Balance Tree

Yu-Tai Ching
Department of Computer Science
National Chiao Tung University

- Dictionary Problem,
- Array, Linked List, Tree.
- A “balance” tree supports $O(\log n)$ search time.
- A sequence of insertions or deletions may cause the tree out of balance,
- Can we maintain the balance of the search tree?

Months of the year example

- The Months of the year, JAN, FEB, MAR, APR, MAY, JUNE, JULY, AUG, SEPT, OCT, NOV, DEC.
- If they are inserted in this sequence, we get Figure 10.8
- If we can carefully construct a tree, we can obtain a balance tree as in Figure 10.9
- If the insertion sequence is sorted, we shall have a skew binary search tree.

AVL Tree

- In 1962, Adelson-Velskii and Landis introduced a binary tree structure that is balanced with respect to the heights of the subtrees.
- As a result, the height of the tree is bounded above by $O(\log n)$.
- The tree structure is called AVL tree.

An empty tree is height-balanced. If T is nonempty, let T_L and T_R be its left and right subtrees. Height of T_L and T_R are respectively h_L and h_R.
T is height balance iff

1. T_L and T_R are height balanced, and
2. $|h_L - h_R| \leq 1$.

Definition: Balance factor, $BF(T)$, of T in a binary tree is defined to be $h_L - h_R$. For any node in AVL tree, $BF(T) = -1, 0, 1$.

Insert the month into an AVL tree in this sequence, MAR, MAY, NOV, AUG, APR, JAN, DEC, JULY, FEB, JUNE, OCT, SEPT. Figure 10.11.

Rebalance Rotation

- If out of balance, rebalancing is carried out using 4 kinds of rotations, LL, RR, LR, and RL.
- LL and RR are symmetric. LR and RL are symmetric.
- Characterized by the nearest ancestor , A, of the inserted node, Y, whose balance factor becomes ± 2.

- LL: new node Y is inserted into the left subtree of the left subtree of A.
- LR: Y is inserted in the right subtree of the left subtree of A.
- RR: Y is inserted in the right subtree of the right subtree of A.
- RL: Y is inserted in the left subtree of the right subtree of A.
- Figure 10.12, 10.13.

Multiway Search Tree

Motivation:

- Cost for a search in AVL tree depends on the number of node visited.
- What if the data is stored in the disk? A comparison = a disk access.
- If we can cache the data in a node, we can cache a "large node"- *m*-way tree is proposed.

An m-way search tree is either empty or satisfies the following properties;

1. The root has at most m subtrees and has the following structure: $n, A_0, (E_1, A_1), (E_2, A_2), \ldots, (E_n, A_n)$ where $A_i, 0 \leq i \leq n < m$ are pointers to subtrees and the $E_i, 1 \leq i \leq n < m$ are elements. Each element E_i has a key $E_i.K$.
2. $E_i.K < E_{i+1}.K, 1 \leq i \leq n$.
3. Let $E_0.K = -\infty$ and $E_{n+1}.K = \infty$. All keys in the subtree A_i are less than $E_{i+1}.K$ and greater than $E_i.K, 0 \leq i \leq n$.
4. The subtree $A_i, 0 \leq i \leq n$ are also m-way search trees.

- The maximum number of nodes in a tree of degree m and height h,
- $\sum_{0 \leq i \leq h-1} m^i = (m^h - 1)/m - 1.$
- $n = m^h - 1$, then $h = \log_m n$.
- Searching in an m-way tree.

B-Tree

Definition: A B-Tree of order m is an m-way search tree that either is empty or satisfies the following properties,

1. The root node has at least two children.
2. All nodes other than the root node and external nodes has at least $\lceil m/2 \rceil$ children.
3. All external nodes are at the same level.

- $m = 3$, nodes of B-Tree have degree 2, and 3, a 2-3 tree.
- $m = 4$, nodes of R-Tree have degree 2, 3, and 4, a 2-3-4 tree.