Trees: Applications

Binary trees:
Each vertex has degree either 1 or 2.

Binary tree is a very fundamental data structure in C5, used, e.g., to store data in a "sorted order" (binary search trees) or (heaps). There are more generalized data structure trees as 2-3 trees, B-trees. Each node has between 2 to 3 children.
(i) $\Rightarrow$ (iii): No circuit $\Rightarrow$ $G$ is connected with exactly $(n-1)$ edges.

Proof by induction on $n$:

$n=1$: The graph contains one vertex and 0 edges. This case is trivial.

$n>1$: If $G$ has no circuits then it must have a vertex with 1-degree, also called a leaf. Otherwise, if all degrees $\geq 2$, $G$ must contain a circuit.

We do not prove this part formally.
characterizing Trees

Proof of Theorem:

(i) $\Rightarrow$ (ii): A cycle in a graph is a circuit, and a circuit can be shortened to a cycle. Thus circuit $\iff$ cycle.

Let circuit $\Rightarrow$ a cycle that may be self-intersecting.

Circuit: $(V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8, V_i)$

Cycle $\subset$ $(V_1, V_3, V_5, V_7, V_8, V_i)$. 
Next, let us consider a leaf $V$ in $G$. We now remove $V$ from $G$ and the edge $e$ connecting $V$ to the rest of $G$.

Then we are left with a graph on $(n-1)$ vertices. Call this graph $G'$.

**Claim:** $G'$ is connected, with no circuits.

**Proof:** $G$ is connected, the removal of $V$ and $e$ does not violate connectivity. For any two vertices in $G'$ are connected by a path not passing through $V$. Thus, $G$ does not have circuits $\Rightarrow$ so does $G'$. 
No, apply induction hypothesis on \( G' \).

\( G' \) is connected w/o circuits, has \((n-1)\) vertices \( \Rightarrow G' \) must have \((n-2)\) edges.

Now recover \( G \) from \( G' \): return \( v \) and \( e \) to \( G \) \( \Rightarrow \) we now have \((n-2)+1=n-1\) edges.  

\[(\text{Done!})\]

\((\ref{3}) \Rightarrow (\ref{4})\):

Suppose for contradiction that \( G \) has a pair of vertices \( u,v \), with two different walks (using distinct edges) from \( u \) to \( v \). The these two walk create a circuit.
(i) \implies (ii): First (i) implies that between any pair of vertices \( u, v \) there is a path connecting them (the walk can be shortened to a path).

\( \implies G \) is connected.

Next, if \( G \) had a cycle, then it would contain two vertices \( u', v' \) along this cycle, but then we would have 2 paths between \( u' \) to \( v' \).
(iii) $\Rightarrow$ (ii): Assume $G$ is connected, with $(n-1)$ edges.

For contradiction, assume $G$ contains a cycle $\langle V_1, V_2, \ldots, V_k, V_1 \rangle$. Remove the edge $e = V_1, V_2$ from the graph.

This cycle is "broken", but $G$ is still connected.

Any path connecting $V, U$ using $e$, can now replace $e$ with $\langle V_1, V_k, \ldots, V_2 \rangle$.

$\Rightarrow G - e + e$ is a connected graph on $n$ vertices with $(n-2)$ edges.
We now keep removing edges closing cycles, until $G$ has no cycles.

Suppose we removed $k \geq 1$ edges.

$\Rightarrow G$ is now connected, has $n$ vertices [vertices are not removed], and contains $n-1-k$ edges.

Since we showed that a connected graph on $n$ vertices w/o cycles must have $(n-1)$ edges $\Rightarrow k = 0$.

$\Rightarrow$ Initially, $G$ had no cycles.

Remark:

Trees are smallest connected graphs; if we remove an edge $\Rightarrow$ graph is disconnected.
**Theorem:** Let $G$ be a tree on $n \geq 1$ vertices. Then it must contain at least 2 leaves.

**Proof:** We know that $G$ must contain $(n-1)$ edges. Let $V_1, V_2, \ldots, V_n$ be the vertices of $G$. Then

$$\sum_{i=1}^{n} \deg(V_i) = 2(n-1) = 2n - 2.$$

We know that $G$ must contain one leaf, say $V_1$. Then

if all other vertex degrees $\geq 2$, then the degree sum is $\geq 2(n-1) + 1 = 2n - 1$,

but this is a contradiction to
Minimum Spanning Trees

\[ W(T) = 3 \quad \text{and} \quad W(T) = 9 \]

We are given a graph \( G = (V, E) \) and a weight function \( w : E \to \mathbb{N} \).

\( w(e) = \text{weight of } e \)

\( W(T) = \text{weight of spanning tree} \)

We will study two algorithms:
- Kruskal.
- Prim.
How to find a spanning tree?

There are many more spanning trees! Typically, number of spanning trees can be exponential in \( n \).

Theorem [Who proof?]

outside \( \rightarrow \) 8 trees

The course scope, but number of binary trees related to our discussion on \( n \) vertices is \( \frac{2n!}{n+1} \),

where \( \binom{2n}{n} \) is the \( n \)th catalan number.
Graph Traversal Algorithms

We give only a sketch of these algorithms:

BFS:

We begin with $s$, then visit $V_1, V_2, V_3$ [put them in a queue]. Then process $V_1, V_2, V_3$ in this order: From $V_1$ we go to $V_5$, $V_5$ is already visited by $s$ then $V_1$ does not visit it. Then $V_2$ visits $V_6$, $V_5$ is already visited by $V_1$, $V_2$ visits $V_7$. At last, $V_5$ visits $V_8$.

The marked (red) edges are the tree edges.
procedure BFs (G, s)
1. construct an empty queue Q.
Mark each vertex in G as "unseen".
2. Mark s as "seen".
3. enqueue s into Q.
4. while (Q ≠ Ø) do:
5.    u = dequeue(Q)
6.    for each neighbor w of u do
7.    if w is unseen then
8.        mark w as "seen".
9.        enqueue w into Q.
10.   end if
11. end for
12. end while.
The BFS algorithm finds a spanning tree, but not necessarily the minimum one.

Running time of BFS is proportional to:

\[ |V| + 2|E| = |V| + \sum_{v \in V} \text{deg}(v) \]

The BFS algorithm has many interesting properties, covered in CS courses:
- It can detect cycles in graphs.
- Find all shortest paths (according to number of edges) from a source to all other vertices.
- Compute a pair of vertices with largest edge-distance, etc.
We begin with $s$, and keep walking along edges as long as we do not see a leaf. We backtrack at leaves, vertices that have been visited before, we never close a cycle, we walk only along tree edges.

DFS algorithm has many interesting properties...

DFS creates a spanning tree.
Kirchhoff's Theorem

This theorem applies to any row $i$ and column $j$, that is, to any cofactor of $M$. \( \Rightarrow \) all cofactors have the same value

8 spanning trees.

Theorem:

Number of labeled trees with \( n \geq 3 \) vertices is \( n^{n-2} \). [Application of Kirchhoff's theorem]