Tutorial #10. AVL trees

Theoretical part

Properties

- Red-black trees are balanced trees, but balance property is different – path from root to any leaf should include the **same number of black nodes**. This property itself means nothing, but according to other rules longest path will never be more than twice longer, than the shortest. These rules are:
    - The root is black.
    - All leaves (NIL) are black. Usually we just consider null pointers as black leaves.
    - If a node is red, then both children are black. (in other words, never 2 red nodes in a row!)

Insertion

Start insertion as you do in BST. If new node is first in a tree (root) – just make it black (to preserve the rules). Otherwise – make it red.

If **parent is black**: everything is fine (none of rules are violated). We are done.

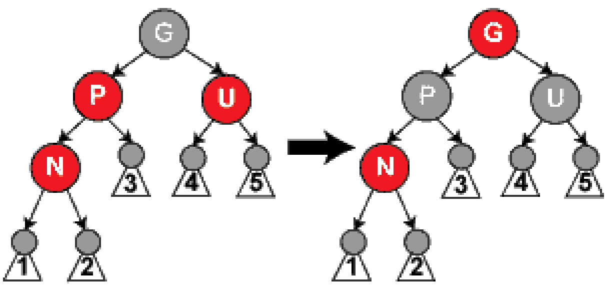
If **parent is red**, we've got the problem. "**Double red problem**".

Double red problem

1) **Parent and uncle are red**: color them in black and make grandpa red (flip color between levels is preserving "black count" rule). This can lead to:

- grandpa is root - make him black.

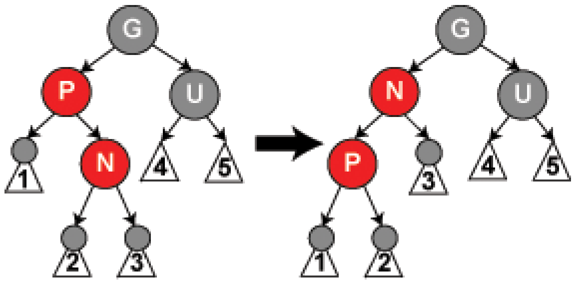
- great-grandpa is red - restart insertion from grandpa's point (assume we inserted grandfather's node, so we again have "double red problem").



We are done.

2a) **Parent is red, uncle is black** (+inserted is right, parent is left): rotate left (N becomes parent for right parent)

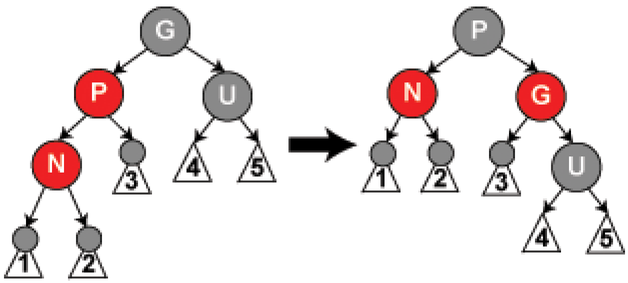
2b) **Parent is red, uncle is black** (+inserted is left, parent is right): rotate right (N becomes parent for left parent)



After (2) tree is still in violated state. So, go on to (3) for **ex-parent** (P node).

3a) **parent is red, uncle is black** (+inserted is left, parent is left): rotate grandpa to the right, make him red, parent - black.

3b) **parent is red, uncle is black** (+inserted is right, parent is right): rotate grandpa to the left, make him red, parent - black.



We're done.

Deletion

When we delete from BST, then we replace deleted node with successor/predecessor that has maximum one child (always). That's why we can consider deletion as deletion of single-child (not more than single child) element always (is node has no children – we can call "child" any of NULL references).

Deleting red

Deleted is red - replace deleted with a child-subtree (all properties preserve).

Deleting black

Easy case

Deleted is black, child is red - replace deleted with a child and make this child black.

Hard case

Deleted and child nodes both black (e.g. when deleted is the last node with 2 nulls). Removal of any black node unbalances the tree.

1) Firstly, replace deleted node with his child.

2) Brother (sibling) of deleted is now brother of new node.

And now let's go:

1) **Deleted was a root node;** his child is a new root. We are done.

2) **Deleted had a red brother**. Flip parent and brother colors and then rotate left (right) over parent to make brother the grandpa of son. Then go to 4, 5, or 6.

3) **Parent, brother, brother's children are black**. Make brother red! Start from (1) for a parent (he is black)!

For next conditions consider child node of deleted becomes left child after deletion.

4) **Parent is red, sibling and his children are black**. Flip their colors, we are done.

5) **Brother is black, his children are different and right is black** (but we want him to be red!). Rotate right over brother to bring red on top. Then flip his and bro's colors. go (6).

6) **Brother is black, his right child is red.** Rotate left over parent. Flip parent's and sibling's colors. Make ex-sibling's right child black. We are done.

For 4-5-6 consider symmetry!

Practical part

1) Implement Red-Black node class.
2) Implement “uncle” and “grandparent” methods.
3) Implement insertion operations for RB-tree.
4) (*) Measure algorithm complexity properties.
    a. Insert 1000000 (1M) values from 0 to 999999 into the tree.
    b. For each insertion, measure **operation time** and **tree height**.
    c. Build 2 graphs that show operation time and tree height depending on number of elements in a tree. Do they fit O(log n) estimation?
    d. Compare to AVL tree measurement on the same graph.
    e. Create a report in LaTeX and send PDF to TA.