S. Schwoon / D. Suwimonteerabuth Winter Term 2008/09

Fundamental Algorithms

Exercise Sheet 6

1. In this exercise, we are interested in two algorithms that work together. The first algorithm takes an array a of length n whose elements are in the range 1 to k as the parameter. It processes a and returns something for the second algorithm in $\mathcal{O}(n+k)$ time. The second algorithm takes the output of the first algorithm together with two integers u and v as the parameters, and returns the number of elements in a that fall into the range $[u..v]$.

   Design both algorithms such that the second algorithm always takes $\mathcal{O}(1)$ time. What is your output of the first algorithm?

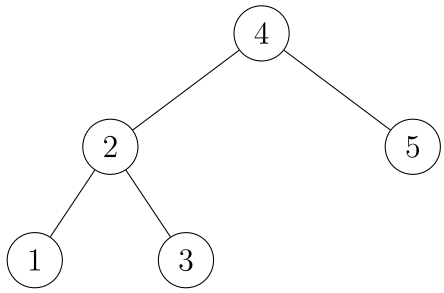
2. A *binary search tree* is a binary tree where nodes are ordered, i.e. for every node `n` of the tree all nodes reachable via `n.left` (resp. `n.right`) have smaller (resp. bigger-or-equal) values than `n.value`.

   Write down two procedures that operate on binary search trees. Analyze the complexities.

   (a) The procedure `findMin` takes a binary search tree as the parameter, and returns the node in the tree that contains the smallest value.

   (b) The procedure `remove` takes a binary search tree and an integer x as parameters, and removes a node containing the value x. The resulting tree must be a binary search tree.

3. Given a node n in a binary tree t, the height of n is the number of nodes in the longest path from n to any leaf of t. The height of a binary tree is defined as the height of its root. For instance, the heights of the nodes with values 4 and 5 in the tree below are 3 and 1, respectively. Since the node with value 4 is the root, the height of the tree is also 3.



   (a) Determine the minimum and maximum numbers of nodes in a binary tree of height h.

   (b) Extend the data type `BinNode` in the lecture to include an extra item `height`, which always indicates the height of a node. Redesign the algorithm `binary_tree_insert` in the lecture such that it always updates the `height` values correctly.

4. A binary tree is *full* if all of its nodes have either zero or two children. Let B_n be the set of full binary trees with $2n+1$ nodes, where each node has a distinct value in the range $[1, 2n+1]$. For instance, the tree in the previous example is full and belongs to the set B_2.

   (a) Determine B_0, B_1, B_2, and B_3 by drawing all possible full binary trees with 1, 3, 5, and 7 nodes, respectively.

   (b) Determine $|B_n|$ for any $n \geq 0$.