

## CSCI2100: Regular Exercise Set 9

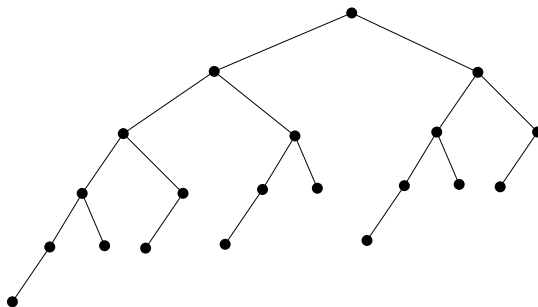
Prepared by Yufei Tao

**Problem 1.** In the class, we proved that if  $f(h)$  denotes the smallest number of nodes in a balanced binary tree of height  $h$ , it must hold that

$$f(h) = 1 + f(h - 1) + f(h - 2).$$

Give a balanced binary tree of height 6 with  $f(6)$  nodes.

**Solution.**



**Problem 2.** Let  $T$  be a binary tree of  $n$  nodes. For each node  $u$  of  $T$ , define its *count* as the number of nodes in its subtree (remember that the subtree includes the node itself). Describe an algorithm to compute the counts of all the nodes in  $T$  (you can assume that each node has reserved a memory cell for you to store the count). Your algorithm must terminate in  $O(n)$  time.

**Solution.** We will design a recursive algorithm to perform a *post order* traversal of  $T$ . This algorithm at its termination will have computed the counts of all the nodes.

If the tree has only a single node, set its count to 1 and return. Otherwise, the algorithm proceeds as follows:

- Recursively compute the counts of all the nodes in the left subtree of the root.
- Recursively compute the counts of all the nodes in the right subtree of the root.
- Let  $c_1$  be the count of the left child, and  $c_2$  be the count of the right child. Set the count of the root to  $1 + c_1 + c_2$ .

To see that the running time is  $O(n)$ , observe that the algorithm essentially crosses each edge of the tree twice: once from the parent to the child, and another time the other way around. There are only  $n - 1$  edges in the tree.

**Problem 3.** Let  $T$  be a binary search tree (BST) of on a set  $S$  of  $n$  integers. Let  $x$  and  $y$  be two integers in  $S$ . Describe an algorithm to find the lowest common ancestor  $A$  of the nodes in  $T$  storing  $x$  and  $y$ , respectively. If  $A$  is at level  $\ell$  (recall that the root is at level 0), your algorithm must finish in  $O(1 + \ell)$  time.

**Solution.** Without loss of generality, suppose that  $x < y$ . Let  $v$  be the root of  $T$ . If the key  $k$  of  $v$  equals either  $x$  or  $y$ , return  $v$ . Otherwise:

- If  $x < k < y$ , report  $v$ , and finish.

- If  $y < k$ , set  $v$  to its left child, and repeat the above steps at (the new)  $v$ .
- Otherwise, set  $v$  to its right child, and repeat the above steps at (the new)  $v$ .

**Problem 4.** Let  $T$  be a binary search tree (BST) of on a set  $S$  of  $n$  integers. Describe an  $O(\log n + k)$ -time algorithm to answer the following query: given an interval  $[a, b]$ , report all the integers of  $S$  that fall in  $[a, b]$ . Here,  $k$  is the number of integers reported.

**Solution.** First, find the successor  $a'$  of  $a$ , and the predecessor  $b'$  of  $b$ . Then, find the lowest common ancestor, denoted as node  $A$ , of the nodes  $a'$  and  $b'$ . This takes  $O(\log n)$  time in total.

Denote by  $\Pi_1$  the path from  $A$  to node  $a'$ , and  $\Pi_2$  the path from  $A$  to node  $b'$ . For every node on these two paths, report its key if the key falls in  $[a, b]$ . This takes  $O(\log n)$  time.

For every node  $u$  on  $\Pi_1$  other than  $A$ , do the following: if  $u$  is the left child of its parent  $p$ , then report all the keys in the right subtree of  $p$ . If  $k_u$  keys are reported, this step takes  $O(1 + k_u)$  time.

For every node  $u$  on  $\Pi_2$  other than  $A$ , do the following: if  $u$  is the right child of its parent  $p$ , then report all the keys in the left subtree of  $p$ . If  $k_u$  keys are reported, this step takes  $O(1 + k_u)$  time.

Overall the cost is  $O(\log n + k)$ , noticing that

$$\sum_{u \in \Pi_1 \cup \Pi_2 \setminus \{A\}} k_u \leq k.$$

**Problem 5.** Let  $S$  be a set of  $n$  key-value pairs of the form  $(t, v)$ . Denote by  $m$  the number of distinct keys in all the pairs of  $S$ . Describe a data structure to support the following queries efficiently: given an interval  $[a, b]$ , report all the pairs  $(t, v) \in S$  such that  $t \in [a, b]$ . Your structure must use  $O(n)$  space, and answer a query in  $O(\log m + k)$  time, where  $k$  is the number of pairs reported.

**Solution.** Create a BST on the  $m$  distinct keys. At each node  $u$  of the tree, use a linked list to chain up all the key-value pairs  $(t, v)$  where  $t$  equals the key of  $u$ . The query algorithm is the same as the one for Problem 4, except that for every node  $u$  whose key falls in  $[a, b]$ , we should report everything in its linked list.