



8-QUEEN SOLVER

USING

GENETIC ALGORITHM





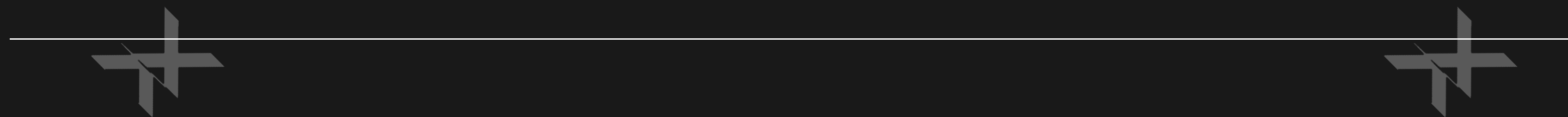
Andika Laksana Putra

5025211001



Muhammad Naufal Baihaqi

5025211103



GENETIC ALGORITHM



Populasi

Seleksi

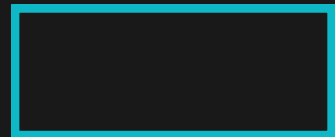
Crossover

mutasi

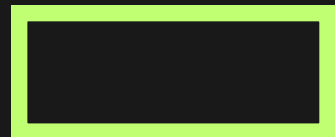
GEN – KROMOSOM – POPULASI



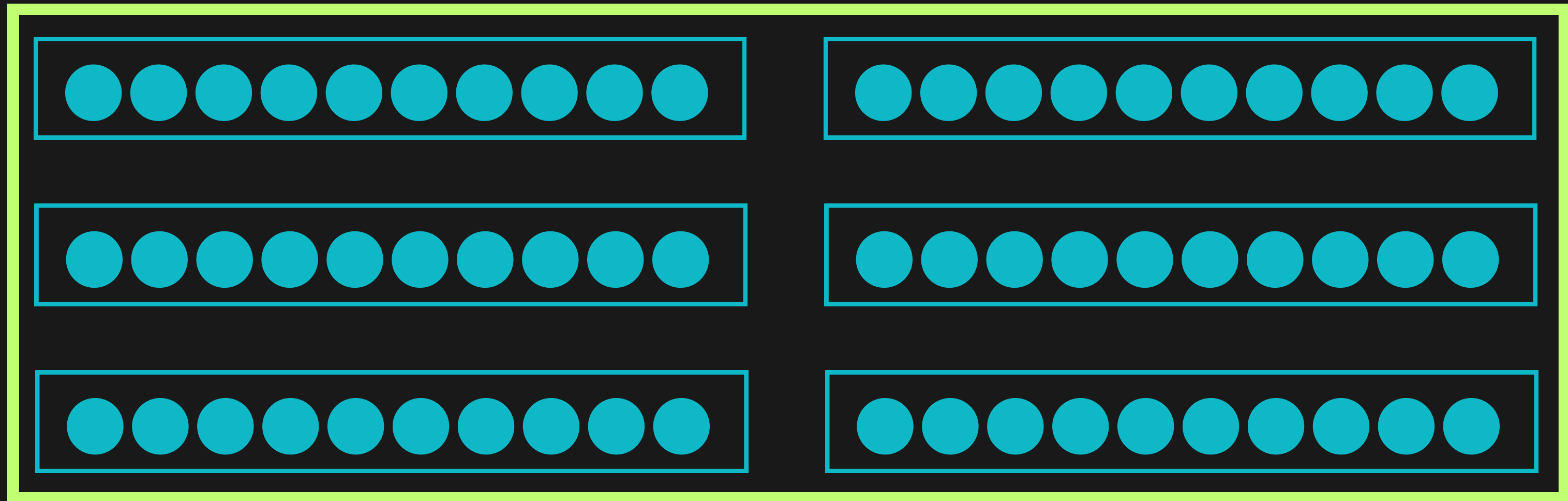
Gen



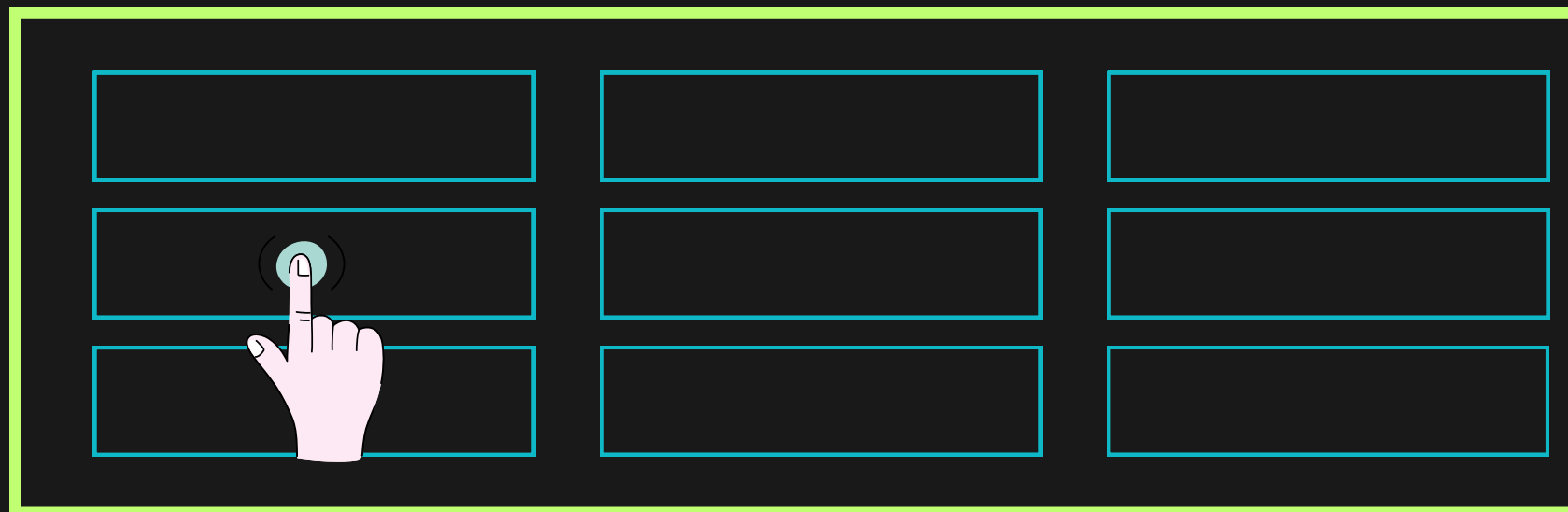
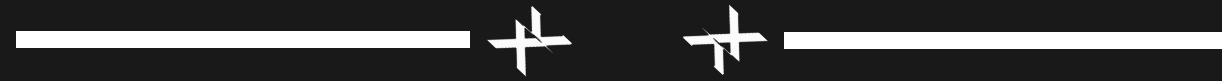
Kromosom



Populasi



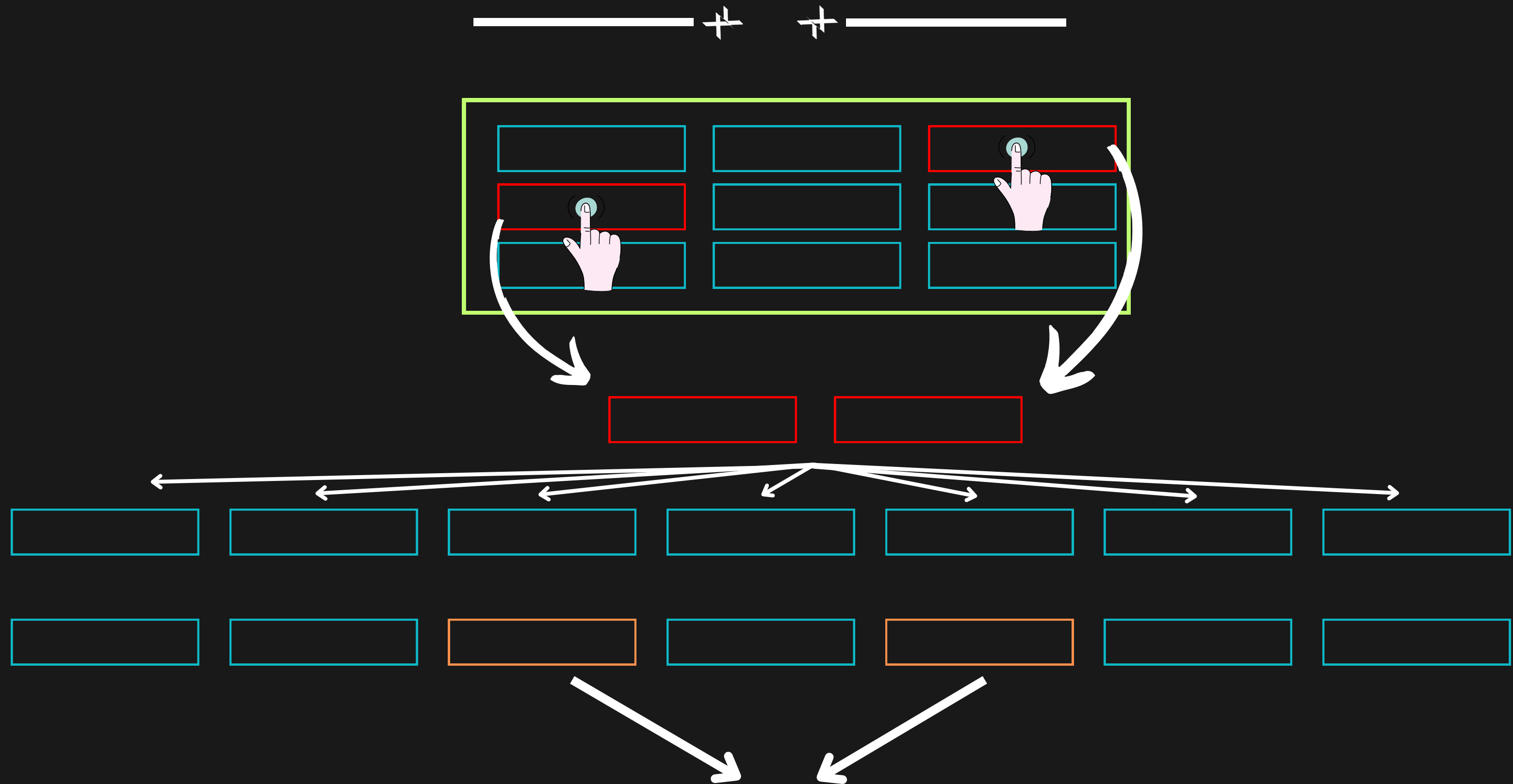
SELEKSI



Menyeleksi individu-individu /
kromosom-kromosom dalam
suatu populasi berdasarkan
fitness value nya

Fitness value merupakan ukuran
seberapa baik solusi tersebut
dalam menyelesaikan masalah
yang diberikan

CROSSOVER

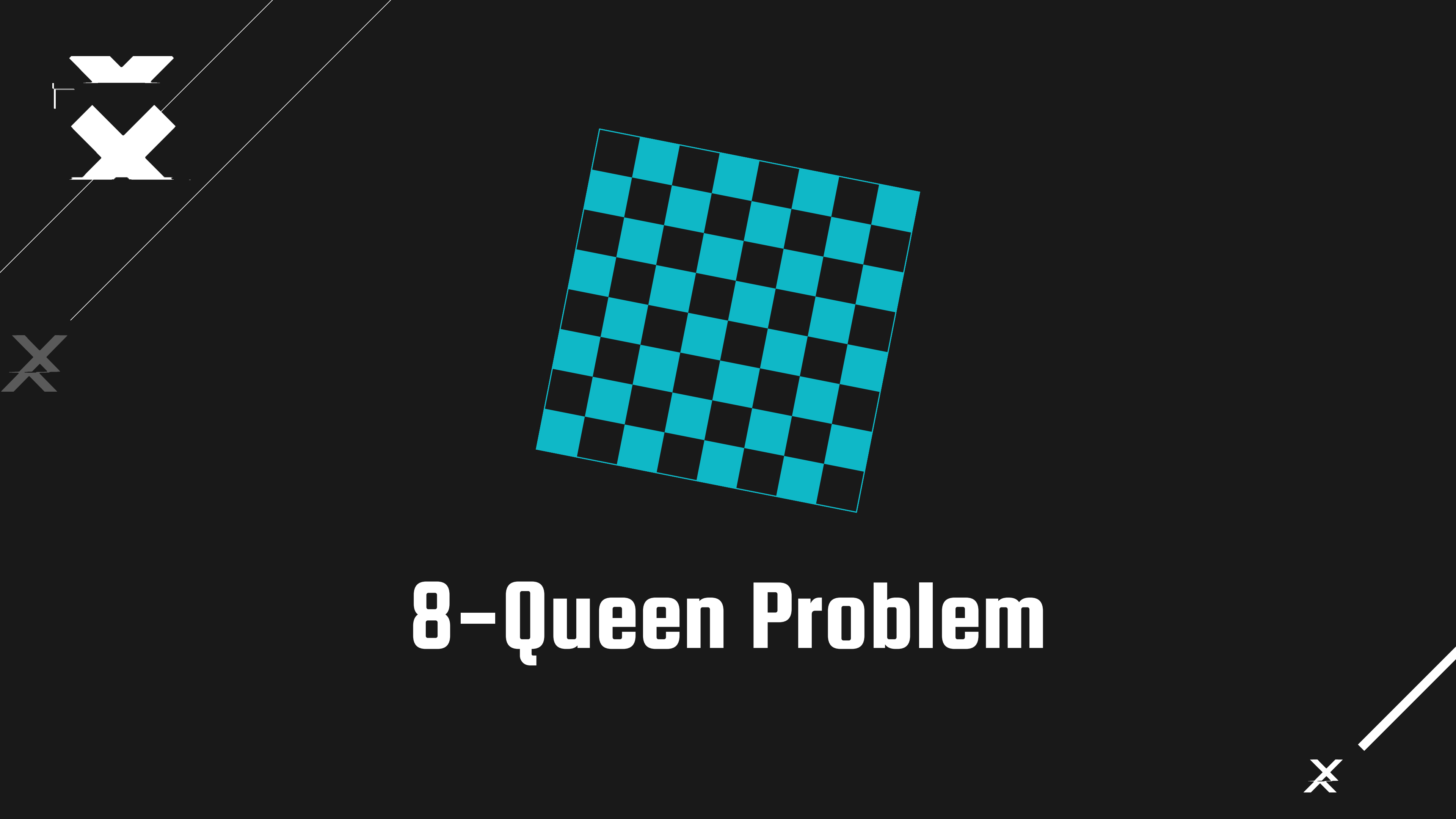


Dipilih secara acak untuk jadi parent selanjutnya

MUTASI



Pada proses mutasi, beberapa gen dalam kromosom dipilih secara acak dan nilainya diubah. Hal ini dilakukan agar populasi menjadi lebih beragam dan dapat mengeksplorasi solusi yang berbeda



8-Queen Problem


```

# Making random chromosomes
def random_chromosome(size):
    return [random.randint(0, size - 1) for _ in range(size)]

# Calculating fitness
def fitness(chromosome, maxFitness):
    horizontal_collisions = (
        sum([chromosome.count(queen) - 1 for queen in chromosome]) / 2
    )
    diagonal_collisions = 0

    n = len(chromosome)
    left_diagonal = [0] * (2 * n - 1)
    right_diagonal = [0] * (2 * n - 1)
    for i in range(n):
        left_diagonal[i + chromosome[i] - 1] += 1
        right_diagonal[len(chromosome) - i + chromosome[i] - 2] += 1

    diagonal_collisions = 0
    for i in range(2 * n - 1):
        counter = 0
        if left_diagonal[i] > 1:
            counter += left_diagonal[i] - 1
        if right_diagonal[i] > 1:
            counter += right_diagonal[i] - 1
        diagonal_collisions += counter

    # 28-(2+3)=23
    return int(maxFitness - (horizontal_collisions + diagonal_collisions))

```

Fungsi "random_chromosome" digunakan untuk menghasilkan kromosom acak dengan ukuran sebanyak size. Kromosom tersebut merepresentasikan susunan queen pada papan catur dengan ukuran size x size.


Selanjutnya, Fungsi "fitness" merupakan digunakan untuk menghitung nilai fitness dari kromosom chromosome terhadap nilai fitness maksimum yang mungkin maxFitness. Fungsi ini menghitung jumlah konflik antara pasangan queen pada papan catur

```
# Doing cross_over between two chromosomes
```


```
def crossover(x, y):  
    n = len(x)  
    child = [0] * n  
    for i in range(n):  
        c = random.randint(0, 1)  
        if c < 0.5:  
            child[i] = x[i]  
        else:  
            child[i] = y[i]  
    return child
```

```
# Randomly changing the value of a random index of a chromosome
```

```
def mutate(x):  
    n = len(x)  
    c = random.randint(0, n - 1)  
    m = random.randint(0, n - 1)  
    x[c] = m  
    return x
```

 Fungsi "crossover(x,y)" digunakan untuk melakukan operasi crossover antara dua kromosom (individu) yang dipilih dari populasi. Pada kasus ini, kromosom direpresentasikan sebagai array 1 dimensi, yang berisi nilai indeks dari kolom di mana ratu ditempatkan pada baris yang sesuai. Operasi crossover dilakukan dengan cara mengambil bagian awal kromosom x sampai dengan titik acak, kemudian ditambahkan bagian sisa kromosom yang diambil dari kromosom y.

Fungsi "mutate(x)" digunakan untuk mengembalikan kromosom baru yang diperoleh dengan mengubah nilai acak pada kromosom x.





```
# Calculating probability
def probability(chromosome, maxFitness):
    return fitness(chromosome, maxFitness) / maxFitness
```

```
# Roulette-wheel selection
def random_pick(population, probabilities):
    populationWithProbabilty = zip(population, probabilities)
    total = sum(w for c, w in populationWithProbabilty)
    r = random.uniform(0, total)
    upto = 0
    for c, w in zip(population, probabilities):
        if upto + w >= r:
            return c
        upto += w
    assert False, "Shouldn't get here"
```

Fungsi "probability" Fungsi ini digunakan untuk menghitung probabilitas `chromosome` terhadap nilai fitness maksimum yang mungkin.

Fungsi "random_pick" digunakan untuk memilih kromosom acak dari populasi berdasarkan probabilitas yang diberikan oleh `probabilities`.



```

def genetic_queen(population, maxFitness):
    mutation_probability = 0.1
    new_population = []
    sorted_population = []
    probabilities = []
    for n in population:
        f = fitness(n, maxFitness)
        probabilities.append(f / maxFitness)
        sorted_population.append([f, n])

    sorted_population.sort(reverse=True)

    # Elitism
    new_population.append(sorted_population[0][1]) # the best gen
    new_population.append(sorted_population[-1][1]) # the worst gen

    for i in range(len(population) - 2):

        chromosome_1 = random_pick(population, probabilities)
        chromosome_2 = random_pick(population, probabilities)

        # Creating two new chromosomes from 2 chromosomes
        child = crossover(chromosome_1, chromosome_2)

        # Mutation
        if random.random() < mutation_probability:
            child = mutate(child)

        new_population.append(child)
        if fitness(child, maxFitness) == maxFitness:
            break
    return new_population

```

Fungsi "genetic_queen" digunakan untuk menjalankan algoritma genetika untuk menyelesaikan problem N-Queens. Fungsi ini mengambil populasi kromosom acak population dan maksimum nilai fitness yang mungkin maxFitness sebagai argumen, kemudian mengembalikan populasi baru kromosom hasil operasi genetika.

```

# prints given chromosome
def print_chromosome(chrom, maxFitness):
    print(
        "Chromosome = {}, Fitness = {}".format(str(chrom), fitness(chrom, maxFitness))
    )

# prints given chromosome board
def print_board(chrom):
    board = []


    for x in range(nq):
        board.append(["x"] * nq)

    for i in range(nq):
        board[chrom[i]][i] = "Q"

    def print_board(board):
        for row in board:
            print(" ".join(row))

    print()
    print_board(board)

```


 Fungsi "print_chromosome" digunakan untuk mencetak kromosom `chrom` beserta nilai fitnessnya terhadap nilai fitness maksimum yang mungkin `maxFitness`.

Fungsi "print_board" digunakan untuk mencetak papan catur yang merepresentasikan kromosom chrom. Papan catur tersebut terdiri dari nq baris dan kolom, di mana setiap baris dan kolom merepresentasikan satu queen. Pada setiap queen ditandai dengan simbol "Q", sedangkan kotak-kotak kosong ditandai dengan simbol "x".



Output akan menampilkan berapa kali percobaan crossover dan mutate dari berbagai kromosom sehingga maximum fitnessnya mencapai nilai "28" lalu menampilkan Array fitness dan tampilan solusi 8 Queen.

OUTPUT

```
PS D:\Code\Python> python -u "d:\Code\Python\kelom.py"
Maximum Fitness = 26
=== Generation 30 ===
Maximum Fitness = 27
=== Generation 40 ===
Maximum Fitness = 27
=== Generation 50 ===
Maximum Fitness = 27
=== Generation 60 ===
Maximum Fitness = 27
=== Generation 70 ===
Maximum Fitness = 27
=== Generation 80 ===
Maximum Fitness = 27
=== Generation 90 ===
Maximum Fitness = 27
=== Generation 100 ===
Maximum Fitness = 27
=== Generation 110 ===
Maximum Fitness = 27
=== Generation 120 ===
Maximum Fitness = 27
=== Generation 130 ===
Maximum Fitness = 27

Solved in Generation 131!
Chromosome = [1, 6, 2, 5, 7, 4, 0, 3], Fitness = 28

x x x x x x Q x
Q x x x x x x x
x x Q x x x x x
x x x x x x x Q
x x x x x Q x x
x x x Q x x x x
x Q x x x x x x
x x x x Q x x x
```

THANK YOU