PICT, PUNE

LP-II

Assigment-01

31118

DOP: 7-Jan'22
DOS: 13-Jan'22

Title: BFS and DFS algorithm.

Problem Statement:

Implement depth first search algorithm & Breadth first search algorithm, Use an undirected graph & develop a recursive algorithm for searching all the vertices of a graph or tree data structure

Objectives:

i) To learn about graph-search algorithms.

ii) To formalize & implement constraints in search problems.

Software & Hardware Requirements:

windows-10 OS (64-bit)

8-GB RAM & 512 GB SSD

VS-Code-latest version

Python-3.8

Theory-Related Concepts:

BFS:

1) Breadth-first-search for graph is a graph searching algorithm.

2) It starts from root node & explores all the neighbouring nodes.

Algorithm:

i) select a start node

ii) Create queue & enqueue start node

iii) while queue is not empty repeat step 4 & 5.

iv) Dequeu a node N & enqueue all neighbours of N that are not explored

v) set explored of N to true.

vi) exit.

time: O(V+E) space: O(E)

DFS:

1) It is graph searching algorithm.

2) It start from root node & go in recursive way. It uses stack data algorithm.

Algorithm:

i) Select any arbitary vertex & call dfs for that node

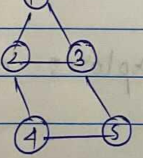
ii) mark current node as visited

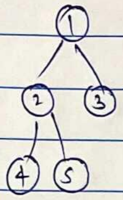
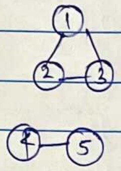
iii) for all adjacent unvisited nodes call dfs.

time complexity: O(V+E)

V ⇒ vertices E ⇒ Edges

Test cases

Graph	Expected	Actual	Result
	1 2 3 5 4 1 2 3 4 5	1 2 3 5 4 1 2 3 4 5	Pass

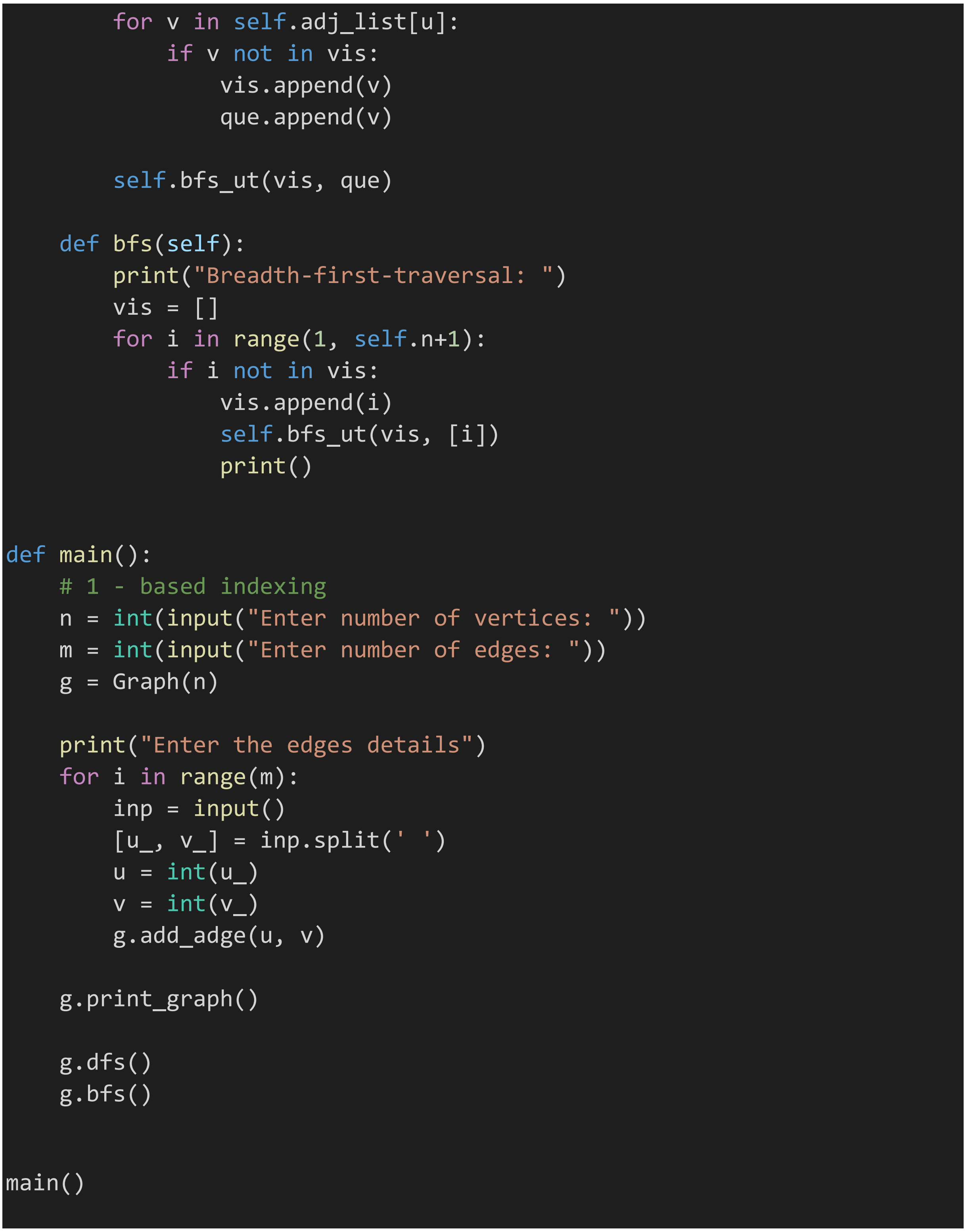
	1 2 3 4 5 1 2 3 4 5	1 2 3 4 5 1 2 3 4 5	Pass
	1 2 3 4 5 1 2 3 4 5	1 2 3 4 5 1 2 3 4 5	Pass.

Conclusion:

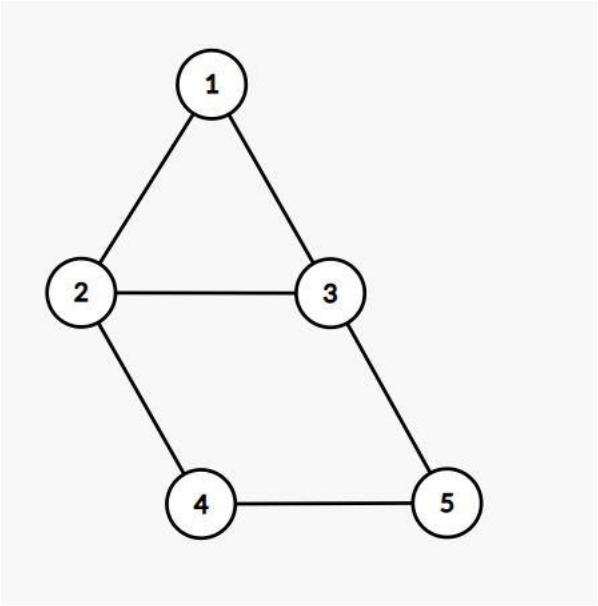
Through this assignment we have successfully implemented & learned graph search techniques.

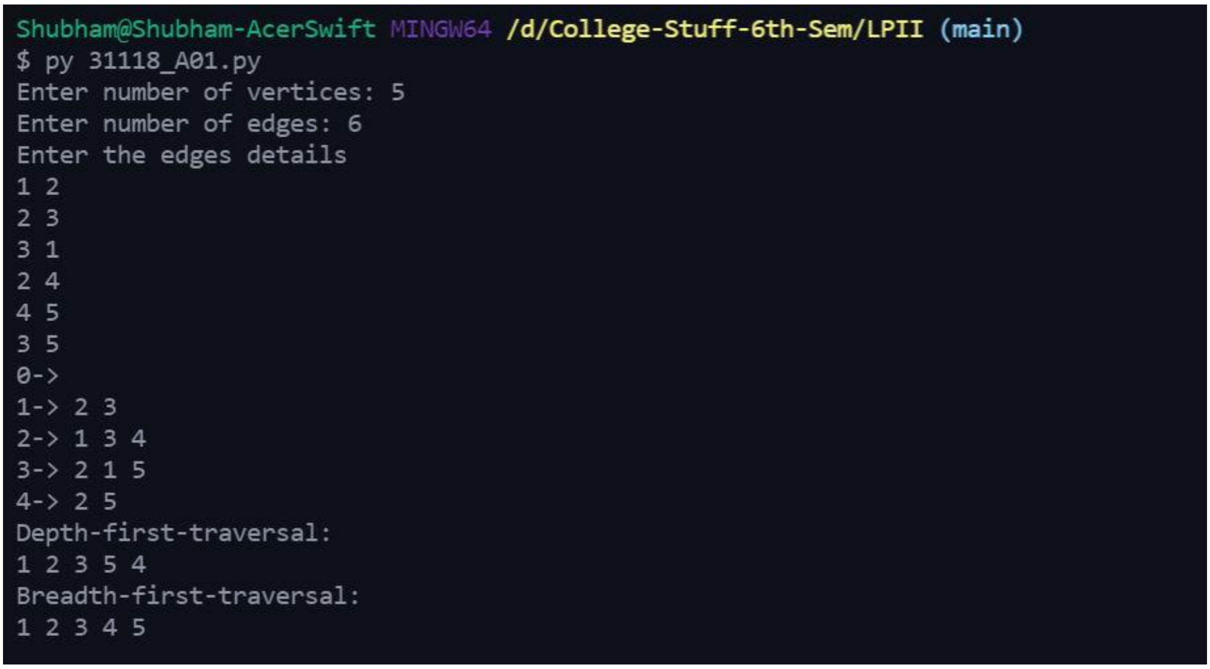
Code:

```
# Shubham - 31118

# undirected graph
class Graph:
    def __init__(self, n) -> None:
        self.n = n
        self.m = 0
        self.adj_list = [[] for _ in range(n+7)]

    def add_adge(self, u, v):
        self.m += 1
        self.adj_list[u].append(v)
        self.adj_list[v].append(u)

    def print_graph(self):
        print("Graph (Adjacency List Representation):")
        for u in range(self.n):
            print(u, end="-> ")
            for v in self.adj_list[u]:
                print(v, end=" ")
            print()

    def dfs_ut(self, u, vis):
        vis.append(u)
        print(u, end=" ")
        for v in self.adj_list[u]:
            if v not in vis:
                self.dfs_ut(v, vis)

    def dfs(self):
        print("Depth-first-traversal: ")
        vis = []
        for i in range(1, self.n+1):
            if i not in vis:
                self.dfs_ut(i, vis)
                print()

    def bfs_ut(self, vis, que):
        if not que:
            return

        u = que[0]
        que.pop(0)

        print(u, end=" ")
```

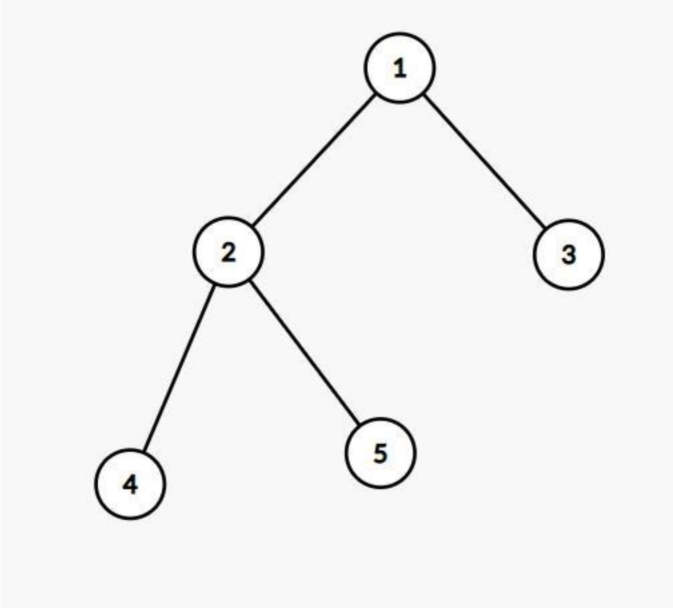


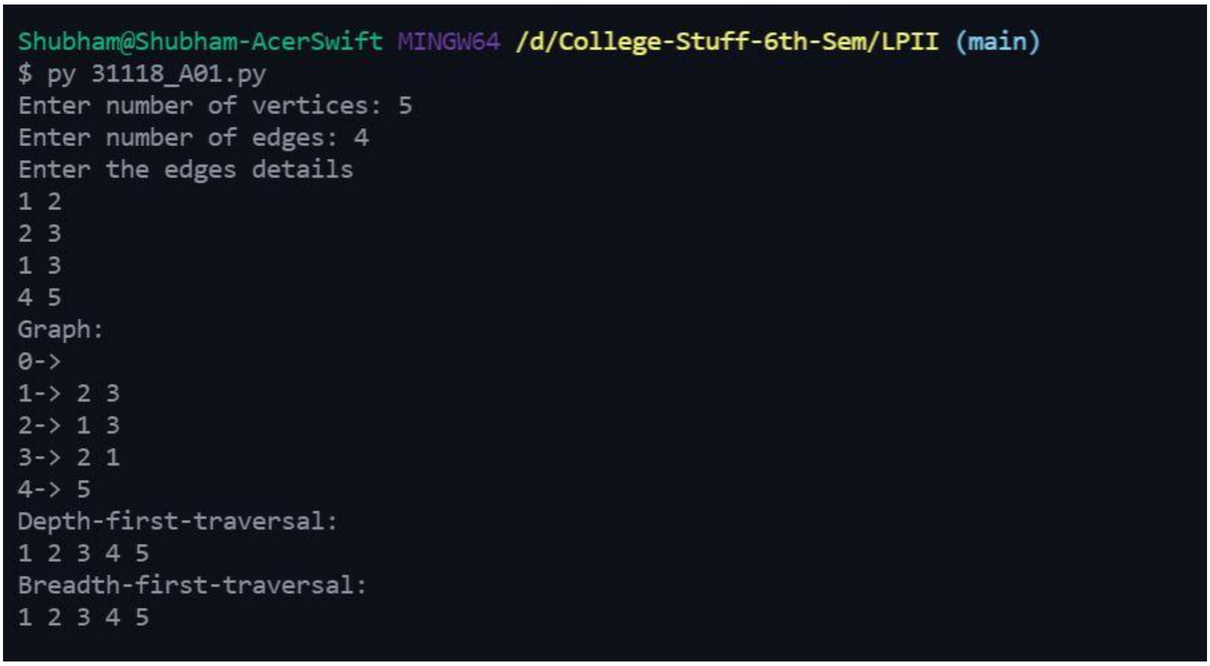
```
        for v in self.adj_list[u]:
            if v not in vis:
                vis.append(v)
                que.append(v)

        self.bfs_ut(vis, que)

    def bfs(self):
        print("Breadth-first-traversal: ")
        vis = []
        for i in range(1, self.n+1):
            if i not in vis:
                vis.append(i)
                self.bfs_ut(vis, [i])
                print()


def main():
    # 1 - based indexing
    n = int(input("Enter number of vertices: "))
    m = int(input("Enter number of edges: "))
    g = Graph(n)

    print("Enter the edges details")
    for i in range(m):
        inp = input()
        [u_, v_] = inp.split(' ')
        u = int(u_)
        v = int(v_)
        g.add_adge(u, v)

    g.print_graph()

    g.dfs()
    g.bfs()


main()
```



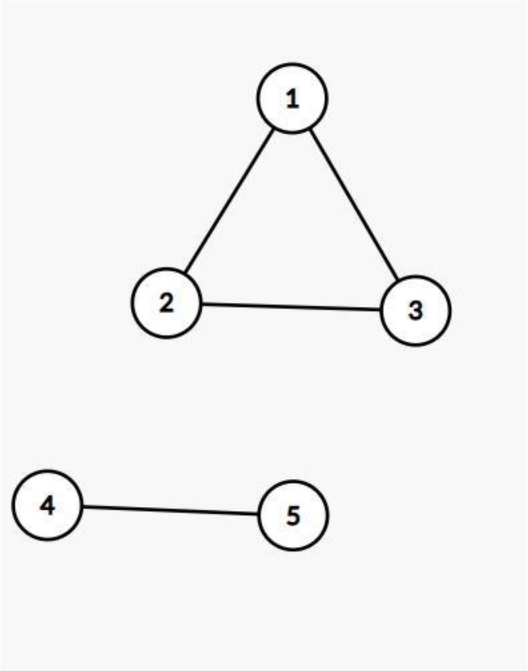


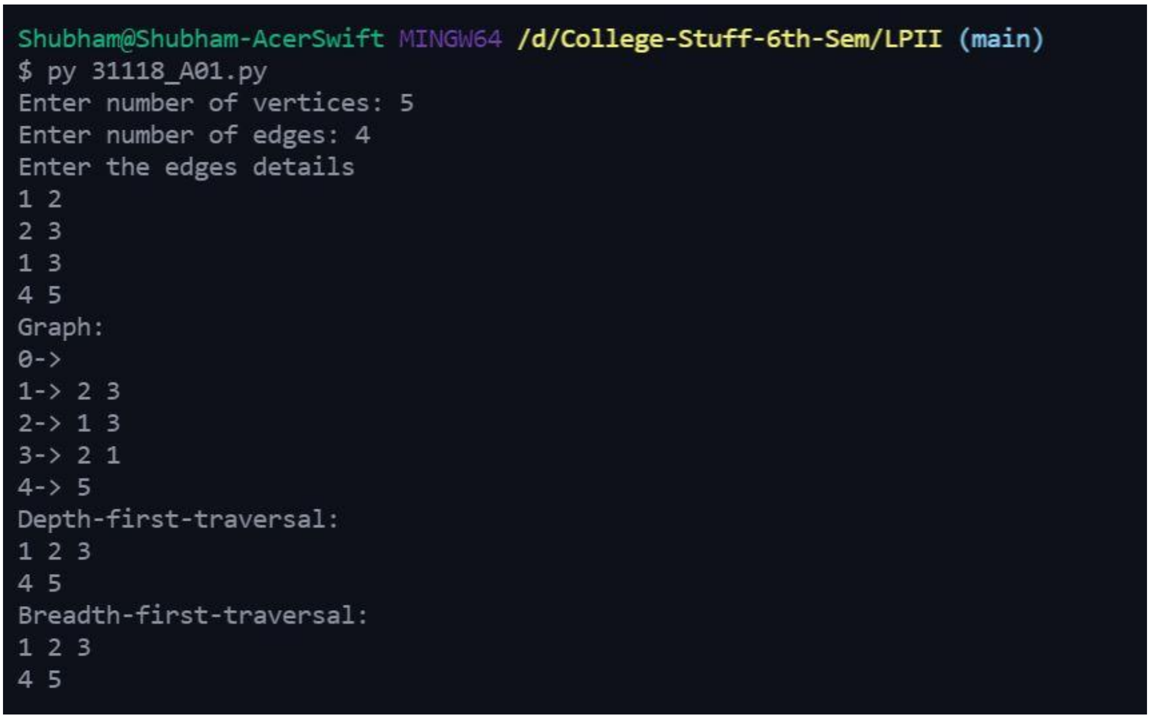
```
Shubham@Shubham-AcerSwift MINGW64 /d/College-Stuff-6th-Sem/LPII (main)
$ py 31118_A01.py
Enter number of vertices: 5
Enter number of edges: 6
Enter the edges details
1 2
2 3
3 1
2 4
4 5
3 5
0->
1-> 2 3
2-> 1 3 4
3-> 2 1 5
4-> 2 5
Depth-first-traversal:
1 2 3 5 4
Breadth-first-traversal:
1 2 3 4 5
```





```
Shubham@Shubham-AcerSwift MINGW64 /d/College-Stuff-6th-Sem/LPII (main)
$ py 31118_A01.py
Enter number of vertices: 5
Enter number of edges: 4
Enter the edges details
1 2
2 3
1 3
4 5
Graph:
0->
1-> 2 3
2-> 1 3
3-> 2 1
4-> 5
Depth-first-traversal:
1 2 3 4 5
Breadth-first-traversal:
1 2 3 4 5
```





```
Shubham@Shubham-AcerSwift MINGW64 /d/College-Stuff-6th-Sem/LPII (main)
$ py 31118_A01.py
Enter number of vertices: 5
Enter number of edges: 4
Enter the edges details
1 2
2 3
1 3
4 5
Graph:
0->
1-> 2 3
2-> 1 3
3-> 2 1
4-> 5
Depth-first-traversal:
1 2 3
4 5
Breadth-first-traversal:
1 2 3
4 5
```