

Problem 1. (8 points) Quickies.

- a. (1 point) The worst-case time to find the *second-smallest* key in a 2-3-4 tree with n keys (and no duplicate keys) is Θ (_____), if the operation that finds it is implemented in an intelligent way.

- b. (1 point) Suppose Algorithm A runs in $O(n \log n)$ worst-case time, and Algorithm B runs in $\Omega(n^2)$ worst-case time. Is it *always* true that, given the same input, Algorithm A takes less time than Algorithm B? If “yes,” explain why. If “no,” give a counterexample.

- c. (1 point) Suppose the variable `e` references an `Exception`. Suppose the method that declared this variable executes the line `System.out.println(e);`. Is this an acceptable thing to do in Java? If so, what does Java do when it executes the line? (We aren't asking what string gets printed; instead, tell us what Java does internally to print an exception.) If not, explain why it's not acceptable, and describe the error we get.

- d. (1 point) Draw a four-node tree, with numerical keys (one in each node), that could be either a valid binary heap or a valid binary search tree.

- e. (2 points) Draw the binary search tree that results if you start with an empty binary search tree and insert the following keys in the following order: 0, 8, 2, 5, 3, 7, 1; then remove 8.

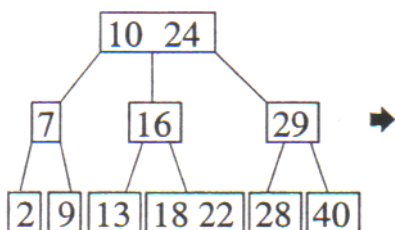
- f. (2 points) Your friend tells you that his binary search tree implementation rebalances the tree after every insert and delete operation, so that it always has the minimum possible height. This seems like a good idea, because the height of the search tree never exceeds $\log_2 n$. But it's really a terrible idea, because (unlike with a 2-3-4 tree) it cannot be done quickly. Give an argument why your friend's implementation *must* sometimes take $\Omega(n)$ time per operation. (You get to pick the tree and the sequence of insert/delete operations.)

- g. (1 **bonus point**) What is the central tenet of *Britney Theory*?

Problem 2. (7 points) **Trees.**

a. (2 points) Draw the height-2 binary tree whose postorder traversal is X R L T M S W.

b. (2 points) Draw the following 2-3-4 tree after you execute `remove(24)` on it.



c. (3 points) The input to the following `BinaryTreeNode` constructor is an array `entries` of `Entry` objects, and two array indices `first` and `last`. The constructor creates a binary tree that contains all the entries from `entries[first]` to `entries[last]` in an “inorder” traversal. The tree should be balanced as perfectly as possible. Fill in the blanks.

```

class BinaryTreeNode {
    Entry entry;
    BinaryTreeNode parent;
    BinaryTreeNode leftChild, rightChild;

    public BinaryTreeNode (Entry[] entries, int first, int last) {
        parent = null; leftChild = null; rightChild = null;
        int mid = _____;
        entry = entries[mid];
        if (mid > first) {
            leftChild = new BinaryTreeNode (entries, _____, _____);
            leftChild.parent = _____;
        }
        if (mid < last) {
            rightChild = new BinaryTreeNode (entries, _____, _____);
            rightChild.parent = _____;
        }
    }
}
    
```

Problem 3. (4 points) **Asymptotic Analysis.**

Prove formally and rigorously, omitting no details, that $\max\{x, y\} \in \Theta(x + y)$. (Both x and y are positive variables that can grow arbitrarily large.)

