Datastructuren

Data Structures

Hendrik Jan Hoogeboom

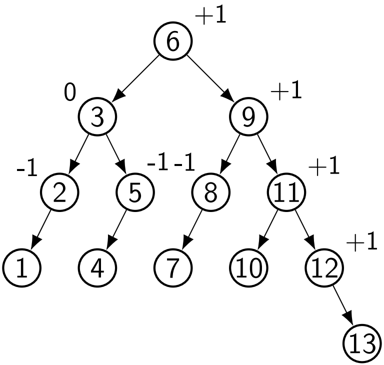
Informatica – LIACS
Universiteit Leiden

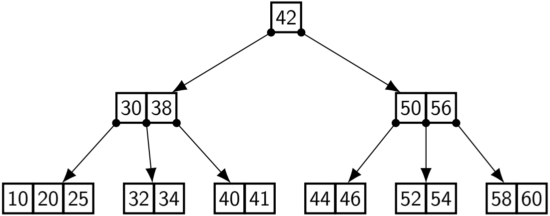
najaar 2023

Table of Contents I

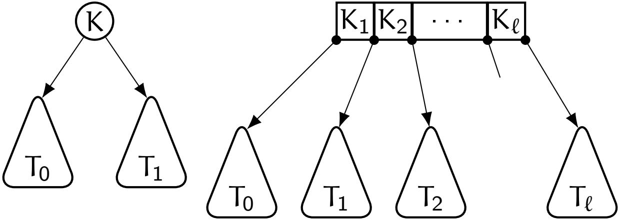
Contents

balanced trees: AVL-tree, B-tree





multiway search tree



$$T_0 < K_1 < T_1 < \cdots < K_\ell < T_\ell$$

Contents

B-tree (Bayer & McCreight, 1972)

Definition

A *B-tree of order* m is a multi-way search tree such that

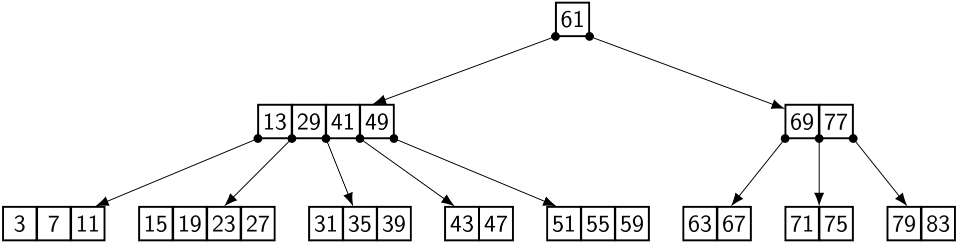
- every node has at most m children
  (contains at most $m - 1$ keys),
- every node (other than the root) has at least $\lceil \frac{m}{2} \rceil$ children
  (contains at least $\lceil \frac{m}{2} \rceil - 1$ keys),
- the root contains at least one key, and
- all leaves are on the same level of the tree.

B-tree of order 5

$m = 5$

$\lceil \frac{5}{2} \rceil = 3 \leqslant \text{children} \leqslant 5$

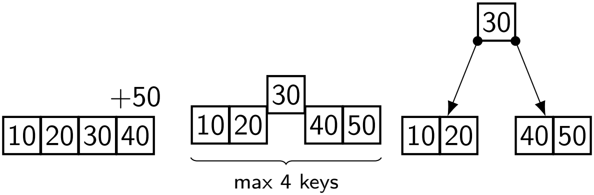
$2 \leqslant \text{keys} \leqslant 4$



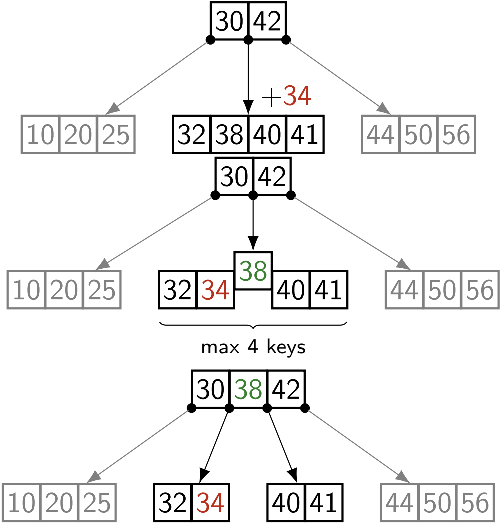
adding keys

- Add the new key to a leaf.
- When over maximal capacity, split leaf, move middle key up.
  Recurse.
- Splits can reach the root.
  We then obtain a new root with a single key.

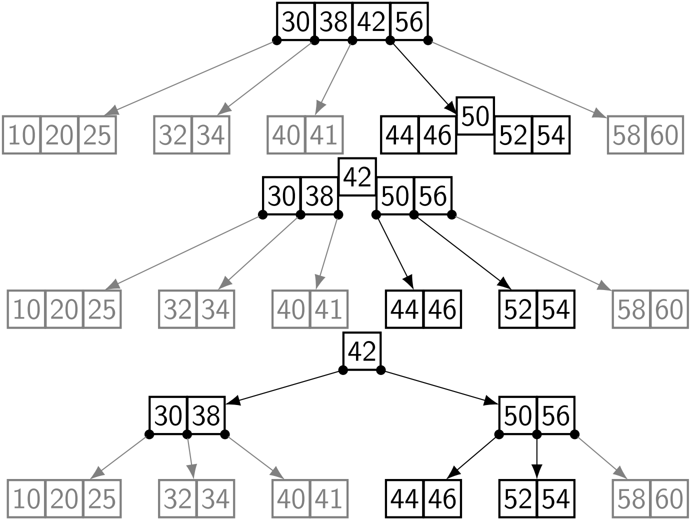
adding keys



adding keys


30 42
+34
10 20 25
32 38 40 41
44 50 56
30 42
10 20 25
38
32 34
40 41
44 50 56
max 4 keys
30 38 42
10 20 25
32 34
40 41
44 50 56

adding keys

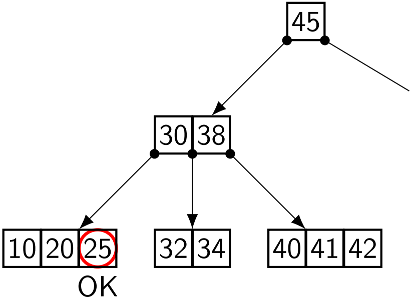

30 38 42 56
10 20 25
32 34
40 41
44 46
50
52 54
58 60
42
30 38
50 56
10 20 25
32 34
40 41
44 46
52 54
58 60
42
30 38
50 56
10 20 25
32 34
40 41
44 46
52 54
58 60

Contents

deleting keys

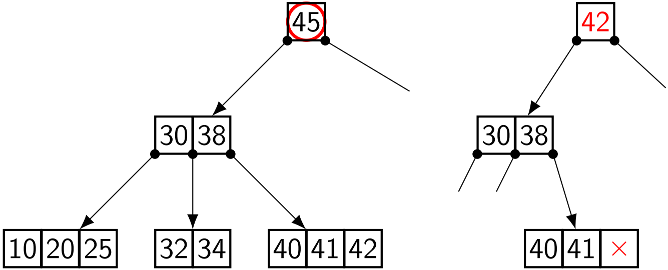
- For non-leaves: swap key with predecessor (key moves to a leaf)
- If below minimal capacity, get key from sibling with surplus, *via* parent.
- If no siblings with surplus: merge with sibling and get separating key from parent. Recurse with parent.
- Due to recursion, deletion may reach the root, and can collapse a level.

deleting keys (order 5)

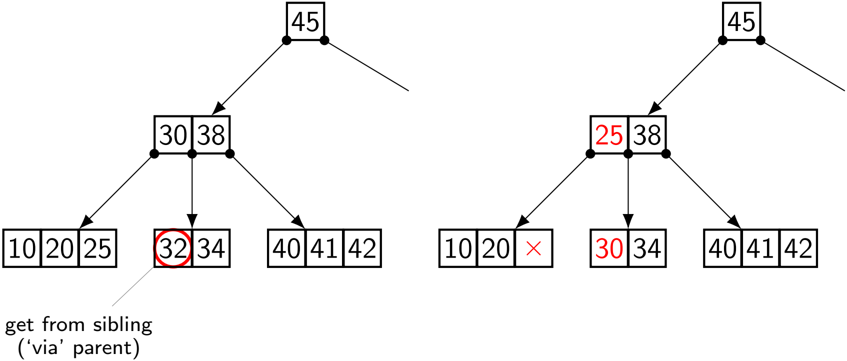


deleting keys (order 5)

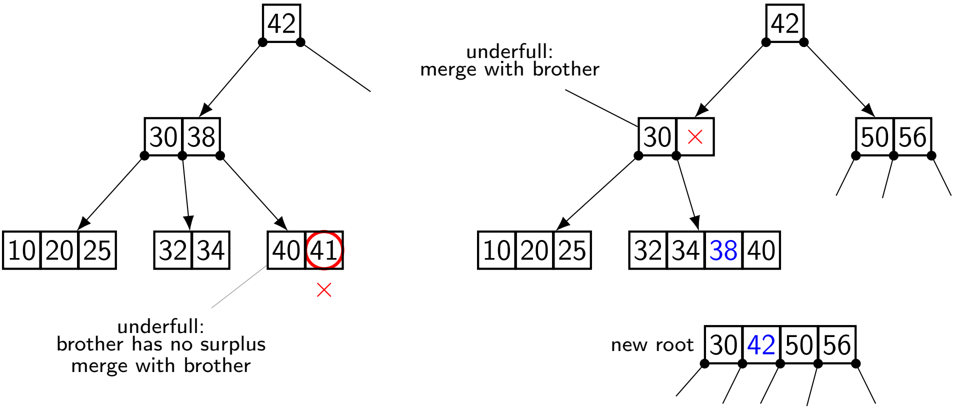
swap predecessor



deleting keys (order 5)

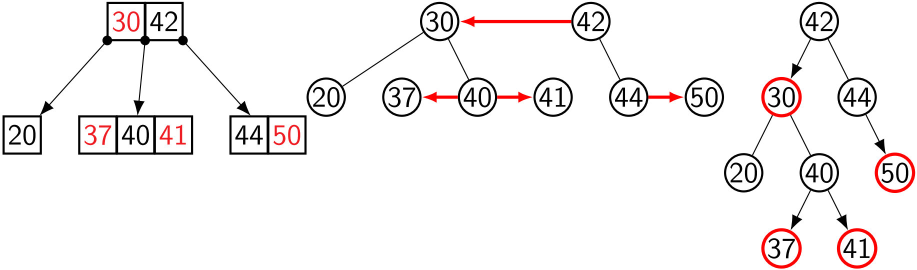


deleting, ctd (order 5)

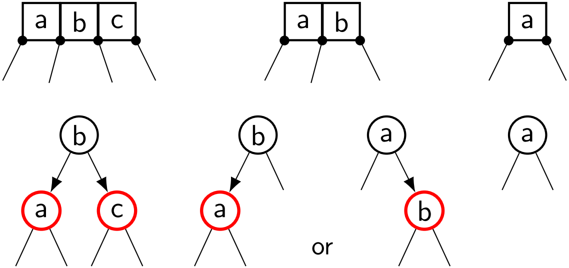


Contents

2-4-tree to red-black tree



correspondence



Definition

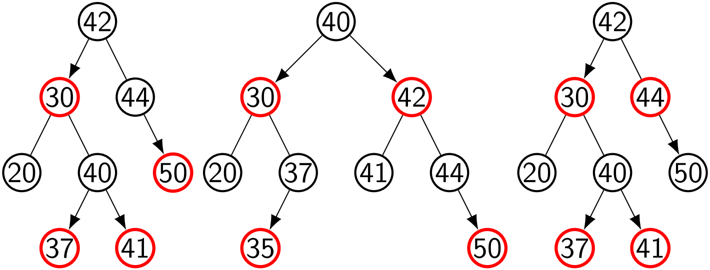
A *red-black tree* is a

- binary search tree

such that each node is either black or red, where

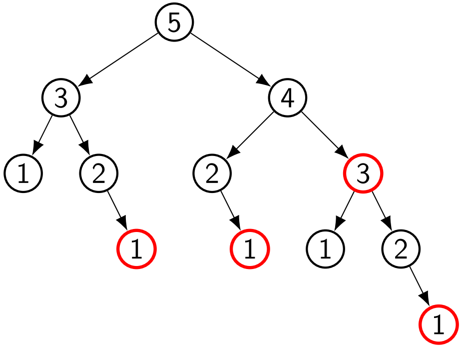
- the root is black,
- no red node is the child of another red node,
- the number of black nodes on each path from root to extended leaf (NIL-pointers) is the same.

examples



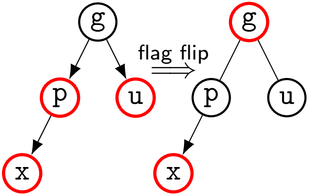
not this one

fun fact ☒

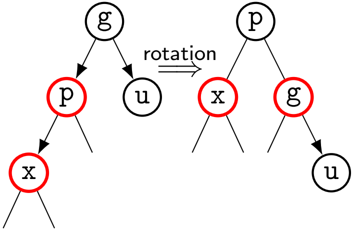


every AVL-tree can be red-black coloured

restructuring red-red nodes



red uncle

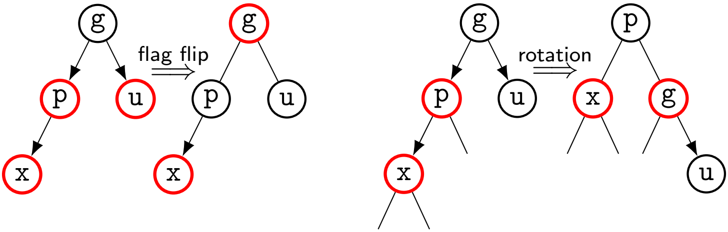


black uncle

insertion in red-black tree

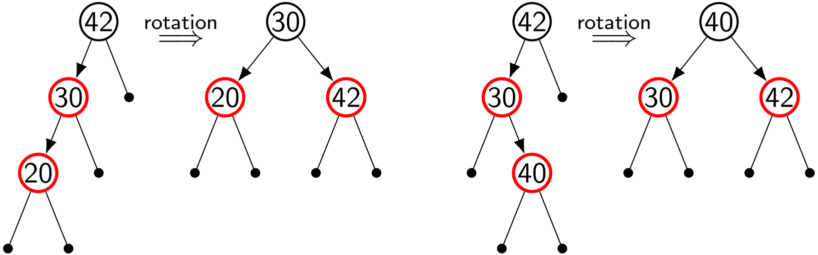
Insert as red leaf. problem: red node with red parent, then:

- if uncle is red: *flag-flip*. continue at grandparent.
- if uncle is black: *rotate* (see AVL-trees), repaint and stop.

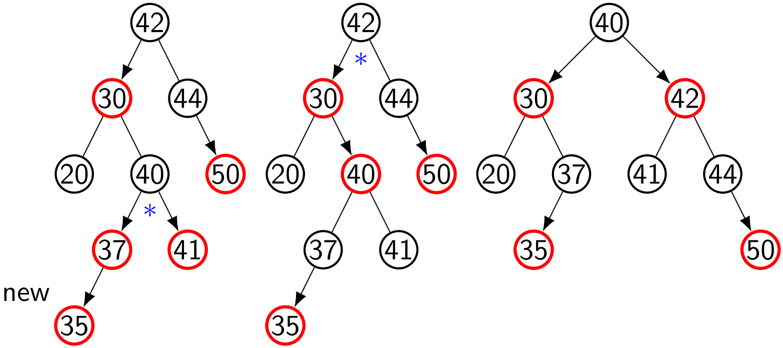


if the root has been coloured red, make it black.

just classical single/double rotation



adding 35


42
30
44
20
40
50
*
37
41
new
35
42
*
30
44
20
40
50
37
41
35
40
30
42
20
37
41
44
35
50

red black trees are popular

GNU C++ stl_tree.h

"Red-black tree class, designed for use in implementing STL associative containers (set, multiset, map, and multimap). The insertion and deletion algorithms are based on those in Cormen, Leiserson, and Rivest, Introduction to Algorithms (MIT Press, 1990), except that ..."

Linux kernel

"There are a number of red-black trees in use in the kernel. The anticipatory, deadline, and CFQ I/O schedulers all employ rbtrees to track requests; the packet CD/DVD driver does the same. The high-resolution timer code uses an rbtree to organize outstanding timer requests. The ext3 filesystem tracks directory entries in a red-black tree. Virtual memory areas (VMAs) are tracked with red-black trees, as are epoll file descriptors, cryptographic keys, and network packets in the 'hierarchical token bucket' scheduler." lwn.net/Articles/184495/

end.