**CSE422\_10\_Lab\_Assignment002\_Turnitin\_Summer2024**

**Name : Fathin Ishrak | Student ID : 20301027**

**Part 1:**

#Part 1

import random

with open('20301027\_Fathin Ishrak\_CSE422\_10\_Lab\_Assignment02\_InputFile\_Summer2024.txt', 'r') as file:

data = file.readlines()

N, T = map(int, data[0].split())

courses = [line.strip() for line in data[1:]]

assert T >= N, "Number of timeslots must be greater than or equal to the number of courses."

chromosome\_length = N \* T

def fitness(chromosome):

overlap\_penalty = 0

consistency\_penalty = 0

timeslot\_segments = [chromosome[i\*N:(i+1)\*N] for i in range(T)]

for segment in timeslot\_segments:

overlap\_penalty += max(0, sum(segment) - 1)

course\_counts = [0] \* N

for segment in timeslot\_segments:

for i in range(N):

if segment[i] == 1:

course\_counts[i] += 1

consistency\_penalty += sum(abs(count - 1) for count in course\_counts)

total\_penalty = overlap\_penalty + consistency\_penalty

return -total\_penalty

def generate\_initial\_population(pop\_size):

population = []

for \_ in range(pop\_size):

chromosome = [0] \* chromosome\_length

for i in range(N):

timeslot = random.randint(0, T-1)

chromosome[timeslot \* N + i] = 1

population.append(chromosome)

return population

def select\_parents(population, fitnesses):

parents = []

for \_ in range(2):

tournament = random.sample(list(zip(population, fitnesses)), 3)

parents.append(max(tournament, key=lambda x: x[1])[0])

return parents

def crossover(parent1, parent2):

point = random.randint(1, chromosome\_length - 1)

child1 = parent1[:point] + parent2[point:]

child2 = parent2[:point] + parent1[point:]

return child1, child2

def mutate(chromosome, mutation\_rate):

for i in range(chromosome\_length):

if random.random() < mutation\_rate:

chromosome[i] = 1 - chromosome[i]

def genetic\_algorithm(pop\_size, mutation\_rate, max\_generations):

population = generate\_initial\_population(pop\_size)

best\_fitness = float('-inf')

best\_chromosome = None

for generation in range(max\_generations):

fitnesses = [fitness(chrom) for chrom in population]

new\_population = []

for \_ in range(pop\_size // 2):

parent1, parent2 = select\_parents(population, fitnesses)

child1, child2 = crossover(parent1, parent2)

mutate(child1, mutation\_rate)

mutate(child2, mutation\_rate)

new\_population.extend([child1, child2])

population = new\_population

fitnesses = [fitness(chrom) for chrom in population]

current\_best\_fitness = max(fitnesses)

current\_best\_chromosome = population[fitnesses.index(current\_best\_fitness)]

if current\_best\_fitness > best\_fitness:

best\_fitness = current\_best\_fitness

best\_chromosome = current\_best\_chromosome

return best\_chromosome, best\_fitness

pop\_size = 100

mutation\_rate = 0.01

max\_generations = 1000

best\_solution, best\_fitness = genetic\_algorithm(pop\_size, mutation\_rate, max\_generations)

with open('output1.txt', 'w') as file:

file.write(''.join(map(str, best\_solution)) + '\n')

file.write(str(best\_fitness) + '\n')

**Part 2:**

#Part 2

import random

def generate\_initial\_population(pop\_size, chromosome\_length):

population = []

for \_ in range(pop\_size):

chromosome = [random.randint(0, 1) for \_ in range(chromosome\_length)]

population.append(chromosome)

return population

def two\_point\_crossover(parent1, parent2):

length = len(parent1)

point1 = random.randint(1, length - 2)

point2 = random.randint(point1 + 1, length - 1)

offs1 = parent1[:point1] + parent2[point1:point2] + parent1[point2:]

offs2 = parent2[:point1] + parent1[point1:point2] + parent2[point2:]

return offs1, offs2

def read\_input\_file(filename):

with open(filename, 'r') as file:

lines = file.readlines()

N = int(lines[0].split()[0])

T = int(lines[0].split()[1])

courses = [line.strip() for line in lines[1:]]

return N, T, courses

def write\_output\_file(filename, parent1, parent2, offs1, offs2):

with open(filename, 'w') as file:

file.write("Parent 1: " + ''.join(map(str, parent1)) + "\n")

file.write("Parent 2: " + ''.join(map(str, parent2)) + "\n")

file.write("Offspring 1: " + ''.join(map(str, offs1)) + "\n")

file.write("Offspring 2: " + ''.join(map(str, offs2)) + "\n")

def main():

input\_filename = "20301027\_Fathin Ishrak\_CSE422\_10\_Lab\_Assignment02\_InputFile\_Summer2024.txt"

output\_filename = "output2.txt"

N, T, courses = read\_input\_file(input\_filename)

chromosome\_length = N \* T

pop\_size = 10

population = generate\_initial\_population(pop\_size, chromosome\_length)

parent1, parent2 = random.sample(population, 2)

offs1, offs2 = two\_point\_crossover(parent1, parent2)

write\_output\_file(output\_filename, parent1, parent2, offs1, offs2)

if \_\_name\_\_ == "\_\_main\_\_":

main()