CO21BTECH11002
Aayush Kumar
# OS2 Programming Assignment 3

The tetsCS( ) function:

In the testCS( ) function each thread enters the critical section ‘k’ (user defined) times. During each iteration, the threads requests to enter the critical section, at this time the local time is noted. It then waits while some other thread is in the critical section. This is implemented using TAS, CAS and BCAS mutual exclusion algorithms. We take help of a global variable ‘lock’ to implement these algorithms. When lock is 1, that means some thread is inside the CS. For BCAS, we use another array named ‘waiting’ which stores the current waiting state of all the threads. Request to CS is granted in cyclic order in BCAS.

Once a thread has entered the critical section, time is again noted. We now update the total waiting time by all threads and the worst waiting time. This update is done in the critical section, hence, there is no case of data race. The critical section work is simulated using sleep() function. Once the CS work is done, exit time is noted to make sure that mutual exclusion is being followed. After exiting the CS, remainder section work is simulated using sleep() function in a similar way.

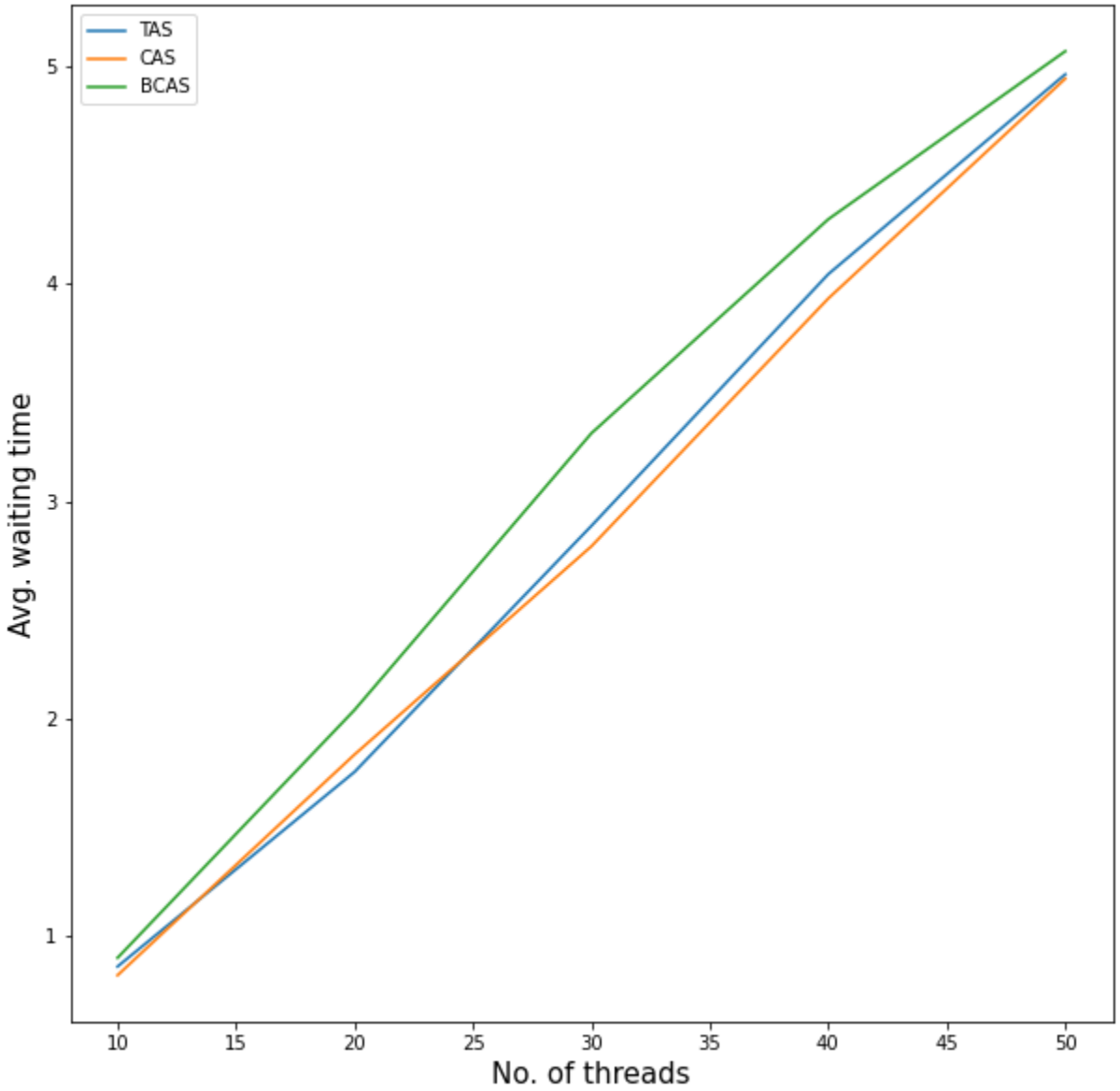
The main( ) function:

We start by reading the value of n - no. of threads and k - no. of times thread enters the CS from the input file. We then create an array of type ‘vals’ for storing the parameters passed in the testCS() function. Then we create n threads and run them. Then wait for threads to finish running and join. Then print the relevant results obtained.

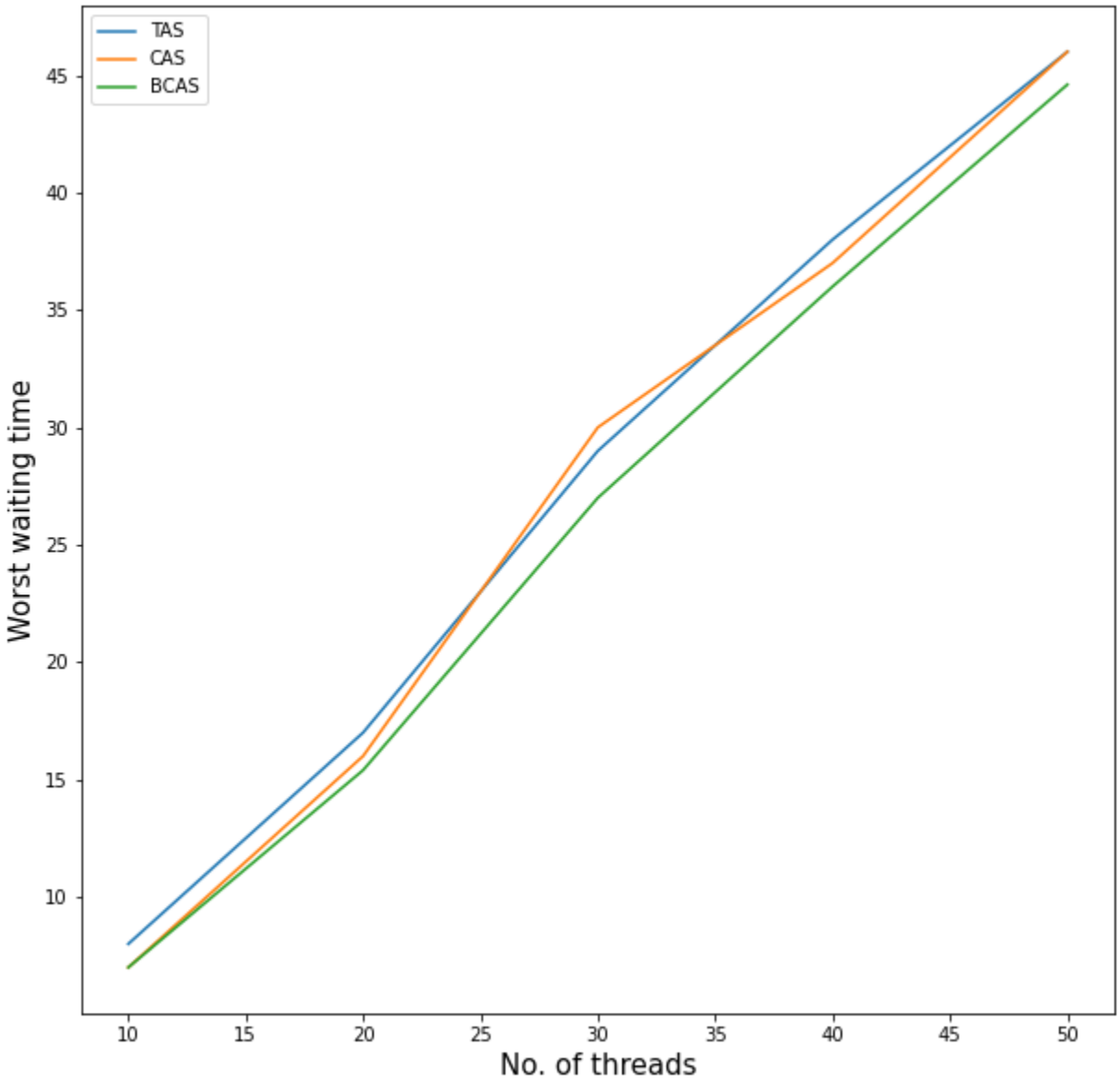
# Results:

After running the program 5 times (taking $\lambda 1 = 0.5, \lambda 2 = 0.5$ ) and taking the average value of time taken (in seconds), we get the following results:

Average waiting time vs no. of threads:



Worst waiting time




## Conclusion:

1. We can see that both average waiting time and worst waiting time increase with the number of threads. This is because the work in CS is done by only one thread at a time. Hence, for large no. of threads, this CS work increases time in a sequential manner.
2. TAS and CAS perform almost similarly but BCAS has a higher average waiting time since it tries to make sure that each thread gets an equal chance

to enter the critical section. This results in longer waiting times before the next request for each thread.

3. TAS and CAS perform similarly but BCAS has lower worst waiting time since it makes sure that no thread is starving.