Pericherla Pranav Varma
CS21BTECH11044

OS-2 OPERATING SYSTEM (CS3523)

PROGRAMMING ASSIGNMENT 4

The Jurassic Park Problem.

REPORT:

I. Low Level Implementation of Code:

- Code starts with semaphore declarations
  car(used for checking car availability),
  arr (used for locking while array car_arr[] is being updated).
- reads “inp-params.txt” file that loads values of
  - Passenger threads -> ps_threads
  - Car threads -> cr_threads
  - λ_p -> the parameter for the exponential wait between 2 successive ride requests made by the passenger.
  - λ_c -> the parameter for the exponential wait between 2 successive ride requests accepted by a car.
  - K -> number of ride requests made by each passenger.
- **#line 154**, Initially all cars are available hence, car_array {which indicates availability of car where car id is array index}, is initialized to 1.
- Pthreads are created **#line 171** where no.of threads = no.of passengers, and each thread is given its passenger id.
- Each thread created above would execute fn – Park().
- Park() –
  - Random value of time with λ_p as t1, λ_c as t2 are generated.
  - Variable car_ride will store total time in µs by that particular passenger in the thread.
  - Creates file “output.txt” if don’t exist and store strings based on required log.
  - **#line 77,** sem_wait(car) would allow passenger to use car if car value > 0. {allow passenger to search free car and get into it noting id of car} *ENTERS CRITICAL SECTION.*
  - **#line 77,** if cars are unavailable, passengers{other process} would busy wait in the line till, some other thread calls sem_post {as this make a car available.}

- **#line 81,** if cars are available then entry of available car in car_arr[] is updated to 0 and car index is stored (implies the current car is taken == not available) this is carefully updated using semaphore{arr} locking.
- **#line 117**,stored car index is used to change entry value in car_arr[] to 1. (This helps in updating car – available after trip finished)
- **#line 121.** By now, passenger has finished his 1st trip hence signals car which implies car is available. Now other passengers waiting in busy wait would take this car and would search for car id in car_arr[]. *ENTERS REMAINDER SECTION.*
- car ride time by each passenger is calculated and stored in variable car_ride. and later used to calculate avg car-ride time.
- Passenger total time is calculated and stored in pass_time (**#line 130**)

- After each thread performs Park() fn. All threads are joined **#line188.**
- Stored passenger total time and car ride time are averaged and displayed in terminal.

II. Graphs and Observations:

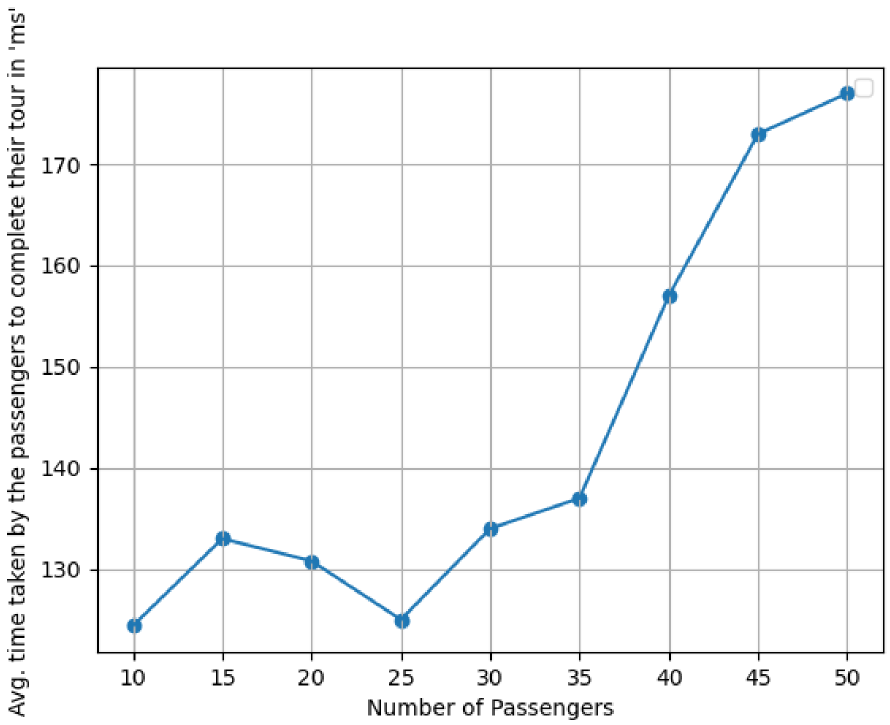
A. Plot 1 :

- Value of $\lambda_p = 5$.
- Value of $\lambda_c = 20$.
- Value of K = 5.
- Value of Cars = 25.

X-axis : Number of Passengers.

Y-axis : T_{avg} taken by passenger to complete tour in museum.

Graph :



Observations:

- We could observe, from n>25. With increase in number of passengers, Avg time taken is also increasing.
- Reason could be, as cars = 25 and if passengers > 25.then as cars are lesser than requried
- passengers must wait till any of the other passengers complete their tour implies car would become available.
- Which implies time taken to complete tour by passenger would increase as he's waiting for cars to get on for a ride.
- for n<25. Cars req are more than number of passengers, thus no passenger would be waiting (busy wait) thus, will have similar average time to complete the tour.

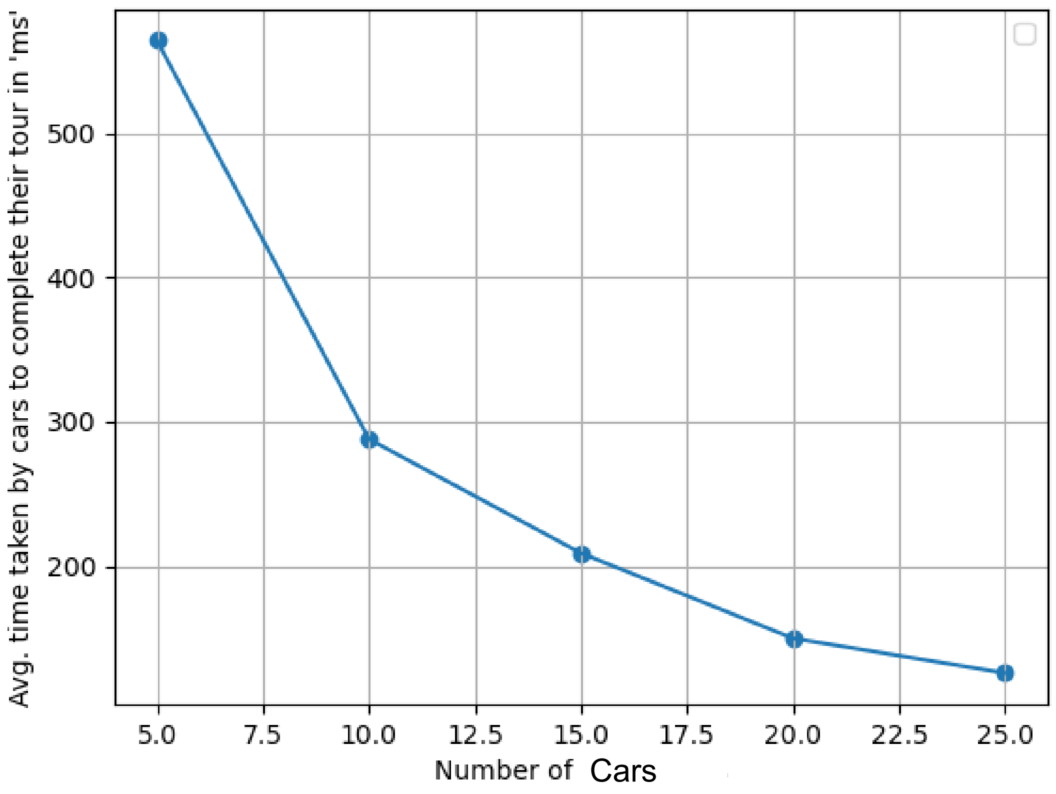
B. Plot 2 :

- Value of $\lambda_p = 5$.
- Value of $\lambda_c = 20$.
- Value of K = 3.
- Value of Passengers = 50.

X-axis : Number of Cars.

Y-axis : T_{avg} taken by car to complete tour in museum.

Graph :



Observations:

- We could observe, as n increases. Avg time taken by car decreases
- This could be because, As Passenger count = 50. With 5 cars, load on each car would be high
- i.e., these 5 cars must ride 50 passengers, thus on increasing cars to 10, now load on each car would decrease
- implies time taken by each car would also gradually decrease
- thus, for passenger count = 50 and cars = 25 have least avg time taken per car when compared to passenger count = 50 and cars = 5.