LAPORAN TUGAS UAS
PEMODELAN & SIMULASI IF-42-Gab01

SIMULASI TRAFFIC FLOW

Kelompok 18:

Muhammad Zalfa Thoriq - 1301194473

Fauzaan Rakantama - 1301194263

IF-42-GAB01

PROGRAM STUDI INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

BANDUNG

2022

DAFTAR ISI

1. Deskripsi Permasalahan

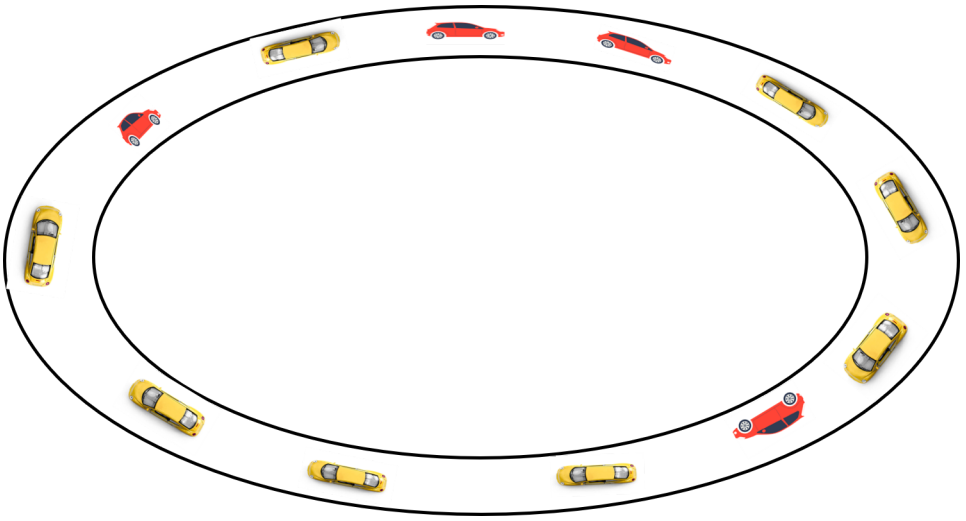
Pemerintah berencana membangun sistem pendeteksi kemacetan jalan raya yang dapat memberikan lokasi titik kemacetan berdasarkan kepadatan kendaraan di satuan unit lokasi. Untuk tujuan tersebut, tim peneliti yang terlibat mencoba membuat sebuah sistem yang dapat menyimulasikan arus kendaraan di jalan raya. Asumsi-asumsi awal jalur kendaraan pun telah disepakati, yaitu:

1. Seluruh kendaraan adalah mobil.
2. Jalur kendaraan memiliki hanya satu jalur berbentuk siklis seperti pada gambar.
3. Perilaku pengendara diatur/dibatasi menjadi hal berikut:

Rule-1 Kendaraan hanya dapat bergerak maju, dan demi keamanan tidak boleh melebihi ke- cepatan tertentu v_{max}.

Rule-2 Pengendara memperhatikan jarak dengan mobil di depan untuk menghindari tabrakan.

Rule-3 Kecenderungan pengendara untuk menginjak rem yang menyebabkan mobil melakukan perlambatan mengikuti peluang tertentu.



Rule-1: Penambahan 1 unit satuan kecepatan untuk kecepatan setiap mobil per satu unit satuan waktu, dan kecepatan setiap mobil tidak boleh melebihi batas kecepatan yang dibolehkan $v(t + 1) = v(t) + 1 \leq v_{max}$.

Rule-2: Misalkan jarak aman antar dua mobil beriringan adalah d, maka kecepatan setiap mobil harus memenuhi $v \leq d - 1$.

Rule-3: Pengemudi memiliki kecenderungan untuk mengerem dengan peluang sebesar p, yang mengakibatkan mobil mengalami perlambatan sebesar 1 satuan unit kecepatan, sehingga jika menginjak rem kecepatan menjadi $v(t+1) = v(t) - 1 \geq 0$

simulasi untuk sistem tersebut dengan parameter tetap

$M = 100$, $p = 0.3$, $v_0 = 0$, $d = 1$, jumlah kendaraan $N = 20$, $t_{max} = 1000$, $v_{max} = 5$

2. Pemodelan Matematika

Setelah melakukan menyederhanaan , dari setiap kendaraan akan mengalami update kecepatan dan posisi dengan menggunakan rumus :

update kecepatan :

$$v(t+1) = \begin{cases} \max\{\min\{v(t)+1, v_{max}, d-1\} - 1, 0\} & \text{dengan peluang } p \\ \min\{v(t)+1, v_{max}, d-1\} & \text{dengan peluang } 1-p \end{cases}$$

update posisi :

$$x(t+1) = \begin{cases} x(t) + v(t+1) & \text{Jika } x(t) + v(t+1) \leq M \\ x(t) + v(t+1) - M - 1 & \text{jika } x(t) + v(t+1) > M \end{cases}$$

3. ALGORITMA

```
[19] import numpy.random as random
     import matplotlib.pyplot as plt
     import numpy as np
     from matplotlib import animation
     from copy import copy
     from operator import itemgetter
```

```
[20] # inisialisasi
     M = 100      #panjang lintasan
     p = 0.3      #probabilitas
     v0 = 0       #kecepatan awal
     N = 20       #banyaknya mobil
     t_max = 1000  #waktu max
     v_max = 5    #kecepatan max
```

```
[21] random.seed(1) #membuat array untuk jalan dan mobil
     jalan = np.array( [ [[0,M+0.5], [0.5,0.5]], [[0,M+0.5], [1.5,1.5]] ] )
     mobil = np.array([[random.randint(1,M), random.randint(1,2)] for i in range(1,N+1)])
     mobil = np.array(sorted(mobil, key=itemgetter(0)))
```

```
     #inisialisasi variable yang akan digunakan pada program
     jml = []
     muter = 0
     a = 0
     v = v0
     count = 0
     movement = []
     antrian = [i for i in range(N)]
     # print(antrian)
```

```
# program
for t in range(t_max):
    x_row = []
    for i in antrian:
        car = mobil[i]
        next_car = mobil[i+1 if i+1 < N else 0]
        # v1
        v = np.min([v+1, v_max])
        # v2
        if (next_car[0] < car[0]):
            d = M - car[0] + next_car[0]
        else:
            d = (next_car[0]-car[0])
        v = np.min([v, d-1])
        # v3
        pr = random.rand()
        if (pr < p):
            v = np.max([0, v-1])

        # update jarak
        x = copy(car[0])
        x = x + v
        if (x >= M):
            x = x - M
        x_row.append(copy([x,car[1]]))

    mobil = copy(x_row)
    movement.append(mobil)

    #kepadatan
    print ('Detik ',t)
    for j in (movement[t]):
        if j[0]>=80 and j[0]<=90:
            count+=1
    print(count/len(movement[t])*100,'%')
    count = 0
print('movement', movement)
```

```
#ratarata
for i in range(len(movement) - 1):
    if a>=len(movement) - 1:
        break
    a = i + 1
    nextemp = movement[a][2]
    temp = movement[i][2]
    if nextemp[0]<temp[0]:
        if nextemp[0] == movement[a][2][0] or nextemp[0] >= movement[a][2][0]:
            jml.append(a)
            muter+=1

avg = jml[-1]/muter
print('Perputaran =',muter)
print('Rata-rata waktu =',avg)
```

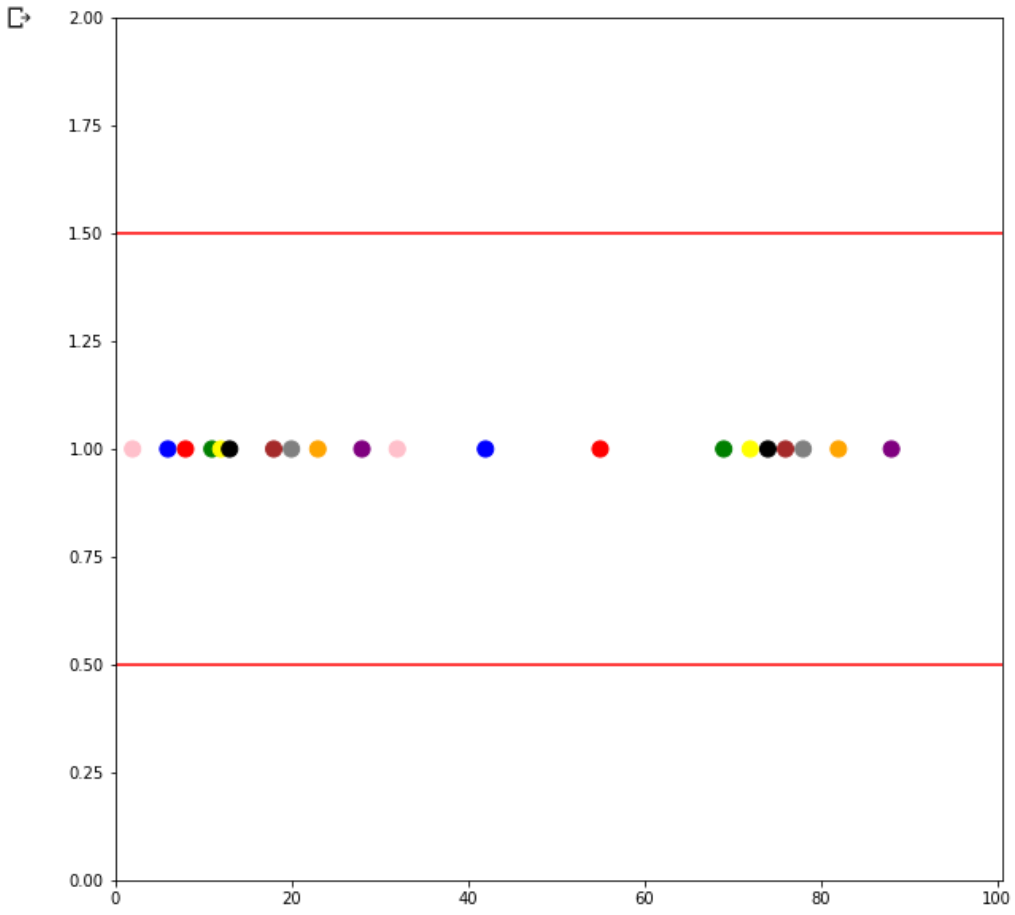
```
Perputaran = 4
Rata-rata waktu = 22.0
```

```
# animasi
fig = plt.figure()
fig.set_size_inches(10, 10)
ax = plt.axes(ylim=(0,2), xlim=(0,M+0.5))
for road in jalan:
    plt.plot(road[0], road[1], c="red")
    x = np.arange(10)
mobil_1 = [x[0] for x in movement[0]]
mobil_2 = [x[1] for x in movement[1]]
color = ['pink', 'blue', 'red', 'green', 'yellow', 'black', 'brown', 'grey', 'orange', 'purple']
car_marker = ax.scatter(mobil_1, mobil_2, c=color, s=100, marker="o")
```

```
[16] def animate(i):
        mobil_position = movement[i]
        car_marker.set_offsets(mobil_position)
        return car_marker

    anim = animation.FuncAnimation(fig, animate, frames=len(movement), interval=100)
    plt.show()
```

3.1 Hasil Simulasi



```
Detik  0
10.0 %
Detik  1
20.0 %
Detik  2
20.0 %
Detik  3
20.0 %
Detik  4
20.0 %
Detik  5
20.0 %
Detik  6
10.0 %
Detik  7
0.0 %
Detik  8
0.0 %
Detik  9
10.0 %
Detik  10
10.0 %
Detik  11
10.0 %
Detik  12
0.0 %
Detik  13
0.0 %
Detik  14
0.0 %
Detik  15
0.0 %
Detik  16
10.0 %
```

```
Detik  17
20.0 %
Detik  18
10.0 %
Detik  19
10.0 %
Detik  20
20.0 %
Detik  21
10.0 %
Detik  22
10.0 %
Detik  23
10.0 %
Detik  24
20.0 %
Detik  25
20.0 %
Detik  26
20.0 %
Detik  27
20.0 %
Detik  28
20.0 %
Detik  29
20.0 %
Detik  30
20.0 %
Detik  31
10.0 %
Detik  32
10.0 %
Detik  33
0.0 %
```

```
Detik  34
0.0 %
Detik  35
0.0 %
Detik  36
10.0 %
Detik  37
10.0 %
Detik  38
20.0 %
Detik  39
10.0 %
Detik  40
20.0 %
Detik  41
20.0 %
Detik  42
10.0 %
Detik  43
10.0 %
Detik  44
10.0 %
Detik  45
10.0 %
Detik  46
10.0 %
Detik  47
10.0 %
Detik  48
10.0 %
Detik  49
10.0 %
Detik  50
10.0 %
```

```
Detik  51
20.0 %
Detik  52
10.0 %
Detik  53
20.0 %
Detik  54
10.0 %
Detik  55
0.0 %
Detik  56
0.0 %
Detik  57
0.0 %
Detik  58
10.0 %
Detik  59
10.0 %
Detik  60
10.0 %
Detik  61
10.0 %
Detik  62
10.0 %
Detik  63
20.0 %
Detik  64
10.0 %
Detik  65
20.0 %
Detik  66
10.0 %
Detik  67
10.0 %
```

```
Detik  68
10.0 %
Detik  69
20.0 %
Detik  70
10.0 %
Detik  71
0.0 %
Detik  72
10.0 %
Detik  73
20.0 %
Detik  74
10.0 %
Detik  75
10.0 %
Detik  76
10.0 %
Detik  77
0.0 %
Detik  78
0.0 %
Detik  79
10.0 %
Detik  80
10.0 %
Detik  81
10.0 %
Detik  82
10.0 %
Detik  83
20.0 %
Detik  84
20.0 %
```

```
Detik  85
20.0 %
Detik  86
20.0 %
Detik  87
20.0 %
Detik  88
10.0 %
Detik  89
10.0 %
Detik  90
20.0 %
Detik  91
10.0 %
Detik  92
0.0 %
Detik  93
10.0 %
Detik  94
10.0 %
Detik  95
20.0 %
Detik  96
10.0 %
Detik  97
20.0 %
Detik  98
10.0 %
Detik  99
0.0 %
```

4. KESIMPULAN

Berdasarkan hasil simulasi dapat disimpulkan bahwa semakin besar waktu density maka diperlukan lebih lama waktu untuk mobil Kembali dari posisi awal , sedangkan jika density semakin kecil maka waktu yang diperlukan semakin cepat. Semakin besar nilai t-max makan akan membuat semakin lama pergerakan mobil. Dapat disimpulkan juga jika nilai m sama dengan 100 makan jalur akan semakin kecil maka dari itu memakai m sama dengan 1000.