2 Trees

What is a tree?

Forests and Trees: A *forest* is a graph with no cycles, a *tree* is a connected forest (so every component of a forest is a tree).

Theorem 2.1 *If G is a forest, then $comp(G) = |V(G)| - |E(G)|$.*

Proof: We proceed by induction on $|E(G)|$. As a base, if $|E(G)| = 0$, then every component is an isolated vertex, so $comp(G) = v(G)$ as required. For the inductive step, we may assume $|E(G)| > 0$ and choose an edge $e \in E(G)$. Now, by Proposition 1.7 and induction on $G - e$, we have

$$\begin{aligned} comp(G) &= comp(G-e) - 1 \\ &= |V(G-e)| - |E(G-e)| - 1 \\ &= |V(G)| - |E(G)|. \end{aligned}$$

□

Corollary 2.2 *If G is a tree, then $|V(T)| = |E(T)| + 1$.*

Leaf: A *leaf* is a vertex of degree 1.

Proposition 2.3 *Let T be a tree with $|V(T)| \geq 2$. Then T has ≥ 2 leaf vertices. Further, if T has exactly 2 leaf vertices, then T is a path.*

Proof: By the above corollary and Theorem 1.1 we have

$$\begin{aligned} 2 &= 2|V(T)| - 2|E(T)| \\ &= 2|V(T)| - \sum_{v \in V(T)} deg(v) \\ &= \sum_{v \in V(T)} (2 - deg(v)). \end{aligned}$$

Since $|V(T)| \geq 2$, every vertex has degree > 0. It follows immediately from this and the above equation that T has ≥ 2 leaf vertices. Further, if T has exactly two leaf vertices, then every other vertex of T has degree 2, and it follows that T is a path. □

Lemma 2.4 *If T is a tree and v is a leaf of T, then $T - v$ is a tree.*

Proof: It is immediate that $T - v$ has no cycle and is connected. $\square$

Note: The above lemma gives us a powerful inductive tool for proving properties of trees.

Proposition 2.5 *If T is a tree and $u, v \in V(T)$, then there is a unique path from u to v.*

Proof: We proceed by induction on $V(T)$. If there is a leaf $w \neq u, v$, then the result follows by applying induction to $T - w$. Otherwise, the result follows from Proposition 2.3. $\square$

Spanning Tree: If $T \subseteq G$ is a tree and $V(T) = V(G)$, we call T is a *spanning tree of* G.

If G is a graph, $H \subseteq G$ and $e \in E(G)$, we let $H + e$ be the subgraph of G obtained from H by adding the edge e and the endpoints of e.

Theorem 2.6 *Let G be a connected graph with $v(G) > 1$. If H is a subgraph of G chosen according to one of the following conditions, then H is a spanning tree.*

(i) *$H \subseteq G$ is minimal so that H is connected and $V(H) = V(G)$.*

(ii) *$H \subseteq G$ is maximal so that H has no cycles.*

Proof: For (i), note that if H has a cycle C and $e \in E(C)$, then $H - e$ is connected (by Proposition 1.7), which contradicts the minimality of H. Thus H has no cycle, and it is a spanning tree.

For (ii), note first that $V(H) = V(G)$ by the maximality of H. If X is the vertex set of a component of H and $X \neq V(G)$, then it follows from the connectivity of G that there exists an edge e of G with one end in X and one end in $V(G) \setminus X$. Now, $H + e$ has no cycle, contradicting the maximality of H. Thus, H has only one component, and it is a spanning tree. $\square$

Proposition 2.7 *Let G be a graph with $|E(G)| = |V(G)| - 1$.*

(i) *If G has no cycle, then it is a tree.*

(ii) *If G is connected, it is a tree.*

Proof: Part (i) follows immediately from Theorem 2.1. For (ii), we have a connected graph G, so we may choose a spanning tree $T \subseteq G$. But then $|E(G)| = |V(G)| - 1 = |V(T)| - 1 = |E(T)|$ so $T = G$ and G is a tree. $\square$

Kruskal's Algorithm

Fundamental Cycles: Let T be a spanning tree of G and let $f \in E(G) \setminus E(T)$. A cycle $C \subseteq G$ with $f \in E(C)$ and $C - f$ a path of T is called a *fundamental cycle of* f *with respect to* T.

Proposition 2.8 *If* T *is a spanning tree of* G *and* $f \in E(G) \setminus E(T)$*, then there is exactly one fundamental cycle of* f *with respect to* T*.*

Proof: This follows immediately from Proposition 2.5. $\square$

Proposition 2.9 *Let* T *be a spanning tree of* G*, let* $e \in E(T)$ *and* $f \in E(G) \setminus E(T)$*.*

(i) *if* e *is in the fundamental cycle of* f*, then* $T - e + f$ *is a tree*

(ii) *if* f *has one end in each component of* $T - e$*, then* $T - e + f$ *is a tree.*

Proof: For (i), note that if e is in the fundamental cycle of f, then $T + f - e$ is a connected graph with one fewer edge than vertex, so it is a tree by Proposition 2.7.

For (ii), observe that if f has one end in each component of $T - e$, then $T - e + f$ is a forest (since f is a cut-edge of $T - e + f$ and $T - e$ is a forest) with one fewer edge than vertex, so it is a tree by Proposition 2.7. $\square$

Weighted graphs and min-cost trees: A *weighted graph* is a graph G together with a *weight* function on the edges $w : E(G) \to \mathbb{R}$. If $T \subseteq G$ is a spanning tree for which $\sum_{e \in E(T)} w(e)$ is minimum, we call T a *min-cost tree.*

Kruskal's Algorithm:

input: A weighted graph G.

output: A min-cost tree T.

procedure: Choose a sequence of edges $e_1, e_2, \ldots, e_m$ according to the rule that e_i is an edge of minimum weight in $E(G) \setminus \{e_1, \ldots, e_{i-1}\}$ so that $\{e_1, \ldots, e_i\}$ does not contain the edge set of a cycle. When no such edge exists, stop and return the subgraph T consisting of all the vertices, and all chosen edges $\{e_1, \ldots, e_m\}$.

Theorem 2.10 *Let G be a connected weighted graph with weight function w. If w is one-to-one, then Kruskal's algorithm returns the unique min-cost tree for G.*

Proof: Let $e_1, \ldots, e_m$ be the sequence of edges chosen by Kruskal's Algorithm, and let T be the subgraph returned by it. It follows from the connectivity of G and Theorem 2.6 that T is a spanning tree. Suppose (for a contradiction) that there is a min-cost tree $T' \neq T$ and let f be the edge of minimum weight in the set $E(T') \setminus E(T)$. Let C be the fundamental cycle of f with respect to T and let $e_i \in E(C)$. Now, part (i) of Proposition 2.9 shows that $T - e_i + f$ is a tree, so, in particular, there is no cycle with edge set included in $\{e_1, \ldots, e_{i-1}, f\}$. So, by Kruskal's Algorithm, we must have $w(f) > w(e_i)$. It follows that every edge in $C - f$ has smaller weight than f. However, $C - f$ is a path with the same ends as f, so there must exist an edge in $C - f$ with one end in each component of $T' - f$. But then, part (ii) of Proposition 2.9 gives us a contradiction to the assumption that T' is a min-cost tree. This contradiction proves that T is the unique min-cost tree of G, as required. $\square$

Distance and Dijkstra's Algorithm

Distance: If $u, v \in V(G)$, the *distance from* u to v is the length of the shortest path from u to v, or ∞ if no such path exists. If G is a weighted graph, then the *distance from* u to v is the minimum of $\sum_{e \in E(P)} w(e)$ over all paths from u to v, or ∞ if no such path exists. In either case, we denote this by $dist_G(u, v)$, or just $dist(u, v)$ if G is clear from context.

Observation 2.11 *The distance function obeys the triangle inequality. So, for all $u, v, w \in V(G)$ we have:*

$$dist(u, v) + dist(v, w) \geq dist(u, w).$$

Shortest Path Tree: If G is a weighted graph and $r \in V(G)$, a tree $T \subseteq G$ (not nec. spanning) is a *shortest path tree* for r if $dist_T(r, v) = dist_G(r, v)$ for every $v \in V(T)$.

Dijkstra's Algorithm:

input: A connected weighted graph G with $w : E(G) \rightarrow \mathbb{R}^+$ and a vertex r.

output: A shortest path tree T for r.

procedure: Start with T_1 the tree consisting of the vertex r. At step i, we have $T_i \subseteq G$. If $V(T_i) = V(G)$ stop and return $T = T_i$. Otherwise, choose an edge uv with $u \in V(T_i)$ and $v \in V(G) \setminus V(T_i)$ so that $w(uv) + dist_T(r, u)$ is minimum and set $T_{i+1} = T_i + uv$.

Theorem 2.12 *Dijkstra's algorithm returns a shortest path tree for r.*

Proof: We prove by induction on i that each T_i in the algorithm is a shortest path tree for r. As a base, note that this is true for T_1 (since all weights are nonnegative). For the inductive step, we shall show that T_{i+1} is a shortest path tree for r assuming this holds for T_i. Assume that $T_{i+1} = T_i + uv$. Let P be the path of minimum distance in G from r to v, let y be the first vertex of P which is not in the tree T, and let the previous edge be xy. Then we have

$$\begin{aligned} dist_G(r,v) &= \sum_{e \in E(P)} w(e) \\ &\geq dist_{T_i}(r,x) + w(xy) \\ &\geq dist_{T_i}(r,u) + w(uv) \\ &= dist_{T_{i+1}}(r,v). \end{aligned}$$

It follows that T_{i+1} is a shortest path tree for r, as required. $\square$

Prüfer codes

In this section, we consider two graphs with the same vertex set to be the same if they have the same adjacencies.

Prüfer Encoding

input: A tree T with vertex set $S \subseteq \mathbb{Z}$ where $|S| = n \geq 2$.

output: A sequence $\mathbf{a} = (a_1, \ldots, a_{n-2})$ with elements in S.

procedure: At step i, we delete from T the smallest leaf vertex v, and we set a_i to be the vertex adjacent to v.

Observation 2.13 *If* $\mathbf{a}$ *is the Prüfer Encoding of the tree T, then the set of vertices which appear in* $\mathbf{a}$ *is exactly the set of non-leaf nodes of T.*

Prüfer Decoding

input:	A sequence $\mathbf{a} = (a_1, \ldots, a_{n-2})$ with elements in $S \subseteq \mathbb{Z}$ where $\lvert S \rvert = n \geq 2$.
output:	An tree T with vertex set S.
procedure:	Start with the graph T where $V(T) = S$ and $E(T) = \emptyset$, and with all vertices unmarked. At step i, add an edge from the smallest unmarked vertex v which does not appear in $(a_i, \ldots, a_{n-2})$ to a_i and mark v. After step $n-2$ is complete, add an edge between the two remaining unmarked vertices and stop.

Theorem 2.14 *Let $S \subseteq \mathbb{Z}$ with $\lvert S \rvert = n \geq 2$. Prüfer Encoding and Decoding are inverse bijections between the set of trees with vertex set S and the set of sequences of length $n-2$ with elements in S.*

Proof: We proceed by induction on n. As a base, observe that when $n = 2$, there is a single sequence of length 0 and a single tree with vertex set S, and the encoding/decoding operations exchange these.

For the inductive step, we may assume $n \geq 3$. Let T be a tree with vertex set S, let $\mathbf{a} = (a_1, \ldots, a_{n-2})$ be the encoding of T, and let T' be the decoding of $\mathbf{a}$. Let v be the smallest leaf of T. Then, by the encoding process, v is adjacent to a_1, and v is smaller than any other vertex which does not appear in $\mathbf{a}$, since (by the above observation) all such vertices are leaves. It follows from the decoding rules that in the tree T', the vertex v is a leaf adjacent to a_1. Now, $T - v$ encodes to $(a_2, \ldots, a_n)$ which decodes to $T' - v$, so by induction $T - v$ and $T' - v$ are the same, and it follows that T and T' are the same.

To complete the inductive step, we still need to show that every sequence $\mathbf{a} = (a_1, \ldots, a_{n-2})$ is the encoding of some tree. To see this, let $v \in S$ be the smallest element which does not appear in $\mathbf{a}$. Then, by induction $(a_2, \ldots, a_{n-2})$ is the encoding of some tree T with vertex set $S \setminus \{v\}$, and we find that $\mathbf{a}$ is an encoding of the tree obtained from T by adding the vertex v and the edge va_1. This completes the proof. $\Box$

Corollary 2.15 *For every $n \geq 2$, there are exactly n^{n-2} trees with vertex set $\{1, 2, \ldots, n\}$.*