import random

import numpy as np

#Generating chromosomes

def generate\_population(chromosone\_numbers):

chromosome\_lst=[]

for i in range(8): #need to upadte the range around 100

chromosome=np.random.randint(0,2,size=chromosone\_numbers)

chromosome\_lst.append(chromosome)

#print(f'Chromosomes\n')

#for i in chromosome\_lst:

#print(i)

#print("\_\_"\*14)

return chromosome\_lst

#Calculating fitness for each chromosome

def fitness\_calculation(chromosome\_lst):

fitness\_score=[]

for i in range(len(chromosome\_lst)):

score=0

#Calculating fitness for each particular chromosome

for idx, gene in enumerate(chromosome\_lst[i]):

if gene!=0:

score+=transaction[idx]

fitness\_score.append(abs(score))

#print(fitness\_score)

return fitness\_score

#Finding the least fit chromosome and checking if the goal is achieved or not

def selection(fitness\_scores):

return fitness\_scores.index(min(fitness\_scores))

#Crossing over between 2 chromosome

def crossover(chr1,chr2):

chr1,chr2=list(chr1),list(chr2)

point=random.randint(1,len(chr1)-2)

chr1\_fh,chr1\_lh=chr1[:point+1],chr1[point+1:]

chr2\_fh,chr2\_lh=chr2[:point+1],chr2[point+1:]

child1=(chr1\_fh+chr2\_lh)

child2=(chr2\_fh+chr1\_lh)

child\_lst=[]

child\_lst.append(child1)

child\_lst.append(child2)

return child\_lst

#Picking a random child and mutating their gene in the chromosome

def mutation(off1,off2):

#print(f"offspring1: {off1}\n offrping 2: {off2}")

#choosing a random point to mutate

point=random.randint(0,len(off1)-1)

#print('mutaion point', point)

if off1[point]==0:

off1[point]=1

else:

off1[point]=0

#print('mutated off 1',off1)

if (off2[point]==0):

off2[point]=1

else:

off2[point]=0

#print('mutated off 2',off2)

return off1,off2

transaction=[]

goal\_achieved=False

output\_sequence=np.array([])#Output list for initialization

inp\_file=open('Input\_file\_02.txt','r')

lines=inp\_file.readlines()

number\_of\_transaction=int(lines[0])

inp\_file.close()

for i in range(1,len(lines)):

transaction\_type,amount=lines[i].split()

if transaction\_type=='l':

transaction.append(-int(amount))

else:

transaction.append(int(amount))

#print(f'Trsancation taken place: {transaction}')

#loop for Genetic Algorithm

for i in range(1500): #need to upadte the range around 150

#generating population

population=generate\_population(number\_of\_transaction)

#finding the fitnesss of each chromose in the population

fitness=fitness\_calculation(population)

#Finding the least\_fit chromosome index and checking if that satisfies the condition

x = selection(fitness)

if fitness[x] == 0:

if list(population[x]) != [0]\*number\_of\_transaction:

#print(i,j)

goal\_achieved=True

output\_sequence=np.array(population[x])

break

for j in range(len(population)):

#randomly choosing two chromose for crossing over

chr1=random.choice(population)

chr2=random.choice(population)

offspr=crossover(chr1,chr2)

offspr1,offspr2=offspr[0],offspr[1]

mutated=mutation(offspr1,offspr2)

#Checking if anyof the mutated chromosome help us achieve the target balance

total\_balance=0

for x,y in enumerate(offspr1):

if x==1:

total\_balance+=transaction[x]

if total\_balance==0:

if offspr1!=([0] \* len(offspr1)):

output\_sequence=np.array(offspr1)

goal\_achieved=True

break

total\_balance=0

for x,y in enumerate(offspr2):

if x==1:

total\_balance+=transaction[x]

if total\_balance==0:

if offspr2!=([0] \* len(offspr2)):

output\_sequence=np.array(offspr2)

goal\_achieved=True

break

#if result achieved terminate the code

if goal\_achieved==True:

break

if output\_sequence.size==0:

#print(i,j)

print(-1)

else:

print(output\_sequence)