1) Write an algorithm for depth first traversal of a graph

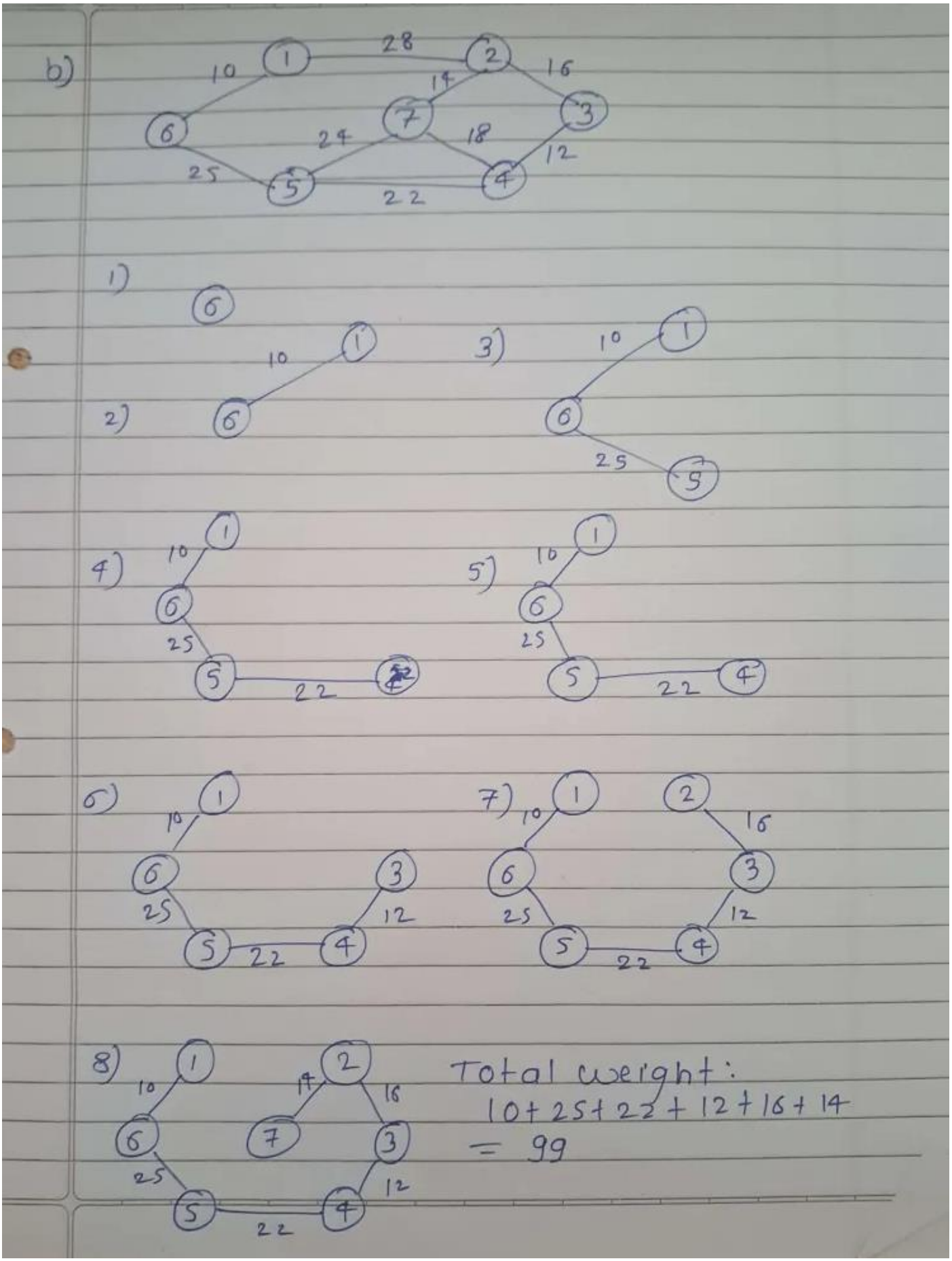
DFS - Algorithm

- **Step 1** - Define a Stack of size number of vertices in the graph.
- **Step 2** - Select any vertex as **starting point** for traversal. Visit that vertex and push it on to the Stack.
- **Step 3** - Visit any one of the non-visited **adjacent** vertices of a vertex which is at the top of stack and push it on to the stack.
- **Step 4** - Repeat step 3 until there is no new vertex to be visited from the vertex which is at the top of the stack.
- **Step 5** - When there is no new vertex to visit then use **back tracking** and pop one vertex from the stack.
- **Step 6** - Repeat steps 3, 4 and 5 until stack becomes Empty.

28

- **Step 7** - When stack becomes Empty, then produce final

2) Construct the minimum spanning tree (MST) for the given graph using Prim's Algorithm staring from vertex 6.



3) What is topological sorting? Find topological sorting of given graph.

Topological sorting for Directed Acyclic Graph (DAG) is a linear ordering of vertices such that for every directed edge u v, vertex u comes before v in the ordering.

Topological Sorting for a graph is not possible if the graph is not a DAG.

Example 3.10.1 *Sort the digraph for topological sort using source removal algorithm.*

SPPU : May-14, Marks 3

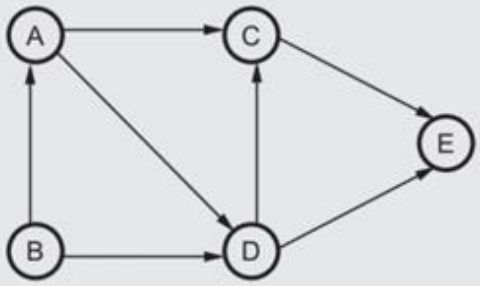


Fig. 3.10.1

Solution : We will follow following steps to obtain topologically sorted list.

Choose vertex B, because it has no incoming edge, delete it along with its adjacent edges.

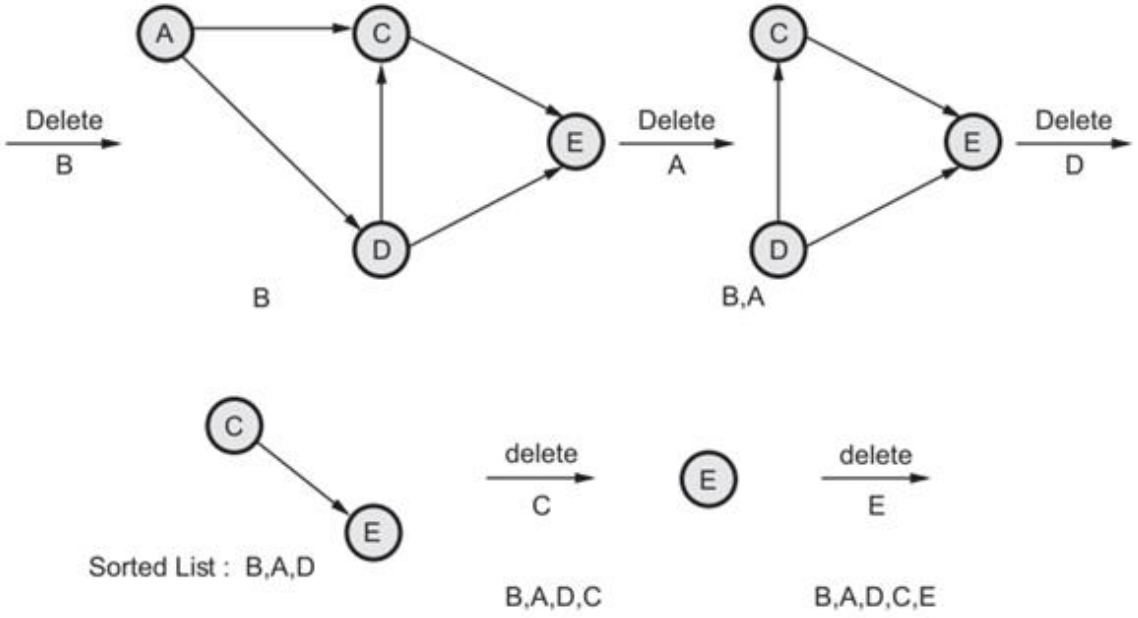


Fig. 3.10.2

Hence the list after topological sorting will be B, A, D, C, E.

4) Write an algorithm for breadth first traversal of a graph.

Step 1 - Define a **Queue** of size total number of vertices in the graph.

Step 2 - Select any vertex as **starting point** for traversal. Visit that vertex and insert it into the Queue.

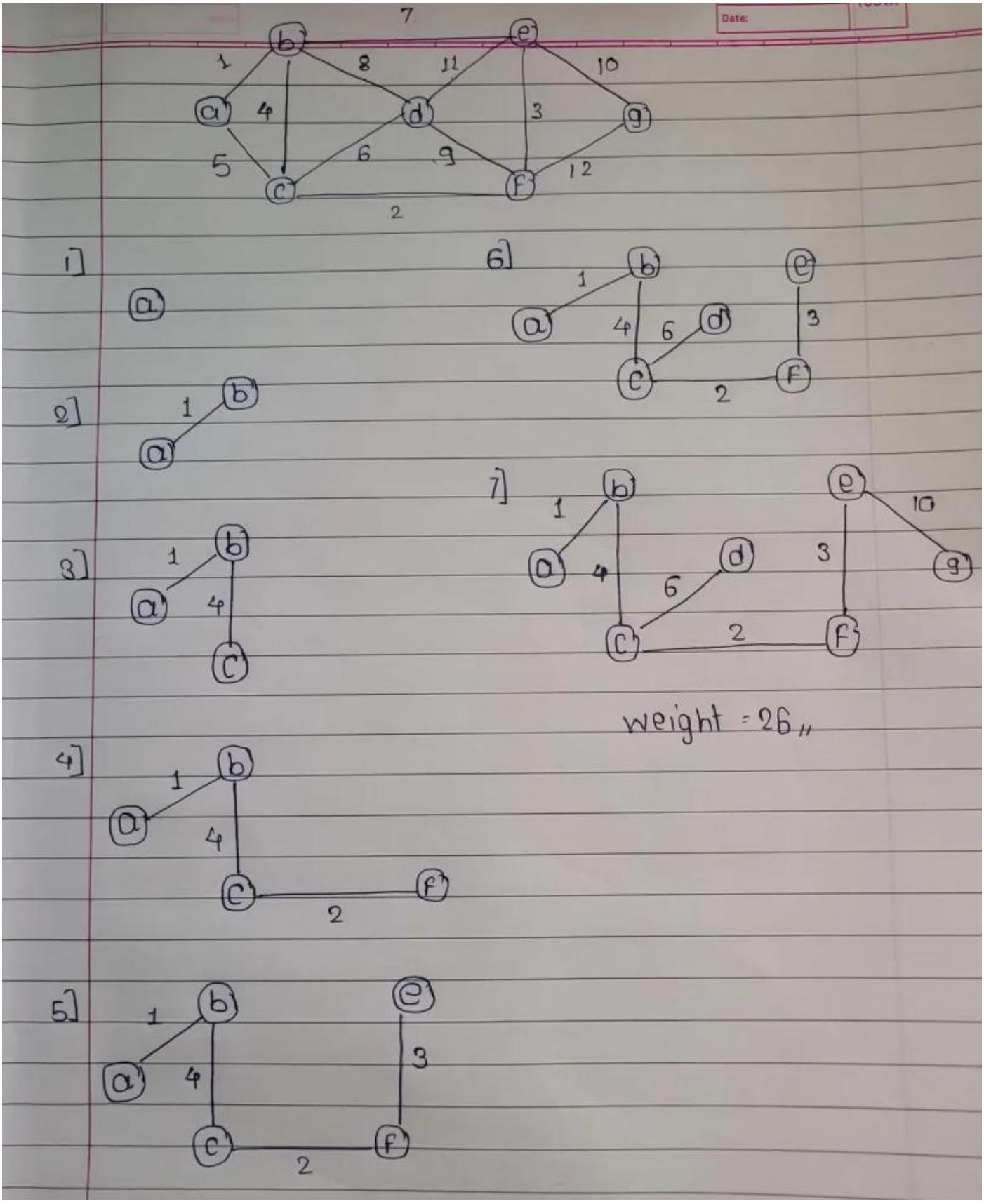
Step 3 - Visit all the **non-visited adjacent vertices** of the vertex which is at front of the Queue and insert them into the Queue.

Step 4 - When there is no new vertex to be visited from the vertex which is at front of the Queue then **delete that vertex**.

Step 5 - Repeat steps 3 and 4 until queue becomes empty.

Step 6 - When queue becomes empty, then produce final spanning tree by removing unused edges from the graph

5) Using Prim's Algorithm, find the cost of minimum spanning tree (MST) of the given graph starting from vertex 'a'



6) Define the following terms : i) Complete Graph ii) Connected Graph iii) Subgraph

3.1.4 A Complete Graph

An undirected graph, in which every vertex has an edge to all other vertices is called a complete graph.

A complete graph with N vertices has $\frac{N(N-1)}{2}$ edges.

Example of a complete graph is shown in Fig. 3.1.4.

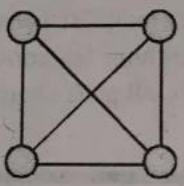


Fig. 3.1.4 : A complete graph

In a complete graph, these is an edge between every pair of vertices.

Number of edges (a pair of vertices) in a graph with n vertices is equal to combination of n elements, taking 2 at a time :

3.1.9 Connected Graph

A graph is said to be connected if there exists a path between every pair of vertices V_i and V_j.

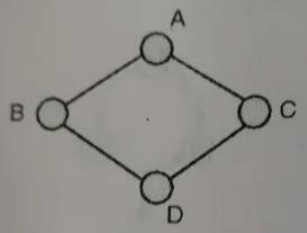
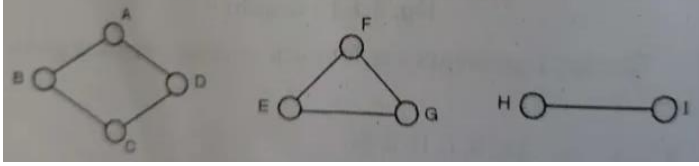
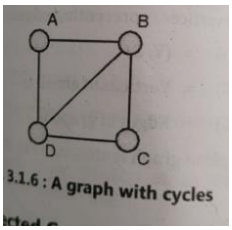


Fig. 3.1.7 : A connected graph





3.1.6 : A graph with cycles

3.1.10 Subgraph

A subgraph of G is a graph G_1 such that $V(G_1)$ is a subset of $V(G)$ and $E(G_1)$ is a subset of $E(G)$.

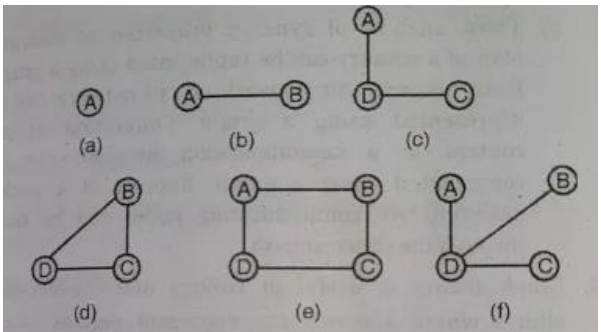


Fig. 3.1.9 : Some subgraphs of the graph of Fig. 3.1.6

7) Write Floyd Warshall Algorithm

1. Create a matrix A^0 of dimension n*n where n is the number of vertices. The row and the column are indexed as i and j respectively. i and j are the vertices of the graph.

Each cell A[i][j] is filled with the distance from the i^{th} vertex to the j^{th} vertex. If there is no path from i^{th} vertex to j^{th} vertex, the cell is left as infinity.

2. Now, create a matrix A^1 using matrix A^0. The elements in the first column and the first row are left as they are. The remaining cells are filled in the following way.

Let k be the intermediate vertex in the shortest path from source to destination. In this step, k is the first vertex. A[i][j] is filled with (A[i][k] + A[k][j]) if (A[i][j] > A[i][k] + A[k][j]).

That is, if the direct distance from the source to the destination is greater than the path through the vertex k, then the cell is filled with A[i][k] + A[k][j].

In this step, k is vertex 1. We calculate the distance from source vertex to destination vertex through this vertex k.

3. Similarly, A^2 is created using A^1. The elements in the second column and the second row are left as they are.

In this step, k is the second vertex (i.e. vertex 2). The remaining steps are the same as in

Calculate the distance from the source vertex to destination vertex through this vertex 2

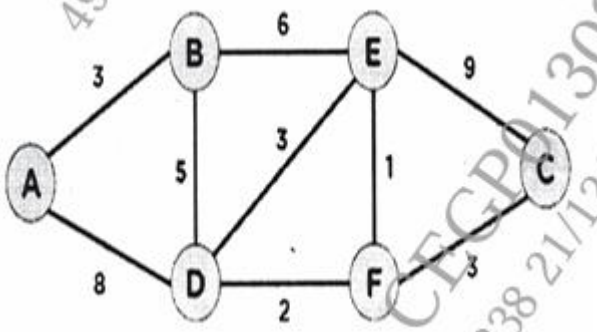
4. Similarly, A^3 and A^4 is also created.

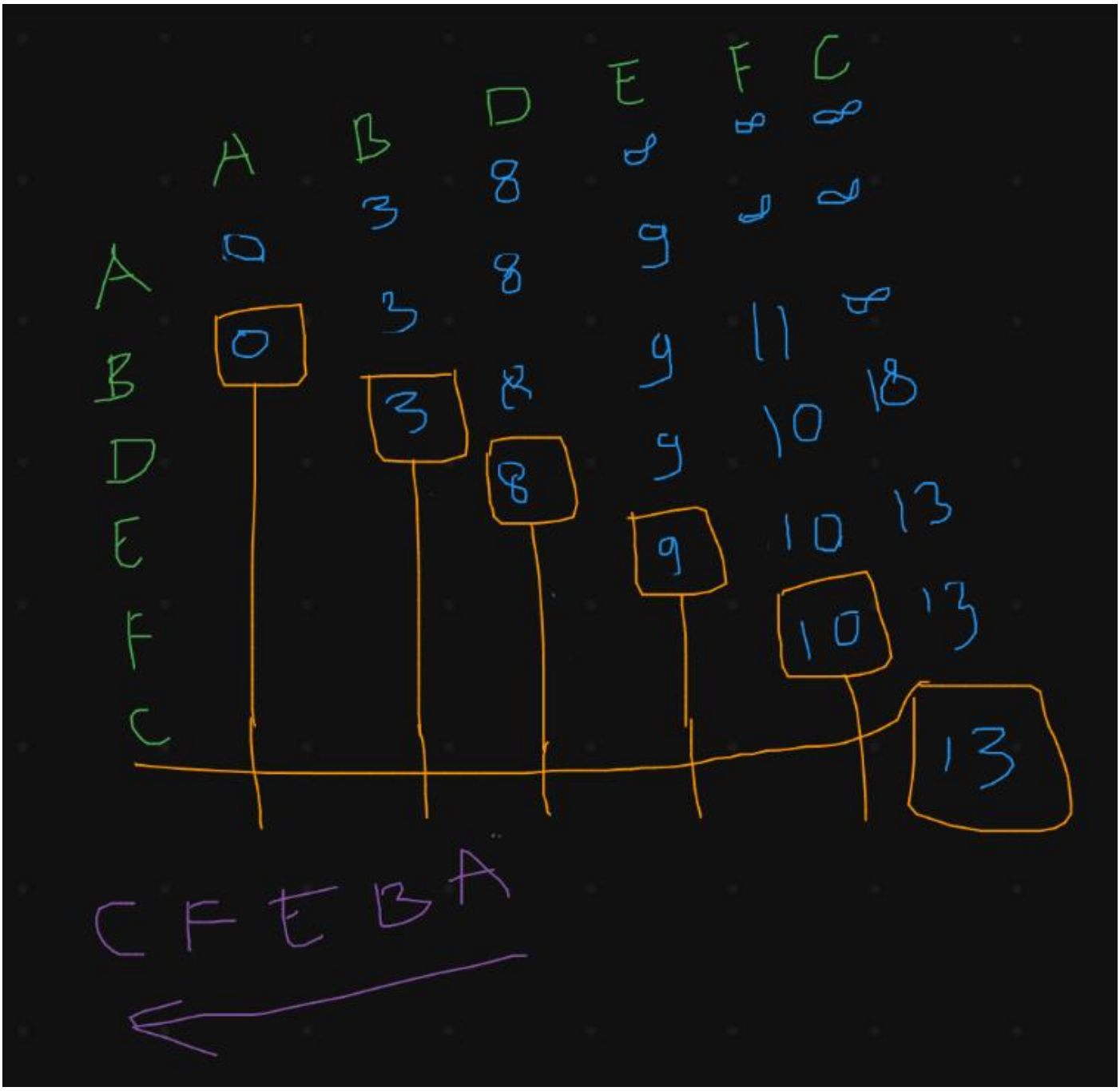
Calculate the distance from the source vertex to destination vertex through this vertex 3

Calculate the distance from the source vertex to destination vertex through this vertex 4

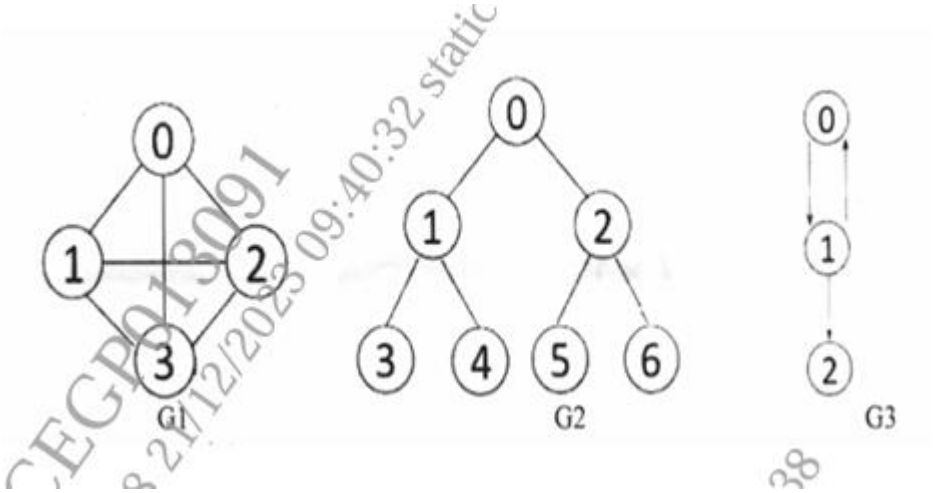
A4 gives the shortest path between each pair of vertices.

8) Apply Dijkstra's Algorithm for the graph given below, and find the shortest path from node A to node C.

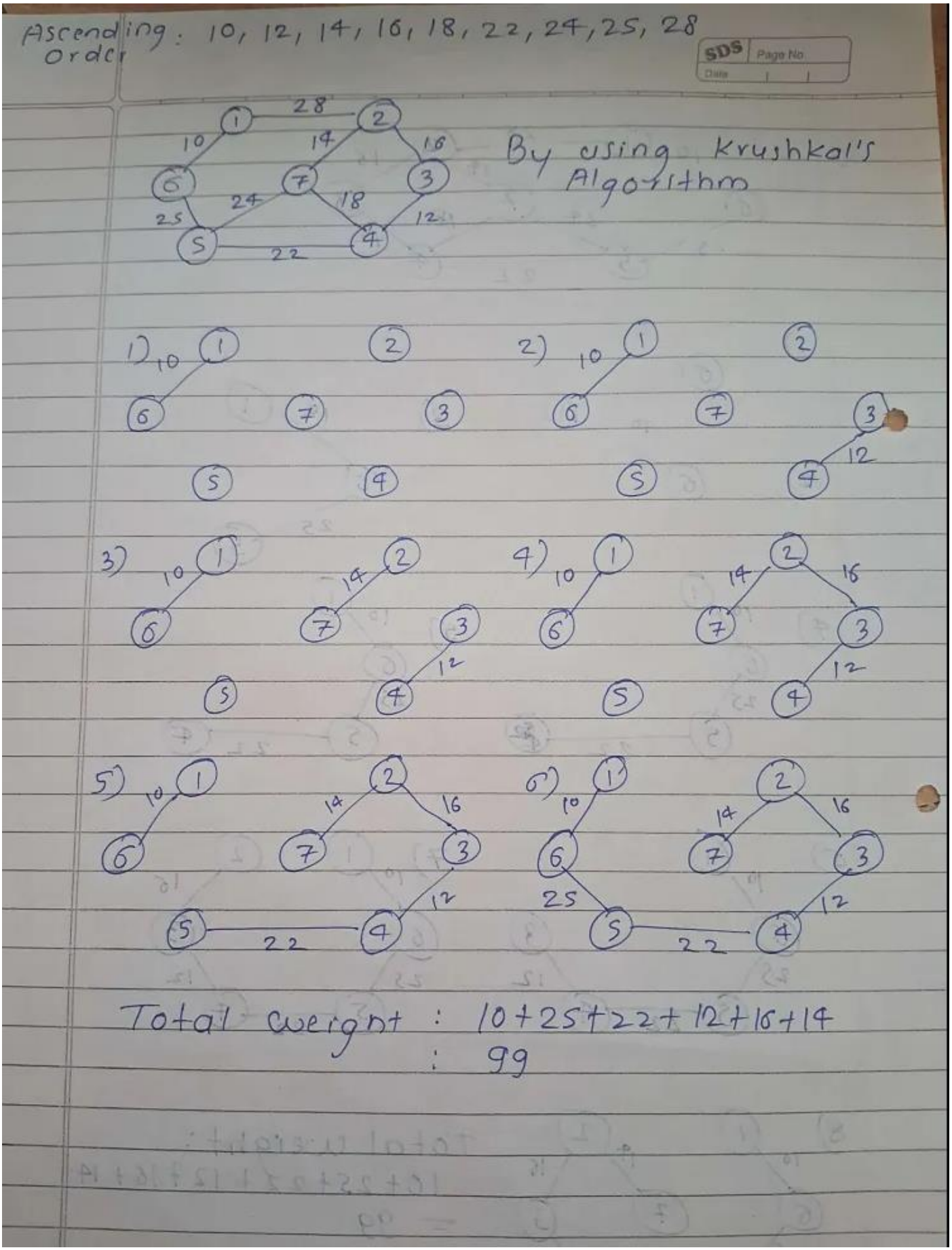




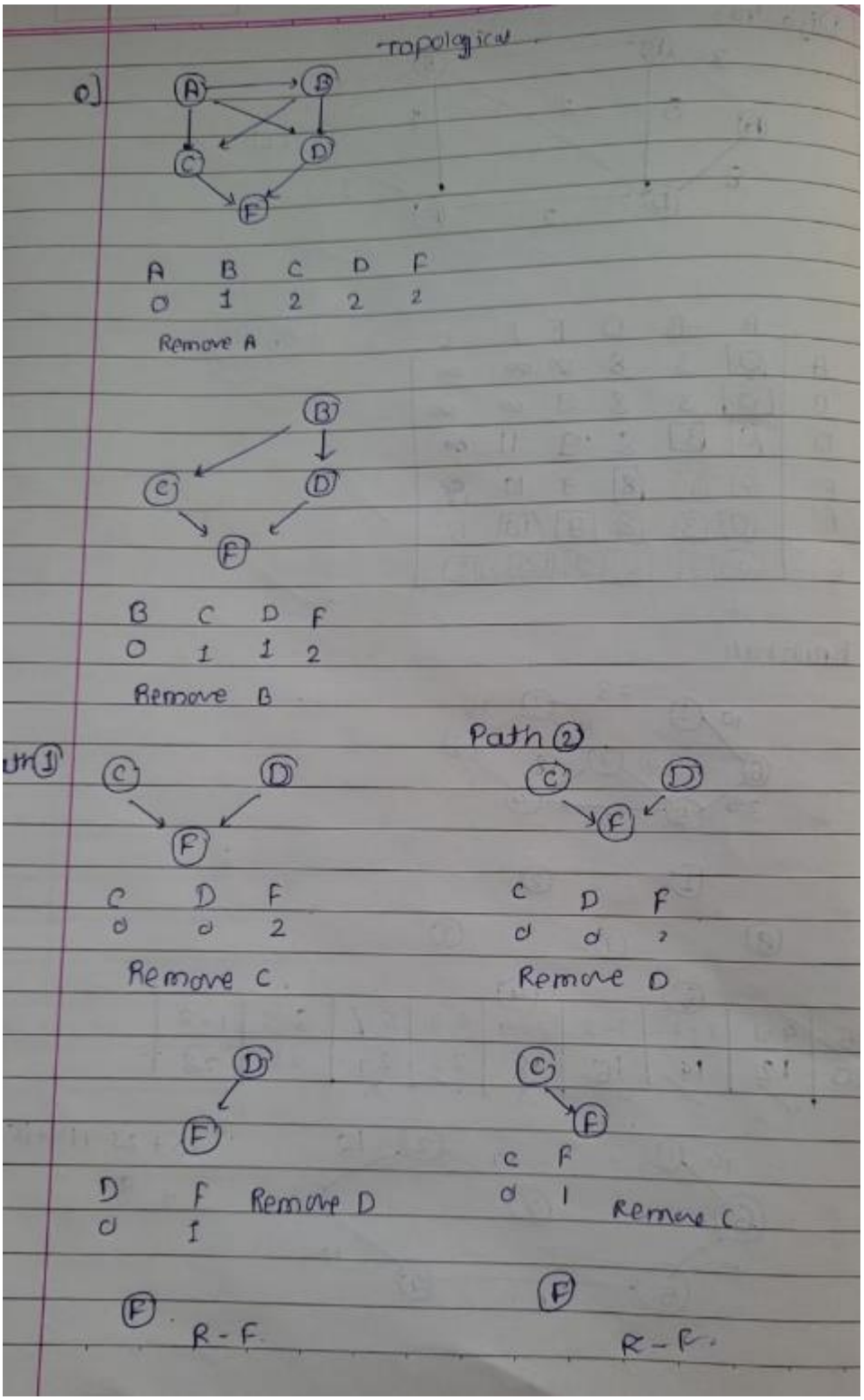
9) Define indegree & outdegree of a directed graph. Write degree for G1 & G2. Write indegree & outdegree of each vertex for G3 graph.



10) Construct the minimum spanning tree (MST) for the given graph using Kruskal's Algorithm.



11) Find the number of different topological orderings possible for the given graph



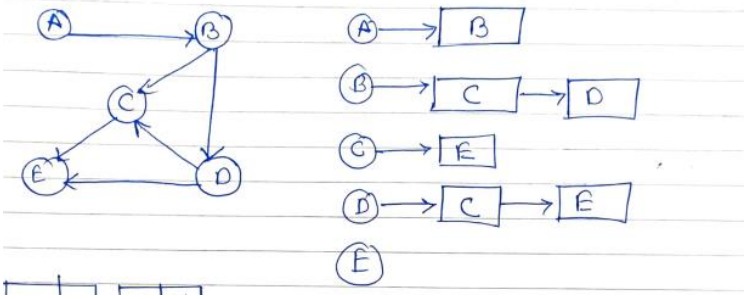
12) Elaborate following terminologies : i) Graph ii) Adjacency List iii) Adjacency Matrix

- A Graph is a non-linear data structure consisting of **nodes** and **edges**.
- The nodes are sometimes also referred to as **vertices** and the edges are lines or arcs that connect any two nodes in the graph.
- More formally a Graph can be defined as,

A graph is a pair of set **<V, E>**, where **V** is the set of vertices and **E** is the set of edges, connecting the pairs of vertices.

Adjacency List.

- In this representation, every vertex of graph contains list of its adjacency vertices.
- A linked representation is an adjacency list.
- You keep a list of neighbours for each vertex in the graph in this representation. It means that each vertex in the graph has a list of its neighbouring vertices.



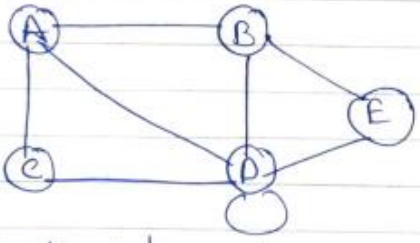
1 Adjacency Matrix :- In this representation, graph can be represented using a matrix of size total no. of vertices by total no. of vertices;

means if a graph with 4 vertices can be represented using a matrix of 4 × 4 size.

In this matrix, rows and columns both represent vertices.

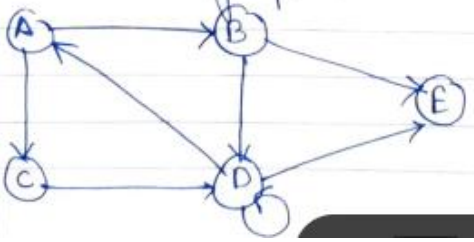
This matrix is filled with either 1 or 0. Here, 1 represent there is an edge from row vertex to column vertex and 0 represent there is no edge from row vertex to column vertex.

For undirected graph



	A	B	C	D	E
A	0	1	1	1	0
B	1	0	0	1	1
C	1	0	0	1	0
D	1	1	1	1	1
E	0	1	0	1	0

For directed graph



	A	B	C	D	E
A	0	1	1	0	0
B	0	0	0	1	1
C	0	0	0	1	0
D	1	0	0	1	1
E	0	0	0	0	0

13) Differentiate between tree and graph

Feature	Tree	Graph
Definition	A connected acyclic graph.	A collection of nodes (vertices) and edges (links).
Cycles	No cycles allowed (acyclic).	Cycles may or may not be present.
Connectivity	Always connected (one path between any two nodes).	May be connected or disconnected.
Edges	Has exactly `n-1` edges for `n` nodes.	Can have any number of edges (0 to `n(n-1)/2`).
Hierarchy	Hierarchical structure (parent-child).	No strict hierarchy; more general connections.
Direction	Usually directed (in rooted trees).	Can be directed or undirected.
Traversal	DFS, BFS used; simpler due to structure.	DFS, BFS used; can be more complex due to cycles.
Example	Family tree, file system hierarchy.	Social network, road map, web links.

14) Write pseudo code for Floyd-Warshall algorithm.

```
Input:
    A 2D array dist[][] representing the weight of the edge from vertex i to vertex j.
    - If there is no edge, set dist[i][j] = ∞
    - Set dist[i][i] = 0 for all i

Algorithm:

for k from 0 to V-1:                    // k is the intermediate vertex
    for i from 0 to V-1:                // i is the starting vertex
        for j from 0 to V-1:            // j is the ending vertex
            if dist[i][k] + dist[k][j] < dist[i][j]:
                dist[i][j] = dist[i][k] + dist[k][j]

Output:
    dist[][] will now contain the shortest distances between all pairs of vertices
```

Notes:

- `V` is the number of vertices in the graph.
- The triple loop checks and updates the shortest path from every vertex `i` to `j` using `k` as an intermediate step.

15) Write Prim's algorithm to find minimum spanning tree

Step 1: Keep a track of all the vertices that have been visited and added to the spanning tree.

Step 2: Initially remove all the parallel edges and loops.

Step 3: Choose a random vertex, and add it to the spanning tree. This becomes the root node.

Step 4: Add a new vertex, say x, such that

x is not in the already built spanning tree.

x is connected to the built spanning tree using minimum weight edge. (Thus, x can be adjacent to any of the nodes that have already been added in the spanning tree).

Adding x to the spanning tree should not form cycles.

Step 5: Repeat the Step 4, till all the vertices of the graph are added to the spanning tree.

Step 6: Print the total cost of the spanning tree.

16) Write the applications of : i) Graph ii) BFS iii) DFS

1) Applications of Graphs:

Graphs are widely used in various fields. Some key applications include:

Application Area	Description
Social Networks	Nodes represent people, edges represent relationships/friendships.
Computer Networks	Routers and devices as nodes, connections as edges.
Google Maps / GPS	Locations as nodes, roads as edges for route finding.
Web Crawling	Pages are nodes, hyperlinks are edges.
Dependency Graphs	Used in compilers, task scheduling (e.g., package installations).
Recommendation Systems	Products/users are nodes; preferences form edges.
AI / Game Development	Pathfinding algorithms on maps or grids (e.g., A*, Dijkstra).

2) Applications of BFS (Breadth-First Search):

BFS explores level by level and is useful where shortest path or minimal steps are needed.

Application	Description
Shortest Path in Unweighted Graphs	BFS finds shortest path by exploring all neighbors first.
Web Crawlers	Visit web pages level-wise from a source link.
Social Networking	Finding people within 'k' connections (friend suggestions).
Broadcasting in Networks	Data spread evenly like BFS across nodes.
Cycle Detection in Undirected Graphs	Can be used to detect loops/cycles.
AI / Puzzle Solving	State-space search like solving Rubik's cube or chess moves.

3) Applications of DFS (Depth-First Search):

DFS goes deep into one path before backtracking, useful for complex traversal or backtracking.

Application	Description
Topological Sorting	For scheduling tasks with dependencies (like course prerequisites).
Cycle Detection in Directed Graphs	DFS is ideal for detecting cycles.
Maze Solving / Path Finding	DFS explores one full path before backtracking, useful in backtracking problems.
Connected Components	Identify all connected subgraphs in a graph.
Solving Puzzles / Games	Like Sudoku, N-Queens using recursive DFS.
Artificial Intelligence (Game Tree)	Simulate different game move paths.