14.4 Trees

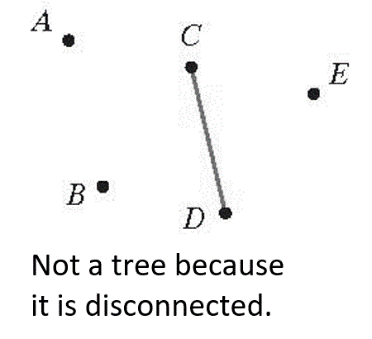
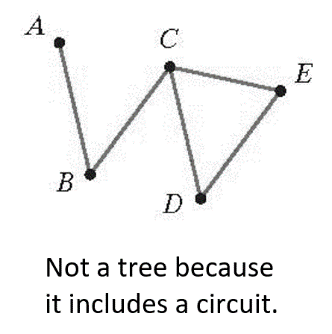
# Objective 1: Understand the definition and properties of a tree

A graph with the smallest number of edges that allows all vertices to be reached from all other vertices is called a **tree**.

**PROPERTIES OF A TREE**

A **tree** is a graph that is connected and has no circuits. All trees have the following properties:

1. There is one and only one path joining any two vertices.
2. Every edge is a bridge.
3. A tree with *n* vertices must have n – 1 edges.

** a graph with vertices A to E. The edges are A B, B C, C D, and D E.  This graph is a tree.
 **

# Objective 2: Find a spanning tree for a connected graph

A **subgraph** is a set of vertices and edges chosen from among those of a given graph.

An original graph and two possible subgraphs with vertices A to G. The original graph has edges A E, B C, B E, C F, D F, E G, and F G. The first possible subgraph has edges A E, B E, C F, D F, E G, and F G. The second possible subgraph has edges A E, B C, B E, C F, D F, and F G. Each subgraph has edges connecting all vertices, but no circuits.


A subgraph that contains all of a connected graph’s vertices, is connected, and contains no circuits is called a **spanning tree**. By removing redundant connections, spanning trees increase the efficiency of the network modeled by the original graph. It is always possible to start with a connected graph, retain all of its vertices, and remove edges until a spanning tree remains. As a tree, the spanning tree must have one less edge than it has vertices.

# Objective 3: Find the minimum spanning tree for a weighted graph

The **minimum spanning tree** for a weighted graph is a spanning tree with the smallest possible total weight. **Kruskal’s Algorithm** is a procedure for finding the minimum spanning tree for a weighted graph.

**KRUSKAL’S ALGORITHM**

To find the minimum spanning tree for a weighted graph, follow these steps:

1. Find the edge with the smallest weight in the graph. If there is more than one, pick one at random. Mark this edge in some way.
2. Find the next-smallest edge in the graph. If there is more than one, pick one at random. Mark this second edge of the tree.
3. Find the next-smallest unmarked edge in the graph that does not create a circuit. If there is more than one, pick one at random. Mark this edge of the tree.
4. Repeat step 3 until all vertices have been included. The marked edges form the desired spanning tree.

The diagram below represents city sidewalks and can be used to determine the minimum distance required to stop at each building.

A map of the campus with buildings A to G and the connecting sidewalks with lengths measured in feet. The campus map is represented by a weighted graph with vertices (buildings) A to G. The edges (sidewalks) and their weights (lengths) are as follows. A B, 251. A D, 249. A G, 274. B C, 225. B D, 245. C D, 253. C E, 259. C F, 256. C G, 264. D G, 251. E F, 262. and F G, 242.
