CS W3134: Data Structures in Java

Lecture #13: Recursion

10/19/04

Janak J Parekh

Administrivia

- HW#3 out
  - Noticeably smaller than the first two, but linked lists are a little tricky.
- HW#1 returned today
  - Solutions for HW#1 and HW#2 will also be up today
- Midterm review for office hours
  - Will occur here until 1pm
- Today's lecture *not* on midterm

Agenda

- Recursion

How to calculate…

- What's the sequence 1, 3, 6, 10, 15, 21, 28, 36…
  - *Triangle* numbers
  - How to do as loop?
  - How to do *as addition on previous result*?
    - Recursion!

A better example

- Simpler, you say?
- What's the sequence 1, 1, 2, 3, 5, 8, …
  - Easy to define in terms of recursion, right?
  - How to iterate over this?
  - In other words, there are problems that are more intuitive recursively

Formalizing Recursion

- Recursive algorithms have the following properties
  - They call themselves
  - They call themselves to solve a smaller problem, and then work with the result
  - There's a *stopping* condition, e.g., a call which is simple enough to solve explicitly
  - Generally avoid explicit loops

Recursion vs. Iteration

- Recursion is, *generally*:
  - A bit less intuitive at first…
  - Simpler to implement / elegant
  - Less efficient
- But… conceptually simpler

Some more examples

- FindMax
- Recursive binary search (p. 268)
- Divide-and-conquer approach
  - Take a big problem, split into smaller problems, solve separately
  - Very powerful methodology, works well with recursion
  - Usually two recursive calls

Method overloading (for HW#3)

- OO concept useful for recursion, but not only
- You can have multiple methods with the same name
  - As long as parameters differ
- For recursive algorithms, often will have a "bootstrap" method
- Let's look at the FindMax example…

Towers of Hanoi

- Three pegs
- Disks all on one peg
- Want to move it to third peg
- Second peg is a "work peg"
- Can't move a disk until all smaller disks have been moved
- Basic intuition
  - Move the top disks from start to intermediate
  - Move the largest disk to destination
  - Move top disks from intermediate to destination

Hanoi (II)

- Three steps:
  - First, move pile from "from" to "inter", using "to" as a work peg
  - Then, move disk from "from" to "to"
  - Then, move remainder of pile from "inter" to "to", using "from" as a work peg
- This works because we don't have to put things consecutively, just that larger disks must go on top of smaller disks
- Page 278 for code

Mergesort

- Classic recursive algorithm
- Split arrays in half, sort each half, and then merge them together
  - "Divide and conquer"
- Sort is the "recursive" call
- Let's do it intuitively first
- Now, psuedocode…

Mergesort (II)

- Key aspect of code on page 287
- The header of the method contains enough information to perform the recursive call
  - In this case, partition information
- Efficiency?
  - Partition: O(1)
  - Merge: O(n)
  - How many times each have to be done? O(log n)
  - Ergo, O(n*log n)
- Disadvantage: lots of memory required

Next time…

- Finish up mergesort
- Two more complex sorts
  - Radix sort
  - Quicksort