# -\*- coding: utf-8 -\*-

"""CSE422\_Lab 2.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1TOcZ7zivxhdcD4r-bv\_5IuedPg5GhQDT

"""

from google.colab import drive

drive.mount('/content/drive')

import random as ran

import numpy as namPie

# lab 2 Task 1

# Run Chase Problem

# Kazi Md. Al-Wakil

# 19301051

#creating enfant

def zebraCrossing (paren1, paren2):

lenguParent = len(paren1)

gimmeIdx = ran.randrange(0,lenguParent) #randomly ekta idx generate korlam

enfant= []

prothomPart = paren1[0:gimmeIdx]

secondPart = paren2[gimmeIdx:]

for itor in prothomPart:

enfant.append(itor) #prothom part nilam first paren theke

for itor in secondPart:

enfant.append(itor) #baki part nilam Second paren theke

return enfant #enfant return kortesi

# mutating the enfant

def evolve(enfant):

lenguEnfant = len(enfant)

whichIdxToEvolve = ran.randrange(0,lenguEnfant)

for i in range(len(enfant)):

if i == whichIdxToEvolve:

if enfant[i] == 1:

enfant[i] = 0

elif enfant[i] == 0:

enfant[i] = 1

return enfant

#choosing baba ma

def paireSelecteur(iqx\_chromosome, SumOfIndividualList, iqx\_targetRuns):

tempuLisu = []

ekdomChoto1 = 9999 #target run er theke amader hate thaka run gula minus kore jei value pabo.

# sei value er ekdom choto value ta best fit, store korbo sei value, setai amader 1st parent

ekdomChoto2 = 9999 #ekdom choto 2nd lowest value ta 2nd paren hisabe store korbo,

ekdomChotoIdx1 = 9999 #ekdom choto value er index

ekdomChotoIdx2 = 9999 #ekdom choto 2nd lowest valu er index

for itor in SumOfIndividualList:

tempuLisu.append(abs(iqx\_targetRuns - itor)) #absolute value nicchi. Target Run er sobcheye kacher value ta ber korbo evabe

# prothom parent choose kortesi

for itor in range(len(tempuLisu)):

if tempuLisu[itor] < ekdomChoto1:

ekdomChoto1 = tempuLisu[itor] #ekdom minimum absolute value ta store korlam

ekdomChotoIdx1 = itor #ekdom minimum absolute value er index store korlam

tempuLisu[ekdomChotoIdx1] = 9999 #prothom paren(Lowest value) ber kora sesh, tai sei value ta baray dicchi,

#2nd parent ber korar somoy tahole r ei parent lowest value hisabe appear hobe na.

# second parent choose kortesi

for itor in range(len(tempuLisu)):

if tempuLisu[itor] < ekdomChoto2:

ekdomChoto2 = tempuLisu[itor]

ekdomChotoIdx2 = itor

tempuLisu[ekdomChotoIdx2] = 9999

#tempuLisu list abar ager moto kore feltesi

tempuLisu[ekdomChotoIdx1] = ekdomChoto1

tempuLisu[ekdomChotoIdx2] = ekdomChoto2

return iqx\_chromosome[ekdomChotoIdx1], iqx\_chromosome[ekdomChotoIdx2]

# shera feet function

def prochurFitnessCalculator(iqx\_playerRun, iqx\_chromosome):

whoIsFitEnough = [] # iqx\_chromosome er sob sample er run ekhane store hobe. nested list

SumOfIndividualList = [] # whoIsFitEnough List er moddhe sob sample er sum ber kore ei list e append korbo

for itor in iqx\_chromosome: #fetching sample from iqx\_chromosome

tempuLisu = []

for jtor, uporerI in zip(iqx\_playerRun, itor): #gun kortesi run list er sathe

tempuLisu.append(jtor\*uporerI) #[68, 25, 70, 53, 71, 55, 66, 29] \* [0, 0, 0, 0, 1, 0, 1, 0]

whoIsFitEnough.append(tempuLisu) # appending this [0, 0, 0, 0, 71, 0, 66, 0]

for itor in whoIsFitEnough:

SumOfIndividualList.append(sum(itor)) # Sum of [0, 0, 0, 0, 71, 0, 66, 0] = 137 # appending 137 to the SumOfList1 list

return SumOfIndividualList

def biyogHowaListFunc(iqx\_chromosome, SumOfIndividualList, iqx\_targetRuns):

tempuLisu = []

for itor in SumOfIndividualList:

tempuLisu.append(abs(iqx\_targetRuns - itor))

return tempuLisu

#choto value and choto value er index dibe

def chotoValueDibeFunc(biyogHowaList):

ekdomChotoZero = 9999

for itor in range(len(biyogHowaList)):

if biyogHowaList[itor] < ekdomChotoZero:

ekdomChotoZero = biyogHowaList[itor] #biyog howa list er sobchey choto value stored

ekdomChotoZeroIdx = itor #biyog howa list er sobchey choto value er index stored

return ekdomChotoZero, ekdomChotoZeroIdx

#ashol algorithm

def genAlgo(iqx\_totalInputs, iqx\_playerRun, iqx\_chromosome, iqx\_targetRuns, mutMargin, iqx\_playerList):

genMargin, genNumber = 22, 0

boolBooliya = True

while boolBooliya == True:

ekdomChotoZero = 9999 #ekhane min value store korbo SumOfIndividualList er. min value 0 hoile goal peye gesi

SumOfIndividualList = prochurFitnessCalculator(iqx\_playerRun, iqx\_chromosome) # fit = [258, 66, 328, 167, 270, 330, 383, 209, 355, 163]

# fit e ekhon sumList chole asche. ekhan theke bujhte parbo, k beshi fit

biyogHowaList = biyogHowaListFunc(iqx\_chromosome, SumOfIndividualList, iqx\_targetRuns) #checking if amar kache emon kono sample ache kina jeta goal er kache or goal itself

ekdomChotoZero, ekdomChotoZeroIdx = chotoValueDibeFunc(biyogHowaList)

if genNumber > genMargin: #genMargin maximum generation. er beshi amra r check korbo na, then break

print(-1)

break

elif ekdomChotoZero == 0: #0 mane amra goal state peye gesi. true hole print kore break

print(iqx\_playerList)

for itor in iqx\_chromosome[ekdomChotoZeroIdx]:

print(itor,end="")

break

else:

newGen = []

for itor in range(len(iqx\_chromosome)):

paren1, paren2 = paireSelecteur(iqx\_chromosome, SumOfIndividualList, iqx\_targetRuns) #2 jon parent select kora hocche. jara best fit among all

enfant = zebraCrossing (paren1, paren2) #enfant of the parens

evolveKihobe = ran.random() #generating random number

if evolveKihobe < mutMargin: #jodi random number mutation threshold theke kom hoy taile mutate hobe

enfant = evolve(enfant)

newGen.append(enfant) #ekta notun gen create hocche

#iqx\_chromosome e new gen add kori

for itor in newGen:

iqx\_chromosome.append(itor)

genNumber = genNumber + 1

print("GenNumber: ",genNumber)

# Storing and initializing all the values

def sovaLineup(inputLine):

tempuLisu = [] #temporary list

iqx\_playerList = [] #player der nam store korbo

iqx\_playerRun = [] #player er run list

iqx\_totalInputs = 0 #total koyta input seta store korbo

iqx\_targetRuns = 0 #Target run ta store korbo

for itor in range(len(inputLine)):

if (itor == 0):

firstLineOfTheInput = inputLine[0]

else:

stri = inputLine[itor]

tempuLisu.append(stri)

firstLineOfTheInputList= firstLineOfTheInput.split()

iqx\_totalInputs, iqx\_targetRuns = int(firstLineOfTheInputList[0]), int(firstLineOfTheInputList[1])

for itor in tempuLisu:

tempuLisu2 = itor.split()

iqx\_playerList.append(tempuLisu2[0])

iqx\_playerRun.append(int(tempuLisu2[1]))

return iqx\_totalInputs, iqx\_targetRuns, iqx\_playerList, iqx\_playerRun

# lab 2 Task 1

reading = open("/content/drive/MyDrive/CSE422 Labs/Lab2\_T1\_Input1.txt","r")

inputLine = reading.readlines()

iqx\_totalInputs, iqx\_targetRuns, iqx\_playerList, iqx\_playerRun = sovaLineup(inputLine) # Storing and initializing all the values

initialPopu = 10 # Population size 10 ta nilam

iqx\_chromosome = [] # choromosome list e 10 ta randomly generated sample ache,

#sei sample e kon kon batsman pick korsi tader data ache.

# 1 hole batsman pick korsi, 0 hole pick kori nai

zeroList = [] #contains only zeros and nothing else

for itor1 in range(iqx\_totalInputs): #zero er list banailam, jate kore only 0 list population matrix e append na hoy

zeroList.append(0)

iqx\_itor = 0

while iqx\_itor < initialPopu: # total 10 ta population list banabo

tempchromo = []

for jtor in range(iqx\_totalInputs):

tempchromo.append(ran.randrange(0,2))

if(tempchromo not in iqx\_chromosome and tempchromo != zeroList):

iqx\_chromosome.append(tempchromo)

iqx\_itor +=1

mutMargin = ((((1+2)\*10)+126)+(2\*100)-(2\*28))/1000 #mutation threshold, er upore gele mutation hobe, otherwise hobe na

genAlgo(iqx\_totalInputs, iqx\_playerRun, iqx\_chromosome, iqx\_targetRuns, mutMargin, iqx\_playerList)