Slide 1

Administrivia

- Reminder: Homework 6 code due today.
- Homework 7 due dates week after next.

Slide 2

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 3

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 4

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).

Minute Essay

- Sketch what a heap of integers (ordered to put smallest values at the top) would look like after the following operations:

  Insert 5, 4, -1, 10, 6, 20.

  Remove (smallest).

Slide 5