Interlude for trees

Joyce Kilmer

Balanced Trees
- Splay
- Red-Black
- AVL
- B-tree

Minimax, see TicTac.java

Players alternate, one might be computer, one human (or two computer players)

Simple rules: win scores +10, loss scores -10, tie is zero
- X maximizes, O minimizes

Assume opponent plays smart
- What happens otherwise?

As game tree is explored is there redundant search?
- What can we do about this?

Balanced Trees and Complexity

A tree is height-balanced if
- Left and right subtrees are height-balanced
- Left and right heights differ by at most one

boolean isBalanced(Tree root)
if (root == null) return true;
return isBalanced(root.left) && isBalanced(root.right) &&
Math.abs(height(root.left) - height(root.right)) <= 1;

Rotations and balanced trees

Height-balanced trees
- For every node, left and right subtree heights differ by at most 1
- After insertion/deletion need to rebalance
- Every operation leaves tree in a balanced state: invariant property of tree

Find deepest node that's unbalanced then make sure:
- On path from root to inserted/deleted node
- Rebalance at this unbalanced point only
What is complexity?

- Assume trees are “balanced” in analyzing complexity
  - Roughly half the nodes in each subtree
  - Leads to easier analysis

- How to develop recurrence relation?
  - What is T(n)?
  - What other work is done?

- How to solve recurrence relation
  - Plug, expand, plug, expand, find pattern
  - A real proof requires induction to verify correctness

Balanced trees we won't study

- B-trees are used when data is both in memory and on disk
  - File systems, really large data sets
  - Rebalancing guarantees good performance both asymptotically and in practice. Differences between cache, memory, disk are important

- Splay trees rebalance during insertion and during search, nodes accessed often closer to root
  - Other nodes can move further from root, consequences?
    - Performance for some nodes gets better, for others …
  - No guarantee running time for a single operation, but guaranteed good performance for a sequence of operations, this is good amortized cost (ArrayList.add)

Balanced trees we will study

- Both kinds have worst-case O(log n) time for tree operations
- AVL (Adel’son-Velskii and Landis), 1962
  - Nodes are “height-balanced”, subtree heights differ by 1
  - Rebalancing requires per-node bookkeeping of height
    - http://people.ksp.sk/~kuko/bak/

- Red-black tree uses same rotations, but can rebalance in one pass, contrast to AVL tree
  - In AVL case, insert, calculate balance factors, rebalance
  - In Red-black tree can rebalance on the way down, code is more complex, but doable
  - Standard java.util.TreeMap/TreeSet use red-black

Rotation doLeft (see AVLSet.java)

- Why is this called doLeft?
  - N will no longer be root, new value in left.left subtree
  - Left child becomes new root
  - Unbalanced by two (not one!)
    - If left, left (or right, right)
      - doLeft (doRight)
    - Otherwise need two
      - doLeft/doRight
  - First to get to left, left
    - Or to right, right

```java
Node doLeft(Node root) {
    Node newRoot = root.left;
    root.left = newRoot.right;
    newRoot.right = root;
    return newRoot;
}
```
Rotation to rebalance

- Suppose we add a new node in right subtree of left child of root
  - Single rotation can’t fix
  - Need to rotate twice

First stage is shown at bottom
- Rotate blue node right
  - (its right child takes its place)
  - This is left child of unbalanced

```java
Node doRight(Node root)
{
    Node newRoot = root.right;
    root.right = newRoot.left;
    newRoot.left = root;
    return newRoot;
}
```

Double rotation complete

- Calculate where to rotate and what case, do the rotations

```java
Node doRight(Node root)
{
    Node newRoot = root.right;
    root.right = newRoot.left;
    newRoot.left = root;
    return newRoot;
}
```

AVL tree practice

- Insert into AVL tree:
  - 18 10 16 12 6 3 8 13 14
    - After adding 16: doLeftRight
doLeft

doRight
  - After 3, doLeft on 16

AVL practice: continued, and finished

- After adding 13, ok
- After adding 14, not ok
  - doRight at 12
Lynn Conway

See Wikipedia and lynnconway.com

- Joined Xerox Parc in 1973
  - Revolutionized VLSI design with Carver Mead

- Joined U. Michigan 1985
  - Professor and Dean, retired '98

- NAE '89, IEEE Pioneer '09

- Helped invent dynamic scheduling early '60s IBM

- Transgender, fired in '68