Summary on Lecture 15, October 30, 2017

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 the root v;

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

Clearly this definition gives nothing but rooted trees.

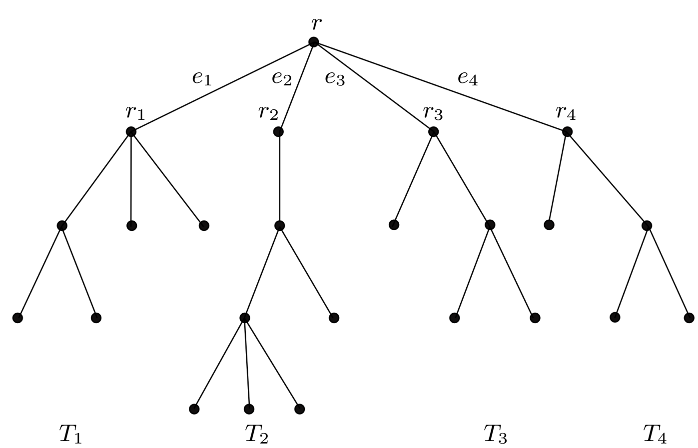
Here is another way to describe the class of rooted trees recursively. We will define a class $\mathcal{R}$ of ordered pairs (T, r) in which T is a tree and r is a vertex of T, called the root of the tree. For convenience, say that (T_1, r_1) and (T_2, r_2) are disjoint in case T_1 and T_2 have no vertices in common. If the pairs $(T_1, r_1), \dots (T_k, r_k)$ are disjoint, then we will say that T is obtained by *hanging* $(T_1, r_1), \dots (T_k, r_k)$ *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) $E(T) = E(T_1) \cup \cdots \cup E(T_k) \cup \{e_1, \dots, e_k\}$, 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 $(T_1, r_1), \dots, (T_k, r_k)$ are disjoint members of $\mathcal{R}$ and if (T, r) is obtained by hanging $(T_1, r_1), \dots, (T_k, r_k)$ 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 w 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)

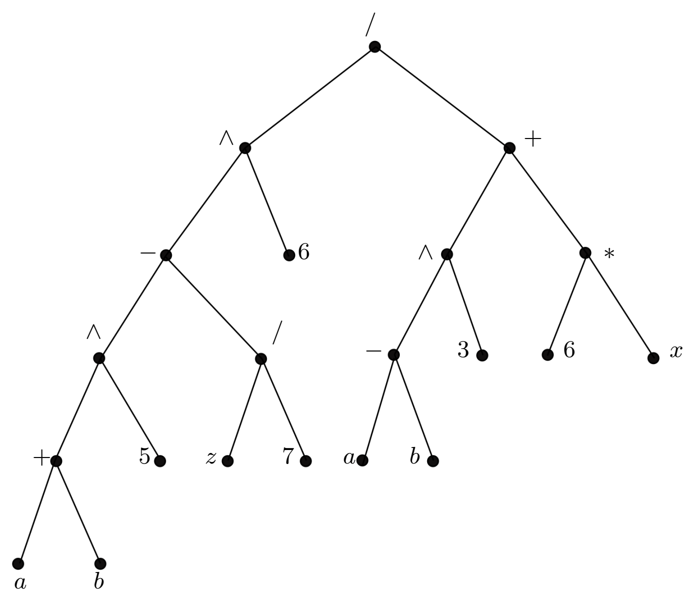
```
Start with empty list L(v)
for each child w of v, from left to right do
Attach Postorder(T_w, w) to the end of the list L(v)
Put v to the end of the list L(v)
Return L(v)
```

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*):

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