5.1. Write a recursive procedure that lists the items in the binary search tree in sorted order, in $O(n)$ time.

5.2. Write a non-recursive procedure that lists the elements of the search tree in sorted order, in $O(n)$ time with $O(1)$ additional memory (nodes have pointers to parents).

5.3. Check that the given tree is a valid binary search tree. Time $O(n)$.

5.4. Prove that there is no algorithm that builds a binary search tree from a given array of n elements faster than in $O(n \log n)$ in the worst case.

5.5. For each node x, count the number $w(x)$ equal to the number of nodes in its subtree (including x itself). Time $O(n)$.

5.6. Using the calculated values of $w(x)$, learn how to find the k-th element in the tree. Time $O(H)$.

5.7. Using $w(x)$, learn how to find the number of items less than x for the given key x. Time $O(H)$.

5.8. For a given node, find the next k nodes in sorted order in $O(H + k)$ time.

5.9. Give an example of two AVL trees that store the same set of elements, but have different heights.

5.10. A binary search tree is said to be weight-balanced if, for any vertex v, $w(v) \geq \lfloor \alpha \cdot w(parent(v)) \rfloor$ is satisfied, where $w(v)$ is the number of vertices in the subtree, α is some positive constant. Prove that the height of such a tree is $O(\log n)$.