UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Estudio del tiempo de ejecución de algoritmos recursivos a través de relaciones de recurrencia lineal

Apellidos, nombre	Sanabria Codesal, Esther[1] (esanabri@mat.upv.es) Casanova Faus, Assumpció[2](casanova@dsic.upv.es) Marqués Hernández, Francisco[2] (pmarques@dsic.upv.es)
Centro	Universitat Politècnica de València

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Resumen

En este artículo vamos a presentar un ejemplo de colaboración entre dos materias básicas: las Matemáticas y la Informática. La idea es mejorar el aprendizaje de los alumnos, reforzando los conceptos impartidos en ambas, poniendo de manifiesto que las materias básicas, que en ocasiones resultan poco interesantes, son importantes para su formación como profesionales. En este caso abordaremos los costes de los algoritmos recursivos, punto de interés para asignaturas de Informática, a través de la resolución de relaciones de recurrencia, herramienta proporcionada por las asignaturas de Matemáticas.

1 Introducción

Aprender a programar es un proceso que necesita que el alumno desarrolle recursos de diversa índole:

- Debe llegar a dominar herramientas específicas, como el lenguaje o lenguajes de programación y sus entornos de desarrollo, además de las técnicas algorítmicas, básicas y avanzadas, de aplicación en el ámbito informático.
- Debe llegar a ser capaz de enfrentarse a la resolución de problemas, en el sentido más amplio, de modo que pueda analizarlos y hacerlos abordables mediante las técnicas anteriores.

En ambos aspectos interviene de forma significativa su formación matemática.

Al mismo tiempo, que el alumno perciba qué conceptos y técnicas matemáticas le respaldan en su tarea de programador, puede ayudarle a:

- Sentirse más interesado en el estudio de las matemáticas.
- Reforzar la asimilación de los conceptos matemáticos, que se revisan con ejemplos procedentes un ámbito diferente.

Consideramos que una buena manera de fomentar esta interrelación es desarrollando uno a uno, y detalladamente, ejemplos concretos como los que se presentan en este artículo, donde se analiza el tiempo de ejecución de los algoritmos recursivos.

2 Objetivos

El trabajo tiene como objetivo poner de manifiesto la estrecha relación entre los conceptos impartidos en asignaturas básicas en los grados y para ello resolveremos problemas propios de asignaturas de informática, utilizando las herramientas proporcionadas por las asignaturas de matemáticas.

3 Análisis del coste de los algoritmos recursivos

Una característica importante de un buen programa informático es el tiempo que toma su ejecución, especialmente cuando se trata con grandes volúmenes de datos y con problemas complejos. El análisis del coste de los algoritmos es la parte de la Programación que se ocupa de estudiarlo, y nos ayuda a fijarnos en los

aspectos de un programa que influyen decisivamente en el tiempo consumido, sobre todo a medida que crece el volumen de los datos de entrada, e independientemente de la máquina concreta en que se ejecuta.

Dicho problema se resuelve contando los pasos que se ejecutan para un volumen de datos o talla del programa. Como se discute en el capítulo 12 de [1], entendemos por paso de programa toda aquella operación que se hace independientemente de lo grande que sea el dato, obteniendo el número de pasos de programas como una función de coste $T(n)$, en donde n es la talla.

Cuando un algoritmo está expresado recursivamente y queremos contar los pasos de programa que toma su ejecución, podemos suponer que las subllamadas recursivas generadas tienen el mismo comportamiento temporal que la llamada inicial, dado que siguen la misma estrategia, pero trabajando con datos de menor talla. Por ello, lo más natural parece ser expresar también $T(n)$ de forma recurrente.

Por ejemplo, dado un entero $n \geq 0$, el algoritmo de la figura 1 calcula recursivamente su representación en binario. El algoritmo viene dado como un método Java que recibe a n como parámetro, y que escribe en la salida una secuencia de 0 y 1 correspondiente a su representación en base 2.

```
/** Dado n>=0, escribe en la salida la
 *  representación en binario de n.
 *  @param n un entero >=0
 */
 public static void binario(int n){
    if (n<=1) System.out.print(n);
    else { binario(n/2);
           System.out.print(n%2);
         }
 }
```

Figura 1. Cálculo de la representación en binario de n.

Se trata de un ejemplo de recursividad lineal no final (según la clasificación dada en el capítulo 11 de [1]), donde en primer lugar obtenemos la representación en binario de $n/2$, y a su derecha se escribe el bit menos significativo (el resto de $n/2$, calculado en Java como $n\%2$). Por ejemplo, para calcular la representación en binario de 18, en primer lugar escribimos la representación de 9, es decir, 1001, escribiendo a continuación 0 para obtener como resultado 10010.

De esta manera, en cada llamada recursiva se va añadiendo un bit más al resultado. Notamos que el bit más significativo del argumento (que vale 1 para cualquier n, salvo en el caso particular $n = 0$) se obtiene en la llamada correspondiente al caso base.

Una vez analizado el problema llegamos a la relación de recurrencia para el coste

$$T(n) = \begin{cases} 1 & n = 1 \\ 1 + T\left(\frac{n}{2}\right) & n > 1 \end{cases}$$

Ecuación 1. Coste del algoritmo de la figura1.

3.1 Solución a través de sustitución

La recurrencia de la ecuación 1 podemos resolverla por sustitución

$$T(n) = 1 + \ T\left(\frac{n}{2}\right) = 2 \ + T\left(\frac{n}{4}\right) \ = 3 + T\left(\frac{n}{8}\right) = \cdots = k + T\left(\frac{n}{2^k}\right)$$

Si hacemos el cambio de variable $n = 2^k$, obtenemos que

$$T(n) = k + T\left(\frac{n}{2^k}\right) = \ log_2 n + T(1) = log_2 n + 1$$

Este resultado nos indica que el coste es proporcional a la parte entera de $log_2 n$, es decir, al número de veces que debemos dividir n por 2 para llegar a 1, o dicho de otro modo, el coste es proporcional al número de cifras de n en base 2.

Precisamente, otro posible análisis del problema consiste en considerar como talla el número de bits de n, al que podemos llamar k. Así, en cada llamada recursiva reducimos en uno el número de bits, y entonces

$$T(k) = \begin{cases} 1 & k = 1 \\ 1 + T(k-1) & k > 1 \end{cases}$$

Ecuación 2. Relación de recurrencia lineal

La resolución por sustitución en este caso es la siguiente

$$T(k) = 1 + T(\,k-1\,) = 2 + T(k-2) = 3 + T(\,k-3) = \cdots = k + T(1) = \ k + 1.$$

3.2 Solución por el método de la ecuación característica

Las recurrencias de las ecuaciones 1 y 2 son relativamente sencillas y se han podido resolver de forma directa por sustitución reiterada de términos.

Encontramos otro ejemplo un poco más complejo en el problema de buscar el mínimo de un array de enteros, entre unas posiciones inicial ini y final fin. En la figura 2 vemos, escrita en Java, una resolución del problema mediante recursión no lineal.

En este método, si $a[ini\,..\,fin]$ tiene un número de componentes n mayor que 1, se reduce a dos subarrays no vacíos de talla las partes enteras por defecto o por exceso de $n/2$, es decir $\lfloor n/2 \rfloor$, o bien $\lceil n/2 \rceil$. Una vez calculados los mínimos de ambas partes, es suficiente comparar ambos mínimos para obtener el mínimo buscado. Si el array tiene a lo sumo un elemento, dicho elemento es trivialmente el mínimo.

Por ejemplo, si a es el array $\{3,4,2,6,0,-2,3\}$, y el rango de índices $ini\,..\,fin$ abarca todo el array, es decir, va desde 0 a 6 inclusive, el problema se reduce a calcular recursivamente el mínimo en el rango $0\,..\,3$ (las cuatro primeras componentes), y el mínimo en el rango $4\,..\,6$ (las tres últimas); estos mínimos son respectivamente 2 y -2, y de la comparación de ambos se obtiene que el mínimo de a es -2.

Así pues, todo problema se va reduciendo recursivamente a dos subproblemas de talla aproximadamente mitad y que en el caso base serán de talla 1.

```java
/** Dado un array a de int, calcula el mínimo de
 *  los valores en a[ini..fin], 0<=ini<=fin<a.length.
 *  @param int[] a
 *  @param int ini
 *  @param int fin
 *  @return el mínimo de a[ini..fin]
 */
public static int minimo(int[] a,int ini,int fin){
    if (ini==fin) return a[ini];
    else { int medio = (ini+fin)/2,
               min1 = minimo(a,ini,medio),
               min2 = minimo(a,medio+1,fin);
           if (min1<min2) return min1; else return min2;
    }
}
```

Figura 2. Cálculo del mínimo de un array en el rango $ini..fin$.

Una vez analizado el problema llegamos a la relación de recurrencia para el coste

$$T(n) = \begin{cases} 1 & n = 1 \\ 1 + + T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) & n > 1 \end{cases}$$

Ecuación 3. Coste el algoritmo de la figura 2.

que aproximaremos por

$$T(n) = \begin{cases} 1 & n = 1 \\ 1 + 2\,T\left(\frac{n}{2}\right) & n > 1 \end{cases}$$

Ecuación 4. Coste el algoritmo de la figura 2.

En este caso podemos resolverla por sustitución

$$T(n) = 1 + 2\ T\left(\frac{n}{2}\right) = 1 + 2\left(1 + 2\,T\left(\frac{n}{4}\right)\right) = \cdots = \sum_{i=1}^{k} 2^{i-1} + 2^k T\left(\frac{n}{2^k}\right)$$

Utilizando el hecho de que $\sum_{i=1}^{k} 2^{i-1} = 2^k - 1$ y haciendo el cambio de variable $n = 2^k$ obtenemos

$$T(n) = 2^k - 1 + 2^k T\left(\frac{n}{2^k}\right) = n - 1 + n\,T(1) = 2n - 1$$

Con el objetivo de utilizar las herramientas matemáticas para resolver la relación de recurrencia obtenida en la ecuación 4, realizamos el cambio de variable $n = 2^k$, donde consideramos que k es la parte entera de $log_2 n$ y $\frac{n}{2} = 2^{k-1}$. Por tanto, la relación de recurrencia lineal viene dada por la sucesión $a_k = T(2^k)$ con $a_0 = T(2^0) = T(1) = 1$ y $a_k = T(2^k) = 1 + 2\,T(2^{k-1}) = 1 + 2a_{k-1},\ k > 0$.

De esta forma hemos modificado el problema inicial transformándolo en un ejercicio típico de ecuaciones en diferencias de la asignatura de matemáticas.

$$a_k = \begin{cases} 1 & k = 0 \\ 1 + 2a_{k-1} & k > 0 \end{cases}$$

Ecuación 5. Relación de recurrencia lineal con coeficientes constantes.

Para resolverlo es necesario encontrar una familia de sucesiones que serán la solución general de la relación de recurrencia. Para ello utilizaremos el siguiente teorema:

"La solución general de una recurrencia lineal $a_n = b(n) + c_1 a_{n-1} \dots + c_k a_{n-k}$, donde $c_i \in R$, $c_k \neq 0$ y $b(n)$ es un función de $n \in N$, viene dada por la expresión $a_n = a_n^h + a_n^p$, donde a_n^h representa la solución general de la ecuación lineal homogénea asociada, es decir $a_n = c_1 a_{n-1} \dots + c_k a_{n-k}$ y a_n^p es una solución particular (cualquiera) de la ecuación completa" ([3]).

Siguiendo el método habitual para resolver ecuaciones en diferencias ([4]), calculamos la solución general de la recurrencia homogénea de coeficientes constantes $a_k - 2a_{k-1} = 0$, considerando la ecuación característica asociada, dada en este caso por $\lambda - 2 = 0$.

Utilizando las soluciones de esta ecuación, en nuestro caso $\lambda = 2$, construimos la familia de sucesiones $a_k^h = C2^k$, donde C es una constante arbitraria, que cumplen la igualdad $a_k - 2a_{k-1} = 0$.

Puesto que la recurrencia lineal de la ecuación 5 tiene la función $b(n) = 1$, es decir $b(n)$ es un polinomio de grado cero, sabemos que sus soluciones particulares tienen la forma $a_k^p = A$, donde A es una constante que tendremos que determinar.

Para ello sustituimos en la relación de recurrencia anterior $a_k - 2a_{k-1} = 1$

$$a_k^p - 2a_k^p = A - 2A = 1 \rightarrow A = -1$$

Luego $a_k^p = -1$ y la familia de sucesiones que cumplen la relación de recurrencia $a_k = 1 + 2a_{k-1}$, es decir, la solución general de la recurrencia viene dada por $a_k = a_k^h + a_k^p = C2^k - 1$, donde C es una constante arbitraria.

Para encontrar la solución particular, es decir la única sucesión que la relación de recurrencia sujeta a la condición inicial $a_0 = 1$, será necesario calcular la constante C. Para ello sustituimos la condición inicial en la solución general

$$1 = a_0 = C2^0 - 1 \rightarrow C = 2$$

Luego la solución de la ecuación 5 viene dada por $a_k = 22^k - 1 = 2^{k+1} - 1$, para todo $k \geq 0$.

Deshaciendo el cambio de variable $n = 2^k$ y recordando que hemos denotado $a_k = T(2^k)$, obtendremos la sucesión que cumple la recurrencia dada por la ecuación 4

$$T(n) = T(2^k) = a_k = 2\,2^k - 1 = 2\,n - 1, \quad n \geq 1.$$

4 Desarrollo del coste de Fibonacci

El último ejemplo que trataremos es la sucesión de Fibonacci definida por la recurrencia:

$$f_n = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ f_{n-1} + f_{n-2} & n > 1 \end{cases}$$

Ecuación 6. Sucesión de Fibonacci.

En la parte izquierda de la figura 3, vemos un método recursivo que implementa el cálculo del término f_n siguiendo literalmente la definición anterior. Mientras que a la derecha, en la misma figura, tenemos una versión iterativa de Fibonacci que calcula f_n con n pasos de programa.

```
/** Cálculo de la sucesión de Fibonacci
 *  @param n un entero >=0
 *  @return el n-ésimo término
 */
public static int fR(int n){
    if (n<=1) return n;
    else return fR(n-1)+
                fR(n-2);
 }
```

```
/** Cálculo de la sucesión de Fibonacci
 *  @param n un entero >=0
 *  @return el n-ésimo término
 */
public static int fI(int n){
    if (n<=1) return n;
    int fAnt2 = 0, fAnt1 = 1, f = 1;
    for (int i = 2; i<n; i++) {
        fAnt2 = fAnt1; fAnt1 = f;
        f = fAnt1 + fAnt2;
    }
    return f;
 }
```

Figura 3. Cálculos recursivo e iterativo de la sucesión de Fibonacci.

Una diferencia que podemos observar entre las dos versiones es que el cálculo recursivo repite muchos cálculos, como vemos por ejemplo en la figura 4, en la que hemos desarrollado el árbol de llamadas recursivas para $n = 5$.

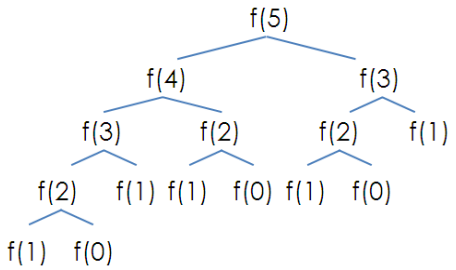

Figura 4. Árbol de llamadas del cálculo recursivo de Fibonacci de 5.

Para medir cómo va a afectar esta redundancia al tiempo de ejecución de este algoritmo, nos vamos a plantear a continuación el estudio del coste mediante las relaciones de recurrencia para $T(n)$.

4.1 Solución a través de sustitución

El coste del cálculo recursivo de Fibonacci se expresa mediante la siguiente relación

$$T(n) = \begin{cases} 1 & 0 \le n \le 1 \\ 1 + T(n-1) + T(n-2) & n > 1 \end{cases}$$

Ecuación 7. Relación de recurrencia del coste de Fibonacci recursivo.

En este caso no podemos resolverla directamente por sustitución, pero sí podemos acotarla por dos sucesiones más simples ya que es una sucesión creciente

$$1 + T(n-2) + T(n-2) \le 1 + T(n-1) + T(n-2) \le 1 + T(n-1) + T(n-1), \qquad n > 1$$

Luego $1 + 2\,T(n-2) \leq\ T(n) \leq 1 +\ 2\,T(n-1),\ n > 1$.
Por tanto aplicando sustitución a la cota inferior obtenemos

$$1 + 2\,T(n-2) = 1 + 2\left(1 + 2\,T(n-4)\right) = \cdots = \sum_{i=1}^{k} 2^{i-1} + 2^k T(n-2k)$$

Sustituyendo $\sum_{i=1}^{k} 2^{i-1} = 2^k - 1$, el término general de la sucesión viene dado por

$$1 + 2\,T(n-2) = 2^k - 1 + 2^k T(n-2k) = \alpha 2^{n/2} - 1\,,\ n > 1$$

donde $\alpha = 2$ si $n = 2\,k$ (par) y $\alpha = 2^{1/2}$ si $n = 2\,k + 1$ (impar). Aplicando el mismo razonamiento de sustitución a la cota superior obtenemos

$$1 + 2\,T(n-1) = 1 + 2\left(1 + 2\,T(n-1)\right) = \cdots = \sum_{i=1}^{k} 2^{i-1} + 2^k T(n-k)$$

Sustituyendo $\sum_{i=1}^{k} 2^{i-1} = 2^k - 1$, el término general de la sucesión en este caso viene dado por una única expresión para todo n

$$1 + 2\,T(n-1) = 2^k - 1 + 2^k T(n-k) = 2\,2^n - 1, \qquad n > 1$$

Por tanto

$$\alpha 2^{n/2} - 1 = 1 + 2\,T(n-2) \leq T(n) \leq 1 +\ 2\,T(n-1) =\ 2^{n+1} - 1, \qquad n > 1$$

donde $\alpha = 2$ si *n es par y* $\alpha = 2^{1/2}$ si *n es impar.*

Por ejemplo, para $n = 5$ la cota inferior corresponde a suponer un desarrollo de llamadas como se muestra en la figura 5, a la izquierda; mientras que la cota superior corresponde a suponer un desarrollo de llamadas recursivas como el de la misma figura a la derecha.

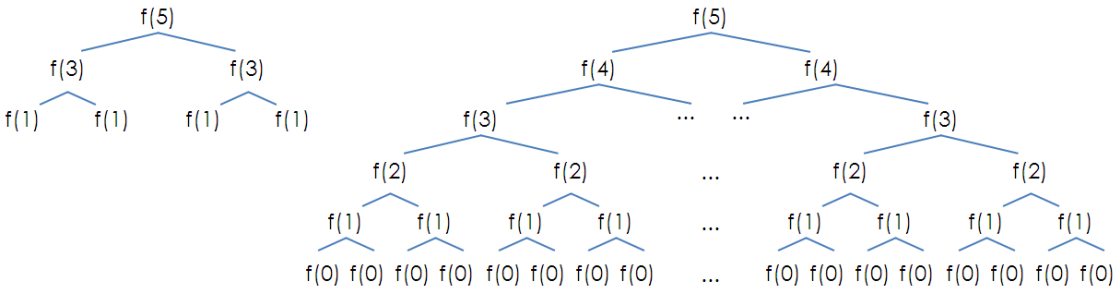

Figura 5. Cotas inferior y superior del comportamiento de Fibonacci recursivo para $n = 5$.

En resumen, las dos cotas anteriores son exponenciales en n (con bases $2^{1/2}$ y 2, respectivamente), lo que puede resultar suficiente para compararlo con el coste del Fibonacci iterativo.

Aun así, podemos ser todavía más precisos si aplicamos las herramientas matemáticas ya conocidas.

4.2 Solución a través de relaciones de recurrencia

Consideramos $T(n) = a_n$ para transformar el problema del coste de la ecuación 7 en un típico ejercicio de ecuaciones en diferencias de la asignatura de matemáticas

$$a_n = \begin{cases} 1 & 0 \le n \le 1 \\ 1 + a_{n-1} + a_{n-2} & n > 1 \end{cases}$$

Ecuación 8. Relación de recurrencia lineal con coeficientes constantes.

Para calcular la solución general de la recurrencia homogénea de coeficientes constantes $a_n = a_{n-1} + a_{n-2}$, necesitamos construir la ecuación característica, dada en este caso por $\lambda^2 - \lambda - 1 = 0$.

Utilizando las soluciones de la ecuación característica $\lambda = \frac{1 \pm \sqrt{5}}{2}$, construimos la familia de sucesiones a_n^h que cumplen la igualdad $a_n = a_{n-1} + a_{n-2}$. Éstas vienen dadas por la expresión $a_n^h = C_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + C_2 \left(\frac{1-\sqrt{5}}{2}\right)^n$, donde C_1 y C_2 son constantes arbitrarias.

Puesto que la recurrencia lineal de la ecuación 8 tiene la función $b(n) = 1$, sabemos que las soluciones particulares tienen la forma $a_n^p = A$, donde A es una constante que tendremos que determinar.

Para ello sustituimos a_n^p en la relación de recurrencia anterior $a_n - a_{n-1} - a_{n-2} = 1$

$$a_n^p - a_{n-1}^p - a_{n-2}^p = A - 2A = 1 \rightarrow A = -1$$

Luego $a_n^p = -1$ y la solución general de la recurrencia viene dada por $a_n = a_n^h + a_n^p = C_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + C_2 \left(\frac{1-\sqrt{5}}{2}\right)^n - 1$, donde C_1 y C_2 son constantes arbitrarias.

Para encontrar la solución particular de la ecuación 8, será necesario calcular las constantes C_1 y C_2. Para ello sustituimos las condiciones iniciales $a_0 = a_1 = 1$ en la solución general

$$1 = a_0 = a_1 \rightarrow C_1 = \frac{1 + \sqrt{5}}{\sqrt{5}},\ C_2 = \frac{1 - \sqrt{5}}{\sqrt{5}}$$

Y por tanto la solución de la recurrencia de la ecuación 7 viene dada por

$$T(n) = a_n = \frac{1 + \sqrt{5}}{\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2}\right)^n + \frac{1 - \sqrt{5}}{\sqrt{5}} \left(\frac{1 - \sqrt{5}}{2}\right)^n - 1, \quad n \ge 0.$$

5 Conclusión

Mediante algunos ejemplos de análisis del coste de algoritmos recursivos, hemos podido comprobar:

- En primer lugar, el interés de conocer y aplicar una herramienta matemática concreta: la resolución de recurrencias lineales, al estudio del comportamiento temporal de un conjunto de algoritmos recursivos.
- Por otra parte, que los problemas de análisis de algoritmos que se ha resuelto puede ayudar a interpretar los pasos dados cuando se aplica dicha herramienta matemática.

De una manera más general, queremos resaltar la importancia de la formación matemática de los estudiantes de Informática para su profesión ([5]). Una muestra notable de esta importancia es el premio "Fronteras del conocimiento" otorgado en España en 2011 al matemático Donald E. Knuth. Como se reseña en [6], el jurado le atribuyó "hacer de la programación informática una ciencia introduciendo técnicas matemáticas para el análisis riguroso de los algoritmos", y reconoció que "aportó elegancia al promover la escritura de un código de programación sencillo, compacto y comprensible de forma intuitiva".

6 Bibliografía

6.1 Libros:

[1] Prieto Sáez, N; Casanova Faus, A; Marqués Hernández, F; y otros: "Empezar a programar usando Java", Ed. UPV, 2012.

[2] Graham, R L; Knuth, D E; Patashnik, O: "Concrete Mathematics: Foundation for Computer Science", Ed. Addison Wesley, 1994.

[3] Fernández Pérez, C; Vázquez Hernández, F J; Vegas Montaner, J M: "Ecuaciones diferenciales y en diferencias", Ed. Thomson, 2003.

[4] García Merayo, F; Hernández Peñalver, G; Nevot Luna, A: "Problemas resueltos de Matemática Discreta", Ed. Thomson, 2003.

6.2 Comunicaciones presentadas en conferencias:

[5] Martínez Hinarejos, C D; Sanabria Codesal, E: "Una experiencia de coordinación entre las asignaturas de Análisis Matemático y Programación de la ETSINF", Actas de las VIII Jornadas de Redes de Investigación en Docencia Universitaria, Alicante, 2010.

6.3 Referencias de fuentes electrónicas:

[6] Entrevista a Donald E. Knuth: "El hombre que habla a las máquinas", El País, 18 de enero de 2011. Disponible en:
http://sociedad.elpais.com/sociedad/2011/01/18/actualidad/1295305202_850215.html