높이균형 이진트리

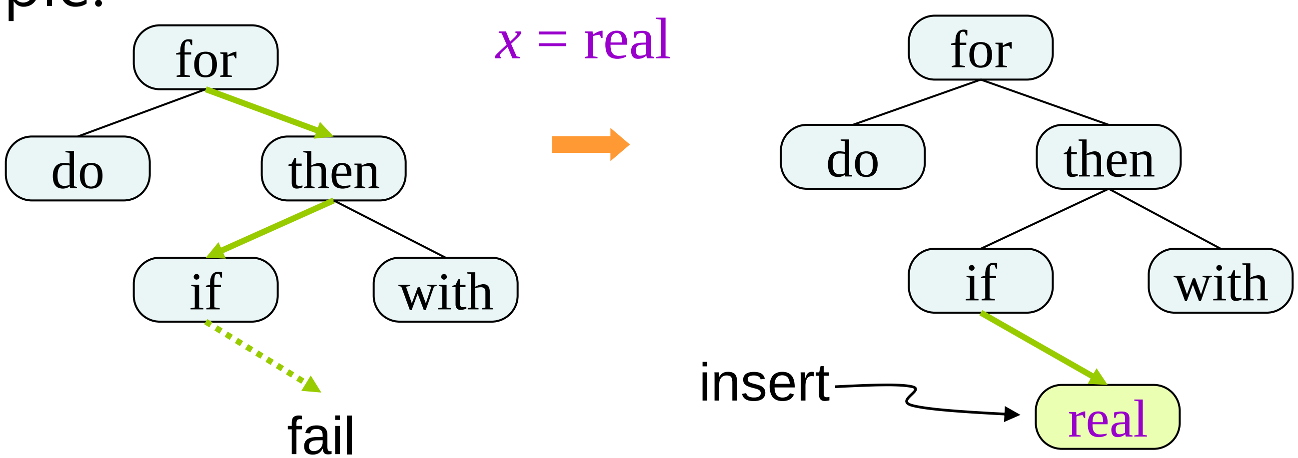
강지훈
jhkang@cnu.ac.kr
충남대학교 컴퓨터공학과

Dynamic Dictionary
By
Binary Search Trees

Dynamic Dictionaries By BST

Binary Search Trees

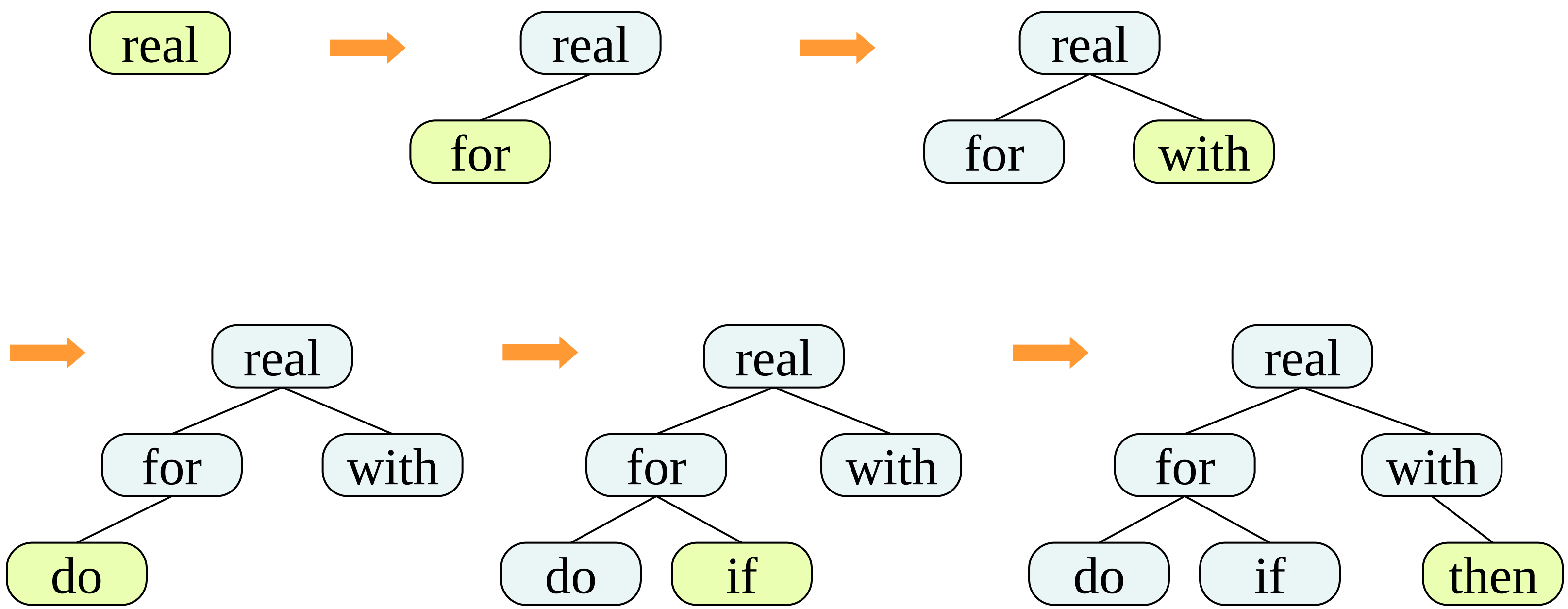
- Search first.
- If we fail to find, we may insert into the fail position.
- Example:



  - Maximum number of comparisons: 3
  - Average number of comparisons: (1+2+2+3+3)/5 = 2.2
- The shape of a binary search tree:
  ⇒ It is determined by the order of insertion.
- Insertion / search time ≈ number of comparisons.

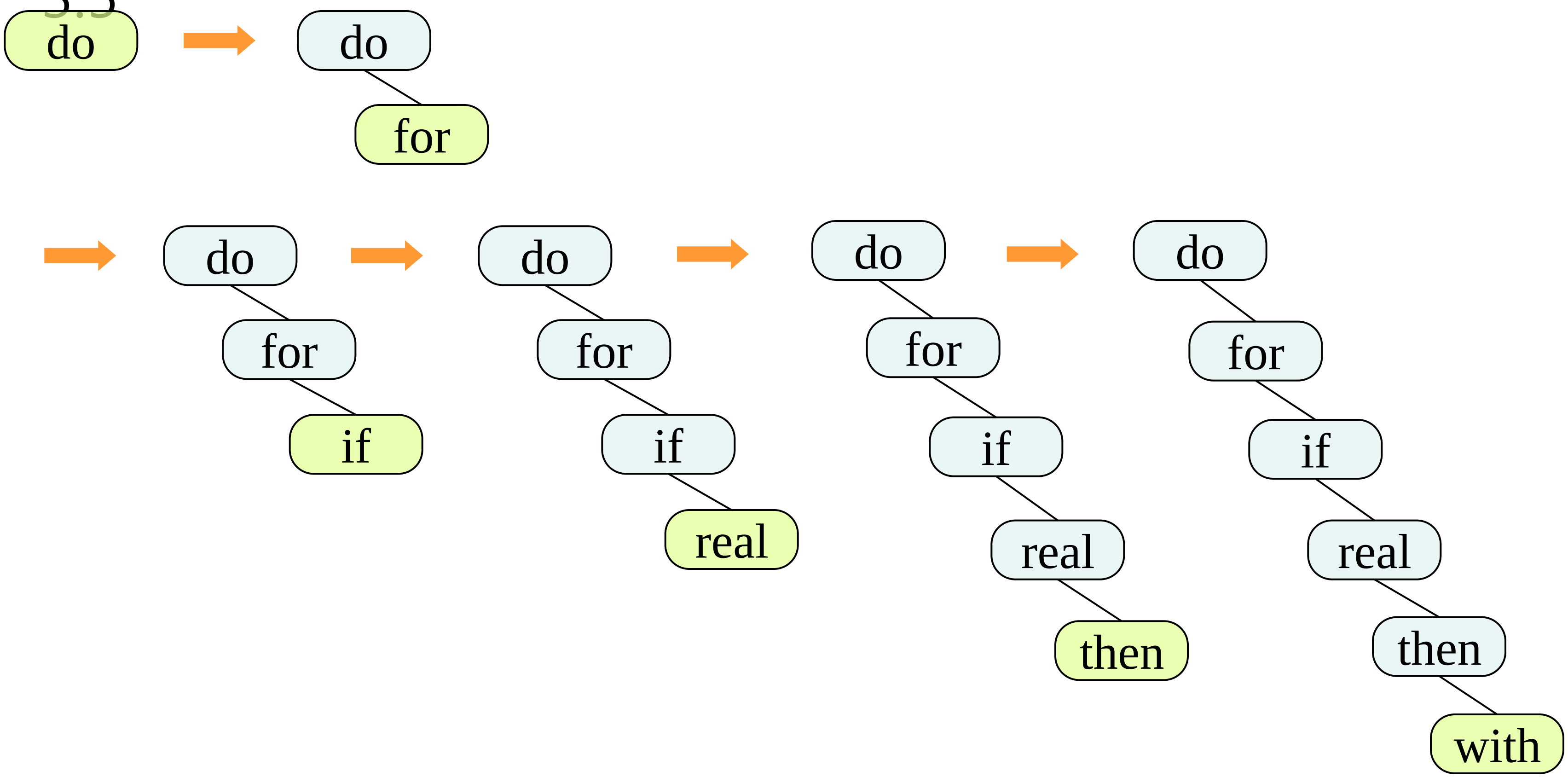
Example: Best case

- Insertion order:
  - real → for → with → do → if → then
- Maximum number of comparisons: 3
- Average number of comparisons: (1+2+2+3+3+3)/6 = 2.3



Example: Worst case

- Insertion order:
  - do → for → if → real → then → with
- Maximum number of comparisons: 6
- Average number of comparisons: (1+2+3+4+5+6)/6 = 3.5



Insertion/Search time in a BST with n nodes.

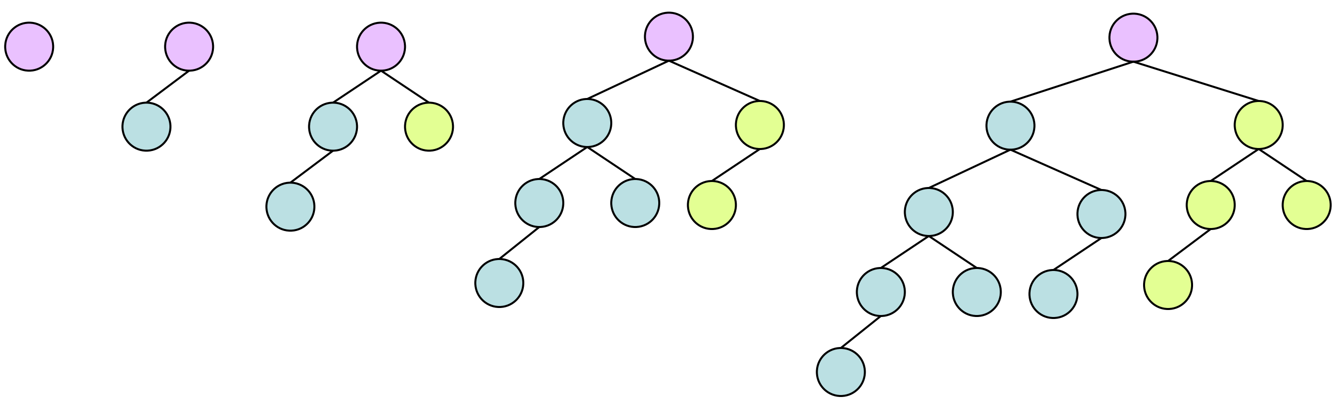
- Observation for time complexities.
  - Best case : a complete BST.
    $T_{bc} = O(\log n)$
  - Worst Case : a degenerate (skewed) BST.
    $T_{wc} = O(n)$
  - Average Case : All permutations of input data are equiprobable, ie, $p(\sigma) = 1/(n!)$ for any permutation.
    $T_{avg} = O(\log n)$
- It may be one of the best ways to maintain a BST as a complete BST.
  - Very high time complexity.
- But, it is possible to keep the trees balanced so as to ensure both an average and worst case retrieval time of $O(\log n)$ for a tree with n nodes.

  $\Rightarrow$ Height Balanced Binary Trees

Dynamic Dictionary By Height-Balanced BST

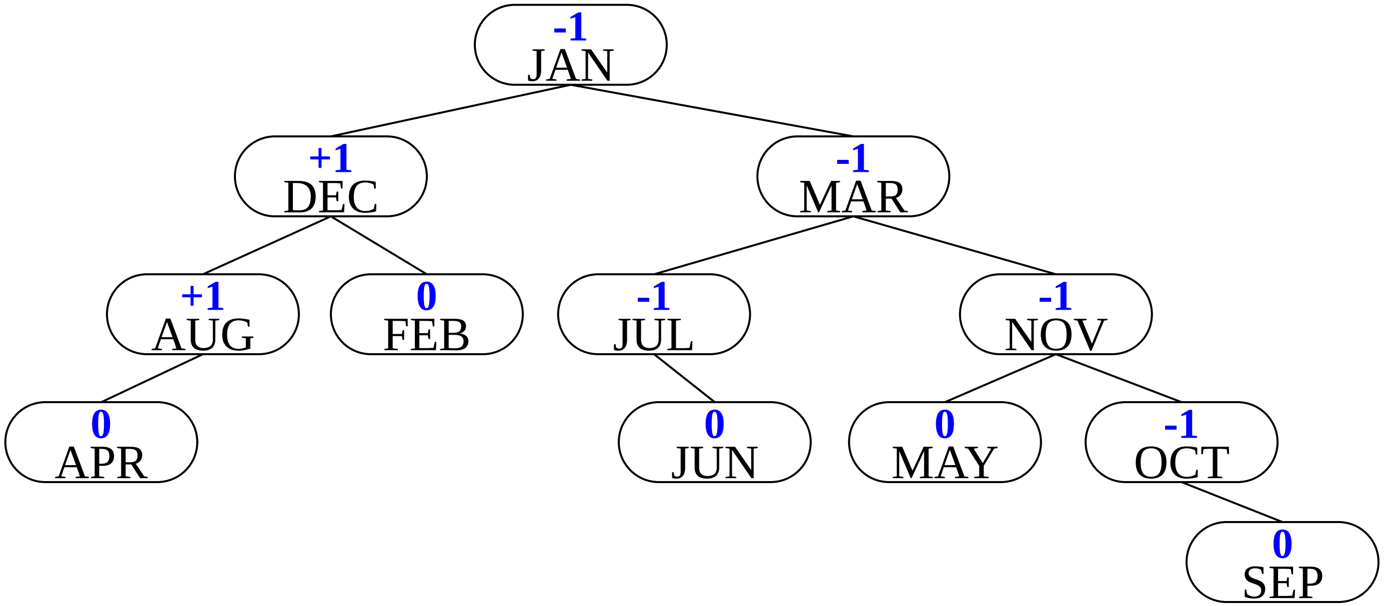
Height Balanced Binary Trees

- AVL-Trees (named from Adelson-Velskii & Landis)
- Search/Insert/Delete : $O(\log n)$
- Definition of height balanced binary trees.
  - An empty tree is height balanced.
  - If T is a nonempty binary tree with T_L and T_R as its left and right subtrees, then T is height balanced
    iff
    1. T_L and T_R are height balanced and
    2. $|h_L - h_R| \leq 1$ where h_L and h_R are the heights of T_L and T_R, respectively.



Balance Factor: $BF(T)$

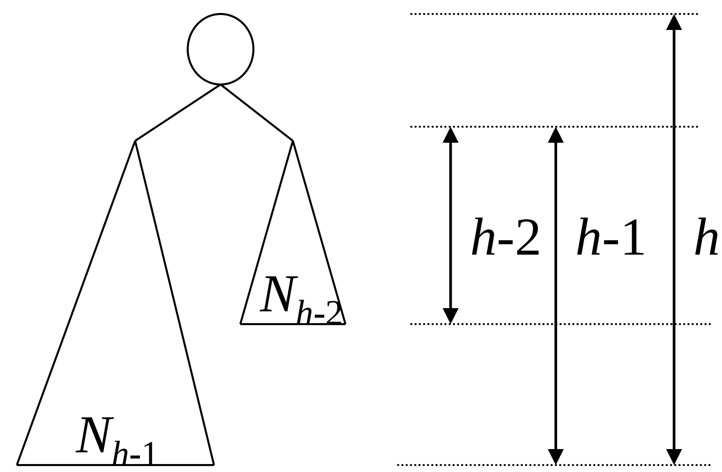
- $BF(T) = h_L - h_R$
  - Clearly, BF is −1, 0, or 1.



Minimum number of nodes N_h.

- Let N_h be the minimum number of nodes in a height balanced tree of height h.

Then, $N_h = \begin{cases} 0 & \text{if } h = 0. \\ 1 & \text{if } h = 1. \\ N_{h-1} + N_{h-2} + 1 & \text{if } h \geq 2. \end{cases}$



The Fibonacci number

$$F_n = \begin{cases} 0 & \text{if } n = 0. \\ 1 & \text{if } n = 1. \\ F_{n-1} + F_{n-2} & \text{if } n \geq 2. \end{cases}$$

The relationship between N_h and F_h is: $N_h = F_{h+2} - 1$ for $h \geq 0$.
(We can prove it by induction easily.)

Complexity of the height h.

Note that $F_h \approx \phi^h / \sqrt{5}$, where $\phi = (1+\sqrt{5})/2$.

$$N_h = F_{h+2} - 1 \approx (\phi^{h+2}/\sqrt{5}) - 1$$

Let $N_h = n$.

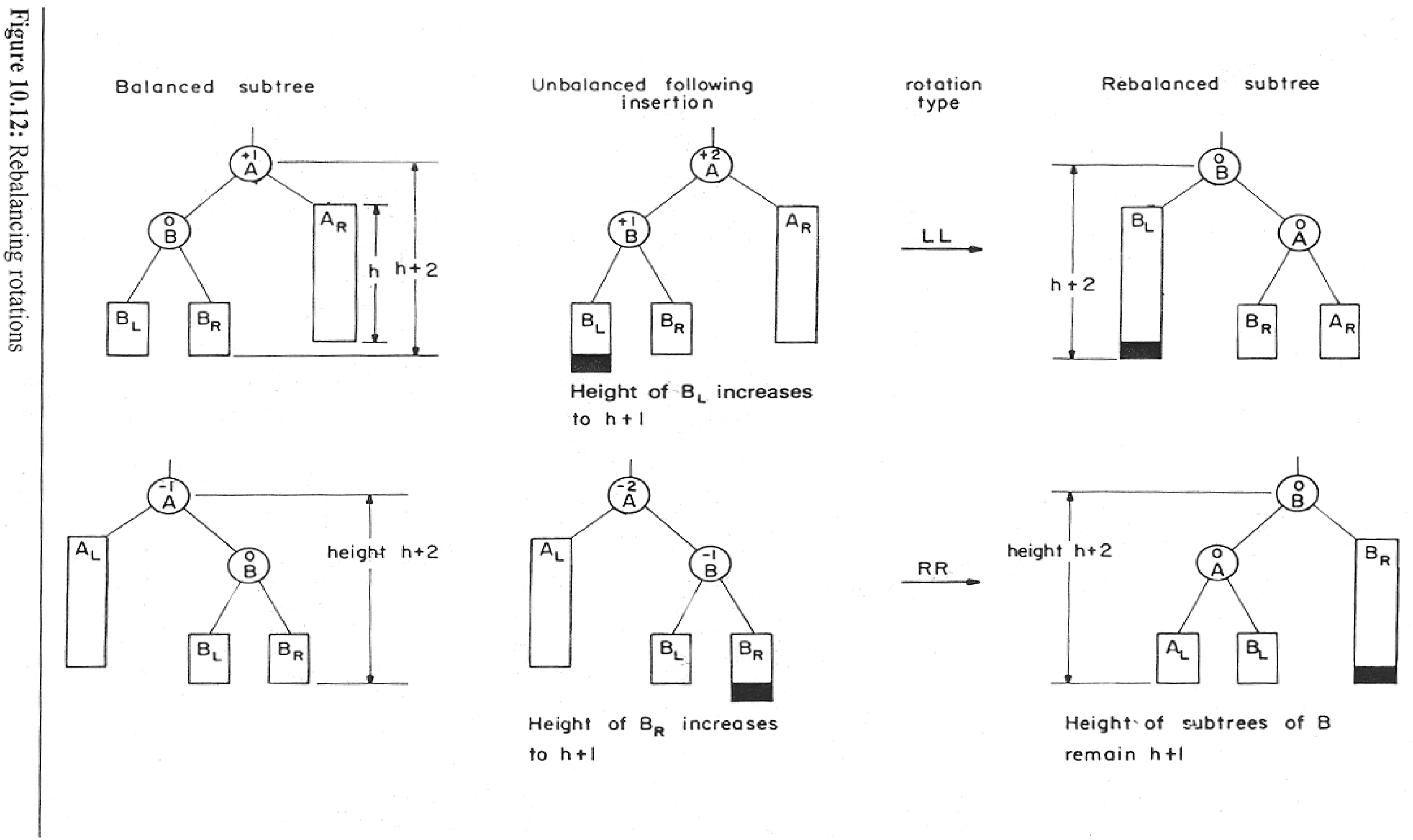
Then, $(\phi^{h+2}/\sqrt{5}) - 1 = n$

$$\phi^{h+2} = \sqrt{5}(n+1)$$

$$h + 2 = \log_\phi(\sqrt{5}(n+1))$$

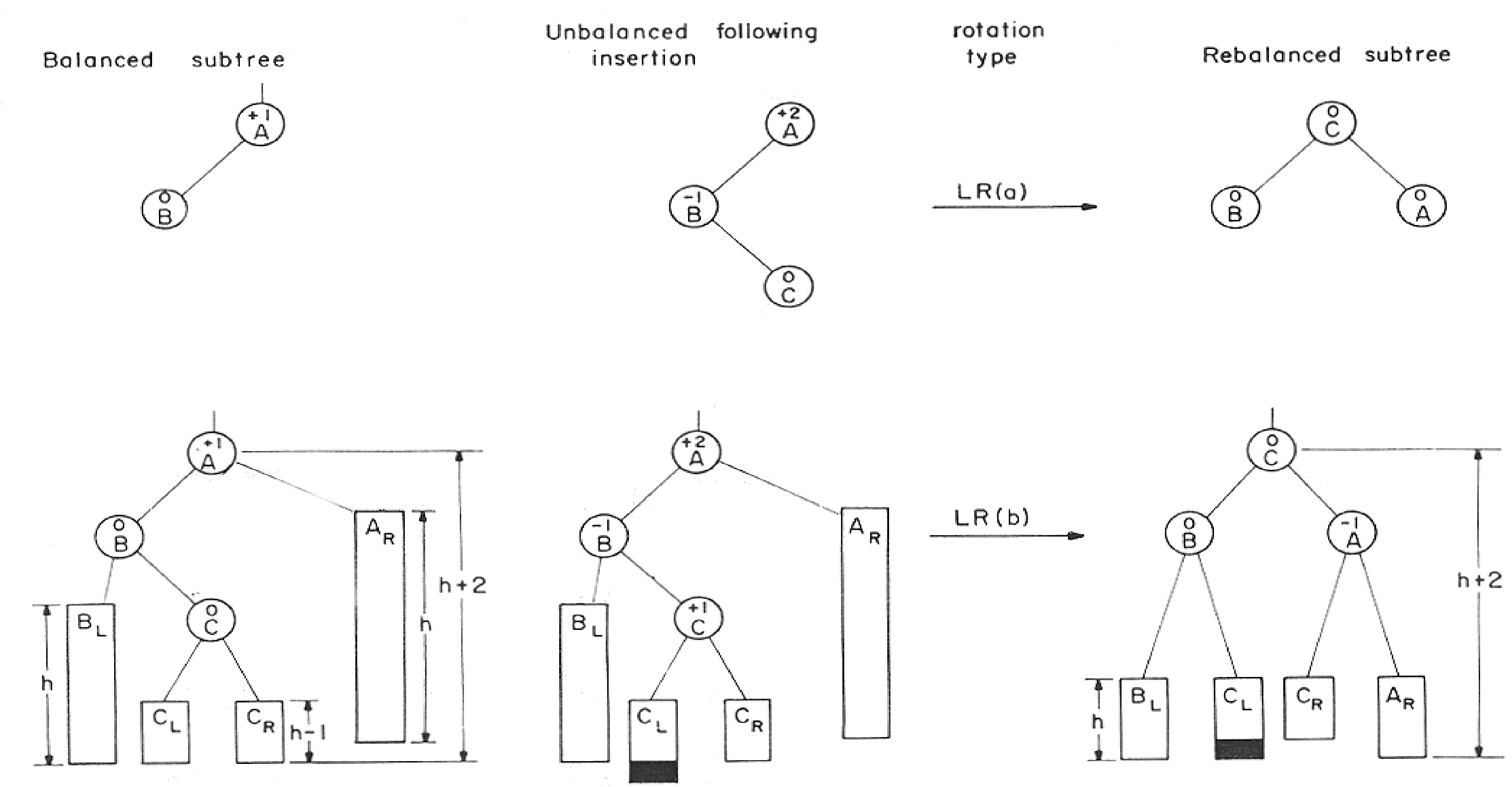
$$h = \log_\phi(\sqrt{5}(n+1)) - 2 = \mathrm{O}(\log n)$$

Rebalancing rotations: LL and RR types



Rebalancing rotations: LR(a) and LR(b) types

Figure 10.12 (continued): Rebalancing rotations



Rebalancing rotations: LR(c) type

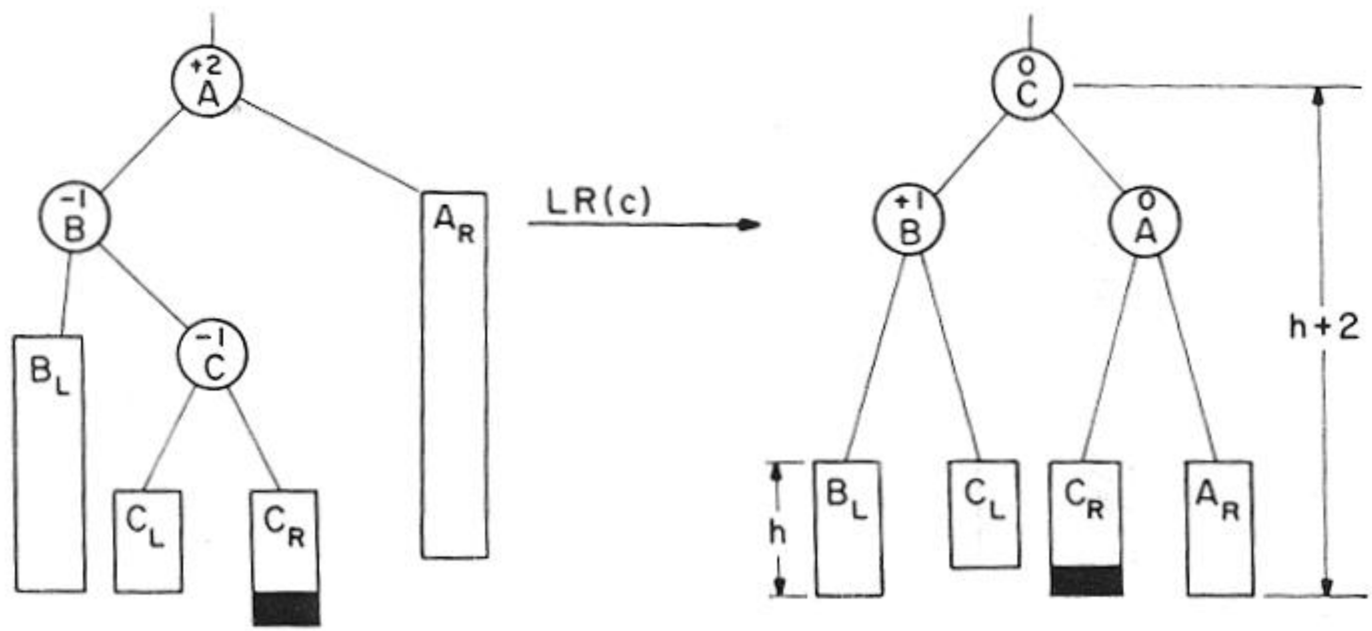


Figure 10.12 (continued): Rebalancing rotations

Insertion into an AVL tree [1]

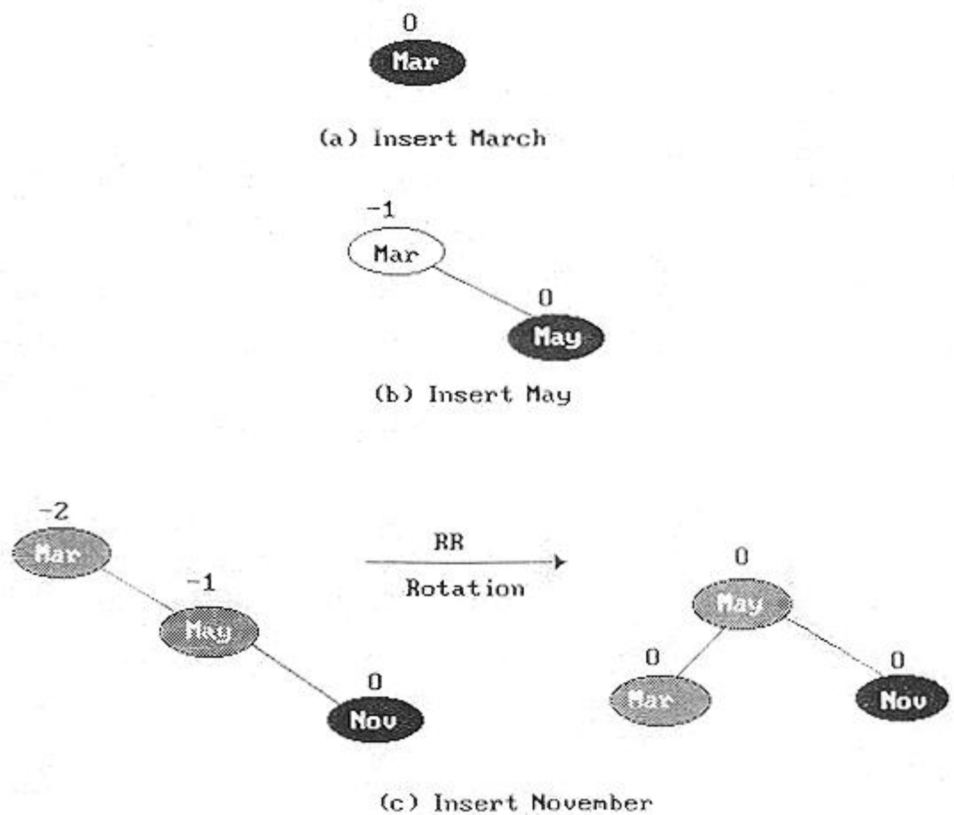


Figure 10.11: Insertion into an AVL tree

Insertion into an AVL tree [2]

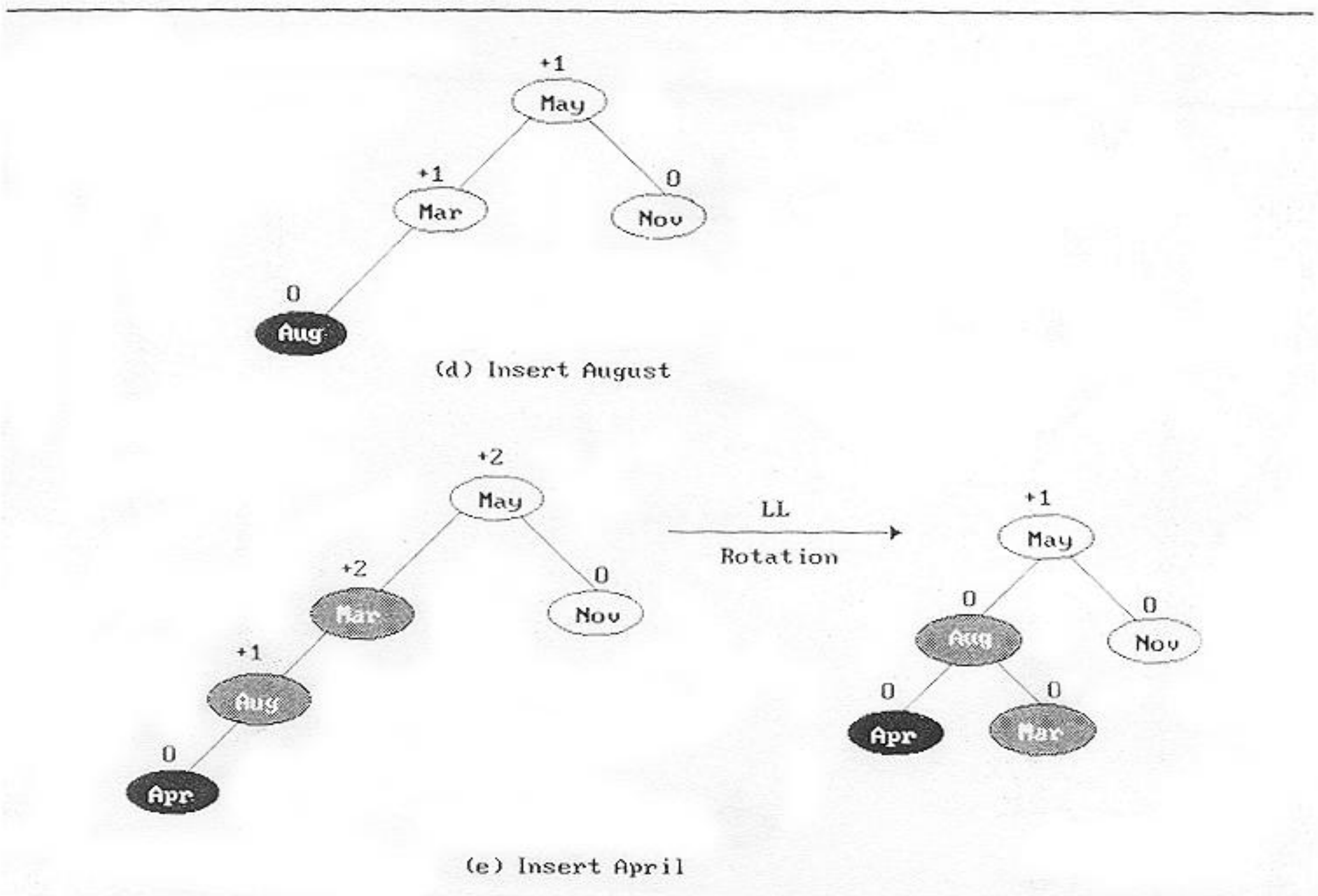
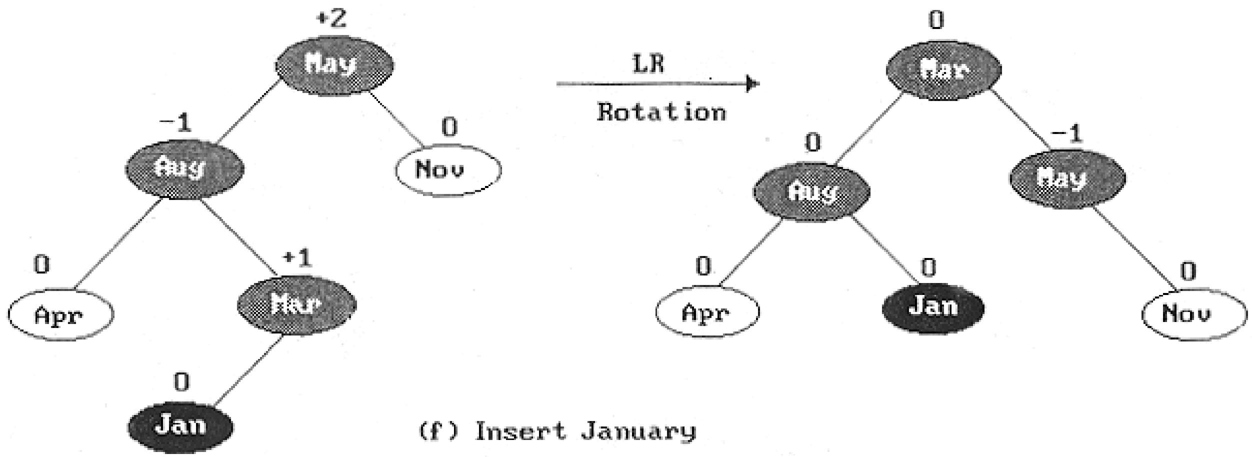
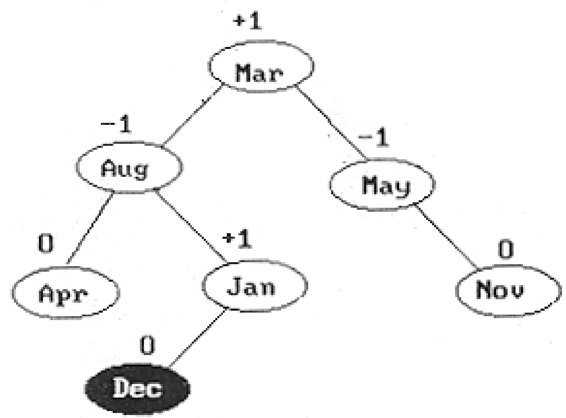


Figure 10.11 (continued): Insertion into an AVL tree

Insertion into an AVL tree [3]

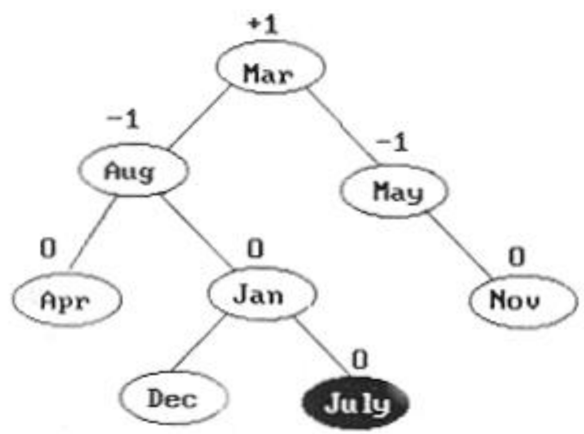


(f) Insert January



(g) Insert December

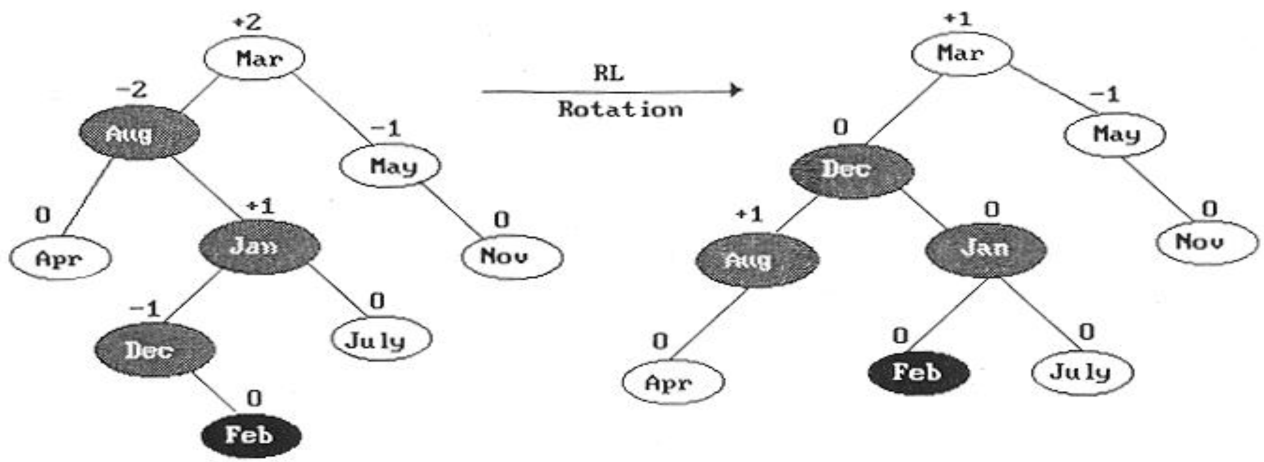
Insertion into an AVL tree [4]



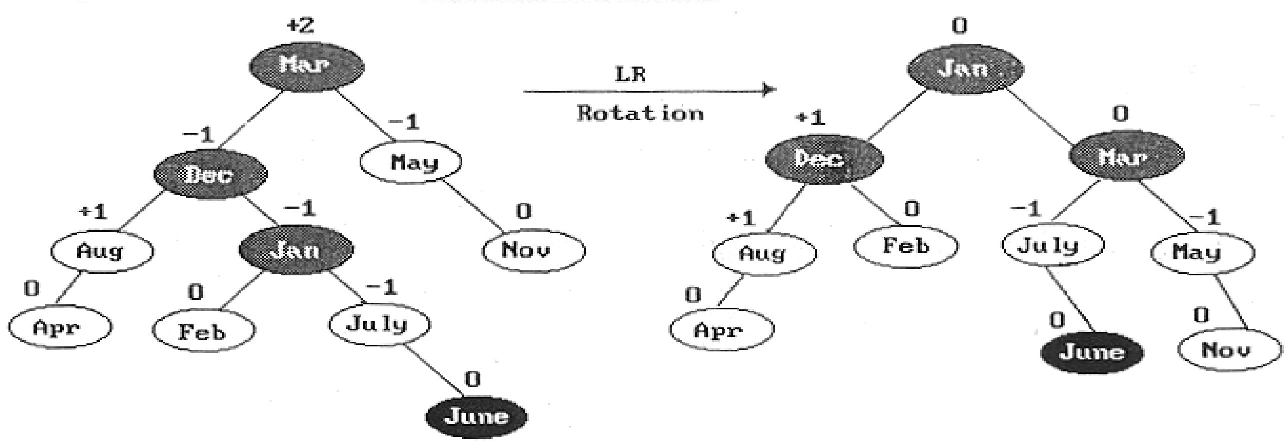
(h) Insert July

Figure 10.11 (continued): Insertion into an AVL tree

Insertion into an AVL tree [5]

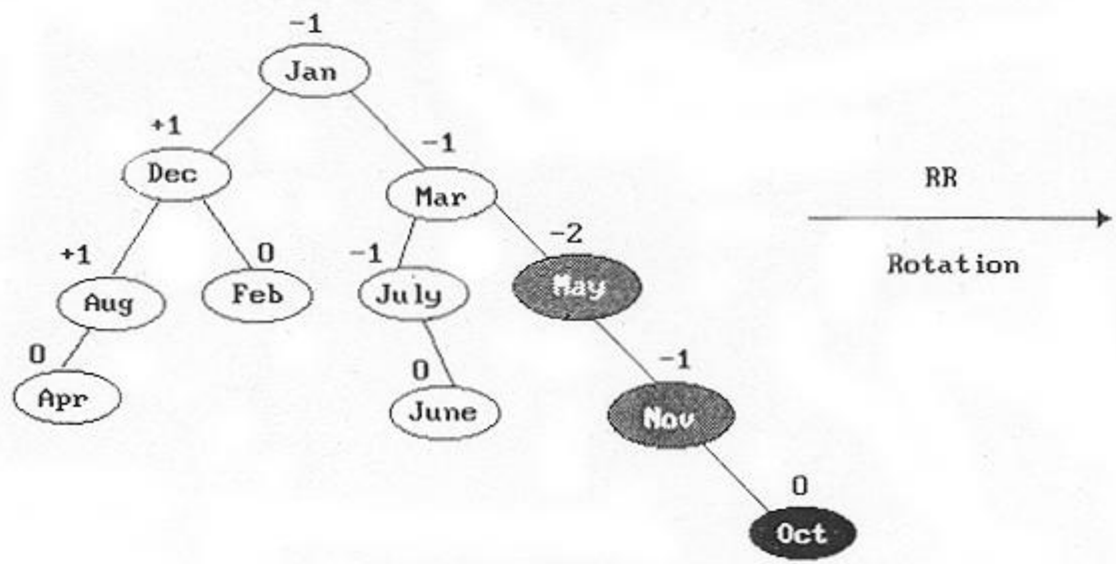


(i) Insert February



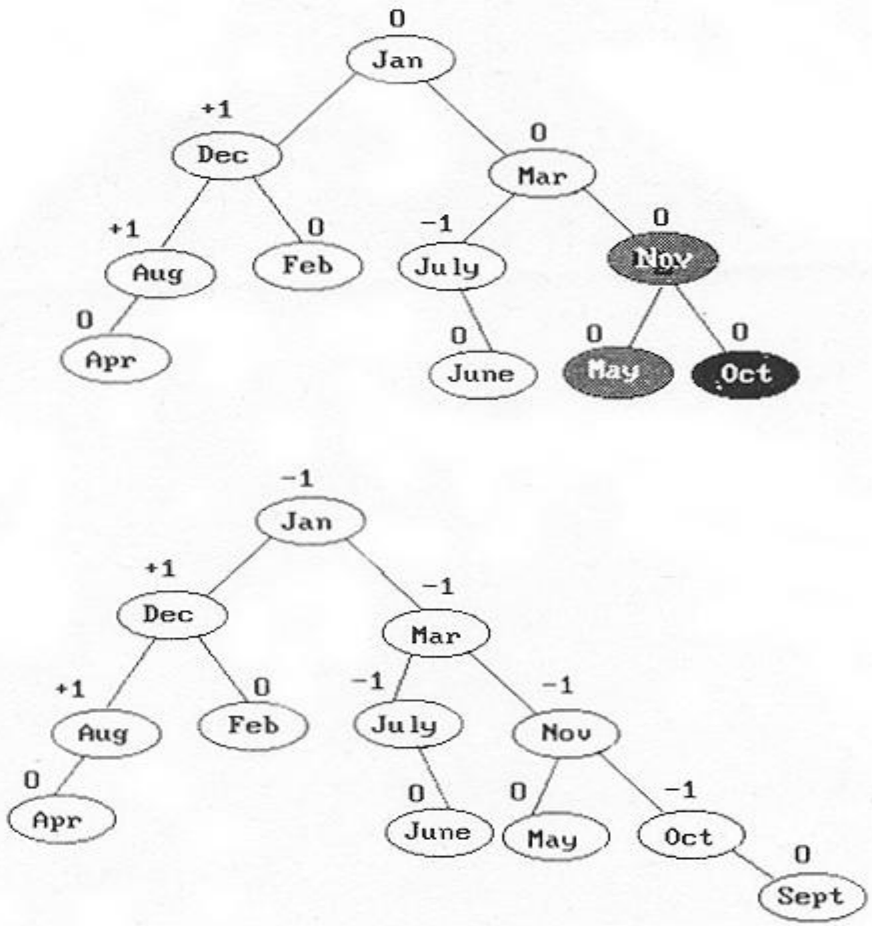
(j) Insert June

Insertion into an AVL tree [6]



(k) Insert October

Insertion into an AVL tree [7]



(1) Insert September

Empirical Study by Karlton et al.

- We assume random insertions.
- No Rebalancing : 0.5349
- Rebalancing with LL or RR : 0.2327
- Rebalancing with LR or RL : 0.2324

Comparison of various structures

operation	(Sorted) Sequential List	(Sorted) Linked List	AVL-Tree
Search for x	$O(\log n)$	$O(n)$	$O(\log n)$
Search for k-th item	$O(1)$	$O(k)$	$O(\log n)$
Remove x	$O(n)$	$O(1)$*	$O(\log n)$
Remove for k-th item	$O(n - k)$	$O(k)$	$O(\log n)$
Add x	$O(n)$	$O(1)$**	$O(\log n)$
Output in order	$O(n)$	$O(n)$	$O(n)$

* Doubly linked chain and position of x is known

** Position of x is known

End
of
Height-Balanced Binary Search Trees