Aral Açıkalın

20161701036

Simulation Project 3 Report

Purpose of this project is to simulate a single toll booth servicing 3 different roads with different car arrival rates. If toll booth is in service, then the other cars which arrive will form a queue and when the servicing of the current car ends then the next car in the queue will be serviced.

This function below produces a random inter-arrival time in seconds. Takes input lambda value (car arrival rate) in minutes Dividing lambda by 60 gives a arrival rate of cars in seconds.

def randomExp(lambd):

    nextArrival=random.expovariate(lambd/60)

    return nextArrival

First, I create 3 random variables (interarrival time) using 3 different lambda values (arrival rates) and store them in a array then simulation checks the smallest value and starts servicing the minimum value. Then create another random arrival time for the same lane that the serviced car is arrived from and store that value in the array. We repeat this process until our simulation ends. This way I simulate this system correctly.

car=[0,0,0]

lane1=randomExp(lambda1)

car[0]=lane1

lane2=randomExp(lambda2)

car[1]=lane2

lane3=randomExp(lambda3)

car[2]=lane3

when serviced program changes the serviced car arrival time to 0 and this block of code checks which index of the array is 0 then with that information creates the correct car arrival time for correct lambda value

if not(lastArrivalLane1+lane1 in car):

       lane1=randomExp(lambda1)

       car[0]=lastArrivalLane1+lane1

elif not(lastArrivalLane2+lane2 in car):

       lane2=randomExp(lambda2)

       car[1]=lastArrivalLane2+lane2

elif not(lastArrivalLane3+lane3 in car):

       lane3=randomExp(lambda3)

       car[2]=lastArrivalLane3+lane3

this code below checks which car should be serviced and then creates a service time for that car. Then checks if previous car is in service or not. If there is a car in service, the new car starts service as soon as the last car ends service.

Then calculating the service begin time, queue time, service end time, time in system, system idle time. And finally putting all this info to an array which is array of serviced cars.

if(car[0]<car[1] and car[0]<car[2]):

            serviceTime=randomExp(mean1)

            lastArrivalLane1=car[0]

            idleTime=0

            prevServiceEndTime=cars[i-1][4]

#if booth is idle then we calculate idle time by arrival time- previous service end time

            if(prevServiceEndTime<lastArrivalLane1):

                #if booth is idle then service begins immidiatly when car arrives

                serviceBegins=lastArrivalLane1

                idleTime=lastArrivalLane1-prevServiceEndTime

            else:

                #otherwise service begins after previous service ends

                serviceBegins=cars[i-1][4]

            queueTime=serviceBegins-lastArrivalLane1

            serviceEndTime=serviceBegins+serviceTime

            timeinSystem=queueTime+serviceTime

whichlane="Lane 1"

            cars.append([lastArrivalLane1,serviceTime,serviceBegins,queueTime,serviceEndTime,timeinSystem,idleTime,whichLane])

            lane1CarCount+=1 #for calculating propobality cars coming from this lane

            car[0]=0 #this car enters service

(there are also else if’s that checks for the other cars if they are smallest or not.)

This code here prints the array of serviced cars as a table if 5th input is true. Used Pandas library for creating a table from the array. Gave each column a name and removed the limit of table rows by using set\_option function.

if(debug==True):

   pandas.set\_option('display.max\_rows', None)

   pandas.set\_option('display.max\_columns', None)

   pandas.set\_option('display.width', None)

   pandas.set\_option('display.max\_colwidth', -1)

   table = pandas.DataFrame(cars, columns = ["Arrival Time","Service Time","Service Begin Time","Queue Time","Service Ending Time","Time in System","Server Idle Time"])

   print(table)

this code counts the cars and calculates average queue time. countCars starts from -1 because the first car in the array is actually not a car. All values of that car is 0 and that element of array is there because if there are no cars in the system then we cant check the last cars service eding time. With this element I prevent an array out of bounds array from happening.

queTimeSum=0

    countCars=-1 #starts from -1 because I add a empty element into the cars array in order to initialize for the first time

    #this for statement is for calculating average queue time

    for car in cars:

        queTimeSum+=car[3]

        countCars+=1

    queTimeAverage=queTimeSum/countCars

    timeinSystemSum=0

this code calculates the avarage time in system

for car in cars:

        timeinSystemSum+=car[5]

    averageTimeinSystemTime=timeinSystemSum/countCars

this code calculates the percentage of the lane arrivals

percentlane1=(lane1CarCount/countCars)\*100

    percentlane2=(lane2CarCount/countCars)\*100

    percentlane3=(lane3CarCount/countCars)\*100

**FULL CODE:**

import math

import random

from timeit import default\_timer as timer

import pandas

def randomExp(lambd):

    '''Creating a random inter-arival time in seconds'''

    #creates a random inter-arrival variable

    #dividing lambda by 60 gives the rate of cars arriving per second so inter-arrival time is in seconds

    nextArrival=random.expovariate(lambd/60)

    return nextArrival

def simulation(lambda1,lambda2,lambda3,mean1,debug):

    '''

    takes 3 lambda values and 1 mean value for 3 lanes and 1 toll booth then

    simulates the system, if as a 5th input true is given it generates the simulation as table

    '''

    #for counting the percentage of cars arriving from the different roads

    lane1CarCount=0

    lane2CarCount=0

    lane3CarCount=0

    #for holding the last arrival times for lanes

    lastArrivalLane1=0

    lastArrivalLane2=0

    lastArrivalLane3=0

    #this array holds all the infos of cars

    #stores respectively;

    #arrival-time, service-time, service begin time, queue time, service end time, time in sistem.

    cars=[[0,0,0,0,0,0,0]] #initialized with all 0's because when checking for the last cars service end time it should be 0 in order to not effect the simulation.

    #this array holds the next inter-arrival time for 3 lanes.

    car=[0,0,0]

    #this block of code creates first 3 inter-arrival time.

    lane1=randomExp(lambda1)

    car[0]=lane1

    lane2=randomExp(lambda2)

    car[1]=lane2

    lane3=randomExp(lambda3)

    car[2]=lane3

    for i in range(1,1000001):

        #this for begins from 1st index because when checking for last cars service end time 0-1 gives out of bounds error for the array.

        #for the same reason it also ends on +1 the desired number.

        #these if/if-else blocks generates a new car inter-arrival time when the last car in lane serviced

        if not(lastArrivalLane1+lane1 in car):

            lane1=randomExp(lambda1)

            car[0]=lastArrivalLane1+lane1

        elif not(lastArrivalLane2+lane2 in car):

            lane2=randomExp(lambda2)

            car[1]=lastArrivalLane2+lane2

        elif not(lastArrivalLane3+lane3 in car):

            lane3=randomExp(lambda3)

            car[2]=lastArrivalLane3+lane3

        #here we check which car is next to get serviced.

        if(car[0]<car[1] and car[0]<car[2]):

            serviceTime=randomExp(mean1)

            lastArrivalLane1=car[0]

            idleTime=0

            prevServiceEndTime=cars[i-1][4]

            #if booth is idle then we calculate idle time by arrival time- previous service end time

            if(prevServiceEndTime<lastArrivalLane1):

                #if booth is idle then service begins immidiatly when car arrives

                serviceBegins=lastArrivalLane1

                idleTime=lastArrivalLane1-prevServiceEndTime

            else:

                #otherwise service begins after previous service ends

                serviceBegins=cars[i-1][4]

            queueTime=serviceBegins-lastArrivalLane1

            serviceEndTime=serviceBegins+serviceTime

            timeinSystem=queueTime+serviceTime

            cars.append([lastArrivalLane1,serviceTime,serviceBegins,queueTime,serviceEndTime,timeinSystem,idleTime])

            lane1CarCount+=1 #for calculating propobality cars coming from this lane

            car[0]=0 #this car enters service

        elif(car[1]<car[0] and car[1]<car[2]):

            serviceTime=randomExp(mean1)

            lastArrivalLane2=car[1]

            idleTime=0

            prevServiceEndTime=cars[i-1][4]

            if(prevServiceEndTime<lastArrivalLane2):

                #if booth is idle service begins at arrival time of the var

                serviceBegins=lastArrivalLane2

                idleTime=lastArrivalLane2-prevServiceEndTime

            else:

                #otherwise service begins after previous service ends

                serviceBegins=cars[i-1][4]

            queueTime=serviceBegins-lastArrivalLane2

            serviceEndTime=serviceBegins+serviceTime

            timeinSystem=queueTime+serviceTime

            cars.append([lastArrivalLane2,serviceTime,serviceBegins,queueTime,serviceEndTime,timeinSystem,idleTime])

            lane2CarCount+=1 #for calculating propobality cars coming from this lane

            car[1]=0 #this car enters service

        elif(car[2]<car[0] and car[2]<car[1]):

            serviceTime=randomExp(mean1)

            lastArrivalLane3=car[2]

            idleTime=0

            prevServiceEndTime=cars[i-1][4]

            if(prevServiceEndTime<lastArrivalLane3):

                #if booth is idle service begins at arrival time of the var

                serviceBegins=lastArrivalLane3

                idleTime=lastArrivalLane3-prevServiceEndTime

            else:

                #otherwise service begins after previous service ends

                serviceBegins=cars[i-1][4]

            queueTime=serviceBegins-lastArrivalLane3

            serviceEndTime=serviceBegins+serviceTime

            timeinSystem=queueTime+serviceTime

            #stores every calculated data for the car that serviced in an array as an array.

            cars.append([lastArrivalLane3,serviceTime,serviceBegins,queueTime,serviceEndTime,timeinSystem,idleTime])

            lane3CarCount+=1 #for calculating propobality cars coming from this lane

            car[2]=0  #this car enters service

    #install pandas library by pip install pandas

    if(debug==True):

        #setting panda to show more than 10 rows of table and to show more than 5 collumns

        pandas.set\_option('display.max\_rows', None)

        pandas.set\_option('display.max\_columns', None)

        pandas.set\_option('display.width', None)

        pandas.set\_option('display.max\_colwidth', -1)

        #giving cars array and setting the name of the collumns.

        table = pandas.DataFrame(cars, columns = ["Arrival Time","Service Time","Service Begin Time","Queue Time","Service Ending Time","Time in System","Server Idle Time"])

        print(table)

    queTimeSum=0

    countCars=-1 #starts from -1 because I add a empty element into the cars array in order to initialize for the first time

    #this for statement is for calculating average queue time

    for car in cars:

        queTimeSum+=car[3]

        countCars+=1

    queTimeAverage=queTimeSum/countCars

    timeinSystemSum=0

    #this for statement is for calculating average time in system

    for car in cars:

        timeinSystemSum+=car[5]

    averageTimeinSystemTime=timeinSystemSum/countCars

    percentlane1=(lane1CarCount/countCars)\*100

    percentlane2=(lane2CarCount/countCars)\*100

    percentlane3=(lane3CarCount/countCars)\*100

print("Number of Cars: ",countCars)

    print("Average Time in System: ",averageTimeinSystemTime," seconds.")

    print("Average Queue Time: ",queTimeAverage, " seconds.")

    print("Lane 1 Percentage: ",percentlane1,"%")

    print("Lane 2 Percentage: ",percentlane2,"%")

    print("Lane 3 Percentage: ",percentlane3,"%")

def main():

    lambda1=int(input("Enter λ1: "))

    lambda2=int(input("Enter λ2: "))

    lambda3=int(input("Enter λ3: "))

    mean=int(input("Enter µ: "))

debug=bool(int(input("Is Debug Mode (0 means false, 1 means true): ")))

    start=timer() #for outputing the run time of the code

    simulation(lambda1,lambda2,lambda3,mean,False)

    print("")

    print("Time Elapsed: ",timer()-start," seconds.")

main()

Enter λ1: 5

Enter λ2: 2

Enter λ3: 3

Enter µ: 12

Is Debug Mode (0 means false, 1 means true): 0

When I give my program the inputs above simulation gives the outputs below;

Number of Cars: 10000

Average Time in System: 28.59422825462932 seconds.

Average Queue Time: 23.519723191593656 seconds.

Lane 1 Percentage: 49.69 %

Lane 2 Percentage: 19.78 %

Lane 3 Percentage: 30.53 %

Number of Cars: 100000

Average Time in System: 29.20627689006213 seconds.

Average Queue Time: 24.194830450033628 seconds.

Lane 1 Percentage: 49.891000000000005 %

Lane 2 Percentage: 20.152 %

Lane 3 Percentage: 29.957 %

Number of Cars: 1000000

Average Time in System: 30.16977951250809 seconds.

Average Queue Time: 25.16708842767486 seconds.

Lane 1 Percentage: 49.8746 %

Lane 2 Percentage: 20.034 %

Lane 3 Percentage: 30.0914 %

I ran the simulation for 10000 ,100000 and 1000000 cars and as I increase the number of cars average queue time and average time in system converges. Average time in system converges to 30 and average queue time converges to 25.

And the difference of average time in system and average queue time is 5. That also makes sense because toll booth serves a car in average of 5 seconds *(12 car/minute service rate, means it services a car in 5 seconds)*. If a car waits in queue for 25 seconds and service time for that car is 5 seconds, then queue time plus service time gives time in system which is 30 in this case.

Lane percentages match the lambda values perfectly. Lambda value of lane 1 is 5 and the percentage of cars arriving from that lane is nearly 50%. Lambda value of lane 2 is 2 and the percentage of cars arriving from that lane is nearly 20%. Lambda value of lane 3 is 3 and the percentage of cars arriving from that lane is nearly 30%.

From these results a believe my simulation runs correctly.