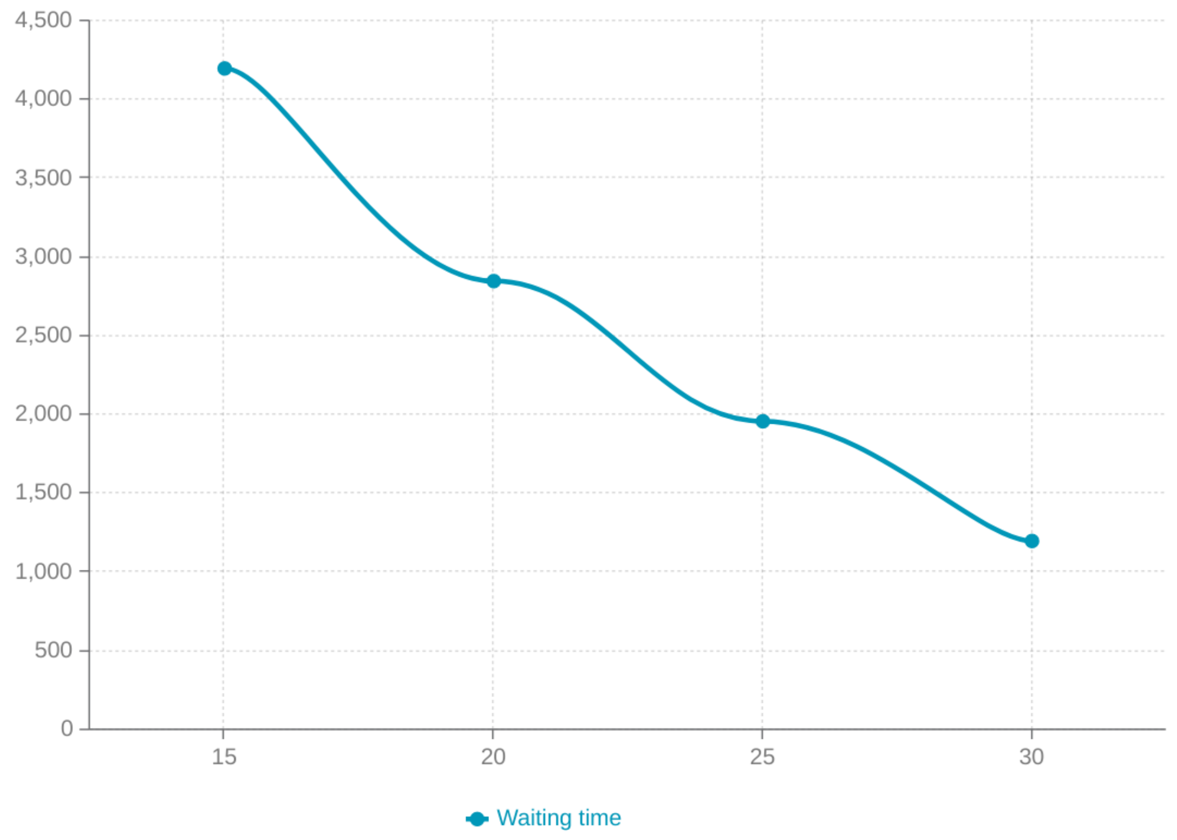
# Theory & Programming Assignment 4:
# The Korean Restaurant Problem

**Vibhanshsu Jain**
**CS19B1027**

**Explanation**:
The complete algorithm is based on the effective use of the semaphore. We are using two semaphore mutex and block for implementing this algorithm.
We are also using a boolean variable must_wait for checking whether the customer needed to wait or not. If the mutex lock is 1, then it will implement the mutex.wait(), which will in turn decrease the value of mutex to 0. Now the code in the if loop will work, which that must_wait is true or eating+1 is greater than x, the number of seats at the restaurant table. If yes then increase waiting and mark must_wait true. After this, again call mutex.signal() and block.wait(). If this condition is false then, increment the waiting and must_wait is true when waiting is true as well as eating is equal to total number of seats available. That means there is a group.

Now, call mutex.wait(), and decrease the counter eating. Now check whether eating is zero or not, if yes then mark the variable k as minimum of x and wait. And decrease the value of both of them by k and call block.signal(k), which in turn increases the value of block.
Also mark mutex_wait true if all are eating, i.e. eating == x.

Now again call mutex.signal



Scale:

X-axis: n

Y-axis: Waiting time in 10^(-3) microseconds

Fixed Parameter:

x = 4

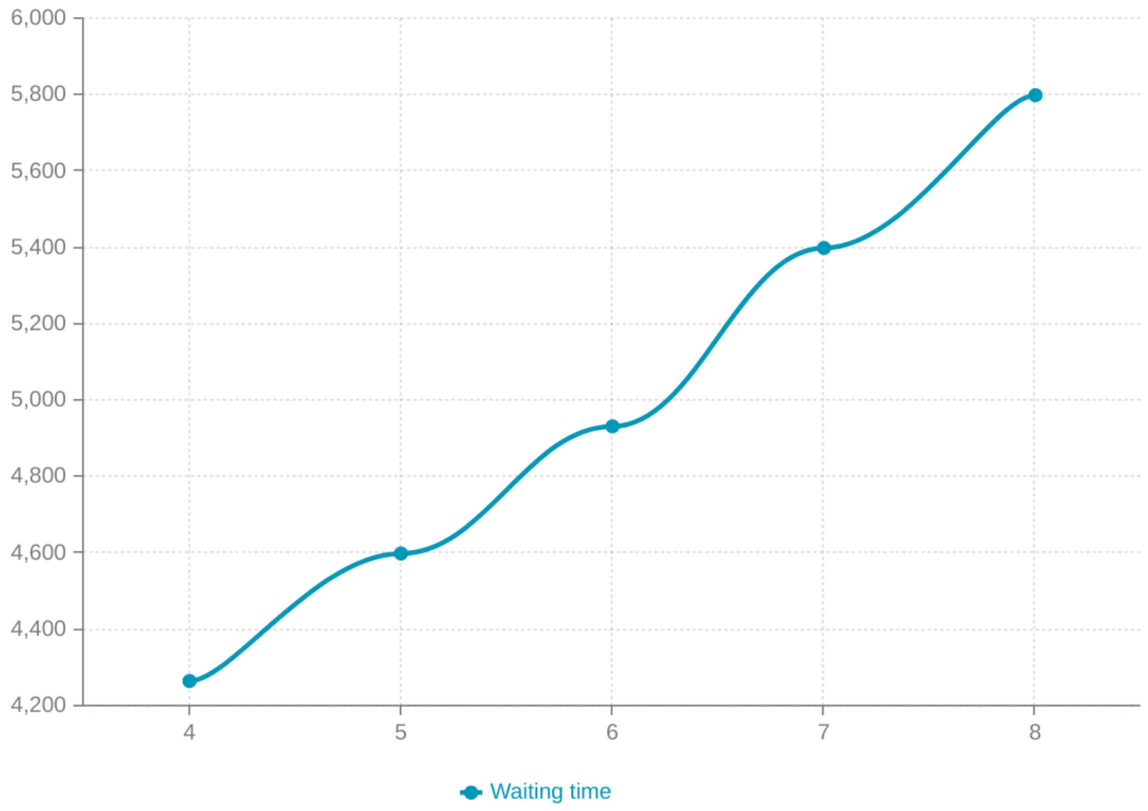
Lambda = 0.1

r = 0.3

I = 0.2

**Observation:**

The waiting time is decreasing as we are increasing the value of n which is absurd, but can be clearly seen this is because of very low values of the other parameters which are kept constant. The parameter, lambda, r and I are kept very low because of which the waiting time is decreasing when the value of n is increased. The waiting time is very less, and decreases for other threads as well, but when more threads are added this decreases the average waiting time.



Scale:

X-axis: x

Y-axis: Waiting time in 10^(-3) microseconds

Fixed Parameter:

x = 4

Lambda = 0.1

r = 0.3

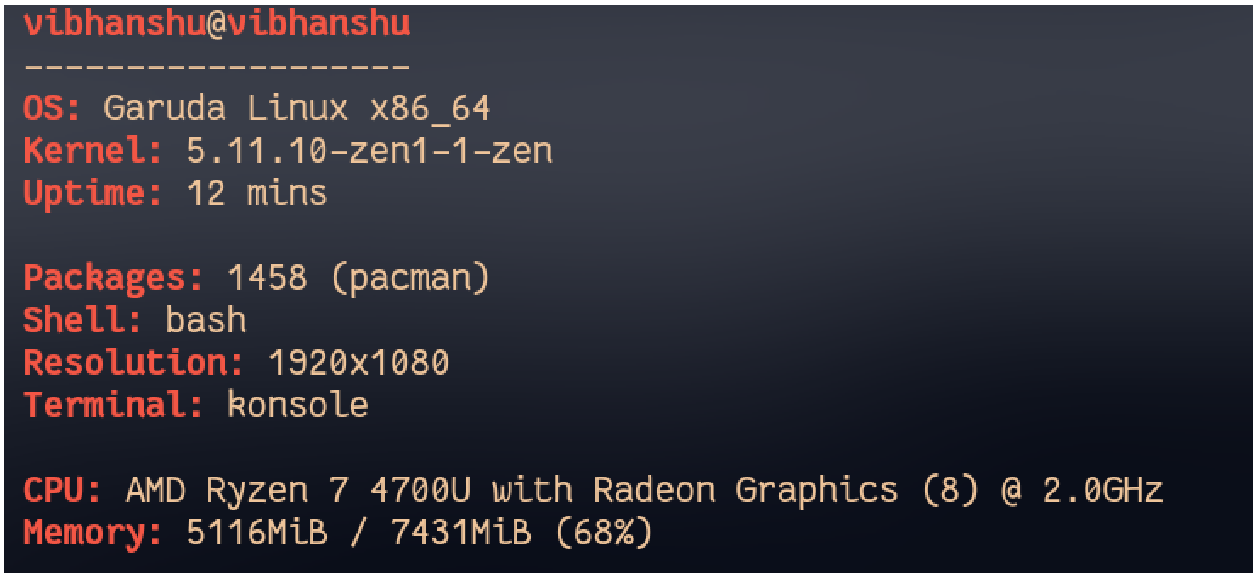
l = 0.2

**Explanation:**

The trend is again highly dependent on the parameters choosen and the system configuration. As our parameters are less, the waiting time is again increasing as we are increasing the number of tables. Because our customer needed to wait for more time and we are increasing the value of x, which in turn increases the average waiting time.

The total average waiting is, 5.

The system which I used to calculate the above results.



```
vibhanshu@vibhanshu
--------------------
OS: Garuda Linux x86_64
Kernel: 5.11.10-zen1-1-zen
Uptime: 12 mins

Packages: 1458 (pacman)
Shell: bash
Resolution: 1920x1080
Terminal: konsole

CPU: AMD Ryzen 7 4700U with Radeon Graphics (8) @ 2.0GHz
Memory: 5116MiB / 7431MiB (68%)
```