K. J. Somaiya College of Engineering, Mumbai-77
(A constituent College of Somaiya Vidyavihar University)

Batch:C1 Roll No:16010122257

Experiment No.9

Grade: AA / AB / BB / BC / CC / CD /DD

Title: Study of Graph traversal methods DFS and BFS

Objective: To understand graph as data structure and methods of traversing Graph

Expected Outcome of Experiment:

CO	Outcome
CO2	Apply linear and non-linear data structure in application development

Websites/books referred:

1. ***https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/***
2. **Ma'am's classroom notes**
3. **https://www.geeksforgeeks.org/difference-between-bfs-and-dfs/**

Abstract: - (Definition of Graph, types of graphs, and difference and similarity between graph & tree)

Graph: A data structure that consists of a set of nodes

(vertices) and a set of edges that relate the

nodes to each other.

The set of edges describes relationships among the vertices

Formal definition of graphs

A graph G is defined as follows:

G=(V,E)

V(G): a finite, nonempty set of vertices

E(G): a set of edges (pairs of vertices)

Vertices are also called nodes and points.

Each edge connects two vertices.

Edges are also called arcs and lines.

Vertices i and j are adjacent vertices iff (i, j) is an edge in the graph

The edge (i, j) is incident on the vertices i and j

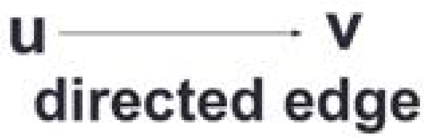
Types of Graph:

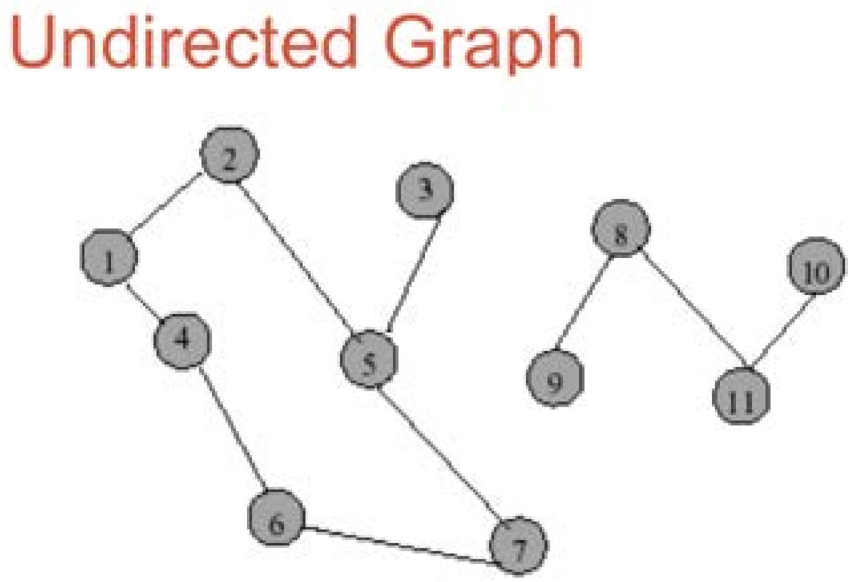
Undirected edge has no orientation (no arrow head)

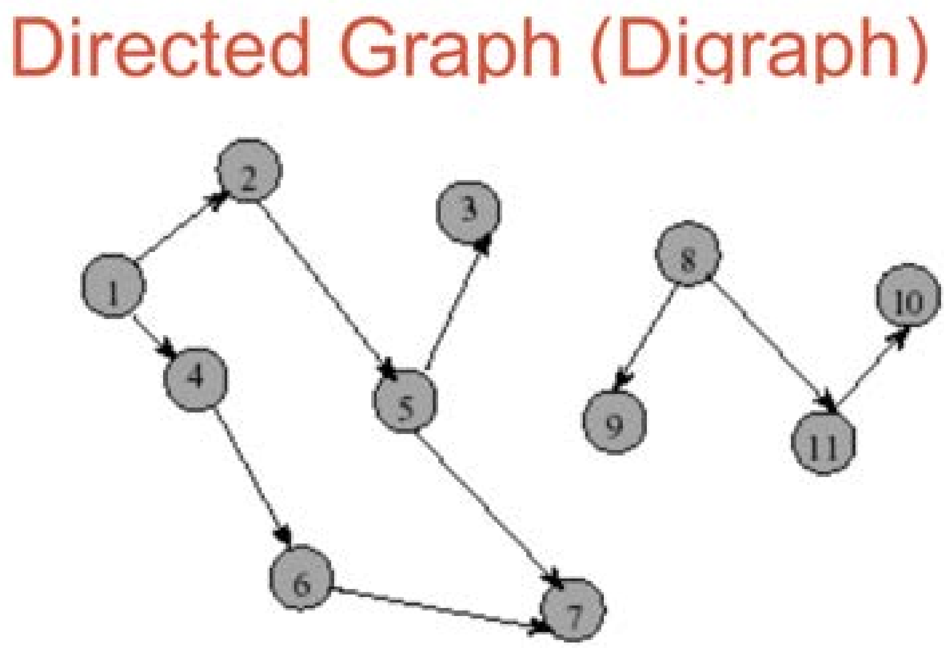
Directed edge has an orientation (has an arrow head)

Undirected graph – all edges are undirected

Directed graph – all edges are directed

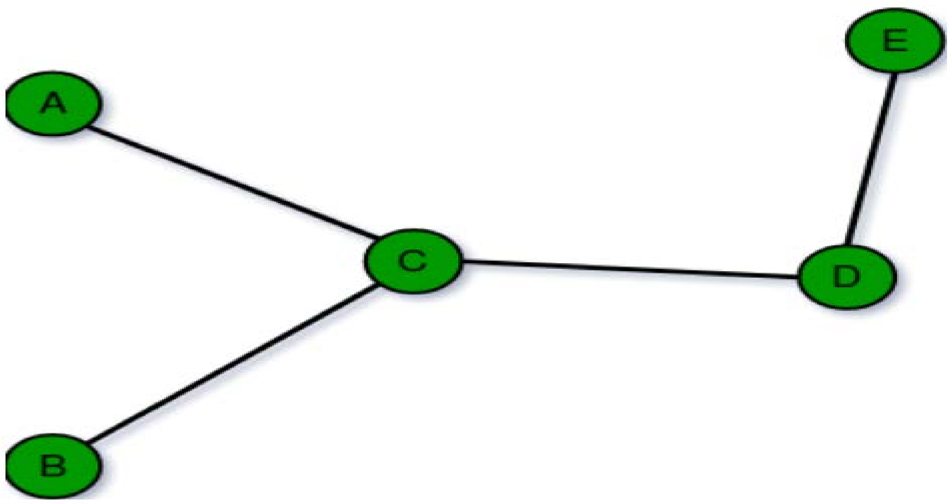






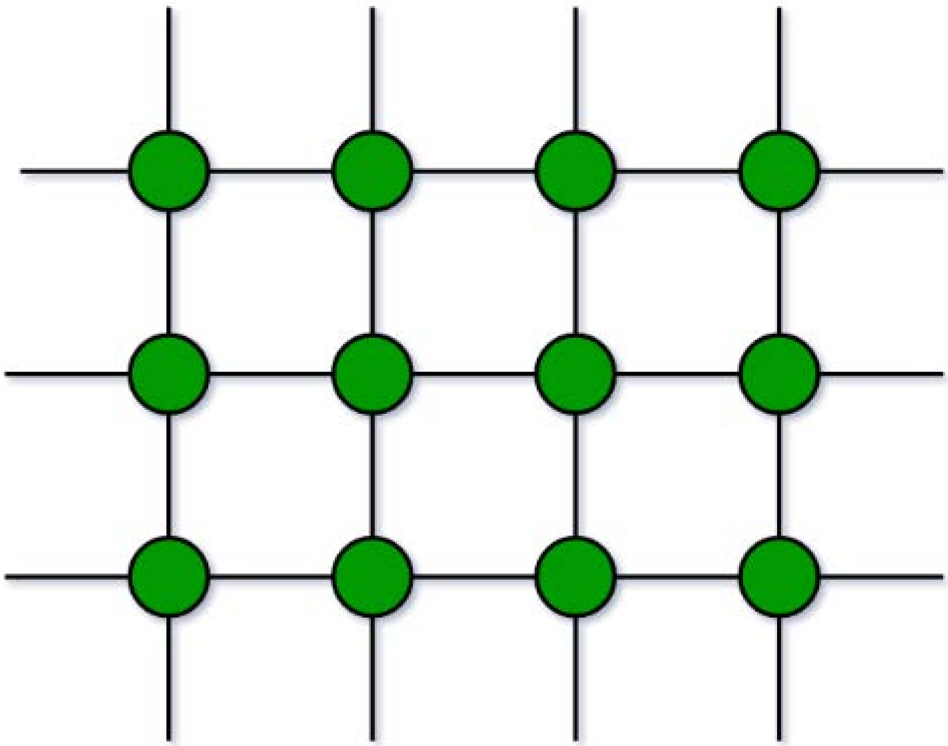
1. Finite Graphs

A graph is said to be finite if it has a finite number of vertices and a finite number of edges. A finite graph is a graph with a finite number of vertices and edges. In other words, both the number of vertices and the number of edges in a finite graph are limited and can be counted.



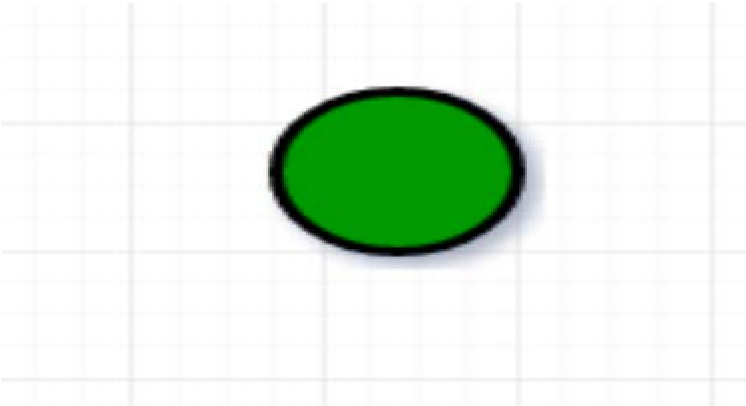
2. Infinite Graph:

A graph is said to be infinite if it has an infinite number of vertices as well as an infinite number of edges.



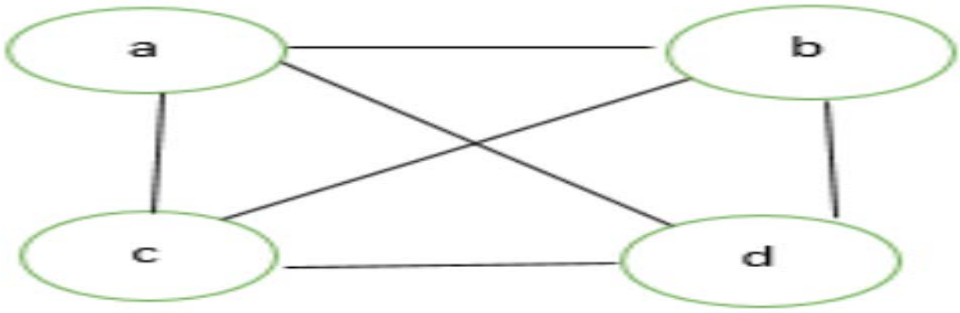
3. Trivial Graph:

A graph is said to be trivial if a finite graph contains only one vertex and no edge. A trivial graph is a graph with only one vertex and no edges. It is also known as a singleton graph or a single vertex graph. A trivial graph is the simplest type of graph and is often used as a starting point for building more complex graphs. In graph theory, trivial graphs are considered to be a degenerate case and are not typically studied in detail



4. Simple Graph:

A simple graph is a graph that does not contain more than one edge between the pair of vertices. A simple railway track connecting different cities is an example of a simple graph.

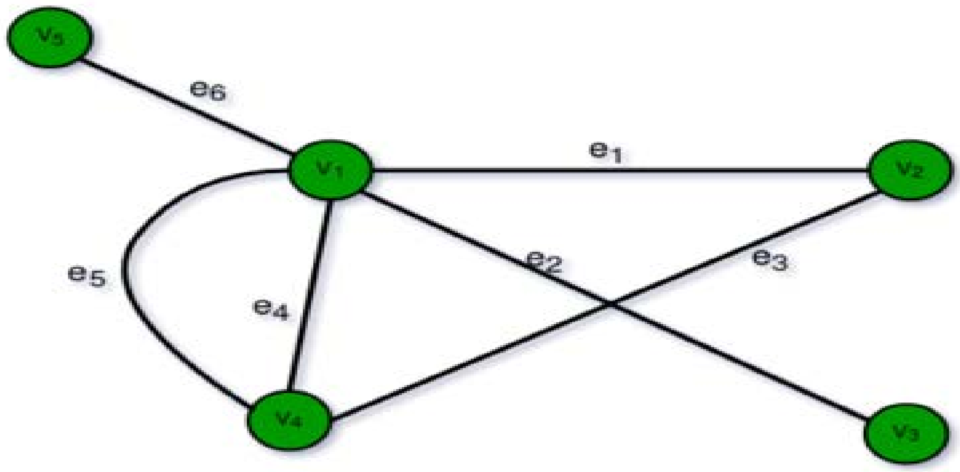


5. Multi Graph:

Any graph which contains some parallel edges but doesn't contain any self-loop is called a multigraph. For example a Road Map.

Parallel Edges: If two vertices are connected with more than one edge then such edges are called parallel edges that are many routes but one destination.

Loop: An edge of a graph that starts from a vertex and ends at the same vertex is called a loop or a self-loop.



:

Difference between graph and tree:

Trees vs graphs

A graph is a group of vertices and edges where an edge connects a pair of vertices whereas a tree is considered as a minimally connected graph which must be connected and free from loops.

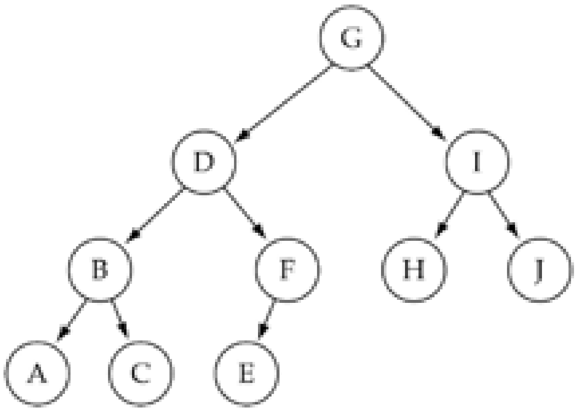
Acyclicity: Trees are a specific type of graph that is acyclic, meaning there are no cycles (loops) in a tree. In contrast, graphs can be cyclic or acyclic. A graph becomes a tree if and only if there is exactly one path between every pair of nodes.

Root Node: Trees have a root node from which all other nodes are descendants. Graphs do not have a concept of a root node, and nodes can be connected in any arbitrary way.

Similarity between graph and tree:

Trees are special cases of graphs!!

(c) Graph3 is a directed graph.



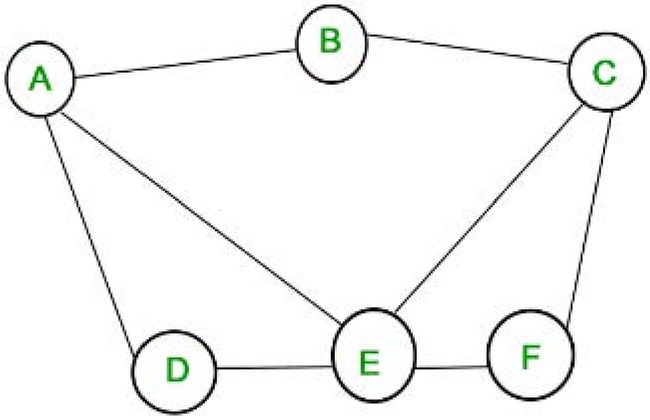
V(Graph3) = { A, B, C, D, E, F, G, H, I, J }
E(Graph3) = { (G, D), (G, J), (D, B), (D, F) (I, H), (I, J), (B, A), (B, C), (F, E) }

Hierarchical Structure: Both trees and certain types of graphs can represent hierarchical relationships. In a tree, each node has a parent-child relationship, forming a hierarchy. Similarly, certain types of graphs, like tree graphs or hierarchical graphs, also exhibit hierarchical relationships among nodes.

Nodes and Edges: Both trees and graphs consist of nodes (or vertices) and edges. Nodes represent entities or elements, and edges represent relationships or connections between nodes.

Solution of Assigned numerical problems using BFS and DFS:

Q. 16



Starting vertex: E

Q16.

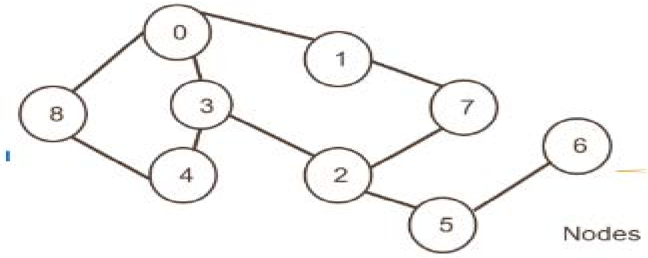
Queue	Marked set	Current	BFS
{ }	{ }	—	—
{E}	{E}	E	E
{ACDF}	{EACDF}	A	EA
{CDFB}	{EACDFB}	C	EAC
{DFB}	{EACDFB}	D	EACD
{FB}	{EACDFB}	F	EACDF
{B}	{EACDFB}	B	EACDFB

BFS= EACDFB

Stack	Marked set	Current	DFS
{ }	{ }	—	—
{E}	{E}	E	E
{ACDF}	{EACDF}	A	EA
{BCDF}	{EACDFB}	B	EAB
{CDF}	{EACDFB}	C	EABC
{DF}	{EACDFB}	D	EABCD
{F}	{EACDFB}	F	EABCDF

DFS= EABCDF

Q. 8



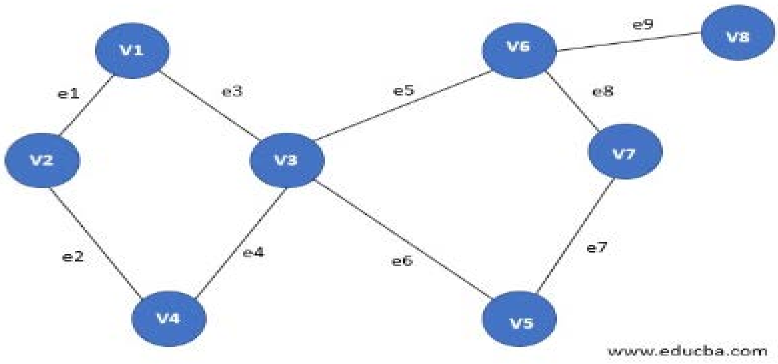
Starting vertex: 5

Queue	Marked set	Current	BFS
<>	<>	—	—
<5>	<5>	5	5
<2,6>	<5,2,6>	2	5,2
<6,3,7>	<5,2,6,3,7>	6	5,2,6
<3,7>	<5,2,6,3,7>	3	5,2,6,3
<7,0,4>	<5,2,6,3,7,0,4>	7	5,2,6,3,7
<0,4,1>	<5,2,6,3,7,0,4>	0	5,2,6,3,7,0
<4,1,8>	<5,2,6,3,7,0,4,1,8>	4	5,2,6,3,7,0,4
<1,8>	<5,2,6,3,7,0,4,1,8>	1	5,2,6,3,7,0,4,1
<8>	"	8	5,2,6,3,7,0,4,1,8

Queue	Marked set	Current	DFS
<>	<>	—	—
<5>	<5>	5	5
<2,6>	<5,2,6>	2	5,2
<3,7,6>	<5,2,6,3,7>	3	5,2,3
<0,4,7,6>	<5,2,6,3,7,0,4>	0	5,2,3,0
<1,8,4,7,6>	<5,2,6,3,7,0,4,1,8>	1	5,2,3,0,1
<8,4,7,6>	<5,2,6,3,7,0,4,1,8>	8	5,2,3,0,1,8
<4,7,6>	"	4	5,2,3,0,1,8,4
<7,6>	"	7	5,2,3,0,1,8,4,7
<6>	"	6	5,2,3,0,1,8,4,7,6

BFS= 5,2,6,3,7,0,4,1,8
DFS= 5,2,3,0,1,8,4,7,6

Q. 5



Starting vertex:v7

5.

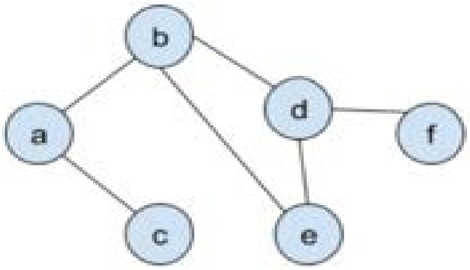
Queue	Marked set	Current	BFS
{ }	{ }	—	—
{v7}	{v7}	v7	v7
{v5, v6}	{v7, v5, v6}	v5	v7, v5
{v6, v3}	{v7, v5, v6, v3}	v6	v7, v5, v6
{v3, v8}	{v7, v5, v6, v3, v8}	v3	v7, v5, v6, v3
{v8, v1, v4}	{v7, v5, v6, v3, v8, v1, v4}	v8	v7, v5, v6, v3, v8
{v1, v4}	"	v1	v7, v5, v6, v3, v8, v1
{v4, v2}	{v7, v5, v6, v3, v8, v1, v4, v2}	v4	v7, v5, v6, v3, v8, v1, v4
{v2}	{v7, v5, v6, v3, v8, v1, v4, v2}	v2	v7, v5, v6, v3, v8, v1, v4, v2

Queue	Marked set	Current	DFS
{ }	{ }	—	—
{v7}	{v7}	v7	v7
{v5, v6}	{v7, v5, v6}	v5	v7, v5
{v3, v6}	{v7, v5, v6, v3}	v3	v7, v5, v3
{v1, v4, v6}	{v7, v5, v6, v3, v1, v4}	v1	v7, v5, v3, v1
{v2, v4, v6}	{v7, v5, v6, v3, v1, v4, v2}	v2	v7, v5, v3, v1, v2
{v4, v6}	{v7, v5, v6, v3, v1, v4, v2}	v4	v7, v5, v3, v1, v2, v4
{v8, v6}	{v7, v5, v6, v3, v1, v4, v2, v8}	v8	v7, v5, v3, v1, v2, v4, v8
{v6}	"	v6	v7, v5, v3, v1, v2, v4, v8, v6

BFS = v7, v5, v6, v3, v8, v1, v4, v2

DFS = v7, v5, v3, v1, v2, v4, v8, v6

Q. 18



Graph

Starting vertex: d

Queue	Marked set	Current	BFS
< >	< >	—	—
<d>	<d>	d	d
<b, e, f>	<d, b, e, f>	b	d, b
<e, f, a>	<d, b, e, f, a>	e	d, b, e
<f, a>	<d, b, e, f, a>	f	d, b, e, f
<a, c>	<d, b, e, f, a, c>	a	d, b, e, f, a
<c>	"	c	d, b, e, f, a, c

BFS = d b e f a c

Queue	Marked set	Current	DFS
< >	< >	—	—
<d>	<d>	d	d
<b, e, f>	<d, b, e, f>	b	d, b
<a, e, f>	<d, b, a, e, f>	a	d, b, a
<c, e, f>	<d, b, a, c, e, f>	c	d, b, a, c
<e, f>	"	e	d, b, a, c, e
<f>	"	f	d, b, a, c, e, f

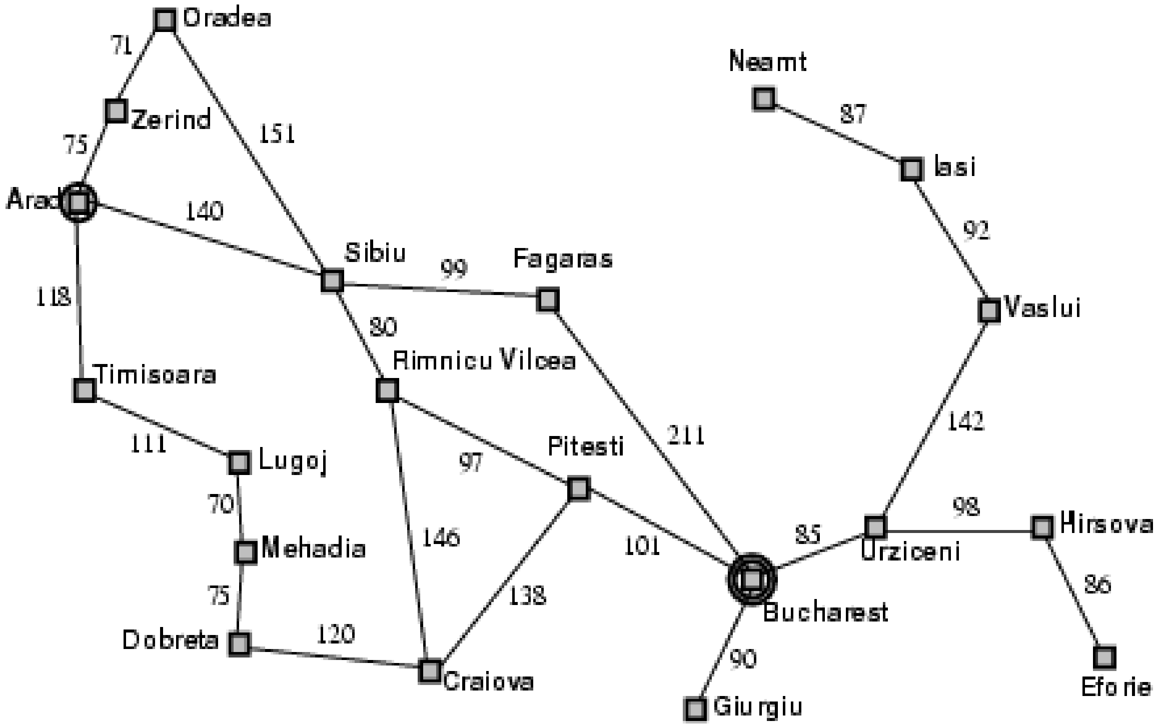
DFS = d b a c e f

Post lab questions-

a. **Differentiate between BFS and DFS.**

S. No.	Parameters	BFS	DFS
1.	Stands for	BFS stands for Breadth First Search.	DFS stands for Depth First Search.
2.	Data Structure	BFS(Breadth First Search) uses Queue data structure for finding the shortest path.	DFS(Depth First Search) uses Stack data structure.
3.	Definition	BFS is a traversal approach in which we first walk through all nodes on the same level before moving on to the next level.	DFS is also a traversal approach in which the traverse begins at the root node and proceeds through the nodes as far as possible until we reach the node with no unvisited nearby nodes.
4.	Technique	BFS can be used to find a single source shortest path in an unweighted graph because, in BFS, we reach a vertex with a minimum number of edges from a source vertex.	In DFS, we might traverse through more edges to reach a destination vertex from a source.
5.	Conceptual Difference	BFS builds the tree level by level.	DFS builds the tree sub-tree by sub-tree.
6.	Approach used	It works on the concept of FIFO (First In First Out).	It works on the concept of LIFO (Last In First Out).
7.	Suitable for	BFS is more suitable for searching vertices closer to the given source.	DFS is more suitable when there are solutions away from source.

b. **Give sequence of the nodes visited as per BFS and DFS strategy for following example. Source- Arad, Destination- Bucharest (Traversal would stop after destination is reached)**



BFS

Arad->Zerind -> Sibiu -> Timisoara -> Oradea ->Fagaras -> Rimnicu Vilcea -> Lugoj -> Bucharest

DFS

Arad->Zerind->Oradea->Sibiu->Fagaras->Bucharest

Conclusion: -In this experiment we learnt about two types of traversals in a graph and implemented them.