N - Queen Problem Using Genetic Algorithm

Professor : **Dr.AliShakiba**

By : **Maryam Mohammadabadi**

1.ابتدا کتابخوانه ی مورد نظر را import میکنیم.

```
import random
```

2.سپس برای ساخت کرموزوم تصادفی یک لیست در بازه (1,size) میسازیم

```
random_chromosome = lambda size: [random.randint(1,size) for _ in range(size)]
random_chromosome.__doc__='making random chromosomes'
```

3.تابع fitness کرموزوم را به عنوان پارامتر دریافت میکند و جفت ملکه های غیر حمله را محاسبه میکند.

```
def fitness(chromosome,maxFitness=None):
    n = len(chromosome)
    if maxFitness==None:
        maxFitness=(n*(n-1))/2
    horizontal_collisions = sum([chromosome.count(queen)-1 for queen in chromosome])/2

    left_diagonal=[0]*(2*n)
    right_diagonal=[0]*(2*n)
    for index,chrom in enumerate(chromosome):
        left_diagonal[index+chrom-1]+=1
        right_diagonal[n-index+chrom-2]+=1

    diagonal_collisions = 0
    for i in range(2*n-1):
        counter = 0
        if left_diagonal[i]  > 1: counter += left_diagonal[i]-1
        if right_diagonal[i] > 1: counter += right_diagonal[i]-1
        diagonal_collisions+= counter / (n-abs(i-n+1))

    return int(maxFitness-(horizontal_collisions + diagonal_collisions))
```

4. سپس باید احتمال انتخاب از تابع Fitness را محاسبه کنیم. احتمال کروموزوم در probability قرار میگیرد.

```
probability=lambda chromosome, fitness,maxFitness=1: fitness(chromosome)/maxFitness
```

5.تابع انتخاب تصادفی که جمعیت (population) و احتمالات (probabilities)را به عنوان پارامتر دریافت میکند.

```
def random_pick(population, probabilities):
    # tmp={tuple(i):j for i,j in zip(population, probabilities)}
    # return list(max(tmp,key=lambda x:tmp[x]))
    r = sum(probabilities)*random.random()
    upto = 0
    for c, w in zip(population, probabilities):
        upto+=w
        if upto>=r: return c
    raise RuntimeError("This is unreachable state :(")
```

6.تابع reproduce (پیوند)دو کروموزوم را دریافت میکند و پیوند(crossover) بین انها برقرار میکند.

```
def reproduce(x, y):
    assert len(x)==len(y)
    '''doing cross_over between two chromosomes'''
    c = random.randint(0,len(x)-1)
    return x[:c]+y[c:]
```

7.تابع mutate (جهش) یک کروموزوم را به عنوان پارامتر دریافت میکند و به صورت تصادفی مقادیر یک کروموزم را تغیر میدهد

```
def mutate(x):
    '''randomly changing the value of a random index of a chromosome'''
    n = len(x)
    x[random.randint(0,n-1)]=random.randint(1,n)
    return x
```

8.در تابع genetic_queen ابتدا احتمالات را بااستفاده از تابع probability به دست می اورد.

سپس در یک حلقه For بااستفاده از تابع random_pick دو تا از بهترین کروموزم ها را انتخاب کرده و بااستفاده از تابع reproduce دو کروموزوم جدید ایجاد میشود.

سپس در مرحله بعدی یک عدد تصادفی ایجاد کرده با استفاده از random.random() و ان را با mutationProbability مقایسه میکنیم.

اگر mutationProbability کمتر از عدد تصدفی ایجاده شده بود , فرزندان ایجاد شده(child) را به تابع mutate (جهش) میدهیم.

و سپس انها را در لیست new_population قرار میدهیم.

این تابع تا زمانی ادامه پیدا میکند که fitness کروموزوم ها به حداکثر خود برسد یا تمام جمعیت را پیمایش کند.

در اخر جمعیت جدید را برمیگرداند

```
def genetic_queen(population, fitness,maxFitness):

    mutationProbability = 0.03
    new_population = []

    probabilities = [probability(n, fitness,maxFitness) for n in population]

    for _ in population:

        x = random_pick(population, probabilities) #best chromosome 1
        y = random_pick(population, probabilities) #best chromosome 2

        child = reproduce(x, y) #creating two new chromosomes from the best 2 chromosomes

        if random.random() < mutationProbability:

            child = mutate(child)

        new_population.append(child)

        if fitness(child) == maxFitness: break

    return new_population
```

9.تابع main سایز را به عنوان پارامتر دریافت میکند .در ابتدا جمعیت اولیه را 100قرار میدهد

سپس maxFitness را با فرمول (nq*(nq-1))/2 محاسبه کرده.

حلقه while برای ایجاد همه سناریو های ممکن

و حلقه for بعدی کروموزم مورد نظر را در جمعیت پیدا کرده و board آن را برمیگرداند.

```
def main(nq,initialPopulation=100):

    maxFitness = (nq*(nq-1))/2

    generationCount = 1

    population = [random_chromosome(nq) for _ in range(initialPopulation)]

    while not maxFitness in [fitness(chrom) for chrom in population]:
        # new generation starts
        population = genetic_queen(population, fitness,maxFitness)
        generationCount += 1
    generationCount-=1
    for chrom in population:
        if fitness(chrom) == maxFitness:
            return "\n".join('x '*(i-1)+'Q '+'x '*(nq-i) for i in chrom),generationCount
```

و در اخر TABLE_SIZE را از کاربر دریافت کرده و تابع مربوطه را فراخوانی میکنیم

```
if __name__ == "__main__":
    TABLE_SIZE = int(input("Enter Number of Queens: "))
    print("{}\n\nsolution found after {} generations".format(*main(TABLE_SIZE)))
```

تست خروجی

برای ورودی 5 :

```
Enter Number of Queens: 5
```

خروجی:

```
Enter Number of Queens: 5
x x x x Q
x Q x x x
x x x Q x
Q x x x x
x x Q x x

solution found after 8 generations
```

97143045

مریم محمدابادی

1400/2/30