Recursive Functions

ACSL Material

What is recursion?

- A definition that defines an object in terms of itself is said to be *recursive*.
- In computer science, recursion refers to a function or subroutine that calls itself, and it is a fundamental paradigm in programming.
- A recursive program is used for solving problems that can be broken down into sub-problems of the same type, doing so until the problem is easy enough to solve directly.

ACSL Definitions

- Indirect recursion: when a function calls another function which eventually calls the original function.
  - Ex: Function A calls function B, then while B is running, A is called again (either by B or a function that B calls)
- Single recursion: recursion with a single reference to itself.
  - Ex: factorials
- Multiple recursion: when a recursive function has multiple self references.
  - Ex: fibonacci sequence
- Infinite recursion: a recursive function that never returns because it keeps calling itself.
  - Ex: you mix up a < with a > and your while loop crashes your IDE ;)

Important ACSL Info

- This ACSL category focuses on mathematical recursive functions rather than programming algorithms.
    - While many mathematical functions can be done iteratively more efficiently than they can be done recursively, many algorithms in computer science must be written recursively.
    - You will typically be asked to evaluate a recursive function for some specific value. It's important that you are careful and neat.
- Technique: Work your way down the recursive calls until there are no more calls, and then work your way back up.

Sample Problem 1

- Find g(11) given the following

$$g(x) = \begin{cases} g(x-3)+1 & \text{if } x > 0 \\ 3x & \text{otherwise} \end{cases}$$

Top Down Approach

$g(11) = g(8) + 1$

$g(8) = g(5) + 1$

$g(5) = g(2) + 1$

$g(2) = g(-1) + 1$

$g(-1) = -3$

$g(11) = 0 + 1 = 1$

$g(8) = -1 + 1 = 0$

$g(5) = -2 + 1 = -1$

$g(2) = -3 + 1 = -2$

Sample Problem 2

- Find f(12,6) given the following definition of f:

$$f(x,y) = \begin{cases} f(x-y, y-1) + 2 & \text{when } x > y \\ x + y & \text{otherwise} \end{cases}$$

$\downarrow$

$f(12,6) = f(6,5) + 2$

$f(6,5) = f(1,4) + 2$

$f(1,4) = 1 + 4 = 5$

$\uparrow$

$f(12,6) = 7 + 2 = 9$

$f(6,5) = 5 + 2 = 7$

ACSL Word Problem

Consider the following recursive algorithm for painting a square:

1. Given a square.
2. If the length of a side is less than 2 feet, then stop.
3. Divide the square into 4 equal size squares (i.e., draw a “plus” sign inside the square).
4. Paint one of these 4 small squares.
5. Repeat this procedure (start at step 1) for each of the 3 unpainted squares.

If this algorithm is applied to a square with a side of 16 feet (having a total area of 256 sq. feet), how many square feet will be painted?

ACSL Word Problem Solution

- In the first pass, we get four squares of side 8. One is painted, three are unpainted.
- Next, we have 3*4 squares of side 4. Three are painted (area=$3*4^2$), nine are not.
- Next, we have 9*4 squares of side 2. Nine are painted (area = $9*2^2$), 27 are not.
- Finally, we have 27*4 squares of side 1. Twenty-seven are painted.
- Therefore, the total painted is $1*8^2 + 3*4^2 + 9*2^2 + 27*1^2 = 175$.

Components of a Recursive Func

This is not ACSL material, so feel free to skip, but the structure can help you understand how recursion works.

There are three important rules of thumb in developing recursive programs:

- The recursion has a *base case*—we always include a conditional statement as the first statement in the program that has a `return`.
- Recursive calls must address subproblems that are *smaller* in some sense, so that recursive calls converge to the base case.
- Recursive calls should not address subproblems that *overlap*.

Links

- http://www.categories.acsl.org/wiki/index.php?title=Recursive_Functions
- Algorithms by Robert Sedgewick and Kevin Wayne