LAPORAN TUGAS BESAR
SIMULASI RANDOM WALK 2D
Penyebaran Virus

Laporan ini dibuat untuk memenuhi tugas besar

Mata kuliah Pemodelan dan Simulasi

Disusun oleh:

1. Ghina Khoerunnisa (NIM: 1301181066)
2. Delvanita Sri Wahyuni (NIM: 1301184014)

S1 INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

BANDUNG

2021

Daftar Isi

1. DESKRIPSI PERMASALAHAN

Proses penyebaran suatu penyakit/virus dapat disimulasikan secara sederhana dengan menggunakan Random Walk. Pada metode ini, setiap individu direpresentasikan sebagai partikel yang bergerak bebas secara acak. Proses simulasi diawali dengan mendefinisikan sejumlah individu dari suatu komunitas yang sudah terinfeksi. Setelah itu, simulasi dilakukan dengan mendefinisikan perubahan posisi dari masing-masing individu secara acak. Secara sederhana, proses infeksi terjadi pada saat individu sehat berada pada posisi yang sama dengan individu yang terinfeksi. Selain itu, individu yang sudah sembuh diasumsikan memiliki imun terhadap penyakit/virus sehingga tidak akan terinfeksi untuk kedua kalinya. Proses simulasi berakhir setelah tidak ada lagi individu yang terinfeksi.

Secara lebih detail, ruang simulasi perlu didefinisikan untuk menghindari pergerakan individu yang terlalu menyebar. Terkait hal ini, maka individu yang bergerak melebihi batas area perlu dikontrol dengan menggunakan metode periodic boundary condition (PBC). Selain itu, penerapan karantina wilayah pada level tertentu dapat direpresentasikan dengan mendefinisikan suatu variabel yang menentukan probabilitas suatu individu untuk bergerak. Hasil simulasi tersebut dapat menunjukkan fluktuasi jumlah individu yang terinfeksi tiap harinya dan waktu yang diperlukan oleh komunitas untuk pulih dari wabah penyakit/virus atau tidak ada lagi individu yang terinfeksi. Pada kasus ini, satu iterasi diasumsikan sebagai satu hari.

2. PEMODELAN MATEMATIKA

2.1. Kondisi Simulasi

- Jumlah individu: 200
- Rasio individu terinfeksi: 5%
- Probabilitas individu bergerak: 80%
- Waktu pemulihan: 10 hari
- Ukuran ruang simulasi: 20 x 20 unit

3. DOKUMENTASI

3.1. Algoritma

1. Inisialisasi variabel scalar
    - Jumlah individu
    - Rasio individu yang terinfeksi
    - Waktu pemulihan’
    - Ukuran ruang simulasi
    - Waktu permulihan
    - Ukuran ruang simulasi
    - Probabilitas individu bergerak

2. Inisialisasi variabel list
    - Posisi masing-masing individu
    - Status kesehatan individu (individu dengan rasio tertentu berstatus terinfeksi)
    - Status imunitas individu
    - Waktu terinfeksi individu
3. Iterasi
    - Selama jumlah individu terinfeksi > 0:
        - Untuk setiap individu:
        - Update posisi berdasarkan probabilitas individu bergerak
        - Koreksi posisi dengan PBC
        - Update wkatu terinfeksi untuk individu yang sudah terinfeksi
        - Update status kesehatan individu – recovery
            - Jika waktu terinfeksi > waktu pemulihan, maka individu yang terinfeksi didefinisikan pulih
        - Update status imun individu (individu yang sudah pulih memiliki imun sehingga tidak akan terinfeksi lagi)
            - Update status kesehatan individu – infection
            - Hitung jarak individu sehat dengan individu terinfeksi dan individu tersebut belum memiliki imun, maka individu sehat tersebut terinfeksi
        - Hitung dan simpan jumlah individu terinfeksi

3.2. Screenshot Program

In [1]:
```
from random import shuffle # untuk mengacak nilai pada list
import numpy as np # untuk memanipulasi array
import matplotlib.pyplot as plt # untuk membuat dan menampilkan plot
from collections import Counter # untuk menghitung banyak nilai dari list
```

Inisialisasi variabel scalar

In [2]:
```
jumlah_individu = 200 # jumlah individu yang diobservasi
rasio_infeksi = 0.05 # rasio individu terinfeksi
prob_bergerak = 0.8 # probabilitas/kemungkinan individu bergerak
waktu_pulih = 10 # batas waktu suatu individu dapat dikatakan pulih
x_min = 0 # batas awal sumbu x
x_max = 20 # batas akhir sumbu x
y_min = 0 # batas awal sumbu y
y_max = 20 # batas akhir sumbu y
x_range = x_max - x_min # rentang nilai sumbu x
y_range = y_max - y_min # rentang nilai sumbu y
```

Inisialisasi variabel list

In [3]:
```
x = np.random.randint(low=x_min, high=x_max, size=jumlah_individu) # posisi x masing-masing individu
y = np.random.randint(low=x_min, high=x_max, size=jumlah_individu) # posisi y masing-masing individu
```

In [4]:
```
# status kesehatan individu
# membuat list status kesehatan individu yang terdiri dari nilai 0 dan 1 dengan keterangan sebagai berikut:
# status = 0 (belum terinfeksi) *warna biru*
# status = 1 (terinfeksi) *warna merah*
status = [0] * int((1-rasio_infeksi)*jumlah_individu) + [1] * int((rasio_infeksi)*jumlah_individu)
shuffle(status) # mengacak nilai di dalam list status
# status imunitas individu
# membuat list status imunitas individu yang diinisialisasikan dengan nilai 0 (belum punya imun) sebanyak individu
# imun = 0 (tidak punya imun)
# imun = 1 (punya imun)
imun = [0 for i in range(jumlah_individu)]
# membuat list untuk menyimpan waktu terinfeksi setiap individu
waktu_terinfeksi = [0 for i in range(jumlah_individu)]
```

Fungsi fungsi

In [5]:
```
# membuat scatter plot persebaran individu dengan status 0 dan 1 dan disimpan pada folder sebaran
def make_scatter(x,y,status,i):
    colormap = np.array(['b', 'r'])
    plt.scatter(x, y, s=25, c=colormap[status], alpha=0.5)
    plt.title('Persebaran Hari ke-{:d}'.format(i), fontweight="bold")
    plt.savefig('./sebaran/fig_{:04d}.png'.format(i), format='png', dpi=1000, bbox_inches='tight')
    plt.clf()
```

In [6]:
```
# membuat line plot banyaknya individu terinfeksi disetiap harinya dan disimpan pada folder grafik
def make_lineplot(day, n, i):
    plt.plot(day,n)
    plt.title('Grafik Hari ke-{:d}'.format(i), fontweight="bold")
    plt.xlabel('Hari')
    plt.ylabel('Banyak Individu Terinfeksi')
    plt.xlim(0,100)
    plt.ylim(0,120)
    plt.axhline(c='black')
    plt.axvline(c='black')
    plt.savefig('./grafik/fig_{:04d}.png'.format(i), format='png', dpi=1000, bbox_inches='tight')
    plt.clf()
```

In [7]:
```
# mengupdate posisi dari suatu individu
def update_posisi(random_bergerak, x, y):
    # jika bilangan random yang dihasilkan kurang dari probabilitas bergerak berarti individu tersebut bergerak.
    # jika bergerak maka random kembali suatu bilangan antara 0-1 dan hasilnya akan menentukan arah geraknya
    # dan mengembalikan nilai x dan y terbaru.
    # jika tidak bergerak maka akan mengembalikan nilai x dan y tanpa diubah.
    if random_bergerak <= prob_bergerak:
        random_arah = np.random.rand()
        # bergerak ke kanan
        if random_arah <= 0.25:
            x += 1
        # bergerak ke bawah
        elif random_arah <= 0.5:
            y -= 1
        # bergerak ke kiri
        elif random_arah <= 0.75:
            x -= 1
        # bergerak ke atas
        else:
            y += 1
    return x,y
```

In [8]:
```
# mengkoreksi posisi dari suatu individu jika setelah diupdate nilainya diluar jangkauan yang dibatasi (20x20)
def koreksi_posisi(x,y):
    if x > x_max:
        x -= x_range
    if x < x_min:
        x += x_range
    if y > y_max:
        y -= y_range
    if y < y_min:
        y += y_range
    return x,y
```

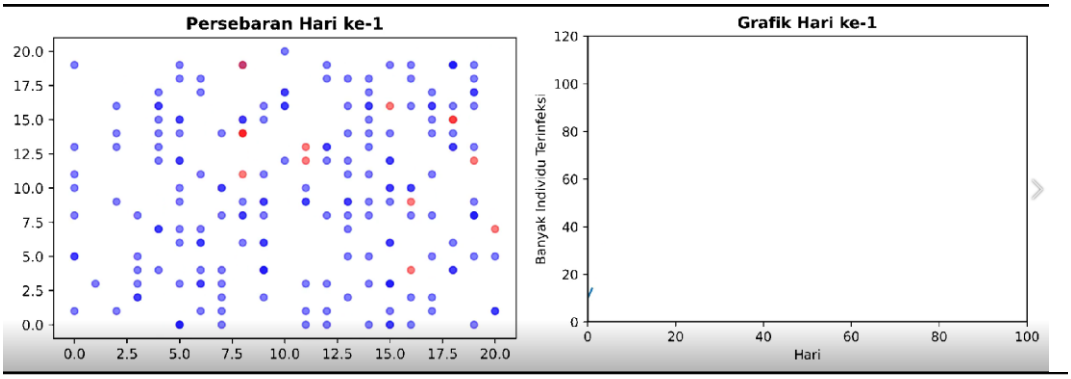
In [9]:
```
# menghitung jarak antara 2 individu
def jarak(x1,y1,x2,y2):
    return ((x1-x2)**2 + (y1-y2)**2)**0.5
```

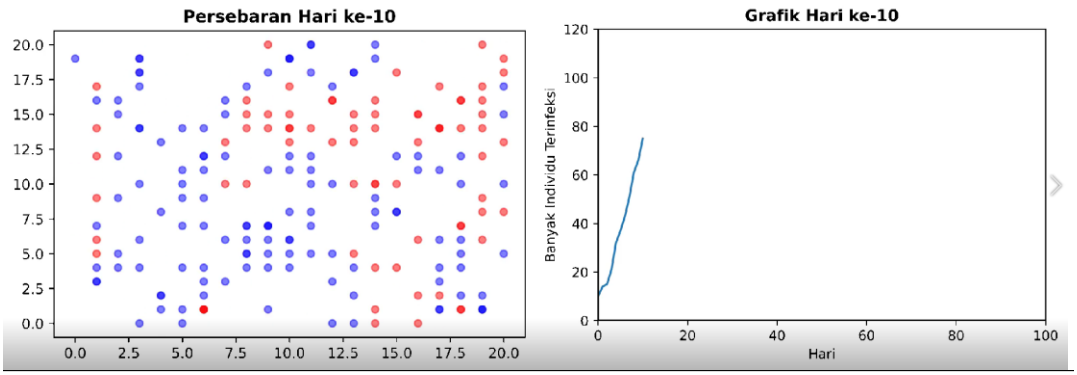
Program utama

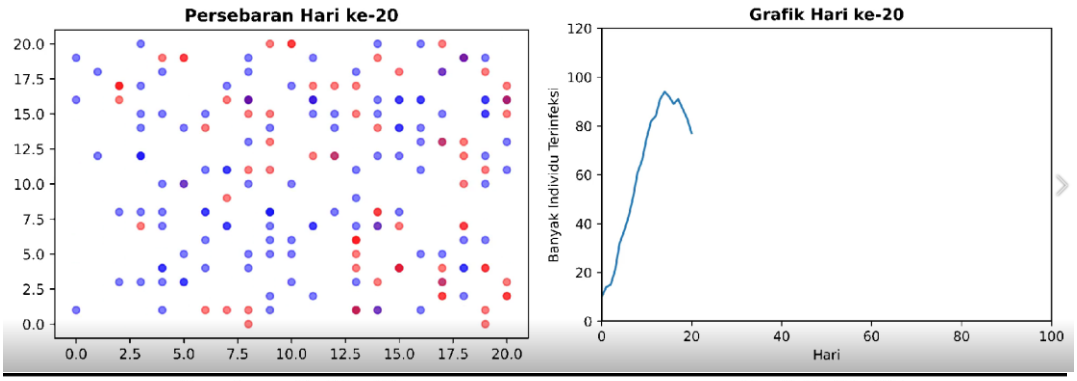
In [10]:
```
# keadaan mula-mula
day=0
days = [day]
terinfeksi = int(0.05*jumlah_individu) # banyaknya individu terinfeksi (5% dari 200 individu)
n = [terinfeksi]
make_scatter(x,y,status,day) # persebaran pada keadaan mula-mula
make_lineplot(days,n,day) # grafik kenaikan banyaknya individu terinfeksi
print("Hari ke-",day,":",terinfeksi)
# iterasi selama masih ada yang terinfeksi
while 1 in status:
    # iterasi untuk semua individu
    for i in range(jumlah_individu):
        # update posisi x dan y berdasarkan probabilitas individu bergerak
        new_posisi = update_posisi(np.random.rand(),x[i],y[i])
        # Koreksi posisi dengan PBC
        x[i], y[i] = koreksi_posisi(new_posisi[0],new_posisi[1])
        # ketika status => 1 yang artinyanya individu terinfeksi
        if status[i] == 1:
            # maka cek waktu terinfeksinya, jika waktu terinfeksinya lebih dari waktu pulih (10 hari)
            if waktu_terinfeksi[i] > waktu_pulih:
                # maka ubah statusnya menjadi pulih / tidak terinfeksi (0) dan individu tersebut memiliki imun (1)
                status[i] = 0
                imun[i] = 1
            else:
                # jika waktu terinfeksinya kurang dari sama dengan waktu pulih maka tambah 1 waktu terinfeksinya
                waktu_terinfeksi[i] +=1
        # ketika status => 0 yang artinya individu sehat / tidak terinfeksi
        elif status[i] == 0:
            # ambil indeks dari individu2 yang terinfeksi
            indeks_terinfeksi = [j for j in range(len(status)) if status[j] == 1]
            # kemudian untuk setiap individu terinfeksi lakukan:
            for idx in indeks_terinfeksi:
                # hitung jaraknya dengan individu yang diobservasi
                # jika jaraknya 0 (berada pada posisi yang sama) dan individu yang diobservasi tidak memiliki imun
                if jarak(x[i],y[i],x[idx],y[idx]) == 0 and imun[i] == 0:
                    # maka ubah statusnya menjadi terinfeksi (1) dan keluar dari perulangan for
                    status[i] = 1
                    break
                else:
                    # jika tidak memenuhi kondisi di atas maka statusnya tetap tidak terinfeksi (0)
                    status[i] = 0
    # tambah hari observasi
    day += 1
    # masukkan ke list untuk membuat line plot
    days.append(day)
    # hitung jumlah individu terinfeksi
    ket = Counter(status)
    # masukkan banyak indivisu terinfeksi ke list untuk membuat line plot
    n.append(ket[1])
    # buat scatter plot persebaran posisi saat iterasi ini
    make_scatter(x,y,status,day)
    # buat line plot grafik banyak individu terinfeksi saat iterasi ini
    make_lineplot(days,n,day)
    print("Hari ke-",day,":",ket[1])
```

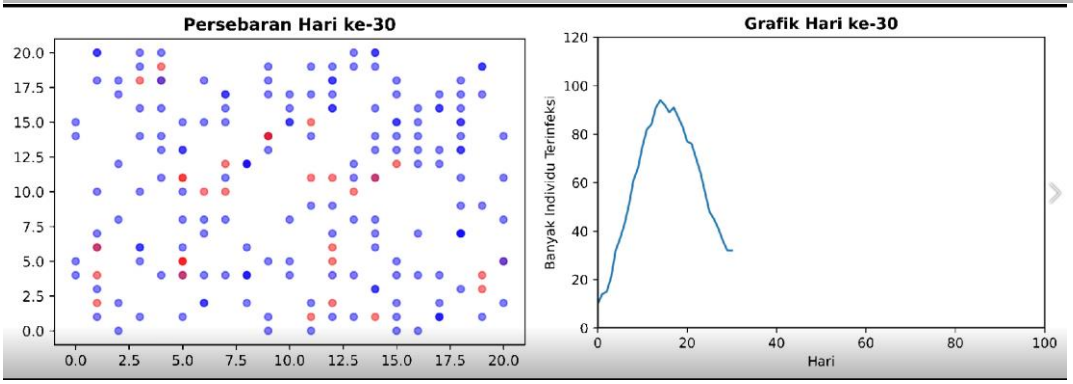
3.3. Hasil Simulasi

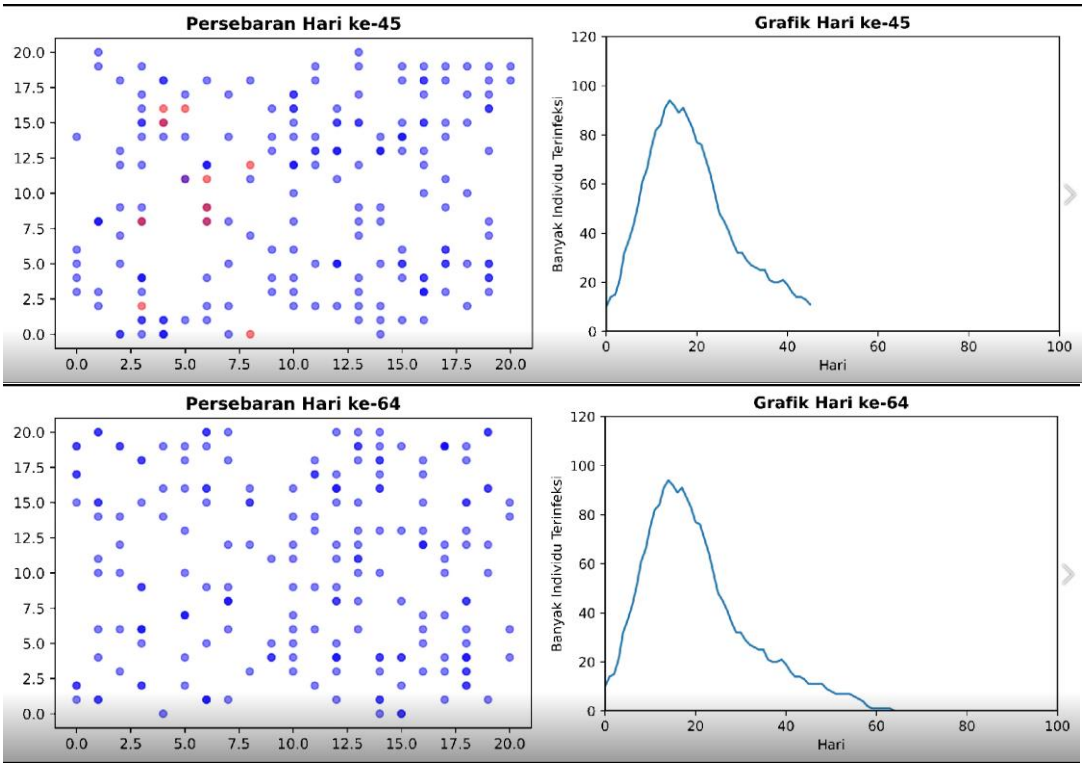
```
Hari ke- 0 : 10
Hari ke- 1 : 14
Hari ke- 2 : 15
Hari ke- 3 : 21
Hari ke- 4 : 32
Hari ke- 5 : 37
Hari ke- 6 : 43
Hari ke- 7 : 51
Hari ke- 8 : 61
Hari ke- 9 : 66
Hari ke- 10 : 75
Hari ke- 11 : 82
Hari ke- 12 : 84
Hari ke- 13 : 91
Hari ke- 14 : 94
Hari ke- 15 : 92
Hari ke- 16 : 89
Hari ke- 17 : 91
Hari ke- 18 : 87
Hari ke- 19 : 83
Hari ke- 20 : 77
Hari ke- 21 : 76
Hari ke- 22 : 70
Hari ke- 23 : 64
Hari ke- 24 : 56
Hari ke- 25 : 48
Hari ke- 26 : 45
Hari ke- 27 : 41
Hari ke- 28 : 36
Hari ke- 29 : 32
Hari ke- 30 : 32
Hari ke- 31 : 29
Hari ke- 32 : 27
Hari ke- 33 : 26
Hari ke- 34 : 25
Hari ke- 35 : 25
Hari ke- 36 : 21
Hari ke- 37 : 20
Hari ke- 38 : 20
Hari ke- 39 : 21
Hari ke- 40 : 19
Hari ke- 41 : 16
Hari ke- 42 : 14
Hari ke- 43 : 14
Hari ke- 44 : 13
Hari ke- 45 : 11
Hari ke- 46 : 11
Hari ke- 47 : 11
Hari ke- 48 : 11
Hari ke- 49 : 9
Hari ke- 50 : 8
Hari ke- 51 : 7
Hari ke- 52 : 7
Hari ke- 53 : 7
Hari ke- 54 : 7
Hari ke- 55 : 6
Hari ke- 56 : 5
Hari ke- 57 : 4
Hari ke- 58 : 2
Hari ke- 59 : 1
Hari ke- 60 : 1
Hari ke- 61 : 1
Hari ke- 62 : 1
Hari ke- 63 : 1
Hari ke- 64 : 0
```


Persebaran Hari ke-1
Grafik Hari ke-1
Banyak Individu Terinfeksi
Hari


Persebaran Hari ke-10
Grafik Hari ke-10
Banyak Individu Terinfeksi
Hari


Persebaran Hari ke-20
Grafik Hari ke-20
Banyak Individu Terinfeksi
Hari


Persebaran Hari ke-30
Grafik Hari ke-30
Banyak Individu Terinfeksi
Hari



4. KESIMPULAN

Berdasarkan simulasi yang dilakukan dapat disimpulkan bahwa dengan kondisi yang diberikan waktu pemulihan yang dibutuhkan oleh komunitas tersebut adalah 64 hari dengan individu terinfeksi paling banyak terjadi pada hari ke-14 yaitu sebanyak 94 individu.