Chapter 4: Trees

9 Properties of Trees

1. Definitions:
   - forest - a graph that contains no cycles
   - tree - a connected forest

2. Theorem: Let $T$ be a graph with $n$ vertices. Then the following statements are equivalent:
   
   (i) $T$ is a tree;
   
   (ii) $T$ contains no cycles, and has $n - 1$ edges;
   
   (iii) $T$ is connected, and has $n - 1$ edges;
   
   (iv) $T$ is connected, and each edge is a bridge;
   
   (v) any two vertices of $T$ are connected by exactly one path;
   
   (vi) $T$ contains no cycles, but the addition of any new edge creates exactly one cycle.

   pf. We assume that $n \geq 2$. (0therwise trivial) (i) $\Rightarrow$ (ii)
   
   • Induction on $n$. Clear for $n = 2$.
   
   • Assume $n$ vertices and remove an edge. Disconnects $T$ into 2 trees ($T_1$, $T_2$) with $n_1$, $n_2$ vertices. ($n = n_1 + n_2$)
   
   • By induction, $T_1$ has $n_1 - 1$ vertices, and $T_2$ has $n_2 - 1$ vertices.
   
   • It follows that $T$ has $(n_1 - 1) + (n_2 - 1) + 1 = n - 1$ vertices.

   (ii) $\Rightarrow$ (iii)
   If $T$ is disconnected, then each component is disconnected with one less edge than vertex. This implies that the number of vertices in $T$ is $k \geq 2$ (# of components) more than the number of edges, which is a contradiction.

   (iii) $\Rightarrow$ (iv)
   Removing any edge of $T$ results in a graph with $n$ vertices and $n - 2$ edges, which must be disconnected by Theorem 5.2.

   (iv) $\Rightarrow$ (v)
   If any pair of vertices were connected by 2 paths, then the union of the paths would form a cycle, contradicting the fact that each edge is a bridge.

   (v) $\Rightarrow$ (vi)
   If $T$ contained a cycle, then no edges of that cycle would be a bridge, which contradicts (v). If an edge $e$ is added to $T$, then since both vertices incident to $e$ are already connected in $T$, a cycle is created. The cycle is unique by Exercise 5.11.

   (vi) $\Rightarrow$ (i)
   Suppose that $T$ is disconnected. Then, adding an edge connecting two components does not create a cycle, which contradicts (vi).

3. Corollary: If $G$ is a forest with $n$ vertices and $k$ components, then $G$ has $n - k$ edges.

   pf. Apply the previous theorem to each component of $G$. 
4. Definitions:

- **Spanning tree** - a tree that connects all vertices of a connected graph \( G \).
- **Spanning forest** - a forest that contains all vertices of \( G \).

5. Procedure for constructing a spanning tree of a connected graph \( G \). (Extends to forest)

(a) Pick a cycle and remove any edge (resulting graph remains connected).

(b) Repeat this with remaining cycles.

6. Example (p. 47, pr. 9.5)

Draw all spanning trees of the following graph.

![Graph Diagram]

7. Definitions: Assume \( G \) has \( n \) vertices, \( m \) edges, and \( k \) components

- **Cycle rank** - the total number of edges \( (\gamma(G) = m - n + k) \) removed from \( G \) to create a spanning forest.
- **Cutset rank** - the number of edges \( (\xi(G) = n - k) \) in a spanning forest of \( G \).
- **Complement** (of \( T \) in \( G \)) - graph obtained by removing edges of \( T \) from \( G \).

8. **Theorem**: If \( T \) is any spanning forest of a graph \( G \), then

(i) each cutset of \( G \) has an edge in common with \( T \);

(ii) each cycle of \( G \) has an edge in common with the complement of \( T \).

pf.

(i) Let \( C^* \) be a cutset of \( G \), which splits a component of \( G \) into subgraphs \( H \) and \( K \) if removed. Since \( T \) is a spanning forest, it must contain an edge joining a vertex of \( H \) to a vertex of \( K \). (This is the common edge.)

(ii) Let \( C \) be a cycle of \( G \) with no edge in common with the complement of \( T \). It follows that \( C \) must be contained in \( T \), which is a contradiction.

9. Example (p. 47, pr. 9.11)

Let \( T_1 \) and \( T_2 \) be spanning trees of a connected graph \( G \).

(i) If \( e \) is any edge of \( T_1 \), show that there exists an edge \( f \) of \( T_2 \) such that the graph \( (T_1 - \{e\}) \cup \{f\} \) (obtained from \( T_1 \) on replacing \( e \) by \( f \)) is also a spanning tree.

(ii) Deduce that \( T_1 \) can be ‘transformed’ into \( T_2 \) by replacing the edges of \( T_1 \) one at a time by edges of \( T_2 \) in such a way that a spanning tree is obtained at each stage.
10 Counting Trees

1. Enumerating alkanes $\text{C}_n\text{H}_{2n+2}$ with a given number of carbon atoms is equivalent to counting the number of trees where each vertex is of degree 1 or 4. (Cayley, 1850’s)

2. Graph Enumeration Problems:
   - Solved: number of graphs, connected graphs, trees, and Eulerian graphs with a given number of vertices and edges.
   - Unsolved: general results for planar and Hamiltonian graphs (Appendix gives some info)

3. Theorem (Cayley, 1889): There are $n^{n-2}$ distinct labeled trees on $n$ vertices.
   pf. (Proof due to Prüfer and Clark; Second proof in book.)
   - One-to-one correspondence between labeled trees of order $n$ and sequences $(a_1, a_2, \ldots, a_{n-2})$, where $a_i \in \mathbb{Z}$, $1 \leq a_i \leq n$.
   - There are $n^{n-2}$ such sequences.
   - Assume that $n \geq 3$, since trivial if $n = 1, 2$.
   - Let $T$ be a labeled tree of order $n$.
   - Let $a_1$ be the vertex adjacent to the smallest labeled end-vertex. (Remove end-vertex and edge.)
   - Repeat until only 2 vertices left: sequence is $(a_1, a_2, \ldots, a_{n-2})$.
   - Example:

4. Example (p. 51, pr. 10.1)
   Verify directly that there are exactly 125 labeled trees on 5 vertices.

   Total number: \( \frac{(5!)}{2} + (5 \cdot 4 \cdot 3) + 5 = 125 \). (Middle: vertices 4, 3, 5)
11 More Applications

1. Minimum Connector Problem: We want to build a railway network connecting \( n \) cities (A passenger must be able to travel from any city to any other.), and we want to minimize the amount of track used. (Find minimum weight spanning tree from \( n^{n-2} \) possibilities.

- **Greedy algorithm:** Sequentially choose edges of minimum weight such that no cycle is created.

- **Theorem:** Let \( G \) be a connected graph with \( n \) vertices. The following construction gives a solution to the minimum connector problem:
  
  i. let \( e_1 \) be an edge of \( G \) of smallest weight;
  
     ii. define \( e_2, e_3, \ldots, e_{n-1} \) by choosing at each stage a new edge of smallest possible weight that doesn’t form a cycle with the previous edges. The resulting spanning tree, \( T \), is the solution.

  **pf.**
  
  - \( T \) is a spanning tree by Theorem 9.1.
  
  - Prove that \( T \) is minimum by contradiction. Assume \( S \) is spanning tree with \( w(S) < w(T) \).
  
  - Let \( e_k \) be the first edge chosen in \( T \) that is not in \( S \).
  
  - Adding \( e_k \) to \( S \) creates a unique cycle \( C \).
  
  - Let \( e \) be an edge of \( C \) that’s in \( S \) but not \( T \). Create \( S' \) by replacing \( e \) with \( e_k \) ⇒ \( w(S') \leq w(S) \). (Otherwise, we would have chosen \( e \) for \( T \) instead of \( e_k \).
  
  - Note: \( S' \) has one more edge in common with \( T \) than \( S \).
  
  - Repeat this process to change \( S \) into \( T \). Since the total weight decreases at each step, we conclude that \( w(T) \leq w(S) \), which is a contradiction.

- **Example:** (p. 57, pr. 11.1)
  Use the greedy algorithm to find a minimum-weight spanning tree in the graph below. (Add weights.)

- Lower bound for traveling salesman problem:
  
  - Any solution consists of two edges adjacent to a vertex \( v \) and a spanning tree for remaining vertices.
  
  - Pick a vertex \( v \). Find a min weight spanning tree for remainder.
  
  - Pick two smallest weights of edges incident to \( v \). Result provides lower bound.
  
  - Go through on previous example.

2. Book goes through Chemical Molecule Enumeration and Electrical Networks
3. Searching Trees: Tree often used for hierarchical structure root (i.e. computer file)
   We need a systematic method to look find any vertex without visiting any vertex too often.

   (a) **Breadth first search:** Visit all vertices adjacent to a current vertex before proceeding to
       the next vertex.

   (b) **Depth first search:** Go as deeply as possible into a tree before backtracking to go to
       other vertices.

4. Example: (p. 58, pr. 11.8)
   Perform a breadth first search and a depth first search on the tree below.

![Tree Diagram]