Name : Aditya Wanjale
Roll no: 31282

LP-II Assignment No-1

- **TITLE :** DFS and BFS

- **PROBLEM STATEMENT :**

Implement depth first search algorithm and breadth first search algorithm. Use an undirected graph and develop a recusive algorithm for searching all the vertices of a graph or tree data structure.

- **LEARNING OBJECTIVES :**

1. To understands searching algorithms - depth first and breadth first
2. To make use of recursive function while implementation.

- **S/W AND H/W REQUIREMENTS :**

1. S/W - Pycharm (Python IDE), Python
2. HW - 64-bit Windows 10 OS

- **THEORY :**

1. Undirected Graph

- An undireeted graph is a set of nodes and a set of links between the nodes.. Each Node is called a vertex, each link called an edge, and each edge connets two vertices. The order of the two connected vertices is unimportant. An undirected graph is a finite set of vertices together with a finite set of edges.

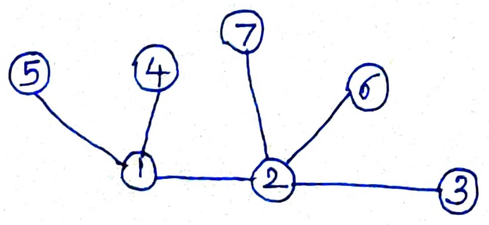
2. BFS

- Breadth first search for a graph is similar to breadth first traversal of a tree. Unlike trees, graphs may contain cycles, so we may come to the same node again. To avoid processing a node more than once, we use a boolean visited array. For simplicity, it is assumed that all vertices are reachable from the starting vertex.

3. DFS

- Depth first search for a graph is similar to depth first traversal of a tree. To avoid processing a node from than once, we used a boolean visited array. The algorithm starts at the root node and explores as far as possible along each branch before backtracking. The idea is to start from the root and mark the node and move to adjacent unmarked node and continue this loop until there is no unmarked adjacent node. Then backtrack and check for other unmarked nodes and traverse them. Finally print the nodes in path.

eg :- Consider the following graph -



BFS : 1, 2, 4, 5, 7, 3, 6

DFS : 1, 2, 3, 6, 7, 4, 5

- CONCLUSION :

Hence we implemented depth first and breadth first searching algorithms for an undirected graph using recursive functions.

Name – Aditya Wanjale

Roll no – 31282

Assignment No 01

Code :

```
class Graph :
  def __init__(self):
    self.graph = {}     # dictionary to store vertices and edges
    self.BSFsearch = []     # list to store BSF vertex list
    self.DSFsearch = []     # list to store DSF vertex list

  def add_vertex(self, vertex):
    if vertex not in self.graph :
      self.graph[vertex] = []     # vertex : key,  array of corresponding vertices : value
    else:
      print("Vertex " + vertex + " already present in graph.")

  def add_edge(self, vertex1, vertex2):
    if vertex1 and vertex2 in self.graph :
      self.graph[vertex1].append(vertex2)
      self.graph[vertex2].append(vertex1)
    else:
      print("These pair of vertex not present in graph.")

  def BSF(self, vertex):
    if vertex not in self.BSFsearch :
      self.BSFsearch.append(vertex)          # append vertex to list

    adjacentVertexList = self.graph[vertex]
    newAdjacentVertexList = []
    for j in adjacentVertexList:
            if j not in self.BSFsearch:
                newAdjacentVertexList.append(j)     # to avoid edge's vertex repeat ambiguity

        for i in adjacentVertexList :
            if i not in self.BSFsearch :
                self.BSFsearch.append(i)

        for i in newAdjacentVertexList :
            if i in self.BSFsearch :
                self.BSF(i)            # exlopre a vertex then move to next vertex

    def displayBSF(self):
        print("BSF order - ")
        for i in self.BSFsearch:
            print(i, end=", ")

        print()
        print("Vertices and adges - ", end=" ")
        print(self.graph)

    def DSF(self, vertex):
        if vertex not in self.DSFsearch :
            self.DSFsearch.append(vertex)          # append vertex to list

        adjacentVertexList = self.graph[vertex]
        newAdjacentVertexList = []

        for j in adjacentVertexList:
            if j not in self.DSFsearch:
                newAdjacentVertexList.append(j)     # to avoid edge's vertex repeat ambiguity
        for i in newAdjacentVertexList :
            self.DSF(i)          # exlopre a vertex till it ends then return to earlier vertex


    def displayDSF(self):
        print("DSF order - ")
        for i in self.DSFsearch:
            print(i, end=", ")


        print()
        print("Vertices and adges - ", end=" ")
        print(self.graph)


g=Graph();
while (True) :
    print("Menu")
    print("1. Add vertices\n2. Add edges\n3. Perform BSF\n4. Perform DSF")
    choice = int(input("Enter choice - "))
    if(choice==1):
        n = int(input("Enter total number of vertices - "))
        for i in range(0,n):
            a = int(input("Vertex - "))
            g.add_vertex(a)
    elif (choice==2) :
        v1 = int(input("Enter vertex 1 of edge - "))
        v2 = int(input("Enter vertex 2 of edge - "))
        g.add_edge(v1,v2)
    elif (choice==3) :
        g.BSF(1)
        g.displayBSF()
    elif (choice==4) :
        g.DSF(1)
        g.displayDSF()
    else :
        break
```

Breadth First Search Output :

```
C:\Users\Lenovo\PycharmProjects\SE\venv\Scripts\python.exe
C:/Users/Lenovo/PycharmProjects/SE/venv/31282_Aditya_LP2_Assign01.py

Menu

1. Add vertices

2. Add edges

3. Perform BSF

Enter choice - 1

Enter total number of vertices - 7

Vertex - 1

Vertex - 2

Vertex - 3

Vertex - 4

Vertex - 5

Vertex - 6

Vertex - 7

Menu

1. Add vertices

2. Add edges

3. Perform BSF

Enter choice - 2

Enter vertex 1 of edge - 1

Enter vertex 2 of edge - 2

Menu

1. Add vertices

2. Add edges
```

```
3. Perform BSF

Enter choice - 2

Enter vertex 1 of edge - 1

Enter vertex 2 of edge - 4

Menu

1. Add vertices

2. Add edges

3. Perform BSF

Enter choice - 2

Enter vertex 1 of edge - 1

Enter vertex 2 of edge - 5

Menu

1. Add vertices

2. Add edges

3. Perform BSF

Enter choice - 2

Enter vertex 1 of edge - 2

Enter vertex 2 of edge - 3

Menu

1. Add vertices

2. Add edges

3. Perform BSF

Enter choice - 2

Enter vertex 1 of edge - 2

Enter vertex 2 of edge - 6

Menu

1. Add vertices

2. Add edges

3. Perform BSF

Enter choice - 2

Enter vertex 1 of edge - 7
```

Enter vertex 2 of edge - 2

Menu

1. Add vertices

2. Add edges

3. Perform BSF

Enter choice - 3

BSF order -

1, 2, 4, 5, 3, 6, 7,

Vertices and adges - {1: [2, 4, 5], 2: [1, 3, 6, 7], 3: [2], 4: [1], 5: [1], 6: [2], 7: [2]}

Menu

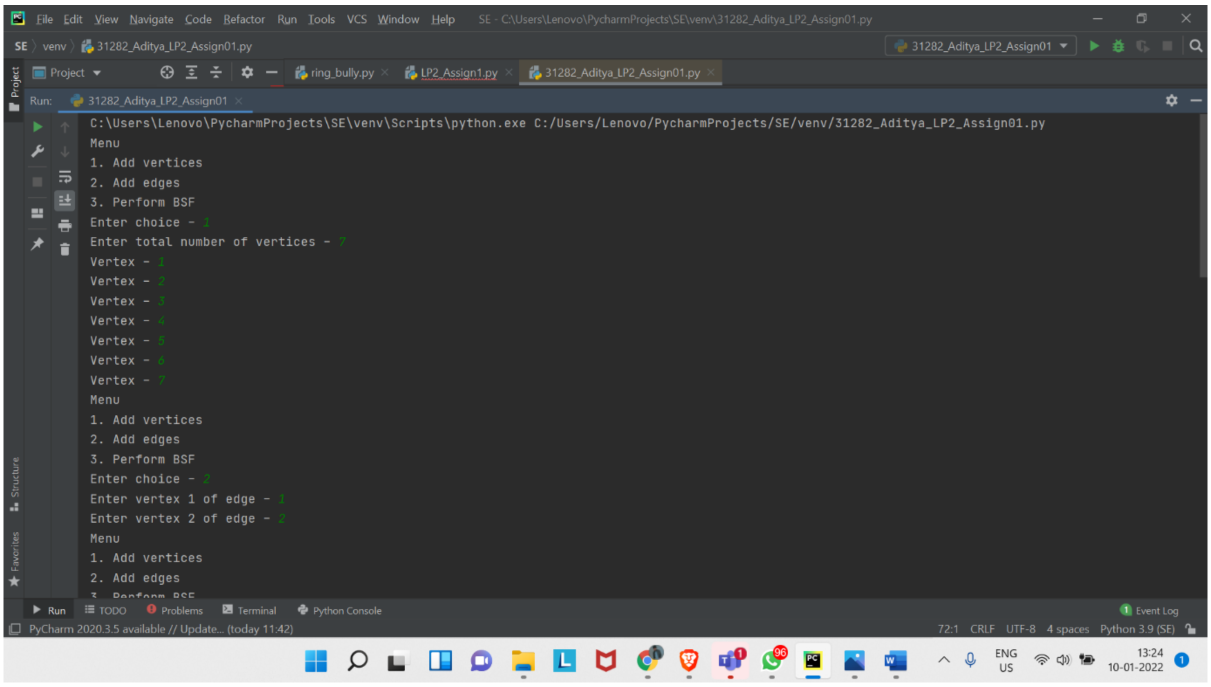
1. Add vertices

2. Add edges

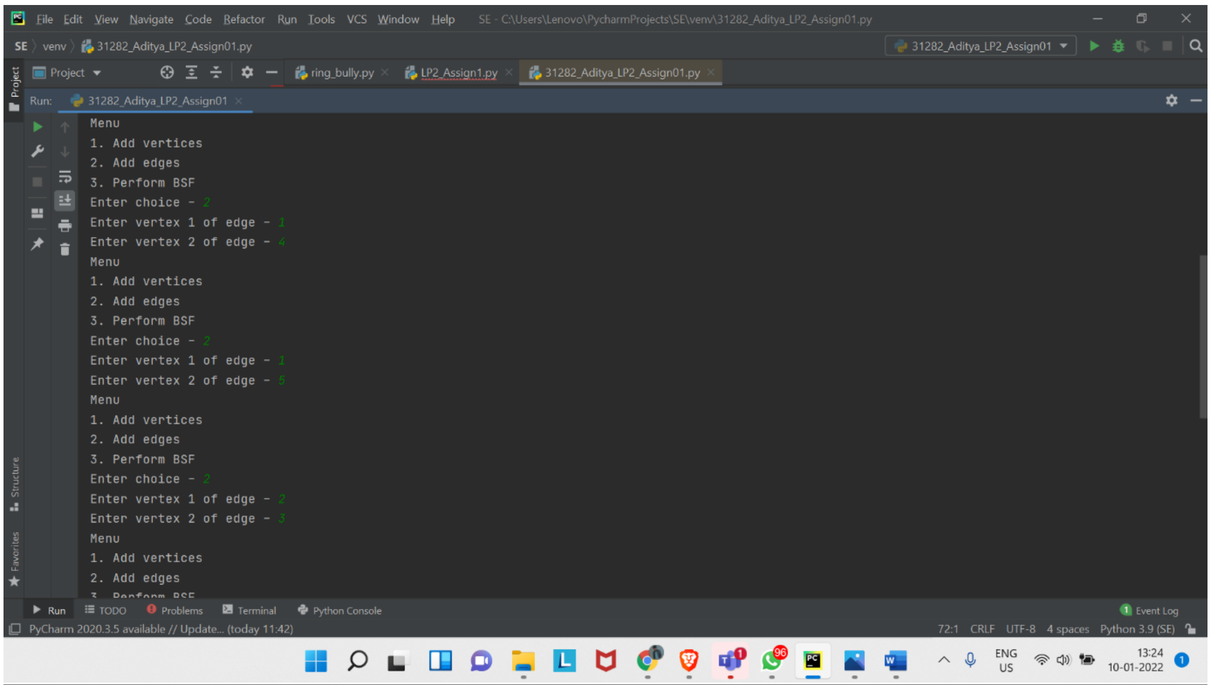
3. Perform BSF

Enter choice - 4

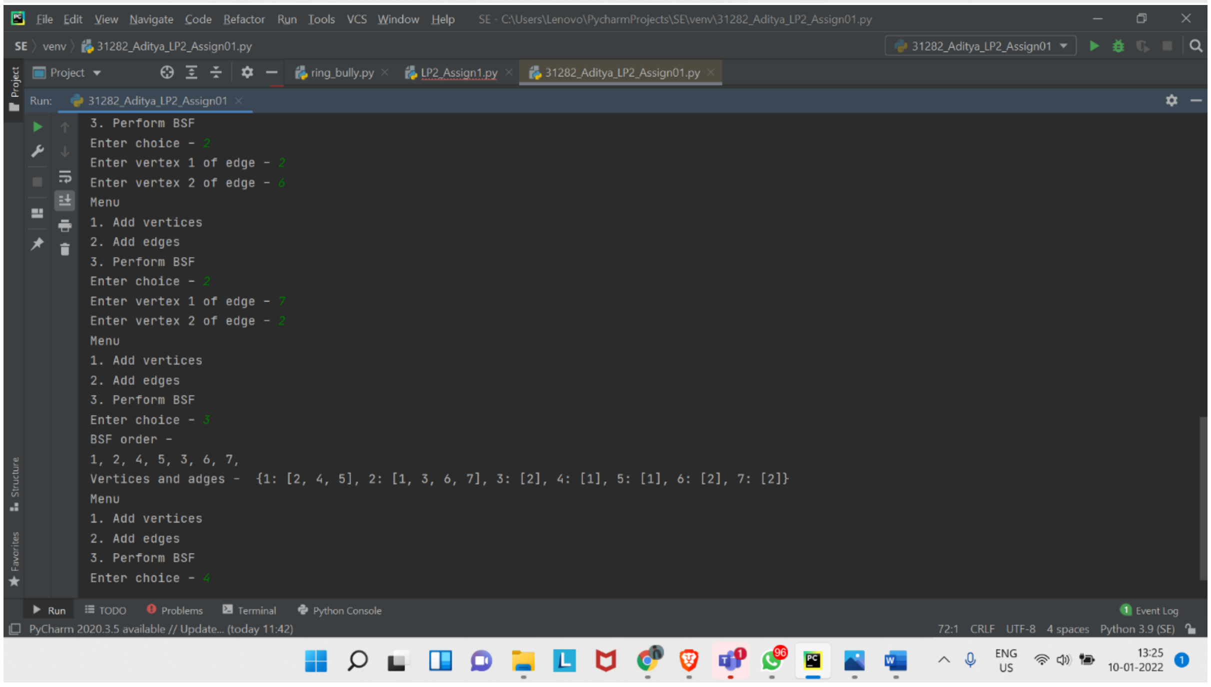
Process finished with exit code 0



```
C:\Users\Lenovo\PycharmProjects\SE\venv\Scripts\python.exe C:/Users/Lenovo/PycharmProjects/SE/venv/31282_Aditya_LP2_Assign01.py
Menu
1. Add vertices
2. Add edges
3. Perform BSF
Enter choice - 1
Enter total number of vertices - 7
Vertex - 1
Vertex - 2
Vertex - 3
Vertex - 4
Vertex - 5
Vertex - 6
Vertex - 7
Menu
1. Add vertices
2. Add edges
3. Perform BSF
Enter choice - 2
Enter vertex 1 of edge - 1
Enter vertex 2 of edge - 2
Menu
1. Add vertices
2. Add edges
3. Perform BSF
```



```
Menu
1. Add vertices
2. Add edges
3. Perform BSF
Enter choice - 2
Enter vertex 1 of edge - 1
Enter vertex 2 of edge - 4
Menu
1. Add vertices
2. Add edges
3. Perform BSF
Enter choice - 2
Enter vertex 1 of edge - 1
Enter vertex 2 of edge - 5
Menu
1. Add vertices
2. Add edges
3. Perform BSF
Enter choice - 2
Enter vertex 1 of edge - 2
Enter vertex 2 of edge - 3
Menu
1. Add vertices
2. Add edges
```



```
3. Perform BSF
Enter choice - 2
Enter vertex 1 of edge - 2
Enter vertex 2 of edge - 6
Menu
1. Add vertices
2. Add edges
3. Perform BSF
Enter choice - 2
Enter vertex 1 of edge - 7
Enter vertex 2 of edge - 2
Menu
1. Add vertices
2. Add edges
3. Perform BSF
Enter choice - 3
BSF order -
1, 2, 4, 5, 3, 6, 7,
Vertices and adges -  {1: [2, 4, 5], 2: [1, 3, 6, 7], 3: [2], 4: [1], 5: [1], 6: [2], 7: [2]}
Menu
1. Add vertices
2. Add edges
3. Perform BSF
Enter choice - 4
```

Depth First Search Output :

```
C:\Users\Lenovo\PycharmProjects\SE\venv\Scripts\python.exe
C:/Users/Lenovo/PycharmProjects/SE/venv/31282_Aditya_LP2_Assign01.py

Menu

1. Add vertices

2. Add edges
```

```
3. Perform BSF

4. Perform DSF

Enter choice - 1

Enter total number of vertices - 7

Vertex - 1

Vertex - 2

Vertex - 3

Vertex - 4

Vertex - 5

Vertex - 6

Vertex - 7

Menu

1. Add vertices

2. Add edges

3. Perform BSF

4. Perform DSF

Enter choice - 2

Enter vertex 1 of edge - 1

Enter vertex 2 of edge - 2

Menu

1. Add vertices

2. Add edges

3. Perform BSF

4. Perform DSF

Enter choice - 2

Enter vertex 1 of edge - 1

Enter vertex 2 of edge - 4

Menu

1. Add vertices

2. Add edges

3. Perform BSF
```

```
4. Perform DSF
Enter choice - 2
Enter vertex 1 of edge - 1
Enter vertex 2 of edge - 5
Menu
1. Add vertices
2. Add edges
3. Perform BSF
4. Perform DSF
Enter choice - 2
Enter vertex 1 of edge - 2
Enter vertex 2 of edge - 3
Menu
1. Add vertices
2. Add edges
3. Perform BSF
4. Perform DSF
Enter choice - 2
Enter vertex 1 of edge - 2
Enter vertex 2 of edge - 6
Menu
1. Add vertices
2. Add edges
3. Perform BSF
4. Perform DSF
Enter choice - 2
Enter vertex 1 of edge - 2
Enter vertex 2 of edge - 7
Menu
1. Add vertices
2. Add edges
```

3. Perform BSF

4. Perform DSF

Enter choice - 4

DSF order -

1, 2, 3, 6, 7, 4, 5,

Vertices and adges - {1: [2, 4, 5], 2: [1, 3, 6, 7], 3: [2], 4: [1], 5: [1], 6: [2], 7: [2]}

Menu

1. Add vertices

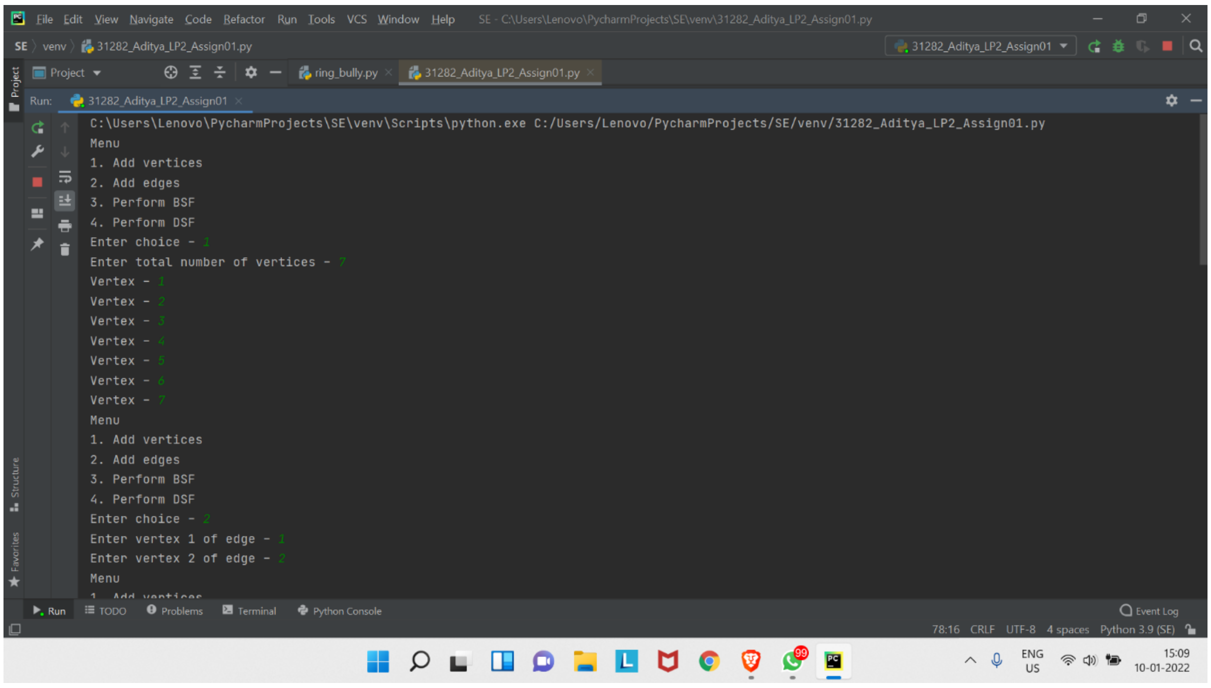
2. Add edges

3. Perform BSF

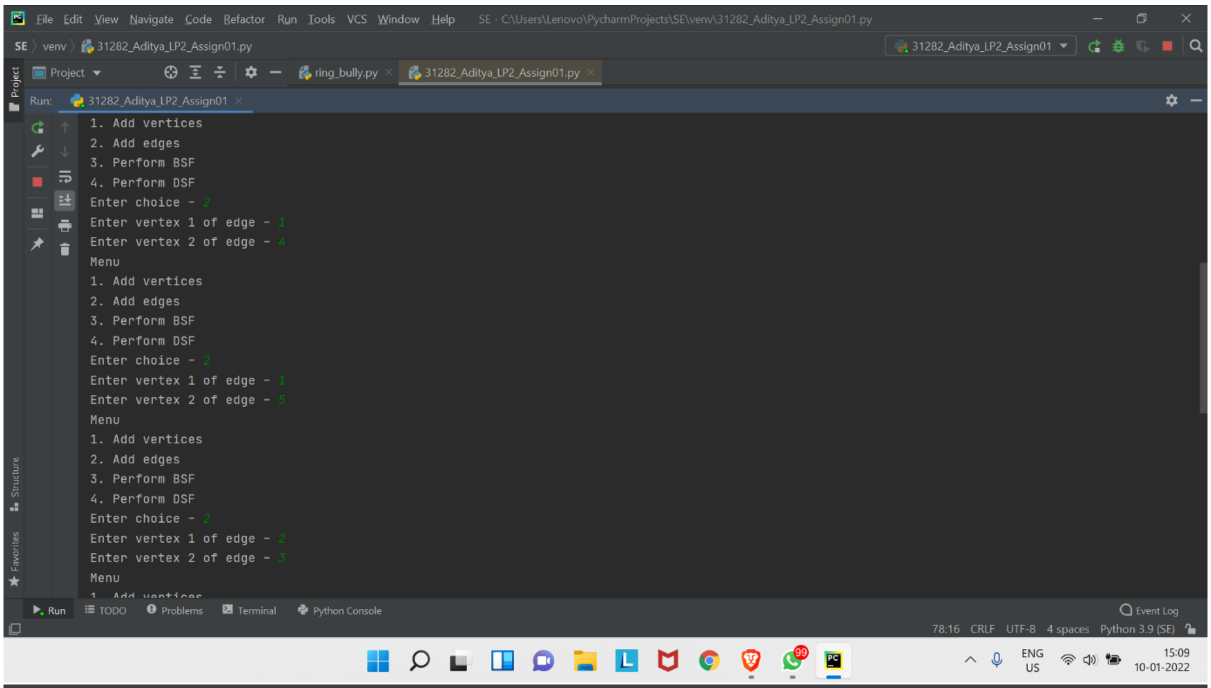
4. Perform DSF

Enter choice - 4

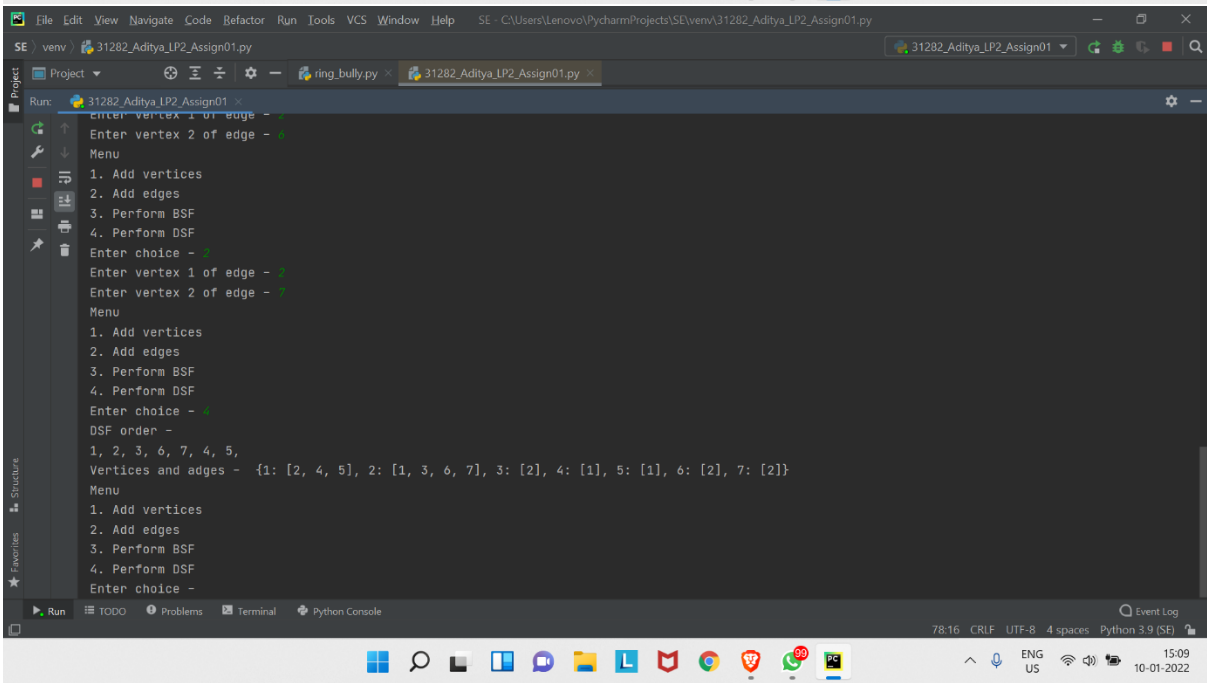
Process finished with exit code 0


File Edit View Navigate Code Refactor Run Tools VCS Window Help
SE - C:\Users\Lenovo\PycharmProjects\SE\venv\31282_Aditya_LP2_Assign01.py
SE venv 31282_Aditya_LP2_Assign01.py
Project
ring_bully.py
31282_Aditya_LP2_Assign01.py
Run: 31282_Aditya_LP2_Assign01
Run TODO Problems Terminal Python Console Event Log
78:16 CRLF UTF-8 4 spaces Python 3.9 (SE)
ENG US
15:09
10-01-2022

```
C:\Users\Lenovo\PycharmProjects\SE\venv\Scripts\python.exe C:/Users/Lenovo/PycharmProjects/SE/venv/31282_Aditya_LP2_Assign01.py
Menu
1. Add vertices
2. Add edges
3. Perform BSF
4. Perform DSF
Enter choice - 1
Enter total number of vertices - 7
Vertex - 1
Vertex - 2
Vertex - 3
Vertex - 4
Vertex - 5
Vertex - 6
Vertex - 7
Menu
1. Add vertices
2. Add edges
3. Perform BSF
4. Perform DSF
Enter choice - 2
Enter vertex 1 of edge - 1
Enter vertex 2 of edge - 2
Menu
1. Add vertices
```



```
1. Add vertices
2. Add edges
3. Perform BSF
4. Perform DSF
Enter choice - 2
Enter vertex 1 of edge - 1
Enter vertex 2 of edge - 4
Menu
1. Add vertices
2. Add edges
3. Perform BSF
4. Perform DSF
Enter choice - 2
Enter vertex 1 of edge - 1
Enter vertex 2 of edge - 5
Menu
1. Add vertices
2. Add edges
3. Perform BSF
4. Perform DSF
Enter choice - 2
Enter vertex 1 of edge - 2
Enter vertex 2 of edge - 3
Menu
```



```
Enter vertex 2 of edge - 6
Menu
1. Add vertices
2. Add edges
3. Perform BSF
4. Perform DSF
Enter choice - 2
Enter vertex 1 of edge - 2
Enter vertex 2 of edge - 7
Menu
1. Add vertices
2. Add edges
3. Perform BSF
4. Perform DSF
Enter choice - 4
DSF order -
1, 2, 3, 6, 7, 4, 5,
Vertices and adges -  {1: [2, 4, 5], 2: [1, 3, 6, 7], 3: [2], 4: [1], 5: [1], 6: [2], 7: [2]}
Menu
1. Add vertices
2. Add edges
3. Perform BSF
4. Perform DSF
Enter choice -
```