Bahria University, Islamabad

Department of Software Engineering

Artificial Intelligence Lab

(Fall-2021)

Teacher: Engr. M Waleed Khan

Student : M Iqrar Ijaz Malik

Enrollment : 01-131182-021

Lab Journal: 6

Date: 10-11-2021

Task No:	Task Wise Marks		Documentation Marks		Total Marks (20)
	Assigned	Obtained	Assigned	Obtained	
1	15		5		

Comments:

Signature

LAB

Lab No: 6
IMPLEMENTING GRAPH ALGORITHS

Introduction

- Graphs are very useful data structures in solving many important mathematical challenges.
- For example computer network topology or analysing molecular structures of chemical compounds. They are also used in city traffic or route planning and even in human languages and their grammar

OBJECTIVE:

- Introduce the basics of python.
- Code in python.
- Implement Graph Algorithms in Python

Tools Used

Tool used to perform this task is **PyCharm Community Addition**

Task 1: Implement DFS and BFS

Code For BSF

```
class Graph:
    def __init__(self, name=""):
        self.name = name
        self.neighborList = {}
        self.nodeList = {}

    def add_node(self, node):
        self.nodeList[node] = True

    def add_edge(self, node, nodebis):
        try:
            self.neighborList[node].append(nodebis)
        except:
            self.neighborList[node] = []
            self.neighborList[node].append(nodebis)
        try:
            self.neighborList[nodebis].append(node)
        except:
            self.neighborList[nodebis] = []
            self.neighborList[nodebis].append(node)

    def neighbors(self, node):
        try:
            return self.neighborList[node]
        except:
            return []
```

```
            return []

    def nodes(self):
        return self.nodeList.keys()

    def delete_edge(self, node, nodebis):
        self.neighborList[node].remove(nodebis)
        self.neighborList[nodebis].remove(node)

    def delete_node(self, node):
        del self.nodeList[node]
        try:
            for nodebis in self.neighborList[node]:
                self.neighborList[nodebis].remove(node)
            del self.neighborList[node]
        except:
            return "error"
```

```
def BFS(graph, start, end):
    step_count = 0
    explored = set()
    frontiers = [start]
    new_path = []
    if start == end:
        print("Goal is at start :" + str(start))
    while frontiers:

        step_count += 1
        node = frontiers.pop(0)
        explored.add(node)
        new_path.append(node)

        if node == end:
            print("Breath First Search\n")
            print("Sequence Is : ", new_path)
            print("The Number of steps are : ", step_count)
            return

        neighbours = graph.neighbors(node)

        for neighbour in neighbours:
            if neighbour not in frontiers and neighbour not in explored:
                frontiers.append(neighbour)
    return "No such Node Exist "


if __name__ == "__main__":
    G = Graph("BFS")

    # Creating Graph
    G.add_node(1)
    G.add_node(2)
    G.add_node(3)
    G.add_node(4)
    G.add_node(5)
    G.add_node(6)
    G.add_node(7)
    G.add_node(8)
    G.add_node(9)
    G.add_node(10)
    G.add_node(11)
    G.add_node(12)
    G.add_edge(1, 2)
    G.add_edge(1, 7)
    G.add_edge(1, 8)
    G.add_edge(2, 3)
```

```
95      G.add_edge(3, 5)
96      G.add_edge(8, 9)
97      G.add_edge(8, 12)
98      G.add_edge(9, 10)
99      G.add_edge(9, 11)
100
101     # Finding the shortest Path using breath first search
102     BFS(G, 1, 5)
103
```

Screenshot

```
C:\Users\01-131182-021\AppData\Local\Microsoft\WindowsApps\python3.9.exe "F:/Study/Lectures Semister 7/AI/AI Assignment
Breath First Search

Sequence Is :  [1, 2, 7, 8, 3, 6, 9, 12, 4, 5]
The Number of steps are :  10

Process finished with exit code 0
```

- **Code For DFS**

```
class Graph:
    def __init__(self, name=""):
        self.name = name
        self.neighborList = {}
        self.nodeList = {}

    def add_node(self, node):
        self.nodeList[node] = True

    def add_edge(self, node, nodebis):
        try:
            self.neighborList[node].append(nodebis)
        except:
            self.neighborList[node] = []
            self.neighborList[node].append(nodebis)
        try:
            self.neighborList[nodebis].append(node)
        except:
            self.neighborList[nodebis] = []
            self.neighborList[nodebis].append(node)

    def neighbors(self, node):
        try:
            return self.neighborList[node]
        except:
            return []
```

```
    def nodes(self):
        return self.nodeList.keys()

    def delete_edge(self, node, nodebis):
        self.neighborList[node].remove(nodebis)
        self.neighborList[nodebis].remove(node)

    def delete_node(self, node):
        del self.nodeList[node]
        try:
            for nodebis in self.neighborList[node]:
                self.neighborList[nodebis].remove(node)
            del self.neighborList[node]
        except:
            return "error"

```

```
def DFS(graph, start, end):
    step_count = 0
    explored = set()
    frontiers = [start]
    new_path = []
    if start == end:
        return "The starting node is the Goal :" + start
    while frontiers:

        step_count += 1
        node = frontiers.pop()
        explored.add(node)
        new_path.append(node)

        if node == end:
            print("Depth First Search\n")
            print("Sequence Is : ", new_path)
            print("The Number of steps are : ", step_count)
            return

        neighbours = graph.neighbors(node)

        for neighbour in neighbours:
            if neighbour not in frontiers and neighbour not in explored:
```

```
            if neighbour not in frontiers and neighbour not in explored:
                frontiers.append(neighbour)
    return "No such Node Exist"


if __name__ == "__main__":
    G = Graph("DFS")

    # Creating Graph
    G.add_node(1)
    G.add_node(2)
    G.add_node(3)
    G.add_node(4)
    G.add_node(5)
    G.add_node(6)
    G.add_node(7)
    G.add_node(8)
    G.add_node(9)
    G.add_node(10)
    G.add_node(11)
    G.add_node(12)
    G.add_edge(1, 2)
    G.add_edge(1, 7)
    G.add_edge(1, 8)
    G.add_edge(2, 3)
```

```
91      G.add_edge(1, 8)
92      G.add_edge(2, 3)
93      G.add_edge(2, 6)
94      G.add_edge(3, 4)
95      G.add_edge(3, 5)
96      G.add_edge(8, 9)
97      G.add_edge(8, 12)
98      G.add_edge(9, 10)
99      G.add_edge(9, 11)
100
101     # Finding shortest Path
102     DFS(G, 1, 9)
103
```

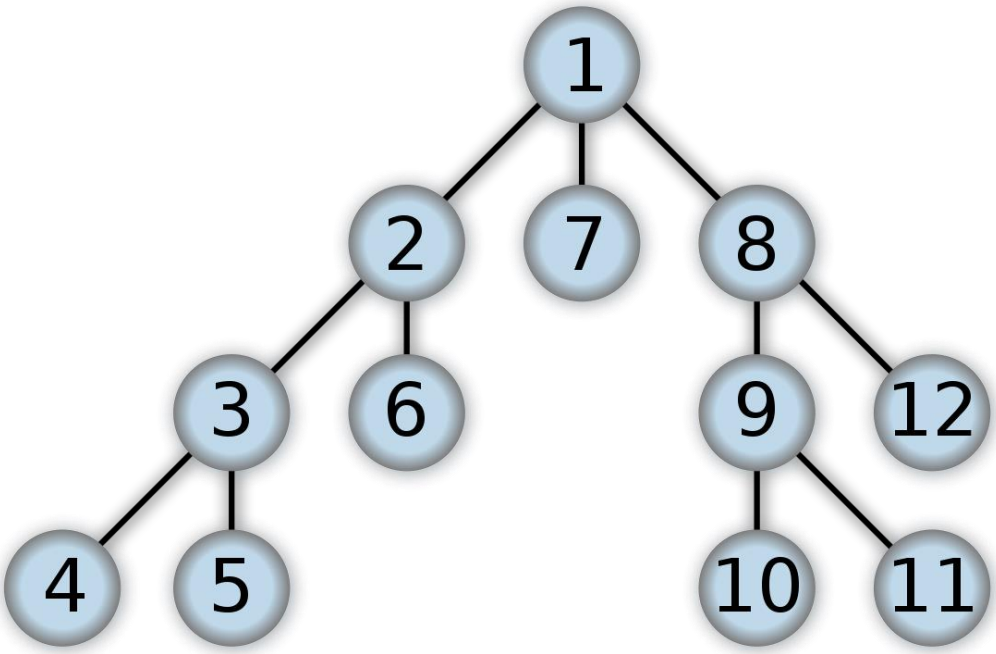
- **Output**

```
C:\Users\01-131182-021\AppData\Local\Microsoft\WindowsApps\python3.9.exe "F:/Study/Lectures Semister 7/AI/AI Assignment
Depth First Search

Sequence Is :  [1, 8, 12, 9]
The Number of steps are :  4

Process finished with exit code 0
```

- **Graph Used**



Conclusion

I completed the tasks given to us and pasted the output above.