Data Structures and Algorithms (12)

Instructor: Ming Zhang
Textbook Authors: Ming Zhang, Tengjiao Wang and Haiyan Zhao

Higher Education Press, 2008.6 (the "Eleventh Five-Year" national planning textbook)

https://courses.edx.org/courses/PekingX/04830050x/2T2014/

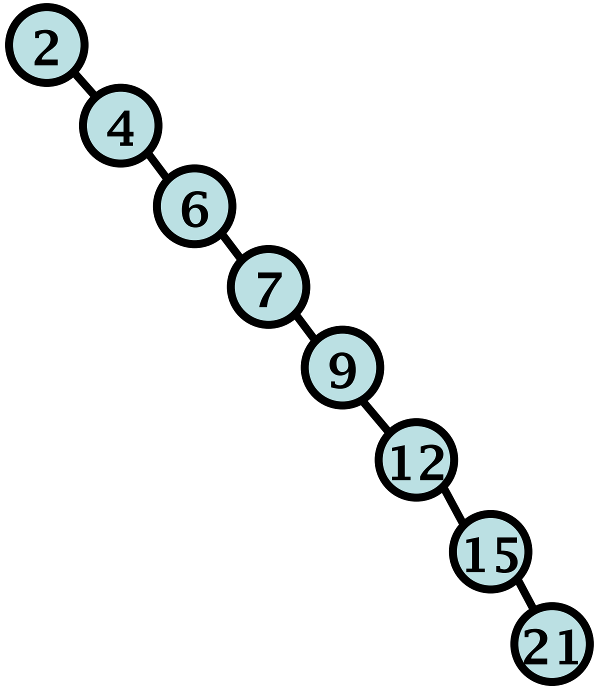
Chapter 12 Advanced data structure

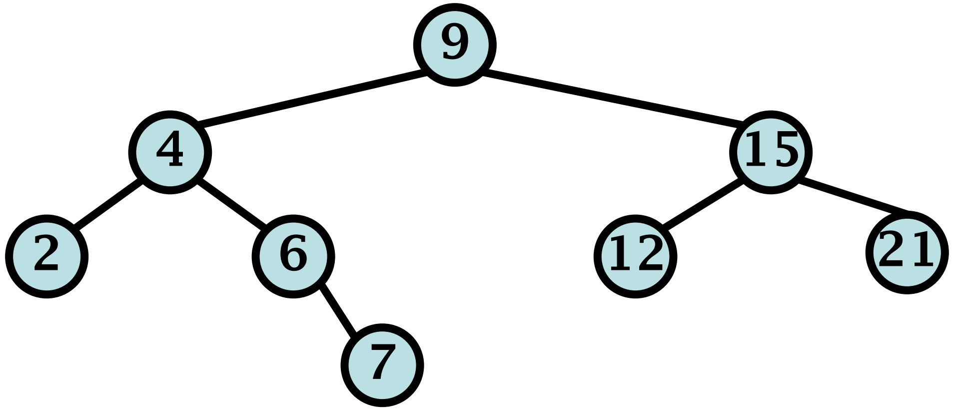
- 12.1 Multidimensional Array
- 12.2 Generalized Lists
- 12.3 Storage management
- 12.4 Trie
- 12.5 Improved binary search tree
  - 12.5.1 Balanced binary search tree
    - Concept and inserting operation of AVL tree
    - Deleting operation and efficiency analysis of AVL tree
  - 12.5.2 Splay Tree

12.5 Improved binary search tree

12.5.1 AVL

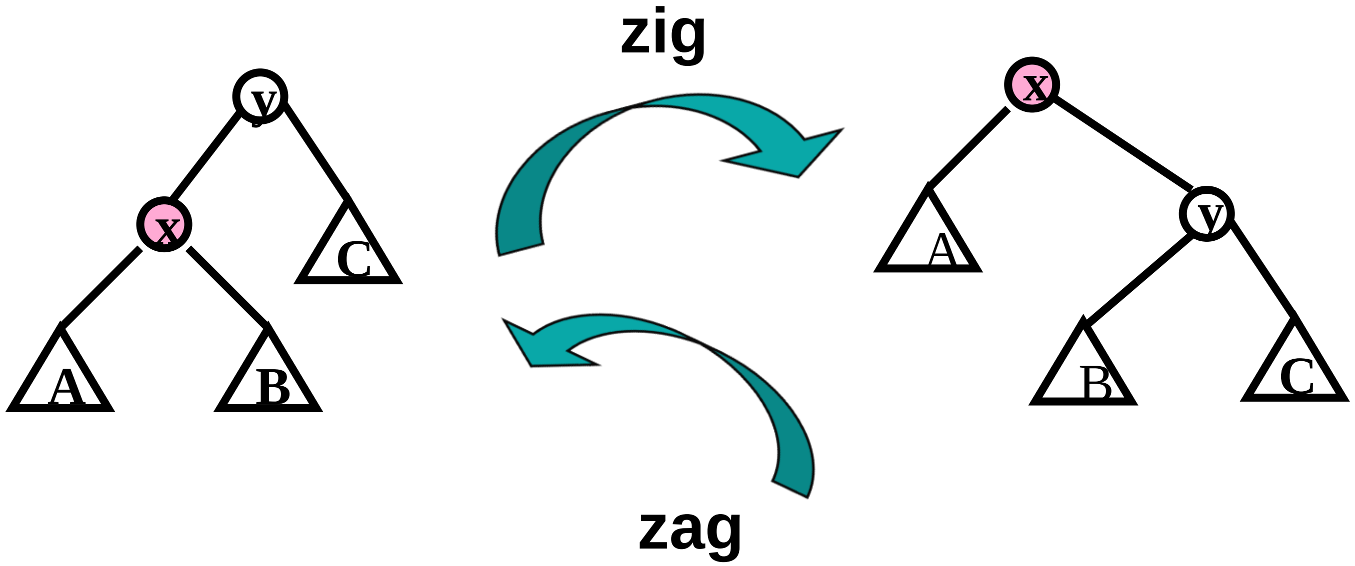
- The performance of BST operations are affected by the input sequence
  - Best O(log n); Worst O(n)
- Adelson-Velskii and Landis
  - AVL tree, a balanced binary search tree
  - Always O(log n)





12.5.1 AVL

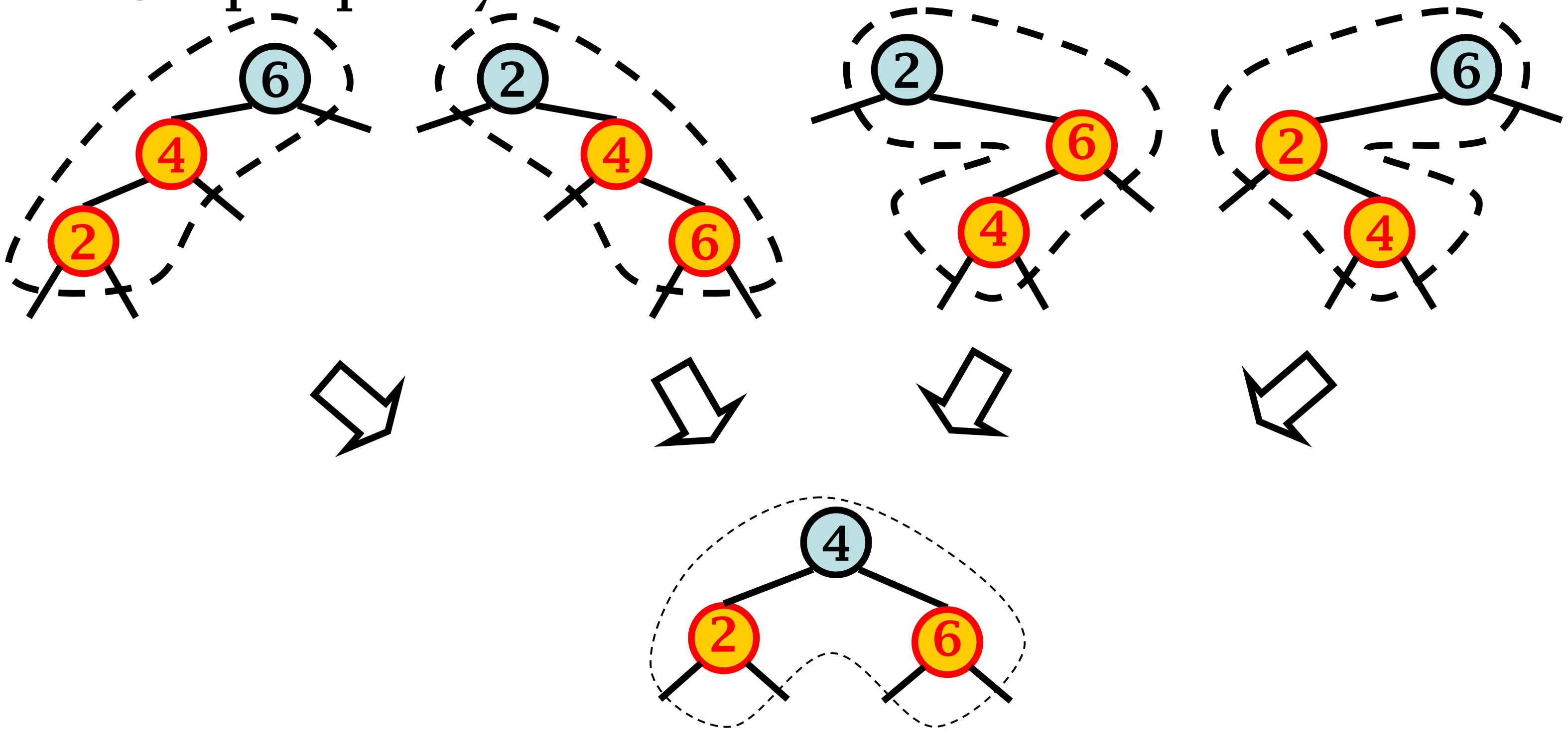
- **Single Rotation**
  - Swap the node with its father, while keeping the property of BST



12.5 Improved binary search tree

12.5.1 AVL

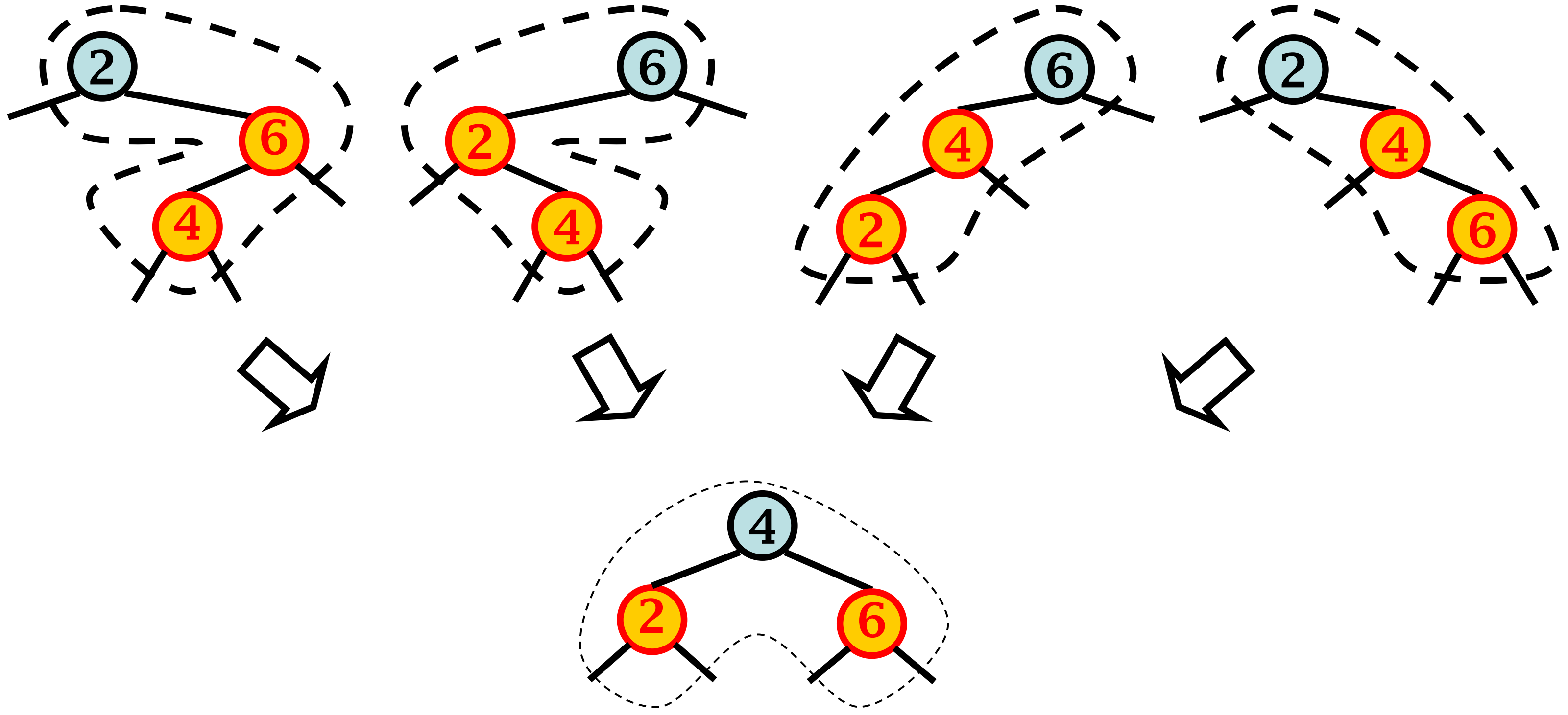
- Single Rotation and Double Rotation: Keep the BST property.



12.5 Improved binary search tree

12.5.1 AVL

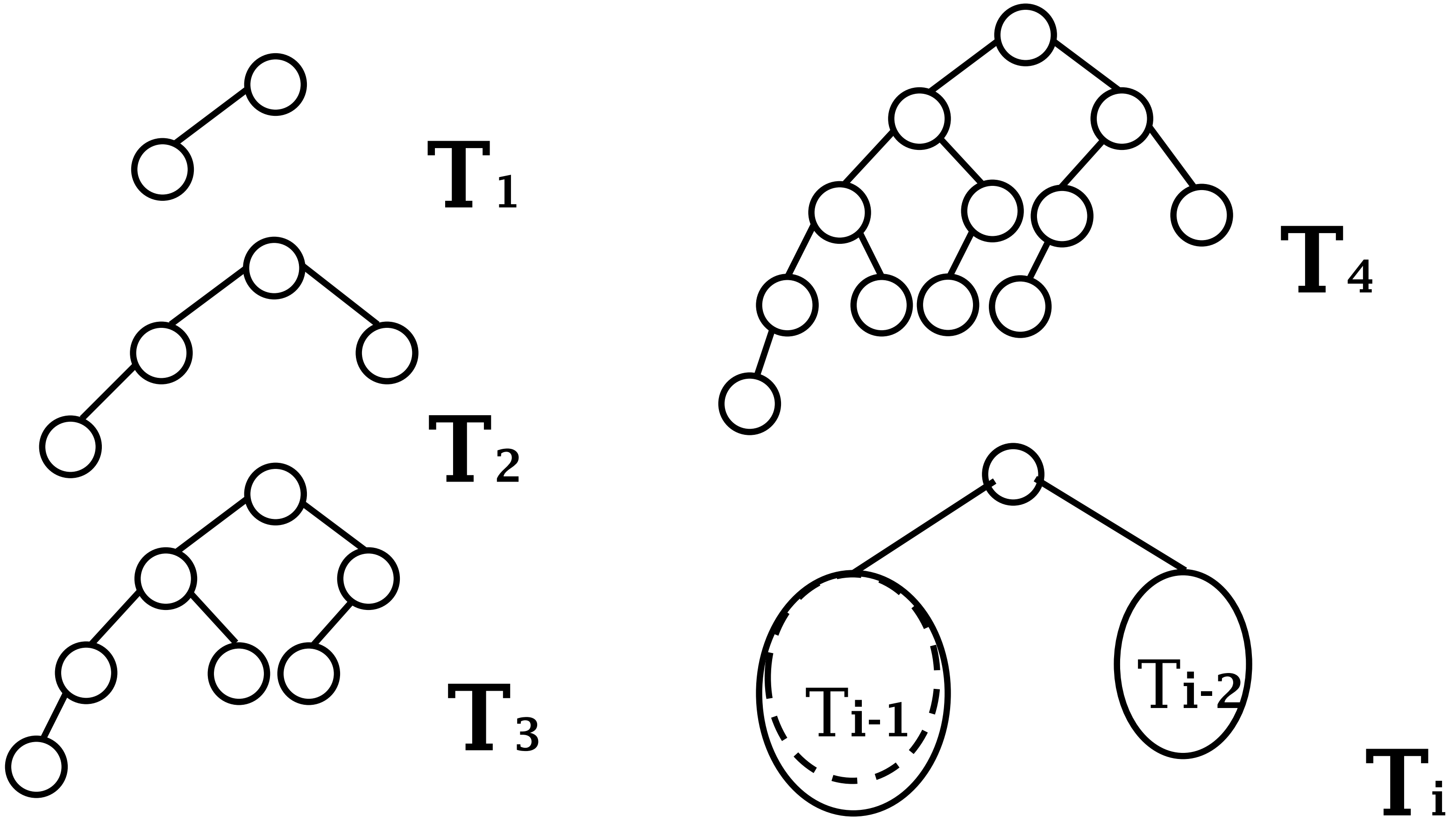
- Equivalent rotation: Keep the BST property



AVL

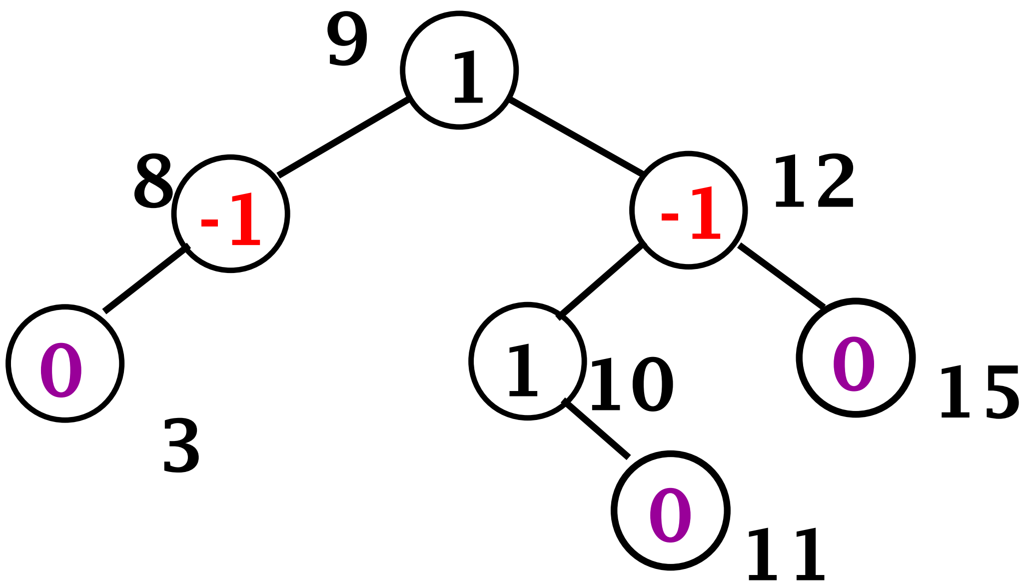
- Empty tree is allowed
- The height of AVL tree with n nodes is O(log n)
- If T is an AVL tree
  - Then the left and right subtree of T: T_L, T_R are also AVL trees
  - And $|h_L - h_R| \le 1$
    - h_L, h_R are the heights of its left and right subtree.

Examples of AVL tree



Balance Factor

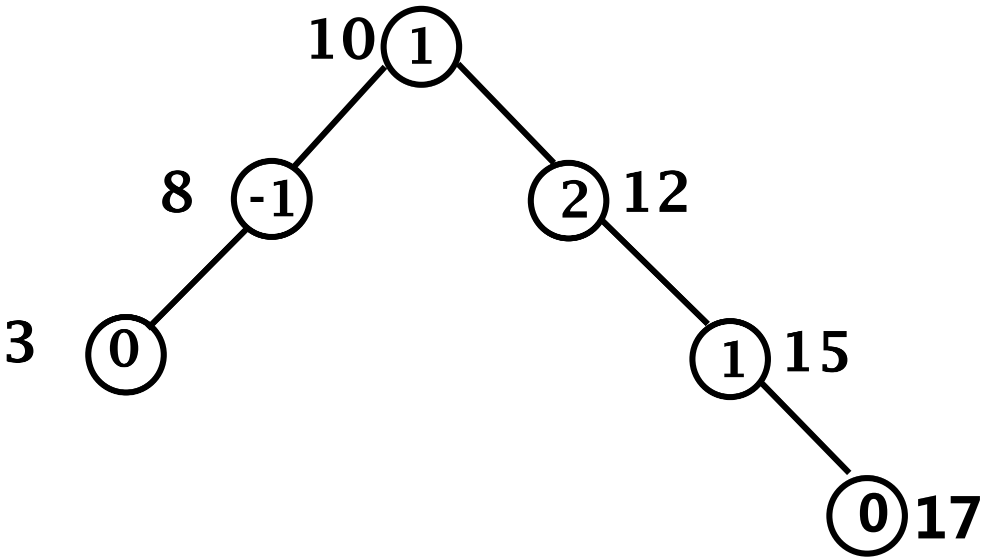
- Balance Factor , *bf* (x):
  - $bf(x) = height(x_{rchild}) - height(x_{lchild})$
- Balance Factor might be 0, 1 and -1



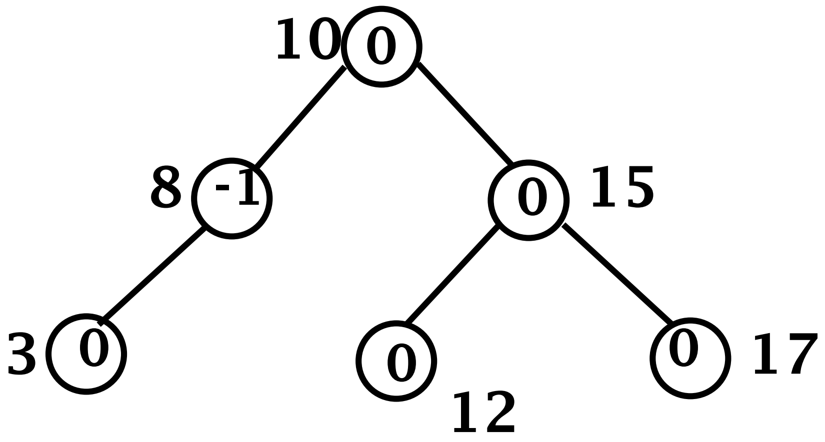
Insertion in an AVL tree

- Just like BST: insert the current node as a leaf node
- Situations during adjustment
  - The current node was balanced. Now its left or right subtree becomes heavier.
    - Modify the balance factor of the current node
  - The current node had a balance factor of ±1. Now the current node becomes balanced.
    - Height stays the same. Do not modify.
  - The current node had a balance factor of ±1. Now the heavier side becomes heavier
    - Unbalanced
    - “dangerous node”

Rebalance



Become unbalanced after inserting 17



Adjustment

12.5 Improved binary search tree

- Unbalanced situation occurs after insertion
- Insert the current node as leaf node in BST
- Assume a is the most close node to the current node. And its absolute value of balance factor is not zero.
- The current node s with key is in its left subtree or its right subtree.
- Assume that it is inserted into the right subtree. The original balance factor:
    - (1) *bf*(a) = -1
    - (2) *bf*(a) = 0
    - (3) *bf*(a) = +1

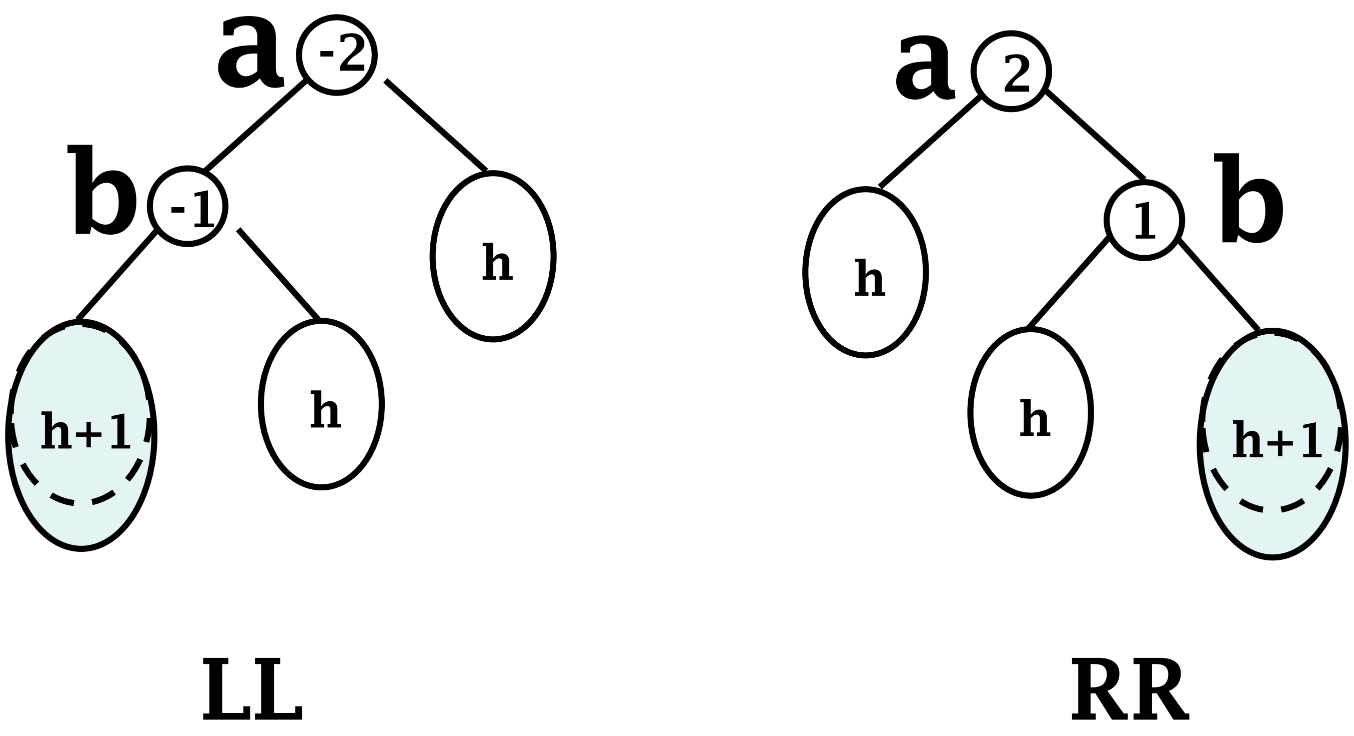
12.5 Improved binary search tree

- Assume a is the most close node to the current node s. And its absolute value of balance factor is not zero.
  - S is in a's left subtree or right subtree.
- Assume S is in the right subtree. Because balance factors of nodes in paths from s to a change from 0 to +1. So as for node a:
  1. *bf*(a) = –1, then *bf*(a) = 0, and the height of node a's subtree stays the same.
  2. *bf*(a) = 0, then *bf*(a) = +1, and the height of node a's subtree stays the same.
     - Because of the definition of a (*bf*(a) ≠ 0), we can know that node a is the root.
  3. *bf*(a) = +1, then *bf*(a) = +2, and adjustment is needed.

Unbalanced Cases

- The balance factors of any nodes must be **0, 1, –1**
- a's left subtree was heavier, *bf*(a) = –1, and *bf*(a) become -2 after insertion.
  - LL: insert into the left subtree of a's left child.
    - Left heavier + left heavier, *bf*(a) become –2
  - LR: insert into the right subtree of a's left child.
    - Left heavier + right heavier, *bf*(a) become –2
- Likewise, *bf*(a) = 1, and *bf*(a) become 2 after insertion
  - RR: the node that causes unbalanced is in the right subtree of a's right child.
  - RL: the node that causes unbalanced is in the left subtree of a's right child.

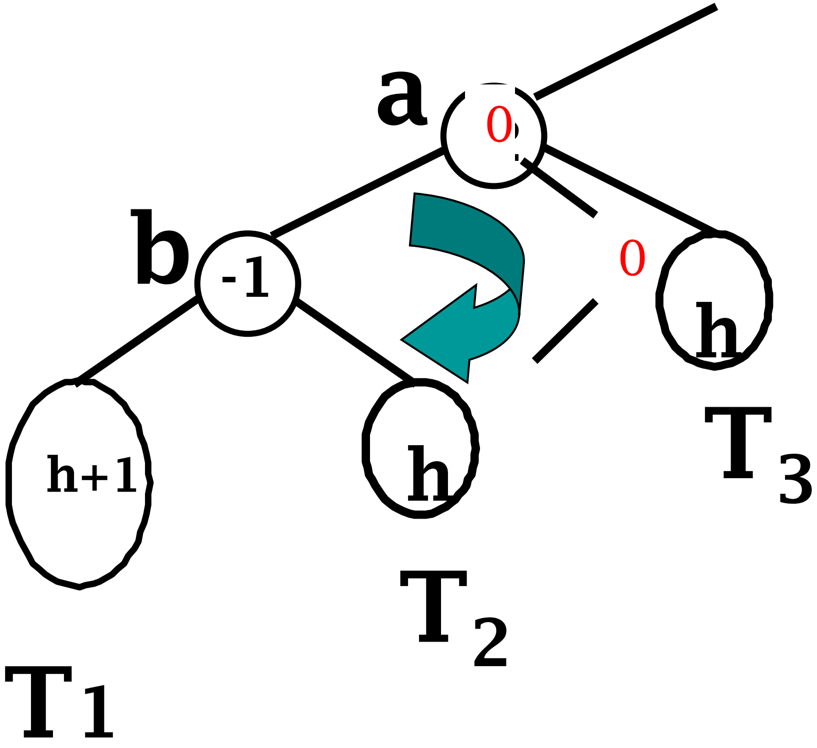
Unbalanced Cases



Summary of unbalanced cases

- LL is symmetry with RR, and LR is symmetry with RL.
- Unbalanced nodes happen on the path from inserted node to the root.
- Its balance factor must be 2 or -2
  - If 2, the balance factor before insertion is 1
  - If -2, the balance factor before insertion is -1

LL single rotation



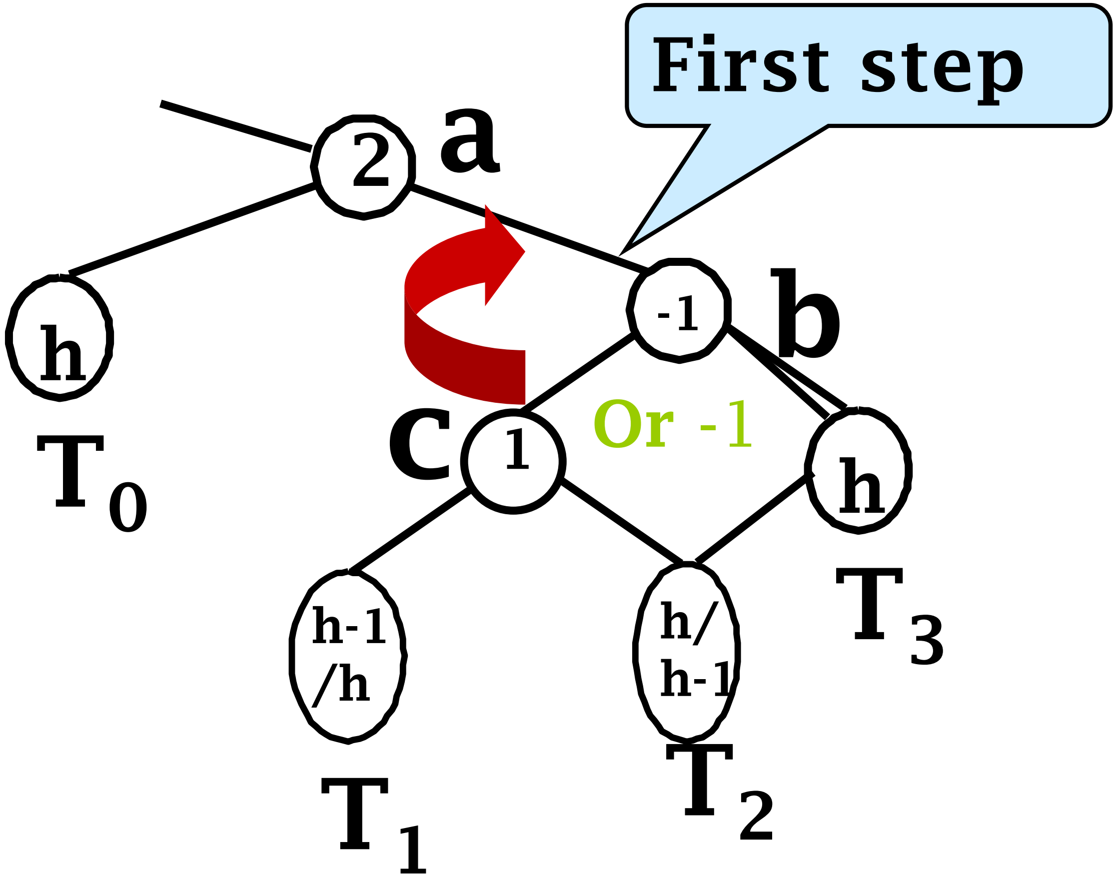
Insight of Rotations

- Take RR for instance, there are 7 parts
  - Three nodes: a, b, c
  - Four subtrees T_0, T_1, T_2, T_3
    - The structure will not change after making c's subtree heavier.
    - T_2, c, T_3 could be regarded as b's right subtree.
- Goal: construct a new AVL structure
  - Balanced
  - Keep the BST property
    - T_0 a T_1 b T_2 c T_3

Double Rotation

- RL or LR needs double rotation.
  - They are symmetry with each other
- We discuss about RL only
  - LR is the same.

First step of RL double rotation

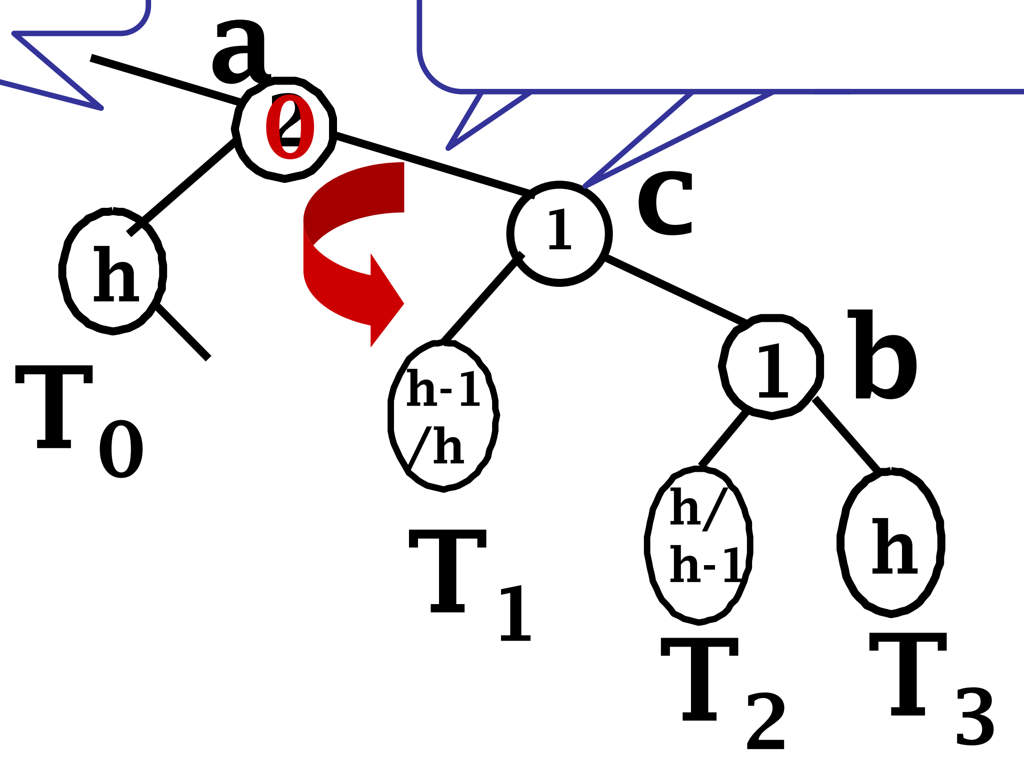


Height of a’s subtree is h+2 before inserting
Height of a’s subtree is h+3 after inserting

Second step of RL double rotation

Balance factor is meaningless in the middle status

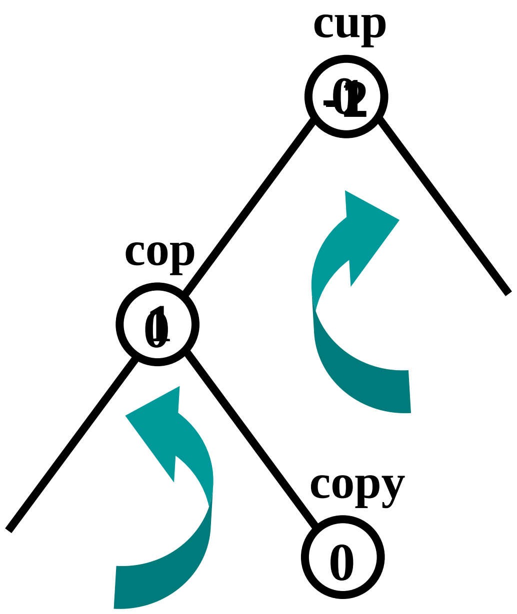
Balance factor of a is -1 or 0
Balance factor of b is 0 or 1



Insight of Rotations

- Doing any rotations (RR, RL, LL, LR)
- New tree keeps the BST property
- Few pointers need to be modified during rotations.
- Height of the new subtree is h+2, and heights of subtrees before insertion stay same
- Rest parts upon node a (if not empty) are always balanced

AVL tree after inserting word : cup, cop, copy, hit, hi, his and hia

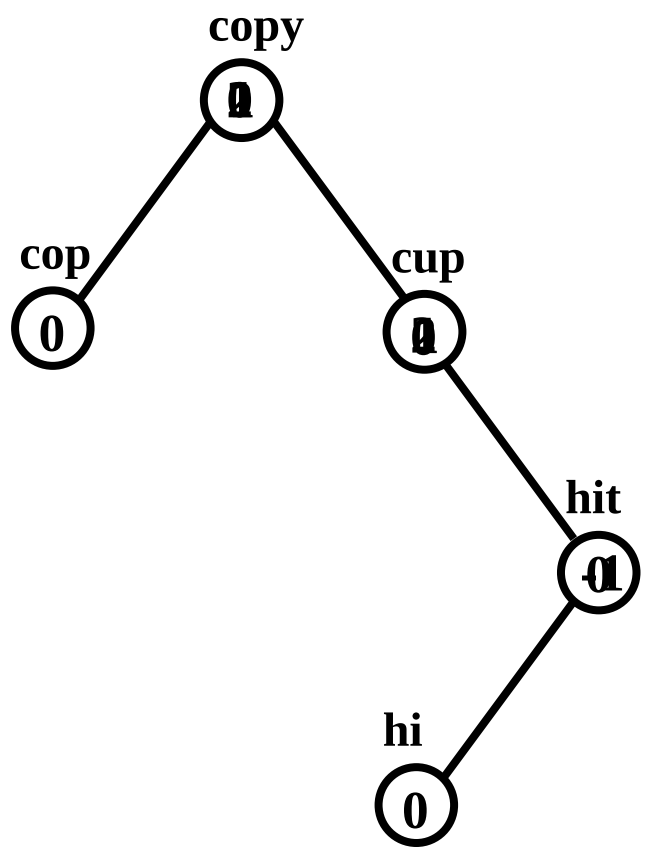


Unbalanced after inserting copy

LR double rotation

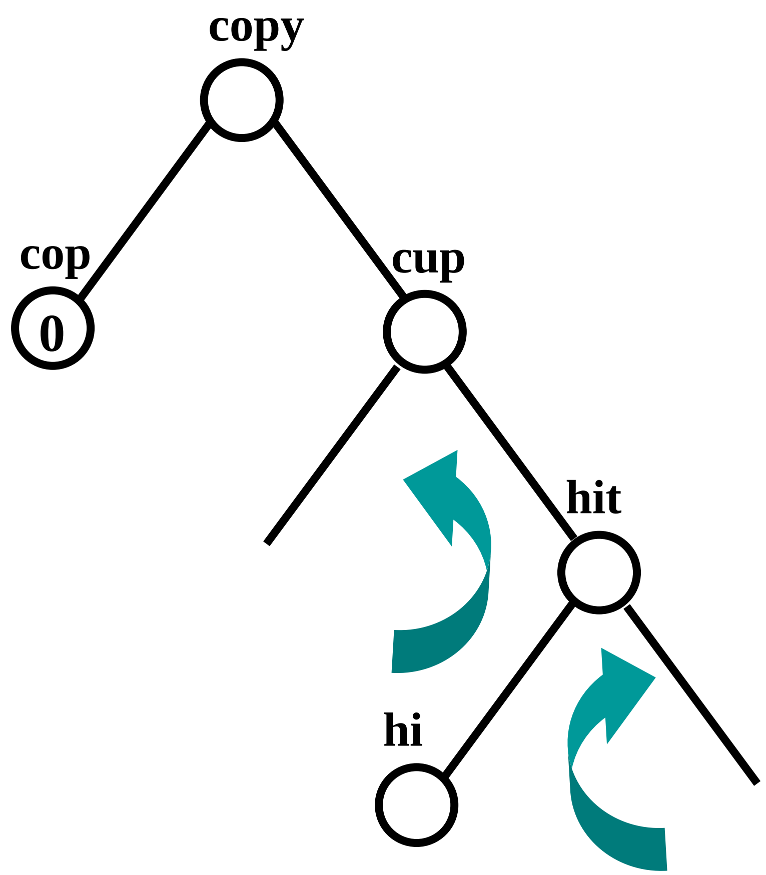
12.5 Improved binary search tree

AVL tree after inserting word : cup, cop, copy, hit, hi, his and hia



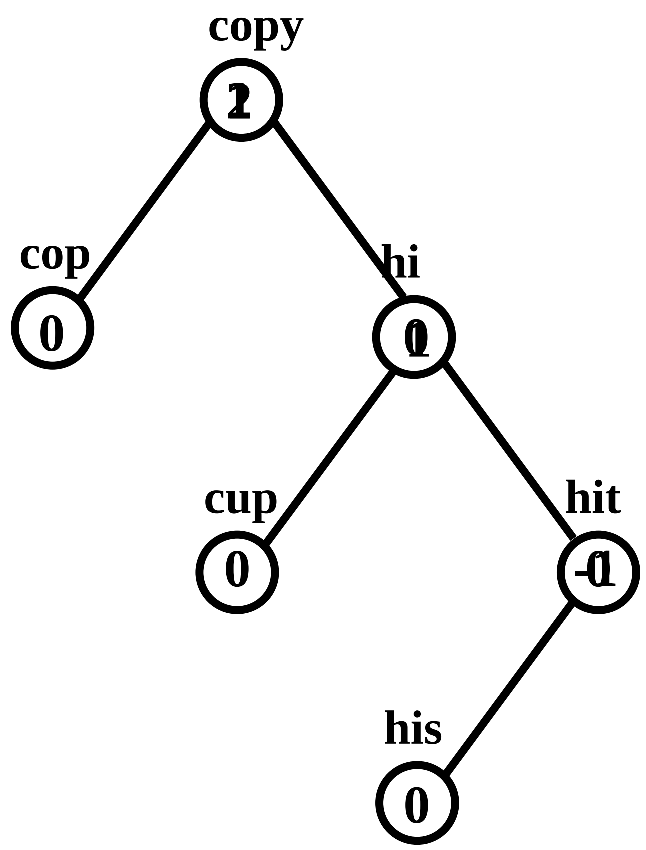
AVL tree after inserting word : cup, cop, copy, hit, hi, his and hia

RL double rotation



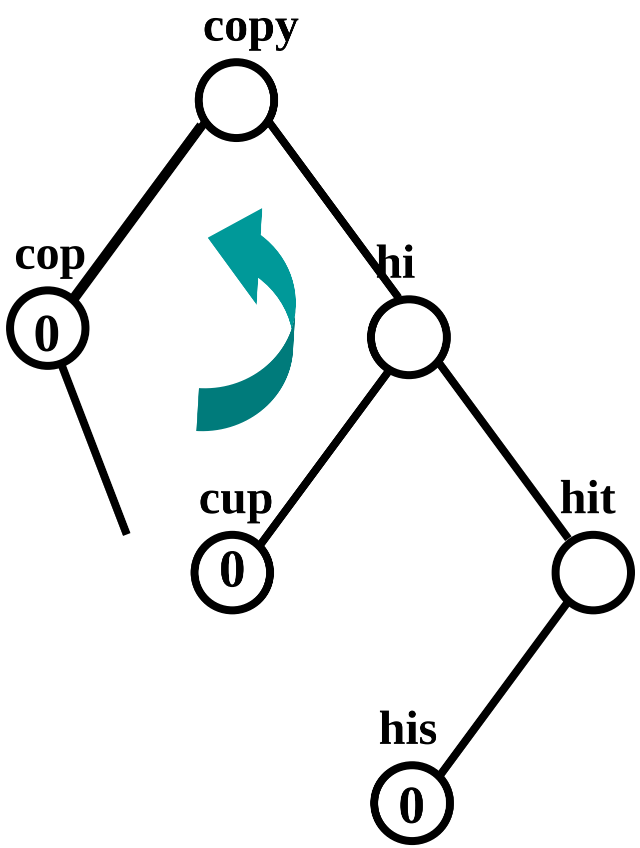
12.5 Improved binary search tree

AVL tree after inserting word : cup, cop, copy, hit, hi, his and hia



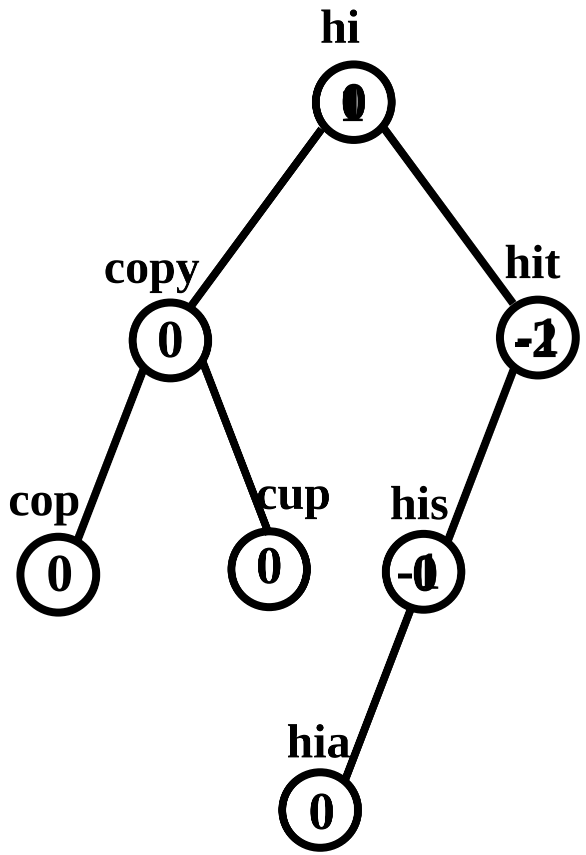
AVL tree after inserting word : cup, cop, copy, hit, hi, his and hia

RR single rotation



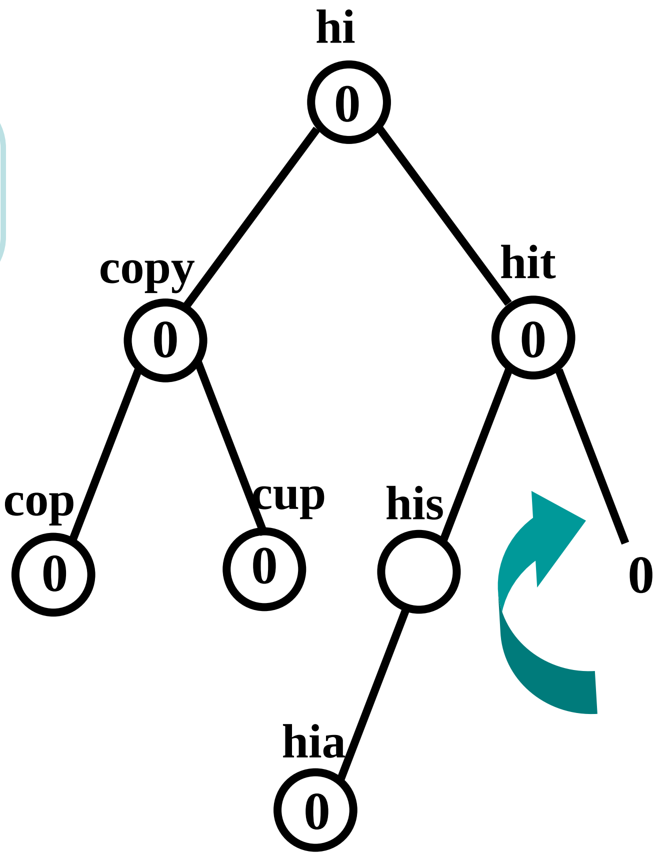
12.5 Improved binary search tree

AVL tree after inserting word : cup, cop, copy, hit, hi, his and hia



AVL tree after inserting word : cup, cop, copy, hit, hi, his and hia

LL single rotation



Discussions

- Can we modify the definition of balance factor of AVL tree? For example, allow the height difference as big as 2.
- Insert 1, 2, 3, ..., 2^k-1 into an empty AVL tree consecutively. Try to prove the result is a complete binary tree with height k.

Data Structures and Algorithms

Thanks

the National Elaborate Course (Only available for IPs in China)
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/
Ming Zhang, Tengjiao Wang and Haiyan Zhao
Higher Education Press, 2008.6 (awarded as the "Eleventh Five-Year" national planning textbook)