CSCI 136
Data Structures & Advanced Programming

Jeannie Albrecht

Lecture 27

April 25, 2014

Administrative Details

- Darwin lab
  - Part 1 due Monday
  - Part 2 due next Monday
- Midterm 2
  - Wednesday @ 1:00 in Wege
  - Covers Ch 7, 8, 10-13, Closed book
  - No class on Wednesday
- Review session Tuesday 9:30-10:30 in TCL 202
- Info session Monday night at 9:00?

2

Last Time

- Wrapped up discussion on priority queues and heaps
- Discussed heapsort

3

Today's Outline

- Finish discussing heapsort
- Start discussing binary search trees (Ch 14)

4

Why Heapsort?

- Heapsort is slower than Quicksort in general
- Any benefits to heapsort?
  - *Guaranteed* O(n log n) runtime
  - Constant space overhead
- Works well on mostly sorted data, unlike quicksort
- Good for incremental sorting

5

Tree Wrapup

- General Binary Trees
  - Express hierarchical relationships
  - Ordering is based on some external notion
    - i.e., ancestry, game boards, decisions, etc.
- Heap
  - Partially ordered (complete) binary tree based on priorities
  - Node invariants: parent has higher priority than both children

6

Binary Search Tree

- Binary search trees maintain a *total* ordering among elements
- Definition: A BST is either:
  - Empty
  - A tree where root value is greater than or equal to all values in left subtree, and less than or equal to all values in right subtree; left and right subtrees are also BSTs
- Examples

7

BST Operations

- `add(Object item)`
- `contains(Object item)`
- `get(Object item)`
- `remove(Object item)`
- Runtime of above operations?
  - All O(log n)!
- `iterator()`
- Questions: How can we get a *sorted* list of all elements in BST?
  - In-order iterator

8

Example Usage: Dictionary

- Create a BST of ComparableAssociations
  - Order BST by key
  - Two objects are equal if keys are equal
- What would add(word, def) and lookup(word) look like using a BST?
- Different dictionary implementations in CS136

9

Tree Sort

- Can we sort data using a BST?
  - Yes!
- Runtime?
  - To build a tree with n elements, we do n insertions: O(n log n) (as long as tree stays as short as possible…)
  - In order traversal: O(n)
  - Total runtime: O(n log n)
- Advanced sorting comparisons

10

Implementation

- How would be implement a BST?

11