Slide 1

Administrivia

- Reminder: Quiz 3 Wednesday.
- Homework 4 on Web. Due next Monday.

Slide 2

Recursion and Recursive Definitions

- 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.

Slide 3

Recursive Definitions — Sequences

- A silly example:

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

  Try writing down some terms.

- Another example:

$$\begin{aligned} S(1) &= 1 \\ S(2) &= 1 \\ S(n) &= S(n-2) + S(n-1), \quad \text{for } n > 2 \end{aligned}$$

  Try writing down some terms. Anyone recognize this one?

Slide 4

Recursive Definitions — Sets

- Example — could define the set of “integer arithmetic expressions” like this:
  - Integers are expressions.
  - If E and F are integer arithmetic expressions, so are $(E + F)$, $(E - F)$, $(E \times F)$, and (E/F).

  Examples?

  Notice that this allows us to generate only “sensible” expressions. Notice also that it’s a bit more restrictive than we might like.

- We could write similar definitions for the wffs of propositional and predicate logic.

Recursive Definitions — Operations

Slide 5

- 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.

Recursive Algorithms

Slide 6

- 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.

Slide 7

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 8

Recursive Algorithms, More Examples

- Two good examples in text — selection sort and binary search.
- Another example — "quicksort".

```
// pre:  i, j are valid indices for L
// post:  L(i) through L(j) are "sorted"
qsort(list L, index i, index j)
        if (i >= j)
                return
        else
                elem pivot = L(i)
                // rearrange L(i+1) through L(j) s.t.:
                //      L(i) .. L(m-1) <= pivot
                //      L(m) = pivot
                //      L(m+1) .. L(j) >= pivot
                index m = split(pivot, L, i, j)
                qsort(L, i, m-1)
                qsort(L, m+1, j)
end qsort
```

(Why does this work?)

Minute Essay

Slide 9

- 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 $S(1), S(2), \ldots S(5)$?