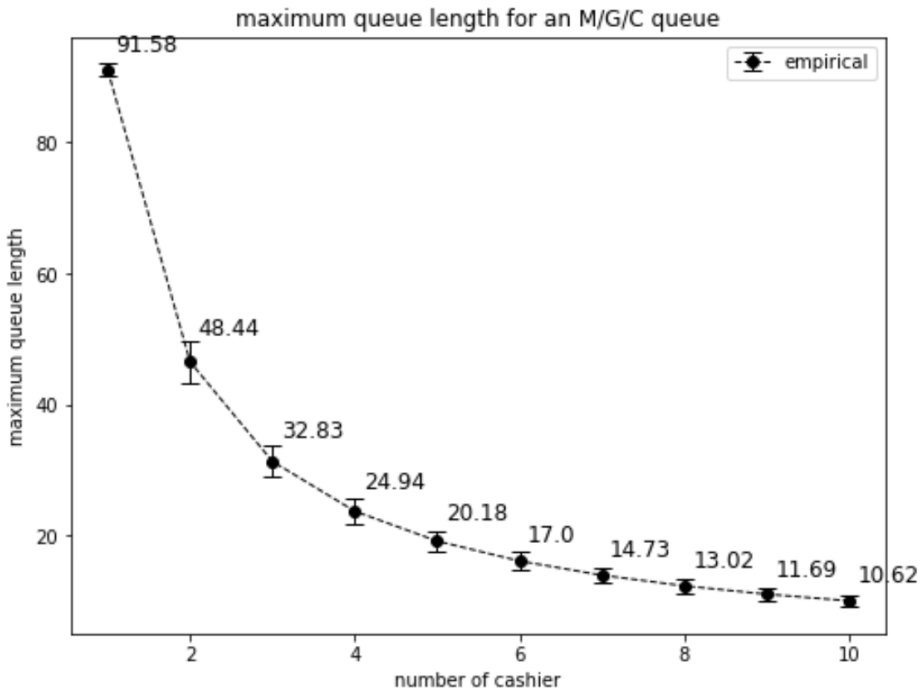


Figure 5. 95% confidence interval for maximum queue length versus number of cashiers for 400 experiments

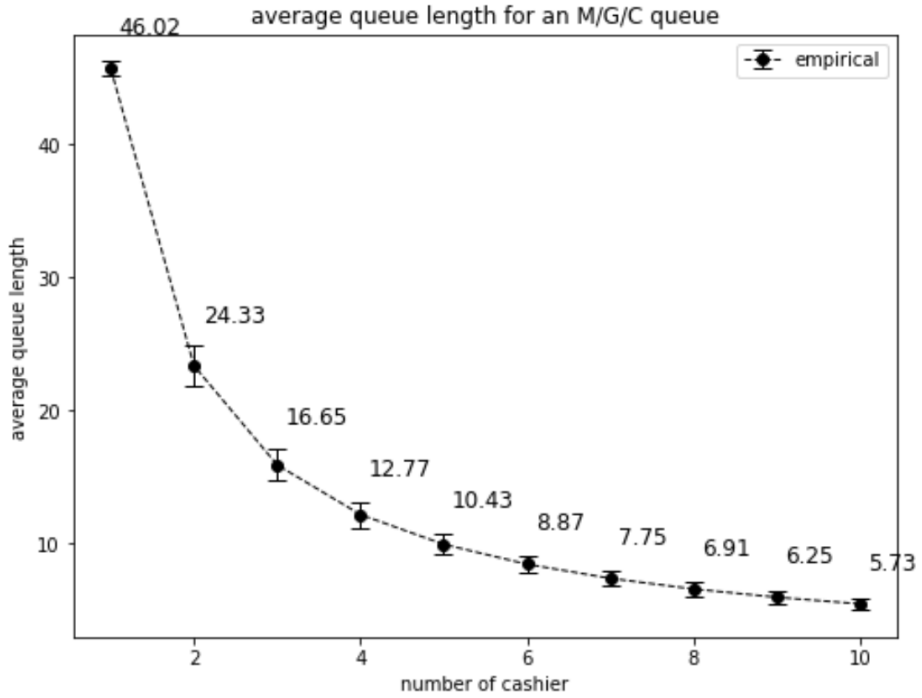


Figure 6. 95% confidence interval for average queue length versus number of cashiers for 400 experiments

Figures 5. and 6 present the experiments' maximum and average queue lengths. To pick the number of cashiers to hire, we might also consider if the grocery store is big enough to have the maximum queue length of customers. Based on our suggestion of hiring four cashiers, we need to ensure that the store can host 25 customers. If not, we will need to increase the number of cashiers.

Another solution can be hiring more cashiers during peak times. Based on Figure 6., we know that the average queue length for four cashiers is around 13 people. Therefore, if we can hire more cashiers during time with many customers, we might efficiently save hiring costs for the grocery store as well.

The simulation code is available here.

Profit Maximization

Beyond wait time, we can also set our target variable as profit. To analyze profits, we need to make assumptions on hiring costs per cashier, and total customers served.

Our initial model assumes that our total customers served will be similar (around 270 customers) regardless of the number of cashiers, as you can see if Figure 7. However, we will expect that the more cashiers we have, the more customers we can serve.

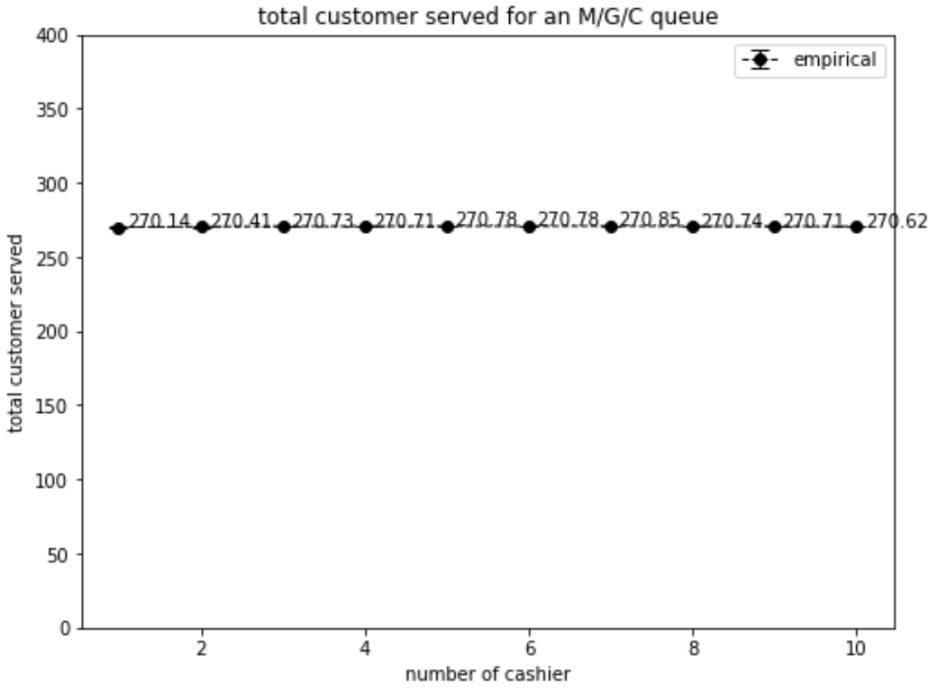
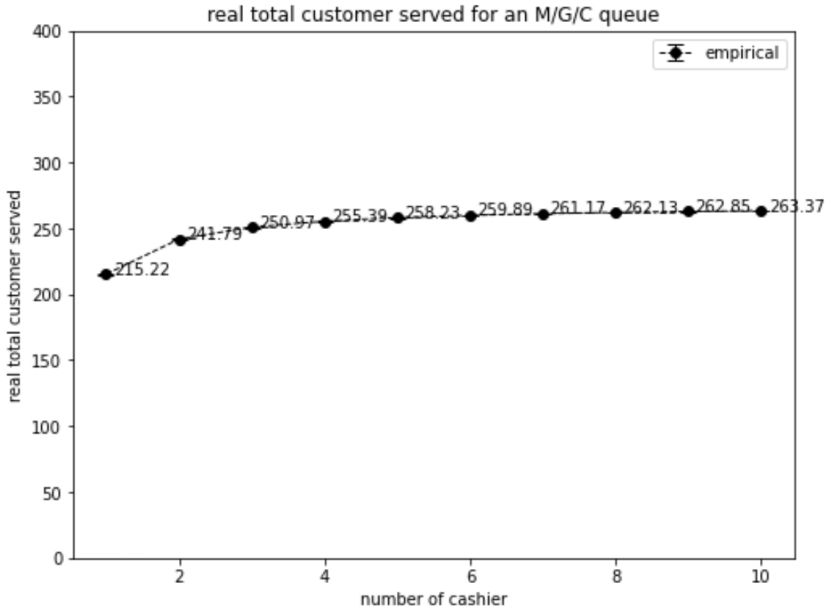


Figure 7. 95% confidence interval for total customers served versus number of cashiers for 400 experiments

We will include this assumption by introducing the drop rate, and we assume the customer drop out rate will grow proportionally with the wait time due to dissatisfaction. We can add a method in simulation for customer dropping, but to simplify, we assume that the drop rate will be 40%, and the number of drop customers = average wait time * drop rate.

From the result, we can see that the total customer served plateaus around three cashiers. This indicates that our total profits will stay the same after three cashiers. From this standpoint, we suggest that the grocery store hire three cashiers. This calculation is a simplified method to maximize the profit, but we might need real-world investigation to understand customers' dropping behavior from the queue.



Limitation

Several limitations don't align with the real-world scenario and theoretical assumptions.

- The independence assumption in the queue theory states that customers will randomly pick a queue regardless of people's choices. Yet. Since customers enter the shortest queue, the independence assumption doesn't hold.

- The average rate assumption doesn't hold either because the average customer arrival rate will change by the time of the day. For instance, grocery stores are usually much busier after 5 pm when people finish work. Therefore, this assumption only holds when we consider a limited time interval of the day.
- An infinite queue is impossible due to a physical store's size limitation.
- It is unrealistic to assume that every cashier provides equal quality of service and doesn't change over time and the line size.
- Customers might leave the queue due to various circumstances. (i.e., when the wait time is too long when they have other duties to fulfill)
- Serving time will depend on the number of items held by the customers. We aren't exactly sure it will follow a normal distribution.

Conclusion

Based on the analysis, we will be hiring more than ten cashiers if possible. If the grocery store is low on hiring budget, we suggest hiring around four cashiers for the best balance between hiring cost and customer wait time. However, we also describe the limitation of this simulation on service time distribution and arrival rate. Considering all the uncertainty and confounding factors, we will suggest testing our recommendations for two weeks in the store and modifying our simulations based on the real-world outcomes.

References

Wikimedia Foundation. (2021, December 20). *M/g/K queue*. Wikipedia. Retrieved September 22, 2022, from https://en.wikipedia.org/wiki/M/G/k_queue

Minerva. CS166 Session 3 - Implementing a simulation. M/G/1 breakout solution.ipynb. Retrieved September 22, 2022, from https://forum.minerva.edu/app/courses/2463/sections/9418/classes/65771

Appendix for common questions

How does the code calculate the average wait time?

- In the normal queue, after we finish serving the customers, we will check if the customer needs extended service

- If they don’t need extended service, we mark it as 0.
    - We create a tuple (wait time, 0) and add it in the wait time list:
      `self.wait_time.append((leaving_time - self.arrival_time.popleft(), 0))`
- If they need extended service, we mark it as 1.
    - Since we don’t know the leaving time after the extended service yet, we append the arrival time in a tuple (arrival time, 1):
      `self.wait_time.append((self.arrival_time.popleft(), 1))`
    - In our ExtendedService object, we record all the leave time after the manager finish serving customers.
    - Since we know that our queue follows a first come, first serve rule, and we only have one manager. Therefore, we know the earliest arrival will always get served first, and leave first. To calculate the wait time, we can subtract the earliest leave time and the earliest arrival time. For example, if the arrival time is :[1, 5, 7, 10], and the leave time is [5, 10, 15, 30], then the wait time will be: [5-1, 10-5, 15-7, 30-10] = [4, 5, 8, 20]

Why do we create an extra class for extended service?

We might wonder why we create extra classes for extended service rather than putting it directly into QueueSystem or Queue class.

We don’t put it in Queue class because we have multiple queues, but we only have one manager that serves people from all the queues. If we put it in the Queue class, we will assume every queue contains an extended queue.

In this case, it seems that putting the extended queue in a QueueSystem class is a good idea. However, we are limited by the python implementation constraints. First, we will only decide if a customer needs an extended service after being served. It hints that we need to call the extended service after serving the customer, a method in the Queue class. Second, we define a Queue class as contained in the QueueSystem class, indicating we can query everything from the queue class but not vice versa. Hence, if we call the extended service after finishing serving the customer, we cannot add QueueSystem class in this method.

Based on the two reasons, it’s logical to put an external class for extended service so both QueueSystem class and Queue class can access it when they need. I will love to know how to design a better structure for the code.

LOs

- cs166-Modeling: Model or interpret a scenario using mathematical equations, network theory, cellular automata, or other techniques.
- cs166-PythonImplementation:
    - I have used OOP to structure my code, and insert print statements for both regular and extended queues to ensure my code is correct.
- cs166-CodeReadability: I have PEP-8 docstrings and comments for my code and create readable variables throughout the assignment.
- cs166-TheoreticalAnalysis: I use M/G/1 as a proxy to test against my code and provide the mathematical output, and also state the constraints when applying the theories.
- cs166-EmpiricalAnalysis: I compare the result between the simulation and the theoretical result. Though they don't macth well, I explain and discuss the intuitions on why simulation might be more reasonable.
- cs166-Professionalism: I created a report with a Python notebook that readers can check. I also include an Appendix section to explain my code design and common confusion.

Acknowledgment

I brainstormed with Chau about the code structure and discussed how to improve the code with Kaly.

Word Count: 2635 words