Slide 1

Administrivia

- Reminder — Homework 6 code due today. Comments on design (homeworks 4, 5, and 6) mailed. Grades for homework 4 in work.
- Quicksort and mergesort added to sample programs page, under "sorting and searching". Also several possibly useful examples under "GUIs".

Slide 2

Minute Essay From Last Lecture

- We said quicksort is usually faster than, say, bubble sort, but there are unusual cases in which it's not. One is if the data is already sorted (and the pivot is chosen as the first element). What's another case in which quicksort would not be especially quick?

  Reverse order!

CSCI 1321 November 13, 2003

Slide 3

Trees — Mathematical Definition

- One definition —
  - Set of nodes, one called root.
  - Set of edges (directed connections between nodes).
  - Root has no incoming edges; all other nodes have exactly one (from parent).
  - Each node can have 0 or more outgoing edges (to children — if none, leaf node).
- Another, recursive definition — tree is one node connected by edges to 0 or more subtrees.
- This is a general tree — e.g., to represent hierarchy such as filesystem.

Slide 4

Implementing Trees

- Define `Node` data structure, analogous to linked list, with reference to data and references to children (linked list or `Vector` or . . .).
- Easier if number of children is limited to two, and this turns out to be sufficiently useful in practice — "binary tree". Then `Node` consists of pointers to data and left and right subtrees.

Slide 5

Tree Traversals

- For linked lists we defined a way to visit all elements — "iterator". Is there something analogous for trees?
- Well — three orders that are easy to define and implement:
  - Preorder — root first.
  - Postorder — root last.
  - Inorder — leftmost subtree first, then root, then remaining subtrees. (Admittedly a little weird for non-binary trees.)
- Sketch some code for at least one of these.

Slide 6

Sorted Binary Trees (Binary Search Trees)

- Key property — everything in the left subtree is smaller than the root, and everything in the right is bigger.
- Why is this useful? If you want a data structure to hold a collection that will be searched frequently, what are the choices? and how fast is each to search? to modify (insert/remove)? Compare approximate times for arrays (sorted and unsorted), linked lists (sorted and unsorted), sorted binary tree.
- Sketch some code for `add` and `find`. `remove` next time.

Slide 7

Minute Essay

- None — Quiz 5.