Administrivia

Slide 1

- Reminder: Homework 3 due today.

Recursion and Recursive Definitions — Review/Recap

Slide 2

- Idea of recursion closely related to idea of induction — “build on previous smaller cases”.
- First look at recursive definitions. To define something recursively:
  - Define one or more “base cases”.
  - Define remaining cases in terms of other (“smaller”) cases.
- Last time we looked at recursive definitions of sequences and sets. (Notice revision to slide about sets.)

Slide 3

Recursive Definitions — Operations

- Example — factorial.
- Example — multiplication of non-negative integers, defined in terms of addition.
- Example — (integer) division of a non-negative integer by a positive integer, defined in terms of subtraction.

Slide 4

Recursive Algorithms

- Recursive definitions of sequences or operations often can be turned into recursive algorithms with little effort.
- Examples — function to compute n-th Fibonacci number, function to do division by repeated subtraction.
- Efficiency considerations:
  - In terms of computer time/memory usage, recursion is almost always worse than iteration — but not always, and sometimes not much worse.
  - In terms of human effort to get program running correctly, recursion may be much better.
- Examples in text — selection sort and binary search. Quicksort and mergesort are other good ones.

Slide 5

Reasoning About Recursive Algorithms

- A recursive algorithm "works" if:
  - It works for the base case(s).
  - For other cases, it works *assuming* the recursive calls work.
  - The recursion eventually stops — recursive calls are always "smaller", and eventually reduce to base cases.
- We could formalize this as a proof by induction.

Slide 6

Recurrence Relations

- Recall the silly example of defining a sequence recursively:

$$\begin{aligned} S(1) &= 1 \\ S(n) &= S(n-1) \times 10, \quad \text{for } n > 1 \end{aligned}$$

  Expanding out some terms, it seems fairly obvious that an equivalent definition would be $S(n) = 10^{n-1}$ — a "closed-form solution" to the recurrence relation given in the second line of the definition.
- We'll look at various ways to get from a recursive definition to a closed-form one, because the latter are easier to compute, but sometimes it will be much easier to write down the definition recursively.

Slide 7

Solving Recurrence Relations, Continued

- For the silly example

$$\begin{aligned} S(1) &= 1 \\ S(n) &= S(n-1) \times 10, \quad \text{for } n > 1 \end{aligned}$$

  we guessed a solution of $S(n) = 10^{n-1}$. Can we verify that this is the same as the recursive definition? yes, via a proof by induction . . .

- Call this method “expand, guess, verify”.
- Try another example — section 2.5 problem 3.

Slide 8

Solving Recurrence Relations, Continued

- Is there another way? In general, probably not, but there are some formulas for some frequently-occurring special cases.
- One is “first-order linear” recurrence relations with constant coefficients. If

$$S(n) = cS(n-1) + g(n)$$

  then we can show (see textbook for derivation) that

$$S(n) = c^{n-1}S(1) + \sum_{i=2}^{n}(c^{n-i}g(i))$$

- Apply this to the two problems we did earlier — we should get the same results.

Slide 9

Minute Essay

- Consider the following recursive definition of a sequence:

$$\begin{aligned} S(1) &= 1 \\ S(n) &= 10S(n-1)+1, \quad \text{for } n > 1 \end{aligned}$$

What are $S(1), S(2), \ldots S(5)$?

Slide 10

Minute Essay Answer

- The first few terms:

$S(1) = 1$

$S(2) = 11$

$S(3) = 111$

$S(4) = 1111$

$S(5) = 11111$