Recursion notes

Prof Bill - Jan 2020

Reading:

- ➔ Sedgewick Java 2.3 Recursion, introcs.cs.princeton.edu/java/23recursion
- ➔ Recursion animation, www.cs.usfca.edu/~galles/visualization/Algorithms.html

Definitions

Java definition - a **recursive method** makes one or more calls to itself

/* CSC 220 - how does your computer manage multiple calls of the same method? */

Recursive solution has two parts:

- **base case** - fixed value, end of recursion
- **recursive case** - function defined in terms of itself

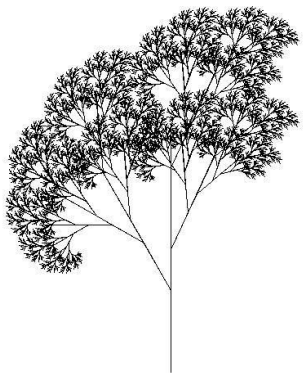
Factorial example:; iterative and recursive solutions; en.wikipedia.org/wiki/Factorial

```
n! = n * (n-1) * (n-2) * … 2 * 1

n! = n * (n-1)!   // recursive case

0! = 1   // base case
```

recursive data structure - self-referential data structures (like linked lists or trees); these structures can often be intuitively accessed with recursive code

More generally - **Recursion** occurs when a thing is defined in terms of itself or of its type, en.wikipedia.org/wiki/Recursion

“Tree (below) created using the Logo programming language and relying heavily on recursion. Each branch can be seen as a smaller version of a tree.”
- en.wikipedia.org/wiki/Recursion_(computer_science)



Recursive structures are found in mathematics and nature. These structures rely on smaller instances of the same structure, ala fractals like the Mandelbrot set. There are lots of recursive structures in nature: sunflowers, seashells, ferns, etc.

- ➢ Mandelbrot set on Wikipedia, en.wikipedia.org/wiki/Mandelbrot_set
- ➢ Fractals are recursive objects, en.wikipedia.org/wiki/Fractal
- ➢ Good stuff on fractals, www.nilsdougan.com/?page=writings/onfractals
- ➢ In maths, recurrence relations are recursive equations, en.wikipedia.org/wiki/Recurrence_relation

Designing recursive algorithms

Key steps:

1. Identify base case (end of recursion) and
2. recursive case (make problem smaller)

So, solution steps are: identify base case (end of recursion), and then recursive case (making problem successively smaller). Recursion must get smaller or it blows up.

Other examples (identify the base case, recursive case!):

- Recursive multiplication: n*m = n + (n*(m-1))
- Recursive power: x^y = x * x^(y-1)

- Fibonacci:

```
F(N) = F(N-1) + F(N-2)
F1 = 1
F0 = 0
```

Recursion plus and minus:

--- Minus - recursion is more expensive than an iterative solution.

+++ Plus - elegance and simplicity of recursion solution may be worth it.

/* CSC 220 (again): Why is recursion so much more expensive than iteration? */

tail recursion: when the recursive function call(s) is at the end of the method

Tail recursion is important because 1) it's pretty common, and 2) it's usually easy to convert tail recursion to iteration. Motivation: iteration is faster than recursion! Fibonacci is a good example of this.

Examples

Summation example: recursive, iterative, and equation, en.wikipedia.org/wiki/1_%2B_2_%2B_3_%2B_4_%2B_%E2%8B%AF

Binary search tree - recursive and iterative solutions to search and traversal, en.wikipedia.org/wiki/Binary_search_tree

Fibonacci - $F_n = F_{n-1} + F_{n-2}$; we can program recursive or iterative solutions; great comparison here, dev.to/khalilsaboor/fibonacci-recursion-vs-iteration--474l

Binary search is O(log n).
Pseudocode:

```
// array must be sorted!
int binarySearch( array, value, first, last)
    if last < first
        return not found
    mid := (first + last) / 2
    if a[mid] = value
        return mid
    if value < a[mid]
        return binarySearch(a, value, first, mid-1)
    else
        return binarySearch(a, value, mid+1, last)
```

Binary search too, www.codecodex.com/wiki/Binary_search#Pseudocode

```
function binarySearch(a, value, left, right)
    while left ≤ right
        mid := floor((right-left)/2)+left
        if a[mid] = value
            return mid
        if value < a[mid]
            right := mid-1
        else
            left  := mid+1
    return not found
```

thanks… yow, bill

PS - Class field trip… we all get Mandelbrot set tattoos, www.askideas.com/10-mandelbrot-tattoo-designs