Spanning Trees of undirected graphs

For a simple undirected graph G, a **spanning tree** of G is a subgraph T of G such that T is a tree and T contains every vertex of G.

Theorem: Every connected graph G has a spanning tree.

Proof: While there is a circuit in G, remove one edge of the circuit. Repeat. Removing one edge of the circuit does not change connectivity, and eventually no circuits can remain (because there are only finitely many edges to be removed). So, the end result is a tree which is a subtree of G. $\square$

Spanning Trees of undirected graphs

For a simple undirected graph G, a **spanning tree** of G is a subgraph T of G such that T is a tree and T contains every vertex of G.

Theorem: Every connected graph G has a spanning tree.

Proof: While there is a circuit in G, remove one edge of the circuit. Repeat. Removing one edge of the circuit does not change connectivity, and eventually no circuits can remain (because there are only finitely many edges to be removed). So, the end result is a tree which is a subtree of G. $\square$

Question: Given a graph G, can we efficiently compute a spanning tree for G?

Spanning Trees of undirected graphs

For a simple undirected graph G, a **spanning tree** of G is a subgraph T of G such that T is a tree and T contains every vertex of G.

Theorem: Every connected graph G has a spanning tree.

Proof: While there is a circuit in G, remove one edge of the circuit. Repeat. Removing one edge of the circuit does not change connectivity, and eventually no circuits can remain (because there are only finitely many edges to be removed). So, the end result is a tree which is a subtree of G. □

Question: Given a graph G, can we efficiently compute a spanning tree for G?

Answer: Yes. Even for edge-weighted graphs, we can compute a **minimum-cost spanning tree** efficiently. (The cost of a spanning tree is the sum of its edge costs.)

Prim’s algorithm for a minimum spanning tree

Input: Connected, edge-weighted, undirected graph $G = (V, E, w)$.

Output: A minimum-cost spanning tree T for G.

Algorithm:

```
Initialize: $T := \{e\}$, where $e$ is a minimum-weight edge in $E$.
for $i := 1$ to $n - 2$ do
    Let $e^1 :=$ a minimum-weight edge incident to
              some vertex in $T$, and not forming a circuit
              if added to $T$;
    $T := T \cup \{e^1\}$;
end for
Output the tree $T$.
```