Simulasi Event Diskrit (Studi Kasus : Simulasi Sistem Transportasi Bus)

Adi Purnama
13514006@std.stei.itb.ac.id

Hafizh Afkar Makmur
13514062@std.stei.itb.ac.id

1-03-2018

1 Deskripsi Masalah

Sebuah bus memiliki tiga titik pemberhentian, yaitu Air Terminal 1 , Air Terminal 2 , dan Car Rental 3. Bus memulai rute dari pemberhentian 3 , lalu ke 2, ke 1 , kemudian kembali ke 3 , dan begitu seterusnya (loop). Bus memiliki kapasitas 20 penumpang. Kecepatan bus adalah 30 mil per jam, dengan jarak antar titik pemberhentian Car Rental ke Air Terminal 1 sejauh 4.5 Mil, Air Terminal 1 ke Air Terminal 2 sejauh 1 mil , dan Air Terminal 2 ke Car Rental 3 sejauh 3.5 Mil. Calon penumpang data ke setiap titik pemberhentian dengan distribusi independent exponential interarrival time. Rata rata kedatangan calon penumpang per jam pada masing masing titik pemberhentian untuk Air Terminal 1 adalah 14 orang per jam, Air Terminal 2 adalah 10 orang per jam, dan Car Rental 3 adalah 24 orang per jam.

Penumpang yang berasal dari Air Terminal memiliki tujuan ke Car Rental. Sementara itu, penumpang yang berasal dari Car Rental memiliki tujuan ke Air Terminal 1 atau 2 dengan probabilitas tujuan air terminal 1 adalah 0.583 dan probabilitas tujuan air terminal 2 adalah 0.417. Aturan antrian (baik itu antrian di titik pemberhentian, maupun antrian saat ingin keluar dari bus) adalah First In First Out. Bus akan menunggu di masing-masing titik pemberhentian selama minimal 5 menit. Setelah itu , bus akan berangkat. Waktu yang diperlukan penumpang untuk memasuki bus dan keluar bus bervariasi setiap orangnya. Waktu untuk turun dari bus memiliki distribusi uniform antara 16 sampai 24 detik. Sementara itu, waktu untuk masuk ke bus memiliki distribusi uniform antara 15 sampai 25 detik.

Jalankan simulasi sampai 80 jam. Lalu berikan laporan statistik sebagai berikut :

1. Panjang Antrian di tiap titik pemberhentian (rata rata & maksimum)

2. Waktu tunggu yang dirasakan penumpang saat menunggu bis datang di titik pemberhentian (rata rata & maksimum)
3. Jumlah penumpang yang ada di dalam bis (rata rata & maksimum)
4. Waktu tunggu bis di titik pemberhentian (rata rata, maksimum, minimum)
5. Waktu bis melakukan 1 loop. 1 Loop didefinisikan sejak keberangkatan dari Car Rental (rata rata , maksimum, minimum)
6. Waktu total perjalanan yang dirasakan penumpang. Waktu total dimulai sejak penumpang datang ke titik pemberhentian sampai keluar dari bis. (rata rata & maksimum)

2 Rancangan Solusi

Untuk menyelesaikan permasalahan ini, dilakukan dua tahap pengerjaan. Langkah pertama adalah mensimulasikan proses sesuai dengan deskripsi di atas. Setelah proses tersebut dapat disimulasikan menggunakan program komputer, dilakukan langkah selanjutnya yaitu perhitungan data statistik. Berikut adalah komponen utama dari program simulasi ini.

2.1 Kelas Passanger

Kelas untuk menyimpan informasi yang dimiliki oleh seorang penumpang, seperti waktu kedatangan , waktu keberangkatan, kota asal dan kota tujuan. Kelas passanger memiliki dua buah operasi untuk keluar dan masuk bis yaitu enteringTheBus() dan leavingTheBus()

```
class Passanger:
def __init__(self,id,originStation,
    arrivedTimeAtOriginStation):
 self.id = id
 self.originStation = originStation
 self.targetStation = 0
 self.arrivedTimeAtOriginStation =
     arrivedTimeAtOriginStation
 self.arrivedTimeAtTargetStation = 0
 self.timeEnteringTheBus = 0
 self.timeLeavingTheBus = 0
 self.delayTimeInQueue = 0
def enteringTheBus(self, timeEnteringTheBus):
 self.timeEnteringTheBus = timeEnteringTheBus
 self.delayTimeInQueue = self.timeEnteringTheBus - self.
     arrivedTimeAtOriginStation
def leavingTheBus(self, timeLeavingTheBus):
 self.timeLeavingTheBus = timeLeavingTheBus
```

2.2 Kelas StationQueue

Kelas yang merepresentasikan sebuah stasiun pemberhentian bis. Komponen yang menyusun kelas ini antara lain :

2.2.1 futurePassangerQueue

List yang berisi seluruh calon penumpang yang akan datang ke stasiun pemberhentian ini sejak awal sampai akhir waktu simulasi. List ini di-generate saat awal simulasi (method initializeAllFutureEvents()), menggunakan waktu antar kedatangan dengan distribusi eksponensial.

2.2.2 simulationTime

Menyimpan waktu simulasi saat ini

2.2.3 currentQueue

List yang berisi seluruh calon penumpang yang sedang menunggu antrian pada waktu simulationTime. List ini diupdate menggunakan method updateStationQueue dengan cara memindahkan penumpang dari futurePassangerQueue ke currentQueue berdasarkan waktu simulasi saat ini. Method updateStationQueue dipanggil setiap kali waktu simulasi bergerak maju.

```
class StationQueue:
 def __init__(self,passangerPerHour,originStation):
  self.futurePassangerQueue = []
  self.currentQueue = []
  self.simulationTime = 0
  self.isFirstStart = True
  self.maxNumberQueue = 0
  self.qtArea_nPeopleOnQueue = 0
  self.initializeAllFutureEvents(passangerPerHour,
      originStation)

 def initializeAllFutureEvents(self,passangerPerHour,
     originStation):
  global_time = 0.0
  simulation_end = 81
  simulation_end = simulation_end * 3600
  n_passanger = 0
  stationList = []
  while global_time < simulation_end:
   passanger_arrival = random.expovariate(
       passangerPerHour) * 3600
   global_time += passanger_arrival
   if (global_time <= simulation_end):
```

```
    n_passanger += 1
    a = Passanger(n_passanger, originStation, global_time)
    self.futurePassangerQueue.append(a)

def updateStationQueue(self,time):
 if (self.isFirstStart):
  self.simulationTime = 0
  self.isFirstStart = False
 else:
  while (self.futurePassangerQueue[0].
      arrivedTimeAtOriginStation <= time):
   passanger = self.futurePassangerQueue.pop(0)
   self.currentQueue.append(passanger)
  self.qtArea_nPeopleOnQueue += len(self.currentQueue) *
      (time - self.simulationTime)
  self.simulationTime = time
  if (self.maxNumberQueue < len(self.currentQueue)):
   self.maxNumberQueue = len(self.currentQueue)

def peekFuturePassanger(self):
 return self.futurePassangerQueue[0]
```

2.3 Kelas Bus

Kelas untuk merepresentasikan sebuah Bus. Program utama dari simulasi ini adalah sebagai berikut

```
bus = Bus()
while True:
 bus.unloadPassangers()
 bus.loadPassangers()
 bus.movePlace()
```

2.3.1 unloadPassangers()

Method untuk menurunkan seluruh penumpang yang ingin turun dari Bus.

```
def unloadPassangers(self):
 if (self.nOfPassangerWantToUnloadHere > 0):
  for x in range(self.nOfPassangerWantToUnloadHere):
   passangerWantToUnload = self.passangersInsideBus.pop
       (0)
   self.unloadPassanger(passangerWantToUnload)
```

2.3.2 unloadPassanger()

Method untuk menurunkan seorang penumpang yang ingin turun dari Bus. Method unloadPassangerDuration() berfungsi untuk men-generate uniform random number antara 16 sampai 24 detik.

```
def unloadPassanger(self,passanger):
  unloadTime = self.unloadPassangerDuration()
  passanger.timeArrivedAtStation = self.simulationTime
  self.simulationTime += unloadTime
  passanger.timeLeavingTheBus = self.simulationTime
  passanger.targetStation = self.currentPosition
  self.arrivedPassangers.append(passanger)
  if (passanger.originStation == 3):
   self.nOfPassangerFromCarRental -= 1
  else:
   self.nOfPassangerFromTerminal -= 1
  self.qtArea_nPassangerInsideBus = self.
      qtArea_nPassangerInsideBus + (self.simulationTime -
      passanger.timeEnteringTheBus)
  self.timeoutCheck()
```

2.3.3 loadPassangers()

Method untuk menunggu dan memasukan penumpang yang ingin naik ke Bus. Algoritma yang digunakan adalah sebagai berikut

1. Naikkan seluruh penumpang yang saat ini sudah menunggu di stasiun
2. Ramalkan kedatangan penumpang di masa depan (panggil method peekFuturePassanger() untuk mendapatkan penumpang teratas di queue futurePassangerQueue). Jika penumpang tersebut akan datang sebelum batas waktu tunggu berakhir. Tunggu hingga penumpang itu datang. Jika tidak, lanjutkan perjalanan ke stasiun berikutnya.

```
def loadPassangers(self):
  currentSessionbusWaitingTime = self.
      getMaxbusWaitingTime()
  currentStationQueue = self.StationQueue[self.
      currentPosition]
  isWaitingPhase = True
  while (self.simulationTime <=
      currentSessionbusWaitingTime) and (not self.isFull()
      ) and (isWaitingPhase):

   # Load All Passanger Who Already Waiting in Station
       Queue as Long Bus is Not Full
```

```
while (len(currentStationQueue.currentQueue) > 0) and
    (not self.isFull()):
 passanger = currentStationQueue.currentQueue.pop(0)
 self.loadPassanger(passanger)
 # Additional Waiting Time Due People Loading
 if (self.simulationTime >
     currentSessionbusWaitingTime):
  currentSessionbusWaitingTime = self.simulationTime

# Peek Future Passanger. If There is Future Passanger
    Who Will Come Faster Before WaitTime Ends. Wait. If
     No. Wait Until WaitTime Ends. Then Go..
if (currentStationQueue.peekFuturePassanger().
    arrivedTimeAtOriginStation <=
    currentSessionbusWaitingTime):
 self.setsimulationTime(currentStationQueue.
     peekFuturePassanger().arrivedTimeAtOriginStation)
else:
 isWaitingPhase = False

 # Force Wait Until 5 Minutes (Even Is Predicted There
     Is No Passanger Will Come Anyway)
 if (self.simulationTime <
     currentSessionbusWaitingTime):
  self.setsimulationTime(currentSessionbusWaitingTime)

 # Collect The Statistics Data (Waiting Time Duration)
 busWaitingTime = self.simulationTime - self.
     timeArrivedAtStation
 self.addbusWaitingTimeStatistics(busWaitingTime)

 # Collect The Statistics Data (Loop Time)
 if ((self.currentPosition == 3) and (self.
     isFreshStart)):
  self.carRentalDepartureTime = self.simulationTime
 if ((self.currentPosition == 3) and (not self.
     isFreshStart)):
  loopTime = self.simulationTime - self.
      carRentalDepartureTime
  self.carRentalDepartureTime = self.simulationTime
  self.addLoopTimeStatistics(loopTime)
```

2.3.4 loadPassanger()

Method untuk menaikan satu penumpang dari stasiun ke dalam bis.

```
def loadPassanger(self,passanger):
 loadTime = self.loadPassangerDuration()
 passanger.enteringTheBus(self.simulationTime + loadTime)
 self.passangersInsideBus.append(passanger)
 self.simulationTime = passanger.timeEnteringTheBus
 if (passanger.originStation == 3):
  self.nOfPassangerFromCarRental += 1
 else:
  self.nOfPassangerFromTerminal +=1
 self.timeoutCheck()
```

2.3.5 movePlace()

Method yang berfungsi untuk memindahkan bus dari satu stasiun ke stasiun selanjutnya.

```
def movePlace(self):
  self.isFreshStart = False
  # [STATS] Check if Current Number of People in The Bus
      Breaks Record
  if (self.maxNPassangers < len(self.passangersInsideBus)
      ):
   self.maxNPassangers = len(self.passangersInsideBus)
  if (self.currentPosition == 3):
   self.currentPosition = 1
   self.simulationTime = self.simulationTime + (4.5 /
       self.busSpeed * 3600)
   self.determinePassangerTargetStation()
   self.nOfPassangerWantToUnloadHere = self.
       nOfPassangerToTerminal1
   self.nOfPassangerFromTerminal = 0
  elif (self.currentPosition == 1):
   self.currentPosition = 2
   self.simulationTime = self.simulationTime + (1 / self.
       busSpeed * 3600)
   self.nOfPassangerWantToUnloadHere = self.
       nOfPassangerToTerminal2
   self.nOfPassangerToTerminal1 = 0
  else:
   self.currentPosition = 3
   self.simulationTime = self.simulationTime + (4.5 /
       self.busSpeed * 3600)
```

```
    self.nOfPassangerWantToUnloadHere = self.
        nOfPassangerFromTerminal
    self.nOfPassangerToTerminal2 = 0
  self.timeArrivedAtStation = self.simulationTime
  self.timeoutCheck()
```

2.3.6 determinePassangerTargetStation()

Saat melakukan perpindahan, bus juga mencatat tujuan perjalanan dari masing-masing penumpang. Seluruh penumpang yang berasal dari Air Terminal memiliki tujuan ke Car Rental. Sementara itu, penumpang dari Car Rental dapat memiliki tujuan ke Air Terminal 1 atau Air Terminal 2 berdasarkan probabilitas tertentu. Probabilitas ini dihitung menggunakan method determinePassanger-TargetStation()

```
def determinePassangerTargetStation(self):
  for x in range(self.nOfPassangerFromCarRental):
    if (random.uniform(0,1) <= 0.583):
      self.nOfPassangerToTerminal1 += 1
    else:
      self.nOfPassangerToTerminal2 += 1
```

2.3.7 timeoutCheck()

Method timeoutCheck() dipanggil setiap kali waktu simulasi bergerak maju. Method ini berfungsi untuk melakukan refresh terhadap antrian pada tiap stasiun dan mengakhiri simulasi apabila waktu simulasi sudah melebihi batas (80 Jam)

```
def timeoutCheck(self):
  self.refreshQueue()
  if self.simulationTime >= (80 *3600):
    self.printStatistics()
    sys.exit()
```

2.4 Statistik

Secara umum, perhitungan statistik di dalam kasus ini dapat digolongkan menjadi dua jenis

2.4.1 Rata-rata, maksimum, minimum

Data tersebut dikumpulkan dalam sebuah struktur data list, lalu dihitung maksimum, minimum , dan rata-ratanya. Sebagai contoh , waktu tunggu yang dirasakan penumpang saat menunggu bis datang di titik pemberhentian (rata rata & maksimum). Langkah menghitung statistik ini adalah sebagai berikut :

1. Setiap penumpang memiliki informasi "Waktu menunggu di stasiun". Kumpulkan informasi ini dari seluruh penumpang, lalu masukkan ke dalam sebuah struktur data list.
2. Lakukan operasi avg() dan max() terhadap struktur data list tersebut.

Sebagian besar perhitungan statistik yang dapat diselesaikan dengan cara ini, antara lain :

1. Panjang maksimum antrian di tiap titik pemberhentian
2. Waktu tunggu yang dirasakan penumpang saat menunggu bis datang di titik pemberhentian (rata rata & maksimum)
3. Jumlah maksimum penumpang yang ada di dalam bis
4. Waktu tunggu bis di titik pemberhentian (rata rata, maksimum, minimum)
5. Waktu bis melakukan 1 loop. 1 Loop didefinisikan sejak keberangkatan dari Car Rental (rata rata , maksimum, minimum)
6. Waktu total perjalanan yang dirasakan penumpang. Waktu total dimulai sejak penumpang datang ke titik pemberhentian sampai keluar dari bis. (rata rata & maksimum)

Semua perhitungan statistik di atas mengikuti pola yang sama, yaitu kumpulkan seluruh informasi ke dalam sebuah list , lalu operasikan list tersebut dengan fungsi avg(), min(), atau max(). Berikut adalah potongan source code untuk menghitung statistik.

```
def printLoopTimeStatistics(self):
  l = self.loopTimeStatistics
  avgLoopTime = avg(l)
  maxLoopTime = max(l)
  minLoopTime = min(l)
  print "Loop Time"
  print " Max: " ,
  timeFormatter(maxLoopTime)
```

```
print " Avg: ",
timeFormatter(avgLoopTime)
print " Min: ",
timeFormatter(minLoopTime)
```

2.4.2 Rata Rata Panjang Antrian

Statistik mengenai rata-rata panjang antrian dihitung menggunakan langkah sebagai berikut.

1. Untuk setiap panjang antrian, hitung durasi terjadinya
2. Jumlahkan hasil kali panjang antrian dengan durasi terjadinya
3. Bagi hasil penjumlahan tersebut dengan waktu akhir simulasi

Sebagai contoh, perhatikan tabel di bawah ini :

Panjang Antrian	Durasi Waktu Terjadi (s)
1	2.3
2	7.1
3	2.3

Rata-rata panjang antrian dapat dihitung sebagai berikut

$$rataRataPanjangAntrian = \frac{(1x2.3) + (2x7.1) + (3x2.3)}{2.3 + 7.1 + 2.3}$$

Pada kasus ini, terdapat dua perhitungan statistik yang melibatkan rata-rata panjang antrian, yaitu :

1. Rata-rata panjang antrian di setiap stasiun pemberhentian
2. Rata-rata jumlah penumpang yang menaiki bus

Untuk rata-rata panjang antrian di setiap stasiun pemberhentian, jumlahkan seluruh durasi waktu menunggu yang dialami oleh setiap penumpang pada suatu stasiun, lalu bagi dengan waktu akhir simulasi. Total seluruh waktu menunggu yang dialami oleh penumpang pada suatu stasiun disimpan pada atribut qtAreanPeopleOnQueue di kelas StationQueue. Setiap kali waktu simulasi dimajukan sebesar a detik, qtAreanPeopleOnQueue akan ditambah dengan (a x jumlahOrangdiAntrianSaatIni)

```
def updateStationQueue(self,time):
    self.qtArea_nPeopleOnQueue += len(self.currentQueue) *
        (time - self.simulationTime)
```

Lalu, bagi nilai itu dengan waktu akhir simulasi untuk mendapatkan rata-rata panjang antrian.

```
def printNumberPersonQueueStatistics(self):
 print "Average Number Queue"
 print " Station1 :",
 print self.StationQueue[1].qtArea_nPeopleOnQueue / self.
    simulationTime
 print " Station2 :",
 print self.StationQueue[2].qtArea_nPeopleOnQueue / self.
    simulationTime
 print " Station3 :",
 print self.StationQueue[3].qtArea_nPeopleOnQueue / self.
    simulationTime
```

Untuk rata-rata jumlah penumpang yang menaiki bus, jumlahkan seluruh durasi waktu selama di dalam bis yang dialami oleh setiap penumpang, lalu bagi dengan waktu akhir simulasi.

3 Hasil Simulasi

3.1 Statistik Bus

Statistik	Max	Avg	Min
Bus Waiting Time	0:13:43	0:09:17	0:05:00
Bus Loop Time	0:48:17	0:45:15	0:38:12
Number of People On The Bus	20	13.9710474206	-

Hal menarik yang dapat disimpulkan dari hasil statistik ini adalah sebagai berikut :

1. Rata-rata, bus menunggu di stasiun selama sekitar 9 menit
2. Rata-rata, waktu yang diperlukan bus untuk melakukan satu putaran adalah 45 menit
3. Rata-rata, bus selalu terisi 14 orang dari total kapasitas 20 (Utilisasi 69%)

3.2 Statistik Stasiun

Statistik	Air Terminal 1	Air Terminal 2	Car Rental
Average Number Queue	6.18643751187	7.70652576188	9.31329665976
Avg Delay Time	0:25:04	0:45:50	0:23:13
Max Delay Time	1:16:40	3:40:37	1:08:15
Max Number Queue	22	39	28

Hal menarik yang dapat disimpulkan dari hasil statistik ini adalah sebagai berikut :

1. Rata-rata , stasiun Car Rental cenderung lebih ramai dibanding stasiun yang lain dengan panjang antrian rata-rata 9 orang.

2. Dapat disimpulkan bahwa kualitas layanan terburuk ada di Stasiun Air Terminal 2. Penumpang akan merasakan waktu menunggu lebih lama jika dia menunggu di Stasiun Air Terminal 2 (45 Menit). Pada kasus terburuk, penumpang dapat menunggu hingga 3 jam 40 menit untuk menunggu bus di Stasiun Air Terminal 2. Hal ini dapat disebabkan karena bus yang datang sudah penuh, sehingga harus menunggu bus untuk melakukan satu putaran lagi. Mengingat waktu rata-rata bus untuk melakukan 1 putaran adalah 45 menit, dalam kasus ini, penumpang harus menunggu bis hingga 4 putaran karena bus tersebut selalu penuh. Pada kasus terburuk, panjang antrian dapat mengular hingga 39 orang, kejadian ini juga terjadi di Stasiun Air Terminal 2.

3. Meskipun Stasiun Air Terminal 2 cenderung sepi dikunjungi (10 orang/-jam), kualitas layanan di stasiun ini cenderung paling buruk. Hal ini disebabkan karena ramainya calon penumpang dari Stasiun Air Terminal 1 (14 orang/jam) dan Stasiun Car Rental (24 orang/jam). Sebelum ke Stasiun Air Terminal 2, bus harus melalui Stasiun Car Rental & Stasiun Air Terminal 1 . Saat bus sampai di Stasiun Air Terminal 2, kemungkinan bus sudah penuh. Hal ini mengakibatkan calon penumpang harus menunggu bus selanjutnya datang.

3.3 Statistik Penumpang

Statistik	Max	Avg	Min
Passanger In System Duration	3:56:19	0:46:48	0:12:54

Hal menarik yang dapat disimpulkan dari hasil statistik ini adalah sebagai berikut :

1. Rata-rata, penumpang menghabiskan waktu perjalanan selama 46 menit.

2. Durasi waktu tercepat yang dialami penumpang saat melakukan perjalanan adalah hanya 12 menit. Hal ini dapat disebabkan karena jarak tempuh yang dekat, dan bis yang tidak terlalu lama menunggu di stasiun.

3. Pada kasus terburuk, penumpang dapat menghabiskan waktu hingga sekitar 4 jam di perjalanan. Hal ini dapat disebabkan oleh kombinasi beberapa faktor, diantaranya waktu menunggu bus yang lama, jauhnya jarak tempuh, dan lamanya bus singgah di stasiun pemberhentian.

3.4 Source Code

Source code program simulasi ini dapat diakses online di `https://github.com/adipurnama141/BusSimulator`

```
import random, sys

def timeFormatter(inputSecond):
 m, s = divmod(inputSecond, 60)
 h, m = divmod(m, 60)
 print "%d:%02d:%02d" % (h, m, s)

def avg(l):
 return reduce(lambda x, y: x + y, l) / len(l)

class Passanger:
 def __init__(self,id,originStation,
     arrivedTimeAtOriginStation):
  self.id = id
  self.originStation = originStation
  self.targetStation = 0
  self.arrivedTimeAtOriginStation =
      arrivedTimeAtOriginStation
  self.arrivedTimeAtTargetStation = 0
  self.timeEnteringTheBus = 0
  self.timeLeavingTheBus = 0
  self.delayTimeInQueue = 0
 def enteringTheBus(self, timeEnteringTheBus):
  self.timeEnteringTheBus = timeEnteringTheBus
  self.delayTimeInQueue = self.timeEnteringTheBus - self.
      arrivedTimeAtOriginStation
 def leavingTheBus(self, timeLeavingTheBus):
  self.timeLeavingTheBus = timeLeavingTheBus

class StationQueue:
 def __init__(self,passangerPerHour,originStation):
  self.futurePassangerQueue = []
  self.currentQueue = []
  self.simulationTime = 0
  self.isFirstStart = True
```

```
    self.maxNumberQueue = 0
    self.qtArea_nPeopleOnQueue = 0
    self.initializeAllFutureEvents(passangerPerHour,
        originStation)
  def initializeAllFutureEvents(self,passangerPerHour,
      originStation):
    global_time = 0.0
    simulation_end = 81
    simulation_end = simulation_end * 3600
    n_passanger = 0
    stationList = []
    while global_time < simulation_end:
      passanger_arrival = random.expovariate(
          passangerPerHour) * 3600
      global_time += passanger_arrival
      if (global_time <= simulation_end):
        n_passanger += 1
        m, s = divmod(global_time, 60)
        h, m = divmod(m, 60)
        a = Passanger(n_passanger,originStation,global_time)
        self.futurePassangerQueue.append(a)
  def updateStationQueue(self,time):
    if (self.isFirstStart):
      self.simulationTime = 0
      self.isFirstStart = False
    else:
      while (self.futurePassangerQueue[0].
          arrivedTimeAtOriginStation <= time):
        passanger = self.futurePassangerQueue.pop(0)
        self.currentQueue.append(passanger)
      self.qtArea_nPeopleOnQueue += len(self.currentQueue) *
          (time - self.simulationTime)
      self.simulationTime = time
      if (self.maxNumberQueue < len(self.currentQueue)):
        self.maxNumberQueue = len(self.currentQueue)
  def peekFuturePassanger(self):
    return self.futurePassangerQueue[0]


class Bus:
  def __init__(self):
    # Time Related
    self.timeArrivedAtStation = 0
    self.carRentalDepartureTime = 0
    self.simulationTime = 0.0
    # Passangers
```

```
 self.passangersInsideBus = []
 self.arrivedPassangers =[]
 self.nOfPassangerFromTerminal = 0
 self.nOfPassangerFromCarRental = 0
 self.nOfPassangerToTerminal1 = 0
 self.nOfPassangerToTerminal2 = 0
 self.nOfPassangerWantToUnloadHere = 0
 # Station Related
 self.StationQueue = dict()
 self.StationQueue[1] = StationQueue(14,1)
 self.StationQueue[2] = StationQueue(10,2)
 self.StationQueue[3] = StationQueue(24,3)
 self.currentPosition = 3
 self.busSpeed = 30 #Miles Per Hour
 #Statistics
 self.busWaitingTimeStatistics = []
 self.loopTimeStatistics = []
 self.qtArea_nPassangerInsideBus = 0
 self.maxNPassangers = 0
 self.isFreshStart = True

def unloadPassangers(self):
 if (self.nOfPassangerWantToUnloadHere > 0):
  for x in range(self.nOfPassangerWantToUnloadHere):
   passangerWantToUnload = self.passangersInsideBus.pop
       (0)
   self.unloadPassanger(passangerWantToUnload)

def loadPassangers(self):
 currentSessionbusWaitingTime = self.
     getMaxbusWaitingTime()
 currentStationQueue = self.StationQueue[self.
     currentPosition]
 isWaitingPhase = True
 while (self.simulationTime <=
     currentSessionbusWaitingTime) and (not self.isFull()
     ) and (isWaitingPhase):

  # Load All Passanger Who Already Waiting in Station
      Queue as Long Bus is Not Full
  while (len(currentStationQueue.currentQueue) > 0) and
      (not self.isFull()):
   passanger = currentStationQueue.currentQueue.pop(0)
   self.loadPassanger(passanger)
   # Additional Waiting Time Due People Loading
```

```
            if (self.simulationTime >
                currentSessionbusWaitingTime):
             currentSessionbusWaitingTime = self.simulationTime

            # Peek Future Passanger
            # If There is Future Passanger Who Will Come Faster
                Before WaitTime Ends. Wait...
            # If No. Wait Until WaitTime Ends. Then Go..
            if (currentStationQueue.peekFuturePassanger().
                arrivedTimeAtOriginStation <=
                currentSessionbusWaitingTime):
             self.setsimulationTime(currentStationQueue.
                peekFuturePassanger().arrivedTimeAtOriginStation)
            else:
             isWaitingPhase = False
             # Force Wait Until 5 Minutes (Even Is Predicted There
                 Is No Passanger Will Come Anyway)
             if (self.simulationTime <
                currentSessionbusWaitingTime):
              self.setsimulationTime(currentSessionbusWaitingTime)
             # Collect The Statistics Data (Waiting Time Duration)
             busWaitingTime = self.simulationTime - self.
                timeArrivedAtStation
             self.addbusWaitingTimeStatistics(busWaitingTime)
             # Collect The Statistics Data (Loop Time)
             if ((self.currentPosition == 3) and (self.
                isFreshStart)):
              self.carRentalDepartureTime = self.simulationTime
             if ((self.currentPosition == 3) and (not self.
                isFreshStart)):
              loopTime = self.simulationTime - self.
                 carRentalDepartureTime
              self.carRentalDepartureTime = self.simulationTime
              self.addLoopTimeStatistics(loopTime)

        def movePlace(self):
         self.isFreshStart = False
         # [STATS] Check if Current Number of People in The Bus
             Breaks Record
         if (self.maxNPassangers < len(self.passangersInsideBus)
             ):
          self.maxNPassangers = len(self.passangersInsideBus)
         if (self.currentPosition == 3):
          self.currentPosition = 1
          self.simulationTime = self.simulationTime + (4.5 /
              self.busSpeed * 3600)
```

```
    self.determinePassangerTargetStation()
    self.nOfPassangerWantToUnloadHere = self.
        nOfPassangerToTerminal1
    self.nOfPassangerFromTerminal = 0
   elif (self.currentPosition == 1):
    self.currentPosition = 2
    self.simulationTime = self.simulationTime + (1 / self.
        busSpeed * 3600)
    self.nOfPassangerWantToUnloadHere = self.
        nOfPassangerToTerminal2
    self.nOfPassangerToTerminal1 = 0
   else:
    self.currentPosition = 3
    self.simulationTime = self.simulationTime + (4.5 /
        self.busSpeed * 3600)
    self.nOfPassangerWantToUnloadHere = self.
        nOfPassangerFromTerminal
    self.nOfPassangerToTerminal2 = 0
   self.timeArrivedAtStation = self.simulationTime
   self.timeoutCheck()

  def timeoutCheck(self):
   self.refreshQueue()
   if self.simulationTime >= (80 *3600):
    self.printStatistics()
    sys.exit()

  def refreshQueue(self):
   self.StationQueue[1].updateStationQueue(self.
       simulationTime)
   self.StationQueue[2].updateStationQueue(self.
       simulationTime)
   self.StationQueue[3].updateStationQueue(self.
       simulationTime)

  def printStatistics(self):
   print "Total Passanger Served : " + str(len(self.
       arrivedPassangers))
   self.printBusWaitingTimeStatistics()
   self.printLoopTimeStatistics()
   self.printNumberPeopleOnBusStatistics()
   self.printNumberPersonQueueStatistics()
   self.printDelayTimeInQueueStatistics()
   self.printPersonInSystemStatistics()

  def unloadPassanger(self,passanger):
```

```
 unloadTime = self.unloadPassangerDuration()
 passanger.timeArrivedAtStation = self.simulationTime
 self.simulationTime += unloadTime
 passanger.timeLeavingTheBus = self.simulationTime
 passanger.targetStation = self.currentPosition
 self.arrivedPassangers.append(passanger)
 if (passanger.originStation == 3):
  self.nOfPassangerFromCarRental -= 1
 else:
  self.nOfPassangerFromTerminal -= 1
 self.qtArea_nPassangerInsideBus = self.
     qtArea_nPassangerInsideBus + (self.simulationTime -
     passanger.timeEnteringTheBus)
 self.timeoutCheck()

def loadPassanger(self,passanger):
 loadTime = self.loadPassangerDuration()
 passanger.enteringTheBus(self.simulationTime + loadTime
     )
 self.passangersInsideBus.append(passanger)
 self.simulationTime = passanger.timeEnteringTheBus
 if (passanger.originStation == 3):
  self.nOfPassangerFromCarRental += 1
 else:
  self.nOfPassangerFromTerminal +=1
 self.timeoutCheck()

def loadPassangerDuration(self):
 rand_var = random.uniform(0,1)
 processed_var = (rand_var * 10) + 15
 return processed_var

def unloadPassangerDuration(self):
 rand_var = random.uniform(0,1)
 processed_var = (rand_var * 8) + 16
 return processed_var

def determinePassangerTargetStation(self):
 for x in range(self.nOfPassangerFromCarRental):
  if (random.uniform(0,1) <= 0.583):
   self.nOfPassangerToTerminal1 += 1
  else:
   self.nOfPassangerToTerminal2 += 1

def printNumberPersonQueueStatistics(self):
 print "Average Number Queue"
```

```
        print "␣Station1␣:",
        print self.StationQueue[1].qtArea_nPeopleOnQueue / self
            .simulationTime
        print "␣Station2␣:",
        print self.StationQueue[2].qtArea_nPeopleOnQueue / self
            .simulationTime
        print "␣Station3␣:",
        print self.StationQueue[3].qtArea_nPeopleOnQueue / self
            .simulationTime
        print "Max␣Number␣Queue"
        print "␣Station1␣:",
        print self.StationQueue[1].maxNumberQueue
        print "␣Station2␣:",
        print self.StationQueue[2].maxNumberQueue
        print "␣Station3␣:",
        print self.StationQueue[3].maxNumberQueue

    def printBusWaitingTimeStatistics(self):
        l = self.busWaitingTimeStatistics
        avgBusWaitTime = avg(l)
        maxBusWaitTime = max(l)
        minBusWaitTime = min(l)
        print "Bus␣Waiting␣Time"
        print "␣Max␣:␣",
        timeFormatter(maxBusWaitTime)
        print "␣Avg␣:␣",
        timeFormatter(avgBusWaitTime)
        print "␣Min␣:␣",
        timeFormatter(minBusWaitTime)

    def printLoopTimeStatistics(self):
        l = self.loopTimeStatistics
        avgLoopTime = avg(l)
        maxLoopTime = max(l)
        minLoopTime = min(l)
        print "Loop␣Time"
        print "␣Max:␣",
        timeFormatter(maxLoopTime)
        print "␣Avg:␣",
        timeFormatter(avgLoopTime)
        print "␣Min:␣",
        timeFormatter(minLoopTime)

    def printNumberPeopleOnBusStatistics(self):
        #Calculate Area Under Qt Based On Remaining Passanger
            in The Bus
```

```
  for x in range(len(self.passangersInsideBus)):
   self.qtArea_nPassangerInsideBus = self.
       qtArea_nPassangerInsideBus + (self.simulationTime -
        self.passangersInsideBus[x].timeEnteringTheBus)
  #Calculate Avg Number of People on The Bus
  avgNumberOfPeopleOnTheBus = self.
      qtArea_nPassangerInsideBus / self.simulationTime
  print "Number of People On The Bus"
  print " Max : ",
  print self.maxNPassangers
  print " Avg : ",
  print avgNumberOfPeopleOnTheBus

# Calculate "Time Spent In System" Statistics by Using
    List of Arrived Passanger
def printPersonInSystemStatistics(self):
  penumpangInSystemTime = []
  for x in range(len(self.arrivedPassangers)):
   penumpangInSystemTime.append(self.arrivedPassangers[x
       ].timeLeavingTheBus - self.arrivedPassangers[x].
       arrivedTimeAtOriginStation)
  l = penumpangInSystemTime
  avgInSystemTime = avg(l)
  maxInSystemTime = max(l)
  minInSystemTime = min(l)
  print "Passanger In System Duration"
  print " Max : ",
  timeFormatter(maxInSystemTime)
  print " Avg : ",
  timeFormatter(avgInSystemTime)
  print " Min : ",
  timeFormatter(minInSystemTime)

# Calculate "Delay Time In Queue" Statistics by Using
    List of Arrived Passanger
def printDelayTimeInQueueStatistics(self):
  max_delayTime = [-1,-1,-1]
  sum_delayTime = [0,0,0]
  avg_delayTime = [0,0,0]
  n_passangerInStation  = [0,0,0]
  for x in range(len(self.arrivedPassangers)):
   station_type = self.arrivedPassangers[x].originStation
       - 1
   n_passangerInStation[station_type] += 1
   sum_delayTime[station_type] += self.arrivedPassangers[
       x].delayTimeInQueue
```

```
    if (self.arrivedPassangers[x].delayTimeInQueue >
        max_delayTime[station_type]):
     max_delayTime[station_type] = self.arrivedPassangers[
         x].delayTimeInQueue
   print "Max Delay Time"
   for x in range(0,3):
    avg_delayTime[x] = sum_delayTime[x] /
        n_passangerInStation[x]
    print " Queue " + str(x+1) + " : ",
    timeFormatter(max_delayTime[x])
   print "Avg Delay Time"
   for x in range(0,3):
    print " Queue " + str(x+1) + " : ",
    timeFormatter(avg_delayTime[x])

  def addbusWaitingTimeStatistics(self,busWaitingTime):
   self.busWaitingTimeStatistics.append(busWaitingTime)

  def addLoopTimeStatistics(self,loopTime):
   self.loopTimeStatistics.append(loopTime)

  def getMaxbusWaitingTime(self):
   return self.simulationTime + (5 * 60)

  def setsimulationTime(self,newtime):
   self.simulationTime = newtime
   self.timeoutCheck()

  def isFull(self):
   return len(self.passangersInsideBus) >= 20

  def getNPassanger(self):
   return len(self.passangersInsideBus)


bus = Bus()
while True:
 bus.unloadPassangers()
 bus.loadPassangers()
 bus.movePlace()
```