IE306 2nd HOMEWORK

1.11.2022

Ömercan Mısırlıoğlu

2020402261

ABSTRACT

In this homework, in the first question a hand simulation is completed according to the way taught in the class. In the first part of the second question, the results obtained in the first question are calculated and tested via computer program and the results obtained have been compared to the hand-simulation results. The comparison results show that hand-simulation and computer program results are the same, which shows a consistent and true simulation. Additionally, for question 2-b a simulator is coded and results have been reported. Finally, the integrity of the computer program has been tested and compared with the Little's Law formula. Results turned out to be same, which shows the simulation in 2-b works as intended and correctly. Further comments, program codes, and reports are provided in the report along with the programming codes that are in the .zip file.

Question 1)

- In this question, event-scheduling algorithm is used while doing the hand-simulation, and the following results are obtained.

Q1)

Time	LQ(t)	$LS_1(t)$	$LS_2(t)$	IntArr	St_1	St_2	Future Event List A	Future Event List D_1	Future Event List D_2	Q, S_1, S_2 Set	Time In Queue
0.00	0.00	1.00	0.00	9.80	17.47	-	9.80	17.47	∞	0 – 0	0
9.80	0.00	1.00	1.00	16.32	-	17.00	26.12	17.47	21.80	0 – 0, 9.8	0
17.47	0.00	0.00	1.00	-	-	-	26.12	∞	21.80	0 – 9.8	0
21.80	0.00	0.00	0.00	-	-	-	26.12	∞	∞	0 –	0
26.12	0.00	1.00	0.00	6.20	17.36	-	30.32	41.48	∞	0 – 24.12	0
30.32	0.00	1.00	1.00	15.75	-	22.00	46.07	41.48	52.32	0 – 24.12, 30.32	0
35.00	0.00	1.00	1.00	-	-	-	46.07	41.48	52.32	0 – 24.12, 30.32	0

Random numberss: 0.380 Arrival 1
0.456 Service 1.1
0.832 Arrival 2
0.391 Service 2.1
0.020 Arrival 3
0.480 Service 1.2
0.875 Arrival 4
0.759 Service 2.2
0.905
0.593
0.560

Question 2a)

In this question, a computer program is coded and prepared for simulating the situation in question 1 for 35 minutes, and the following results and the codes are obtained.

- The code for question 2a

```
import numpy as np


#I CREATED A CLASS CALLED SIMULATION TO EASILY CREATE THE MODEL IN OBJECT ORIENTED MANNER

random = [0.380, 0.496, 0.832, 0.391, 0.020, 0.480, 0.975, 0.759, 0.905, 0.593, 0.560] #these are the random values
class SimQuestion2b:
    def __init__(self): #this constructor initializes the attributes of the simulation at the beginning, with the help of this, I can control t
        self.num_in_sv1 = 1 #number of customers in server 1
        self.num_in_sv2 = 0 #number of customers in server 2
        self.num_in_queue = 0 #number of customers in queue
        self.sv1_usagetime = 0 #totalusagetime of server 1
        self.sv2_usagetime = 0 #totalusagetime of server 2
        self.sv1_utilization = 0 #utilization of server 1
        self.sv2_utilization = 0 #utilization of server 2


        self.clock = 0.0 #system clock
        self.t_arrival = self.generate_interarrival() #the first arrival generated at time 0.0
        self.t_depart1 = self.generate_svtime1() #service time of the customer that is in server 1 when simulation begins
        self.t_depart2 = float('inf') #since it is empty, service time is infinite for server 2

        self.num_arrivals = 0 #total number of arrivals
        self.num_departs = 0 #total number of departures
        self.total_wait = 0.0 #total waiting time in queue
        self.avg_queue_time = 0.0
        self.tot_usage_customer = 0 #total time that customers stayed in the system
        self.avg_num_cus = 0 #average number of customers in the system
        self.totalqueuecust = 0 #total number of customers waited in the queue
        self.total_num_cus = 1 #total number of customers that have used the system



    def advance_time(self):
        t_event= min(self.t_arrival, self.t_depart1, self.t_depart2) #choosing the closest event to jump
        self.total_wait += self.num_in_queue*(t_event- self.clock) #total queue waiting time update
        self.tot_usage_customer += (self.num_in_sv1+self.num_in_sv2+self.num_in_queue)*(t_event-self.clock) #calculating the total usage of sy:
        self.reporter()
        self.clock = t_event #jumping to the next event
        self.avg_num_cus = self.tot_usage_customer/self.clock #calculating average number of customers in the system
        self.avg_queue_time = self.total_wait/self.clock #calculating the average queue time
        self.sv1_utilization = self.sv1_usagetime/self.clock
        self.sv2_utilization = self.sv2_usagetime/self.clock
        if ((self.t_arrival <= self.t_depart1) and (self.t_arrival <= self.t_depart2)):  #there are 4 probable situations here and I use these
            self.handle_arrival_event()                                                   #to give priority to the earliest one
            self.total_num_cus += 1 #updating total number of customers
        elif ((self.t_arrival > self.t_depart1) and (self.t_arrival <= self.t_depart2)):
            self.handle_depart1_event()
        elif (self.t_depart1 <= self.t_depart2): #here departure from sv 1 has a priority
            self.handle_depart1_event()
        elif (self.t_depart1 > self.t_depart2):
            self.handle_depart2_event()
    def handle_arrival_event(self):
        if(self.num_in_queue <= 0 and self.num_in_sv1 < 1): #here the first server has a priority, when a customer comes first
            self.num_in_sv1 += 1
            self.t_arrival = self.clock + self.generate_interarrival()                           # the availability of th
            svtime1 = self.generate_svtime1() #
            self.t_depart1 = self.clock + svtime1 #these 3 lines store the total time the server 1 is used

            self.sv1_usagetime += svtime1 #
            self.num_arrivals += 1


        elif(self.num_in_queue <= 0 and self.num_in_sv2 < 1): #if there is no queue and the first server is busy, second server is chosen and
            self.num_in_sv2 += 1
            self.t_arrival = self.clock + self.generate_interarrival()
            svtime2 = self.generate_svtime2() #
            self.t_depart2 = self.clock + svtime2 #these 3 lines store the total time the server 2 is used

            self.sv2_usagetime += svtime2 #
            self.num_arrivals += 1


        else: #if both of the servers are busy, person enters the queue
            self.num_in_queue += 1
            self.t_arrival = self.clock + self.generate_interarrival()
            self.totalqueuecust += 1
            self.num_arrivals += 1
```

```
    def handle_depart1_event(self): #function to handle the departures of server 1
        self.num_in_sv1 -= 1
        self.num_departs += 1
        if (self.num_in_queue > 0):
            svtime1 = self.generate_svtime1() #
            self.t_depart1 = self.clock + svtime1 #these 3 lines store the total time the server 2 is used

            self.sv1_usagetime += svtime1 #
            self.num_in_sv1 += 1
            self.num_in_queue -= 1
        else:
            self.t_depart1 = float('inf')

    def handle_depart2_event(self): #function to handle the departures of server 2
        self.num_in_sv2 -= 1
        self.num_departs += 1
        if (self.num_in_queue > 0 and self.num_in_sv1 > 0):
            svtime2 = self.generate_svtime2() #
            self.t_depart2 = self.clock + svtime2 #these 3 lines store the total time the server 2 is used

            self.sv2_usagetime += svtime2 #
            self.num_in_sv2 += 1
            self.num_in_queue -= 1
        else:
            self.t_depart2 = float('inf')

    def generate_interarrival(self):
        x = random[0]
        random.remove(random[0])
        print("Arrival at: ", self.clock+(6+x*10))
        return (6+x*10)
```

```
    def generate_svtime1(self):
        x = random[0]
        random.remove(random[0])
        print("Departure form sv 1 at: ", self.clock+(14+x*7))
        return (14+x*7)


    def generate_svtime2(self):
        x = random[0]
        if x < 0.18:
            random.remove(random[0])
            print("Departure form sv 2 at: ", self.clock+8)
            return (8)

        elif x < 0.48:
            random.remove(random[0])
            print("Departure form sv 2 at: ", self.clock+12)
            return (12)

        elif x < 0.78:
            random.remove(random[0])
            print("Departure form sv 2 at: ", self.clock+22)
            return (22)

        else:
            random.remove(random[0])
            print("Departure form sv 2 at: ", self.clock+33)
            return (33)

    def reporter(self):
            print("----------------------------------")
            print("LS1 at", self.clock, "= ", self.num_in_sv1)
            print("LS2 at", self.clock, "= ", self.num_in_sv2)
            print("LSQ at", self.clock, "= ", self.num_in_queue)
            print("----------------------------------")

#BELOW I CREATED 4 DIFFERENT RANDOM SEEDS FOR 4 SimQuestion2bS AS DESCRIBED
np.random.seed(1)
s1 = SimQuestion2b()


# REPORTING AND PRINTING PART IS BELOW

while 1:
    if s1.clock<=35:
        s1.advance_time()
    else:
        break
```

```
Arrival at:  9.8
Departure form sv 1 at:  17.472
-----------------------------------
LS1 at 0.0 =  1
LS2 at 0.0 =  0
LSQ at 0.0 =  0
-----------------------------------
Arrival at:  24.12
Departure form sv 2 at:  21.8
-----------------------------------
LS1 at 9.8 =  1
LS2 at 9.8 =  1
LSQ at 9.8 =  0
-----------------------------------
-----------------------------------
LS1 at 17.472 =  0
LS2 at 17.472 =  1
LSQ at 17.472 =  0
-----------------------------------
-----------------------------------
LS1 at 21.8 =  0
LS2 at 21.8 =  0
LSQ at 21.8 =  0
-----------------------------------
Arrival at:  30.32
Departure form sv 1 at:  41.480000000000004
-----------------------------------
LS1 at 24.12 =  1
LS2 at 24.12 =  0
LSQ at 24.12 =  0
-----------------------------------
Arrival at:  46.07
Departure form sv 2 at:  52.32
-----------------------------------
LS1 at 30.32 =  1
LS2 at 30.32 =  1
LSQ at 30.32 =  0
-----------------------------------
```

Question 2b)

In this part of the question, a computer program for simulating a wide range of time has been prepared and coded. The results obtained are shown below along with the code.

- The code of question 2b

```python
import numpy as np


#I CREATED A CLASS CALLED SIMULATION TO EASILY CREATE THE MODEL IN OBJECT ORIENTED MANNER

class SimQuestion2b:
    def __init__(self): #this constructor initializes the attributes of the simulation at the beginning, with the help of this, I can control t
        self.num_in_sv1 = 1 #number of customers in server 1
        self.num_in_sv2 = 0 #number of customers in server 2
        self.num_in_queue = 0 #number of customers in queue
        self.sv1_usagetime = 0 #totalusagetime of server 1
        self.sv2_usagetime = 0 #totalusagetime of server 2
        self.sv1_utilization = 0 #utilization of server 1
        self.sv2_utilization = 0 #utilization of server 2


        self.clock = 0.0 #system clock
        self.t_arrival = self.generate_interarrival() #the first arrival generated at time 0.0
        self.t_depart1 = self.generate_svtime1() #service time of the customer that is in server 1 when simulation begins
        self.t_depart2 = float('inf') #since it is empty, service time is infinite for server 2

        self.num_arrivals = 0 #total number of arrivals
        self.num_departs = 0 #total number of departures
        self.total_wait = 0.0 #total waiting time in queue
        self.avg_queue_time = 0.0
        self.tot_usage_customer = 0 #total time that customers stayed in the system
        self.avg_num_cus = 0 #average number of customers in the system
        self.totalqueuecust = 0 #total number of customers waited in the queue
        self.total_num_cus = 1 #total number of customers that have used the system

    def advance_time(self):
        t_event= min(self.t_arrival, self.t_depart1, self.t_depart2) #choosing the closest event to jump
        self.total_wait += self.num_in_queue*(t_event- self.clock) #total queue waiting time update
        self.tot_usage_customer += (self.num_in_sv1+self.num_in_sv2+self.num_in_queue)*(t_event-self.clock) #calculating the total usage of sy:
        self.clock = t_event #jumping to the next event
        self.avg_num_cus = self.tot_usage_customer/self.clock #calculating average number of customers in the system
        self.avg_queue_time = self.total_wait/self.clock #calculating the average queue time
        self.sv1_utilization = self.sv1_usagetime/self.clock
        self.sv2_utilization = self.sv2_usagetime/self.clock
        if ((self.t_arrival <= self.t_depart1) and (self.t_arrival <= self.t_depart2)):  #there are 4 probable situations here and I use these
            self.handle_arrival_event()                                                     #to give priority to the earliest one
            self.total_num_cus += 1 #updating total number of customers
        elif ((self.t_arrival > self.t_depart1) and (self.t_arrival <= self.t_depart2)):
            self.handle_depart1_event()
        elif (self.t_depart1 <= self.t_depart2): #here departure from sv 1 has a priority
            self.handle_depart1_event()
        elif (self.t_depart1 > self.t_depart2):
            self.handle_depart2_event()


    def handle_arrival_event(self):
        if(self.num_in_queue <= 0 and self.num_in_sv1 < 1): #here the first server has a priority, when a customer comes first
            self.num_in_sv1 += 1                                                    # the availability of the 1st server is checked, if availab:
            svtime1 = self.generate_svtime1() #
            self.t_depart1 = self.clock + svtime1 #these 3 lines store the total time the server 1 is used
            self.sv1_usagetime += svtime1 #
            self.num_arrivals += 1
            self.t_arrival = self.clock + self.generate_interarrival()
        elif(self.num_in_queue <= 0 and self.num_in_sv2 < 1): #if there is no queue and the first server is busy, second server is chosen and t
            self.num_in_sv2 += 1
            svtime2 = self.generate_svtime2() #
            self.t_depart2 = self.clock + svtime2 #these 3 lines store the total time the server 2 is used
            self.sv2_usagetime += svtime2 #
            self.num_arrivals += 1
            self.t_arrival = self.clock + self.generate_interarrival()
        else: #if both of the servers are busy, person enters the queue
            self.num_in_queue += 1
            self.totalqueuecust += 1
            self.num_arrivals += 1
            self.t_arrival = self.clock + self.generate_interarrival()


    def handle_depart1_event(self): #function to handle the departures of server 1
        self.num_in_sv1 -= 1
        self.num_departs += 1
        if (self.num_in_queue > 0):
            svtime1 = self.generate_svtime1() #
            self.t_depart1 = self.clock + svtime1 #these 3 lines store the total time the server 2 is used
            self.sv1_usagetime += svtime1 #
            self.num_in_sv1 += 1
            self.num_in_queue -= 1
        else:
            self.t_depart1 = float('inf')

    def handle_depart2_event(self): #function to handle the departures of server 2
        self.num_in_sv2 -= 1
        self.num_departs += 1
        if (self.num_in_queue > 0 and self.num_in_sv1 > 0):
            svtime2 = self.generate_svtime2() #
            self.t_depart2 = self.clock + svtime2 #these 3 lines store the total time the server 2 is used
            self.sv2_usagetime += svtime2 #
            self.num_in_sv2 += 1
            self.num_in_queue -= 1
        else:
            self.t_depart2 = float('inf')
```

```python
    def generate_interarrival(self):
        return np.random.uniform(6,16)

    def generate_svtime1(self):
        return np.random.uniform(14,21)

    def generate_svtime2(self):
        x = np.random.uniform(0,1)
        if x < 0.18:
            return (8)
        elif x < 0.48:
            return (12)
        elif x < 0.78:
            return (22)
        else:
            return (33)


#BELOW I CREATED 4 DIFFERENT RANDOM SEEDS FOR 4 SimQuestion2bS AS DESCRIBED
np.random.seed(1)
s1 = SimQuestion2b()
np.random.seed(2)
s2 = SimQuestion2b()
np.random.seed(3)
s3 = SimQuestion2b()
np.random.seed(4)
s4 = SimQuestion2b()
```

```python
# REPORTING AND PRINTING PART IS BELOW

while 1:
    if s1.clock<=7000:
        s1.advance_time()
    else:
        print("Results for seed 1")
        print("Average queue time is: " , s1.avg_queue_time)
        print("Average number of customers in the system is: " , s1.avg_num_cus)
        print("The average utilization of server 1: " , s1.sv1_utilization)
        print("The average utilization of server 2: " , s1.sv2_utilization)
        print("Probability of Customer not waiting in the queue: ", 1-s1.totalqueuecust/s1.total_num_cus)
        print("------------------------------------------------------------")
        break

while 1:
    if s2.clock<=7000:
        s2.advance_time()
    else:
        print("Results for seed 2")
        print("Average queue time is: " , s2.avg_queue_time)
        print("Average number of customers in the system is: " , s2.avg_num_cus)
        print("The average utilization of server 1: " , s2.sv1_utilization)
        print("The average utilization of server 2: " , s2.sv2_utilization)
        print("Probability of Customer not waiting in the queue: ", 1-s2.totalqueuecust/s2.total_num_cus)
        print("------------------------------------------------------------")
        break
```

```python
while 1:
    if s3.clock<=7000:
        s3.advance_time()
    else:
        print("Results for seed 3")
        print("Average queue time is: " , s3.avg_queue_time)
        print("Average number of customers in the system is: " , s3.avg_num_cus)
        print("The average utilization of server 1: " , s3.sv1_utilization)
        print("The average utilization of server 2: " , s3.sv2_utilization)
        print("Probability of Customer not waiting in the queue: ",1-s3.totalqueuecust/s3.total_num_cus)
        print("------------------------------------------------------------")
        break

while 1:
    if s4.clock<=7000:
        s4.advance_time()
    else:
        print("Results for seed 4")
        print("Average queue time is: " , s4.avg_queue_time)
        print("Average number of customers in the system is: " , s4.avg_num_cus)
        print("The average utilization of server 1: " , s4.sv1_utilization)
        print("The average utilization of server 2: " , s4.sv2_utilization)
        print("Probability of Customer not waiting in the queue: ", 1-s4.totalqueuecust/s4.total_num_cus)
        print("------------------------------------------------------------")
        print("------------------------------------------------------------")
        print("------------------------------------------------------------")
        break

print("AVERAGES OF 4 DIFFERENT SEEDS AND ESTIMATE FOR THE SYSTEM")
print("Queue time estimate: " , (s4.avg_queue_time+s3.avg_queue_time+s2.avg_queue_time+s1.avg_queue_time)/4)
print("Average number of customers in the system estimate: " , (s4.avg_num_cus+s3.avg_num_cus+s2.avg_num_cus+s1.avg_num_cus)/4)
print("The average utilization of server 1 estimate: " , (s4.sv1_utilization+s3.sv1_utilization+s2.sv1_utilization+s1.sv1_utilization)/4)
print("The average utilization of server 2 estimate: " , (s4.sv2_utilization+s3.sv2_utilization+s2.sv2_utilization+s1.sv2_utilization)/4)
print("Probability of Customer not waiting in the queue estimate: ", ((1-s4.totalqueuecust/s4.total_num_cus)+(1-s3.totalqueuecust/s3.total_num_
```

```
Results for seed 1
Average queue time is:  0.33851362413742364
Average number of customers in the system is:  2.0221434150264552
The average utilization of server 1:  0.8658125922318692
The average utilization of server 2:  0.8150994007145146
Probability of Customer not waiting in the queue:  0.5414710485133021
------------------------------------------------------------
Results for seed 2
Average queue time is:  0.22124746675880883
Average number of customers in the system is:  1.8704012758232902
The average utilization of server 1:  0.8474782321907768
The average utilization of server 2:  0.8008876565106975
Probability of Customer not waiting in the queue:  0.5792778649921507
------------------------------------------------------------
Results for seed 3
Average queue time is:  0.22437903539964196
Average number of customers in the system is:  1.877575649284369
The average utilization of server 1:  0.8624121504461071
The average utilization of server 2:  0.7889882338203199
Probability of Customer not waiting in the queue:  0.5896226415094339
------------------------------------------------------------
Results for seed 4
Average queue time is:  0.20911438969450544
Average number of customers in the system is:  1.8578742941756592
The average utilization of server 1:  0.8486944824762075
The average utilization of server 2:  0.7998015891735747
Probability of Customer not waiting in the queue:  0.5796875
------------------------------------------------------------
------------------------------------------------------------
------------------------------------------------------------
AVERAGES OF 4 DIFFERENT SEEDS AND ESTIMATE FOR THE SYSTEM
Queue time estimate:  0.24831362899759496
Average number of customers in the system estimate:  1.9069986585774434
The average utilization of server 1 estimate:  0.8560993643362401
The average utilization of server 2 estimate:  0.8011942200547767
Probability of Customer not waiting in the queue estimate:  0.5725147637537216
```

Little's Law)

- Average number of customers in the system at any arbitrary point in time = (average number of arrivals per time unit) x (average time spent in the system)

Since our simulation works the same for every seed, using only the first seed will be enough to prove that Little's Law holds. For our first seed:

2.022 = (638/7006) * (14168/638)

= 2.022

Little's Law holds for our results. These values are obtained from the program and can be obtained again using the same seed.

REFERENCES

- Lectures Slides 5-6-7
- Paul Grogan. (2016, October 29). *Queuing System Discrete Event Simulation in Python (Event-scheduling)*. YouTube. https://www.youtube.com/watch?v=oJyf8Q0KLRY
- *Python Object Oriented Programming*. (n.d.). Retrieved November 1, 2022, from https://www.programiz.com/python-programming/object-oriented-programming
- *Python Functions (def): Definition with Examples*. (n.d.). Retrieved November 1, 2022, from https://www.programiz.com/python-programming/function