CSE221

Lecture 14:
Red-Black Trees

Fall 2021
Young-ri Choi

Acknowledgment: The content of this file is based on the slides of the textbook as well as the slides provided in former lectures at UNIST.

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Red-Black Trees

- Balanced binary search tree
- Guarantee O(log n) insertion, search, delete
- Definition
  - Binary search tree that every node/pointer is colored either red or black
  - Leaf nodes do not contain data
    - Call them "external nodes"

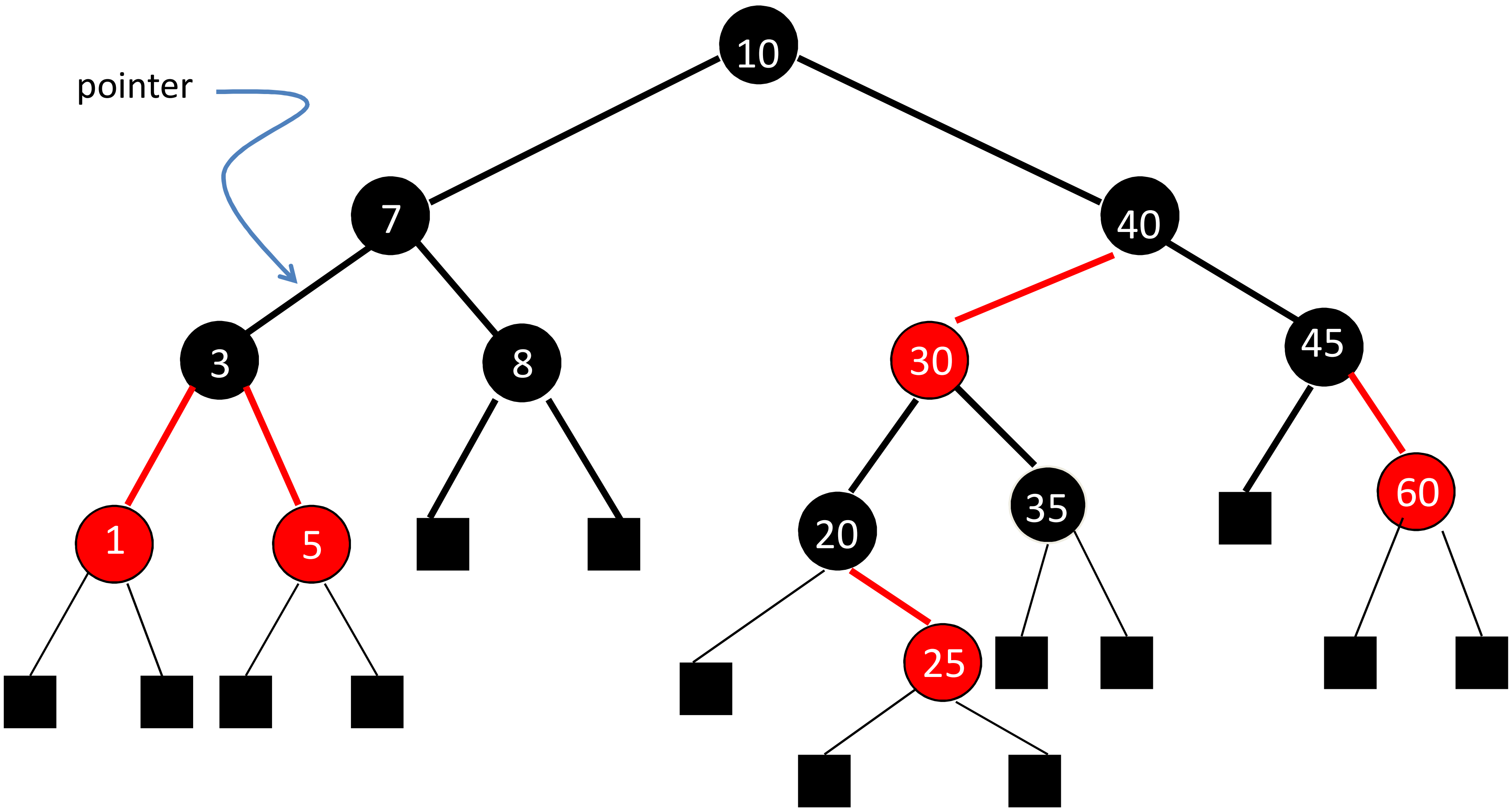
Colored Nodes

- The root and all external nodes are <u>black</u>
- No root-to-external-node path has two consecutive red nodes
  - Red node must have two black children
- All root-to-external-node paths have the same number of black nodes

Colored Pointers

- Pointers from an internal node to an external node are black
- No root-to-external-node path has two consecutive red pointers
- All root-to-external-node paths have the same number of black pointers

Example Red-Black Tree

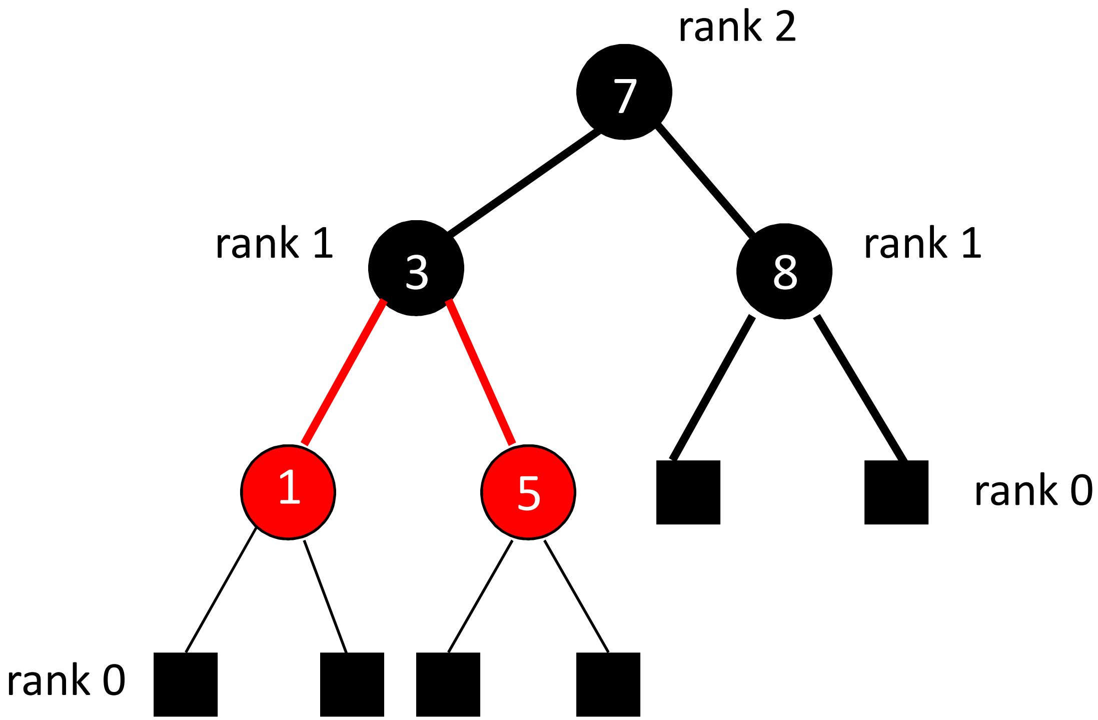


- The pointer from a parent to a black child is black and to a red child is red.

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Properties

- Rank : # of black pointers on any path from a node to any external node



Properties

- If P and Q are two root-to-external-node paths

  length(P) <= 2*length(Q)

  length= the number of pointers on the path

- Longest path length is bounded by 2*shortest path length
  - Shortest path : B-B-B-....-B
  - Longest path : B-R-B-R...-B
  - Number of B must be same for all paths by definition, so the above statement is true

Properties

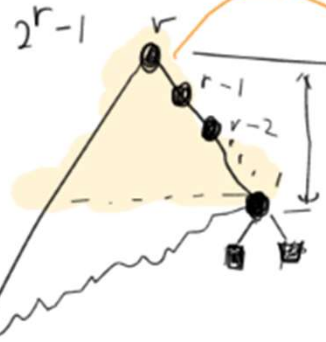
- h : height, r : rank of the root, n : # of nodes

① $h \le 2r$

longest path length $\le$ 2·shortest path length

$h \le$ longest path length $\le 2 \cdot r$

② $n \ge 2^r - 1$



$r \Rightarrow 2^r - 1$ nodes

$\therefore n \ge 2^r - 1$

$\therefore \log_2(n+1) \ge r \ge \frac{1}{2}h$ (by ① & ②)

$\therefore h \le 2\log_2(n+1)$

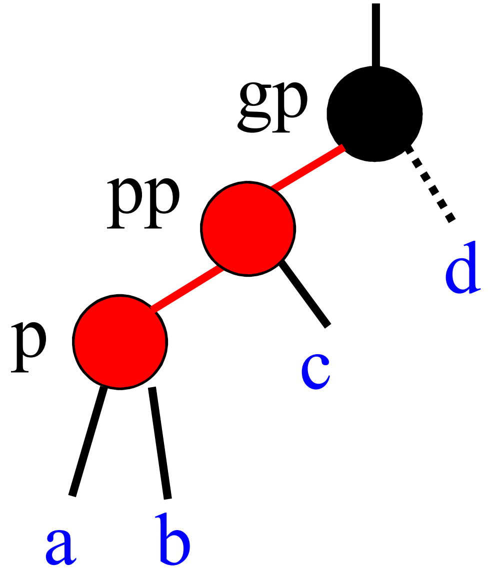
$\Rightarrow$ search, Insert, delete $\simeq O(h) = O(\log n)$

Insert

- Same as regular binary search tree insertion
- Must assign color
  - If the tree is empty, new node is root so assign black
  - If the tree was not empty, assign black node in the path?
    - NO! Why?
    - Will violate same # of black nodes for all paths
      - difficult to resolve
  - Then, assign red?
    - May (or may not) lead to two consecutive red nodes in the path?
    - Can be resolved by rotation and color flips

Classification of 2 Red Nodes/Pointers

pp: parent
gp: grand parent

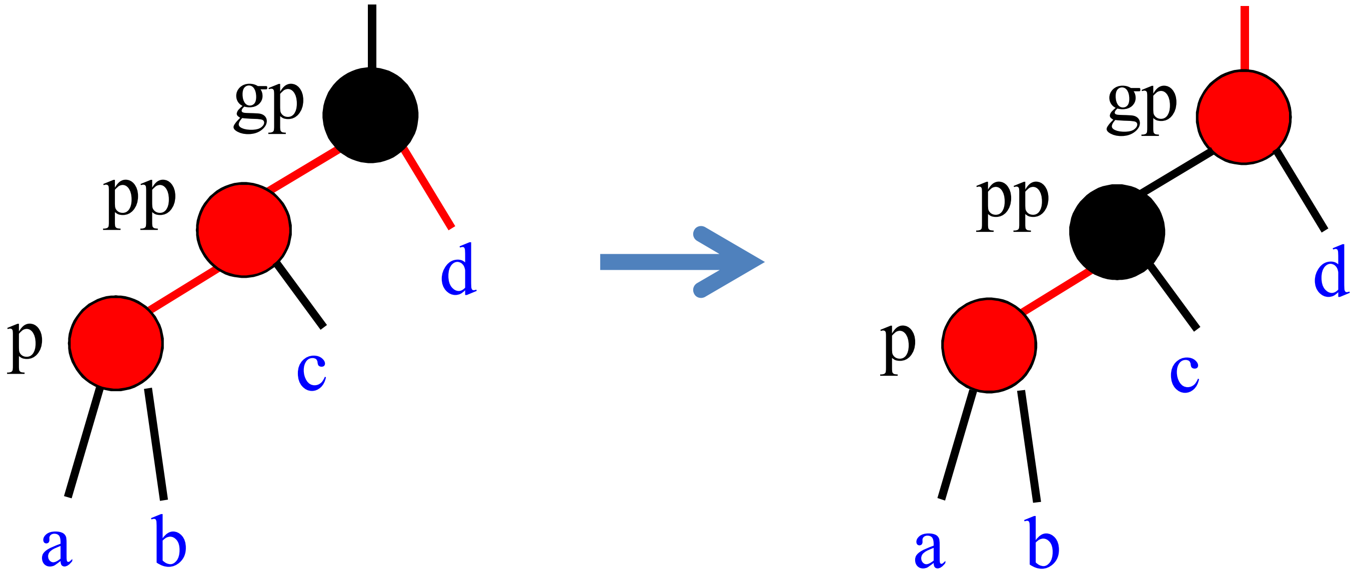


LLb example

- XYz
  - X: relationship (Left or Right) between gp and pp
  - Y: relationship (Left or Right) between pp and p
  - z: color (red or black) of the other-side node of gp

XYr

- Perform color flipping

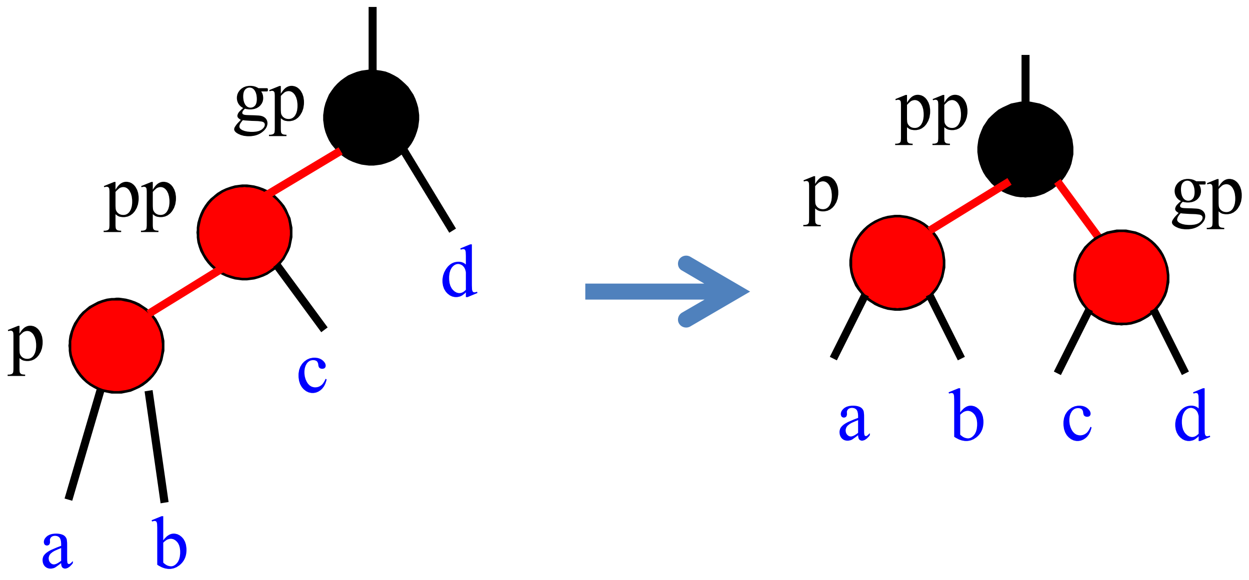


- Flip color of pp, gp, d and pointers of gp
  - Does not increase # of black
- Reapply transformation to gp by p ← gp
  - Because gp’s parent might be red

LLb

- Perform rotation

RRb is symetric

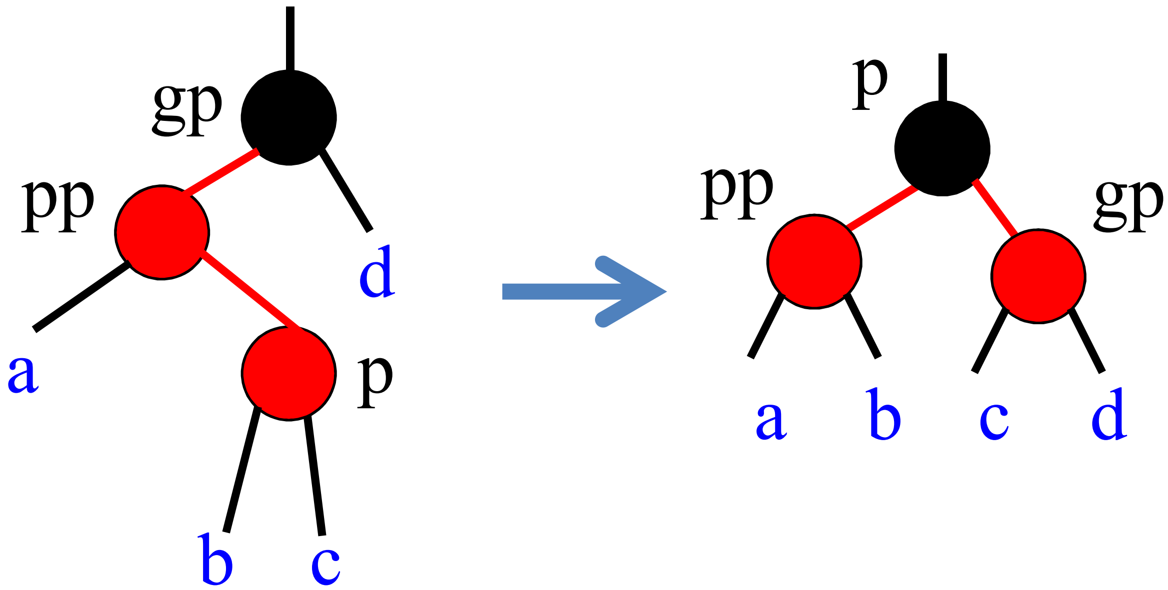


- Same as LL rotation of AVL tree (same inorder)
- Flip color of pp and gp after rotation
- No reapply: root of this subtree is still black

LRb

- Perform rotation

RLb is symetric

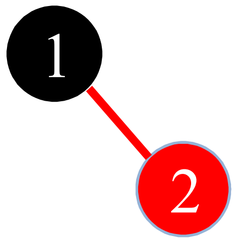


- Same as LR rotation of AVL tree (same inorder)
- Flip color of p and gp
- No reapply

Insert Example

- Insert 1
  - –Root

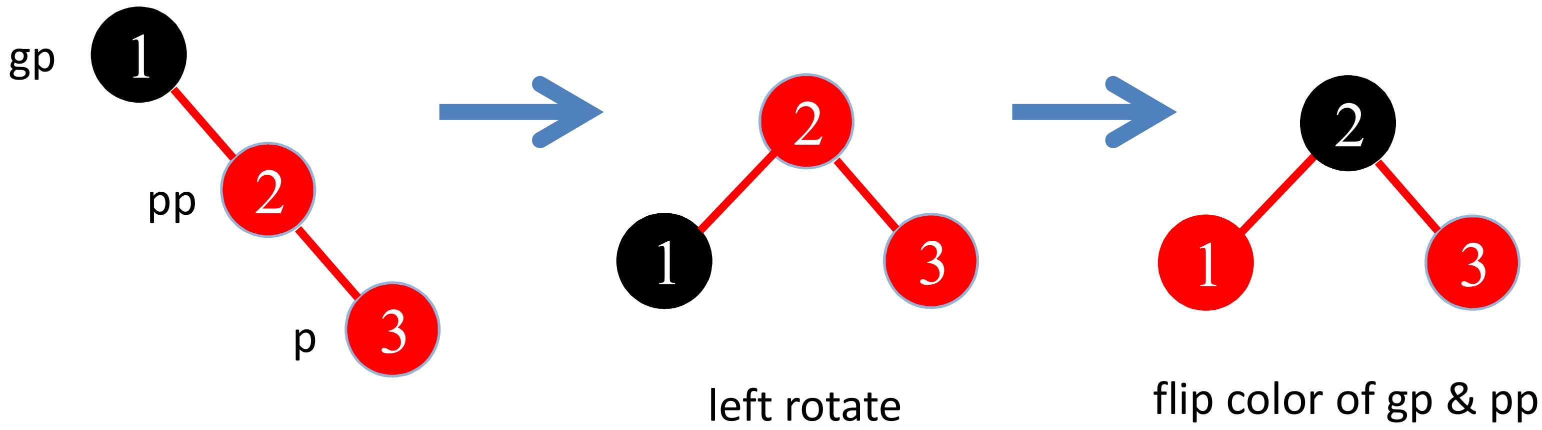
- Insert 2
  - –Red



ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

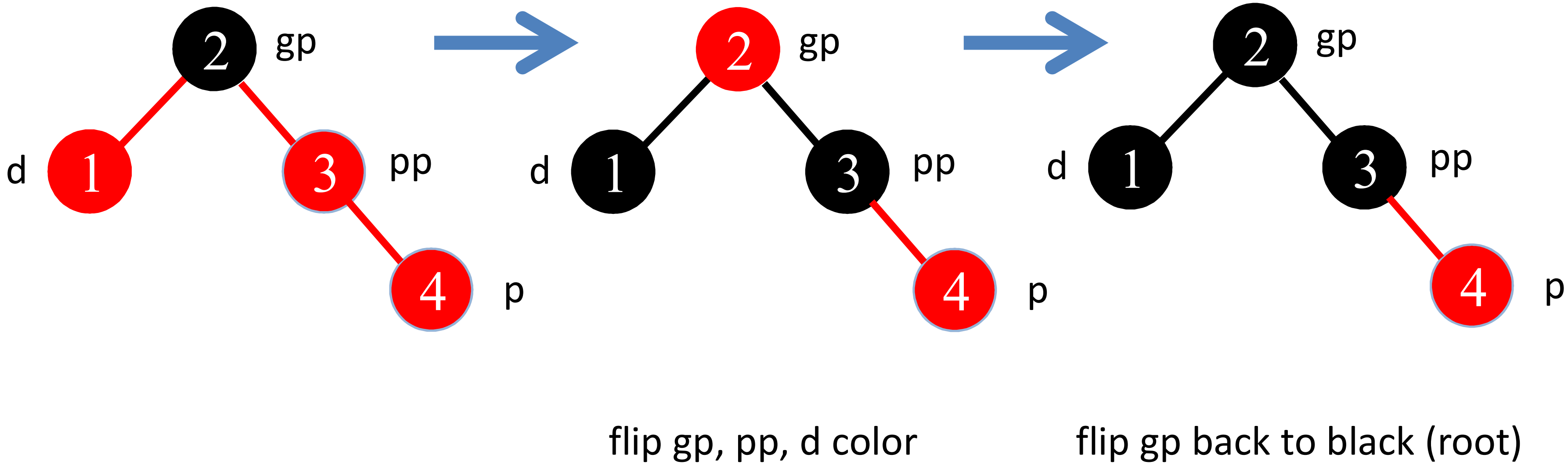
Insert Example

- Insert 3
  - –RRb: rotate (external node at left child of gp: black)



Insert Example

- Insert 4
  - RRr: flip color



- If gp is not root
  - Move up and reapply transformation because gp's parent might be red
  - p ← gp, pp ← gp's pp, gp ← gp's gp

Insert Example

- Insert 5
  - RRb: rotate