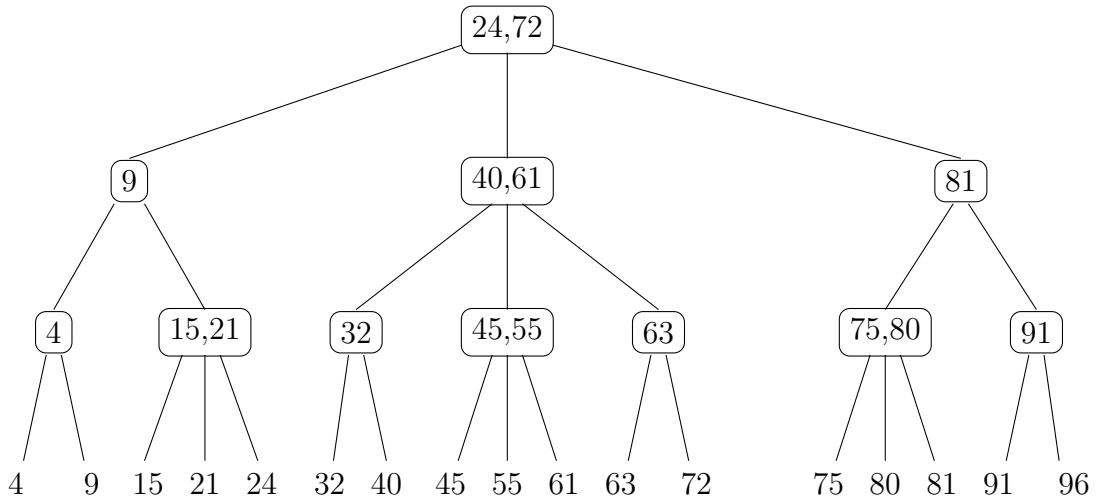
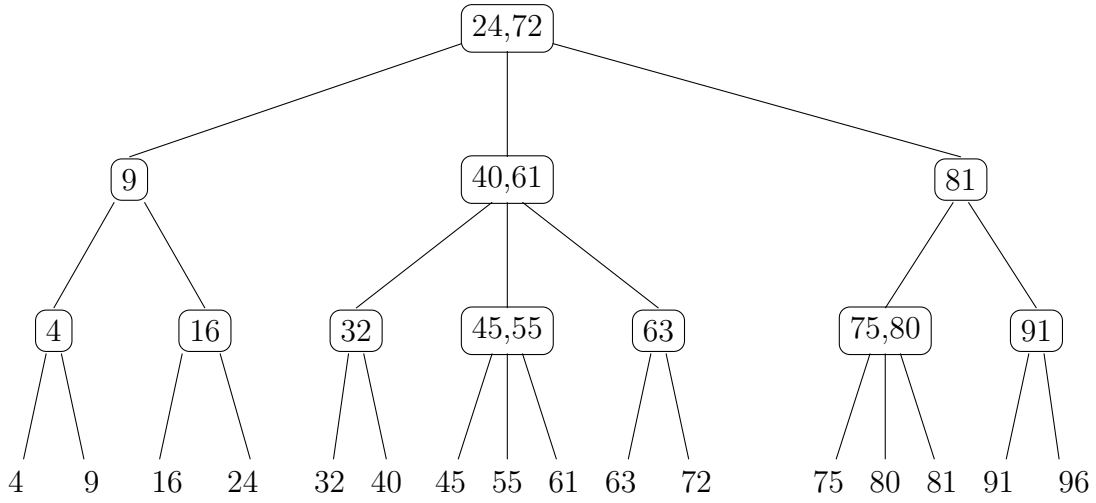
Fundamental Algorithms, Section 003
Homework 5, Fall 22. Due date: Wednesday, October 12, 4:55pm on Gradescope

1. a. Perform the operation Insert(58) on the following 2–3 tree. Show each step of the update.



b. Perform the operation Delete(16) on the following 2–3 tree. Show each step of the update.



2. Give an $O(n)$ time algorithm to output the items in a 2–3 tree in sorted order.

3. a. Clearly characterize those 2-3 trees which will increase their height when an item with value v is inserted; i.e. provide a specification such that those trees satisfying the condition increase their height when v is inserted, but every tree not meeting the condition keeps the same height.

b. Clearly characterize those 2-3 trees which will decrease their height when an item with value v is deleted; i.e. provide a specification such that those trees satisfying the condition decrease their height when v is deleted, but every tree not meeting the condition keeps the same height.

Justify your answers.

4. Consider maintaining a database for a biologist storing results concerning pea plant experiments. Each experiment records a distinct (integer) id and a plant height. We would like to support the following query:

How many plants have heights in the range $[h_1, h_2]$?

You may assume the heights are unique.

Also insertions and deletions of experiments will be allowed, where the experiments to be deleted are identified by their ids. These queries have the form Insert(id, ht) and Delete(id), respectively.

Show how to support these operations in time $O(\log n)$, where n is the database size.

You may state and use results already shown in class.

Hint. Suppose the Delete queries took the form Delete(ht) instead of Delete(id). How would you solve the problem?
For the actual problem, you need to perform searches on both id and height. How can this be done?

Challenge problem. Do not submit.
Consider maintaining counts of the number of items in each subtree of a 2–3 tree. We will keep $n - 1$ counts for a tree storing n items. Each non-leaf node will keep counts for all but its rightmost subtree (i.e. one or two counts according as it has two or three children). Show how to maintain these counts as nodes are changed due to insertions and deletions of items (the challenging part is to handle splits, joins, and transfers of siblings). It may be convenient to also keep a count of the total number of items in the tree.