Slide 1

Administrivia

- Reminder: Homework 7 design due Tuesday.

Slide 2

Sorted Binary Trees (Binary Search Trees) — Recap

- 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? this one may be better than the others.
- Sample programs page has example code (and also code for a recursive implementation of a sorted linked list).

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?
- How to insert and remove? Compare (big-O) running times.
- Note also that we can store a complete binary tree in an array.

Slide 5

Heaps — Inserting and Removing Elements

- Basic idea is that we start an insert operation, or a remove-the-largest operation, in a way that maintains the property that the tree is complete. (So, for inserting we add to the lowest level of the tree at the right, and for removing the largest element we replace the top element with the rightmost node at the lowest level.)
- But that breaks the other property (each node larger than its children), so . . .
  - For inserting, we "walk" from the new node up the tree, exchanging with the parent node if they're not in the right order (as sketched in class).
  - For removing, we "walk" from the replaced root node down the tree, exchanging with the largest child if at least one child is larger (also as sketched in class).

Slide 6

Heaps — Storing in an Array

- How to store a complete tree in an array? Store "levels" from top to bottom, each level left to right, as sketched in class.
- How to navigate up and down the tree? figure out how to map from a node's index to the indices of its parent and left and right children . . .
  - Left child of node n is $2n + 1$, right child is $2n + 2$.
  - Parent of node n is $(n - 1)/2$, where the division is integer division (which discards the remainder).

Homework 7

- Homework 7 asks you to do another implementation of the game priority queue and compare its performance with the first one.
- The mechanism for doing this involves using command-line arguments. So, a short discussion . . .

Slide 7

Command-Line Arguments

- Many mechanisms for starting programs provide a way of passing them information without using files or standard input — "command-line arguments". Example — when you type at the command line

  ```
  ls -l myfile
  ```

  `-l` and `myfile` are passed to the `ls` in this way.
- C programs can receive command-line arguments by declaring `main` as

  ```
  int main(int argc, char *argv[])
  ```

  or equivalent, where `argc` is the number of arguments and `argv` is an array of C-style strings. By convention the zero-th argument is something identifying the program (e.g., its name). So in the `ls` example above, there would be three arguments . . .

Slide 8

Command-Line Arguments, Continued

Slide 9

- Java `main` methods also receive command-line arguments via arguments passed to `main`. `main` must always be declared with an argument of type `String[]`, which is a Java array containing the arguments. A Java equivalent of `ls` would get only two arguments for the example of the previous slide.
- Eclipse unfortunately doesn't make it that easy to invoke programs with command-line arguments that vary from execution to execution, but it's possible. An alternative is to run the program from the command line:

```
java MainClass arg1 arg2
```

  or for your game something like

```
java -classpath bin:PAD2.jar MainClass arg1
arg2
```

  (Replace ":" with ";" on Windows.)

Minute Essay

Slide 10

- Sketch what a sorted binary tree of integers would look like after adding the following:

  5, 4, -1, 10, 6, 20.
- Now sketch what a heap of integers (ordered to put smallest values at the top) would look like after adding the same values.

Slide 11

Minute Essay Answer

- The BST:

```
      5
     / \
    4   10
   /   /  \
 -1   6    20
```

- The heap:

```
      -1
     /    \
    5      4
   / \    /
 10   6  20
```