Summary on Lecture 9, August 3, 2015

Trees: definitions and basic properties

Definition. A connected graph $G = (V, E)$ is a *tree* is G has no cycles.

Theorem 1. *Let $T = (V, E)$ be a tree, and $v, v' \in V(T)$. Then there exists a unique path from v to v'.*

Proof. Assume there are two different paths connecting v and v'. Then there is a cycle. □

Definition. Let $G = (V, E)$ be a graph. A tree $T \subset G$ is a *spanning tree of* G if $V(T) = V(E)$.

Theorem 2. *Let $G = (V, E)$ be a finite graph. Then G is connected if and only if there exists a spanning tree T of G.*

Proof. Let $T \subset G$ be a spanning tree. Then for any two vertices $v, v' \in V(T) = V(G)$ there exists a path in T (and, consequently, in G) which connects v and v'.

On the other hand, if G is connected, then we remove all loops from G to obtain graph G_1, which is still connecetd. If G_1 is a tree, we are done. If not, we find a cycle in G_1, and choose an edge e_1 in that cycle. Then $G_2 = G_1 \setminus \{e_1\}$ is a connected graph. If G_2 has a cycle, we repeat the procedure. Finally we will obtain a connected graph $G_k \subset G$ which has no cycles since the original graph is finite, i.e., G_k is a spanning tree. □

Let $G = (V, E)$ be a graph. We say that a vertex $v \in V$ is a *leaf* of G if $\deg v = 1$. We need the following observation:

Lemma 1. *Let $T = (V, E)$ be a finite tree. There there are at least two leaves v, v' in T.*

Proof. Consider a longest path in T, and let v and v' are its end-points. We notice that $v \neq v'$ since T is a tree. Then v and v' are both leaves, indeed, if not, we can extend a path by at least one edge. □

Theorem 2. *Let $T = (V, E)$ be a finite tree. Then $|V| = |E| + 1$.*

Proof. Induction on $k = |V|$. Theorem 2 obviously holds if $k = 1$ and $k = 2$. Assume Theorem 2 holds for all trees $T' = (V', E')$ with $|V| \leq n$. Consider a tree $T = (V, E)$ with $|V| = n + 1$. Then by Lemma 1, there exists a leaf $v \in V'$ with a single edge e. We prune the tree T at v, to obtain a tree $T' = (V', E')$, where $V' = V \setminus \{v\}$, $E' = E \setminus \{e\}$. By induction, $|V'| = |E'| + 1$. Since $|V| = |V'| + 1$ and $|E| = |E'| + 1$, we obtain that $|V| = |E| + 1$. □

Theorem 3. *The following statements are equivalent:*

(a) *A graph $G = (V, E)$ is a finite tree.*

(b) *A graph $G = (V, E)$ is connected, but a removal of any edge will make it disconnected.*

(c) *A graph $G = (V, E)$ contains no cycles and $|V| = |E| + 1$.*

(d) *A graph $G = (V, E)$ is connected and $|V| = |E| + 1$.*

Proof. (a)$\Longrightarrow$(b) Let $T = (V, E)$ be a finite tree. Assume that a removing an edge e from G keeps $G \setminus \{e\}$ connected. Let v, v' be the end-vertices of e. Then there exists a path in $G \setminus \{e\}$ connecting v and v'. Then we put the edge e back an we obtain a cycle. Thus G could not be a tree in the first place. Contradiction. Thus (a)$\Longrightarrow$(b).

(b)$\Longrightarrow$(c) Let $G = (V, E)$ be as in (b), however, G contains a cycle. Then we can remove an edge e from such a cycle, and the graph $G \setminus \{e\}$ is still connected. Contradiction. Thus $G = (V, E)$ has no cycles, and, by definition, G is a tree. By Theorem 2, $|V| = |E| + 1$.

(c)$\Longrightarrow$(d) Let $G = (V, E)$ be as in (c), however, G is not connected. It means that $G = G_1 \cup \cdots \cup G_r$, where G_i are connective components of G and $r \geq 2$. Then every component G_i has no cycles and it is connected, thus G_i is a tree by definition. Theorem 2 gives that

$$|V(G_1)| = |E(G_1)| + 1, \; \cdots \; |V(G_r)| = |E(G_r)| + 1.$$

We obtain:

$$|V(G)| = |V(G_1)| + \cdots + |V(G_r)| = |E(G_1)| + \cdots + |E(G_r)| + r = |E(G)| + r.$$

At the same time, we have that $|V(G)| = |E(G)| + 1$. Thus $r = 1$. Contradiction.

(d)$\Longrightarrow$**(a)** Let $G = (V, E)$ be as in (d). Then G has a spanning tree $T \subset G$ by Theorem 1. Then $V(T) = V(G)$ and $|E(T)| \leq |E(G)|$. Since T is a tree, $|V(T)| = |E(T)| + 1$, and by assumption, $|V(G)| = |E(G)| + 1$. Thus $|E(T)| = |E(G)|$, i.e., $T = G$. In particular, G is a tree (in fact, it is its own spanning tree). $\square$

Rooted Trees

I would like to describe rooted trees recursively.

Definition.

(B) A graph T with one vertex v and no edges is a [trivial] rooted tree (T, v) with root v;

(R) If (T, r) is a rooted tree with root r, and T' is obtained by attaching a leaf to T, then (T', r) is a rooted tree with root r.

Clearly this definition gives nothing but rooted trees.

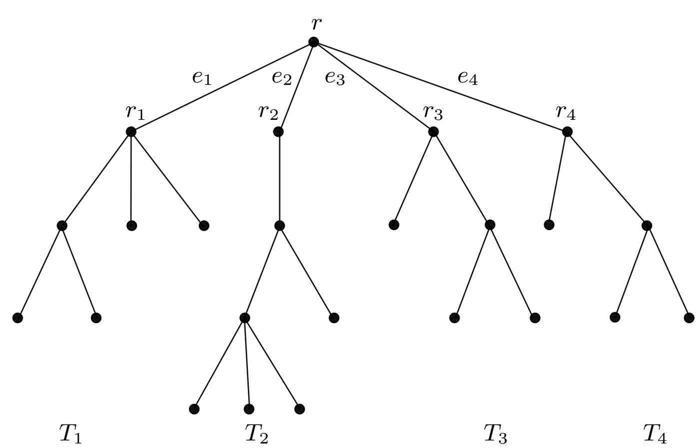
Here is another way to describe the class of rooted trees recursively. We will define a class $\mathcal{R}$. of ordered pairs (T, r) in which T is a tree and r is a vertex of T, called the root of the tree. For convenience, say that (T_1, r_1) and (T_2, r_2) are disjoint in case T_1 and T_2 have no vertices in common. If the pairs $(T_1, r_1), \ldots (T_k, r_k)$ are disjoint, then we will say that T is obtained by *hanging* $(T_1, r_1), \ldots (T_k, r_k)$ *from* r in case

(1) r is not a vertex of any T_i;

(2) $V(T) = V(T_1) \cup \cdots \cup V(T_k) \cup \{r\}$;

(3) $E(T) = E(T_1) \cup \cdots \cup E(T_k) \cup \{e_1, \ldots, e_k\}$, where the edge e_i joins r to r_i.

Here is an illustration of this definition:



Here is the definition of the class $\mathcal{R}$ (of rooted trees):

(B) If T is a graph with one vertex v and no edges, then $(T, v) \in \mathcal{R}$;

(R) If $(T_2, r_1), \ldots, (T_k, r_k)$ are disjoint members of $\mathcal{R}$ and if (T, r) is obtained by hanging $(T_2, r_1), \ldots, (T_k, r_k)$ from r, then $(T, r) \in \mathcal{R}$.

Preorder and Postorder Listings. Let (T, v) be a rooted tree, where v is a root. For each child w of v we denote by (T_w, w) the rooted subtree of (T, v) which starts with the root w. There are two important algorithms to create preodered and postordered listings, **Preorder** (T, v) and **Postorder** (T, v). Here they are:

Preorder (T, v)

    Put v to the list $L(v)$
    for each child of v, from left to right do
    Attach **Preorder** (T_w, w) to the end of the list $L(v)$
    Return $L(v)$

Here we created the list of vertices of (T, v), where all parents are listed before their children.

Postorder (T, v)

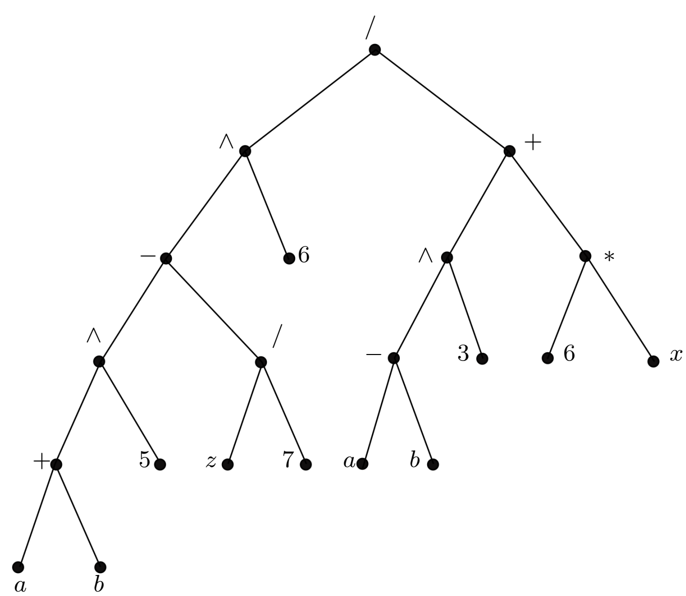
    Start with empty list $L(v)$
    for each child of v, from left to right do
    Attach **Postorder** (T_w, w) to the end of the list $L(v)$
    Put v to the end of the list $L(v)$
    Return $L(v)$

Here we created the list of vertices of (T, v), where all children listed before their parents.

We say that a rooted tree (T, v) is binary if every vertex has at most two chidren. Then we say that (T, v) is a complete binary tree if every vertex has exactly two chidren. It is easy to show (by induction) that a a complete binary tree has odd number of vertices.

Polish Notations. Now we describe an important application. Consider the formula:

$$\frac{((a+b)^5 - z/7)^6}{(a-b)^3 + 6x}$$



Here is the *preorder listing* of this graph (known as *Polish notations*):

$$/ \wedge \; - \; \wedge \; + a \; b \; 5 \; / \; z \; 7 \; 6 \; + \; \wedge \; - a \; b \; 3 \; * 6 \; x$$