Emmanuel Sanchez

Efficiency of Recursion

**Think Recursively**

Recursion is a problem solving approach that can generate simple solutions. Simple to solutions to problems that in other ways might be more difficult to solve.

Recursion can split a problem into one or more simpler versions of itself.

**But is it Effective?**

A perfect an easy example to think recursively is a series of nested boxes (boxes inside boxes).

Ex.

1. if there is only one Box
2. do what is needed for the box

else

1. do what is needed for the outer box
2. do the same process for the nested boxes.

**General Approach**

If the problem is **Small Enough.**

Solve it **Directly**

**else**

Break it into one or more **Smaller Sub-problems**

Solve each sub-problem **recursively**

**Combine** results into solution to whole problem

**Recursion vs Iteration**

Recursion and iteration are similar.

**Iteration:** Loop repetition test determines whether to exit

**Recursion:** Condition test for a base case. Can always write iterative solutions to a problem solve recursively, but recursive code is often simpler than iterative, thus simpler to write, read and debug.

**Efficiency of Recursion**

1. Recursive method is often slower than iterative, why?

* Overhead for loop repetition is smaller
* Overhead for call and return

1. If easier to develop algorithm-using recursion, then code it as recursive method. Don’t optimize prematurely.

**Recursive Definitions using Fibonacci Series**

Definition of fib­I, for integer i > 0:

fib1= 1

fib2= 1

fibn= fibn-1 + fibn-2, for n > 2

public static int fib (int n) {

if (n <= 2)

return 1;

else

return fib (n-1) + fib (n-2);

}

It is frank but an inefficient recursion

For larger values the program pauses a very long time between outputs.

**How to multiply 6 by 3 in a recursive way.**

1. To solve “multiply 6 by 1”

Do not need to solve smaller problems, just solve directly, answer is 6 \* 1 = 6

1. Then reconstructing the solution goes as follows:

Multiply 6 by 1 is 6

Multiply 6 by 2 is 6 + the solution of “multiply by 1,” or 12

Multiply 6 by 3 is 6 + the solution of “multiply by 2,” or 18

Public static int multiply (int m, int n) {

Int ans;

if (n==1) ans = m; //simple case

else ans = m + multiply (m, n-1);

}

**or**

public static int multiply (int m, int n) {

if (n==1) return m;

else return m + multiply (m, n-1)

}

**Iterative method to compute m \* n**

Public static multiply (int m, int n) {

int ans = 0;

for (int i = 1; i <= n; ++i)

ans += m;

return ans;

}

**or**

public static int multiply (int m, int n) {

in ans = 0;

while (n-- > 0)

ans += n;

return ans;

}

Conclusion of efficiency:

Recursive methods often have slower execution times when compared to their iterative counterparts.

The overhead for loop repetition is smaller than the overhead for a method call and return.

If it is easier to conceptualize an algorithm using recursion, then you should code it as a recursive method.

The reduction in efficiency does not outweigh the advantage of readable code that is easy to debug. You can always speed up a recursive solution by changing it into a for loop, which also depends on the components on the machine and situation.