**import** random

**import** numpy **as** np

**def** state(q):

pop=[random.randint(1,q) **for** i **in** range(q)]

**return** pop

**def** getFitness(pops,q): *#tempState= [2, 1, 4, 6, ,5, 3]*

fitnessF=[]

**for** tempState **in** pops:

max\_fitness=(q\*(q-1))/2

xDiag=[]

yDiag=[]

matrix=np.zeros(shape=(len(tempState),len(tempState)))

**for** i **in** range(len(tempState)): *#matrix banaitechi*

x=tempState[i]

*# print(x)*

*# print(matrix.shape)*

matrix[len(tempState)-x-1][i]=1

xDiag.append(len(tempState)-x)

yDiag.append(i)

totalAttPairsForRows=0

*# print("x diag :",xDiag)*

*# print("y diag :",yDiag)*

**for** i **in** range(len(tempState)): *#for horizontal attacking pairs*

sumRow=np.sum(matrix[i],dtype=int)

attPairsInThisRow=(sumRow\*(sumRow-1))/2

totalAttPairsForRows+=attPairsInThisRow

totalDiagonalAttPairs=0

**for** i **in** range(len(xDiag)):

attPairDiag=0 *#for diagonal attacking pairs*

slicedX=xDiag[i+1:]

slicedY=yDiag[i+1:]

**for** j **in** range(len(slicedX)):

**if**(abs(xDiag[i]-slicedX[j]) == abs(yDiag[i]-slicedY[j])):

attPairDiag= attPairDiag+1

totalDiagonalAttPairs=totalDiagonalAttPairs+attPairDiag

ultimateAttackingPairs= totalDiagonalAttPairs + totalAttPairsForRows

fitness=max\_fitness-ultimateAttackingPairs

*# print(matrix)*

*# print("Total horizontal att pairs -",totalAttPairsForRows)*

*# print("Total diagonal att pairs -",totalDiagonalAttPairs)*

*# print("Total att pairs -", ultimateAttackingPairs)*

fitnessF.append(fitness)

**return** fitnessF

**def** selection(pop, populationFitness):

fitnessProbability = []

**for** i **in** range(len(populationFitness)):

fitnessProbability.append((populationFitness[i] / sum(populationFitness)))

a=[0 **for** i **in** range(len(population))]

**for** i **in** range(len(population)):

a[i]=i

size=1

**return** pop[np.random.choice(a,size,**True**,fitnessProbability)[0]]

**def** crossover(x,y):

crossOverPoint=np.random.randint(0,queens,dtype=int)

newChildFirstH=x[crossOverPoint:]

newChildSecondHalf=y[:crossOverPoint]

**return** newChildFirstH+newChildSecondHalf

**def** mutation(ch):

ranIdx=np.random.randint(0,queens)

randPos=np.random.randint(0,queens)

ch[ranIdx]=randPos

**return** ch

**def** geneticAlgo(pop,q,mut\_tr=0.3):

max\_fitness=(q\*(q-1))/2

generation=0

**while True**:

generation+=1

new\_pop = []

allFitness=getFitness(pop,q)

**if** generation%1000==0:

print(**"Max fit -{} Generation {} "**.format(max(allFitness),generation))

**if** max(allFitness)==max\_fitness **or** generation==200000 :

**return** (pop,allFitness,generation)

**for** i **in** range(len(pop)):

x=selection(pop,allFitness)

y=selection(pop,allFitness)

child=crossover(x,y)

**if**(np.random.random()< mut\_tr):

child=mutation(child)

new\_pop.append(child)

pop=new\_pop

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

queens = int(input(**"Please Enter Number of Queens: "**))

startPopSize=10

population = [state(queens) **for** i **in** range(startPopSize)]

popu,fit,generation=geneticAlgo(population,queens)

print(**"Child {}, Max Fitness {}, Generation {}"**.format(popu[fit.index(max(fit))],max(fit),generation))