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.

    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.

#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.

#as first event creating a arrival event

    randArrival=randomExp(lambd)

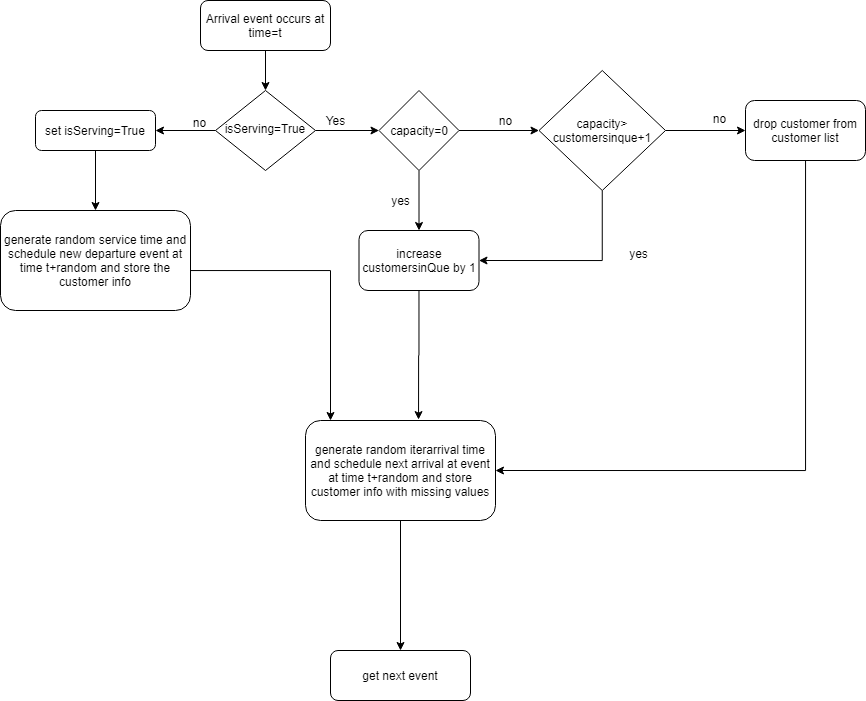
    customerNo=0

    #stores events as time, event flag and customer no

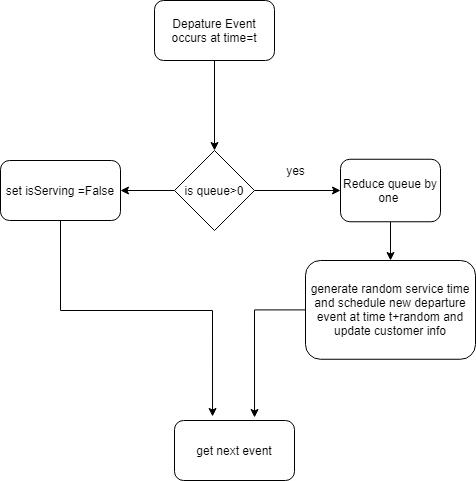
    events=[[time+randArrival,"Arrival",customerNo]]

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)

//arrival event code explanation

//departure event code explanation