TREES:

- We show a special bipartite graph, which is connected and contains no cycles. It is one of a very important family of graphs that we shall study in this section.
- The word TREES suggests branching out from a root and never completing a cycle.
- Trees as graphs have many applications, especially in data storage, searching, and communication.

Definition: A graph is called a tree if it is connected and contains no cycles.

Question 3.2.1. *Which of the graphs in Fig. 3.2.1 is a tree? Why?*

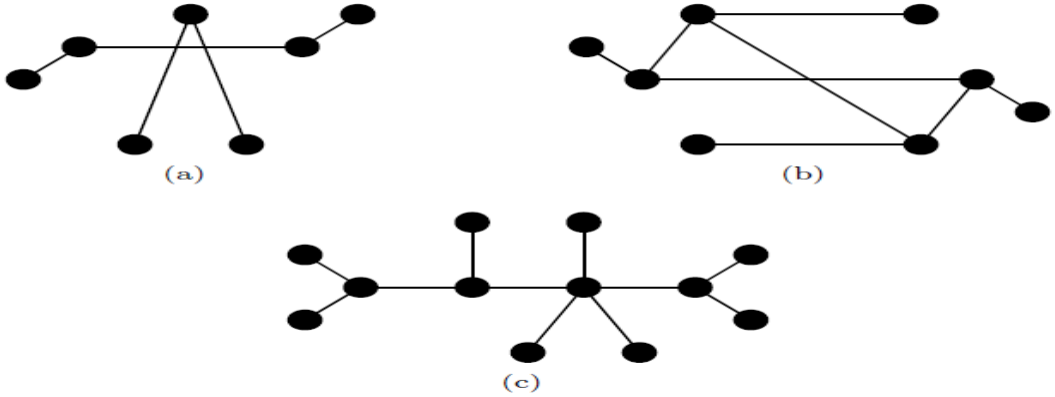
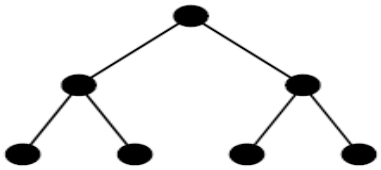
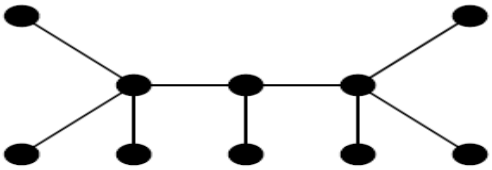


Fig. 3.2.1

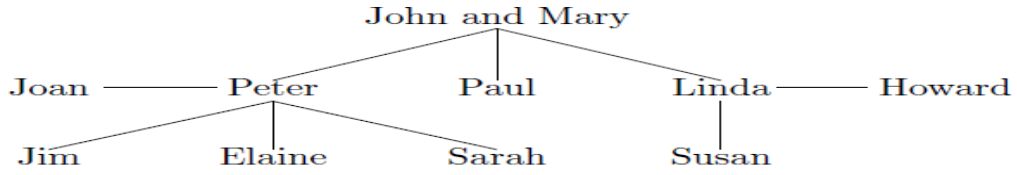
Examples of Trees:



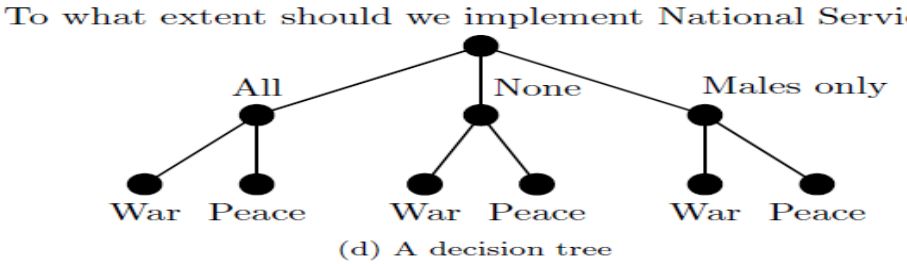
(a) A binary tree



(b) A caterpillar



(c) A family tree



(d) A decision tree

Remarks:

- A tree is connected, and thus it is not surprising that every two of its
- vertices are joined by a path.
- An important feature of trees that other connected graphs do not have is the uniqueness of such a path joining any two vertices.

Theorem: Let G be a graph. Then G is a tree if and only if every two vertices in G are joined by a unique path.

ACYCLIC, TREE, FOREST, & LEAF:

Definition 3.1 A graph G is

- ***acyclic*** if there are no cycles or circuits in the graph.
- a ***tree*** if it is both acyclic and connected.
- a ***forest*** if it is an acyclic graph.

In addition, a vertex of degree 1 is called a ***leaf.***

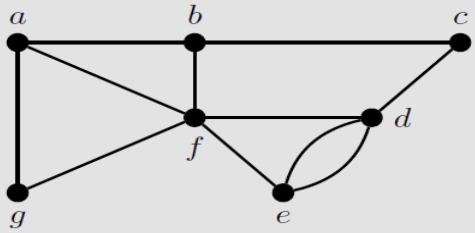
SOME RELATIONS:

- ✓ A forest is simply a graph in which every component is itself a tree.]
- ✓ A tree is a connected forest, and every component of a forest is a tree.
- ✓ A graph with no cycles has no odd cycles; hence trees and forests are bipartite.
- ✓ Paths are trees. A tree is a path if and only if its maximum degree is 2.

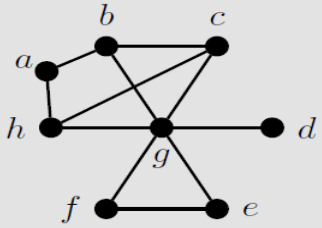
Spanning Trees: A spanning tree is a spanning subgraph that is also a tree.

- ➢ A graph that is a tree has exactly one spanning tree; the full graph itself.

Example 3.1 For each of the graphs below, find a spanning tree and a subgraph that does not span.

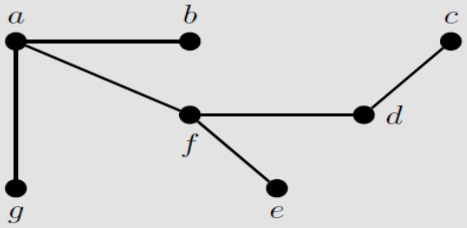


G_1

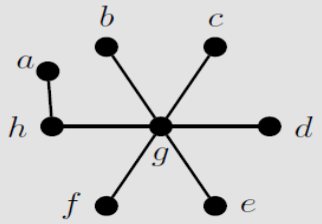


G_2

Solution: To find a spanning tree, we must form a subgraph that is connected, acyclic, and includes every vertex from the original graph. The graphs T_1 and T_2 below are two examples of spanning trees for their respective graphs; other examples exist.

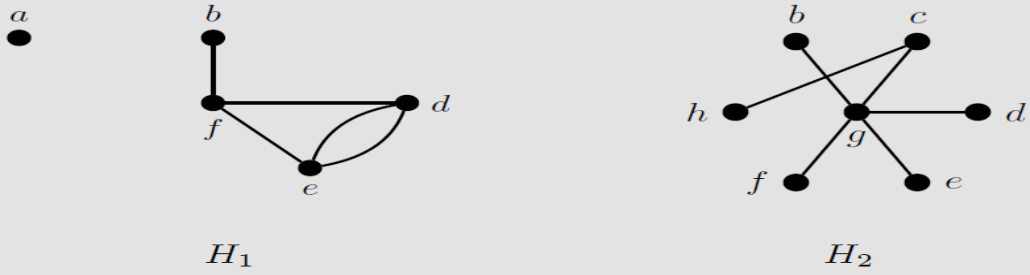


T_1



T_2

The subgraph H_1 below is neither spanning nor a tree since some vertices from G_1 are missing and there is a multi-edge (and hence a circuit) between d and e. The subgraph H_2 below is not spanning since it does not contain vertex a, but it is a tree since no circuits or cycles exist. As above, these are merely examples and other non-spanning subgraphs exist.



H_1 H_2

Question: Find all spanning trees of graph G in Fig. 3.3.1(a). How many are there?

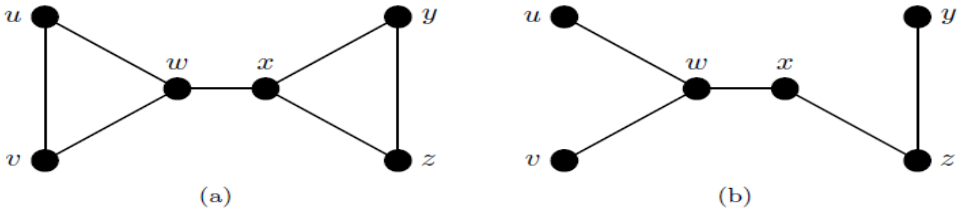


Fig. 3.3.1

Question: Under what conditions will a graph have a spanning tree? Clearly, the more difficult criteria is the tree, not spanning, since every graph contains a spanning subgraph.

Answer: The existence of a spanning tree of graph G is directly linked to the connectedness of G.

Theorem: Let G be a graph. Then G is connected if and only if G contains a spanning tree.

Minimum Spanning Trees:

Definition 3.3 Given a weighted graph $G = (V, E, w)$, T is a ***minimum spanning tree***, or MST, of G if it is a spanning tree with the least total weight.

Algorithms To Find MST:

1. Kruskal's Algorithm
2. Prim's Algorithm

Kruskal's Algorithm:

Joseph Kruskal was an American mathematician best known for his work in statistics and computer science. This algorithm is unique in that it is both efficient and optimal while still easily implemented and understandable for a non-scientist.

Kruskal's Algorithm

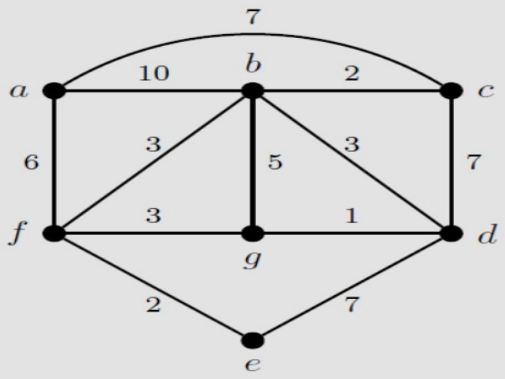
Input: Weighted connected graph $G = (V, E)$.

Steps:

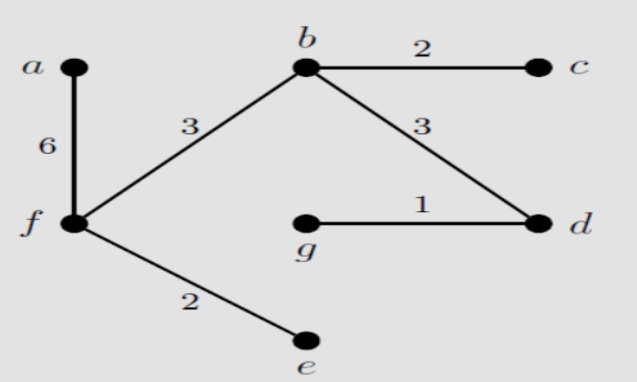
1. Choose the edge of least weight. Highlight it and add it to $T = (V, E')$.
2. Repeat Step (1) so long as no circuit is created. That is, keep picking the edges of least weight but skip over any that would create a cycle in T.

Output: Minimum spanning tree T of G.

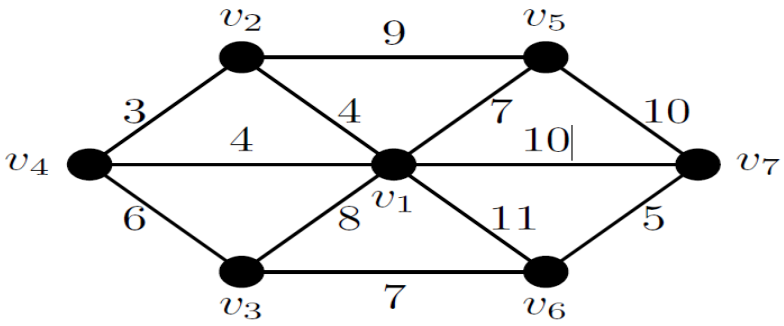
Example 3.2 Find the minimum spanning tree of the graph G below using Kruskal's Algorithm.

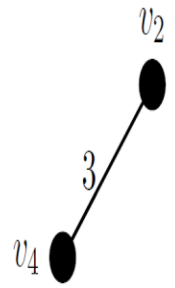


Output: The following tree with total weight 17.

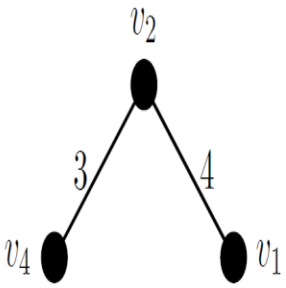


Example: Applying Kruskal's Algorithm on the weighted graph given below, we have the following sequence of subgraphs of G.

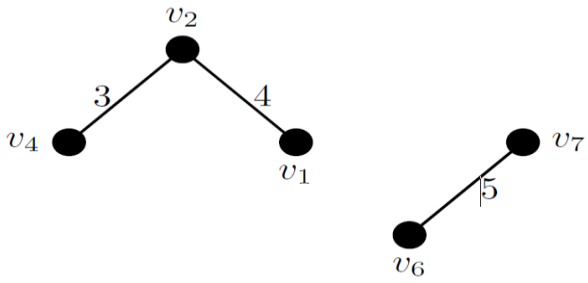




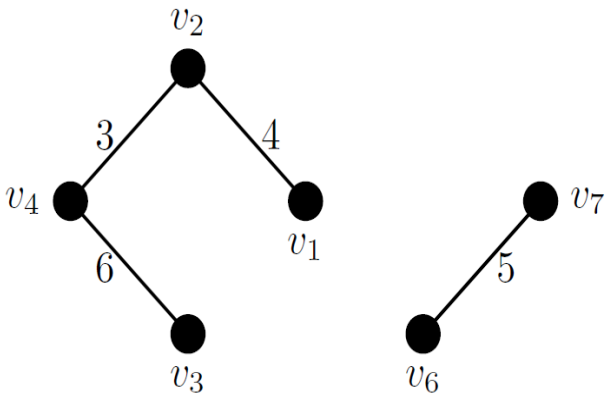
(a) $E_1 = \{v_2v_4\}$



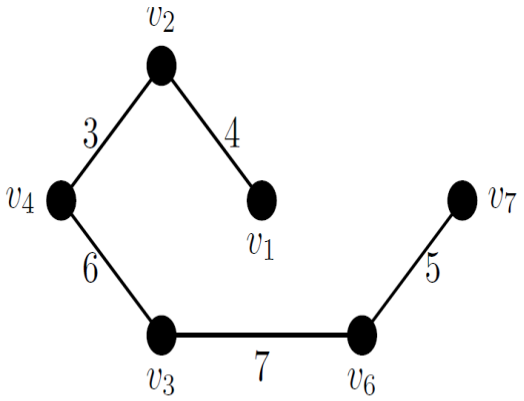
(b) $E_2 = \{v_2v_4, v_1v_2\}$



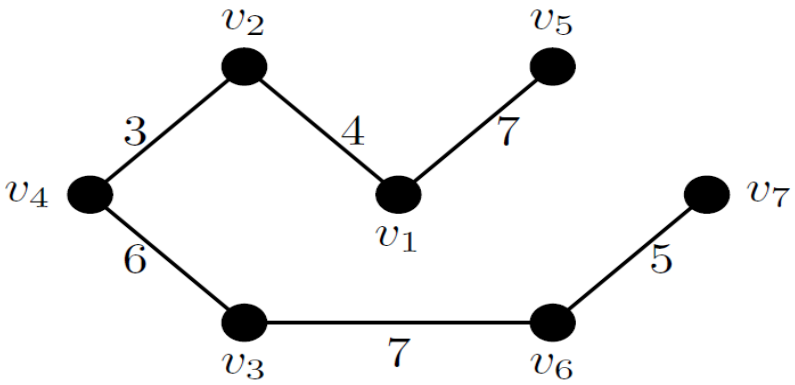
(c) $E_3 = \{v_2v_4, v_1v_2, v_6v_7\}$



(d) $E_4 = \{v_2v_4, v_1v_2, v_6v_7, v_3v_4\}$



(e) $E_5 = \{v_2v_4, v_1v_2, v_6v_7, v_3v_4, v_3v_6\}$



(f) $E_6 = \{v_2v_4, v_1v_2, v_6v_7, v_3v_4, v_3v_6, v_1v_5\}$

output: The tree T induced by E_6 is a minimum spanning tree of G. The weight of T is 3 + 4 + 5 + 6 + 7 + 7 = 32.

Prim's Algorithm:

The algorithm described below is widely known as Prim's Algorithm, named for the American mathematician and computer scientist Robert C. Prim.

- Prim's Algorithm contrasts from Kruskal's in that the structure obtained in each step is itself a tree.
- By the end of the process, a spanning tree will be found.

Prim's Algorithm

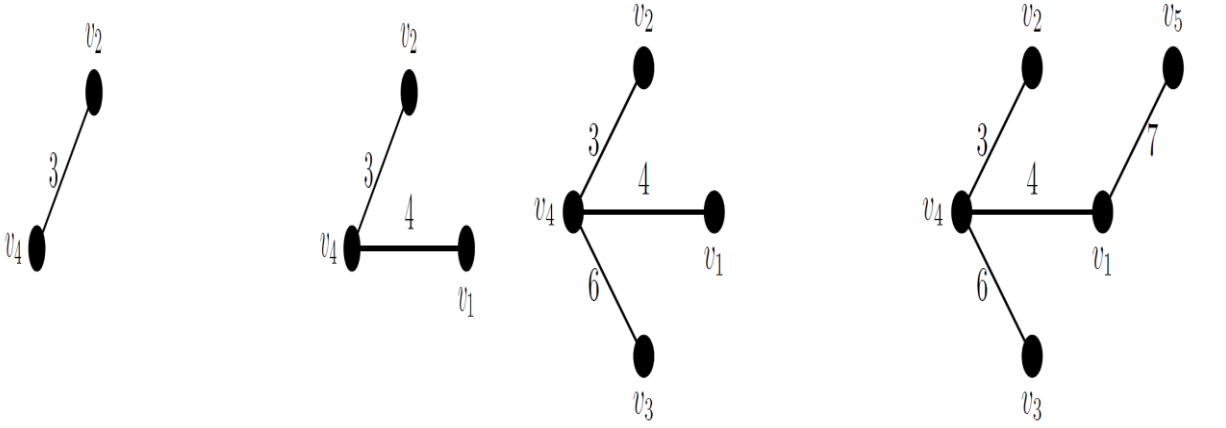
Input: Weighted connected graph $G = (V, E)$.

Steps:

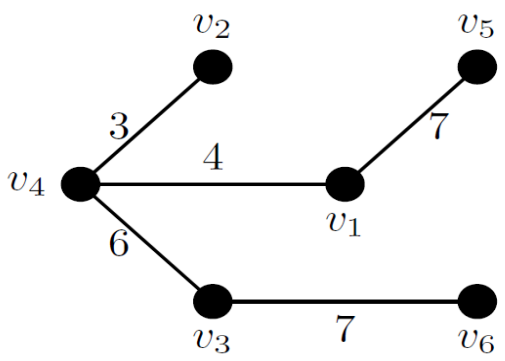
1. Let v be the root. If no root is specified, choose a vertex at random. Highlight it and add it to $T = (V', E')$.
2. Among all edges incident to v, choose the one of minimum weight. Highlight it. Add the edge and its other endpoint to T.
3. Let S be the set of all edges with exactly endpoint from $V(T)$. Choose the edge of minimum weight from S. Add it and its other endpoint to T.
4. Repeat Step (3) until T contains all vertices of G, that is $V(T) = V(G)$.

Output: Minimum spanning tree T of G.

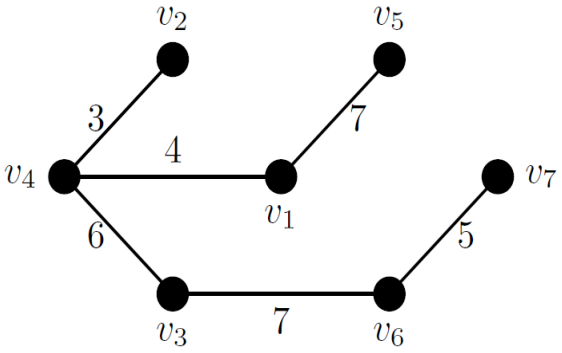
Example: Applying Prim's Algorithm on the weighted graph in Previous Example, we have the following sequence of subgraphs of G.



(a) $E_1 = \{v_2v_4\}$ (b) $E_2 = \{v_2v_4, v_1v_4\}$ (c) $E_3 = \{v_2v_4, v_1v_4, v_3v_4\}$ (d) $E_4 = \{v_2v_4, v_1v_4, v_3v_4, v_1v_5\}$



(e) $E_5 = \{v_2v_4, v_1v_4, v_3v_4, v_1v_5, v_3v_6\}$



(f) $E_6 = \{v_2v_4, v_1v_4, v_3v_4, v_1v_5, v_3v_6, v_6v_7\}$

The tree T induced by E_6 is a minimum spanning tree of G. The weight of T is 3 + 4 + 5 + 6 + 7 + 7 = 32.

Remarks:

- Note that using Kruskal's Algorithm, the subgraph induced by E_i for each i is always connected whereas this is generally not true if we use Prim's Algorithm.
- Given any weighted connected graph G, the weight of its minimum spanning tree is unique but not necessarily the spanning tree that has this unique weight. For example, the above minimum spanning trees obtained by Kruskal's or Prim's algorithm are different but the weights are equal.