CS W3134: Data Structures in Java

Lecture #16: Quicksort
11/4/04
Janak J Parekh

Administrivia

- HW#2 returned today at end of class
  - Grades not up yet
- If you haven't started HW#4…
  - Brief discussion on alphabetic radix sort

Agenda

- Quicksort
  - Two parts: partition and recursion
- Begin trees as time permits

Quicksort: Partition

- Relies on concept of *partition*
  - A number s.t. two groups are formed: those smaller than the number, and those larger than the number
  - "Pivot"
  - Walk from both edges
    - If left is smaller than pivot, walk left
    - If right is larger than pivot, walk right
    - Otherwise, swap the two
    - What if we cross?
  - Last element is the pivot?
- Code? p. 338

Quicksort: Recursion

- Given pivot, we:
  - Partition the array in two;
  - Quicksort the left "half";
  - Quicksort the right "half".
- And recurse!
- That's it (p. 338)
  - Well, must be very, very careful
- Analysis?
  - *Usually* O(n log n), and in-memory
- But there are some problems…

Quicksort: Picking the pivot

- Imagine a reverse-sorted array
- How long does Quicksort take then? $O(n^2)$
- How can we fix this?
  - Pick pivot more intelligently
  - Two popular mechanisms:
    - Random
    - Median-of-three
- Also, inefficient for small arrays
  - Use insertion sort as a degenerate case…

Trees

- Linked Lists are generally connected to *one* other link
- What if we connect to multiple other links?
- A Tree is one generalization of a Linked List
- Key definition: no "cycles" amongst children
  - Graphs are more general
- Terminology
  - Node, Edge, Path, Root, Parent, Child, Leaf, Subtree, Level

Binary search trees

- What's a binary tree?
  - Two children, always
- Main concept:
  - Max(left subtree) must be < current node, min(right subtree) must be > current node
- Why?
  - Combines advantages of a linked list and an ordered array
  - Can insert fast and search fast
  - Unlimited growth
  - Relatively fast indexed access

Writing the Tree in Java

- "Node" class, with left and right children
- Data in node as well
- Very similar to Link
- Main "Tree" class that links to *root*, with find, insert, delete, etc. methods

Operations in a BST

- Search
  - Simple: walk left or right depending if < or > than current
  - If we hit the bottom, we can't find it
  - O(log N) time
- Insert
  - "Search", and then put in the appropriate place
  - Need a "current" and a "parent" pointer, similar to linked-list

Traversing the tree

- Unlike search, want to walk in an abstract order, sort of like arrays
- Three means of traversal; all recursive
  - Inorder
    - Visit left subtree
    - Visit node
    - Visit right subtree
  - Preorder
  - Postorder
  - The latter two have use in expressions (pg. 386)

Next time

- Continue trees