**워 게임 모델링 중간고사 대체과제 car rental**

1. 사용 software: Python
2. Simulation code
3. import math  
   import random as rd  
     
   def interarrival(i):  
    lamb=[0,14/3600,10/3600,24/3600]  
    x=math.log(1-rd.random())  
    x=(-1)\*x/lamb[i]  
    return int(x)  
     
   def unloadtime():  
    return int(8\*rd.random()+16.5)  
     
   def loadtime():  
    return int(10\*rd.random()+15.5)  
     
   def destination(i):  
    if i == 3:  
    if rd.random() < 0.583:  
    return 1  
    else:  
    return 2  
    else:  
    return 3  
     
   Time=540  
   Bus=[]  
   Bus\_location=1  
   Busloc=[0,2,3,1]  
   transit\_time=[0,120,540,540]  
   loc\_1=[]  
   loc\_2=[]  
   loc\_3=[]  
   loc\_1\_time=0  
   loc\_2\_time=0  
   loc\_3\_time=0  
   location=[0,loc\_1,loc\_2,loc\_3]  
   location\_time=[0,loc\_1\_time,loc\_2\_time,loc\_3\_time]  
   avg1=[0,0,0]  
   max1=[0,0,0]  
   avg2=[0,0,0]  
   max2=[0,0,0]  
   avg3=0  
   max3=0  
   avg4=[0,0,0]  
   max4=[0,0,0]  
   min4=[9999,9999,9999]  
   avg5=[0,0,0]  
   max5=[0,0,0]  
   min5=[9999,9999,9999]  
   avg6=0  
   max6=0  
   min6=9999  
   loop=[-1,-1,-1,0]  
   num2=[0,0,0]  
   num3=0  
   num4=[0,0,0]  
   num5=[0,0,0]  
   num6=0  
   simulation\_time = 288000  
     
   while Time < simulation\_time:  
    departure\_time=Time+300  
    if len(Bus) != 0:  
    remove\_list=[]  
    for people in Bus:  
    if people[1] == Bus\_location:  
    remove\_list.append(people)  
    Time += unloadtime()  
    avg6 += Time - people[0]  
    num6 += 1  
    if max6 < Time - people[0]:  
    max6 = Time - people[0]  
    if min6 > Time - people[0]:  
    min6 = Time - people[0]  
    for remove in remove\_list:  
    Bus.remove(remove)  
     
    ans1 = 0  
    while True:  
    if len(location[Bus\_location]) == 0:  
    location\_time[Bus\_location] += interarrival(Bus\_location)  
    location[Bus\_location].append([location\_time[Bus\_location], destination(Bus\_location), 0])  
    ans1 += 1  
    if len(Bus) >= 20:  
    break  
    if location[Bus\_location][0][0] > Time:  
    if location[Bus\_location][0][0] <= departure\_time:  
    delay = Time - location[Bus\_location][0][0]  
    avg2[Bus\_location-1] += delay  
    num2[Bus\_location-1]+=1  
    if delay > max2[Bus\_location-1]:  
    max2[Bus\_location-1] = delay  
    Time += loadtime()  
    location[Bus\_location][0][2] = Time  
    Bus.append(location[Bus\_location][0])  
    location[Bus\_location].remove(location[Bus\_location][0])  
    else:  
    if Time < departure\_time:  
    Time = departure\_time  
    break  
    else:  
    delay = Time - location[Bus\_location][0][0]  
    avg2[Bus\_location-1] += delay  
    num2[Bus\_location-1] += 1  
    if delay > max2[Bus\_location - 1]:  
    max2[Bus\_location - 1] = delay  
    Time += loadtime()  
    location[Bus\_location][0][2] = Time  
    Bus.append(location[Bus\_location][0])  
    location[Bus\_location].remove(location[Bus\_location][0])  
     
    if max1[Bus\_location-1] < ans1:  
    max1[Bus\_location-1] = ans1  
     
    avg3 += len(Bus)  
    num3 += 1  
    if len(Bus)>max3:  
    max3 = len(Bus)  
     
    avg4[Bus\_location-1] += Time-departure\_time+300  
    num4[Bus\_location-1] += 1  
    if min4[Bus\_location-1] > Time-departure\_time+300:  
    min4[Bus\_location-1] = Time - departure\_time + 300  
    if Time-departure\_time > max4[Bus\_location-1]:  
    max4[Bus\_location-1] = Time-departure\_time  
     
    if loop[Bus\_location] != -1:  
    avg5[Bus\_location-1] += Time - loop[Bus\_location]  
    num5[Bus\_location-1] += 1  
    if max5[Bus\_location-1] < Time - loop[Bus\_location]:  
    max5[Bus\_location - 1] = Time - loop[Bus\_location]  
    if min5[Bus\_location-1] > Time - loop[Bus\_location]:  
    min5[Bus\_location - 1] = Time - loop[Bus\_location]  
    loop[Bus\_location] = Time  
     
    Time += transit\_time[Bus\_location]  
    Bus\_location = Busloc[Bus\_location]  
     
     
     
   avg1[0] = avg2[0]/simulation\_time  
   avg1[1] = avg2[1]/simulation\_time  
   avg1[2] = avg2[2]/simulation\_time  
     
   avg2[0] = avg2[0]/num2[0]  
   avg2[1] = avg2[1]/num2[1]  
   avg2[2] = avg2[2]/num2[2]  
     
   avg3 = avg3 / num3  
     
   avg4[0] = avg4[0]/num4[0]  
   avg4[1] = avg4[1]/num4[1]  
   avg4[2] = avg4[2]/num4[2]  
     
   avg5[0] = avg5[0]/num5[0]  
   avg5[1] = avg5[1]/num5[1]  
   avg5[2] = avg5[2]/num5[2]  
     
   avg6 = avg6 / num6  
     
   print(avg1)  
   print(max1)  
   print(avg2)  
   print(max2)  
   print(avg3)  
   print(max3)  
   print(avg4)  
   print(max4)  
   print(min4)  
   print(avg5)  
   print(max5)  
   print(min5)  
   print(avg6)  
   print(max6)  
   print(min6)
4. Simulation code 설명

def interarrival(i):  
 lamb=[0,14/3600,10/3600,24/3600]  
 x=math.log(1-rd.random())  
 x=(-1)\*x/lamb[i]  
 return int(x)  
  
def unloadtime():  
 return int(8\*rd.random()+16.5)  
  
def loadtime():  
 return int(10\*rd.random()+15.5)  
  
def destination(i):  
 if i == 3:  
 if rd.random() < 0.583:  
 return 1  
 else:  
 return 2  
 else:  
 return 3

위와 같이 문제에서 사용 문제에서 되는 난수 발생함수를 정의하였다. Interarrival time, unloading time, loading time, 그리고 car rental에서의 destination을 결정하는 총 4개의 남수 발생 함수를 정의하였다. Interarrival time의 경우 exponential distribution function의 cdf의 inv function을 사용하였다. int함수를 사용함에 따라 unload와 load함수가 int를 사용하지 않을 때와 mean이 같을 수 있게 0.5를 더하였다.

Time=540  
Bus=[]  
Bus\_location=1  
Busloc=[0,2,3,1]  
transit\_time=[0,120,540,540]  
loc\_1=[]  
loc\_2=[]  
loc\_3=[]  
loc\_1\_time=0  
loc\_2\_time=0  
loc\_3\_time=0  
location=[0,loc\_1,loc\_2,loc\_3]  
location\_time=[0,loc\_1\_time,loc\_2\_time,loc\_3\_time]

simulation\_time = 288000

Simulation에 사용되는 변수를 지정하였다. 시간은 second를 단위로 하여 총 simulation time은 80\*60\*60=288000이다. 이동시간 역시 transit time list에 second로 저장하였다. Bus는 Bus에 탑승한 people의 list이며 location은 각 location의 queue의 list이다. Bus\_location은 현재 Bus의 location이고 Time은 현재 시간이며 Bus가 car\_rental을 즉시 출발하므로 총 이동시간인 540이 지난 Time=540과 Bus\_location=1로 simulation을 시작한다.

while Time < simulation\_time:

Code는 simulation time을 초과하기까지 반복하며 while문 내부는 unload, load 두 개의 code로 구분되어 있다.

1. Unload

if len(Bus) != 0:  
 remove\_list=[]  
 for people in Bus:  
 if people[1] == Bus\_location:  
 remove\_list.append(people)  
 Time += unloadtime()  
 avg6 += Time - people[0]  
 num6 += 1  
 if max6 < Time - people[0]:  
 max6 = Time - people[0]  
 if min6 > Time - people[0]:  
 min6 = Time - people[0]  
 for remove in remove\_list:  
 Bus.remove(remove)

people은 [location 도착시간, destination(각 location의 i=1,2,3), Bus 탑승시간]의 3개의 값을 가지는 list이며(load part에서 설명) people[1]= destination이 Bus\_location과 일치하는 people을 remove list에 추가하고 이후에 Bus에서 제거한다. Time은 unloadtime()만큼 추가하며 location도착시간과 현재 버스에서 unload하는 시간인 Time 만큼의 차이인 Time-people[0]을 총 system 이용시간으로 계산하여 문제 (6)의 값인 avg6, max6, min6 을 계산한다.

1. Load

ans1 = 0  
while True:  
 if len(location[Bus\_location]) == 0:  
 location\_time[Bus\_location] += interarrival(Bus\_location)  
 location[Bus\_location].append([location\_time[Bus\_location], destination(Bus\_location), 0])  
 ans1 += 1  
 if len(Bus) >= 20:  
 break  
 if location[Bus\_location][0][0] > Time:  
 if location[Bus\_location][0][0] <= departure\_time:  
 delay = Time - location[Bus\_location][0][0]  
 avg2[Bus\_location-1] += delay  
 num2[Bus\_location-1]+=1  
 if delay > max2[Bus\_location-1]:  
 max2[Bus\_location-1] = delay  
 Time += loadtime()  
 location[Bus\_location][0][2] = Time  
 Bus.append(location[Bus\_location][0])  
 location[Bus\_location].remove(location[Bus\_location][0])  
 else:  
 if Time < departure\_time:  
 Time = departure\_time  
 break  
 else:  
 delay = Time - location[Bus\_location][0][0]  
 avg2[Bus\_location-1] += delay  
 num2[Bus\_location-1] += 1  
 if delay > max2[Bus\_location - 1]:  
 max2[Bus\_location - 1] = delay  
 Time += loadtime()  
 location[Bus\_location][0][2] = Time  
 Bus.append(location[Bus\_location][0])  
 location[Bus\_location].remove(location[Bus\_location][0])

이 simulation은 고정시간 증가법을 사용하지 않는다. 대신 각 Bus가 location에서 people을 load할 때, location의 queue에서 people을 arrivaltime() 함수를 이용해 생성한다. 이전에 정의했던 location\_time list에 가장 최근에 queue에 생성된 people의 Time을 기록하므로 현재 time까지 people을 계속 생성하며 Bus에 append 시킨다. 정답에 필요한 시간을 계산하는데 전혀 문제가 없는 것은 people의 생성 시간인 location에 도착한 시간과 현재 Bus에 load하는 시간의 차이로 계산을 하므로 people을 이전에 생성하여 queue에 대기시키는 것과 계산은 동일하다.

위의 code에서 Bus정원이 capacity인 20을 초과하거나 다음 생성된 people의 location 도착시간이 현재 Time 이후이며 현재 Time이 departure\_time(Bus가 location에 도착한지 5min(=300sec)가 지난 때)를 지났을 때 break를 통하여 Load 과정을 종료하며 people의 location 도착시간이 현재 Time 이후지만 departure\_time 이전일 때는 Bus에 append 한다.

ans1은 load과정에서 queue에서 people이 생성되는 총 횟수로 queue에서 people을 미리 생성하지 않으므로 queue에 존재하는 총 people의 수로 근사하기 위해 사용한다. 이 값으로 문제(1)의 maximum number in each queue를 계산한다.

delay는 location도착 시간부터 load까지의 시간, 즉 queue에서의 총 대기시간이며 이 값으로 (2)를 구한다.

Time += transit\_time[Bus\_location]  
Bus\_location = Busloc[Bus\_location]

unload와 load가 끝난 이후 Bus의 location간의 이동시간을 추가하고 Bus location을 변경한다.

1. statistics
2. if max1[Bus\_location-1] < ans1:  
    max1[Bus\_location-1] = ans1  
     
    avg3 += len(Bus)  
    num3 += 1  
    if len(Bus)>max3:  
    max3 = len(Bus)  
     
    avg4[Bus\_location-1] += Time-departure\_time+300  
    num4[Bus\_location-1] += 1  
    if min4[Bus\_location-1] > Time-departure\_time+300:  
    min4[Bus\_location-1] = Time - departure\_time + 300  
    if Time-departure\_time > max4[Bus\_location-1]:  
    max4[Bus\_location-1] = Time-departure\_time  
     
    if loop[Bus\_location] != -1:  
    avg5[Bus\_location-1] += Time - loop[Bus\_location]  
    num5[Bus\_location-1] += 1  
    if max5[Bus\_location-1] < Time - loop[Bus\_location]:  
    max5[Bus\_location - 1] = Time - loop[Bus\_location]  
    if min5[Bus\_location-1] > Time - loop[Bus\_location]:  
    min5[Bus\_location - 1] = Time - loop[Bus\_location]  
    loop[Bus\_location] = Time  
     
   avg1[0] = avg2[0]/simulation\_time  
   avg1[1] = avg2[1]/simulation\_time  
   avg1[2] = avg2[2]/simulation\_time  
     
   avg2[0] = avg2[0]/num2[0]  
   avg2[1] = avg2[1]/num2[1]  
   avg2[2] = avg2[2]/num2[2]  
     
   avg3 = avg3 / num3  
     
   avg4[0] = avg4[0]/num4[0]  
   avg4[1] = avg4[1]/num4[1]  
   avg4[2] = avg4[2]/num4[2]  
     
   avg5[0] = avg5[0]/num5[0]  
   avg5[1] = avg5[1]/num5[1]  
   avg5[2] = avg5[2]/num5[2]  
     
   avg6 = avg6 / num6

위와 같은 과정으로 (1) ~ (6)을 구하며 avg는 avg에 총 값을 더하고 더할 때 마다 num에 추가하여 최종적으로 avg/num으로 구한다.

print(avg1)  
print(max1)  
print(avg2)  
print(max2)  
print(avg3)  
print(max3)  
print(avg4)  
print(max4)  
print(min4)  
print(avg5)  
print(max5)  
print(min5)  
print(avg6)  
print(max6)  
print(min6)

simulation의 결과값은 아래와 같다.

[7.854763888888889, 5.219305555555556, 8.15703125]  
[16, 14, 20]  
[2045.363471971067, 1813.220747889023, 1274.6744438415626]  
[7031, 6581, 3459]  
17.018072289156628  
20  
[403.14414414414415, 314.26126126126127, 684.9090909090909]  
[305, 202, 522]  
[217, 116, 312]  
[2603.3818181818183, 2603.3818181818183, 2602.8]  
[2986, 2846, 2843]  
[2131, 2041, 2112]  
2688.4211645838873  
8267  
718

1. Number in each queue

|  |  |  |  |
| --- | --- | --- | --- |
|  | Location 1 | Location 2 | Location 3 |
| Avg | 7.85 | 5.22 | 8.16 |
| Max | 16 | 14 | 20 |

1. Delay in each queue

|  |  |  |  |
| --- | --- | --- | --- |
|  | Location 1 | Location 2 | Location 3 |
| Avg | 2045 | 1813 | 1275 |
| Max | 7031 | 6581 | 3459 |

1. Number on the bus

Avg = 17.02

Max = 20

1. Time the bus is stopped

|  |  |  |  |
| --- | --- | --- | --- |
|  | Location 1 | Location 2 | Location 3 |
| Avg | 403 | 314 | 685 |
| Max | 305 | 202 | 522 |
| Min | 217 | 116 | 312 |

1. Time for the bus to make a loop

|  |  |  |  |
| --- | --- | --- | --- |
|  | Location 1 | Location 2 | Location 3 |
| Avg | 2603 | 2603 | 2603 |
| Max | 2986 | 2846 | 2843 |
| Min | 2131 | 2041 | 2112 |

1. Time a person is in the system by arrival location

Avg = 2688

Max = 8267

Min = 718