Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

Program: Bachelor of Science in Computer Science and Engineering

Course No: CSE 4108

Course Title: Artificial Intelligence Lab

Assignment No: 03

Date of Submission: 30 / 01 / 2022

Submitted to:

Mr. Faisal Muhammad Shah
Associate Professor, Department of CSE, AUST.

Mr. Md. Siam Ansary
Lecturer, Department of CSE, AUST.

Submitted by,

Name: Nusrat Jahan

Student ID: 180104020

Question 1: Implement Genetic Algorithm for 8 Queens problem using Python. For the mutation step, use Swap mutation.

Solution:

Python Code:

```
import random


def view(li, index):
    print()
    print(f"Solution number {index + 1}: ", end='')
    print(li)
    print()

    for i in range(8):
        x = li[i] - 1
        for j in range(8):
            if j == x:
                print('[Q]', end='')
            else:
                print('[ ]', end='')
        print()

    print()


def getHuristic(instance):
    huristic = []
    for i in range(len(instance)):
        j = i - 1
        huristic.append(0)
        while j >= 0:
            if instance[i] == instance[j] or (abs(instance[i] -
instance[j]) == abs(i - j)):
                huristic[i] += 1
            j -= 1
        j = i + 1
        while j < len(instance):
            if instance[i] == instance[j] or (abs(instance[i] -
instance[j]) == abs(i - j)):
                huristic[i] += 1
            j += 1
    return huristic
```

```
def getFitness(instance):
    clashes = 0
    for i in range(len(instance) - 1):
        for j in range(i + 1, len(instance)):
            if instance[i] == instance[j]:
                clashes += 1
    for i in range(len(instance) - 1):
        for j in range(i + 1, len(instance)):
            if abs(instance[j] - instance[i]) == abs(j - i):
                clashes += 1
    return 28 - clashes


def buildKid(instance1, instance2, crossOver):
    newInstance = []
    for i in range(crossOver):
        newInstance.append(instance1[random.randint(0, 7)])
    for i in range(crossOver, 8):
        newInstance.append(instance2[random.randint(0, 7)])
    return newInstance


def changeChilds(co):
    global father, mother, child1, child2, crossover
    crossover = co
    child1 = buildKid(father, mother, crossover)
    child2 = buildKid(mother, father, crossover)


def changeChromosome(li):
    global crossover, father, mother
    newchange = -1
    while newchange != 0:
        newchange = 0
        tmpli = li
        getHur = getHuristic(tmpli)
        index = getHur.index(max(getHur))
        maxFitness = getFitness(tmpli)
        for i in range(1, 9):
            tmpli[index] = i
            if getFitness(tmpli) > maxFitness:
                maxFitness = getFitness(tmpli)
                newchange = i
            tmpli = li
        if newchange == 0:
```

```
            for i in range(len(li) - 1):
                for j in range(i + 1, len(li)):
                    if li[i] == li[j]:
                        li[j] = random.randint(1, 8)
        else:
            li[index] = newchange


if __name__ == "__main__":
    numberOfSolutions = int(input())

    solutions = []
    crossover = 4
    while len(solutions) < numberOfSolutions:
        father = []
        mother = []
        for i in range(8):
            father.append(random.randint(1, 8))
            mother.append(random.randint(1, 8))
        fitnessFather = getFitness(father)
        fitnessMother = getFitness(mother)
        while fitnessFather != 28 and fitnessMother != 28:
            changeChilds(crossover)
            changeChromosome(child1)
            changeChromosome(child2)
            fitnessFather = getFitness(child1)
            fitnessMother = getFitness(child2)
            father = child1
            mother = child2
            print(father)
            print(mother)
        if getFitness(father) == 28:
            if father not in solutions:
                solutions.append(father)
        else:
            if mother not in solutions:
                solutions.append(mother)

    for i in range(len(solutions)):
        view(solutions[i], i)
```

```python
import random


def view(li, index):
    print()
    print(f"Solution number {index + 1}: ", end='')
    print(li)
    print()

    for i in range(8):
        x = li[i] - 1
        for j in range(8):
            if j == x:
                print('[Q]', end='')
            else:
                print('[ ]', end='')
        print()

    print()


def getHuristic(instance):
    huristic = []
    for i in range(len(instance)):
        j = i - 1
        huristic.append(0)
        while j >= 0:
            if instance[i] == instance[j] or (abs(instance[i] - instance[j]) == abs(i - j)):
                huristic[i] += 1
            j -= 1
        j = i + 1
        while j < len(instance):
            if instance[i] == instance[j] or (abs(instance[i] - instance[j]) == abs(i - j)):
                huristic[i] += 1
            j += 1
    return huristic


def getFitness(instance):
    clashes = 0
    for i in range(len(instance) - 1):
        for j in range(i + 1, len(instance)):
            if instance[i] == instance[j]:
                clashes += 1
    for i in range(len(instance) - 1):
        for j in range(i + 1, len(instance)):
            if abs(instance[j] - instance[i]) == abs(j - i):
```

```
                if abs(instance[j] - instance[i]) == abs(j - i):
                    clashes += 1
    return 28 - clashes


def buildKid(instance1, instance2, crossOver):
    newInstance = []
    for i in range(crossOver):
        newInstance.append(instance1[random.randint(0, 7)])
    for i in range(crossOver, 8):
        newInstance.append(instance2[random.randint(0, 7)])
    return newInstance


def changeChilds(co):
    global father, mother, child1, child2, crossover
    crossover = co
    child1 = buildKid(father, mother, crossover)
    child2 = buildKid(mother, father, crossover)
```

```python
def changeChromosome(li):
    global crossover, father, mother
    newchange = -1
    while newchange != 0:
        newchange = 0
        tmpli = li
        getHur = getHuristic(tmpli)
        index = getHur.index(max(getHur))
        maxFitness = getFitness(tmpli)
        for i in range(1, 9):
            tmpli[index] = i
            if getFitness(tmpli) > maxFitness:
                maxFitness = getFitness(tmpli)
                newchange = i
            tmpli = li
        if newchange == 0:
            for i in range(len(li) - 1):
                for j in range(i + 1, len(li)):
                    if li[i] == li[j]:
                        li[j] = random.randint(1, 8)
        else:
            li[index] = newchange

```

```python
if __name__ == "__main__":
    numberOfSolutions = int(input())

    solutions = []
    crossover = 4
    while len(solutions) < numberOfSolutions:
        father = []
        mother = []
        for i in range(8):
            father.append(random.randint(1, 8))
            mother.append(random.randint(1, 8))
        fitnessFather = getFitness(father)
        fitnessMother = getFitness(mother)
        while fitnessFather != 28 and fitnessMother != 28:
            changeChilds(crossover)
            changeChromosome(child1)
            changeChromosome(child2)
            fitnessFather = getFitness(child1)
            fitnessMother = getFitness(child2)
            father = child1
```

```
            father = child1
            mother = child2
            print(father)
            print(mother)
        if getFitness(father) == 28:
            if father not in solutions:
                solutions.append(father)
        else:
            if mother not in solutions:
                solutions.append(mother)

    for i in range(len(solutions)):
        view(solutions[i], i)

```

```
Solution number 1: [4, 8, 5, 3, 1, 7, 2, 6]

[ ][ ][ ][Q][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][Q]
[ ][ ][ ][ ][Q][ ][ ][ ]
[ ][ ][Q][ ][ ][ ][ ][ ]
[Q][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][Q][ ]
[ ][Q][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][Q][ ][ ]


Solution number 2: [4, 1, 5, 8, 2, 7, 3, 6]

[ ][ ][ ][Q][ ][ ][ ][ ]
[Q][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][Q][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][Q]
[ ][Q][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][Q][ ]
[ ][ ][Q][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][Q][ ][ ]
```

```
Solution number 3: [8, 3, 1, 6, 2, 5, 7, 4]

[ ][ ][ ][ ][ ][ ][Q]
[ ][ ][Q][ ][ ][ ][ ]
[Q][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][Q][ ]
[ ][Q][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][Q][ ][ ]
[ ][ ][ ][ ][ ][Q][ ]
[ ][ ][ ][Q][ ][ ][ ]


Solution number 4: [7, 3, 8, 2, 5, 1, 6, 4]

[ ][ ][ ][ ][ ][Q][ ]
[ ][ ][Q][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][Q]
[ ][Q][ ][ ][ ][ ][ ]
[ ][ ][ ][Q][ ][ ][ ]
[Q][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][Q][ ][ ]
[ ][ ][Q][ ][ ][ ][ ]
```

```
Solution number 5: [8, 2, 5, 3, 1, 7, 4, 6]

[ ][ ][ ][ ][ ][ ][ ][Q]
[ ][Q][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][Q][ ][ ][ ]
[ ][ ][Q][ ][ ][ ][ ][ ]
[Q][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][Q][ ]
[ ][ ][ ][Q][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][Q][ ][ ]

Solution number 6: [8, 4, 1, 3, 6, 2, 7, 5]

[ ][ ][ ][ ][ ][ ][ ][Q]
[ ][ ][ ][Q][ ][ ][ ][ ]
[Q][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][Q][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][Q][ ][ ]
[ ][Q][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][Q][ ]
[ ][ ][ ][Q][ ][ ][ ][ ]
```

```
Solution number 7: [1, 7, 5, 8, 2, 4, 6, 3]

[Q][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][Q][ ]
[ ][ ][ ][ ][ ][Q][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][Q]
[ ][Q][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][Q][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][Q][ ][ ]
[ ][ ][Q][ ][ ][ ][ ][ ]


Solution number 8: [3, 5, 8, 4, 1, 7, 2, 6]

[ ][ ][Q][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][Q][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][Q]
[ ][ ][ ][Q][ ][ ][ ][ ]
[Q][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][Q][ ]
[ ][Q][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][Q][ ][ ]
```

```
Solution number 9: [2, 7, 5, 8, 1, 4, 6, 3]

[ ][Q][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][Q][ ]
[ ][ ][ ][ ][Q][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][Q]
[Q][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][Q][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][Q][ ][ ]
[ ][ ][Q][ ][ ][ ][ ][ ]


Solution number 10: [5, 7, 1, 3, 8, 6, 4, 2]

[ ][ ][ ][ ][Q][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][Q][ ]
[Q][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][Q][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][Q]
[ ][ ][ ][ ][ ][Q][ ][ ]
[ ][ ][ ][Q][ ][ ][ ][ ]
[ ][Q][ ][ ][ ][ ][ ][ ]
```

Question 2: Implement A* search algorithm using Python.

Python Code:

```
def aStarAlgo(start_node, stop_node):
    open_set = set(start_node)
    closed_set = set()
    g = {}
    parents = {}
    g[start_node] = 0
    parents[start_node] = start_node

    while len(open_set) > 0:
        n = None
        for v in open_set:
            if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):
                n = v
        if n == stop_node or Graph_nodes[n] == None:
            pass
        else:
            for (m, weight) in get_neighbors(n):
                if m not in open_set and m not in closed_set:
                    open_set.add(m)
                    parents[m] = n
                    g[m] = g[n] + weight
                else:
```

```
            if g[m] > g[n] + weight:
                g[m] = g[n] + weight
                parents[m] = n
                if m in closed_set:
                    closed_set.remove(m)
                    open_set.add(m)

if n == None:
    print('Path does not exist!')
    return None

if n == stop_node:
    path = []

    while parents[n] != n:
        path.append(n)
        n = parents[n]

    path.append(start_node)
    path.reverse()
    print('Path found: {}'.format(path))
    return path

open_set.remove(n)
closed_set.add(n)
```

```
    print('Path does not exist!')
    return None

def get_neighbors(v):
    if v in Graph_nodes:
        return Graph_nodes[v]
    else:
        return None

def heuristic(n):
    H_dist = {
        'A': 11,
        'B': 6,
        'C': 99,
        'D': 1,
        'E': 7,
        'G': 0,
    }
    return H_dist[n]
Graph_nodes = {
    'A': [('B', 2), ('E', 3)],
    'B': [('C', 1), ('G', 9)],
    'C': None,
    'E': [('D', 6)],
```

```
    'D': [('G', 1)],
}
aStarAlgo('A', 'G')
```

Offline3_2.py

```python
def aStarAlgo(start_node, stop_node):
    open_set = set(start_node)
    closed_set = set()
    g = {}
    parents = {}
    g[start_node] = 0
    parents[start_node] = start_node

    while len(open_set) > 0:
        n = None
        for v in open_set:
            if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):
                n = v
        if n == stop_node or Graph_nodes[n] == None:
            pass
        else:
            for (m, weight) in get_neighbors(n):
                if m not in open_set and m not in closed_set:
                    open_set.add(m)
                    parents[m] = n
                    g[m] = g[n] + weight
                else:
                    if g[m] > g[n] + weight:
                        g[m] = g[n] + weight
                        parents[m] = n
                        if m in closed_set:
                            closed_set.remove(m)
                            open_set.add(m)

        if n == None:
            print('Path does not exist!')
            return None

        if n == stop_node:
            path = []

            while parents[n] != n:
                path.append(n)
```

```
            n = parents[n]

        path.append(start_node)
        path.reverse()
        print('Path found: {}'.format(path))
        return path

    open_set.remove(n)
    closed_set.add(n)

print('Path does not exist!')
return None

def get_neighbors(v):
    if v in Graph_nodes:
        return Graph_nodes[v]
    else:
        return None

def heuristic(n):
    H_dist = {
        'A': 11,
        'B': 6,
        'C': 99,
        'D': 1,
        'E': 7,
        'G': 0,
    }
    return H_dist[n]
Graph_nodes = {
    'A': [('B', 2), ('E', 3)],
    'B': [('C', 1), ('G', 9)],
    'C': None,
    'E': [('D', 6)],
    'D': [('G', 1)],
}
aStarAlgo('A', 'G')
```

```
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.9.exe "D:\Python\PyCharm Community Edition 2021.3.1\
Connected to pydev debugger (build 213.6461.77)
Path found: ['A', 'E', 'D', 'G']

Process finished with exit code 0
```