#!/usr/bin/python3

import random

import math

board\_size=8

#initialize population randomly

def populate():

p=[]

for k in range(100):

b=[]

for i in range(8):

b.append(i)

for i in range(4):

j = random.randint(0, board\_size-1)

k = random.randint(0, board\_size-1)

b[j], b[k] = b[k], b[j]

p.append([b,fitness(b)])

return p

#select random 5 out of 100 population

def random5():

v=[]

for i in range(5):

x=random.randint(0,99)

v.append(population[x])

return v

#compute fitness value of each population

def fitness(population):

fitness=100

for i in range(board\_size):

for j in range(i+1, board\_size):

if math.fabs(population[i] - population[j]) == j - i : #check diagonal checkpoints

fitness-=1 #for every checkpoint fitness value decreases

return fitness

#swap mutation of offspring

def mutate(population):

j = random.randint(0, board\_size-1)

k = random.randint(0, board\_size-1)

population[j], population[k] = population[k], population[j]

return population

#-----------main ---------------------

population=populate()

generation\_count=0

flag=0

while(generation\_count<10000 & flag==0):

rand5=random5()

x=[]

for i in range(5):

x.append(rand5[i][1]) #random 5 out of 100 population

x.sort(reverse=True)

x=x[0:2] # best 2 out of random 5

for i in range(5):

if rand5[i][1]==x[0]:

p1=rand5[i][0]

if rand5[i][1]==x[1]:

p2=rand5[i][0]

#cut and crossfill crossover

crossover=random.randint(0,7)

c1=[]

c2=[]

for i in range(crossover):

c1.append(p1[i])

c2.append(p2[i])

i=crossover

while(len(c1)!=8):

if p2[i] in c1:

i=(i+1)%8

else:

c1.append(p2[i])

i=crossover

while(len(c2)!=8):

if p1[i] in c2:

i=(i+1)%8

else:

c2.append(p1[i])

#mutation if mutation probability is <80%

pm=random.randint(0,99)

if pm<80:

c1=mutate(c1)

pm=random.randint(0,99)

if pm<80:

c2=mutate(c2)

population.append([c1,fitness(c1)])

population.append([c2,fitness(c2)])

#worst 2 fitness values

x=[]

for i in range(102):

x.append(population[i][1])

x.sort()

x=x[0:2]

#selecting best 100 out of 102

i=0

while(len(population)!=100):

if x[0]!=x[1]:

if population[i][1]==x[0]:

population.remove([population[i][0],x[0]])

if population[i][1]==x[1]:

population.remove([population[i][0],x[1]])

i+=1

else:

if population[i][1]==x[0]:

population.remove([population[i][0],x[0]])

i+=1

generation\_count+=1

for i in range(100):

if population[i][1]==100:

flag=1

print("""

--------------------8 Queen Problem-------------------------

Board Size: 8

Representation: Permutations

Recombination: Cut and Crossfill crossover

Recombination Probability: 100%

Mutation: Swap

Mutation Probability: 80%

Parent Selection: Best 2 out of random 5

Survival selection: Replace worst

Population Size: 100

Number of Offsprings: 2

Initialisation: Random

Termination Condition: Solution or 10000 generations

""")

print("--------------Output------------------")

if flag==1:

print("Solution achieved after {} generations".format(generation\_count))

else:

print("Terminated after 10000 generations")

x=[]

for i in range(100):

x.append(population[i][1])

x.sort(reverse=True)

for i in range(100):

if population[i][1]==x[0]:

print("best solution-- {} checkpoints {} ".format(population[i][0],100-x[0]))