CS W3134: Data Structures in Java

Lecture #17: Quicksort, Trees
11/9/04
Janak J Parekh

Administrivia

- HW#4 due Thursday; last-minute questions?
  - Reminder: use the webboard to your advantage
- Important office hour changes
  - Akash will not hold office hours on Wednesday
  - Rachel will hold hers from 10:45am-12:45pm
  - Matthew will hold an extra hour from 6:30pm-7:30pm

Agenda

- Finish Quicksort
- Trees

Quicksort: Picking the pivot

- Imagine a reverse-sorted array
- How long does Quicksort take now? $O(n^2)$
- 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)

Other operations

- Min/max values
- Deleting a node
  - More complicated!
  - If no children, then nuke
  - One child
  - More than one child
    - Make one left, and go all the way right, or;
    - Make one right, and go all the way left
    - Take that node and put it at the deleted node's location
      - Move the right child of the moved node up one notch
    - Book uses latter convention

Tree complexity

- # of levels of a full tree is log N
  - Search, insert, delete is O(log N)
- What if it isn't full? Difficult analysis
  - Insert(1)
  - Insert(2)
  - …
  - In fact, this is the one downside of simple BST trees: easy to make unbalanced
  - There are alternatives; you can read chapter 9 (optional)

Trees as arrays

- Array[0] is the root
- 2*index+1 is the left child
- 2*index+2 is the right child
- Parent of a node is, correspondingly, (index-1)/2
- Actually works surprisingly well, but…
  - No unlimited growth
  - Inefficient use of memory
  - Deletes are slow

Expression trees

- Operators are root and intermediate nodes, operands are leaf nodes
- To create
  - Start with postfix expression and a stack
  - Operand: form unit tree with value and push onto the stack
  - Operator: pop two things off of stack, combine "by" operator, push result on stack
- When done, one element on stack
- What does inorder, preorder, postorder mean?

Next time

- Finish trees
- Start hashing