Binary and rooted trees

1. A leaf node in a binary tree is a node whose left and right subtrees are both empty. A full node is one for which both left and right subtrees are not empty. Prove that in any non-empty binary tree $n_l = n_f + 1$, where n_l is the number of leaf nodes and n_f the number of full nodes. Therefore any binary tree with $n_l \geq 1$ leaf nodes contains at least $2n_l - 1$ nodes. Given a sequence of numbers $n_0, n_1, \ldots, n_{h-1}$ find a necessary and sufficient condition for the existence of a binary tree with exactly n_i leaf nodes at depth i, for $0 \leq i < h$. Describe an $O(n)$ time algorithm to construct one such binary tree, if the condition is satisfied.
2. Give an example of two non-isomorphic binary trees such that for all i, the number of leaf nodes at depth i is the same in both trees, and also the number of full nodes at depth i is the same in both.
3. Let T be a labeled binary tree in which the labels are integers and no two nodes have the same label. Suppose you are given the sequences obtained by preorder and inorder traversal of the tree. Show that it is possible to reconstruct the tree T in $O(n)$ time. Given two sequences S_1, S_2 of distinct integers determine in $O(n)$ time whether there exists a labeled tree T such that S_1 is the preorder and S_2 the inorder traversal of T. Can this be done if the labels are not necessarily distinct? Show that there exist two different labeled binary trees, with distinct labels, whose preorder traversals are equal and also the postorder traversals are equal. Must any two such trees be isomporhic? Is the converse true?
4. Given a fixed binary or rooted tree T with n nodes labeled 0 to $n-1$ arbitrarily, a data structure is required to answer queries `descendant(i,j)` in $O(1)$ time. The query `descendant(i,j)` returns true iff node j is a descendant of node i. You are allowed to use $O(n)$ space and $O(n)$ time for initializing the data structure, but each query must take $O(1)$ time. A more difficult problem is for queries of the kind `lca(i,j)`, where `lca(i,j)` is the least common ancestor of nodes `i,j`. The least common ancestor is a common ancestor with maximum depth or equivalently minimum height. This will be discussed in class.
5. A node j is said to be reachable from a node i in k steps if there exists a path of length k from node i to node j in a binary or rooted tree. Given a tree T (binary or rooted), describe an $O(n)$ time algorithm to find for each node the number of nodes that are reachable from it in k steps, for any specified k. The time required should be independent of k.