CII2M3 INTRODUCTION OF ARTIFICIAL INTELLIGENCE

EVEN SEMESTER SESSION 2021/2022

ASSIGNMENT 1 - SEARCHING

Group No : 10

Section: IF-44-INT

Lecturer Name: Edward Ferdian

Group Member:

NAME	STUDENT ID
ZIDAN RIZKY WIJAYA	1301200226
NUR SABRINA SYAZA BINTI ZAHAR HISHAM	1301213666
NURUL IMAN BINTI NORDIN	1301213669

TABLE OF CONTENT

Youtube link: https://youtu.be/yMUr7Zo5LHc

Readme link:

https://github.com/Sabrinasyaza/ProgrammingAssignment01/blob/main/README.md

QUESTION:

1. Analyze and design ***Genetic Algorithm (GA)*** and implement a program to find x and y values to obtain the **minimum value** from the following function:

$$h(x,y) = \frac{(\cos x + \sin y)^2}{x^2 + y^2}$$

With the following **domain** for x and y:

$$-5 \leq x \leq 5 \text{ dan } -5 \leq y \leq 5$$

Points to **analyze** and **design**:

- Design chromosome and the decode method.
- Population size
- Parent selection method
- Genetic operation methods (crossover and mutation)
- Genetic operation probability P_c and P_m
- Generation replacement method (survivor selection)
- Evolution stopping criteria (loop)

What you need to **implement/develop** in your program:

- Chromosome decoding
- Fitness function
- Parent selection
- Crossover
- Mutation
- Generation replacement

2. **Program Output**

   From the problem defined previously, the **output** from your program should be:

   - **The best chromosome**, and
   - **x and y** value from the best decoded chromosome.

1. PROBLEM DESCRIPTION

Genetic Algorithm (GA) is a metaheuristic search method that has been used in Artificial Intelligence. Genetic Algorithm is usually used to identify the optimal solutions of search problems. This algorithm reflects the method of a selection where the most fit individuals are selected for reproduction in order to produce new offspring of the new generation.

There are five phases that are considered in genetic algorithm which are:

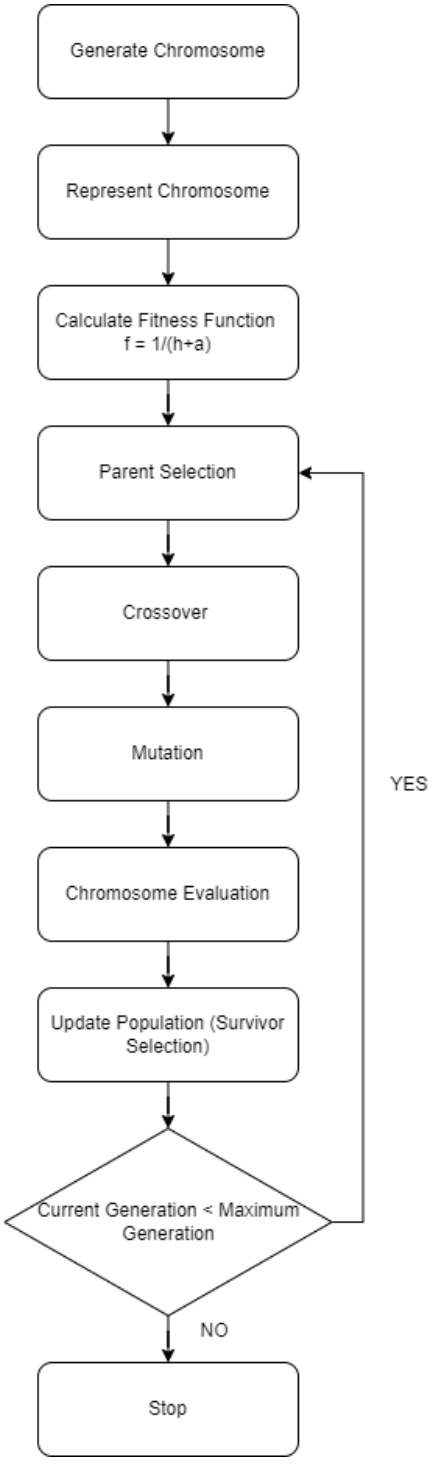
- Initial population
- Fitness function
- Selection
- Crossover
- Mutation

We need to analyze and design Genetic Algorithm (GA) and implement a program to find x dan y values to obtain the minimum value from the following function:

$$h(x, y) = \frac{(\cos x + \sin y)^2}{x^2 + y^2}$$

with the following domain for x and y: $-5 \leq x \leq 5$ dan $-5 \leq y \leq 5$.

This is the flowchart of the Genetic Algorithm that we have used.



2. DESIGN CHROMOSOME AND THE DECODE METHOD (BINARY REPRESENTATION)

Genes are a set of factors (variables) that describe a person. A Chromosome is made up of a string of genes (solution).

The set of genes of a person is represented as a string in terms of an alphabet in a genetic algorithm. The binary values have been used in our coding which are string 0s and 1s. We encoded the genes in a chromosome.

```
class Chromosome:
    def __init__(self, binary = None):
        if binary == None:
            self.binary = random.choices([0, 1], k=10)
        else:
            self.binary = binary
        self.x = self.Representation(dmin_x, dmax_x, self.binary[:5])
        self.y = self.Representation(dmin_y, dmax_y, self.binary[5:])

    def __repr__(self):
        return '{} (x : {}, y : {}) Heuristic: {}'.format(self.binary, self.x, self.y, heuristic(self.x, self.y))

    def Representation(self, dmax, dmin, g):
        br = [2**-i for i in range(1, len(g) + 1)]
        return dmin + ((dmax - dmin) / sum(br) * sum([g[i] * br[i] for i in range(len(g))]))
```

3. POPULATION SIZE

Population size is an important parameter in genetic algorithms since it directly affects the ability to find the best solution in the search space. Many studies have proven that achieving optimal solutions with a big population is more accurate. In our coding we set up the population size which is 40.

```
while len(population) != 40:
    ch = Chromosome()

    if not exist(population, ch):
        population.append(ch)
```

4. PARENT SELECTION METHOD

Parent selection is the process of selecting parents or individuals to generate the new offspring of the new generation. Parental selection is very important to the rate of convergence of Genetic Algorithms, as good parents lead individuals to better and more appropriate solutions. In our coding, we used the **Roulette Wheel Selection method** to choose parents based on its fitness. We have chosen the Roulette Wheel Selection method to select two individuals randomly in order to generate new offspring for the new generation. The higher the fitness of an individual (better chromosomes), the higher the chances of getting selected.

```
def Parent_selection(k):
    parents = []
    fitnesses = list(map(lambda p: fitness(p.x, p.y), population))
    weight = [fitnesses[i] / sum(fitnesses)
                  for i in range(len(population))]

    while len(parents) != k:
        select = random.choices(population, weights=weight)[0]
        if not exist(parents, select):
            parents.append(select)
    return parents
```

5. GENETIC OPERATION METHODS (CROSSOVER AND MUTATION)

Crossover or recombination is a genetic operator that used to change the programming of one or more chromosomes from generation to generation in order to generate new offspring. The crossover process will match the random good parents' to generate better offspring. The method that has been chosen depends on the encoding method. The crossover process usually occurs once the probability has been set. Crossover has general schemes such as Single-Point, Two-Point, n-point crossover, Uniform crossover, Arithmetic crossover etc. In our program we used the Single-Point method which will select a point randomly to generate new offsprings.

Mutation is a part of the Genetic Algorithm that is associated with the searching process in the search space. The function of mutation is to simulate the effect of errors that happen during duplication. Mutation has been observed to be essential for Genetic Algorithm convergence. Because we used the binary encoding method in chromosome in our coding, Bit Flip Mutation has been used which is we chose the random bits between 0 to 100 probability mutation in 0.5% mutation, if the bits is 0 it will be inverted to 1 and vice versa. After that, we put the crossover and mutation results to the population.

```
def Crossover(prnt1, prnt2):
    position = random.randint(1, len(prnt1.binary) - 2)

    child1 = prnt1.binary[:position] + prnt2.binary[position:]
    child2 = prnt2.binary[:position] + prnt1.binary[position:]

    prob = random.uniform(0, 100)
    if prob < 0.5:
        mutated = random.randint(0, len(child1) - 1)
        if  child1[mutated] == 1:
            child1[mutated]
        else:
            child1[mutated] = 1

    prob = random.uniform(0, 100)
    if prob < 0.5:
        mutated = random.randint(0, len(child2) - 1)
        if  child2[mutated] == 1:
            child2[mutated]
        else:
            child2[mutated] = 1

    population.append(Chromosome(child1))
    population.append(Chromosome(child2))
```

6. GENETIC OPERATION PROBABILITY (PC AND PM)

The genetic operation probability is a combination of crossover process and mutation process. The degree of the solution accuracy and convergence can be determined by the probabilities of crossover ($\mathbf{P_c}$) and mutation ($\mathbf{P_m}$). In Adaptive Genetic Algorithm, the probability of the crossover and mutation can be stated based on the fitness values while in Clustering-based Adaptive Algorithm, the probability of the crossover and the mutation can be determined according to the optimization stated. For example, in our program if the probability of the crossover is less than 0.5, the Genetic Algorithm process will generate a mutation of a child.

7. GENERATION REPLACEMENT METHOD (SURVIVOR SELECTION)

The Survivor Selection Policy selects who is to be thrown out of the next generation and who is to be preserved. It is important as it must guarantee that the fittest members are not thrown out of the population while also maintaining population variety. Genetic Algorithm employs a mechanism which is Elitism. Elitism is usually used in Generational Replacement and ensures the best individual survives which means the most fit individuals are guaranteed a place in the next generation. Elitism ensures the solution quality obtained will not decrease from one generation to the next generation. In our coding, key function and lambda function have been used as anonymous functions in sort descending function. The worst chromosome in the population needs to be thrown out.

```
def survivor_selection():
    population.sort(key=lambda p: heuristic(p.x, p.y), reverse=True)

    while len(population) != 50:
        population.pop()
```

8. EVOLUTION STOPPING CRITERIA (LOOP)

There are few stopping criteria to identify when the looping will stop. The first criteria is when the maximum number of generations have reached. Once the process reaches the maximum number of generations, the Genetic Algorithm will automatically stop the looping. For example, based on the program that our group had implemented, the Genetic Algorithm will stop once it reaches the maximum number of the generations that have been stated. Next, Genetic Algorithm also will stop once it detects the maximum time limits. Furthermore, the Genetic Algorithms process also stops if there are no changes in the fitness value for some time limit or generation limit. The Genetic Algorithm process also will terminate once the highest fitness has reached a plateau and no longer produce better results.

THE CODE

```
import random
import math
import matplotlib.pyplot as plt

dmin_x = -5
dmax_x = 5
dmin_y = -5
dmax_y = 5

class Chromosome:
    def __init__(self, binary = None):
        if binary == None:
            self.binary = random.choices([0, 1], k=10)
        else:
            self.binary = binary
        self.x = self.Representation(dmin_x, dmax_x, self.binary[:5])
        self.y = self.Representation(dmin_y, dmax_y, self.binary[5:])

    def __repr__(self):
        return '{} (x : {}, y : {}) Heuristic: {}'.format(self.binary, self.x, self.y, heuristic(self.x, self.y))

    def Representation(self, dmax, dmin, g):
        br = [2**-i for i in range(1, len(g) + 1)]
        return dmin + ((dmax - dmin) / sum(br) * sum([g[i] * br[i] for i in range(len(g))]))

def heuristic(x, y):
    return (((math.cos(x) + math.sin(y))*(math.cos(x) + math.sin(y)))/((x*x) + (y*y)))

def fitness(x, y): #minimum fitness function
    a = 0.0000000000000001
    return 1 / (heuristic (x,y) + a)
```

```
def exist(s, ch):
    check = False
    for i in s:
        if i.binary == ch.binary:
            check = True
    return check

def Parent_selection(k):
    parents = []
    fitnesses = list(map(lambda p: fitness(p.x, p.y), population))
    weight = [fitnesses[i] / sum(fitnesses)
                for i in range(len(population))]

    while len(parents) != k:
        select = random.choices(population, weights=weight)[0]
        if not exist(parents, select):
            parents.append(select)
    return parents

def Crossover(prnt1, prnt2):
    position = random.randint(1, len(prnt1.binary) - 2)

    child1 = prnt1.binary[:position] + prnt2.binary[position:]
    child2 = prnt2.binary[:position] + prnt1.binary[position:]

    prob = random.uniform(0, 100)
    if prob < 0.5:
        mutated = random.randint(0, len(child1) - 1)
        if  child1[mutated] == 1:
            child1[mutated]
        else:
            child1[mutated] = 1
```

```
    prob = random.uniform(0, 100)
    if prob < 0.5:
        mutated = random.randint(0, len(child2) - 1)
        if  child2[mutated] == 1:
            child2[mutated]
        else:
            child2[mutated] = 1

    population.append(Chromosome(child1))
    population.append(Chromosome(child2))

def survivor_selection():
    population.sort(key=lambda p: heuristic(p.x, p.y), reverse=True)

    while len(population) != 40:
        population.pop()

population = []
chromosome = []
generation = 1

while len(population) != 40:
    ch = Chromosome()

    if not exist(population, ch):
        population.append(ch)

survivor_selection()
print('Generation', generation)
print('The Best Chromosome', population[0])

arr_fitness = [0]*130
arr_fitness[generation-1] = fitness(population[0].x, population[0].y)
```

```
while generation < 130:
    parent = Parent_selection(2)
    Crossover(parent[0], parent[1])
    survivor_selection()

    generation += 1
    arr_fitness[generation-1] = fitness(population[0].x, population[0].y)
    print('Generation', generation)
    print('The Best Chromosome', population[0])

plt.plot(range(1, generation + 1), arr_fitness)
plt.title("Graph of Fitness Performance")
plt.xlabel("Generation")
plt.ylabel("Best Fitness")
plt.show()
```

Program Output

```
Generation 1
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 2
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 3
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 4
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 5
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 6
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 7
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 8
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 9
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 10
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 11
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 12
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 13
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
Generation 14
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 15
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 16
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 17
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 18
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 19
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 20
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 21
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 22
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 23
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 24
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 25
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 26
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
```

```
Generation 27
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 28
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 29
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 30
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 31
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 32
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 34
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 35
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 36
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 37
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 38
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 39
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 40
Generation 40
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 41
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 42
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 43
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 44
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 45
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 46
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 48
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 49
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 50
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 51
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 52
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 53
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
```

```
Generation 54
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 55
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 56
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 57
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 58
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 59
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 60
```

```
Generation 60
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 61
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 62
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 63
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 64
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 65
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 66
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
```

```
Generation 69
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 70
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 71
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 72
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 73
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 74
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 75
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
```

```
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 81
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 82
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 83
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 84
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 85
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 86
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
```

```
39316
Generation 87
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 88
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 89
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 90
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 91
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 92
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 93
```

```
39316
Generation 94
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 95
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 96
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 97
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 98
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 99
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 100
```

```
39316
Generation 102
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 103
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 104
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 105
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 106
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 107
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 108
```

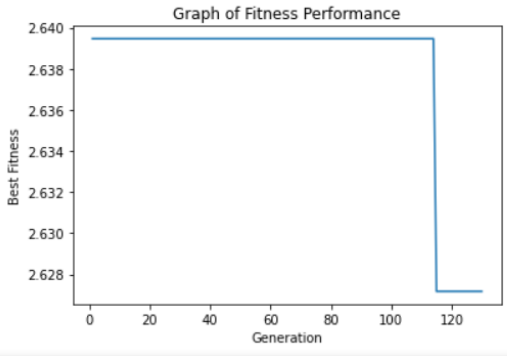
```
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 108
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 109
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 110
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 111
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 112
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 113
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
```

```
Generation 114
The Best Chromosome [1, 1, 0, 0, 0, 1, 0, 0, 1, 1] (x : -2.741935483870968, y : -1.129032258064516) Heuristic: 0.378864939343
39316
Generation 115
The Best Chromosome [0, 0, 1, 1, 1, 1, 0, 1, 0, 0] (x : 2.741935483870968, y : -1.451612903225806) Heuristic: 0.3806371122090
3623
Generation 116
The Best Chromosome [0, 0, 1, 1, 1, 1, 0, 1, 0, 0] (x : 2.741935483870968, y : -1.451612903225806) Heuristic: 0.3806371122090
3623
Generation 117
The Best Chromosome [0, 0, 1, 1, 1, 1, 0, 1, 0, 0] (x : 2.741935483870968, y : -1.451612903225806) Heuristic: 0.3806371122090
3623
Generation 118
The Best Chromosome [0, 0, 1, 1, 1, 1, 0, 1, 0, 0] (x : 2.741935483870968, y : -1.451612903225806) Heuristic: 0.3806371122090
3623
Generation 119
The Best Chromosome [0, 0, 1, 1, 1, 1, 0, 1, 0, 0] (x : 2.741935483870968, y : -1.451612903225806) Heuristic: 0.3806371122090
3623
Generation 120
```

```
3623
Generation 125
The Best Chromosome [0, 0, 1, 1, 1, 1, 0, 1, 0, 0] (x : 2.741935483870968, y : -1.451612903225806) Heuristic: 0.3806371122090
3623
Generation 126
The Best Chromosome [0, 0, 1, 1, 1, 1, 0, 1, 0, 0] (x : 2.741935483870968, y : -1.451612903225806) Heuristic: 0.3806371122090
3623
Generation 127
The Best Chromosome [0, 0, 1, 1, 1, 1, 0, 1, 0, 0] (x : 2.741935483870968, y : -1.451612903225806) Heuristic: 0.3806371122090
3623
Generation 128
The Best Chromosome [0, 0, 1, 1, 1, 1, 0, 1, 0, 0] (x : 2.741935483870968, y : -1.451612903225806) Heuristic: 0.3806371122090
3623
Generation 129
The Best Chromosome [0, 0, 1, 1, 1, 1, 0, 1, 0, 0] (x : 2.741935483870968, y : -1.451612903225806) Heuristic: 0.3806371122090
3623
Generation 130
The Best Chromosome [0, 0, 1, 1, 1, 1, 0, 1, 0, 0] (x : 2.741935483870968, y : -1.451612903225806) Heuristic: 0.3806371122090
3623
```

THE OUTPUT

```
The Best Chromosome [0, 0, 1, 1, 1, 1, 0, 1, 0, 0] (x : 2.741935483870968, y : -1.451612903225806) Heuristic: 0.3806371122090
3623
```



TEAM MEMBER CONTRIBUTION

NAME	TASKS
NUR SABRINA SYAZA BINTI ZAHAR HISHAM (1301213666)	- Program source code - Readme.txt - Report - Presentation
ZIDAN RIZKY WIJAYA (1301200226)	- Report - Presentation
NURUL IMAN BINTI NORDIN (1301213669)	- Program source code - Report - Compile presentation video - Presentation