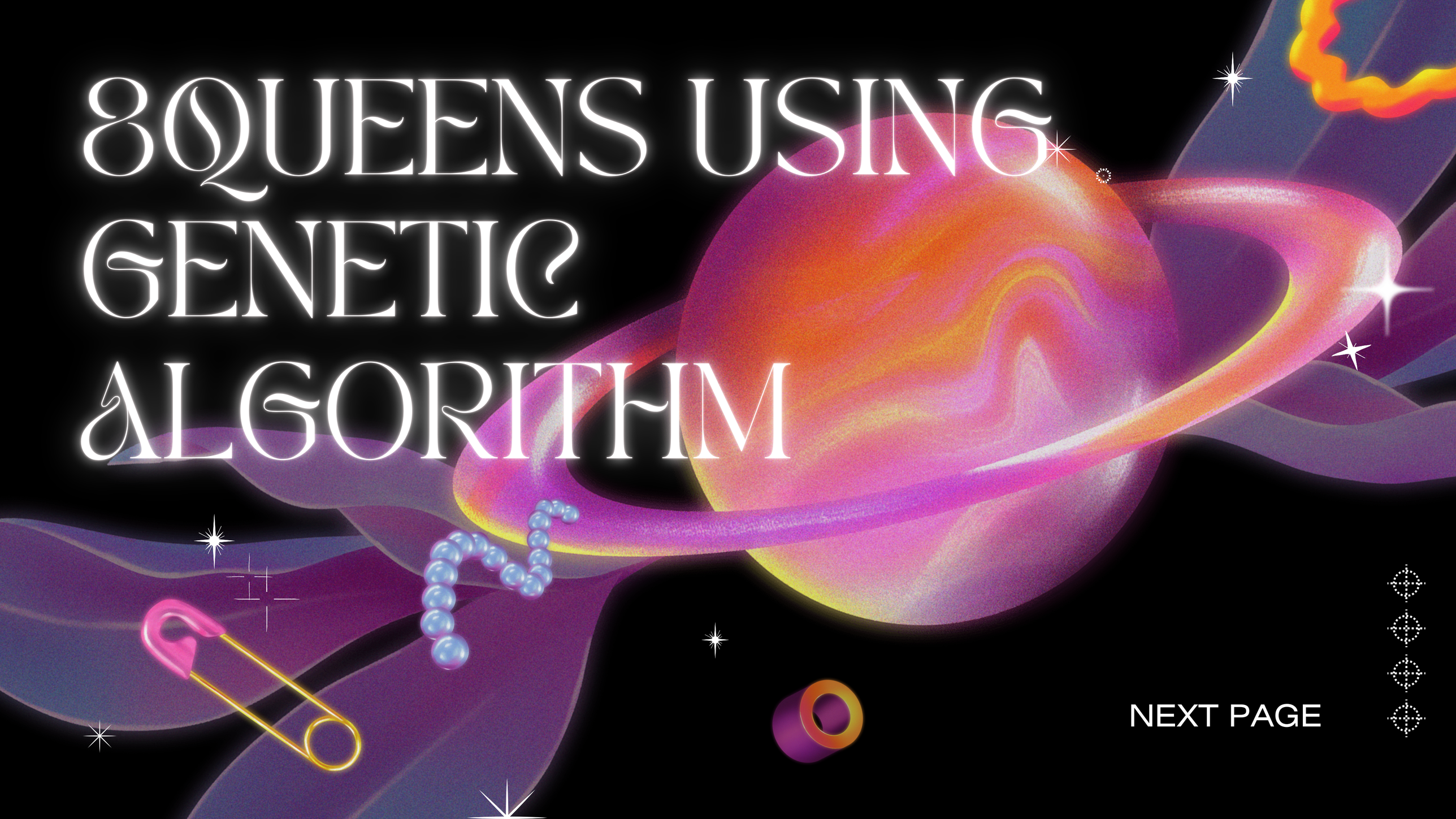
8QUEENS USING GENETIC ALGORITHM



NEXT PAGE

ANGGOTA

NADIAH NURI AISYAH

5025211210

ZAKIA KOLBI

5025211049

SEKAR AMBAR ARUM

5025211041

Genetic Algorithm

Algoritme genetik adalah teknik pencarian yang di dalam ilmu komputer untuk menemukan penyelesaian perkiraan untuk optimisasi dan masalah pencarian. Algoritme genetik adalah kelas khusus dari algoritme evolusioner dengan menggunakan teknik yang terinspirasi oleh biologi evolusioner seperti warisan, mutasi, seleksi alam dan rekombinasi (atau crossover). Teknik ini akan melakukan pencarian dari beberapa solusi yang diperoleh sampai mendapatkan solusi terbaik sesuai dengan kriteria yang telah ditentukan atau yang disebut sebagai fungsi fitness.

Algoritma umum dari algoritma genetik ini dapat dirumuskan menjadi beberapa langkah, yaitu:

- Membentuk suatu populasi individual dengan keadaan acak
- Mengevaluasi kecocokan setiap individual keadaan dengan hasil yang diinginkan
- Memilih individual dengan kecocokan yang tertinggi
- Bereproduksi, mengadakan persilangan antar individual terpilih diselingi mutasi
- Mengulangi langkah 2 - 4 sampai ditemukan individual dengan hasil yang diinginkan

CODE

```python
import random

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

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

    n = len(chromosome)
    left_diagonal = [0] * 2*n
    right_diagonal = [0] * 2*n
    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 / (n-abs(i-n+1))

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

def probability(chromosome, fitness):
    return fitness(chromosome) / maxFitness

```

```python
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"

def reproduce(x, y): #doing cross_over between two chromosomes
    n = len(x)
    c = random.randint(0, n - 1)
    return x[0:c] + y[c:n]

def mutate(x):  #randomly changing the value of a random index of a chromosome
    n = len(x)
    c = random.randint(0, n - 1)
    m = random.randint(1, n)
    x[c] = m
    return x

```

```
53
54  def genetic_queen(population, fitness):
55      mutation_probability = 0.03
56      new_population = []
57      probabilities = [probability(n, fitness) for n in population]
58      for i in range(len(population)):
59          x = random_pick(population, probabilities) #best chromosome 1
60          y = random_pick(population, probabilities) #best chromosome 2
61          child = reproduce(x, y) #creating two new chromosomes from the best 2 chromosomes
62          if random.random() < mutation_probability:
63              child = mutate(child)
64          print_chromosome(child)
65          new_population.append(child)
66          if fitness(child) == maxFitness: break
67      return new_population
68
69  def print_chromosome(chrom):
70      print("Chromosome = {},  Fitness = {}"
71          .format(str(chrom), fitness(chrom)))
72
73  if __name__ == "__main__":
74      nq = int(input("Enter Number of Queens: ")) #say N = 8
75      maxFitness = (nq*(nq-1))/2  # 8*7/2 = 28
76      population = [random_chromosome(nq) for _ in range(100)]
77
78      generation = 1
79
```

```
80      while not maxFitness in [fitness(chrom) for chrom in population]:
81          print("=== Generation {} ===".format(generation))
82          population = genetic_queen(population, fitness)
83          print("")
84          print("Maximum Fitness = {}".format(max([fitness(n) for n in population])))
85          generation += 1
86      chrom_out = []
87      print("Solved in Generation {}!".format(generation-1))
88      for chrom in population:
89          if fitness(chrom) == maxFitness:
90              print("");
91              print("One of the solutions: ")
92              chrom_out = chrom
93              print_chromosome(chrom)
94
95      board = []
96
97      for x in range(nq):
98          board.append(["x"] * nq)
99
100     for i in range(nq):
101         board[nq-chrom_out[i]][i]="Q"
102
```

```
103
104     def print_board(board):
105         for row in board:
106             print (" ".join(row))
107
108     print()
109     print_board(board)
110
```

OUTPUT

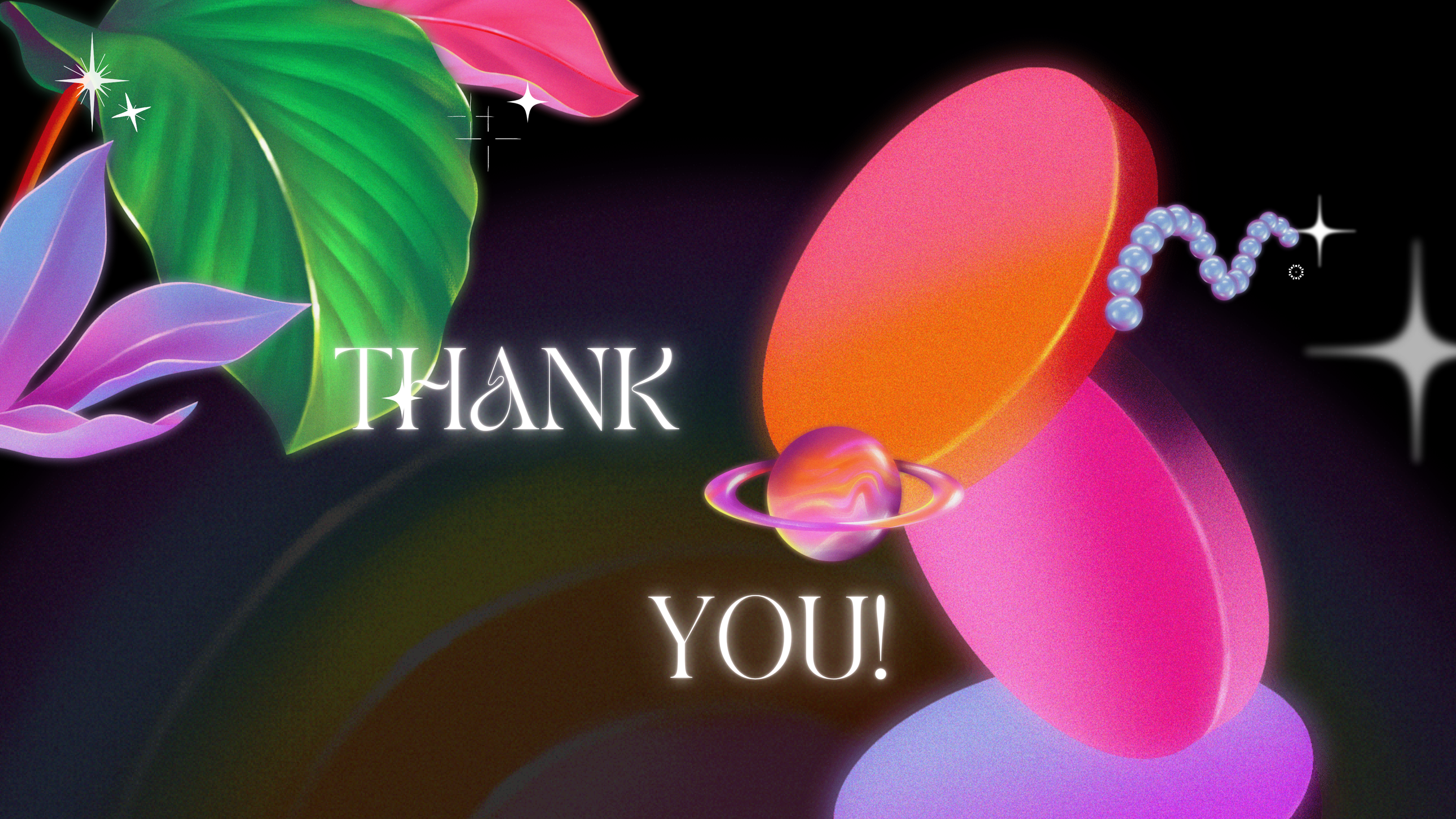
```
Enter Number of Queens: 8
Solved in Generation 0!

One of the solutions:
Chromosome = [7, 1, 3, 8, 6, 5, 4, 2],  Fitness = 28
()
x x x Q x x x x
Q x x x x x x x
x x x x Q x x x
x x x x x Q x x
x x x x x x Q x
x x Q x x x x x
x x x x x x x Q
x Q x x x x x x
mbp13@MBP13s-MacBook-Pro kelompok %
```

```
Maximum Fitness = 27
=== Generation 655 ===
Chromosome = [8, 3, 5, 4, 7, 6, 2, 4],  Fitness = 27
Chromosome = [8, 3, 5, 4, 7, 6, 2, 4],  Fitness = 27
Chromosome = [8, 3, 5, 4, 7, 6, 2, 4],  Fitness = 27
Chromosome = [8, 3, 5, 4, 7, 6, 2, 4],  Fitness = 27
Chromosome = [8, 3, 5, 4, 7, 6, 2, 4],  Fitness = 27
Chromosome = [8, 3, 5, 4, 7, 6, 2, 4],  Fitness = 27
Chromosome = [8, 3, 5, 4, 7, 6, 2, 4],  Fitness = 27
Chromosome = [8, 3, 5, 4, 7, 6, 2, 4],  Fitness = 27
Chromosome = [8, 3, 5, 4, 7, 6, 2, 4],  Fitness = 27
Chromosome = [8, 3, 5, 4, 7, 6, 2, 4],  Fitness = 27
Chromosome = [8, 3, 5, 1, 7, 6, 2, 4],  Fitness = 28

Maximum Fitness = 28
Solved in Generation 655!

One of the solutions:
Chromosome = [8, 3, 5, 1, 7, 6, 2, 4],  Fitness = 28
()
Q x x x x x x x
x x x x Q x x x
x x x x x Q x x
x x Q x x x x x
x x x x x x x Q
x Q x x x x x x
x x x x x x Q x
x x x Q x x x x
```


THANK
YOU!