COMPARACIÓN ENTRE ALGORITMOS RECURSIVOS E ITERATIVOS Y SU MEDICIÓN EN TÉRMINOS DE EFICIENCIA

Comparison between recursive and Iterative algorithms, and its measurement in terms of efficiency

Juan de Dios Murillo Morera
jmurillo@una.ac.cr
Universidad Nacional

Santiago Caamaño Polini
scpolini@gmail.com
Universidad Nacional

Recibido el 1 de junio de 2011. Corregido el 17 de octubre de 2012. Aceptado el 29 de octubre de 2012.

Resumen*:* Se llevarán a cabo comparaciones simples entre algoritmos recursivos e iterativos, para determinar el grado de eficiencia de un problema en particular. Se efectuaron pruebas de comparación y análisis utilizando tres ejemplos en ambos tipos de algoritmos, a los cuales se les aplicaron los criterios de análisis de algoritmos.

Palabras claves: algoritmos, iteración, recursión, eficiencia, notación O.

Abstract: Some simple comparisons were made between recursion and iteration algorithms to determine the efficiency of a particular problem. To perform the tests three examples of algorithms types were used, to which we will apply algorithms analysis criteria.

Keywords: algorithms, iteration, recursive, efficience, O notation.

La presente investigación pretende mostrar una comparación entre algoritmos recursivos e iterativos para determinar las principales características y el grado de eficiencia de cada uno. Se busca dar a conocer los criterios necesarios para la medición de algoritmos, con el fin de que puedan ser aplicados en forma general y así determinar el grado de eficiencia, según el problema que se presente.

Se estableció como objetivo principal de la presente investigación:

- Analizar algoritmos en su forma recursiva e iterativa para determinar el grado de eficiencia según el tipo de algoritmo para un problema en particular.

Asociados al objetivo principal, se presentan los siguientes:

- Definir el concepto de algoritmos recursivos e iterativos.
- Caracterizar los diferentes tipos de recursividad.
- Describir las situaciones convenientes para el uso de la recursividad
- Determinar los criterios que permitan la comparación entre los algoritmos recursivos e iterativos.
- Llevar a cabo un cuadro comparativo que resuma la eficiencia de los algoritmos según determinado problema.

Conceptos previos

Los algoritmos iterativos son aquellos que funcionan a partir de procedimientos cíclicos.

Dichos algoritmos cuentan con una secuencia de instrucciones que se realizan mediante una estructura llamada ciclo, la cual consta de:

- ✓ Un valor inicial
- ✓ Un valor final
- ✓ Un incremento

Por otro lado, los algoritmos recursivos son aquellos que contienen llamados a sí mismos de forma finita y se utilizan mayormente en problemas relacionados con las áreas matemáticas y fundamentalmente cuando el problema está definido en términos de sí mismo. Cabe resaltar que la recursividad es una forma elegante, simple, estructurada, modular y clara de resolver problemas complejos en pocas líneas de código, por lo tanto, resulta de gran utilidad. Asimismo, un punto importante de la recursividad es que usa recursos del sistema, tales como el procesador y la pila del sistema, según el tipo de problema en cantidades variables.

Para aplicar la recursividad se necesita establecer dos partes importantes en el algoritmo:

1) Caso base trivial: Es el paso que indica el fin del algoritmo recursivo. Permite que el problema se resuelva sin tener que volver a llamar el método (Bisbal, 2009).

2) Parte recursiva: Relaciona el resultado del algoritmo. Es donde se da lugar a la llamada del método repetitivamente hasta que este llegue al paso base para finalizar (Bisbal, 2009).

Para escribir un algoritmo recursivo es necesario:

1) Identificar el caso trivial antes mencionado.

2) Escribir varios casos en que se haga el llamado recursivo. Asumiendo que funciona por el proceso de inducción.

3) Garantizar que la secuencia de llamadas se realice en pre orden.

4) Es importante resaltar que siempre se debe evaluar la eficiencia de los algoritmos, esto para seleccionar el que mejor abarque las necesidades de un problema determinado. Debe establecerse el conjunto de operaciones y el orden en el cual se llevará a cabo el algoritmo. Una función es mejor cuantos menos recursos utilice, sea más simple, ordenada, robusta y fácil de entender (Bisbal, 2009).

Existen dos tipos diferentes de recursión que se clasifican según si la función se llama a sí misma directa o indirectamente. Estos tipos se definen de la siguiente forma:

1) Recursión directa: Contiene un llamado a sí misma de forma explícita (Bisbal, 2009).

2) Recursión indirecta: Contiene un llamado a otra función que a su vez llama a la primera función. Es decir, realiza el llamado a sí misma a través de otra función (Bisbal, 2009).

La Figura 1 muestra la forma en que estos dos tipos de recursión se ejecutarían.

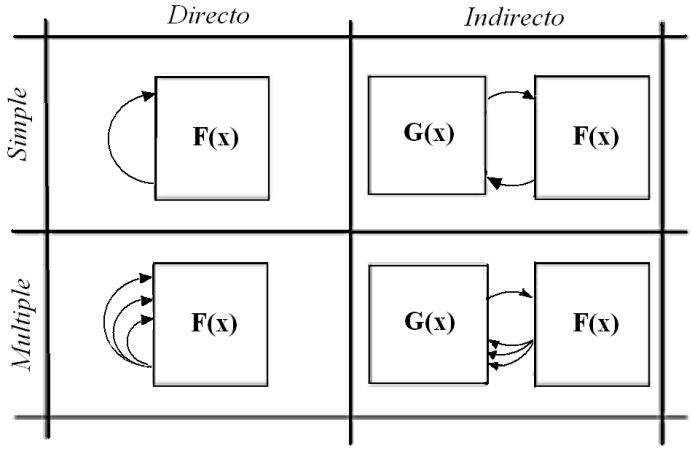


Figura 1. Tipos de recursividad
Nota: Tomado de (Bisbal, 2009)

Estos tipos de recursión se pueden categorizar de la siguiente forma:

- Recursión múltiple: Se genera cuando en la definición aparece más de una vez una llamada recursiva (Martinez, 2003).
- Recursión anidada: Se utiliza cuando se presentan llamadas recursivas en el argumento de la función, es decir, en el envío por parámetros (Martinez, 2003).
- Recursión lineal o simple: Es cuando no hay operaciones pendientes que involucren el llamado recursivo a la función. Esto es, se acumula el resultado sin tener operaciones pendientes, simplemente finaliza en el caso base trivial y devuelve algún valor como resultado. De forma frecuente, actualiza el resultado de la operación en sus parámetros para conservarlo en los llamados (Gupta, 2010).

De esta subcategoría de algoritmos recursivos, se determina que si la ultima instrucción por ejecutar es la llamada recursiva, se puede decir que es recursión final o recursión por cola (tail recursion) (Gupta, 2010).

En la figura 2 se muestran, de forma clara, los dos tipos de recursión y sus sub-categorías.

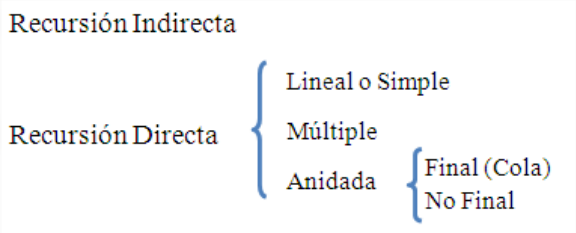


Figura 2. Tipos de recursividad con subcategorías

Es importante determinar que ante un problema específico se debe escoger entre algoritmos recursivos e iterativos. Existen criterios específicos que determinan cuándo es conveniente utilizar la recursividad por motivos de eficiencia, por lo tanto, es justificable cuando (Falqueras, 2010):

1. Se sabe que la función no va a generar demasiada profundidad de llamadas recursivas. Es decir, que un problema que requiera de constantes llamados recursivos no debe solucionarse de esta forma.
2. Se sabe que la función no va a utilizar estructuras de datos locales demasiado grandes. Como es el caso de las listas, vectores, entre otros.
3. Cada llamada no genera, a su vez, llamadas ya resueltas en otras llamadas que se generarán o se han generado antes.
4. La solución no se puede plantear de forma sencilla de otra manera que no sea por recursividad. Por ejemplo, algoritmos que resuelven problemas matemáticos.

Otro factor que incide es el lenguaje de programación en el cual deba resolverse el problema en particular, ya que por sí mismos existen lenguajes que solo trabajan de forma recursiva y otros que pueden incorporar ambos tipos de algoritmos.

Para esta investigación se compararan los algoritmos recursivos en relación con los algoritmos iterativos, para medir el grado de eficiencia que posee cada uno de estos para solucionar un problema específico. Cabe destacar que en ciertas ocasiones es más conveniente utilizar algoritmos iterativos, debido al gasto de recursos o a la gran cantidad de llamados que se realizan en determinado algoritmo.

Procedimientos

Se consideraran tres ejemplos de algoritmos, los cuales se implementaron en forma tanto recursiva como iterativa, con el propósito de realizar una comparación entre cada tipo de problemas. Para ello, debemos definir el concepto de eficiencia:

Eficiencia es cuando se logra cumplir con los objetivos planteados utilizando la menor cantidad de recursos posibles, por lo que se alcanza minimizar el uso memoria, de pasos y de esfuerzo humano (Berzal, 2010).

El análisis de la eficiencia se divide en dos partes fundamentales espacio y tiempo (Berzal, 2010).

- La eficiencia en espacio se mide en función de los tipos y volumen de datos que utilice, y se calcula en bits o bytes.
- La eficiencia en tiempo depende de :

✓ Los datos de entrada al programa.
✓ La calidad del código generado por el compilador.
✓ La rapidez de las instrucciones de la máquina.
✓ La complejidad del algoritmo.

La complejidad no se puede expresar en unidades de tiempo, por lo que se usa la notación asintótica.

Notación O mayúscula. La notación O mayúscula sirve para la comparación asintótica de funciones. Es de gran utilidad para calcular la cantidad de tiempo que toma ejecutar un programa.

Definición de la notación O

$$O(g(x)) = \left\{ \begin{array}{c} f(x) : \text{existen } c, x_0 > 0 \text{ tales que} \\ \forall x \geq x_0 : 0 \leq |f(x)| \leq c|g(x)| \end{array} \right\}$$

(Berzal, 2010)

Resultados y discusión

Las tablas que se muestran a continuación, así como los gráficos, representan los datos que se obtuvieron luego de analizar y evaluar los algoritmos de Fibonacci, Factorial y Máximo Común Divisor, resueltos tanto de forma recursivo como de forma iterativa. Seguidamente se explicará con detalle cada herramienta de evaluación de datos, así como las métricas que se utilizaron para llevar a cabo dichas evaluaciones.

Se utilizó como primera herramienta de evaluación dos tablas, en las que se describe la aplicación de métricas a los algoritmos que fueron seleccionados para ser evaluados.
La Tabla1 representa los datos obtenidos al solucionar los algoritmos de: Fibonacci, Factorial y Máximo Común Divisor (MCD) de forma recursiva. De forma análoga, en la Tabla 2 se muestran los mismos algoritmos, pero con datos evaluados de forma iterativa.

Tabla 1. Resumen de resultados
Algoritmos de forma recursiva

Métricas	Factorial	Fibonacci	MCD
Tiempo de Ejecución	1,42 ms	90.39 ms	0,022 ms
Eficiencia	Lineal	Exponencial	Logarítmica

Tabla 2. Resumen de resultados
Algoritmos de forma iterativa

Métricas	Factorial	Fibonacci	MCD
Tiempo de Ejecución	0,19 ms	0,0596 ms	0,013 ms
Eficiencia	Lineal	Exponencial	Logarítmica

Las tablas anteriores poseen dos métricas a evaluar las cuales son:

- **Eficiencia (Notación O):** Como se explicó en el apartado anterior de Procedimiento, esta notación se utiliza para comparar asintóticamente distintas funciones, en este caso en particular, los costos referentes de los tres algoritmos al resolver el mismo problema.
  La Tabla 3 muestra los ordenes más utilizados para clasificar un algoritmo cuando se realiza un análisis del mismo.

Tabla 3.
Ordenes más utilizadas en el análisis de algoritmos. Tomado de (Berzal, 2010)

Notación	Nombre
O(1)	Orden-constante.
O(log(n))	Orden-logarítmico.
O(n)	Orden-lineal.
O(n log(n))	Orden-lineal-logarítmico.
$O(n^c)$	Orden-potencial
$O(c^n, n>1)$	Orden-exponencial
O(n!)	Orden-factorial
$O(n^n)$	Orden-potencial-exponencial

Cada algoritmo posee un orden específico, dependiendo de la notación que este posea según la tabla anterior, se puede determinar si un algoritmo es más eficiente que otro basándose en la posición en la que se encuentre en dicha tabla. De los datos obtenidos el MCD tanto en forma recursiva e iterativa, es el más eficiente, ya que posee un orden logarítmico (O(log n), luego se encuentra el algoritmo del factorial en sus dos formas y el Fibonacci en forma iterativa, los cuales poseen un orden lineal (O(n)), que resultan ser más eficiente que el algoritmo de Fibonacci en forma recursiva, ya que este posee un orden exponencial.

- **Tiempo de ejecución:**

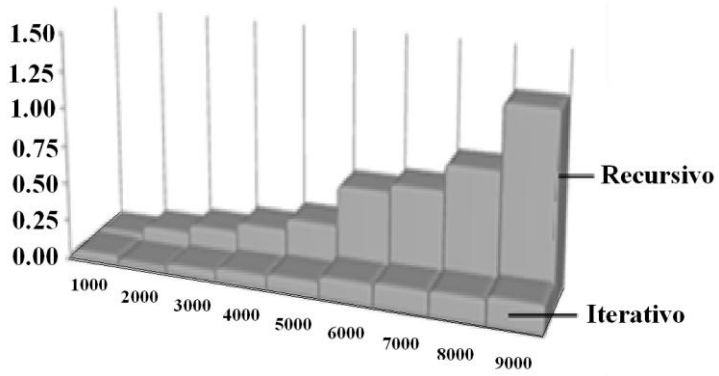
Se procedió a promediar los tiempo de ejecución de cada algoritmo mediante el uso de una herramienta creada en el lenguaje de programación C#, que determina el tiempo empleado por un algoritmo para dar una solución, medida en milisegundos.
Para cada método se promedio el tiempo según treinta ciclos realizados, para garantizar la veracidad de la información.

Se ejecutó el método de factorial iterativo y recursivo treinta veces con el número 9000. De igual forma el Fibonacci se ejecutó treinta veces cn el valor 1000.
Asimismo, el Máximo Común Divisor se ejecutó treinta veces realizando la operación con los valores 702 y 540.

Los datos obtenidos reflejan que para los tres algoritmos en su forma iterativa son más eficientes ya que consumen menos tiempo de ejecución.
Como segunda herramienta de evaluación de datos se procedió a utilizar tres gráficos en los cuales se realizan comparaciones de tiempos entre cada algoritmo en su forma recursiva e iterativa.

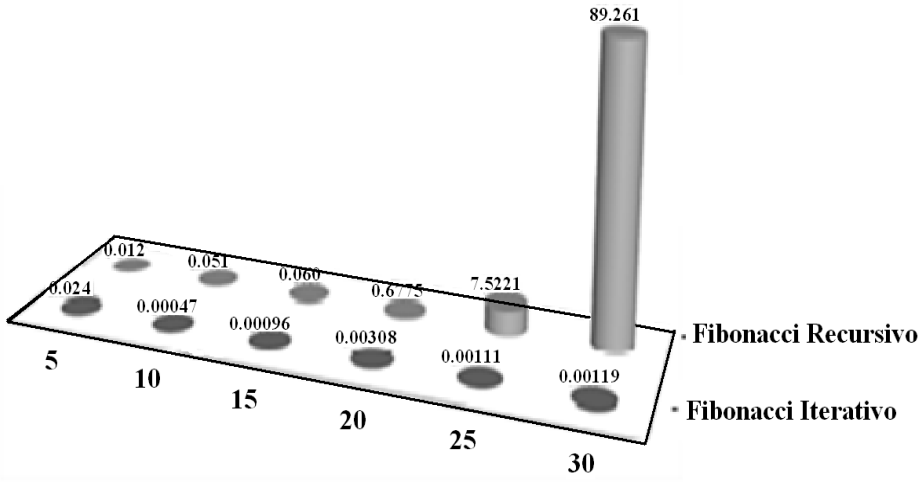
En el Gráfico 1 se muestra la comparación del algoritmo del Factorial recursivo e iterativo. Se procedió a promediar el tiempo de ejecución de dicho algoritmo en 30 ciclos, con nueve valores de n distintos. De esta forma se busco obtener un comportamiento creciente según se valla incrementando el valor de n. Se evidenció que en su forma iterativa consume menos tiempo llevar a cabo la solución.

Grafico 1.
Algoritmo del factorial de forma recursiva e iterativa



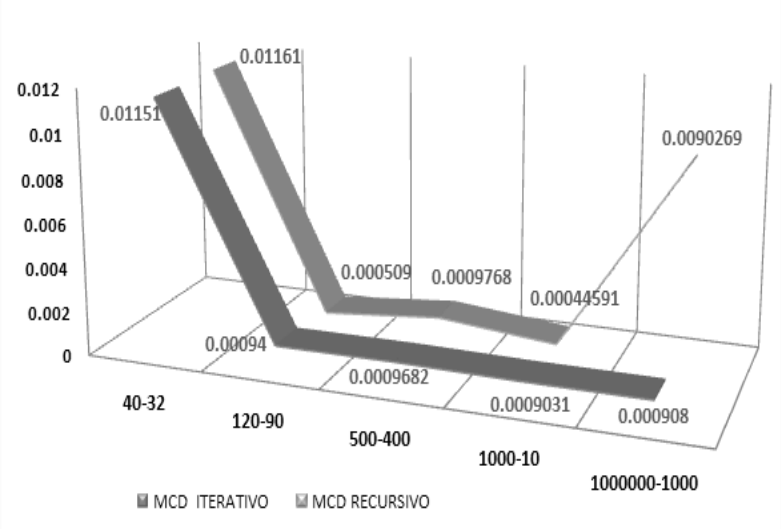
En el Grafico 2 se encuentra la comparación del algoritmo de Fibonacci recursivo e iterativo. Donde se promedió el tiempo de ejecución de dicho algoritmo en 30 ciclos, con seis valores de n distintos. Se obtuvo un mayor grado de velocidad de ejecución por parte de la forma iterativa de este algoritmo. De esta forma se evidencia como puede afectar un orden O(n) a un orden exponencial, como se presentó anteriormente.

Grafico 2.
Algoritmo de Fibonacci de forma recursiva e iterativa



Seguidamente se evaluó el algoritmo de Máximo Común Divisor (MCD) como se muestra en el gráfico 3, se valoró de forma distinta, ya que no depende de un n en particular. Se realizaron cinco ejecuciones con pares de valores distintos, buscando aumentar el grado de complejidad de solución, para encontrar una posible diferencia de tiempos entre las dos formas de solución. En este caso, igual que los anteriores, se evidenció que la forma iterativa es más eficiente aun así perteneciendo al mismo orden que el recursivo.

Grafico 3.
Algoritmo de máximo común divisor de forma recursiva e iterativa



Conclusiones

Los algoritmos iterativos tienden a ser más eficientes que los algoritmos recursivos según un problema en particular, ya que poseen un orden que los hace más eficientes. Cabe destacar que esta relación se pudo realizar ya que los problemas planteados se podían realizar por medio de ambos tipos de algoritmos. Además, se evidencia por medio del análisis de tiempos de ejecución donde claramente los algoritmos iterativos se realizan en menor tiempo que los recursivos siendo así estos más eficientes.

Además, se comprueba que un método que se ejecuta n cantidad de veces, se logra resolver en menor cantidad tiempo, ya que el compilador al realizar el análisis almacena en memoria un direccionamiento al método ya compilado y analizado sintáctica y semánticamente.

Es preciso emplear algoritmos recursivos cuando no se tengan que realizar múltiples llamados al mismo método ya que resultan ser ineficientes. Asimismo, es conveniente tomar en cuenta, que se utilicen algoritmos recursivos cuando no se tenga otra opción para solucionar el problema.

Referencias

Berzal, F. (2010).“*Tipos de Eficiencia de un algoritmo”*. Recuperado 8-8-2010 de: URL = http://elvex.ugr.es/decsai/c/apuntes/algoritmos.pdf

Bisbal, J. (2010). “Manual de algorítmica Recursividad, complejidad, y diseño de algoritmos”. Editorial UOC. Barcelona España, Diciembre *2009*. Recuperado 15-8-2010 de URL = http://books.google.co.cr/books?id=2sSvS0pDfpAC&printsec=frontcover&hl=en#v=onepage&q&f=false.

Falqueras, J. (2010). “*Programación Modular”*. Apuntes del Profesor Juan Falgueras. Departamento de Lenguajes y Ciencias de la Computación. Universidad de Málaga, Malaga España 2004. Recuperado 25-9-2010 de URL = http://juanfc.lcc.uma.es/EDU/PM/4.Recursividad.pdf

Gupta, P. (2010). “Data Structure Using C. *Types of Recursion”*. Editorial Firewall Media. Recuperado 15-10-2010 de URL = http://books.google.co.cr/books?id= WtcO_4t5dYgC&printsec=frontcover&hl=en#v=onepage&q&f=false

Martinez, F. y Quetglas, G. (2003). “Introducción a la programación estructurada en C”. Universidad de Valencia, España *2003*. Recuperado 10-11-2010 de URL = http://books.google.co.cr/books?id=-cVzTPOWf3kC&printsec = frontcover#v= onepage&q&f = false