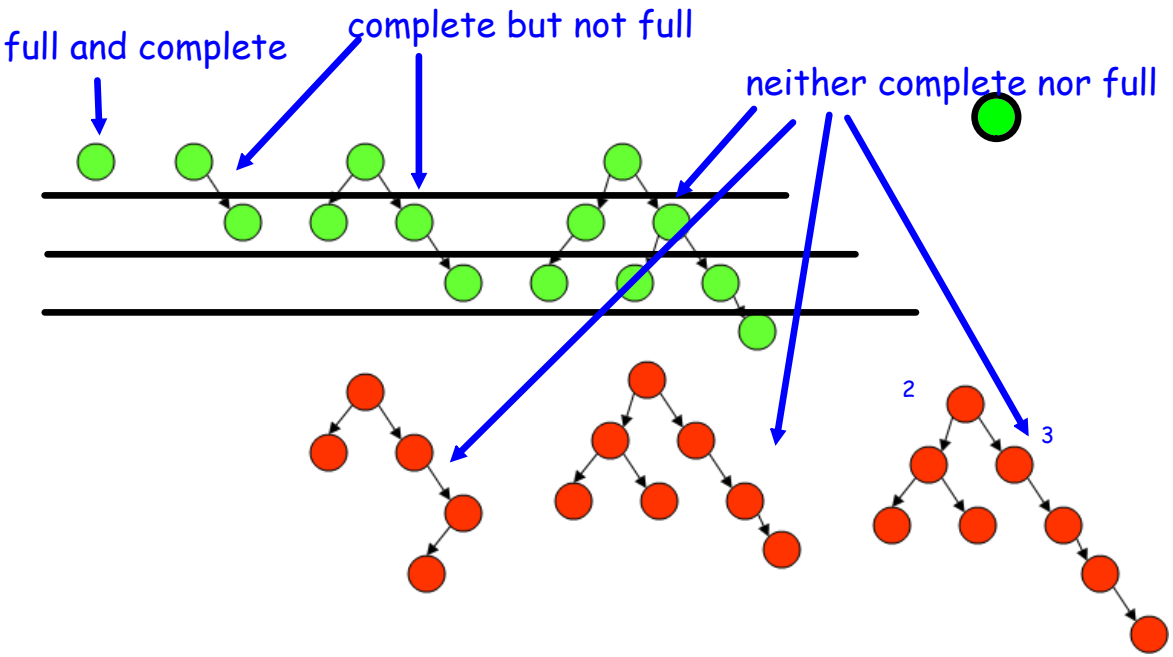
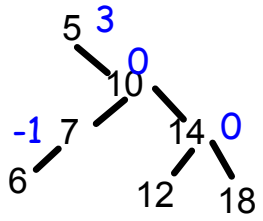
Definitions

- balance ("Bilanz") difference between the heights of the right and left sub-tree
- Height balanced tree ("höhenbilanzierter Baum"): tree where the balances are just -1, 0 or 1
- "Complete" ("vollständig) tree: tree with maxiumum number of elements on each niveau with exception of the lowest level
- "Full" ("voll) tree: tree with maximum number of elements





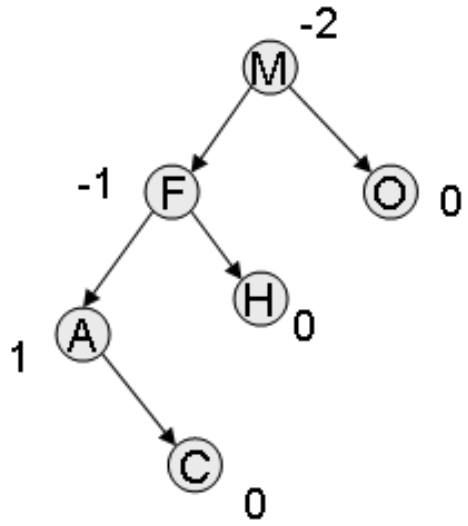
AVL trees

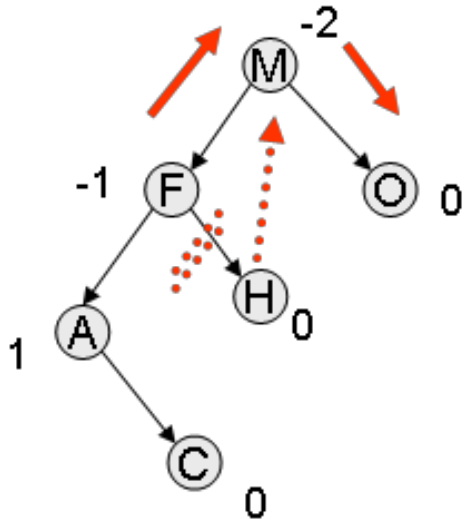
Adelson, Veliskii and Landis

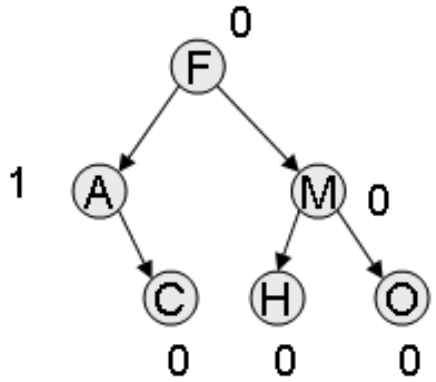
AVL: trees that are height balanced search trees

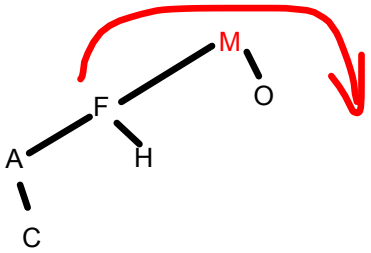
Recipe (algorithm) how to insert new elements into an AVL tree

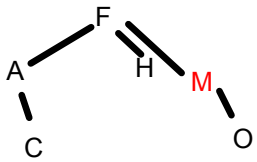
1. insert as usual
2. immediately check for height balance
3. if tree is no longer height balance (B < -1 or B > 1), immediately restructure the tree
    1. Left rotation if B > 1
    2. Right rotation if B < -1
    3. Right-left rotation if B > 1 and B (of sucessor node) is -1
    4. Left-right rotation if B < -1 and B (of sucessor node) is +1

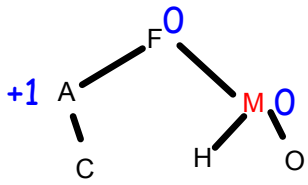












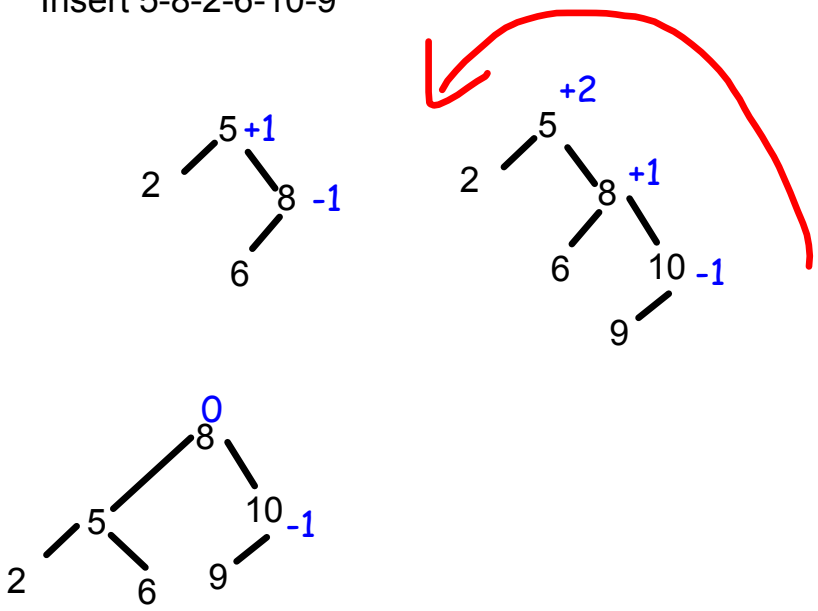
Insert 5-8-2-6-10-9

Recipe (algorithm) how to insert new elements into an AVL tree

1. insert as usual
2. immediately check for height balance
3. if tree is no longer height balance (B < -1 or B > 1), immediately restructure the tree
    1. Left rotation if B > 1
    2. Right rotation if B < -1
    3. Right-left rotation if B > 1 and B (of sucessor node) is -1
    4. Left-right rotation if B < -1 and B (of sucessor node) is +1

a possible right/left node (here H) will become the left/right node of the rotated node (M)

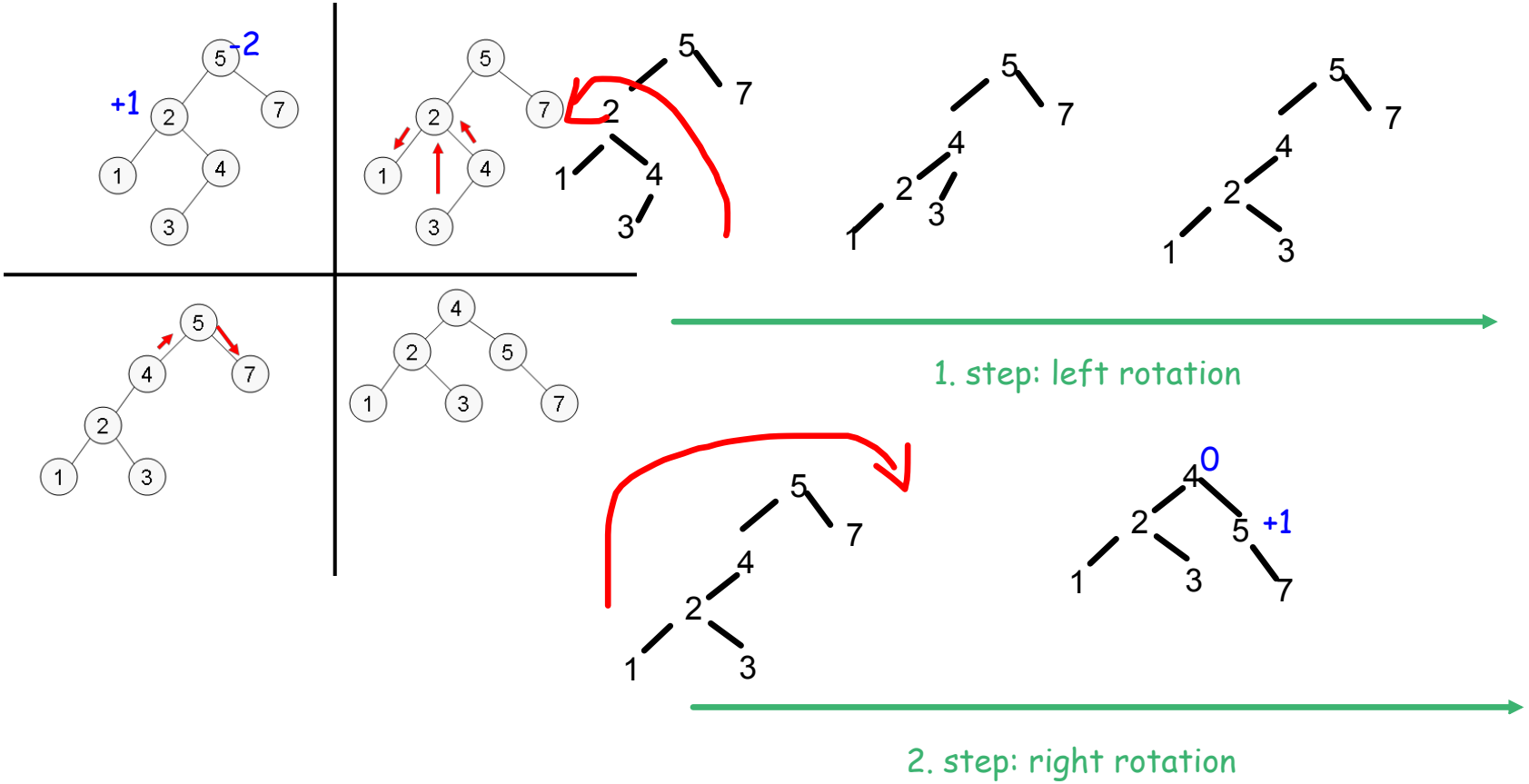
Insert 5-8-2-6-10-9



Recipe (algorithm) how to insert new elements into an AVL tree

1. insert as usual
2. immediately check for height balance
3. if tree is no longer height balance (B < -1 or B > 1), immediately restructure the tree
    1. Left rotation if B > 1
    2. Right rotation if B < -1
    3. Right-left rotation if B > 1 and B (of sucessor node) is -1
    4. Left-right rotation if B < -1 and B (of sucessor node) is +1

a possible right/left node (here H) will become the left/right node of the rotated node (5)



Recipe (algorithm) how to insert new elements into an AVL tree

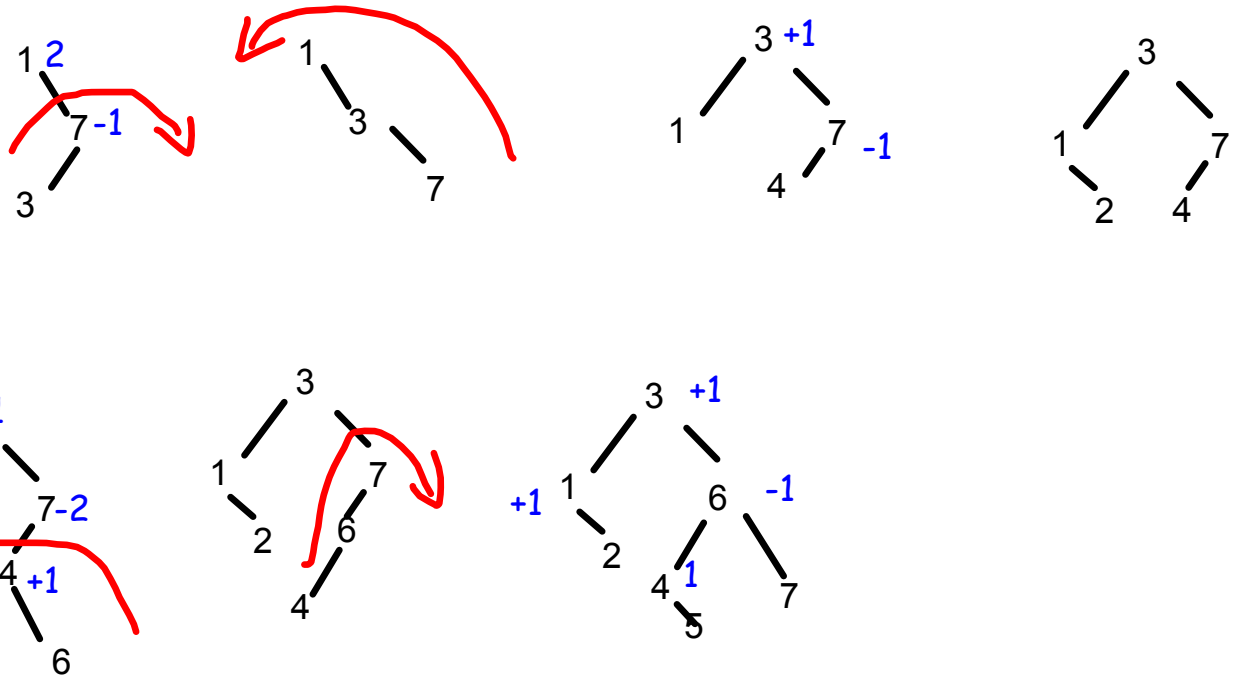
1. insert as usual
2. immediately check for height balance
3. if tree is no longer height balance (B < -1 or B > 1), immediately restructure the tree
   1. Left rotation if B > 1
   2. Right rotation if B < -1
   3. Right-left rotation if B > 1 and B (of sucessor node) is -1
   4. Left-right rotation if B < -1 and B (of sucessor node) is +1

1, 7, 3, 4, 2, 6, 5

Recipe (algorithm) how to insert new elements into an AVL tree

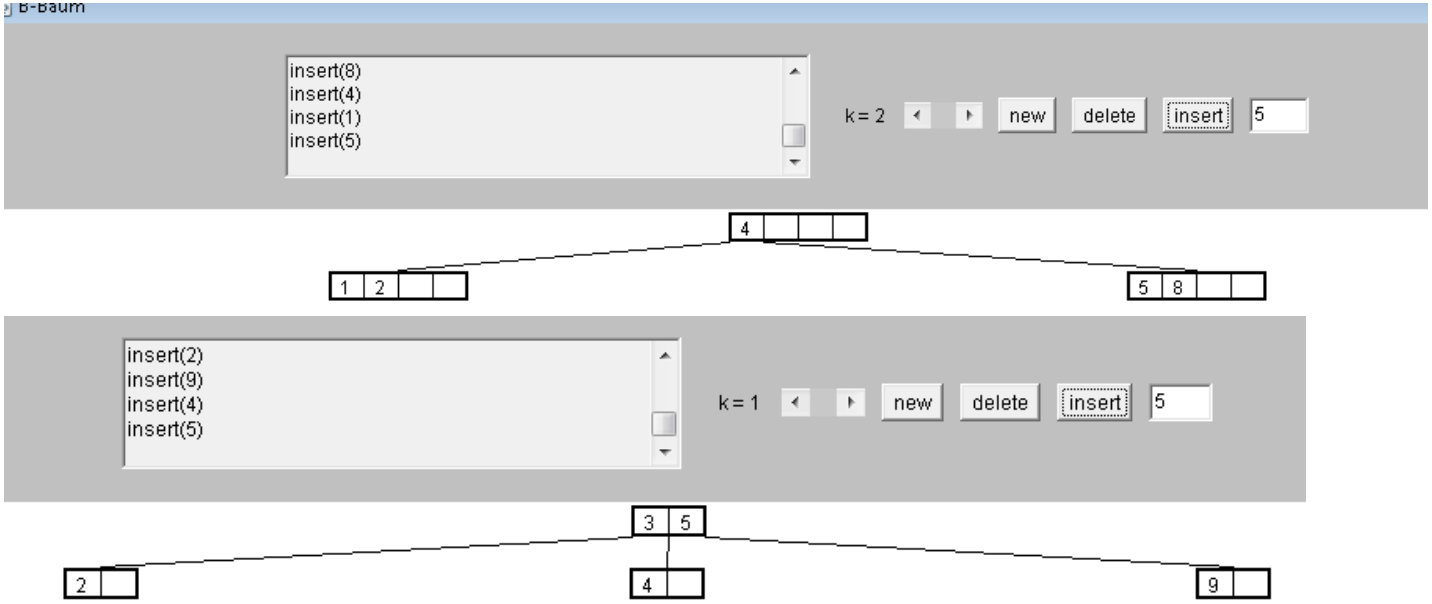
1. insert as usual
2. immediately check for height balance
3. if tree is no longer height balance (B < -1 or B > 1), immediately restructure the tree
    1. Left rotation if B > 1
    2. Right rotation if B < -1
    3. Right-left rotation if B > 1 and B (of sucessor node) is -1
    4. Left-right rotation if B < -1 and B (of sucessor node) is +1

1, 7, 3, 4, 2, 6, 5



B-Trees (B-Bäume)

B does not stand for binary but for its "inventor" Rudolph Bayer



Main characteristics

1. each node (with exception of the root node) contains between k and 2k values.
2. each node (with exception of the root node) has i + 1 children, where i is the number of elements inside the node
3. the tree is sorted
    1. all elements in the right/left sub-tree are bigger/smaller
    2. the elements within a node are sorted as well
4. any path from the root to any leaf has the same length

B-trees and databases

In databases B-trees are used to implement indices to quickly search inside databases.

The nodes of B-tree represent in a database the "pages". Pages are "storage areas" on the hard disk.

The access to these "storage areas" is much slower than the access to memory.

The index / tree structure is kept in fast memory in ordert o quickly navigate to the page in order to search or to insert values into this page. Only then the time consuming access on hard disk happens.