Slide 1

Administrivia

- Reminder: Homework 6 code due today. (Accepted without penalty through next Tuesday.) Homework 7 due dates on Web.

Slide 2

Binary Search Trees, Continued

- Last time we started talking about "binary search trees" — binary trees in which all values in a node's left subtree are smaller than the value in the node, and all values to the right are larger. We sketched code for adding and finding elements.
- How to remove? more difficult to code, but we can draw pictures.

Slide 3

Priority Queues, Revisited

- Several data structures we could use to implement priority queue ADT:
  - Unsorted linked list.
  - Sorted linked list.
  - Sorted binary tree.

  Compare how much work to add/remove if N elements. Can we do better? Maybe!

Slide 4

Heaps

- Heap is another tree-based data structure, with two properties:
  - A node is always "bigger than" both its children.
  - Tree is "complete".
- For a priority queue, we want to retrieve the "biggest" thing (for game problem, smallest update time). Does this seem useful?
- Note also that we can store a complete binary tree in an array.
- How to insert and remove? Compare running times.

Slide 5

Homework 7

- Writeup should be complete, but a short overview:
  - Objective is to write an alternate implementation for priority queue ADT and compare its performance to that of first implementation.
  - To compare performance, need to (1) add something to code to measure execution time, and (2) increase work being done by priority queue to the point where performance differences will show up.
- You can find code for a heap-based priority queue lots of places, but you will probably learn more if you write your own.
- Be sure to save a copy of your existing code before doing this, because you shouldn't include most of these changes in what you turn in as a "final" game (Homework 8).

Slide 6

Minute Essay

- None — quiz.