Trees: undirected, connected, acyclic graphs

Trees are the extreme in minimalistic connectivity. Trees may be rooted, or not.

Do both of these examples meet the definition of a tree?
Trees

Tree = undirected, connected, acyclic graph.

Properties of trees

• **pt 1:** tree of $n$ nodes has $n-1$ edges

• **pt 2:** undirected, connected graph, $G = (V, E)$ with $|E| = |V| - 1$ is a tree.

• **pt 3:** undirected graph is a tree if and only if there is a unique path between any pair of nodes.
Trees: undirected, connected, acyclic graphs

Proof by induction:

$n=1$ cannot have fewer than $n-1=0$ edges. Adding an edge would produce a cycle.

$n=2$: If we remove the edge the tree is no longer connected. If we add an edge, it would produce a cycle: $\circ$, $\bullet$, $\circ$.

$n=|V|$, $|E|=|V|-1$ If we have fewer than $n-1$ edges, the tree will not be connected.

If we have $n-1$ edges and the tree is connected, then there is a path from any node to any node. Adding 1 more edge will produce another path between the endpoints of the new edge. This produces a cycle.
Trees: undirected, connected, acyclic graphs

**pt 2:** An undirected, connected graph, $G = (V, E)$ with $|E| = |V| - 1$ is a tree.

**proof:**

We need to show that $G$ is also acyclic under the given conditions.

If $G$ is undirected and connected, and has $|E| = |V| - 1$, then there is a path from any node to any node (connected). Removing any edge will disconnect the graph. $\Rightarrow$ there are no extra edges. $\Rightarrow$ There are no cycles.
Trees: undirected, connected, acyclic graphs

pt3: undirected graph is a tree if and only if there is a unique path between any pair of nodes.

proof:
we need to prove both directions of the if and only if:

unique paths $\Rightarrow$ graph is tree
• unique paths implies paths exist which implies connected.
• unique paths implies no cycles.
undirected, connected, acyclic graph is a tree

graph is tree $\Rightarrow$ unique paths
• connected implies paths exist
• acyclic implies unique paths

√
Minimum Spanning Trees (MST)

An undirected, connected graph may have unnecessary cycles. For example, a graph may represent a proposed network of wormholes between habitable planets in the galaxy. The nodes of the graph represent the planets and the edges represent the wormholes. The edge weights represent the construction cost of the wormholes. The galactic society wants to enable travel (paths) between any pair of planets, but wants to minimize the construction cost.
Minimum Spanning Trees (MST)

Galactic Wormhole Construction Project Proposal

Possible wormholes and costs in PetaBucks.
Maximum Cost = 26 PB

Proposed wormhole construction and costs in PetaBucks.
Minimum Cost = 10 PB.

There are 3 more MSTs with the same cost. Can you find them?
Minimum Spanning Trees (MST)

Construction: We will use a greedy algorithm to construct MSTs from undirected, connected graphs. Greedy means that we always choose the next step for immediately best choice, ignoring the future choices to be made.

There are 2 obvious options for greedy MST algorithms:
1: Start with the full graph, removing heavy edges that produce cycles.
2: Start with no edges, adding light edges that don’t produce cycles.

It turns out that option 2 is easier to implement efficiently, so we will use it.
def generic_MST(G = (V, E)):
    X = {}  # start with empty set of edges included in MST
    while |X| < |V| - 1:
        pick a set $S \subseteq V$ such that $S$ and $V - S$ are not connected by $X$
        find $e \in E$ such that $e$ is the minimum cost edge between $S$ and $V - S$.
        $X = X \cup \{e\}$

The algorithm for selecting $S \subseteq V$, and the data structures used to represent $S$, $V - S$, and $X$ result in specific MST algorithms. We will look at Prim's and Kruskal's algorithms for MST.
Minimum Spanning Trees (MST)

Greedy Algorithm: Prim

```python
def Prim(G = (V, E)):
    for u in V:
        cost[u] = float('inf')
        prev[u] = nil
    u₀ = any node ∈ V
    cost[u₀] = 0
    Q = make_queue(V, cost)
    while !Q.empty():
        u = Q.deletemin()
        for (u, v) ∈ E and v ∈ S:
            if cost[v] > l(u, v):
                cost[v] = l(u, v)
                prev[v] = u
                Q.decrease_key(v)

    # prev[] defines the |V|-1
    # edges of the MST.
```

choose alphabetically when choice is allowed:

<table>
<thead>
<tr>
<th>set</th>
<th>S</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>{}</td>
<td>3</td>
<td>∞/1</td>
<td>∞/1</td>
<td>∞/1</td>
<td>∞/1</td>
<td>∞/1</td>
<td>∞/1</td>
</tr>
<tr>
<td>{}</td>
<td>2/A</td>
<td>∞/1</td>
<td>4/A</td>
<td>3/A</td>
<td>∞/1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{}</td>
<td>3/B</td>
<td>3/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{}</td>
<td>2/F</td>
<td>4/A</td>
<td>3/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{}</td>
<td>4/A</td>
<td>3/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{}</td>
<td>3/E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

MST Edge
def kruskal(G=(V,E)):
    for u in V:  # create |V| independent sets
        make_set(u)
    X = Set()  # start with no edges in tree
    E' = sorted by weight E
    for (u,v) in E':
        if u and v are not in same set:
            X = X U \{(u,v)\}
        merge the sets u and v belong to.

    # X defines the MST
Minimum Spanning Trees (MST): Greedy Algorithm: Kruskal

Choose alphabetically when choice is possible. 

$E^1$ = sorted version of edges

<table>
<thead>
<tr>
<th>$(u,v)$</th>
<th>$l(u,v)$</th>
<th>$(u,v)$</th>
<th>$l(u,v)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>AD</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>BE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>CF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>BF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>EF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>BC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>DE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>BD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>AB</td>
<td></td>
</tr>
</tbody>
</table>

MST Edges

<table>
<thead>
<tr>
<th>$(u,v)$</th>
<th>$\notin$</th>
<th>Node Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A$</td>
<td>${A3, B3, C3, D3, E3, F3}$</td>
</tr>
<tr>
<td>AD</td>
<td>$A$</td>
<td>${A3, B3, C3, D3, E3, F3}$</td>
</tr>
<tr>
<td>BE</td>
<td>$A$</td>
<td>${A3, B3, C3, D3, E3, F3}$</td>
</tr>
<tr>
<td>CF</td>
<td>$A$</td>
<td>${A3, B3, C3, D3, E3, F3}$</td>
</tr>
<tr>
<td>BF</td>
<td>$A$</td>
<td>${A3, B3, C3, D3, E3, F3}$</td>
</tr>
<tr>
<td>EF</td>
<td>$A$</td>
<td>${A3, B3, C3, D3, E3, F3}$</td>
</tr>
<tr>
<td>BC</td>
<td>$A$</td>
<td>${A3, B3, C3, D3, E3, F3}$</td>
</tr>
<tr>
<td>DE</td>
<td>$A$</td>
<td>${A3, B3, C3, D3, E3, F3}$</td>
</tr>
<tr>
<td>BD</td>
<td>$A$</td>
<td>${A3, B3, C3, D3, E3, F3}$</td>
</tr>
<tr>
<td>AB</td>
<td>$A$</td>
<td>${A3, B3, C3, D3, E3, F3}$</td>
</tr>
</tbody>
</table>
Minimum Spanning Trees (MST): The cut property

Given: \( G = (V, E) \) undirected connected graph
\( T \subseteq E \), a MST for \( G \).

\( S \subseteq V \) and \( V - S \), a cut of the nodes
\( X \subseteq T \), that does not cross the cut.

i.e. \( X \) contains no edges between \( S \) and \( V - S \).

\( e \in E \), the smallest weighted edge that crosses the cut.
\( e' \in E \), some other edge that crosses the cut.

The cut property:

Given \( G, T, S, V - S, X \) and \( e \) as described; \( X \cup \{ e \} \) is part of some MST of \( G \).

\( X = \{ AC, BC, BE, CD, FG, GH \} \)

Other edges exist, but are not drawn.
Minimum Spanning Trees (MST): The cut property

Proof: Let's compare the weights of 2 spanning trees.

T contains $X \cup \exists e'$

$T'$ contains $X \cup \exists e'$, where $e'$ also crosses $S, V-S$ cut.

Let's add $e$ to $T'$, then remove $e'$ from $T'$ to remove the cycle.

$T \leftarrow T' \cup \exists e' - \exists e'$

$T$ is still a tree; we added and removed 1 edge, the new edge replaces the connectivity of the removed edge; still connected and acyclic.

$\text{weight}(T) = \text{weight}(T') + \text{weight}(e) - \text{weight}(e')$

<table>
<thead>
<tr>
<th>Case</th>
<th>Result</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight$(e) &lt;$ weight$(e')$</td>
<td>$\text{weight}(T) &lt; \text{weight}(T')$</td>
<td>$e$ is lightest by definition, therefore weight$(T) \leq$ weight$(T')$, for any choice of $e'$.</td>
</tr>
<tr>
<td>weight$(e) =$ weight$(e')$</td>
<td>$\text{weight}(T) = \text{weight}(T')$</td>
<td></td>
</tr>
<tr>
<td>weight$(e') &gt;$ weight$(e')$</td>
<td>$\text{weight}(T) &gt; \text{weight}(T')$</td>
<td>$e$ is the lightest by definition, this case doesn't occur.</td>
</tr>
</tbody>
</table>

$\Rightarrow T$ is an MST of $G$. 
Minimum Spanning Trees (MST) : Greedy Algorithms

The greedy choice in both MST algorithms, Kruskal and Prim, rely on the cut property. The greedy step is to choose the minimal cost edge to connect nodes in the graph. The cut property states this edge must be part of a MST. The proof of the cut property proves the correctness of both MST algorithms.