Slide 1

Administrivia

- (None.)

Slide 2

Graphs — Recap / Why Interesting

- Idea is to represent relationships among set of items in a way that can be dealt with mathematically. Computer-friendly representations extend that.
- Chapter 6 discusses many interesting “graph algorithms”, most/all with real-world uses. Examples on next slides.

Slide 3

Graph Algorithms

- Given two nodes in a weighted graph, find the "shortest" (minimum-weight) path from one to another. Could use in routing network traffic so it uses paths with least load.
- Find "minimum spanning tree" for a weighted graph — minimum-weight set of edges that connect all nodes.
- "Euler path problem" — is there a path through a graph that uses each edge exactly once? Generalization of a puzzle involving bridges in a German city (picture in text). Euler's solution — path exists if graph has no more than two "odd" nodes.

Slide 4

Graph Algorithms, Continued

- "Travelling salesperson[a] problem" — in a complete weighted graph, of the many closed paths that visit each node exactly once ("Hamiltonian circuits"), which has least weight? Brute-force solution is $O(n!)$, best solution is apparently $O(n^2 2^n)$.

[a] "salesman" in the literature, alas

Slide 5

Trees — Overview

- Many of you probably know trees from POP II. In math terms, we can say a tree is a kind of graph — acyclic, connected, one node designated “root”.
- Can be used to represent any kind of hierarchy, e.g.:
  - Table of contents of a book.
  - Hierarchical help system.
  - Arithmetic expression.

Slide 6

Trees — Terminology

- Some terminology should be familiar: root, subtree, parent, children, root of subtree.
- Other terms:
  - Depth of node (distance from root), height of tree (maximum depth).
  - Binary tree — at most two children per node.
  - Full binary tree.
  - Complete binary tree.

Slide 7

Trees — Recursive Definition

- Tree is either
  - A single node, or
  - The tree formed by combining a root r with (disjoint) subtrees t_1 through t_n.

Slide 8

Computer-Friendly Representation of Trees

- Can of course use any representation that works for general graphs.
- Can also use pointer-based representation — simpler for binary trees, but possible for general trees as well.
- Can also use array representation for binary trees: Number the nodes from 1 to N, and make a 2-by-N array for left/right children.
- Can use an even more compact representation for complete binary trees, using a single array. (Until recently we did this in PAD II (1321). Now?)

Slide 9

Tree Traversals

- For linear data structures (lists, arrays, etc.), basically only one reasonable way to "walk through" the structure to visit each element.
- For trees, there are three "reasonable" ways:
  - Preorder traversal (root first, then subtrees).
  - In-order traversal (leftmost subtree first, then root, then rest).
  - Postorder traversal (subtrees first, then root)
- Functions to perform any of these (e.g., and print each node as it is visited) are almost trivial to write recursively, much more difficult without recursion.

Slide 10

Trees — Special Types

- Special types (familiar from POP II?):
  - Sorted binary tree.
  - Heap.

Slide 11

Trees and Recursion/Induction

- Easy to write other recursive algorithms to operate on trees — e.g., function to find height of tree.
- If we use the recursive definition of a tree, we can prove things about trees using induction. Example from textbook — prove that number of arcs is one less than number of nodes.

Slide 12

Minute Essay

- None — quiz.