AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY DHAKA-1208, BANGLADESH.

Department of Computer Science and Engineering
Spring 2019

Program: Bachelor of Science in Computer Science and Engineering
Course No: CSE 4108
Course Title: Artificial Intelligence Lab

Term Project: 02
Topic: Simulated Annealing for 8-Queens Problem

Date of Submission: October 6, 2019

Submitted to:
Dr. S.M. Abdullah Al-Mamun Professor, Department of
CSE, AUST.

Md. Siam Ansary
Adjunct Faculty, Department of CSE, AUST.

Submitted By:
Name: Syed Sanzam
ID: 16.01.04.042
Group: A2

Preface:

We take the 8-Queens problem to demonstrate **Simulated Annealing,** which is a variation of the **Hill-Climbing Search** algorithm. In this particular variant of the algorithm, a random Uphill state from the Successors is chosen directly. However, sometimes a **Downhill** state is also chosen with a given probability.

Constraints:

- States are given as a list of integer numbers.
- .The probability of choosing the **Uphill** Successor, in this case, is 75% and therefore, the probability of choosing a **Downhill** Successor is 25%.
- Threshold value, which the Maximum Number of Non-Attacking Pairs is taken 27.
- Any successor state that gives more than or equal to 22 Non-Attacking Pairs is considered an Uphill State.

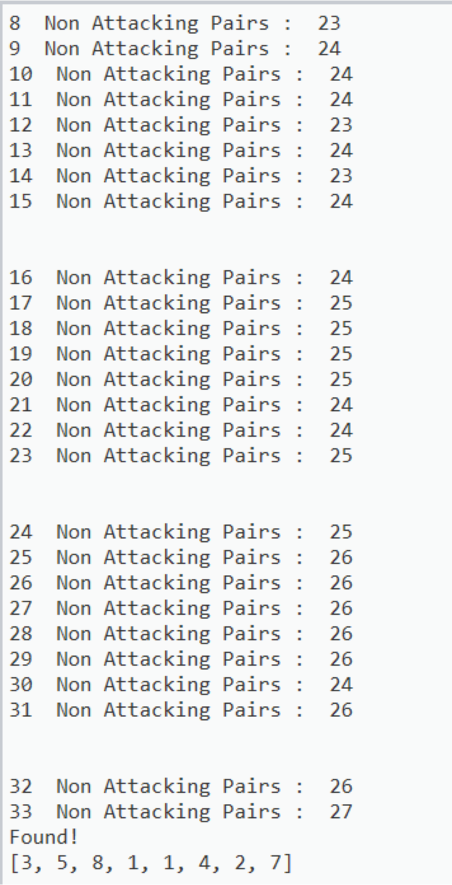
Python Implementation:

simulatedAnnealing.py

```
count = 1
initState = [2,3,4,5,6,5,7,8]

def generateSuccessor():
    global initState
    initState = [2,3,4,5,6,5,7,8]
    state = initState
    length = len(initState)
    successorVal = 1

    for i in range(length):
        indexVal = initState[i]
        for j in range(length):
            if(successorVal != indexVal):
                state[i] = successorVal
                #print(state)
                findNAP(state)
            successorVal = successorVal + 1

        print('\n')
        state = initState
        successorVal = 1

def findNAP(state):
    #Init Variables
    row = 8
    col = 8
    length = len(state)
    matrix = [[0 for i in range(col+1)]for j in range(row+1)]
    sameRow = 0
    n = 0
    global count
    global initState

    #Init matrix with 0-Based Index
```

```python
    for i in range(length):
        matrix[i][state[i] - 1] = 1

    #Calculate Attacking Pair of Queens
    for i in range(row):
        for j in range(col):
            if(matrix[i][j] == 1):
                for k in range(j,col):
                    if(matrix[i][k] == 1):
                        n = n + 1
                if(matrix[i+1][j-1] == 1 or matrix[i+1][j+1] == 1):
                    n = n + 1;

    print(str(count) + " Non Attacking Pairs : " + str(28 - n))
    count = count + 1
    if(28 - n == 27):
        print("Found " + str(state))
        return

    #Selecting Uphill Successor
    if(28 - n > 22):
        initState = state


#Main
generateSuccessor()

```

Output:



```
8  Non Attacking Pairs :  23
9  Non Attacking Pairs :  24
10  Non Attacking Pairs :  24
11  Non Attacking Pairs :  24
12  Non Attacking Pairs :  23
13  Non Attacking Pairs :  24
14  Non Attacking Pairs :  23
15  Non Attacking Pairs :  24


16  Non Attacking Pairs :  24
17  Non Attacking Pairs :  25
18  Non Attacking Pairs :  25
19  Non Attacking Pairs :  25
20  Non Attacking Pairs :  25
21  Non Attacking Pairs :  24
22  Non Attacking Pairs :  24
23  Non Attacking Pairs :  25


24  Non Attacking Pairs :  25
25  Non Attacking Pairs :  26
26  Non Attacking Pairs :  26
27  Non Attacking Pairs :  26
28  Non Attacking Pairs :  26
29  Non Attacking Pairs :  26
30  Non Attacking Pairs :  24
31  Non Attacking Pairs :  26


32  Non Attacking Pairs :  26
33  Non Attacking Pairs :  27
Found!
[3, 5, 8, 1, 1, 4, 2, 7]
```