234-Trees

In a perfect binary tree, all leaves are on the same level . . .

. . . but this is only possible for $n = 2^h - 1$.

Idea: If we allow different number of children, then we can always have all leaves on the same level.

234-Trees

- A node contains one, two, or three elements.
- If an internal node contains k elements $e_1, \dots, e_k$, then it has $k+1$ subtrees $T_0, \dots, T_k$.
- Order condition: Keys in subtree T_i are larger than key of e_i and smaller than key of e_{i+1}.
- All leaves have the same depth.

A 234-Tree of height h has how many nodes?

And so a 234-Tree storing n elements has height at most $\log n + 1$.

Searching in a 234-Tree is more complicated than in a binary search tree, and needs one or two comparisons per node.

Bottom-Up Insertions

First, find the leaf node v where the new key belongs.

Add new element in node v.

If node v now has at most three elements, we are done.

Otherwise, v now has four elements—an overflow. Split v into two nodes v_1 and v_2 with two elements each.

But now the parent z of v needs to have a new element to allow one more subtree. Move first element of v_2 into z.

Now z may overflow. Recursively split z and proceed up the tree.

Example: Insert 4, 6, 12, 15, 3, 5, 10, 8, 11, 17.

Top-Down Insertions

Avoid recursive walk back up the tree.

While searching for the right insertion place, split nodes that have three elements.

So when we reach the leaf node, we can simply insert the element (no overflow can happen).

This can easily be implemented as an iterative procedure.

Example: Insert 4, 6, 12, 15, 3, 5, 10, 8, 11, 17.

Top-Down Deletions

If the element to be deleted is not in a leaf, delete the next higher element instead (it is always in a leaf).

While going down the tree, modify all nodes v that contain only one element (except for the root).
Finally, delete element to be removed from the leaf.

- First try to steal an element from a sibling of v. The node v becomes a 2-element node.
- If siblings have only one element, perform a fusion of the elements from the sibling and a key from the parent. The node v becomes a 3-element node.
- During a fusion operation, the height of the tree can decrease by one.

Example: Delete 3, 4, 12, 13, 14.

Red-black trees

Since binary search trees are easier to search than 234-trees, we can implement our 234-tree as a binary search tree, using just one extra bit for each node.

- Each node is either red or black,
- the root is black,
- the children of a red node are black,
- any path from the root to an empty subtree contains the same number of black nodes.

Searching a red-black tree is completely the same as searching a binary search tree.

Insertions and deletions can be described by rotations and color changes.