**LAB – 1**

**GENETIC ALGORITM**

**(APPLICATION USED JOB SCHEDULING PROBLEM)**

**CODE :**

import random

job\_times = [3, 1, 4, 2,]

POP\_SIZE = 6

GENS = 20

MUTATION\_RATE = 0.1

TOURNAMENT\_SIZE = 5

def create\_population(jobs, size):

    return [random.sample(jobs, len(jobs)) for \_ in range(size)]

def fitness(job\_order):

    time = 0

    total = 0

    for job in job\_order:

        time += job

        total += time

    return total

def select(population):

    tournament = random.sample(population, TOURNAMENT\_SIZE)

    tournament.sort(key=fitness)

    return tournament[0]

def crossover(parent1, parent2):

    size = len(parent1)

    start, end = sorted(random.sample(range(size), 2))

    child = [None] \* size

    child[start:end] = parent1[start:end]

    fill\_values = [item for item in parent2 if item not in child]

    idx = 0

    for i in range(size):

        if child[i] is None:

            child[i] = fill\_values[idx]

            idx += 1

    return child

def mutate(chromosome):

    if random.random() < MUTATION\_RATE:

        i, j = random.sample(range(len(chromosome)), 2)

        chromosome[i], chromosome[j] = chromosome[j], chromosome[i]

    return chromosome

def genetic\_algorithm(job\_times):

    population = create\_population(job\_times, POP\_SIZE)

    best = min(population, key=fitness)

    for gen in range(GENS):

        new\_population = []

        for \_ in range(POP\_SIZE):

            parent1 = select(population)

            parent2 = select(population)

            child = crossover(parent1, parent2)

            child = mutate(child)

            new\_population.append(child)

        population = new\_population

        current\_best = min(population, key=fitness)

        if fitness(current\_best) < fitness(best):

            best = current\_best

        print(f"Generation {gen+1}: Best Fitness = {fitness(best)}")

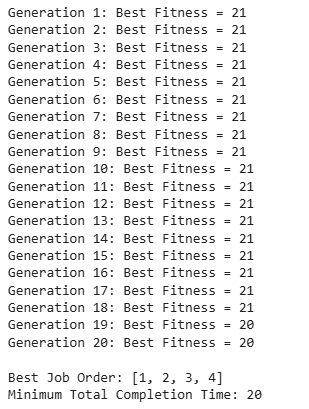
    return best

best\_order = genetic\_algorithm(job\_times)

print("\nBest Job Order:", best\_order)

print("Minimum Total Completion Time:", fitness(best\_order))

**OUTPUT :**

****