Aral Açıkalın 20161701036

**Project 4 Report**

In this project I simulated a bank ATM using Discrete Event Simulation. As for inter arrival time and service time I used the same function I used in the last project.

def randomExp(lambd):

    nextArrival=random.expovariate(lambd/60)

    return nextArrival

This function seen above uses built in random library inside python. And to get the interarrival rates and service time in seconds I divided lambda and mean values that given by 60.

Activity of the simulation is determined by this function. Activities are interarrivals and departures.

    snapshot=[]

    queTime=0

    servedCustomerCount=0

    allSnapshots=[]

    customerLeft=0

    serviceTime=0

    cumulCustomersinQue=0

    cumulCustomersinSystem=0

then I initialized the variables to store the statistics as can be seen above. Snapshot variable is for the snapshots to be printed as output.

This simulation has only one set which is the atm queue. And this queue is ordered by first come first served.

#initializing the system state

    customersinQue=0

    isServing=False

    time=0

    customersinSystem=0

this code above initializes the system states.

    #customer entity list

    customers=[]

I only use customers as entities and initialize the list to hold the customer information and customer attributes.

#as first event creating a arrival event

    randArrival=randomExp(lambd)

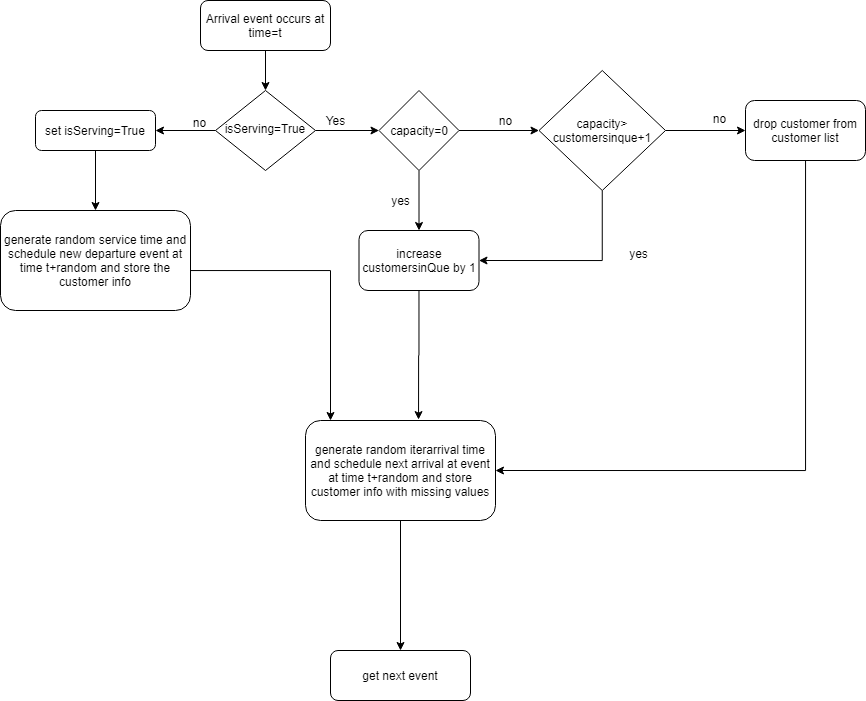
    customerNo=0

    #stores events as time, event flag and customer no

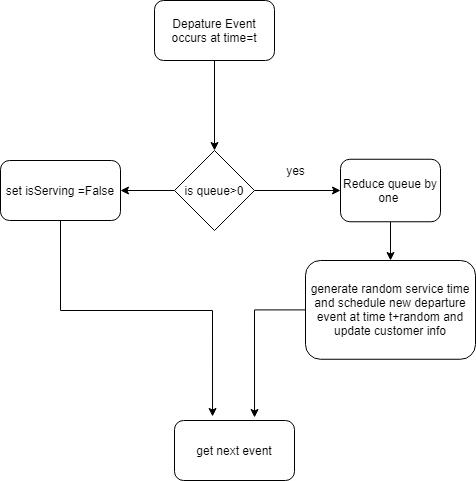
    events=[[time+randArrival,"Arrival",customerNo]]

After the initialization I create my first event and the event that I create is arrival event. Then store this event in the events list in this format seen above.

In this simulation I need only 2 events. Arrival event and Departure event. And I used almost the same flowcharts in the lecture slides.

Arrival Event Flowchart:

Departure Event Flowchart:



if(events[0][1]=="Arrival"):

#code to handle arrival event

elif(events[0][1]=="Departure"):

#code to handle departure event

I handle future event list with an if-else statement. I check if the next event is arrival or departure and handle the appropriate event.

if(not isServing):

    #setting ls(t) to 1

    isServing=True

    customerNo=events[0][2]

I handle arrival event like this;

First, I check if there is someone in service. If there is not, then I change this flag to true. With this I indicate now someone have entered service.

#creating a departure event

  randDeparture=randomExp(mean)

  events.append([time+randDeparture,"Departure",customerNo])

creating a departure event for recently arrived customer.

if(len(customers)-1==customerNo):

     customers.pop(customerNo)

#delete from customers list if the current customer entry is already created

#customer entity stored as customer no , arrival time, departure time, service time and que time

customers.append([customerNo,time,time+randDeparture,randDeparture,time+randDeparture-(randDeparture+time)])

if customer is already in the customers array delete that entry and create a new entry with full customer information. (including departure time, service time, queue time.)

#creating a arrival event

randArrival=randomExp(lambd)

events.append([time+randArrival,"Arrival",customerNo+1])

customers.append([customerNo+1,time+randArrival,None,None,None])

then create a new arrival event for the next customer and store this new customer info but with missing values like queue time. These will be filled when a departure event is created for this customer.

#pop the event that currently handled

    events.pop(0)

    customersinSystem=customersinQue+1

finally I delete the event currently handled then set the customers in system state to customers in queue +1 because there is currently a customer in the system. (customers in queue is actually 0 here)

else:

 #if capacity argument is 0 then there is no capacity in the simulation

     if(capacity!=0):

           #checks if arrival is over capacity

              if(capacity>customersinQue+1):

if ATM is already serving some one I check if I have a capacity (0 means no capacity) then if I have a capacity I check if I are over our capacity.

 customerNo=events[0][2]

 customersinQue+=1

 customersinSystem=customersinQue+1

  #creating a arrival event

randArrival=randomExp(lambd)

 events.append([time+randArrival,"Arrival",customerNo+1])

 customers.append([customerNo+1,time+randArrival,None,None,None])

 #pop the event that currently handled

 events.pop(0)

if I are not over capacity I increase customersinQue by one and create a new arrival event and store the new customer information (partially)

else:

 customerNo=events[0][2]

   customersinSystem=customersinQue+1

  customerLeft+=1

   customers.pop(customerNo)

#creating a arrival event

   randArrival=randomExp(lambd)

   events.append([time+randArrival,"Arrival",customerNo])

   customers.append([customerNo+1,time+randArrival,None,None,None])

   #pop the event that currently handled

   events.pop(0)

if we are over our capacity then we drop the customer on the arrival event. We delete the customer information from our customers array and create a new arrival event.

if(customers!=[]):

      if(customers[events[0][2]][3]!=None):

            serviceTime+=customers[events[0][2]][3]

this if statement in the departure event calculates the cumulative service time if customers have a departure time of the departing customer.

if(customersinQue>0):

     customerNo=events[0][2]

  customersinQue-=1

     customersinSystem=customersinQue+1

     #creating a departure event

     randDeparture=randomExp(mean)

     events.append([time+randDeparture,"Departure",customerNo+1])

     #setting departure time of the customer

     customers[customerNo+1][2]=time+randDeparture

   #setting service time of the customer

   customers[customerNo+1][3]=randDeparture

   #que time = departure time- service time-arrival time

 customers[customerNo+1][4]=time+randDeparture(randDeparture+customers[customerNo+1][1])

#pop the event that currently handled

  events.pop(0)

this checks if there are customers in queue or not. And if there are customers in queue creates a random arrival event for the next customer and stores the next customers information (departure time etc.). then delete this event recently handled.

else:

     customersinSystem=0

     #setting ls(t) to 0

     isServing=False

     events.pop(0)

if there are no customers in queue I set the customers in the system to 0 and set isServing to false and finally remove the event.

Here I don’t store any customer information because when we don’t have anyone in service, we store customer information and when there is no one in queue that means we already stored departing customers information in its arrival event.

        #sorting the events acording to their time

        events.sort()

and finally I sort the events for next event handling.

#collecting cumulative statistics

        cumulCustomersinSystem+=customersinSystem

        cumulCustomersinQue+=customersinQue

        #counting the snapshot count

        snapshotCount+=1

        howManySnapshots=6

        #for printing the first 6 snapshots

        if(len(snapshot)<howManySnapshots):

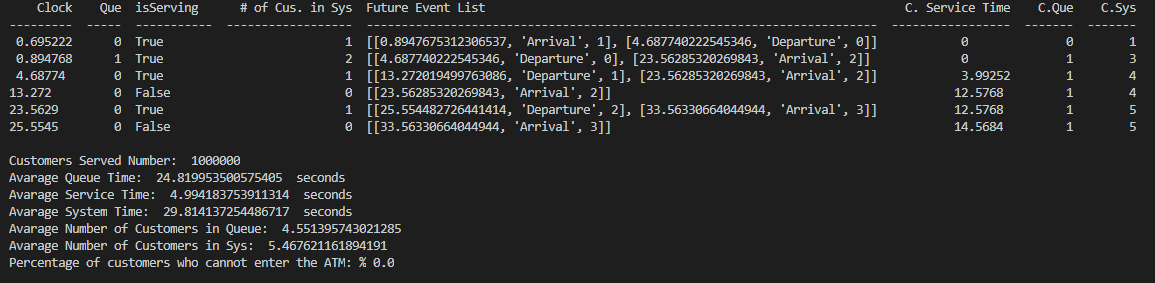
            tempEvents=events.copy()

            snapshot.append([time,customersinQue,isServing,customersinSystem,tempEvents,serviceTime,cumulCustomersinQue,cumulCustomersinSystem])

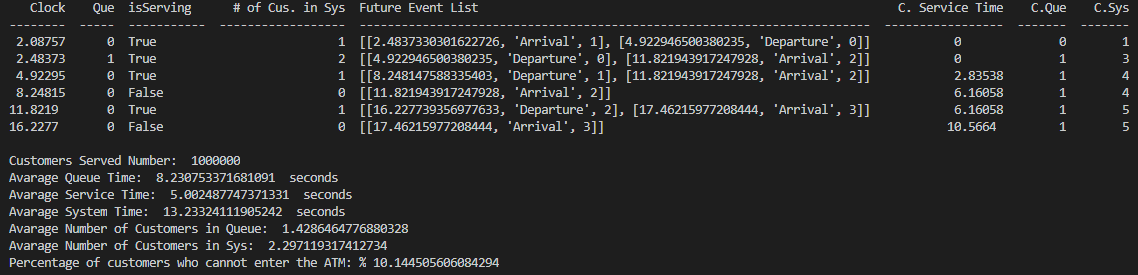
then I simply update cumulative statistics and put the first 6 snapshots in to a array for showing as output.

λ = 10, µ=12, N=infinity ,when inputs are these outputs are below;

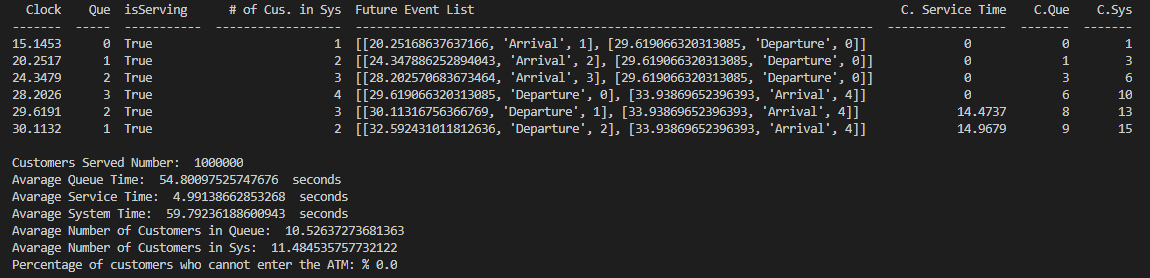
(I put screenshot because table wouldn’t fit to the page)



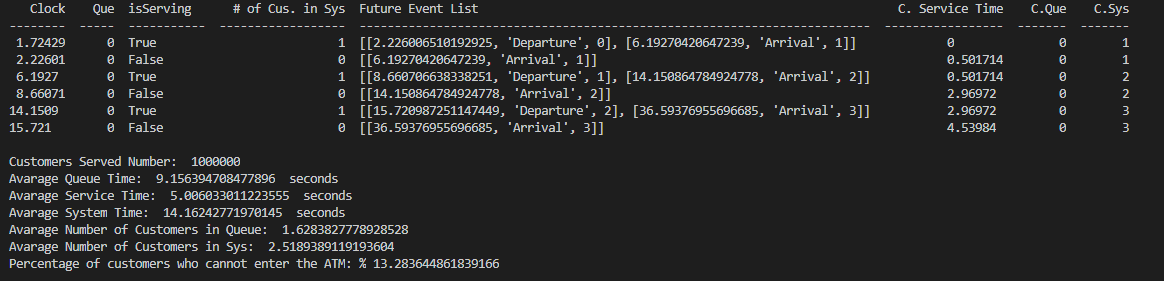
λ = 10, µ=12, N=5;



λ = 11, µ=12, N=infinity;



λ = 11, µ=12, N=5;



I believe my results are correct because previous project inputs were 5, 2 and 3 for lambda which makes 10 in total and the mean was 12. And I get the same result with my previous project result when inputs are “λ = 10, µ=12, N=infinity”. And average customers In queue also makes sense when capacity is 5 because there cant be more than 4 customers in queue and average may produce between 1 and 2. In system it is one more than queue there can be max 5 customers in system which means average can be between 2 and 3.