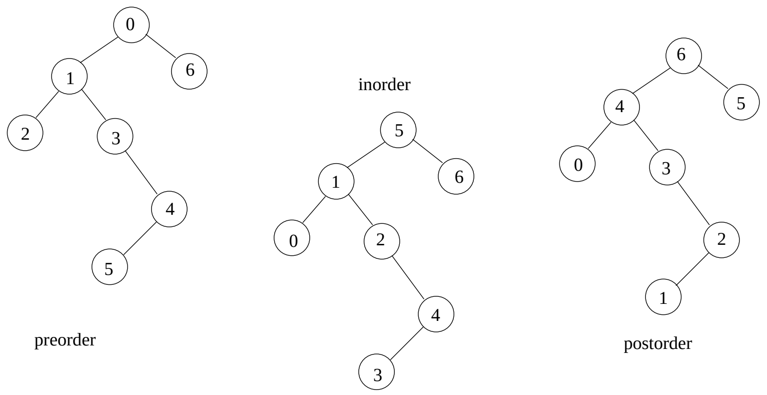
6 Preorder, inorder, and postorder

In a tree or forest, the *subtree* at a node u is the set of all descendants of u, with the same parent function except that u becomes the root.

In a binary tree, the *left* or *right* subtree at a node is the subtree at its left or right child (empty where these children are undefined).

Nowadays, trees are usually illustrated as shown below, with the root highest. They are viewed like family trees rather than real trees. In binary trees the left child is drawn to the left of the right child. If there is enough room,



There are three important ways of ranking the nodes of a binary tree. Two of them are valid for general trees. They are defined recursively. The same tree is illustrated with nodes labelled according to each of the three orders.

The recursion defines, for each node u, the ranks (pre-,in-,post-) of all descendants of u. Let S_ℓ and S_r be the left and right subtrees at u. They can be empty.

- **Preorder.** The node u precedes all nodes in S_ℓ, and they all precede all nodes in S_r.
- **Inorder.** All nodes in S_ℓ precede u and all nodes in S_r follow u.
- **Postorder.** All nodes in S_ℓ precede all nodes in S_r, and they precede u.

These definitions require a bit of assimilation but they are important. Inorder makes sense only for binary trees, but pre- and postorder can be defined for any trees.

(6.1) Lemma *Each of these three orders is a linear order on the nodes, so there are three rankings of the nodes from $0 \ldots n-1$. Also, for any node v, the ranks of the descendants of v, under any of the three orders, form a contiguous interval from lowest to highest. Therefore, if* min *and* max *are the lowest and highest ranks — under any of the three orders — then a node x is a descendant of v if and only if its rank is between* min *and* max.

Proof. Omitted. A formal proof might be difficult. ∎

(6.2) Lemma *If T is a binary tree and its nodes are presented in preorder and also in inorder, then T is uniquely defined by these two orders. Also if presented by the nodes in inorder and also postorder.*

Proof. We shall consider the second part, building the tree from the inorder and postorder ranking. We assume that the number n of nodes has been given, and an array `post2inorder[0..n-1]` so that `post2inorder[p]` is the inorder rank of the node with postorder rank p.

If we list the descendants in T of a particular node u, in inorder, we get a list LuR, and the descendants of u in postorder forms an other list $L'R'u$ where L' is a permutation of L and R' is a permutation of R. (Some of these sets may be empty).

To construct the subtree at u, we need only

- $d(u)$, the number of descendants of u.
- *imin*, the minimal inorder rank of all descendants of u.
- $p(u)$, the postorder rank of u, which is the maximal postorder rank of all descendants of u.
- `post2inorder`.

Given this, suppose that u has a left child v and right child w (they need not exist, of course).

Let m be the inorder rank of u, obtained from `post2inorder`.

- The left child v has $|L| = m - imin$ descendants.
- *imin* is also the minimal inorder rank of all descendants of v.
- The right child w has $|R| = d(w) = d(u) - |L| - 1$ descendants.
- The postorder rank of w is $p(u) - 1$.
- $m + 1$ is the minimal inorder rank of all descendants of w.
- $p(v) = p(w) - |R|$.

Assuming that the subtrees at v and w have been constructed, one constructs the subtree at u. The proof is finished with a simple inductive argument. ∎