Recursion

Mason Vail, Boise State University Computer Science

Examples from Java Foundations, 3rd Ed., by Lewis, DePasquale, and Chase

Recursive Definition

24, 88, 40, 37

A **LIST** is a:

number

or

number comma LIST

Infinite Recursion

Base Case: Every recursive definition must have a non-recursive part - a stopping point - a simple case for which the answer is known.

Without a base case, there’s no way to end a recursion, creating a similar problem to infinite loops.

Recursion in Math

Factorial: N! for any positive N is the product of all integers from 1 to N inclusive.

N! can be expressed recursively:

$$\mathbf{0! = 1}$$

$$\mathbf{N! = N * (N-1)!}$$

Recursive Methods

A method can call itself.

The code must handle base and recursive cases to avoid infinite recursion. Each recursive call should be a simpler version of the problem, getting closer to a base case.

The call stack stores all of the partial solutions until a base case is recognized and recursive calls stop. As methods return, the total solution is assembled.

Recursive Programming Example: Summation

There are better ways to compute the sum of all integers from 1 to N, but a recursive definition would be:

sum(1) = 1

sum(N) = N + sum(N-1)

Recursive Programming Example: Summation

```
public int sum(int num) {
  int result;
  if (num == 1)
    result = 1;
  else
    result = num + sum(num-1);
  return result;
}
```

General Recursive Algorithm

- If the problem can be solved for the current version of the problem (the base case):
  - Solve it.
- Else:
  - Recursively apply the algorithm to one or more smaller versions of the problem.
  - Combine the solutions to the smaller problems to get the solution to the original.

Indirect Recursion

A method calling itself is direct recursion, but a sequence of methods that eventually calls the first can result in indirect recursion.

This is often difficult to understand and debug.

A higher-level recursive method with subroutine method calls can avoid this confusion.

Recursion vs. Iteration

Every problem that can be solved recursively can also be solved iteratively.

Recursive solutions have the overhead of multiple method invocations and may be less efficient (though not necessarily with a change in order).

However, some problems are more easily and elegantly expressed recursively.

Example Problems with Elegant Recursive Solutions

- Maze traversal (depth-first with backtracking)
- Towers of Hanoi

Recursion

Mason Vail, Boise State University Computer Science

Examples from Java Foundations, 3rd Ed., by Lewis, DePasquale, and Chase