Introducción a la complejidad de algoritmos recursivos

January 14, 2011

Funciones temporales en algoritmos recursivos:

- Suelen tener asociadas funciones de coste recursivas (*relaciones recurrentes o recurrencias*).
- Resolver una ecuación de recurrencia consiste en obtener una expresión no recursiva de la función de coste.

Permite resolver recurrencias de la forma:

$$T(n) = \begin{cases} cn^k, & \text{si } 0 \leq \text{n} < \text{b}; \\ aT(n-b) + cn^k, & \text{si n} \geq \text{b}. \end{cases}$$

- a es el número de llamadas recursivas $(a \geq 1)$.
- $n - b$ es el tamaño de los problemas generados
- cn^k es el coste de las instrucciones no recursivas en cada iteración.

Ejemplo

Calcula la complejidad de la siguiente función:

fun *factorial(n)*
  si $n = 0$ **entonces**
    $fact \leftarrow 1$
  si no
    $fact \leftarrow n * factorial(n-1)$
  fin si
  devolver *fact*
fin fun

Solución:

$$T(n) = \begin{cases} c, & \text{si n=0 ;} \\ T(n-1) + c, & \text{si n} \geq \text{0.} \end{cases}$$

$$\Updownarrow$$

$$T(n) = T(n-1) + c =^2 T(n-2) + 2c = \ldots =^n T(0) + nc$$

$$\Updownarrow$$

$$T(n) = cn + c \Rightarrow T(n) \in \Theta(n)$$

Ejemplo

Resuelve la siguiente recurrencia:

$$T(n) = \begin{cases} c, & \textit{si } n = 0; \\ T(n-1) + nc, & \textit{si } n \geq 0. \end{cases}$$

Solución:

Idea clave: No calcular las sumas

$T(n) = T(n-1) + cn$

$$=^2 (T(n-2) + (n-1)c) + nc$$
$$=^3 (T(n-3) + (n-2)c) + (n-1)c + nc$$
$$=^q T(n-q) + c\sum_{j=0}^{q-1}(n-j)$$
$$\dots$$
$$=^n T(0) + c\sum_{j=0}^{n-1}(n-j)$$
$$= (n-1)n + (n-1)(n-2)/2 \Rightarrow T(n) \in \Theta(n^2)$$

Ejemplo

Resuelve la siguiente recurrencia:

$$T(n) = \begin{cases} c, & \textit{si } n = 0; \\ 2T(n-1) + c, & \textit{si } n \geq 0. \end{cases}$$

Solución:

Idea clave: No calcular los productos

$T(n) = 2T(n-1) + c$

$$
\begin{aligned}
&=^2 2(2T(n-2)+c)+c = 2^2T(n-2)+2c+c \\
&=^3 2^2(2T(n-3)+c)+2c+c = 2^3T(n-3)+2^2c+2c+c \\
&=^q 2^qT(n-q)+c\textstyle\sum_{j=0}^{q-1} 2^j \\
&\ldots \\
&=^n 2^nT(0)+c\textstyle\sum_{j=0}^{n-1} 2^j \\
&= 2^n + c2^n - 2 \Rightarrow T(n) \in \Theta(2^n)
\end{aligned}
$$

Permite resolver recurrencias de la forma:

$$T(n) = \begin{cases} cn^k, & \text{si } 1 \leq n < b; \\ aT(n/b) + cn^k, & \text{si } n \geq b. \end{cases}$$

- a es el número de llamadas recursivas $(a \geq 1)$.
- n/b es el tamaño de los problemas generados
- cn^k es el coste de las instrucciones no recursivas en cada iteración.

Ejemplo

Calcula la complejidad de la siguiente función:

```
fun busqueda-binaria(iz,de,X,elem[1..N])
  //iz,de: extremos de la búsqueda
  k ← div((iz + de)/2) //parte entera
  si elem[k] = X entonces
    busqueda ← K
  si no
    si iz = de entonces
      busqueda ← 0 //no existe
    si no
      si elem[k] > X entonces
        busqueda-binaria(iz,k-1,x, elem)
      si no
        busqueda-binaria(k+1,de,x, elem)
      fin si
    fin si
  fin si
  devolver b usqueda
fin fun
```

Solución: Sea m la longitud de la sublista procesada en cada iteración:

$$T(m) = \begin{cases} c, & \text{si m} = 1\ ; \\ T(m/2) + c, & \text{si m} > 1. \end{cases}$$

$$\Updownarrow$$

$$T(m) = T(m/2) + c =^2 T(m/4) + 2c = ... =^k T(1) + kc$$

$$\text{(donde: } \frac{m}{2^k} = 1 \Leftrightarrow k = log(m)\text{)}$$

$$\Updownarrow$$

$$T(m) = clog(m) + c \Rightarrow T(m) \in \Theta(log(m))$$

Ejercicio

Resuelve la siguiente recurrencia:

$$T(n) = \begin{cases} 1, & \textit{si } n = 1; \\ 2T(n-1) + 2n - 1, & \textit{si } n \geq 2. \end{cases}$$

Solución:

$T(n) = 2T(n-1) + 2n - 1$

$$
\begin{aligned}
&=^2 (2T(n-2) + 2(n-1) - 1) + 2n - 1 \\
&= 2^2T(n-2) + 2^2(n-1) - 2 + 2n - 1 \\
&=^3 2^2(2T(n-3) + 2(n-2) - 1) + 2^2(n-1) - 2 + 2n - 1 \\
&= 2^3T(n-3) + 2^3(n-2) - 2^2 + 2^2(n-1) - 2 + 2n - 1 \\
&=^q 2^qT(n-q) + \sum_{j=0}^{q-1} 2^{j+1}(n-j) - \sum_{j=0}^{q-1} 2^j \\
&\ldots \\
&=^{n-1} 2^{n-1}T(1) + \sum_{j=0}^{n-2} 2^{j+1}n - \sum_{j=0}^{n-1} 2^{j+1}j - \sum_{j=0}^{n-2} 2^j \\
&\Rightarrow T(n) \in \Theta(2^n)
\end{aligned}
$$

Ejercicio

Resuelve la siguiente recurrencia:

$$T(n) = \begin{cases} 4, & \textit{si } n = 1; \\ 2T(n/2) + 3n + 2, & \textit{si } n \geq 2. \end{cases}$$

Solución:

$T(n) = 2T(n/2) + 3n + 2$

$$
\begin{aligned}
&=^2 2(2T(\tfrac{n/2}{2}) + 3\tfrac{n}{2} + 2) + 3n + 2\\
&=^2 2^2T(\tfrac{n}{2^2}) + 2\cdot 3n + 2^2 + 2\\
&=^3 2^2(2T(\tfrac{n/2^2}{2}) + 3\tfrac{n}{2^2} + 2) + 2\cdot 3n + 2^2 + 2\\
&=^3 2^2T(\tfrac{n}{2^3}) + 3\cdot 3n + 2^3 + 2^2 + 2\\
&=^i 2^iT(\tfrac{n}{2^i}) + i3n + \textstyle\sum_{j=1}^{i} 2^j\\
&=^i 2^iT(\tfrac{n}{2^i}) + i3n + 2^{i+1} - 2
\end{aligned}
$$

Al caso básico se llega cuando $\frac{n}{2^i} = 1 \Leftrightarrow i = log(n)$*:*

$T(n) = 2^{log(n)}T(1) + 3nlog(n) + 2^{1+log(n)} - 2 =$
$4n + 3nlog(n) + 2n - 2 \in \Theta(nlog(n))$

Ejercicio

Resuelve la siguiente recurrencia:

$$T(n) = \begin{cases} 1, & n = 1; \\ 6, & n = 2; \\ T(n-2) + 3n + 4, & n \geq 3. \end{cases}$$

Solución:

$T(n) = T(n-2) + 3n + 4$

$$
\begin{aligned}
&=^2 T(n-2-2) + 3(n-2) + 4 + 3n + 4 \\
&=^2 T(n-2\cdot 2) - 3\cdot 2 + 2\cdot(3n+4) \\
&=^3 T(n-2\cdot 2-2) + 3(n-2\cdot 2) + 4 - 3\cdot 2 + 2\cdot(3n+4) \\
&=^3 T(n-3\cdot 2) - 2(3\cdot 2) - 3\cdot 2 + 3\cdot(3n+4) \\
&=^q T(n-2q) - \textstyle\sum_{j=1}^{q-1} 6j + q(3n+4) \\
&=^q T(n-2q) - 3q(q-1) + q(3n+4)
\end{aligned}
$$

En este ejemplo tenemos dos casos base. Sin embargo, ambos tienen asociados un tiempo constante $\Rightarrow$ No influye en la complejidad:

- $n - 2q = 1 \Rightarrow q = \frac{n-1}{2} \Rightarrow T(n) \in \Theta(n^2)$.
- $n - 2q = 2 \Rightarrow q = \frac{n-2}{2} \Rightarrow T(n) \in \Theta(n^2)$.

Ejercicio

Resuelve la siguiente recurrencia:

$$T(n) = \begin{cases} 1, & \textit{si } n = 1; \\ 2T(n/2) + nlog(n), & \textit{si } n \geq 2. \end{cases}$$

Solución:

$T(n) = 2T(n/2) + nlog(n)$

$$\begin{aligned}
&=^2 2(2T(\tfrac{n/2}{2}) + \tfrac{n}{2}log(\tfrac{n}{2})) + nlog(n)\\
&=^2 2^2T(\tfrac{n}{2^2}) + nlog(\tfrac{n}{2}) + nlog(n)\\
&=^2 2^2T(\tfrac{n}{2^2}) + 2nlog(n) - n\\
&=^3 2^2(2T(\tfrac{n/2^2}{2}) + \tfrac{n}{2^2}log(\tfrac{n}{2^2}) + 2nlog(n) - n\\
&=^3 2^3T(\tfrac{n}{2^3}) + nlog(\tfrac{n}{2^2}) + 2nlog(n) - n\\
&=^3 2^3T(\tfrac{n}{2^3}) + 3nlog(n) - 2n - n\\
&=^q 2^qT(\tfrac{n}{2^q}) + qnlog(n) - n\textstyle\sum_{j=1}^{q-1} j\\
&=^q 2^qT(\tfrac{n}{2^q}) + qnlog(n) - n\tfrac{(q-1)q}{2}
\end{aligned}$$

Al caso básico se llega cuando: $\frac{n}{2^q} = 1 \Leftrightarrow q = log(n)$. Entonces tenemos:

$$T(n) = 2^{log(n)} + nlog(n)^2 - n\frac{(log(n)-1)(log(n))}{2} = \\ n + \frac{n}{2}(log(n))^2 + \frac{n}{2}log(n) \in \Theta(n(log(n))^2).$$

Ejercicio

Resuelve la siguiente recurrencia:

$$T(n) = \begin{cases} 1, & \textit{si } n = 1; \\ \sum_{i=1}^{n-1} T(i) + n^2, & \textit{si } n \geq 2. \end{cases}$$

Solución:

- El valor de T(n) depende de los valores: T(1),T(2),...,T(n-1).
- Intentamos reducir esta dependencia restando T(n-1) a T(n). Para $n \geq 3$ (para que se pueda aplicar el caso general de T(n-1)) tenemos:
  $T(n) - T(n-1) =$
  $$= \sum_{i=1}^{n-1} T(i) + n^2 - (\sum_{i=1}^{n-2} T(i) + (n-1)^2)$$
  $$= \sum_{i=1}^{n-2} T(i) + T(n-1) + n^2 - \sum_{i=1}^{n-2} T(i) - (n^2 - 2n + 1)$$
  $$= T(n-1) + 2n - 1$$

  luego es equivalente a:

  T(n)=2T(n-1)+2n-1

  Como anteriormente resolvimos: $T(n) \in \Theta(2^n)$.

Ejercicio

Resuelve la siguiente recurrencia:

$$T(n) = \begin{cases} 1, & \textit{si } n = 1; \\ 2\sum_{i=1}^{n-1} T(i) + 1, & \textit{si } n \geq 2. \end{cases}$$

- Seguimos el mismo planteamiento que para el problema anterior.
- Exigimos $n \geq 3$ para que se pueda aplicar el caso general de T(n-1)

$$T(n) - T(n-1) =$$

$$\begin{aligned}
&= 2\sum_{i=1}^{n-1} T(i) + 1 - (2\sum_{i=1}^{n-2} T(i) + 1) \\
&= 2\sum_{i=1}^{n-2} T(i) + 2T(n-1) + 1 - 2\sum_{i=1}^{n-2} T(i) - 1 \\
&= 2T(n-1)
\end{aligned}$$

luego es equivalente a:

$$T(n)=3T(n-1)$$

T(n)=3T(n-1)=

$$
\begin{aligned}
&=^2 3(3T(n-2)) = 3^2T(n-2)\\
&=^3 3^2(3T(n-3)) = 3^3T(n-3)\\
&=^q 3^qT(n-q)
\end{aligned}
$$

Al caso básico llegamos cuando se da $n-q=1$, luego:
$T(n) = 3^{n-1}T(1) = 3^{n-1} \in \Theta(3^n)$.

Ejercicio

Resuelve la siguiente recurrencia:

$$T(n) = \begin{cases} 1, & \textit{si } n = 1; \\ \sum_{i=1}^{n-1} T(n-i) + 1, & \textit{si } n \geq 2. \end{cases}$$

- **Idea clave:** Es lo mismo sumar desde n-1 a 1 que desde 1 a n-1:
$\sum_{i=1}^{n-1} T(n-i) =$
$$= T(n-1) + T(n-2) + ... + T(2) + T(1)$$
$$= T(n-1) + T(n-2) + ... + T(2) + T(1)$$
$$= T(1) + T(2) + ... + T(n-2) + T(n-1)$$
$$= \sum_{j=1}^{n-1} T(j)$$

- Igual que en los casos anteriores restamos T(n-1) a T(n) para $n \geq 3$: T(n)-T(n-1)=
$$= \sum_{j=1}^{n-1} T(j) + 1 - (\sum_{j=1}^{n-2} T(j) + 1)$$
$$= \sum_{j=1}^{n-2} T(j) + T(n-1) + 1 - \sum_{j=1}^{n-2} T(j) - 1$$
$$= T(n-1)$$

Luego: T(n)=2T(n-1).

T(n)=2T(n-1)=

$$=^2 2(2T(n-2)) = 2^2T(n-2)$$
$$=^3 2^2(2T(n-3)) = 2^3T(n-3)$$
$$=^q 2^qT(n-q)$$

Caso base: n-i=1 $\Rightarrow T(n) = 2^{n-1} \in \Theta(2^n)$.

Ejercicio

Calcula la complejidad de la siguiente función:

fun *funcion fib(N)*
  si $N = 0 \vee N = 1$ **entonces**
    $Fib \leftarrow N$
  si no
    $Fib \leftarrow Fib(N-1) + Fib(N-2)$
  fin si
  devolver *F ib*
fin fun

Solución:

$$T(n) = \begin{cases} c, & \text{si n} \leq 1 \text{ ;} \\ T(n-1) + T(n-2) + c, & \text{si n} > 1. \end{cases}$$

Es un caso particular del ejemplo anterior, luego: $T(n) \in \Theta(2^n)$.

Ejercicio

Dado el siguiente algoritmo, determina su complejidad:

fun *calcula(n)*
  si $n > 1$ **entonces**
    aux ← *1*
    desde *i* ← *1* **hasta** *n-1* **hacer**
      aux ← *aux**calcula(i)*
    fin desde
  si no
    aux ← *8*
  fin si
  devolver *a ux*
fin fun

Solución:

$$T(n) = \begin{cases} c, & \text{si n} = 1\ ; \\ \sum_{i=1}^{n-1} T(i) + c, & \text{si n} > 1. \end{cases}$$

$$T(n) - T(n-1) = \sum_{i=1}^{n-1} T(i) + c - (\sum_{i=1}^{n-2} T(i) + c) =$$
$$T(n-1) + \sum_{i=1}^{n-2} T(i) + c - (T \sum_{i=1}^{n-2} T(i) + c)$$
$$\Rightarrow T(n) = 2T(n-1) \Rightarrow T(n) \in \Theta(2^n)$$

Ejercicio

Dado el siguiente algoritmo, determina su complejidad:

```
fun calcular(L[1..n])
   devolver
   calcula(L[1..n],1,n)
fin fun
fun calcula(L[1..n],iz,de)
   si de-iz > 0 entonces
      desde i ← 1 hasta de-iz+1 hacer
         aux ← aux+2
      fin desde
      aux ← aux+calcula(L[1..n],iz,de-1)
      aux ← aux+calcula(L[1..n],iz+1,de)
   si no
      aux ← 1
   fin si
   devolver a ux
fin fun
```

Solución:

m (=de-iz+1) es la longitud de la lista procesada en cada iteración.

$$T(m) = \begin{cases} c, & \text{si m} = 1\ ; \\ 2T(m-1) + cm + c, & \text{si m} > 1. \end{cases}$$

Luego $T(m) = 2T(m-1) \Rightarrow T(m) \in \Theta(2^m)$

- Sea $P(x)$ un polinomio de grado n: $P(x) = \sum_{i=0}^{n} c_i * x^i$.
- Se verifica $P(x) = \prod_{i=0}^{n}(x - r_i)$ donde $\{r_i\}$ es el conjunto de raíces y $r_i \in \mathbb{C}$.
- Puede ocurrir que alguna valor de $\{r_i\}$ esté repetido.
- Entonces:

$$P(x) = \prod_{\forall i}(x - r_i)^{\alpha_i}$$

  donde α_i es el número de repeticiones de r_i.
- A α_i se le denomina multiplicidad de r_i.

- Consideremos la siguiente expresión:

$$a_0 T(n) + a_1 T(n-1) + \ldots + a_k T(n-k) = 0 \tag{1}$$

- Si $G(n)$ y $F(n)$ son soluciones de 1, entonces una combinación lineal de ambas: $\lambda F(n) + \mu G(n)$ también es solución.
- Buscamos soluciones con la forma $T(n) = X^n$ con X constante:

$$a_0 X^n + a_1 X^{n-1} + \ldots + a_k X^{n-k} = 0$$
$$\Downarrow$$
$$X^{n-k}(a_0 X^k + a_1 X^{k-1} + \ldots + a_k) = 0$$
$$\Downarrow$$
$$\text{Polinomio característico} \equiv a_0 X^k + a_1 X^{k-1} + \ldots + a_k = 0$$

- Sean $\{r_i\}$ son raíces de: $a_0 X^k + a_1 X^{k-1} + \ldots + a_k = 0$, entonces

$$T(n) = \sum_{\forall i} cte * r_i^n$$

  son soluciones de 1.
- Se puede demostrar que todas las soluciones son de esta forma.

Ejemplo

Resuelve la siguiente recurrencia:

$$T(n) = \begin{cases} T(n-1) + T(n-2), & \textit{si } n > 1. \\ n, & \textit{en caso contrario ;} \end{cases}$$

- $T(N) - T(n-1) + T(n-2) = 0$
- Polinomio característico: $x^2 - x - 1$
- Raíces: $r_1 = \frac{1+\sqrt{5}}{2}$, $r_2 = \frac{1-\sqrt{5}}{2}$
- $T(n) = c_1 * (\frac{1+\sqrt{5}}{2})^n + c_1 * (\frac{1-\sqrt{5}}{2})^n$, luego $T(n) \in \Theta(cte^n)$.

- Consideremos el polinomio característico:

$$a_0 T(n) + a_1 T(n-1) + \ldots + a_k T(n-k) = 0$$

- Sea $\{r_i\}$ el conjunto de raíces con multiplicidad m_i.
- Entonces:

$$T(n) = \sum_{\forall i} \sum_{q=1}^{m_i} cte_q n^{q-1} r_i^n \tag{2}$$

- Si $m_i = 1$ entonces $cte_q n^0 r_i^n$ se queda en: $cte_q r_i^n$

Ejemplo

Resuelve la siguiente recurrencia:

$$T(n) = \begin{cases} 5T(n-1) - 8T(n-2) + 4T(n-3), & \textit{si } n > 2. \\ n, & \textit{en caso contrario ;} \end{cases}$$

- $T(n) - 5T(n-1) + 8T(n-2) - 4T(n-3) = 0$
- Polinomio característico: $x^3 - 5x^2 + 8x - 4 = (x-1)(x-2)^2$
- Raíces: $r_1 = 1(m = 1)$, $r_2 = 2(m = 2)$
- $T(n) = c_1 * 1^n + c_2 * 2^n + c_3 * n * 2^n$, luego $T(n) \in \Theta(n2^n)$.

- Consideremos la recurrencia:

$$a_0 T(n) + a_1 T(n-1) + \ldots + a_k T(n-k) = b^n p(n)$$

- donde $b = cte$ y $d =$ grado de $p(n)$.
- Es equivalente a resolver el siguiente una ecuación de recurrencia lineal homogénea con el siguiente polinomio característico:

$$(a_0 X^k + a_1 X^{k-1} + \ldots + a_k)(X - b)^{d+1} = 0$$

- En general, dada la ecuación de recurrencia lineal y no homogénea:

$$a_0 T(n) + a_1 T(n-1) + \ldots + a_k T(n-K) = b_1^n p_1(n) + b_2^n p_2(n) + \ldots$$

Se puede demostrar que es equivalente a una ecuación de recurrencia lineal y homogénea, cuyo polinomio característico sea:

$$(a_0 X^k + a_1 X^{k-1} + \ldots + a_k)(X - b_1)^{d_1+1}(X - b_2)^{d_2+1} \ldots$$

donde d_1 y d_2 son los grados de los polinomios $p_1(n)$ y $p_2(n)$.

Ejemplo

Resuelve la siguiente recurrencia:

$$T(n) = \begin{cases} 2T(n-1) + n, & \textit{si } n > 1. \\ c, & \textit{en caso contrario ;} \end{cases}$$

- $T(n-1) - 2T(n-1) = n$
- $b = 1$, $p(n) = n$
- Polinomio característico: $(x-1)^2(x-2)$
- Raíces: $r_1 = 1(m = 2)$, $r_2 = 2(m = 1)$
- $T(n) = c_1 * 2^n + c_2 * 1^n + c_3 * n * 1^n$, luego $T(n) \in \Theta(2^n)$.