1. Introduction to Spanning Tree

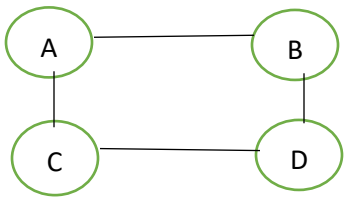
A **spanning tree** of a graph **G** is a subset of G that includes all its vertices and is a **tree** (a connected graph with no cycles). The spanning tree retains the connectivity of the original graph while minimizing the number of edges.

Key Properties of a Spanning Tree:

- A **connected graph** always has at least one spanning tree.
- A **disconnected graph** does not have a spanning tree.
- A spanning tree **does not contain cycles**.
- A spanning tree with **n** vertices has exactly **n-1 edges**.
- **Removing any edge** from a spanning tree makes it disconnected. For example, consider a spanning tree with four vertices: A, B, C, and D, where edges (A-B, B-C, C-D) form the spanning tree. If we remove edge B-C, the tree becomes disconnected, separating A-B from C-D, breaking the connectivity of the graph.
- **Adding an edge** to a spanning tree creates a cycle. For example, consider a spanning tree with four vertices: A, B, C, and D, where edges (A-B, B-C, C-D) form the spanning tree. If we add an edge between A and D, a cycle (A-B-C-D-A) is created, violating the tree structure.

2. Number of Spanning Trees

- The number of possible spanning trees in a connected graph depends on its structure. A **complete graph** with **n** vertices has **n^(n-2)** spanning trees (Cayley's formula). For example, in a complete graph with 4 vertices (A, B, C, D), the number of spanning trees is 4^(4-2) = 16. Each spanning tree represents a subset of the edges that connects all vertices without forming cycles.
- **Example of a Spanning Tree:**
  Vertices A , B , C , D

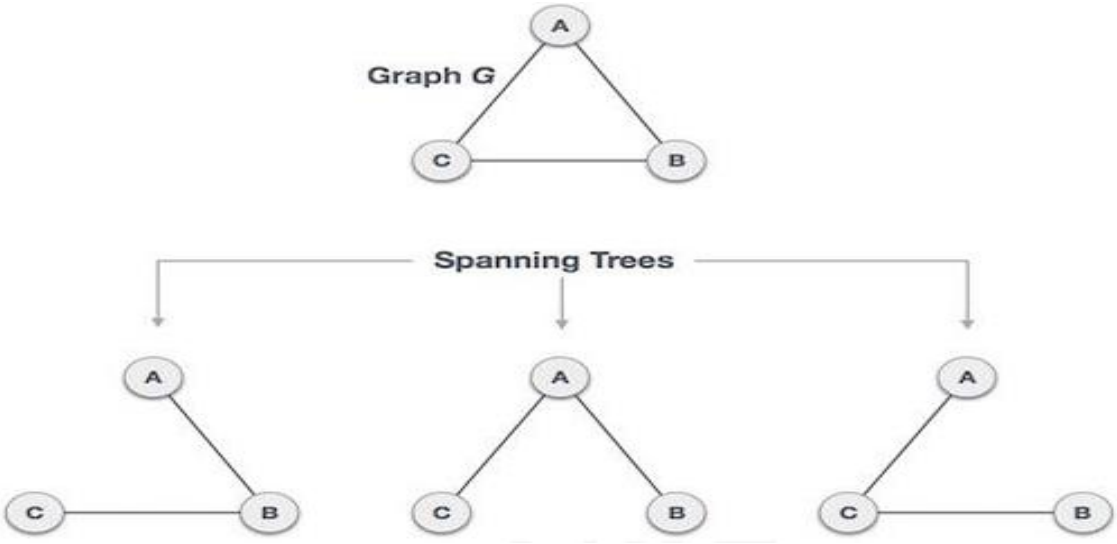


This is one possible spanning tree obtained by removing edges while maintaining connectivity and avoiding cycles.

16 Possible Spanning Trees for a Complete Graph with 4 Vertices:

1. A-B, B-C, C-D
2. A-B, B-D, C-D

3. A-B, A-C, C-D
4. A-B, A-D, C-D
5. A-C, B-C, C-D
6. A-C, A-D, C-D
7. A-D, B-D, C-D
8. A-B, B-C, B-D
9. A-B, B-C, A-D
10. A-B, B-D, A-C
11. A-C, C-D, B-D
12. A-D, C-D, B-D
13. A-C, A-B, B-D
14. A-C, A-D, B-D
15. A-D, A-B, C-D
16. A-D, A-C, B-C



4. Minimum Spanning Tree (MST)

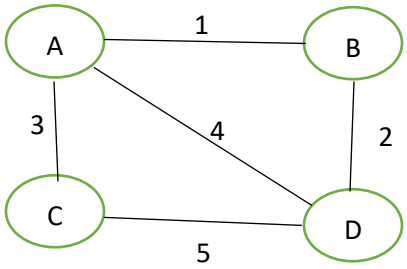
A **Minimum Spanning Tree (MST)** is a spanning tree of a weighted graph with the **minimum possible sum of edge weights**. It is used in scenarios where minimizing cost, distance, or time is important.

Key Algorithms for Finding MST:

- **Kruskal's Algorithm** (Greedy Algorithm):
    1. Sort edges by weight.
    2. Add the smallest edge to the MST unless it forms a cycle.
    3. Repeat until the tree spans all vertices.

Example(1) of Finding an MST

Consider the following weighted graph:

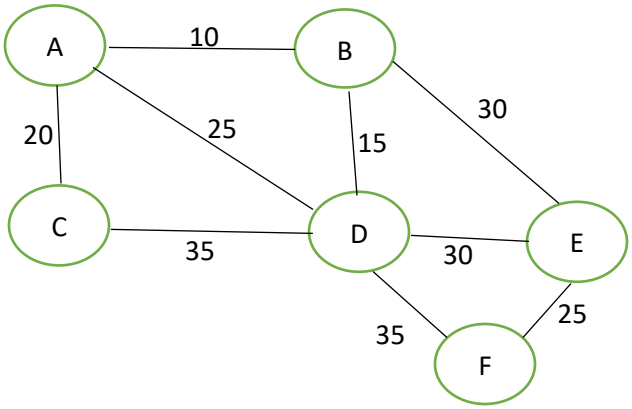


Using **Kruskal's Algorithm**:

1. Pick edge **A-B (1)**.
2. Pick edge **B-D (2)**.
3. Pick edge **A-C (3)**.
4. Ignore **A-D (4)** and **C-D (5)** as they form a cycle.

Final MST includes edges: **(A-B, B-D, A-C)** with a total weight of **6**.

Example (2):



- A ↔ B = **10**
- A ↔ C = **20**
- A ↔ D = **25**

###############

- B ↔ A = **10**
- B ↔ D = **15**
- B ↔ E = **30**

###############

- C ↔ A = **20**
- C ↔ D = **35**

###############

- D ↔ A = **25**
- D ↔ C = **35**
- D ↔ B = **15**
- D ↔ E = **30**
- D ↔ F = **35**

###############

- E ↔ B = **30**
- E ↔ D = **30**
- E ↔ F = **25**

##############

- F ↔ D= **35**
- F ↔ E = **25**

Application of Spanning Tree

Spanning tree is basically used to find a minimum path to connect all nodes in a graph. Common application of spanning trees are :

• Civil Network Planning

• Computer Network Routing Protocol

• Cluster Analysis