Red-Black Trees and AA Trees

Binary Tree Representation Of 2-3-4 Trees

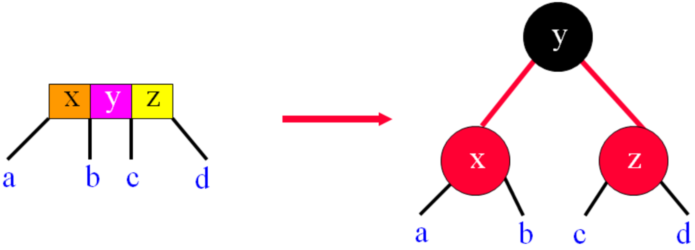
- Problems with 2-3-4 trees.

LC	P1	LMC	P2	RMC	P3	RC

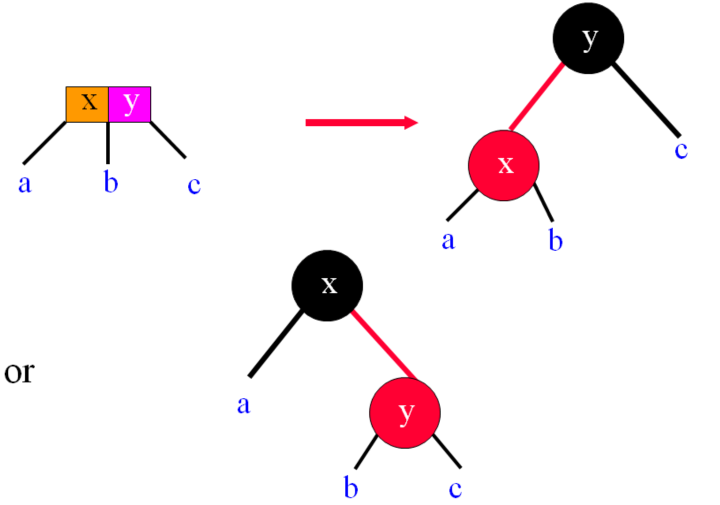
2-3-4 node structure

- 2- and 3-nodes waste space.
- Overhead of moving pairs and pointers when changing among 2-, 3-, and 4-node use.
- Represented as a binary tree for improved space and time performance.

Representation of a 4-node

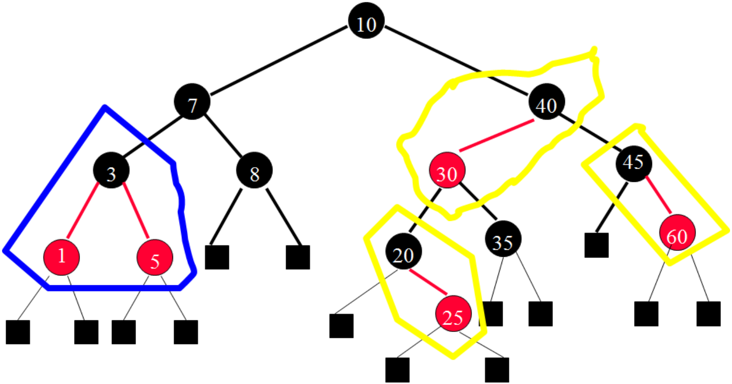


Representation of a 3-node



Representation of a 2-node

An Example



Properties of Binary Tree Representation

- Nodes and edges are colored.
  - The root is **black**.
  - Nonroot black node has a black edge from its parent.
  - Red node has a red edge from its parent.
- Can deduce edge color from node color and vice versa.
- Need to keep either edge or node colors, not both.

Red Black Trees

Colored Nodes Definition

- Binary search tree.
- Each node is colored red or **black**.
- Root and all external nodes are black.
- No root-to-external-node path has two consecutive red nodes.
- All root-to-external-node paths have the same number of **black** nodes
- The height of a red black tree that has n (internal) nodes is between $\log_2(n+1)$ and $2log_2(n+1)$.
- C++ STL implementation
- java.util.TreeMap => red black tree

Red Black Trees

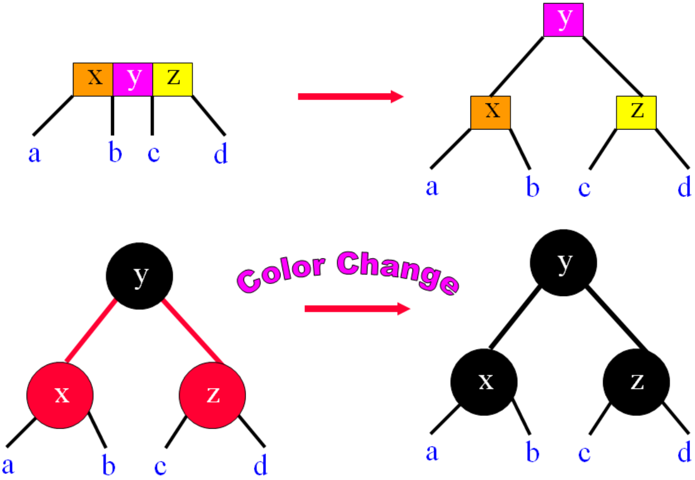
Colored Edges Definition

- Binary search tree.
- Child pointers are colored red or **black**.
- Pointer to an external node is black.
- No root to external node path has two consecutive red pointers.
- Every root to external node path has the same number of **black** pointers.

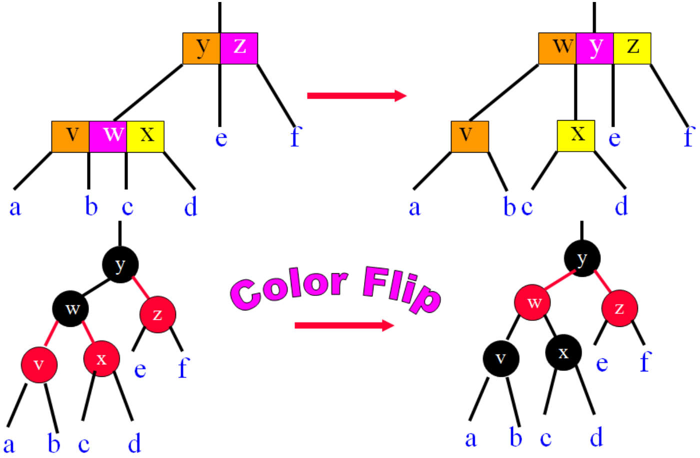
Top-Down Insert

- Mimic 2-3-4 top-down algorithm.
- Split 4-nodes on the way down.

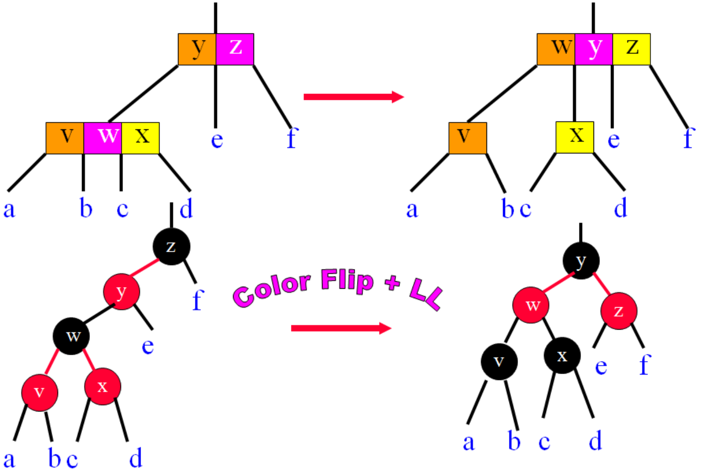
Root Is a 4-node



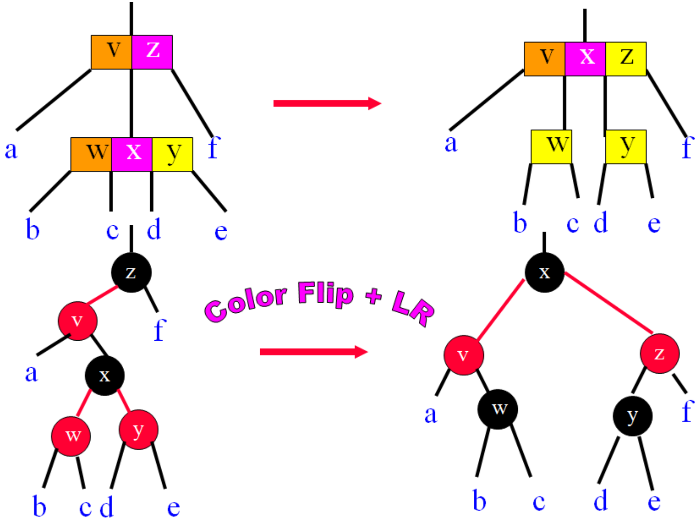
4-node Left Child of 3-node



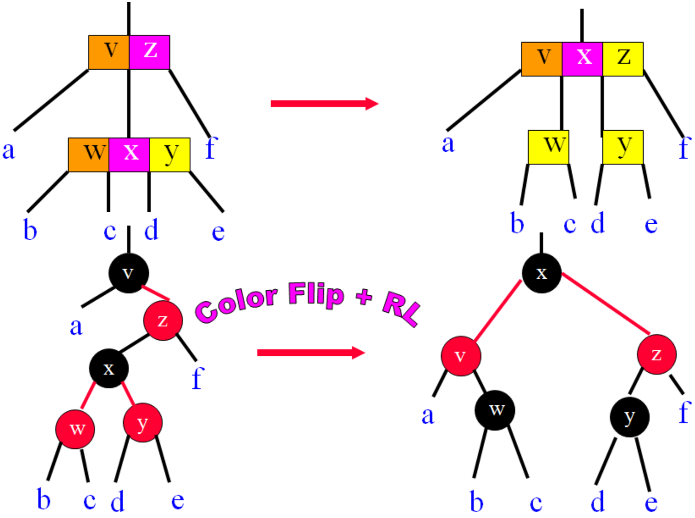
4-node Left Child of 3-node



4-node Middle Child of 3-node



4-node Middle Child of 3-node



4-node Right Child Of 3-node

- One orientation of 3-node requires color flip.
- Other orientation requires RR rotation.

AA Trees

- An *AA tree* satisfies the properties of Red-Black trees plus one more:
  - Every node is colored either red or black
  - The root is black
  - If a node is red, both of its children are black.
  - Every path from a node to a null reference has the same number of black nodes
  - Left children may NOT be red
- Invented by A. Andersson in 1993.

Advantage of AA Trees

- AA trees simplify the algorithms
  - It eliminates half the restructuring cases
  - It simplifies deletion by removing an annoying case
    - if an internal node has only one child, that child must be a red right child
    - We can always replace a node with the smallest child in the right subtree (it will either be a leaf or have a red child)

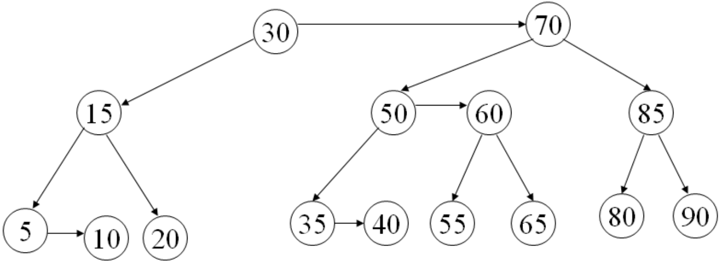
Representing the Balance information

- In each node we store a *level*. The level is defined by these rules
  - If a node is a leaf, its level is 1
  - If a node is red, its level is the level of its parent
  - If a node is black, its level is one less than the level of its parent
- The *level* is the number of left links to a null reference.

Links in an AA tree

- A horizontal link is a connection between a node and a child with equal levels
  - Horizontal links are right references
  - There cannot be two consecutive horizontal links
  - Nodes at level 2 or higher must have two children
  - If a node has no right horizontal link, its two children are at the same level

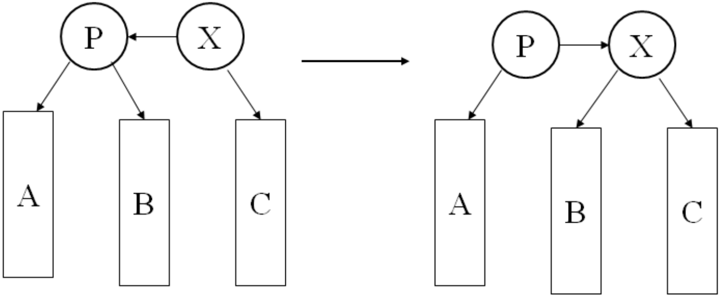
Example of an AA Tree



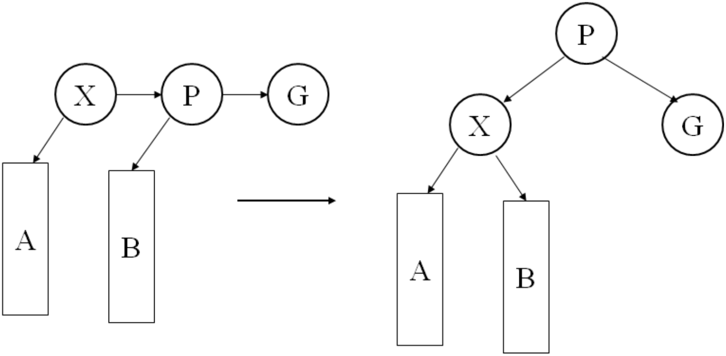
Insertion

- A new item is always inserted at the bottom level
- In the previous example, inserting 2 will create a horizontal left link
- In the previous example, inserting 45 generates consecutive right links
- After inserting at the bottom level, we may need to perform rotations to restore the horizontal link properties

skew - remove left horizontal links



split - remove consecutive horizontal links



skew/split

- A *skew* removes a left horizontal link
- A *skew* might create consecutive right horizontal links
- We should first process a *skew* and then a *split*, if necessary
- After a *split*, the middle node increases a level, which may create a problem for the original parent

An Example

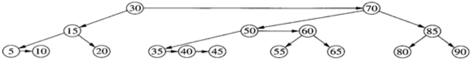


Figure 12.31 After inserting 45 into sample tree

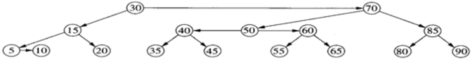


Figure 12.32 After split at 35

Figure 12.33 After skew at 50

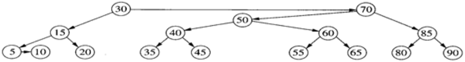


Figure 12.34 After split at 40

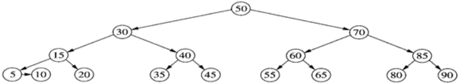


Figure 12.35 Final tree after skew at 70 and split at 30