Ahsanullah University of Science and Technology

Department of Computer Science & Engineering

Course No.	CSE 4108
Course Name	Artificial Intelligence Lab
Assignment No.	04

Submitted To:

Md. Siam Ansary	Tonmoy Hossain
Department of CSE, AUST	Department of CSE, AUST

Submitted By:

Name	Tahiya Ahmed Chowdhury
ID No.	17.02.04.048
Session	Fall – 2020
Section	A (A2)
Date of Submission:	August 23, 2021

Implement Genetic Algorithm for the 8 Queens Problem using Python. For Mutation step, use Swap Mutation.

Python code:

```
import numpy as np
import random

inputSet = []

def randomNumGenerator():
    for i in range(0, 15):
        a = []
        for j in range(0, 8):
            a.append(random.randrange(0,7));
        inputSet.append(a)

def prepareForfitnessCalculation():
    for i in range(0,15):
        b = []
        for j in range(0,8):
            tempLst = []
            for k in range(0,1):
                tempLst.append(inputSet[i][j])
                tempLst.append(j)
            b.append(tempLst)
        fitnessCalculation(sorted(b),i)

def fitnessCalculation(positionList,k):
    h = 0
    for i in range(0,7):
        for j in range(i+1,7):
            t1 = abs(positionList[i][0] - positionList[j][0])
            t2 = abs(positionList[i][1] - positionList[j][1])
            if positionList[i][0] == positionList[j][0]:
                h = h+1
            elif t1 == t2:
                h = h+1
    thisdict[k] = h

def getFirstTwoKeys(dictionary):
    n = 0
    Index = []
    for state in dictionary:
        Index.append(state)
        n = n+1
        if n == 2:
            break
    return Index
```

```
def crossover():
    n = 0; IndexAsc = []; IndexDes = []
    Parent1 = []; Parent2 = []
    Child1 = []; Child2 = []

    IndexAsc = getFirstTwoKeys(sort_dict)
    IndexDes = getFirstTwoKeys(sort_dict_rev)

    for i in range(0,15):
        if(i == IndexAsc[0]):
             for j in range(0,8):
                 Parent1.append(inputSet[i][j])
        elif(i == IndexAsc[1]):
             for j in range(0,8):
                 Parent2.append(inputSet[i][j])

    for j in range(0,4):
        Child1.append(Parent1[j])
        Child2.append(Parent2[j])

    for j in range(4,8):
        Child1.append(Parent2[j])
        Child2.append(Parent1[j])

    m = random.randrange(0,7)
    n = random.randrange(0,7)
    Child1[m], Child1[n] = Child1[n], Child1[m]
    Child2[m], Child2[n] = Child2[n], Child2[m]

    for i in range(0,15):
        if(i == IndexDes[0]):
             for j in range(0,8):
                 inputSet[i][j] = Child1[j]
        elif(i == IndexDes[1]):
             for j in range(0,8):
                 inputSet[i][j] = Child2[j]

    print("\nThe children after crossover:")
    print(Child1)
    print(Child2)
    print("\nThe input set after rearranging:")
    print(inputSet)


randomNumGenerator()
for d in range(0,100):
    thisdict = {}
    prepareForfitnessCalculation()
    print("\nThe input set at first:")
    print(inputSet)
    print("\nThe entry numbers vs fitness scores:")
    print(thisdict)
    sort_dict = dict(sorted(thisdict.items(), key=lambda item: item[1]))
    sort_dict_rev = dict(sorted(thisdict.items(), key=lambda item:
item[1],reverse=True))
    crossover()
```

Analyzing code:

Here the list inputSet is a nested list which contains 8 columns in 15 rows. In each row, the queen positions are denoted in their colums. Thus we have total 15 position sets. The function randomNumGenerator assigns values from the range 0 to 7 i.e the queen positions to the inputSet. Then the inputSet is modified row by row by storing each rows in a tempList to pass it to the fitnessCalculation function. This was fitness of each set is determined and stored in the dictionary values of thisdict with row numbers as the keys. The dictionary is then sorted into ascending and descending order and stored into sort_dict and sort_dict_reverse respectively. The getFirstTwoKeys function is used to get Parent1 and Parent2, the entries with lowest fitness score and create Child1 and Child2 by passing sort_dict. Finally, using the same function, the entries with maximum fitness score is selected using sort_dict_reverse and replaced with Child1 and Child2. Thus a modified inputSet is obtained.

This procedure runs in a loop of 100 times. And each time the fitness score is calculated.

Python output:

```
The input set at first:
[[6, 0, 5, 0, 1, 1, 5, 1], [2, 0, 3, 5, 4, 1, 4, 4], [4, 4, 0, 3, 5, 0, 1, 5], [
1, 5, 1, 0, 3, 1, 1, 4], [3, 4, 0, 5, 4, 3, 3, 6], [4, 2, 0, 6, 5, 4, 3, 6], [4,
 0, 1, 0, 4, 2, 4, 0], [6, 6, 1, 5, 5, 2, 3, 3], [3, 4, 0, 6, 4, 3, 2, 6], [2, 1
, 4, 6, 6, 5, 0, 5], [2, 4, 6, 1, 5, 0, 4, 6], [6, 4, 1, 4, 1, 0, 5, 2], [4, 2,
3, 0, 4, 0, 1, 3], [1, 1, 2, 1, 6, 3, 0, 2], [2, 2, 6, 0, 5, 3, 2, 5]]

The entry numbers vs fitness scores:
{0: 7, 1: 3, 2: 4, 3: 10, 4: 9, 5: 8, 6: 8, 7: 4, 8: 9, 9: 5, 10: 3, 11: 7, 12:
6, 13: 7, 14: 8}

The children after crossover:
[2, 0, 3, 5, 5, 0, 4, 6]
[2, 4, 6, 1, 4, 1, 4, 4]

The input set after rearranging:
[[6, 0, 5, 0, 1, 1, 5, 1], [2, 0, 3, 5, 4, 1, 4, 4], [4, 4, 0, 3, 5, 0, 1, 5], [
2, 0, 3, 5, 5, 0, 4, 6], [2, 4, 6, 1, 4, 1, 4, 4], [4, 2, 0, 6, 5, 4, 3, 6], [4,
 0, 1, 0, 4, 2, 4, 0], [6, 6, 1, 5, 5, 2, 3, 3], [3, 4, 0, 6, 4, 3, 2, 6], [2, 1
, 4, 6, 6, 5, 0, 5], [2, 4, 6, 1, 5, 0, 4, 6], [6, 4, 1, 4, 1, 0, 5, 2], [4, 2,
3, 0, 4, 0, 1, 3], [1, 1, 2, 1, 6, 3, 0, 2], [2, 2, 6, 0, 5, 3, 2, 5]]
```

Analyzing output:

Initially the inputSet and the fitness score for each rows i.e. entries are shown. From the fitness score of above output we can see that entry 01 and 10 has the minimum fitness sores. Thus they are selected as the Parent1 and Parent2 and then Child1 and Child2 is formed from them. Then swap mutation is performed and finally entry 03 and 04, the entries with maximum fitness, are replaced by Child1 and Child2. Thus a modified inputSet is obtained which maybe used for next iteration.