Análisis y Diseño de Algoritmos

Teorema Maestro

Arturo Díaz Pérez

Introducción

☞ Recurrencia general para estrategias divide y vencerás

$$T(1) = 1$$
$$T(n) = aT\left(\tfrac{n}{b}\right) + d(n)$$

Teorema Maestro

- Sean $a \geq 1$ y $b \geq 1$ constantes, sea $d(n)$ una función y sea $T(n)$ definido en los enteros no negativos por la recurrencia

$$T(n) = aT\left(\tfrac{n}{b}\right) + d(n)$$

$T(n)$ puede ser acotado asintóticamente como sigue:

1 Si $d(n) = O(n^{\log_b a-\varepsilon})$ para alguna constante $\varepsilon > 0$, entonces, $T(n) = \Theta(n^{\log_b a})$.

2 Si $d(n) = \Theta(n^{\log_b a})$, entonces, $T(n) = \Theta(n^{\log_b a}\lg n)$.

3 Si $d(n) = \Omega(n^{\log_b a+\varepsilon})$ para alguna constante $\varepsilon > 0$ y si $ad(n/b) \leq cd(n)$ para alguna constante $c < 1$ y todas las n suficientemente grandes, entonces, $T(n) = \Theta(d(n))$.

Análisis y Diseño de Algoritmos — *Maestro-3*

Observaciones (1)

- $d(n)$ no solo tiene que ser menor que $n^{\log_b a}$, debe ser polinomialmente menor.
  - ← Esto es, $d(n)$ debe ser asintóticamente menor que $n^{\log_b a}$ por un factor de n^{ε} para alguna constante $\varepsilon > 0$.

- $d(n)$ no solo tiene que ser mayor que $n^{\log_b a}$, debe ser polinomialmente mayor y además satistacer la condición de regularidad:
  - ← $ad(n/b) \leq cd(n)$ para alguna constante $c < 1$ y todas las $n \geq n_0$.

- Si las dos funciones son del mismo orden, se multiplica por un factor logarítmico y la solución es $T(n)=\Theta(n^{\log_b a}\log n)$.

Análisis y Diseño de Algoritmos — *Maestro-4*

Observaciones (2)

- Los casos anteriores no cubren todos los casos para $d(n)$.
- Hay un hueco entre los casos 1 y 2 cuando $d(n)$ es menor pero no polinomialmente menor que $n^{\log_b a}$.
- Hay un hueco entre los casos 2 y 3 cuando $d(n)$ es mayor pero no polinomialmente mayor que $n^{\log_b a}$.

Ejemplos (1)

- Sea $T(n) = 9T(n/3) + n$.
  - $a = 9$, $b = 3$, $d(n) = n$
  - $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$
  - Ya que $d(n) = O(n^{\log_3 9-\varepsilon})$, para $\varepsilon = 1$, se puede aplicar el Caso 1
  - Por lo tanto, $T(n) = \Theta(n^2)$.
- Sea $T(n) = T(2n/3) + 1$.
  - $a = 1$, $b = 3$, $d(n) = 1$
  - $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$
  - Ya que $d(n) = \Theta(n^{\log_{3/2} 1})$, se puede aplicar el Caso 2
  - Por lo tanto, $T(n) = \Theta(\lg n)$.

Ejemplos (2)

- ☞ Sea $T(n) = 3T(n/4) + n\lg n$.
  - ← $a = 3$, $b = 4$, $d(n) = n\lg n$
  - ← $n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$
  - ← Ya que $d(n) = \Omega(n^{\log_4 3+\varepsilon})$, para $\varepsilon \approx 0.2$, se puede aplicar el Caso 3 si se muestra que la condición de regularidad se presenta para $d(n)$.
  - ← Para un n suficientemente grande $ad(n/b) = 3(n/4)\lg(n/4) \leq (3/4)n\lg n = cd(n)$, para $c = 3/4$.
  - ← Por lo tanto, $T(n) = \Theta(n\lg n)$.

Análisis y Diseño de Algoritmos *Maestro*-7

Ejemplos (3)

- ☞ Sea $T(n) = 2T(n/2) + n\lg n$.
  - ← $a = 2$, $b = 2$, $d(n) = n\lg n$
  - ← $n^{\log_b a} = n^{\log_2 2} = n$
  - ← Ya que $d(n) = n\lg n$ es asintóticamente más grande que n, se puede suponer que el Caso 3 se aplica.
  - ← Sin embargo, $n\lg n$ no es asintóticamente polinomialmente más grande que n.
  - ← El cociente $d(n) = n^{\log_b a} = n\lg n/n = \lg n$ es asintóticamente menor que n^{ε}, para cualquier constante positiva ε.
  - ← Así, esta recurrencia cae en el hueco entre los Casos 2 y 3.

Análisis y Diseño de Algoritmos *Maestro*-8

Lema 1

☞ Sea

$$T(n)=\begin{cases}\Theta(1) & \text{si } n=1\\ aT\left(\frac{n}{b}\right)+d(n) & \text{si } \exists k: n=b^k\end{cases}$$

entonces,

$$T(n)=\Theta(n^{\log_b a})+\sum_{j=0}^{\log_b n-1} a^j d\left(\frac{n}{b^j}\right)$$

Demostración. Ya se hizo en las notas anteriores. Láminas *Análisis*-(24-26)

Análisis y Diseño de Algoritmos *Maestro*-9

Lema 2

☞ Sean a ³ 1 y b ³ 1 constantes y sea $d(n)$ una función no negativa definida sobre las potencias exactas de b. La función g

$$g(n)=\sum_{j=0}^{\log_b n-1} a^j d\left(\frac{n}{b^j}\right)$$

puede ser acotado asintóticamente como sigue:

1 Si $d(n) = O(n^{\log_b a-\varepsilon})$ para alguna constante $\varepsilon > 0$, entonces, $g(n) = O(n^{\log_b a})$.
2 Si $d(n) = \Theta(n^{\log_b a})$, entonces, $g(n) = \Theta(n^{\log_b a}\lg n)$.
3 Si $ad(n/b) \leq cd(n)$ para alguna constante $c < 1$ y para $n \geq b$, entonces, $g(n) = \Theta(d(n))$.

Análisis y Diseño de Algoritmos *Maestro*-10

Demostración Caso 1

☞ Si $d(n) = O(n^{\log_b a-\varepsilon})$, entonces, $d(n/b^i) = O((n/b^i)^{\log_b a-\varepsilon})$

$$g(n) = O\left(\sum_{j=0}^{\log_b n-1} a^j d\left(\frac{n}{b^j}\right)^{\log_b a-e}\right)$$

$$\sum_{j=0}^{\log_b n-1} a^j d\left(\frac{n}{b^j}\right)^{\log_b a-e} = n^{\log_b a-e}\sum_{j=0}^{\log_b n-1}\left(\frac{ab^e}{b^{\log_b a}}\right)^j$$

$$= n^{\log_b a-e}\sum_{j=0}^{\log_b n-1}(b^e)^j$$

$$= n^{\log_b a-e}\left[\frac{(b^e)^{\log_b n}-1}{b^e-1}\right]$$

$$= n^{\log_b a-e}\left[\frac{n^e-1}{b^e-1}\right]$$

Análisis y Diseño de Algoritmos *Maestro*-11

Demostración Caso 1 (cont.)

$$\sum_{j=0}^{\log_b n-1} a^j d\left(\frac{n}{b^j}\right)^{\log_b a-e} = n^{\log_b a-e}\left[\frac{n^e-1}{b^e-1}\right]$$

☞ Ya que b y ε son constantes

$$g(n) = n^{\log_b a-e}O(n^e) = O(n^{\log_b a})$$

Análisis y Diseño de Algoritmos *Maestro*-12

Demostración Caso 2

- Si $d(n) = \Theta(n^{\log_b a})$, entonces, $d(n/b^i) = \Theta((n/b^i)^{\log_b a})$

$$g(n) = \Theta\left(\sum_{j=0}^{\log_b n-1} a^j \left(\frac{n}{b^j}\right)^{\log_b a}\right)$$

$$\sum_{j=0}^{\log_b n-1} a^j \left(\frac{n}{b^j}\right)^{\log_b a} = n^{\log_b a} \sum_{j=0}^{\log_b n-1} \left(\frac{a}{b^{\log_b a}}\right)^j$$

$$= n^{\log_b a} \sum_{j=0}^{\log_b n-1} 1$$

$$= n^{\log_b a} \log_b n$$, por lo tanto

$$g(n) = \Theta(n^{\log_b a} \log_b n) = \Theta(n^{\log_b a} \lg n)$$

Análisis y Diseño de Algoritmos *Maestro*-13

Demostración Caso 3

- Dado que $ad(n/b) \leq cd(n)$ para alguna constante $c < 1$ y para $n \geq b$, e, se tiene que, $a^j d(n/b^j) \leq c^j d(n)$.

$$g(n) = \sum_{j=0}^{\log_b n-1} a^j d\left(\frac{n}{b^j}\right)^{\log_b a}$$

$$g(n) \leq \sum_{j=0}^{\log_b n-1} c^j d(n)$$

$$g(n) \leq d(n) \sum_{j=0}^{\infty} c^j$$

$$= d(n) \frac{1}{1-c}$$, por lo tanto

$$g(n) = O(d(n))$$, de aquí, ya que c es constante

$$g(n) = \Theta(d(n))$$

Análisis y Diseño de Algoritmos *Maestro*-14

Lema 3

- Sean $a \geq 1$ y $b \geq 1$ constantes, sea $d(n)$ una función y sea $T(n)$ definido en los enteros no negativos potencias de b por la recurrencia

$$T(n) = aT\left(\tfrac{n}{b}\right) + d(n)$$

$T(n)$ puede ser acotado asintóticamente como sigue:

1 Si $d(n) = O(n^{\log_b a - \varepsilon})$ para alguna constante $\varepsilon > 0$, entonces, $T(n) = \Theta(n^{\log_b a})$.

2 Si $d(n) = \Theta(n^{\log_b a})$, entonces, $T(n) = \Theta(n^{\log_b a} \lg n)$.

3 Si $d(n) = \Omega(n^{\log_b a + \varepsilon})$ para alguna constante $\varepsilon > 0$ y si $ad(n/b) \leq cd(n)$ para alguna constante $c < 1$ y todas las n suficientemente grandes, entonces, $T(n) = \Theta(d(n))$.

Análisis y Diseño de Algoritmos *Maestro*-15

Demostración Lema 3

- Por el Lema 1

$$T(n) = \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j d\left(\frac{n}{b^j}\right)$$

- Por el Lema 2
- Caso 1

$$T(n) = \Theta(n^{\log_b a}) + O(n^{\log_b a}) = \Theta(n^{\log_b a})$$

- Caso 2

$$T(n) = \Theta(n^{\log_b a}) + \Theta(n^{\log_b a} \lg n) = \Theta(n^{\log_b a} \lg n)$$

- Caso 3

$$T(n) = \Theta(n^{\log_b a}) + \Theta(d(n))$$

← pero $d(n) = \Omega(n^{\log_b a + \varepsilon})$, por lo tanto

$$T(n) = \Theta(d(n))$$

Análisis y Diseño de Algoritmos *Maestro*-16

Observación al Lema 3

- El Lema 3 es válido únicamente cuando n es una potencia exacta de b.
- Tarea. Demostrar que el Teorema Maestro es válido para cuando n no es necesariamente un potencia de b.
  - ← Cormen *et al*. págs. 70-72.

Análisis y Diseño de Algoritmos *Maestro*-17