Resolución de Recurrencias

Juan Manuel Rabasedas

Análisis de Algoritmos

Vimos que para el "Análisis de Algoritmos" vamos a utilizar:

- Análisis Asintótico.
- Modelo de costo basado en Lenguaje.
- Métricas de Work (W) y Span (S)

Cuando usamos todo esto obtenemos recurrencias que necesitamos resolver.

Vamos a ver algunos métodos para resolverlas.

Método de Sustitución

En este método:

1. Adivinamos la solución.
2. Probamos que es correcta con inducción matemática.

Veamos a ver un ejemplo:

$W(0) = k_0$

$W(1) = k_1$

$W(n) = 2\ W(\lfloor \frac{n}{2} \rfloor) + k_2\ n$

- Adivinamos que $W(n) \in O(n\ logn)$
- Probamos que $W(n) \leqslant c\ n\ logn$
- Para probarlo usamos inducción:
    - Supongo que la expresión es válida para cualquier valor menor a n y luego pruebo que es válida para n

Método de Sustitución

- Supongo que vale para $\left\lfloor \frac{n}{2} \right\rfloor$,
- luego mi **HI** es

$$W(\left\lfloor \frac{n}{2} \right\rfloor) \leqslant c \left\lfloor \frac{n}{2} \right\rfloor \; log \left\lfloor \frac{n}{2} \right\rfloor$$

$$\begin{aligned}
W(n) \quad & = 2 \; W(\left\lfloor \tfrac{n}{2} \right\rfloor) + k_2 \; n \\
& \leqslant 2 \; (c \left\lfloor \tfrac{n}{2} \right\rfloor \; log \left\lfloor \tfrac{n}{2} \right\rfloor) + k_2 \; n \\
& \leqslant c \; n \; log \; \tfrac{n}{2} + k_2 \; n \\
& = c \; n \; log \; n - c \; n \; log \; 2 + k_2 \; n \\
& = c \; n \; log \; n - c \; n + k_2 \; n \\
& \leqslant c \; n \; log \; n \qquad\qquad cuando \;\; c \geqslant k_2
\end{aligned}$$

Método de Sustitución

Faltan chequear si vale para los casos base:

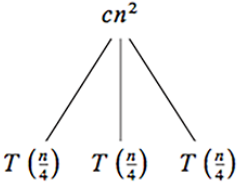
- $W(0) \leqslant c\ 0\ lg\ 0$
- $W(1) \leqslant c\ 1\ lg\ 1$

Solo necesito que valga para $\forall n \geqslant n_0$.
Tomando $n_0 = 2$
$W(2) = 2\ W(1) + 2\ c_2 = 2\ c_1 + 2\ c_2 \leqslant c\ 2\ lg\ 2 = 2c$
$W(3) = 2\ W(1) + 3\ c_2 = 2\ c_1 + 3\ c_2 \leqslant c\ 3\ lg\ 3$

Elegimos c suficientemente grande.

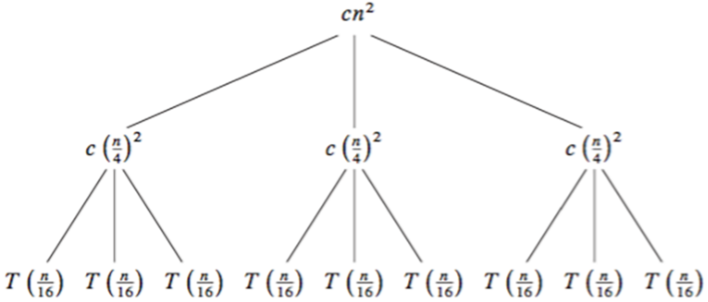
Árbol de Recurrencia

Vamos a ver este método sobre un ejemplo:
$T(1) = c_1$
$T(n) = 3T(n/4) + cn^2$
Representamos la recursión con un árbol,



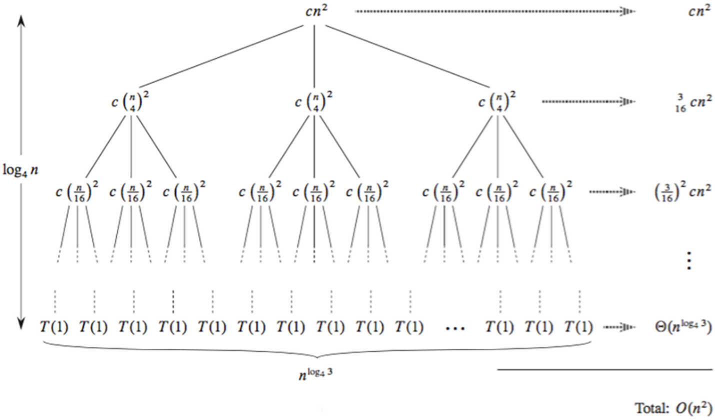
donde la raíz tiene consto cn^2 porque la primer llamada a la función tiene cn^2 unidades de trabajo.
Luego las ramas se corresponden con cada una de las llamadas recursivas.

Árbol de Recurrencia



Los nodos en el segundo nivel tienen costo $c(\frac{n}{4})^2$ porque la función ahora es llamada para un problema de tamaño $\frac{n}{4}$
Se sigue expandiendo el árbol hasta llegar a un caso base.

Árbol de Recurrencia



El último nivel del árbol es un caso especial (caso base).
Sumando las operaciones por nivel y luego sumamos el costo de cada nivel obtenemos el costo total.

Árbol de Recurrencia

Luego el costo del árbol lo podemos calcular como:

$$
\begin{aligned}
T(n) &= cn^2 + \tfrac{3}{16}cn^2 + (\tfrac{3}{16})^2 cn^2 + \ldots + (\tfrac{3}{16})^{log_4 n-1} cn^2 + c_1 n^{log_4 3} \\
&= \textstyle\sum_{i=0}^{log_4 n-1} (\tfrac{3}{16})^i cn^2 + c_1 n^{log_4 3} \\
&= \tfrac{(3/16)^{log_4 n}-1}{(3/16)-1} cn^2 + c_1 n^{log_4 3} \\
&\in O(n^2)
\end{aligned}
$$

Este método nos proporciona un resultado exacto si no hacemos acotaciones en el procedimiento ni descartamos valores.
Generalmente se usa para obtener un candidato (adivinar) y luego usar el método de sustitución.

Funciones Suaves

- Cuando analizamos recurrencias pueden aparecer $\lfloor \cdot \rfloor$ y $\lceil \cdot \rceil$
- Nos gustaría poder sacarlas fácilmente.
- Esto es posible cuando $n = b^k$, es decir luego $n/b = \lfloor n/b \rfloor = \lceil n/b \rceil$
- No siembre el comportamiento asintótico de la función para $n = b^k$ será el mismo para que para cualquier entrada,
- pero en algunas funciones denominadas suaves (smooth) ocurre este comportamiento.
- Funciones suaves: n^r, $n \; log \; n$, $n^2 \log \; n$, $n \log^2 \; n$, etc.
- Funciones NO suaves: 2^n, $n^{log \; n}$, $n!$, etc.
- Una función $f : \mathbb{N} - > \mathbb{R}^+$ es b-suave (smooth) si:
  1. es eventualmente no decreciente y
  2. $f(bn) \in O(f(n))$

  Luego si f es b-suave para **un** $b \geqslant 2$, entonces es suave.
- Luego si f es suave: $g(b^k) \in \Theta(f(b^k)) \Rightarrow g(n) \in \Theta(f(n)) \; \forall b \geqslant 2$

Funciones Suaves

Veamos un ejemplo:

- Si tenemos que calcular una recurrencia W de la forma:
  $W(1) = c_1$
  $W(n) = W(\lfloor n/2 \rfloor) + W(\lceil n/2 \rceil) + c_2 n$
- Podemos sacar $\lfloor \cdot \rfloor$ y $\lceil \cdot \rceil$ y resolver W^s, para $n = 2^k$:
  $W^s(1) = c_1$
  $W^s(n) = 2W^s(n/2) + c_2 n$
- como $W^s \in O(n\ log\ n)$, y $n\ log\ n$ es suave
  entonces $W \in O(n\ log\ n)$

Teorema Maestro

Muchas recurrencias que derivan de algoritmos tienen la forma:

$$T(n) = a\ T(\frac{n}{b}) + f(n)$$

- En muchos casos podemos resolver este tipo de recurrencias utilizando “El teorema maestro”
- Existen varias versiones en la literatura de este teorema.

Teorema Maestro

Sean $a, b \in \mathbb{R}$, $a \geqslant 1$, $b > 1$ y

$$T(n) = aT(n/b) + f(n)$$

$$T(n) = \begin{cases} & \exists\, \epsilon > 0 \cdot f(n) \in \Omega(n^{\log_b a + \epsilon}) \wedge \\ \Theta(f(n)) & \text{si} \quad \exists\, c \in \mathbb{R}, n_0 \in \mathbb{N} \cdot c < 1 \wedge \\ & \forall\, n \geqslant n_0 \cdot a\, f(n/b) \leqslant c\, f(n) \\ \Theta(n^{\log_b a} \lg n) & \text{si } f(n) \in \Theta(n^{\log_b a}) \\ \Theta(n^{log_b a}) & \text{si } \exists\, \epsilon > 0 \cdot f(n) \in O(n^{\log_b a - \epsilon}) \end{cases}$$

Para nuestro ejemplo

$$T(n) = 3T(n/4) + cn^2$$

con $a = 3$, $b = 4$ y $f(n) = n^2$, es el primer caso ya que si tomamos $\epsilon \cong 0.2$ luego $log_4\ 3 + 0.2 \cong 1$ por lo tanto $T(n) \in \Theta(n^2)$

Teorema Maestro

Otra versión del "Teorema maestro":
Sean $a, b, c \in \mathbb{R}$, $a \geqslant 1$, $b > 1$ y

$$T(n) = aT(n/b) + n^c$$

$$T(n) = \begin{cases} O(n^c) & a < b^c \\ \Theta(n^c \lg n) & a = b^c \\ \Theta(n^{log_b a}) & a > b^c \end{cases}$$

En el primer caso solamente se puede probar una cota superior, no Θ. Para probar una cota ajustada haría falta agregar una condición que es equivalente a la condición del teorema enunciado anteriormente.

En ninguno de las dos versiones del teorema se cubren todos los casos posibles.

Referencias

- Introduction to Algorithms. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Cliford Stein.
- Alfred V. Aho, J. E. Hopcroft, and Jeffrey D. Ullman. The Design and Analysis of Computer Algorithms. Addison-Wesley, 1974
- Algorithmics Theory and Practice, Gilles Brassard, Paul Bratley.