12 Trees

12.1 Trees and Their Properties
Trees

A circuit in a graph is a walk that starts and ends at the same point and traverses each edge at most once.

A tree is a connected graph with no circuits.
Which of the following graphs are trees?
TREES

List all trees with 5 or fewer vertices up to isomorphism.
Applications of Trees

and many more...
**Characterizing Trees**

**Theorem.** Let $G$ be a graph with $n$ vertices. The following are equivalent:

(i) $G$ is a tree (i.e. $G$ is connected with no circuits).
(ii) $G$ is connected and has no cycles.
(iii) $G$ is connected and has $n-1$ edges.
(iv) Between any two vertices of $G$ there is a unique walk that does not repeat any edges.

Also:

(v) $G$ has $n-1$ edges and no cycles
(vi) $G$ is connected, but removing any edge makes it disconnected.
(vii) $G$ has no cycles, but adding any edge creates one.

etc...
**Application To Chemistry**

A hydrocarbon has the form \( \text{C}_n \text{H}_{2n+2} \). Carbon has degree 4 and Hydrogen has degree 1.

So we get a graph with \( n+2n+2 = 3n+2 \) vertices and \( \frac{2n+2+4n}{2} = 3n+1 \) edges \( \rightsquigarrow \) tree!

**Problem.** Find all hydrocarbons for \( n=1,2,3,4 \).

methane  ethane  propane  butane  isobutane
Characterizing Trees

Theorem. Let $G$ be a graph with $n$ vertices. The following are equivalent:

(i) $G$ is a tree (i.e. $G$ is connected with no circuits)
(ii) $G$ is connected and has no cycles.
(iii) $G$ is connected and has $n-1$ edges.
(iv) Between any two vertices of $G$ there is a unique walk that does not repeat any edges.

Sketch of Proof.

$(i) \iff (ii)$ Show, for any graph: cycle $\iff$ circuit
   Idea: prune a circuit to get a cycle.

$(i) \implies (iii)$ Show that no circuits $\implies$ degree 1 vertex.
   Then use induction on $n$.

$(i) \implies (iv)$ Given two different paths, construct a circuit.

$(iv) \implies (ii)$ In a cycle, can find two paths btw same vertices.

$(iii) \implies (ii)$ Remove $k$ edges to get a tree. Show $k=0$. $\blacksquare$
12.2 Spanning Trees
Spanning Trees

A spanning tree for a graph \( G \) is a subgraph that is a tree and that contains every vertex.

Note: Only connected graphs have spanning trees.

If we have a subgraph \( H \) of a weighted graph \( G \), the weight of \( H \) is the sum of the weights of its edges.

A minimal spanning tree for a weighted graph is a spanning tree of least weight.

Application: Given a network of roads, which roads should you pave so that (a) all towns are connected and (b) we use the least amount of asphalt?
Spanning Trees

How to find a spanning tree?

One answer: Delete all edges until there are no cycles.

Example. How many spanning trees can you find?

```
// Diagram
```

Question. How to find all spanning trees? How many are there?

Could hunt for cycles, delete edges. Inefficient!
Depth-first Search and Breadth-first Search

Depth-first: Start at some point in the graph. Draw a long path, go as far as possible. When you hit a wall (= degree 1 vertex), or an edge that creates a cycle with your path, back up one step and go in a new direction.

![Graph diagram]

Breadth-first: Use as many edges from start point as possible. Then from the endpoints of all those edges use as many edges as possible, etc.
Kirchhoff's Theorem

Given a graph with vertices $v_1, \ldots, v_n$, make a matrix $M$ with $(i,i)$-entry the degree of $v_i$ and all other $(i,j)$-entries given by: $-1$ if $v_i v_j$ is an edge, 0 otherwise.

**Theorem.** Given a graph $G$, make the matrix $M$ as above. Delete the $i$th row and the $j$th column to obtain the matrix $M'$. Then:

$$(-1)^{i+j} \det(M') = \text{# spanning trees for } G.$$
**Kirchhoff’s Theorem**

**Example.**

\[
\begin{pmatrix}
3 & -1 & -1 & -1 \\
-1 & 3 & -1 & -1 \\
-1 & -1 & 2 & 0 \\
-1 & -1 & 0 & 2
\end{pmatrix}
\]

\[
\begin{vmatrix}
3 & -1 & -1 \\
-1 & 2 & 0 \\
-1 & 0 & 2
\end{vmatrix} = 3 \begin{vmatrix}
2 & 0 \\
0 & 2
\end{vmatrix} - (-1) \begin{vmatrix}
-1 & 0 \\
-1 & 2
\end{vmatrix} + (-1) \begin{vmatrix}
-1 & 2 \\
-1 & 0
\end{vmatrix}
\]

\[
= 3 \cdot 4 + 1 \cdot (-2) - 1 \cdot 2
\]

\[
= 8.
\]

What are the 8 spanning trees? Find them all!
12.3 Minimal Spanning Tree Algorithms
Kruskal's Algorithm

Goal: Find a minimal spanning tree for a given graph.
Want something more efficient than enumerating all trees.

The Algorithm. Set $T = \emptyset$.
Consider all edges $e$ so $T \cup \{e\}$ has no circuits.
Choose the edge $e$ of smallest weight with this property.
Replace $T$ with $T \cup \{e\}$.
Repeat until $T$ is a spanning tree.

Note: The number of steps is one less than the # of vertices.

Kruskal's algorithm is an example of a "greedy algorithm"
Kruskal's Algorithm

Find minimal spanning trees for the following weighted graphs.
Krushkal's Algorithm

Why does the algorithm work?

Let $e_1,...,e_{n-1}$ be the edges chosen by Krushkal's algorithm, in order.

Prove the following statement by induction:

\( \{e_1,...,e_k\} \) is contained in some minimal spanning tree.

Base case: $k=0$, i.e. $\emptyset$ contained in some minimal spanning tree.  

Suppose \( \{e_1,...,e_k\} \) contained in some minimal spanning tree $T$, but $e_{k+1}$ is not in $T$. \( \Rightarrow \) $T \cup e_{k+1}$ has a cycle.

There is an edge $f$ contained in this cycle that is not equal to $e_1,...,e_{k+1}$ (the $e_i$ form a tree, so they form no cycles).

Now, $f$ and $e_{k+1}$ have same weight, otherwise weight of $T-f+e_{k+1}$ is less than weight of $T$. We see $T-f+e_{k+1}$ is the desired tree.  
\[ \blacksquare \]
**Prim's Algorithm**

Idea: Grow a tree from a vertex.

The algorithm.

- Set $T = V$ (any vertex)
- Choose an edge $e$ of minimal weight so $T \cup \{e\}$ is a tree
- Replace $T$ with $T \cup \{e\}$
- Repeat until $T$ is a spanning tree.

Note: We know $T \cup \{e\}$ is a tree if $T \cap e$ is a single vertex.
Prim's Algorithm

Find minimal spanning trees for the following weighted graphs.
Krushkal's Algorithm vs. Prim's Algorithm

What is the complexity?  

Size = # edges  

Cost = # comparisons

Krushkal: \( O(n \log n + n^2) \)

Prim: \( O(n^2) \)

Check these! Idea: order the remaining edges. Then, need to check which can be added to the current tree by comparing the endpoints of each edge with the vertices of the current tree.

The advantage over Krushkal's algorithm is that there are fewer edges to check at each step. In fact, Prim is \( O(n^2) \).