Warm-up

Problem 1. Let A be an array holding n distinct integer values. We say that a tree T is a *pre-order realization of A* if T holds the values in A and a pre-order traversal of T visits the values in the order they appear in A.

Design an algorithm that given an array produces a pre-order realization of it.

Problem 2. Let A be an array holding n distinct integer values. We say that a tree T is a *post-order realization of A* if T holds the values in A and a post-order traversal of T visits the values in the order they appear in A.

Design an algorithm that given an array produces a post-order realization of it.

Problem solving

Problem 3. Design a linear time algorithm that given a tree T computes for every node u in T the size of the subtree rooted at u.

Problem 4. In a binary tree there is a natural ordering of the nodes on a given level of the tree, i.e., the left-to-right order that you get when you draw the tree. Design an algorithm that given a tree T and a level k, visits the nodes in level k in this natural order. Your algorithm should perform the whole traversal in $O(n)$ time.

Problem 5. Design an algorithm that given a binary tree T and a node u, returns the node that would be visited after u in a pre-order traversal. Your algorithm should *not* compute the full traversal and then search for u in that traversal.

Problem 6. Design an algorithm that given a binary tree T and a node u, returns the node that would be visited after u in an in-order traversal. Your algorithm should *not* compute the full traversal and then search for u in that traversal.

Problem 7. Design an algorithm that given a binary tree T and a node u, returns the node that would be visited after u in a post-order traversal. Your algorithm should *not* compute the full traversal and then search for u in that traversal.

Problem 8. The balance factor of a node in a binary tree is the absolute difference in height between its left and right subtrees (if the left/right subtree is empty we consider its height to be -1). Design an algorithm for computing the balance factor of **every** node in the tree in $O(n)$ time.

Problem 9. Describe an algorithm for performing an Euler tour traversal of a binary tree that runs in linear time and **does not** use a stack or recursion.

Problem 10. For any pair of nodes in a tree there is a unique simple path (one that does not repeat vertices) connecting them. Design a linear time algorithm that finds the longest such path in a tree.

Advanced problem solving

Problem 11. Design a linear time algorithm that given a sorted sequence of n values, builds a binary search tree T holding these values such that the height of T is $\lfloor \log_2 n \rfloor$. For simplicity, you can assume that $n = 2^k - 1$ for some positive integer k.

Problem 12. Design an external memory data structure for maintaining a list that supports index-based insertions and deletions in $O(n/B)$ transfers.