

**LAPORAN TUGAS PEMROGRAMAN GENETIC ALGORITHM  
MATA KULIAH PENGANTAR KECERDASAN BUATAN**



**Disusun oleh :**

Famardi Putra Muhammad Raffly ( 1301204391)

Muhammad Daffa' Ibrahim ( 1301204051 )

**PROGRAM STUDI S1 INFORMATIKA**

**FAKULTAS INFORMATIKA**

**UNIVERSITAS TELKOM**

**TAHUN 2022**

## I. PENDAHULUAN

Algoritma Genetika merupakan algoritma untuk mencari solusi terbaik sesuai dengan nilai fitness dari permasalahan yang memiliki banyak solusi dan didasarkan pada evolusi genetika yang terjadi pada makhluk hidup yang pertama kali dikembangkan pada tahun 1975 oleh John Holland dari New York. Algoritma ini menggunakan pendekatan evolusi Darwin di bidang Biologi seperti pewarisan sifat, seleksi alam, mutasi gen dan *crossover*.

Untuk mencari solusi terbaik pada algoritma ini, dapat dimulai dengan membuat solusi - solusi dan dibentuk menjadi populasi dengan individu secara acak. Kemudian, mengevaluasi nilai fitness dari tiap individu dan menyeleksi nya agar dapat memilih individu terbaik (Orang tua). Individu orang tua yang terpilih akan melakukan pindah silang (*crossover*) yang kemudian hasil pindah silang tersebut dimutasi untuk mendapatkan individu baru. Setelah itu masuk ke langkah seleksi survivor (*Steady-State*) dimana individu yang lemah akan digantikan dengan individu hasil dari mutasi sebelumnya. Semua proses tersebut diulang hingga menemukan individu dari populasi dengan nilai fitness terbaik.

## II. ANALISIS MASALAH

### 1. Desain Kromosom dan Metode Dekode

Diberikan sebuah kasus untuk menemukan nilai minimum dari fungsi 
$$h(x, y) = \frac{(\cos x + \sin y)^2}{x^2 + y^2}$$
 menggunakan algoritma genetika dengan batasan  $-5 \leq x \leq 5$  dan  $-5 \leq y \leq 5$ . Pada kasus tersebut, kita dapat merepresentasikan individu dengan menggunakan representasi real, integer, dan biner. Tetapi, karena kasus tersebut mencari sebuah nilai, kami menerapkan representasi integer untuk merepresentasikan individu.

#### 1.1 Desain Kromosom

x1	x2	x3	y1	y2	y3
5	2	8	1	6	2

Berikut algoritma dari fungsi kromosom :

```
#Membuat individu/kromosom baru
def individu(length):
    indiv = [random.randint(0,9) for x in range(length)]
    return indiv
```

## 1.2 Metode Dekode

$$x = r_{min} + \frac{r_{max} - r_{min}}{\sum_{i=1}^N 9 * 10^{-i}} (g_1 * 10^{-1} + g_1 * 10^{-2} + \dots + g_N * 10^{-N})$$

Berikut algoritma dari fungsi dekode :

```
#Fungsi decode
def decode(krom, interval):
    sigma = 0
    g = 0
    #rumus integer
    for i in range(len(krom)) :
        n = krom[i]
        g += (n * (10**-(i+1)))
        sigma += (9 * (10**-(i+1)))
    hasil = interval[0] + (((interval[1]-interval[0]) / sigma) * g)
    return hasil
```

## 2. Ukuran Populasi

Ukuran populasi merupakan jumlah populasi yang akan diproses pada langkah-langkah selanjutnya. Populasi sendiri merupakan kumpulan dari banyak kromosom, dalam satu populasi akan terdapat lebih dari satu kromosom. Sebelum menghitung ukuran populasi, kita membutuhkan panjang suatu kromosom dan jumlah kromosom yang diinginkan dalam satu populasi. Dalam kasus ini, kami menentukan N panjang dari tiap kromosom, kemudian kami menentukan N banyak kromosom dalam satu populasi. Berikut merupakan algoritma desain ukuran populasi :

```
#Membuat individu/kromosom baru
def individu(length):
    indiv = [random.randint(0,9) for x in range(length)]
    return indiv

#Membuat populasi dari individu/kromosom yang telah dibuat
def populasi(pop,length):
    pops = [individu(length) for x in range(pop)]
    return pops
```

## 3. Fungsi Fitness

Untuk mencari sebuah nilai minimum dari fungsi  $h(x,y)$ , kami menggunakan fungsi minimasi dengan rumus  $f = \frac{l}{(h+a)}$ . Kemudian, kami juga membuat fungsi untuk menghitung nilai fitness dari populasi yang ada. Berikut merupakan algoritma untuk fungsi minimasi dan menghitung fitness :

```
def split(krom):
    spl = (krom[:len(krom)//2], krom[len(krom)//2:])
    return spl

#Fungsi decode
def decode(krom, interval):
    sigma = 0
    g = 0
    #rumus integer
    for i in range(len(krom)):
        n = krom[i]
        g += (n * (10**-(i+1)))
        sigma += (9 * (10**-(i+1)))
    hasil = interval[0] + (((interval[1]-interval[0]) / sigma) * g)
    return hasil
```

```
#Fungsi minimasi
def fit_value(x,y):
    return 1/(0.01 + function(x,y))

#Menghitung nilai fitness dari masing masing individu/kromosom pada populasi
def hitungfitness(populasi):
    fitness_populasi=[]
    for i in range(len(populasi)):
        x,y = split(populasi[i])
        kromX = decode(x,intervalX)
        kromY = decode(y,intervalY)
        fit_value(kromX,kromY)
        fitness_populasi.append(fit_value(kromX,kromY))

    return fitness_populasi
```

#### 4. Metode Pemilihan Orang tua

Kami menggunakan metode Tournament Selection dalam pemilihan orangtua. Metode ini akan memilih sejumlah k kromosom secara acak(random) dari populasi yang berukuran N kromosom yang mana akan dipilih 2 kromosom dengan nilai fitness terbaik yang akan dijadikan sebagai orangtua. Pada kasus ini, kami memilih 4 kromosom secara acak dari populasi dan memilih 2 kromosom yang memiliki nilai fitness terbaik. Berikut merupakan algoritma desain dalam pemilihan orang tua :

```
#Seleksi 10 individu yang ada pada populasi sehingga menjadi 2 parent
def seleksi_orangtua(populasi):
    fitness_cand = []
    cand = []
    for i in range(4):
        n = random.randint(0,len(populasi)-1)
        cand.append(populasi[n])
    fitness_cand = hitungfitness(cand)

    parent1 = fitness_cand[0]
    for j in range(len(fitness_cand)):
        if (fitness_cand[j] >= parent1):
            parent1 = fitness_cand[j]
            idx_parent1 = fitness_cand.index(parent1)

    parent2 = fitness_cand[0]
    if (parent2 == parent1):
        parent2 = fitness_cand[1]

    idx_parent2 = 0
    for k in range(len(fitness_cand)):
        if(fitness_cand[k] != parent1 and fitness_cand[k] >= parent2):
            parent2 = fitness_cand[k]
            idx_parent2 = fitness_cand.index(parent2)

    return (cand[idx_parent1], cand[idx_parent2])
```

## 5. Metode Operasi Genetik (pindah silang dan mutasi)

Metode Operasi Genetik meliputi *cross over* (pindah silang) dan mutasi. Metode *cross over* yang kami gunakan dalam kasus ini adalah rekombinasi satu titik (*Single-point crossover*). Kemudian kami menggunakan metode *fix point* dalam menentukan titik potong. Dalam kasus ini, kami menjadikan titik tengah dari panjang kromosom sebagai titik potong.

Metode mutasi yang kami gunakan adalah metode memilih nilai secara acak. Jumlah gen yang akan dimutasi pada kasus ini adalah 2 gen. kami akan memilih secara acak 2 gen yang nantinya akan diganti dengan gen baru berupa integer acak dalam interval  $[0,9]$ . Berikut merupakan algoritma *crossover* dan mutasi :

```
#Menyilangkan kedua parent tersebut dan menghasilkan child yang baru
def cross_over(parent1,parent2,pejuang):
    c1 = []
    c2 = []
    #n adalah titik silang
    n = (len(parent1) // 2)
    if random.random() >= peluang:
        c1 = parent1
        c2 = parent2
    else:
        c1[:n]=parent1[:n]
        c1[n:]=parent2[n:]
        c2[:n]=parent2[:n]
        c2[n:]=parent1[n:]
    return (c1,c2)
```

```
#Mutasi child tersebut sehingga menghasilkan child yang baru
def mutasi(child,pejuang):
    if random.random() <= peluang:
        #posisi mutasi
        x = random.randint(1,len(child)-1)
        y = random.randint(1,len(child)-1)
        while (y == x):
            y = random.randint(1,len(child)-1)

        #nilai mutasi
        m = random.randint(0,9)
        n = random.randint(0,9)
        child[x] = m
        child[y] = n
    return child
```

## 6. Probabilitas operasi genetik ( $P_c$ dan $P_m$ )

Semakin tinggi angka probabilitas *crossover* ( $P_c$ ) dan probabilitas mutasi ( $P_m$ ), maka semakin besar kemungkinan terjadinya *crossover* dan mutasi dalam sebuah gen pada kromosom. Begitu juga sebaliknya, semakin rendah angka probabilitas *crossover* ( $P_c$ ) dan probabilitas mutasi ( $P_m$ ), semakin kecil kemungkinan sebuah gen dalam kromosom untuk melakukan *crossover* dan mutasi. Pada kasus di atas, kami menggunakan 0.7 untuk angka probabilitas *crossover* ( $P_c$ ) dan probabilitas mutasi ( $P_m$ ).

## 7. Metode Pergantian Generasi (Seleksi Survivor)

Metode pergantian generasi atau seleksi survivor yang kami gunakan adalah metode *Steady-State* dengan *Fitness-based selection*. Metode *Steady-State* akan menghapus beberapa kromosom yang terdapat pada populasi sebelumnya, kemudian digantikan dengan kromosom child hasil operasi genetik (pindah silang dan mutasi). *Fitness-based selection* akan menghapus N kromosom yang memiliki nilai Fitness paling buruk dalam populasi tersebut. Dalam kasus ini, kami menghapus 2 kromosom terburuk dari suatu populasi, kemudian digantikan dengan 2 child hasil operasi genetik (pindah silang dan mutasi).

```
#Seleksi_Survivor dengan regenerasi yaitu mengganti individu/kromosom terburuk dengan child baru hasil mutasi
def regenerasi(populasi,c1,c2):
    fit = hitungfitness(populasi)
    fitness1 = hitungfitness(populasi)
    fitness2 = fitness1
    fitness2.sort()

    a = fitness2[0]
    b = fitness2[1]

    #Mencari individu terburuk pertama dan menghapusnya
    for i in range(len(populasi)) :
        if fit[i] == a :
            x = i

    populasi.pop(x)
    populasi.append(c1) #memasukkan child1 baru ke dalam populasi

    #Mencari individu terburuk kedua dan menghapusnya
    for j in range(len(populasi)) :
        if fit[j] == b :
            y = j

    populasi.pop(y)
    populasi.append(c2) #memasukkan child2 baru ke dalam populasi

    return populasi
```

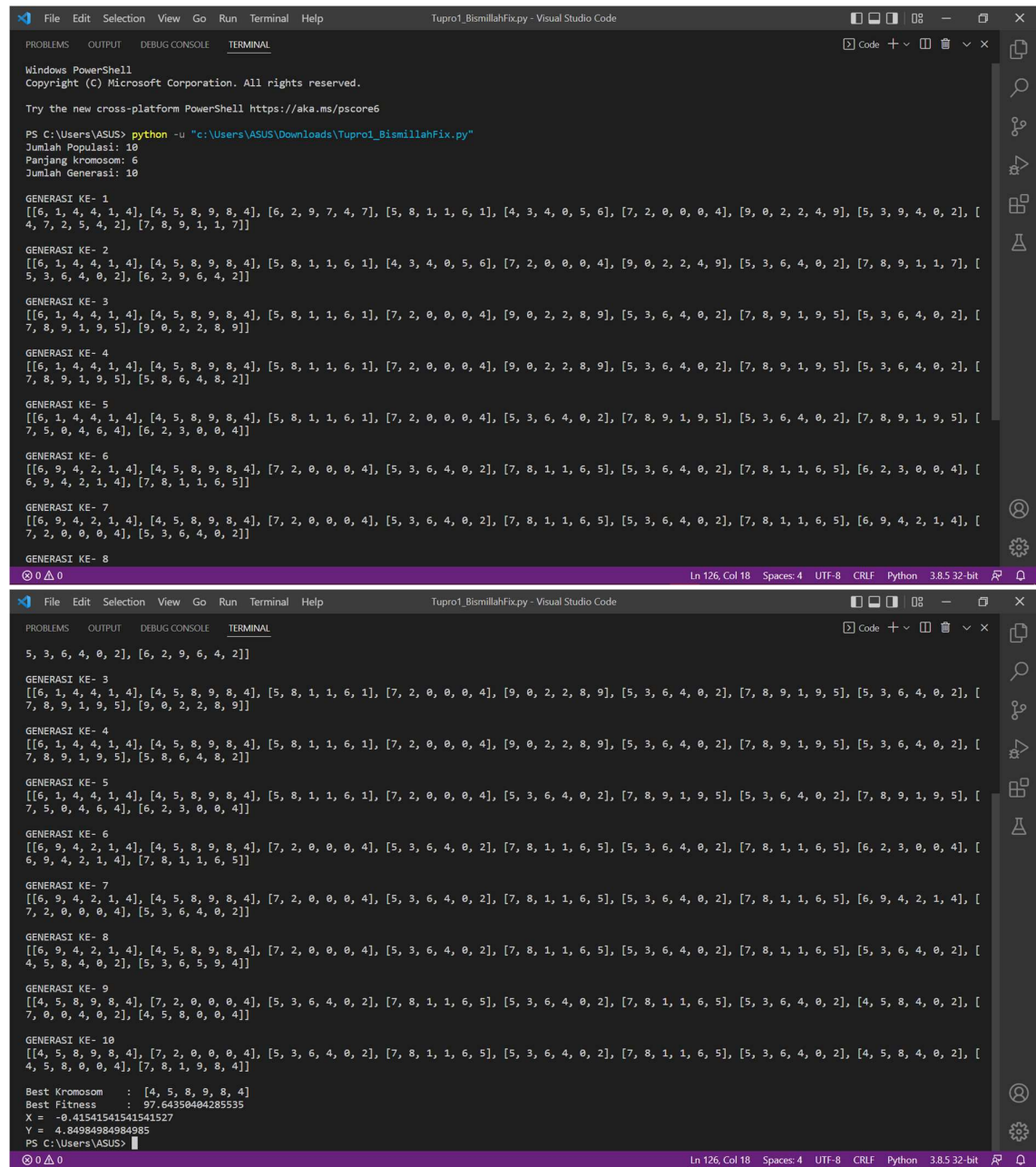
## 8. Kriteria penghentian evolusi (loop)

- a) Model Populasi: Generational Model.
- b) Seleksi Orangtua: *Tournament Selection*.
- c) Rekombinasi: *Single-point crossover* dengan pemilihan titik potong *fix point*.
- d) Mutasi: Pemilihan nilai secara acak.
- e) Seleksi Survivor: *Steady-state* dengan *Fitness-based selection*.

### III. RUNNING PROGRAM

#### 1. Pengujian 1

Pada pengujian pertama, kami gunakan sebagai acuan kami untuk melakukan pengujian berikutnya. Kami membuat 10 kromosom dalam 1 populasi yang panjang setiap kromosomnya 6 dan akan beregenerasi sebanyak 10 kali. Setelah program dijalankan, kami mendapatkan nilai fitness terbaik yaitu 97,64.



```
File Edit Selection View Go Run Terminal Help
Tupro1_BismillahFix.py - Visual Studio Code

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Windows PowerShell
Copyright (c) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\ASUS> python -u "c:\Users\ASUS\Downloads\Tupro1_BismillahFix.py"
Jumlah Populasi: 10
Panjang Kromosom: 6
Jumlah Generasi: 10

GENERASI KE- 1
[[6, 1, 4, 4, 1, 4], [4, 5, 8, 9, 8, 4], [6, 2, 9, 7, 4, 7], [5, 8, 1, 1, 6, 1], [4, 3, 4, 0, 5, 6], [7, 2, 0, 0, 0, 4], [9, 0, 2, 2, 4, 9], [5, 3, 9, 4, 0, 2], [4, 7, 2, 5, 4, 2], [7, 8, 9, 1, 1, 7]]

GENERASI KE- 2
[[6, 1, 4, 4, 1, 4], [4, 5, 8, 9, 8, 4], [5, 8, 1, 1, 6, 1], [4, 3, 4, 0, 5, 6], [7, 2, 0, 0, 0, 4], [9, 0, 2, 2, 4, 9], [5, 3, 6, 4, 0, 2], [7, 8, 9, 1, 1, 7], [5, 3, 6, 4, 0, 2], [6, 2, 9, 6, 4, 2]]

GENERASI KE- 3
[[6, 1, 4, 4, 1, 4], [4, 5, 8, 9, 8, 4], [5, 8, 1, 1, 6, 1], [7, 2, 0, 0, 0, 4], [9, 0, 2, 2, 8, 9], [5, 3, 6, 4, 0, 2], [7, 8, 9, 1, 9, 5], [5, 3, 6, 4, 0, 2], [7, 8, 9, 1, 9, 5], [9, 0, 2, 2, 8, 9]]

GENERASI KE- 4
[[6, 1, 4, 4, 1, 4], [4, 5, 8, 9, 8, 4], [5, 8, 1, 1, 6, 1], [7, 2, 0, 0, 0, 4], [9, 0, 2, 2, 8, 9], [5, 3, 6, 4, 0, 2], [7, 8, 9, 1, 9, 5], [5, 3, 6, 4, 0, 2], [7, 8, 9, 1, 9, 5], [5, 8, 6, 4, 8, 2]]

GENERASI KE- 5
[[6, 1, 4, 4, 1, 4], [4, 5, 8, 9, 8, 4], [5, 8, 1, 1, 6, 1], [7, 2, 0, 0, 0, 4], [5, 3, 6, 4, 0, 2], [7, 8, 9, 1, 9, 5], [5, 3, 6, 4, 0, 2], [7, 8, 9, 1, 9, 5], [7, 5, 0, 4, 6, 4], [6, 2, 3, 0, 0, 4]]

GENERASI KE- 6
[[6, 9, 4, 2, 1, 4], [4, 5, 8, 9, 8, 4], [7, 2, 0, 0, 0, 4], [5, 3, 6, 4, 0, 2], [7, 8, 1, 1, 6, 5], [5, 3, 6, 4, 0, 2], [7, 8, 1, 1, 6, 5], [6, 2, 3, 0, 0, 4], [6, 9, 4, 2, 1, 4], [7, 8, 1, 1, 6, 5]]

GENERASI KE- 7
[[6, 9, 4, 2, 1, 4], [4, 5, 8, 9, 8, 4], [7, 2, 0, 0, 0, 4], [5, 3, 6, 4, 0, 2], [7, 8, 1, 1, 6, 5], [5, 3, 6, 4, 0, 2], [7, 8, 1, 1, 6, 5], [6, 9, 4, 2, 1, 4], [7, 2, 0, 0, 0, 4], [5, 3, 6, 4, 0, 2]]

GENERASI KE- 8
5, 3, 6, 4, 0, 2], [6, 2, 9, 6, 4, 2]]

GENERASI KE- 3
[[6, 1, 4, 4, 1, 4], [4, 5, 8, 9, 8, 4], [5, 8, 1, 1, 6, 1], [7, 2, 0, 0, 0, 4], [9, 0, 2, 2, 8, 9], [5, 3, 6, 4, 0, 2], [7, 8, 9, 1, 9, 5], [5, 3, 6, 4, 0, 2], [7, 8, 9, 1, 9, 5], [9, 0, 2, 2, 8, 9]]

GENERASI KE- 4
[[6, 1, 4, 4, 1, 4], [4, 5, 8, 9, 8, 4], [5, 8, 1, 1, 6, 1], [7, 2, 0, 0, 0, 4], [9, 0, 2, 2, 8, 9], [5, 3, 6, 4, 0, 2], [7, 8, 9, 1, 9, 5], [5, 3, 6, 4, 0, 2], [7, 8, 9, 1, 9, 5], [5, 8, 6, 4, 8, 2]]

GENERASI KE- 5
[[6, 1, 4, 4, 1, 4], [4, 5, 8, 9, 8, 4], [5, 8, 1, 1, 6, 1], [7, 2, 0, 0, 0, 4], [5, 3, 6, 4, 0, 2], [7, 8, 9, 1, 9, 5], [5, 3, 6, 4, 0, 2], [7, 8, 9, 1, 9, 5], [7, 5, 0, 4, 6, 4], [6, 2, 3, 0, 0, 4]]

GENERASI KE- 6
[[6, 9, 4, 2, 1, 4], [4, 5, 8, 9, 8, 4], [7, 2, 0, 0, 0, 4], [5, 3, 6, 4, 0, 2], [7, 8, 1, 1, 6, 5], [5, 3, 6, 4, 0, 2], [7, 8, 1, 1, 6, 5], [6, 2, 3, 0, 0, 4], [6, 9, 4, 2, 1, 4], [7, 8, 1, 1, 6, 5]]

GENERASI KE- 7
[[6, 9, 4, 2, 1, 4], [4, 5, 8, 9, 8, 4], [7, 2, 0, 0, 0, 4], [5, 3, 6, 4, 0, 2], [7, 8, 1, 1, 6, 5], [5, 3, 6, 4, 0, 2], [7, 8, 1, 1, 6, 5], [6, 9, 4, 2, 1, 4], [7, 2, 0, 0, 0, 4], [5, 3, 6, 4, 0, 2]]

GENERASI KE- 8
[[6, 9, 4, 2, 1, 4], [4, 5, 8, 9, 8, 4], [7, 2, 0, 0, 0, 4], [5, 3, 6, 4, 0, 2], [7, 8, 1, 1, 6, 5], [5, 3, 6, 4, 0, 2], [7, 8, 1, 1, 6, 5], [5, 3, 6, 4, 0, 2], [4, 5, 8, 4, 0, 2], [5, 3, 6, 5, 9, 4]]

GENERASI KE- 9
[[4, 5, 8, 9, 8, 4], [7, 2, 0, 0, 0, 4], [5, 3, 6, 4, 0, 2], [7, 8, 1, 1, 6, 5], [5, 3, 6, 4, 0, 2], [7, 8, 1, 1, 6, 5], [5, 3, 6, 4, 0, 2], [4, 5, 8, 4, 0, 2], [7, 0, 0, 4, 0, 2], [4, 5, 8, 0, 0, 4]]

GENERASI KE- 10
[[4, 5, 8, 9, 8, 4], [7, 2, 0, 0, 0, 4], [5, 3, 6, 4, 0, 2], [7, 8, 1, 1, 6, 5], [5, 3, 6, 4, 0, 2], [7, 8, 1, 1, 6, 5], [5, 3, 6, 4, 0, 2], [4, 5, 8, 4, 0, 2], [4, 5, 8, 0, 0, 4], [7, 8, 1, 9, 8, 4]]

Best Kromosom : [4, 5, 8, 9, 8, 4]
Best Fitness : 97.6435040285535
X = -0.41541541541541527
Y = 4.84984984984985
PS C:\Users\ASUS>
```



## 2. Pengujian 2

Pada pengujian kedua ini, kami membuat kromosom pada populasi menjadi 2 kali lebih banyak dari pengujian sebelumnya yang panjang setiap kromosomnya 6 dan akan beregenerasi sebanyak 10 kali. Didapatkan nilai fitness terbaik sebesar 97,80.

```
Tuproi_BismillahFix.py - Visual Studio Code

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\ASUS> python -u "c:\Users\ASUS\Downloads\Tuproi_BismillahFix.py"
Jumlah Populasi: 20
Panjang kromosom: 6
Jumlah Generasi: 10

GENERASI KE- 1
[[0, 7, 7, 9, 5, 8], [7, 4, 2, 5, 6, 7], [7, 2, 6, 9, 4, 2], [6, 3, 4, 4, 2, 0], [9, 0, 8, 1, 9, 9], [2, 3, 9, 6, 2, 3], [4, 9, 1, 7, 5, 6], [8, 6, 6, 5, 6, 3], [2, 5, 7, 6, 7, 9], [8, 1, 3, 7, 7, 6], [6, 0, 1, 9, 1, 0], [8, 4, 7, 2, 7, 0], [6, 6, 7, 4, 3, 4], [5, 1, 2, 7, 2, 8], [1, 6, 2, 3, 4, 7], [5, 3, 6, 7, 7, 4], [7, 6, 9, 8, 5], [8, 6, 4, 2, 4, 1], [9, 3, 4, 5, 1, 8], [1, 2, 8, 5, 8, 9]]

GENERASI KE- 2
[[0, 7, 7, 9, 5, 8], [7, 4, 2, 5, 6, 7], [7, 2, 6, 9, 4, 2], [6, 3, 4, 4, 2, 0], [9, 1, 8, 1, 8, 9], [2, 3, 9, 6, 2, 3], [8, 6, 6, 5, 6, 3], [2, 5, 7, 6, 7, 9], [8, 1, 3, 7, 7, 6], [6, 0, 1, 9, 1, 0], [8, 4, 7, 2, 7, 0], [6, 6, 7, 4, 3, 4], [1, 6, 2, 3, 4, 7], [5, 3, 6, 7, 7, 4], [7, 6, 6, 9, 8, 5], [8, 6, 4, 2, 4, 1], [9, 3, 4, 5, 1, 8], [1, 2, 8, 5, 8, 9], [9, 1, 8, 1, 8, 9], [8, 6, 4, 2, 4, 1]]

GENERASI KE- 3
[[0, 7, 7, 9, 5, 8], [7, 4, 2, 5, 6, 7], [7, 2, 6, 9, 4, 2], [6, 3, 4, 4, 2, 0], [9, 1, 8, 1, 8, 9], [2, 3, 9, 6, 2, 3], [8, 6, 6, 5, 6, 3], [2, 5, 7, 6, 7, 9], [8, 1, 3, 7, 7, 6], [6, 0, 1, 9, 1, 0], [8, 4, 7, 2, 7, 0], [6, 6, 7, 4, 3, 4], [5, 3, 6, 7, 7, 4], [8, 6, 4, 2, 4, 1], [9, 3, 4, 5, 1, 8], [1, 2, 8, 5, 8, 9], [9, 1, 8, 1, 8, 9], [8, 6, 4, 2, 4, 1], [8, 1, 3, 2, 4, 1], [8, 7, 9, 7, 7, 6]]

GENERASI KE- 4
[[0, 7, 7, 9, 5, 8], [7, 4, 2, 5, 6, 7], [7, 2, 6, 9, 4, 2], [6, 3, 4, 4, 2, 0], [9, 1, 8, 1, 8, 9], [2, 3, 9, 6, 2, 3], [8, 6, 6, 5, 6, 3], [2, 5, 7, 6, 7, 9], [8, 1, 3, 7, 7, 6], [6, 0, 1, 9, 1, 0], [6, 6, 7, 4, 3, 4], [8, 6, 4, 2, 4, 1], [9, 3, 4, 5, 1, 8], [1, 2, 8, 5, 8, 9], [9, 1, 8, 1, 8, 9], [8, 6, 4, 2, 4, 1], [8, 1, 3, 2, 4, 1], [8, 7, 9, 7, 7, 6], [9, 3, 4, 5, 1, 8]]

GENERASI KE- 5
[[0, 7, 7, 9, 5, 8], [7, 4, 2, 5, 6, 7], [7, 2, 6, 9, 4, 2], [6, 3, 4, 4, 2, 0], [9, 1, 8, 1, 8, 9], [2, 3, 9, 6, 2, 3], [8, 6, 6, 5, 6, 3], [2, 5, 7, 6, 7, 9], [8, 1, 3, 7, 7, 6], [6, 0, 1, 9, 1, 0], [8, 6, 4, 2, 4, 1], [9, 3, 4, 5, 1, 8], [1, 2, 8, 5, 8, 9], [9, 1, 8, 1, 8, 9], [8, 6, 4, 2, 4, 1], [8, 1, 3, 2, 4, 1], [9, 3, 4, 1, 8, 9], [9, 4, 8, 5, 1, 8], [2, 3, 9, 5, 9, 1], [8, 6, 6, 6, 2, 3]]

GENERASI KE- 6
[[0, 7, 7, 9, 5, 8], [7, 4, 2, 5, 6, 7], [6, 3, 4, 4, 2, 0], [9, 1, 8, 1, 8, 9], [2, 3, 9, 6, 2, 3], [8, 6, 6, 5, 6, 3], [2, 5, 7, 6, 7, 9], [8, 1, 3, 7, 7, 6], [6, 0, 1, 9, 1, 0], [8, 6, 4, 2, 4, 1], [9, 3, 4, 5, 1, 8], [1, 2, 8, 5, 8, 9], [9, 1, 8, 1, 8, 9], [8, 6, 4, 2, 4, 1], [9, 3, 4, 1, 8, 9], [9, 4, 8, 5, 1, 8], [2, 3, 9, 5, 9, 1], [8, 6, 6, 6, 2, 3], [8, 6, 6, 5, 6, 7], [7, 4, 2, 6, 2, 3]]

GENERASI KE- 7
[[0, 7, 7, 9, 5, 8], [7, 4, 2, 5, 6, 7], [6, 3, 4, 4, 2, 0], [9, 1, 8, 1, 8, 9], [2, 3, 9, 6, 2, 3], [8, 6, 6, 5, 6, 3], [2, 5, 7, 6, 7, 9], [8, 1, 3, 7, 7, 6], [6, 0, 1, 9, 1, 0], [8, 6, 4, 2, 4, 1], [9, 3, 4, 5, 1, 8], [1, 2, 8, 5, 8, 9], [9, 1, 8, 1, 8, 9], [8, 6, 4, 2, 4, 1], [9, 3, 4, 1, 8, 9], [9, 4, 8, 5, 1, 8], [2, 3, 9, 5, 9, 1], [8, 6, 6, 6, 2, 3], [8, 6, 6, 5, 6, 7], [7, 4, 6, 6, 1, 3]]

GENERASI KE- 8
[[0, 7, 7, 9, 5, 8], [7, 1, 3, 5, 6, 7], [9, 1, 8, 1, 8, 9], [2, 3, 9, 6, 2, 3], [8, 6, 6, 5, 6, 3], [2, 4, 4, 6, 7, 9], [8, 1, 3, 7, 7, 6], [6, 0, 1, 9, 1, 0], [8, 6, 4, 2, 4, 1], [9, 3, 4, 9, 1, 7], [1, 2, 8, 5, 8, 9], [9, 1, 8, 1, 8, 9], [2, 3, 9, 5, 9, 1], [8, 6, 6, 6, 2, 3], [8, 6, 6, 5, 6, 7], [7, 4, 6, 6, 1, 3], [7, 1, 3, 5, 6, 7], [7, 4, 6, 6, 1, 3], [9, 3, 4, 9, 1, 7], [2, 4, 4, 6, 7, 9], [2, 3, 4, 6, 3, 9], [2, 4, 4, 5, 9, 1]]

GENERASI KE- 9
[[7, 1, 3, 5, 6, 7], [9, 1, 8, 1, 8, 9], [2, 3, 9, 6, 2, 3], [8, 6, 6, 5, 6, 3], [2, 4, 4, 6, 7, 9], [8, 1, 3, 7, 7, 6], [6, 0, 1, 9, 1, 0], [9, 3, 4, 9, 1, 7], [1, 2, 8, 5, 8, 9], [9, 1, 8, 1, 8, 9], [2, 3, 9, 5, 9, 1], [8, 6, 6, 6, 2, 3], [8, 6, 6, 5, 6, 7], [7, 4, 6, 6, 1, 3], [7, 1, 3, 5, 6, 7], [7, 4, 6, 6, 1, 3], [9, 3, 4, 9, 1, 7], [2, 4, 4, 6, 7, 9], [2, 3, 4, 6, 3, 9], [2, 4, 4, 5, 9, 1]]

GENERASI KE- 10
[[7, 1, 3, 5, 6, 7], [9, 1, 8, 1, 8, 9], [2, 3, 9, 6, 2, 3], [8, 6, 6, 5, 6, 3], [2, 4, 4, 6, 7, 9], [8, 1, 3, 7, 7, 6], [6, 0, 1, 9, 1, 0], [9, 3, 4, 9, 1, 7], [1, 2, 8, 5, 8, 9], [9, 1, 8, 1, 8, 9], [2, 3, 9, 5, 9, 1], [8, 6, 6, 6, 2, 3], [8, 6, 6, 5, 6, 7], [7, 4, 6, 6, 1, 3], [7, 1, 3, 5, 6, 7], [7, 4, 6, 6, 1, 3], [2, 3, 4, 6, 3, 9], [2, 4, 4, 5, 9, 1], [8, 5, 6, 5, 6, 5], [7, 1, 3, 6, 2, 3]]

Best Kromosom : [1, 2, 8, 5, 8, 9]
Best Fitness : 97.80788290712952
X = -3.7187187187187187
Y = 0.8958958958958956
PS C:\Users\ASUS>
```



### 3. Pengujian 3

Untuk pengujian selanjutnya, kami membuat jumlah kromosom pada populasi dan jumlah regenerasi sama dengan pengujian awal. Namun, untuk panjang kromosomnya kami tambahkan 2 kali lebih besar dari sebelumnya. Sehingga, didapatkan nilai fitness yang terbaik yaitu 98,87.

```
File Edit Selection View Go Run Terminal Help Tupro1_BismillahFix.py - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\ASUS> python -u "c:\Users\ASUS\Downloads\Tupro1_BismillahFix.py"
Jumlah Populasi: 10
Panjang kromosom: 12
Jumlah Generasi: 10

GENERASI KE- 1
[[6, 9, 8, 5, 2, 2, 4, 3, 8, 3, 7, 6], [2, 8, 4, 8, 6, 1, 1, 0, 8, 0, 5, 8], [8, 4, 0, 5, 4, 0, 3, 5, 6, 4, 1, 7], [6, 1, 5, 2, 2, 9, 0, 3, 2, 1, 9, 1], [3, 0, 9, 8, 7, 2, 2, 3, 0, 7, 8, 7], [7, 1, 1, 3, 6, 6, 0, 4, 6, 4, 2, 1], [4, 5, 0, 8, 6, 2, 7, 2, 4, 7, 7, 5], [9, 7, 6, 4, 5, 0, 0, 6, 5, 1, 0, 2], [7, 2, 8, 0, 5, 2, 0, 1, 6, 1, 4, 7], [2, 0, 2, 5, 6, 4, 7, 6, 1, 1, 4, 7]]

GENERASI KE- 2
[[6, 9, 8, 5, 2, 2, 4, 3, 8, 3, 7, 6], [2, 8, 4, 8, 6, 1, 1, 0, 8, 0, 5, 8], [6, 1, 5, 2, 2, 9, 0, 3, 2, 1, 9, 1], [3, 0, 9, 8, 7, 2, 2, 3, 0, 7, 8, 7], [7, 1, 1, 3, 6, 6, 0, 4, 6, 4, 2, 1], [9, 7, 6, 4, 5, 0, 0, 6, 5, 1, 0, 2], [7, 2, 8, 0, 5, 2, 0, 1, 6, 1, 4, 7], [2, 0, 2, 5, 6, 4, 7, 6, 1, 1, 4, 7], [7, 1, 1, 3, 6, 6, 0, 6, 5, 3, 0, 2], [9, 7, 6, 4, 5, 0, 0, 4, 6, 4, 2, 1]]

GENERASI KE- 3
[[2, 8, 4, 8, 6, 1, 1, 0, 8, 0, 5, 8], [6, 1, 5, 2, 2, 9, 0, 3, 2, 1, 9, 1], [7, 1, 1, 3, 6, 6, 0, 4, 6, 4, 2, 1], [9, 7, 6, 4, 5, 0, 0, 6, 5, 1, 0, 2], [7, 2, 8, 0, 5, 2, 0, 1, 6, 1, 4, 7], [2, 0, 2, 5, 6, 4, 7, 6, 1, 1, 4, 7], [7, 1, 1, 3, 6, 6, 0, 6, 5, 3, 0, 2], [9, 7, 6, 4, 5, 0, 0, 4, 6, 4, 2, 1], [2, 8, 4, 3, 6, 1, 0, 6, 5, 3, 0, 2], [7, 1, 1, 3, 6, 6, 1, 0, 8, 0, 5, 6]]

GENERASI KE- 4
[[2, 8, 4, 4, 6, 2, 1, 1, 8, 0, 1, 8], [7, 1, 1, 3, 6, 6, 0, 4, 6, 4, 2, 1], [9, 7, 6, 4, 5, 0, 0, 6, 5, 1, 0, 2], [7, 2, 8, 0, 5, 2, 0, 1, 6, 1, 4, 7], [2, 0, 2, 5, 6, 4, 7, 6, 1, 1, 4, 7], [7, 1, 1, 3, 6, 6, 0, 6, 5, 3, 0, 2], [9, 7, 6, 4, 5, 0, 0, 4, 6, 4, 2, 1], [7, 1, 1, 3, 6, 6, 1, 0, 8, 0, 5, 6], [2, 8, 4, 4, 6, 2, 1, 1, 8, 0, 1, 8], [2, 0, 4, 4, 6, 2, 1, 1, 8, 0, 1, 8]]

GENERASI KE- 5
[[2, 8, 4, 4, 6, 2, 1, 1, 8, 0, 1, 8], [7, 1, 1, 3, 6, 6, 0, 4, 6, 4, 2, 1], [7, 2, 8, 0, 5, 2, 0, 1, 6, 1, 4, 7], [2, 0, 2, 5, 6, 4, 7, 6, 1, 1, 4, 7], [7, 1, 1, 3, 6, 6, 0, 6, 5, 3, 0, 2], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6], [2, 8, 4, 4, 6, 2, 1, 1, 8, 0, 1, 8], [2, 8, 4, 4, 6, 2, 1, 1, 8, 0, 1, 8], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6]]

GENERASI KE- 6
[[2, 8, 1, 2, 6, 2, 1, 1, 8, 0, 1, 8], [7, 2, 8, 0, 5, 2, 0, 1, 6, 1, 4, 7], [7, 1, 1, 3, 6, 6, 0, 6, 5, 3, 0, 2], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6], [2, 8, 1, 2, 6, 2, 1, 1, 8, 0, 1, 8], [2, 8, 1, 2, 6, 2, 1, 1, 8, 0, 1, 8], [2, 8, 1, 2, 6, 2, 1, 1, 8, 0, 1, 8], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6], [2, 8, 1, 2, 6, 2, 1, 1, 8, 0, 1, 8], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6]]

GENERASI KE- 7
[[2, 8, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6], [2, 8, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [2, 8, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6], [2, 8, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6], [2, 8, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6]]

GENERASI KE- 8
[[2, 8, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6], [2, 8, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [2, 8, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6], [2, 8, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6], [2, 8, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6]]

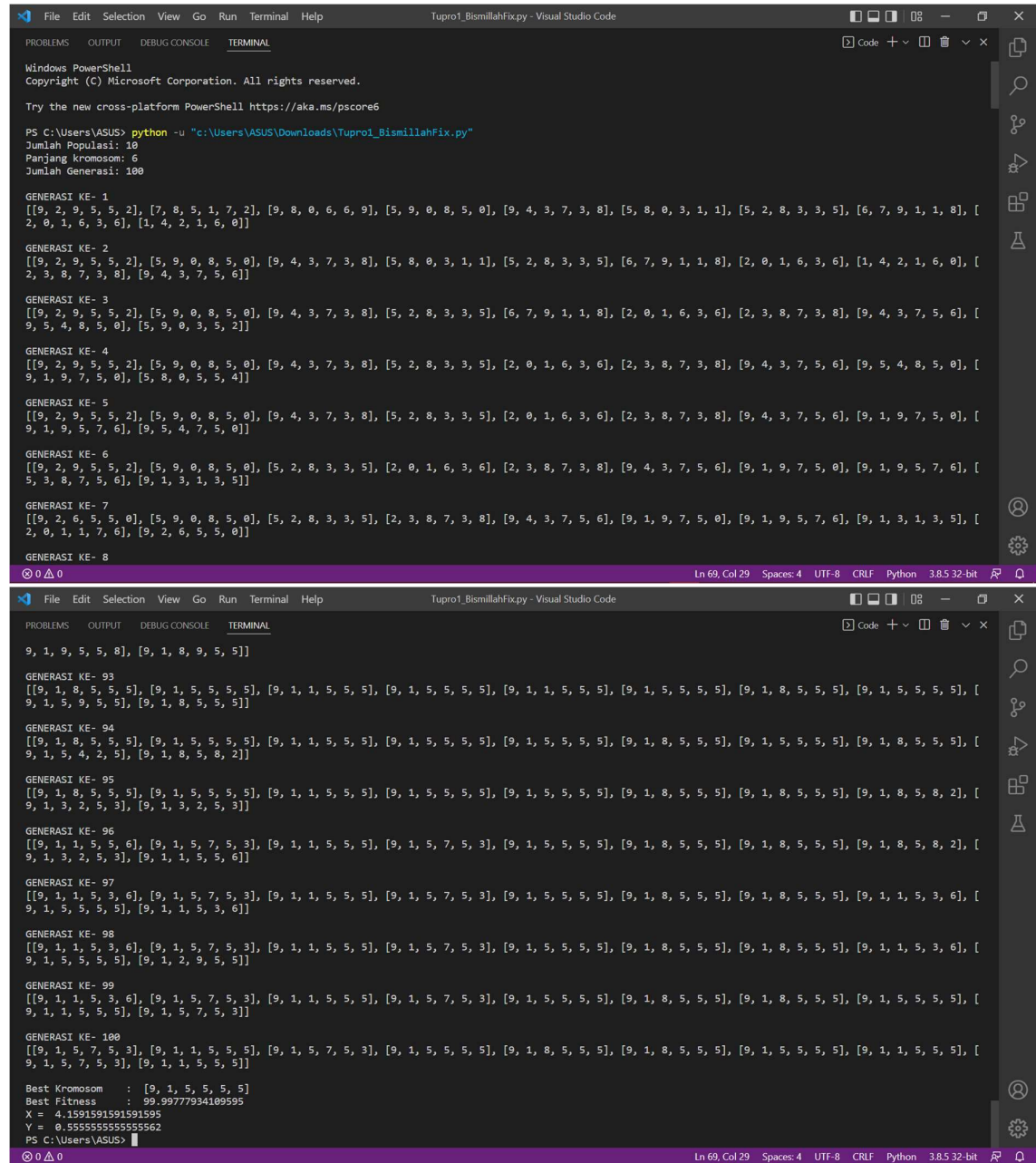
GENERASI KE- 9
[[2, 8, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6], [2, 8, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [2, 8, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6], [2, 8, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6], [2, 8, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6]]

GENERASI KE- 10
[[2, 4, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6], [2, 4, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [2, 4, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [7, 1, 1, 4, 6, 6, 1, 0, 8, 0, 5, 6], [2, 8, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0], [2, 8, 0, 9, 6, 5, 1, 1, 8, 0, 0, 0], [2, 8, 0, 9, 6, 5, 1, 1, 8, 0, 0, 0], [2, 4, 7, 2, 6, 2, 1, 1, 8, 0, 1, 0]]

Best Kromosom : [2, 8, 0, 9, 6, 5, 1, 1, 8, 0, 0, 0]
Best Fitness : 98.87910831948308
X = -2.1903471903471905
Y = -3.81999881999882
PS C:\Users\ASUS>
```

## 4. Pengujian 4

Pada pengujian terakhir, kami membuat jumlah generasi 10 kali lebih besar dari sebelumnya dengan jumlah kromosom dalam populasi dan panjang nya masih sama. Setelah program dijalankan, kami mendapatkan nilai fitness terbaik yaitu 99,99.



```
Windows PowerShell
Copyright (c) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\ASUS> python -u "c:\Users\ASUS\Downloads\Tuprot1_BismillahFix.py"
Jumlah Populasi: 10
Panjang kromosom: 6
Jumlah Generasi: 100

GENERASI KE- 1
[[9, 2, 9, 5, 5, 2], [7, 8, 5, 1, 7, 2], [9, 8, 0, 6, 6, 9], [5, 9, 0, 8, 5, 0], [9, 4, 3, 7, 3, 8], [5, 8, 0, 3, 1, 1], [5, 2, 8, 3, 3, 5], [6, 7, 9, 1, 1, 8], [2, 0, 1, 6, 3, 6], [1, 4, 2, 1, 6, 0]]

GENERASI KE- 2
[[9, 2, 9, 5, 5, 2], [5, 9, 0, 8, 5, 0], [9, 4, 3, 7, 3, 8], [5, 8, 0, 3, 1, 1], [5, 2, 8, 3, 3, 5], [6, 7, 9, 1, 1, 8], [2, 0, 1, 6, 3, 6], [1, 4, 2, 1, 6, 0], [2, 3, 8, 7, 3, 8], [9, 4, 3, 7, 5, 6]]

GENERASI KE- 3
[[9, 2, 9, 5, 5, 2], [5, 9, 0, 8, 5, 0], [9, 4, 3, 7, 3, 8], [5, 2, 8, 3, 3, 5], [6, 7, 9, 1, 1, 8], [2, 0, 1, 6, 3, 6], [2, 3, 8, 7, 3, 8], [9, 4, 3, 7, 5, 6], [9, 5, 4, 8, 5, 0], [5, 9, 0, 3, 5, 2]]

GENERASI KE- 4
[[9, 2, 9, 5, 5, 2], [5, 9, 0, 8, 5, 0], [9, 4, 3, 7, 3, 8], [5, 2, 8, 3, 3, 5], [2, 0, 1, 6, 3, 6], [2, 3, 8, 7, 3, 8], [9, 4, 3, 7, 5, 6], [9, 5, 4, 8, 5, 0], [9, 1, 9, 7, 5, 0], [5, 8, 0, 5, 5, 4]]

GENERASI KE- 5
[[9, 2, 9, 5, 5, 2], [5, 9, 0, 8, 5, 0], [9, 4, 3, 7, 3, 8], [5, 2, 8, 3, 3, 5], [2, 0, 1, 6, 3, 6], [2, 3, 8, 7, 3, 8], [9, 4, 3, 7, 5, 6], [9, 1, 9, 7, 5, 0], [9, 1, 9, 5, 7, 6], [9, 5, 4, 7, 5, 0]]

GENERASI KE- 6
[[9, 2, 9, 5, 5, 2], [5, 9, 0, 8, 5, 0], [5, 2, 8, 3, 3, 5], [2, 0, 1, 6, 3, 6], [2, 3, 8, 7, 3, 8], [9, 4, 3, 7, 5, 6], [9, 1, 9, 7, 5, 0], [9, 1, 9, 5, 7, 6], [5, 3, 8, 7, 5, 6], [9, 1, 3, 1, 3, 5]]

GENERASI KE- 7
[[9, 2, 6, 5, 5, 0], [5, 9, 0, 8, 5, 0], [5, 2, 8, 3, 3, 5], [2, 3, 8, 7, 3, 8], [9, 4, 3, 7, 5, 6], [9, 1, 9, 7, 5, 0], [9, 1, 9, 5, 7, 6], [9, 1, 3, 1, 3, 5], [2, 0, 1, 1, 7, 6], [9, 2, 6, 5, 5, 0]]

GENERASI KE- 8
9, 1, 9, 5, 5, 8], [9, 1, 8, 9, 5, 5]]

GENERASI KE- 93
[[9, 1, 8, 5, 5, 5], [9, 1, 5, 5, 5, 5], [9, 1, 1, 5, 5, 5], [9, 1, 5, 5, 5, 5], [9, 1, 1, 5, 5, 5], [9, 1, 5, 5, 5, 5], [9, 1, 8, 5, 5, 5], [9, 1, 5, 5, 5, 5], [9, 1, 5, 9, 5, 5], [9, 1, 8, 5, 5, 5]]

GENERASI KE- 94
[[9, 1, 8, 5, 5, 5], [9, 1, 5, 5, 5, 5], [9, 1, 1, 5, 5, 5], [9, 1, 5, 5, 5, 5], [9, 1, 5, 5, 5, 5], [9, 1, 8, 5, 5, 5], [9, 1, 5, 5, 5, 5], [9, 1, 8, 5, 5, 5], [9, 1, 5, 4, 2, 5], [9, 1, 8, 5, 8, 2]]

GENERASI KE- 95
[[9, 1, 8, 5, 5, 5], [9, 1, 5, 5, 5, 5], [9, 1, 1, 5, 5, 5], [9, 1, 5, 5, 5, 5], [9, 1, 5, 5, 5, 5], [9, 1, 8, 5, 5, 5], [9, 1, 8, 5, 5, 5], [9, 1, 8, 5, 8, 2], [9, 1, 3, 2, 5, 3], [9, 1, 3, 2, 5, 3]]

GENERASI KE- 96
[[9, 1, 1, 5, 5, 6], [9, 1, 5, 7, 5, 3], [9, 1, 1, 5, 5, 5], [9, 1, 5, 7, 5, 3], [9, 1, 5, 5, 5, 5], [9, 1, 8, 5, 5, 5], [9, 1, 8, 5, 5, 5], [9, 1, 8, 5, 8, 2], [9, 1, 3, 2, 5, 3], [9, 1, 1, 5, 5, 6]]

GENERASI KE- 97
[[9, 1, 1, 5, 3, 6], [9, 1, 5, 7, 5, 3], [9, 1, 1, 5, 5, 5], [9, 1, 5, 7, 5, 3], [9, 1, 5, 5, 5, 5], [9, 1, 8, 5, 5, 5], [9, 1, 8, 5, 5, 5], [9, 1, 1, 5, 3, 6], [9, 1, 5, 5, 5, 5], [9, 1, 1, 5, 3, 6]]

GENERASI KE- 98
[[9, 1, 1, 5, 3, 6], [9, 1, 5, 7, 5, 3], [9, 1, 1, 5, 5, 5], [9, 1, 5, 7, 5, 3], [9, 1, 5, 5, 5, 5], [9, 1, 8, 5, 5, 5], [9, 1, 8, 5, 5, 5], [9, 1, 1, 5, 3, 6], [9, 1, 5, 5, 5, 5], [9, 1, 2, 9, 5, 5]]

GENERASI KE- 99
[[9, 1, 1, 5, 3, 6], [9, 1, 5, 7, 5, 3], [9, 1, 1, 5, 5, 5], [9, 1, 5, 7, 5, 3], [9, 1, 5, 5, 5, 5], [9, 1, 8, 5, 5, 5], [9, 1, 8, 5, 5, 5], [9, 1, 5, 5, 5, 5], [9, 1, 1, 5, 5, 5], [9, 1, 5, 7, 5, 3]]

GENERASI KE- 100
[[9, 1, 5, 7, 5, 3], [9, 1, 1, 5, 5, 5], [9, 1, 5, 7, 5, 3], [9, 1, 5, 5, 5, 5], [9, 1, 8, 5, 5, 5], [9, 1, 8, 5, 5, 5], [9, 1, 5, 5, 5, 5], [9, 1, 1, 5, 5, 5], [9, 1, 5, 7, 5, 3], [9, 1, 1, 5, 5, 5]]

Best Kromosom : [9, 1, 5, 5, 5, 5]
Best Fitness : 99.99777934109595
X = 4.1591591591591595
Y = 0.5555555555555556
PS C:\Users\ASUS>
```

#### IV. KESIMPULAN

Dari percobaan *running program* diatas, dapat terlihat perbedaan dari tiap percobaan. Percobaan 1 merupakan titik acuan untuk tiap percobaan lainnya. Percobaan 2 memiliki jumlah populasi lebih banyak dibandingkan percobaan 1, percobaan 3 memiliki panjang kromosom/ jumlah gen lebih banyak dibandingkan percobaan 1, dan percobaan 4 memiliki jumlah regenerasi lebih banyak dibandingkan percobaan 1.

Hasil percobaan *running program* diatas menunjukkan bahwa nilai fitness dari percobaan 2, percobaan 3, dan percobaan 4 lebih besar dibandingkan percobaan 1. Hal ini menandakan bahwa Jumlah populasi, panjang kromosom/ jumlah gen, dan jumlah regenerasi akan berbanding lurus dengan nilai fitness.

Penggunaan metode pemilihan orangtua, metode *cross over* (pindah silang), metode mutasi, metode pergantian seleksi (seleksi *survivor*) yang tepat ; probabilitas mutasi dan probabilitas rekombinasi yang semakin kecil ; serta jumlah populasi, panjang kromosom/jumlah gen, dan jumlah regenerasi yang semakin banyak, maka akan semakin besar kemungkinan untuk mendapatkan hasil yang optimal.

## **V. PEMBAGIAN JOBDESK**

Famardi Putra Muhammad Raffly :

Proses pemrograman

- Membuat fungsi (function)
- Membuat fungsi (individu)
- Membuat fungsi (populasi)
- Membuat fungsi (split)
- Membuat fungsi (decode)
- Membuat fungsi (hitung fitness)
- Membuat main program
- Berperan dalam mengatasi eror dan merapikan code

Pembuatan laporan

- Membuat pendahuluan
- Membuat desain kromosom dan metode decode
- Membuat fungsi fitness
- Membuat seleksi orangtua
- Probabilitas operasi Genetik
- Running program

Muhammad Daffa' Ibrahim:

Proses pemrograman

- Membuat fungsi (seleksi orangtua)
- Membuat fungsi (crossover)
- Membuat fungsi (mutasi)
- Membuat fungsi (regenerasi)
- Membuat fungsi (best kromosom)
- Berperan dalam mengonsep dan menerapkan alur logika yang dibutuhkan dalam code

Pembuatan Laporan

- Membuat Ukuran Populasi
- Membuat Metode Operasi Genetik (pindah silang dan mutasi)
- Membuat Metode Pergantian Generasi (Seleksi Survivor)
- Membuat Kriteria penghentian evolusi (loop)
- Menentukan jumlah populasi, panjang kromosom, dan jumlah generasi dalam running program
- Membuat Kesimpulan

**Link Video :**

[https://drive.google.com/file/d/1oU5x3uI\\_4XvrWrWIn5ZQFuk3xdynoLxW/view?usp=sharing](https://drive.google.com/file/d/1oU5x3uI_4XvrWrWIn5ZQFuk3xdynoLxW/view?usp=sharing)

**Referensi :**

Suharjito.(2016). *Algoritma Genetika dengan Python*. Diakses pada 2 April 2022, dari <https://onlinelearning.binus.ac.id/computer-science/post/algoritma-genetika-dengan-python>