2-3 Trees

A ***2-3 tree*** is a tree that can have nodes of two kinds: 2-nodes and 3-nodes.
A ***2-node*** contains a single key *K* and has two children, just like a classic BST.
A ***3-node*** contains two ordered keys *K*1 and *K*2 (*K*1<*K*2*)* and has three children.
< *K*1
(*K*1, *K*2)
> *K*2
All its leaves must be on the same level.
In other words, a 2-3 tree is always perfectly height-balanced: the length of a path from the root to a leaf is the same for every leaf.

Insertion
We always insert a new key *K* in a leaf, except for the empty tree.
The appropriate leaf is found by performing a search for *K*.

- If the leaf is a 2-node, we insert *K* there as either the first or the second key, depending on whether *K* is smaller or larger than the node's old key.
- If the leaf is a 3-node, we split the leaf in two: the smallest of the three keys (two old ones and the new key) is put in the first leaf, the largest key is put in the second leaf, and the middle key is promoted to the old leaf's parent.
- (If the leaf happens to be the tree's root, a new root is created to accept the middle key.)
- Note that promotion of a middle key to its parent can cause the parent's overflow (if it was a 3-node) and hence can lead to several node splits along the chain of the leaf's ancestors.