Practice Problems for Midterm 1

Midterm Exam 1 will be in class on **Tue, Oct 15**. The exam will be closed-book, closed-notes, but you will be allowed one sheet of notes, front and back (handwritten or typeset, your choice). Please plan to bring your university ID with you during the exam.

Disclaimer: These practice problems have been extracted from old homework assignments and exams. Material changes from semester to semester. These do **not** reflect the actual coverage, difficulty, or length of the midterm exam.

Problem 1. Short answer questions. Except where noted, explanations are not required, but may be given for helping with partial credit.

(a) A binary tree is *full* if every node either has 0 or 2 children. Given a full binary tree with n total nodes, what is the maximum number of leaf nodes? What is the minimum number? Give your answer as a function of n (no explanation needed).

(b) What is the minimum and maximum number of levels in a 2-3 tree with n nodes. (Define the number of levels to be the height of the tree plus one.) Hint: Recall the formula for the geometric series: $\sum_{i=0}^{m-1} c^i = (c^m - 1)/(c - 1)$.

(c) You have an AVL tree containing n keys, and you insert a new key. As a function of n, what is the maximum number of rotations that might be needed as part of this operation? (A double rotation is counted as two rotations.) Explain briefly.

(d) Repeat (c) in the case of deletion. (You can give your answer as an asymptotic function of n.)

(e) Both skip lists and B-trees made use of nodes containing a variable number of elements. (In the skip list, and node has a variable number of pointers, and in a B-tree a node has a variable number of keys/children.) In one case, we allocated nodes of variable size and in the other case, we allocated nodes of the same fixed size. Why did we do things differently in these two cases?

(f) Suppose you know that a very small fraction of the keys in a data structure are to be accessed most of the time, but you do not know which these keys are. Among the data structures we have seen this semester, which would be best for this situation? Explain briefly.

(g) Unbalanced search trees and treaps both support dictionary operations in $O(\log n)$ "expected time." What difference is there (if any) in the meaning of "expected time" in these two contexts?

(h) Splay trees are known to support efficient *finger search queries*. What is a "finger search query"?

(i) Consider a splay tree containing n keys $a_1 < a_2 < \cdots < a_n$. Let x, y, and z be any three consecutive elements in this sorted sequence. Suppose that we perform `splay`(x) followed immediately by `splay`(z). What (if anything) can be said about the depth of y at this time? (Recall that the depth of a node is the number of edges between it and the root.)

Problem 2. Consider an arbitrary multi-way tree T. We showed that it can be represented as a binary tree T' by using the `firstChild-nextSibling` representation. If we think of the `firstChild` link as being the left link and the `nextSibling` link as being the right link, then traversals of T correspond to traversals of T'.

(a) A *preorder* traversal of T' is equivalent to which of the following traversals of T? Preorder, postorder, neither.

(b) An *inorder* traversal of T' is equivalent to which of the following traversals of T? Preorder, postorder, neither.

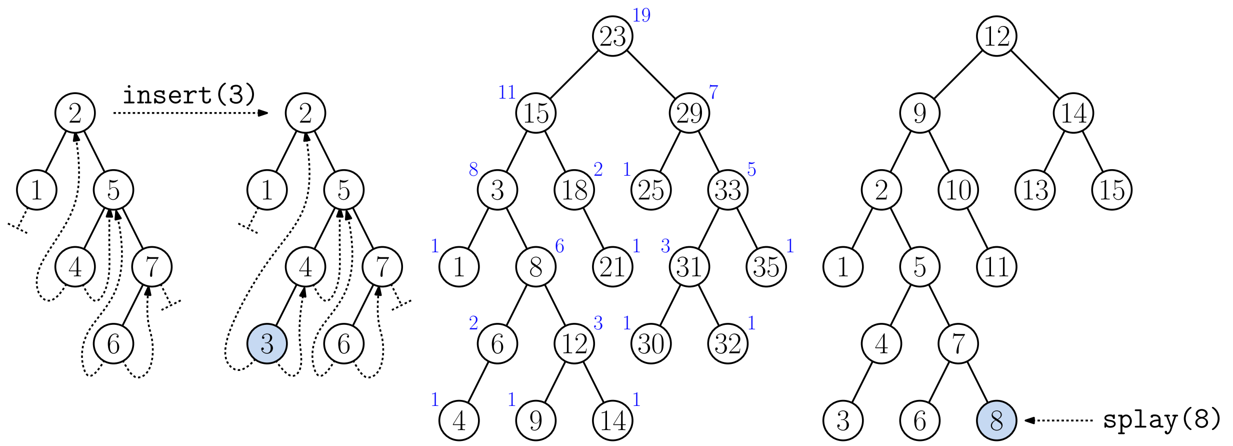
(c) A *postorder* traversal of T' is equivalent to which of the following traversals of T? Preorder, postorder, neither.

Problem 3. You a given a threaded binary search tree T (not necessarily balanced). Recall that each node has additional fields `p.leftIsThread` (resp., `p.rightIsThread`). These indicate whether `p.left` (resp., `p.right`) points to an actual child or it points to the inorder predecessor (resp., successor).

Present pseudocode for each of the following operations. Both operations should run in time proportional to the height of the tree.

(a) `void T.insert(Key x, Value v)`: Insert a new key-value pair (x, v) into T and update the node threads appropriately (see Fig. 1(a)).

(b) `BinaryNode preorderSuccessor(BinaryNode p)`: Given a non-null pointer to any node p in T, return a pointer to its *preorder successor*. (Return `null` if there is no preorder successor.)



(a) (b) (c)

Figure 1: Problems 3, 4 and 5.

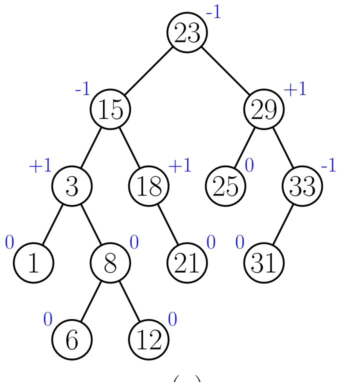
Problem 4. You are given a binary search tree T with n nodes in which each node `p` contains an additional field, `p.size`, containing the number of nodes in the subtree rooted at `p` (shown in blue in Fig. 1(a)). For each of the following queries, present pseudocode for function that answers the query in time proportional to the height of the tree.

(a) `Key T.findKth(int k)`: Given an integer k, where $1 \leq k \leq n$, returns the k-th smallest key in the search tree. (For example, for the tree in Fig. 1(b), `T.findKth(10) = 18`.)

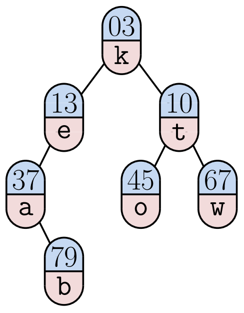
(b) `int T.range(Key x1, Key x2)`: Given two key values $x_1 \leq x_2$, returns a count the number of nodes in the tree whose key value x satisfies $x_1 \leq x \leq x_2$. (For example, in Fig. 1(b), `T.range(6,30) = 12`, as it includes $\{6, 8, 9, 12, 14, 15, 18, 21, 23, 25, 29, 30\}$.)

Problem 5. Consider the splay tree shown in Fig. 1(c). Show the result of performing the operation `splay(8)` on this tree. (You need only show the final result. Intermediate results can be shown to help with partial credit.)

Problem 6. Consider the AVL tree shown in Fig. 2(a). (The balance factors are shown in blue.) Show the result of performing the operation `delete(25)` on this tree. (You need only show the final tree, but intermediate trees can be given for assigning partial credit.)



(a)



(b)

Figure 2: Problems 6 and 7.

Problem 7. Consider the treap tree shown in Fig. 2(b). (Keys are letters and priorities are numbers.) Show the result of performing the operation `delete("k")` on this treap. (You need only show the final tree, but intermediate trees can be given for assigning partial credit.)

Problem 8. You are given a skip list with n nodes in which, rather than promoting each node to the next higher level with probability 1/2, we promote each node with probability p, for $0 < p < 1$.

(a) Given a skip list with n keys, what is the expected number of keys that contribute to the ith level. (Recall that the lowest level is level 0.) Briefly explain.

(b) Show that (excluding the header and sentinel nodes) the total number of links in such a skip list (that is, the total size of all the skip list nodes) is expected to be at most $n/(1-p)$. (Hint: It may be useful to recall the formula for the geometric series from Problem 1(b).)

Problem 9. Show that if all nodes in a splay tree are accessed (splayed) in sequential order, the resulting tree consists of a linear chain of left children.

Problem 10. You are given a degenerate binary search tree with n nodes in a left chain as shown in Fig. 3, where $n = 2^k - 1$ for some $k \geq 1$.

(a) Derive an algorithm that, using only single left- and right-rotations, converts this tree into a perfectly balanced complete binary tree (see Fig. 3). (Hint: Start by getting the root node into proper position, and then work recursively from there.)

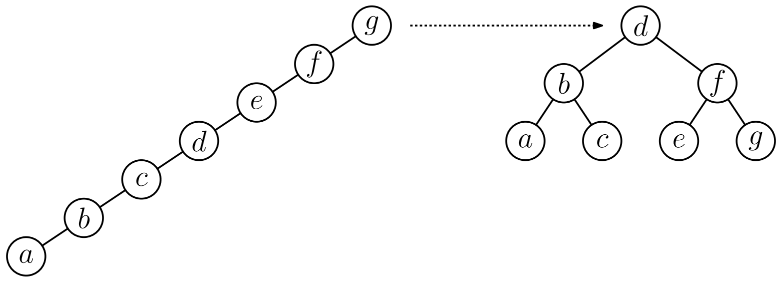


Figure 3: Problem 10.

(b) As an asymptotic function of n, how many rotations are needed to achieve this? $O(\log n)$? $O(n)$? $O(n \log n)$? $O(n^2)$? Briefly justify your answer.