Contents

- Introduction to advanced data structures:
- Introduction/Fundamentals of the analysis of algorithms
- Recurrences:
  - The substitution method
  - Recursive tree method
  - Masters method

Master Theorem

- In the analysis of algorithms, the master theorem for divide-and-conquer recurrences provides an asymptotic analysis (using Big O notation) for recurrence relations of types that occur in the analysis of many divide and conquer algorithms.

Theorem 4.1 (Master theorem)

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

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

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a-\epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a+\epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

Master Theorem

The master method applies to recurrences of the form

$$T(n) = a\,T(n/b) + f(n)\ ,$$

where $a \geq 1$, $b > 1$, and f is asymptotically positive.

Idea of master theorem

Recursion tree:

$h = \log_b n$

$$f(n) \quad\cdots\cdots\quad f(n)$$

$$a$$

$$f(n/b) \quad f(n/b) \quad \cdots \quad f(n/b) \quad\cdots\quad a\,f(n/b)$$

$$a$$

$$f(n/b^2) \quad f(n/b^2) \quad \cdots \quad f(n/b^2) \quad\cdots\quad a^2 f(n/b^2)$$

$$\vdots$$

$$T(1) \qquad\qquad n^{\log_b a}\,T(1)$$

$$\#\text{leaves} = a^h = a^{\log_b n} = n^{\log_b a}$$

Three common cases

Compare $f(n)$ with $n^{\log_b a}$:

1. $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$.
   - $f(n)$ grows polynomially slower than $n^{\log_b a}$ (by an n^{ε} factor).

   Solution: $T(n) = \Theta(n^{\log_b a})$.

Idea of master theorem

Recursion tree:

$h = \log_b n$

$$f(n) \quad \cdots\cdots \quad f(n)$$

a

$$f(n/b) \quad f(n/b) \quad \cdots \quad f(n/b) \quad \cdots\cdots \quad a\,f(n/b)$$

a

$$f(n/b^2) \quad f(n/b^2) \quad \cdots \quad f(n/b^2) \quad \cdots\cdots \quad a^2 f(n/b^2)$$

$$\vdots$$

$$T(1) \qquad\qquad n^{\log_b a}\,T(1)$$

$$\Theta(n^{\log_b a})$$

CASE 1: The weight increases geometrically from the root to the leaves. The leaves hold a constant fraction of the total weight.

Three common cases

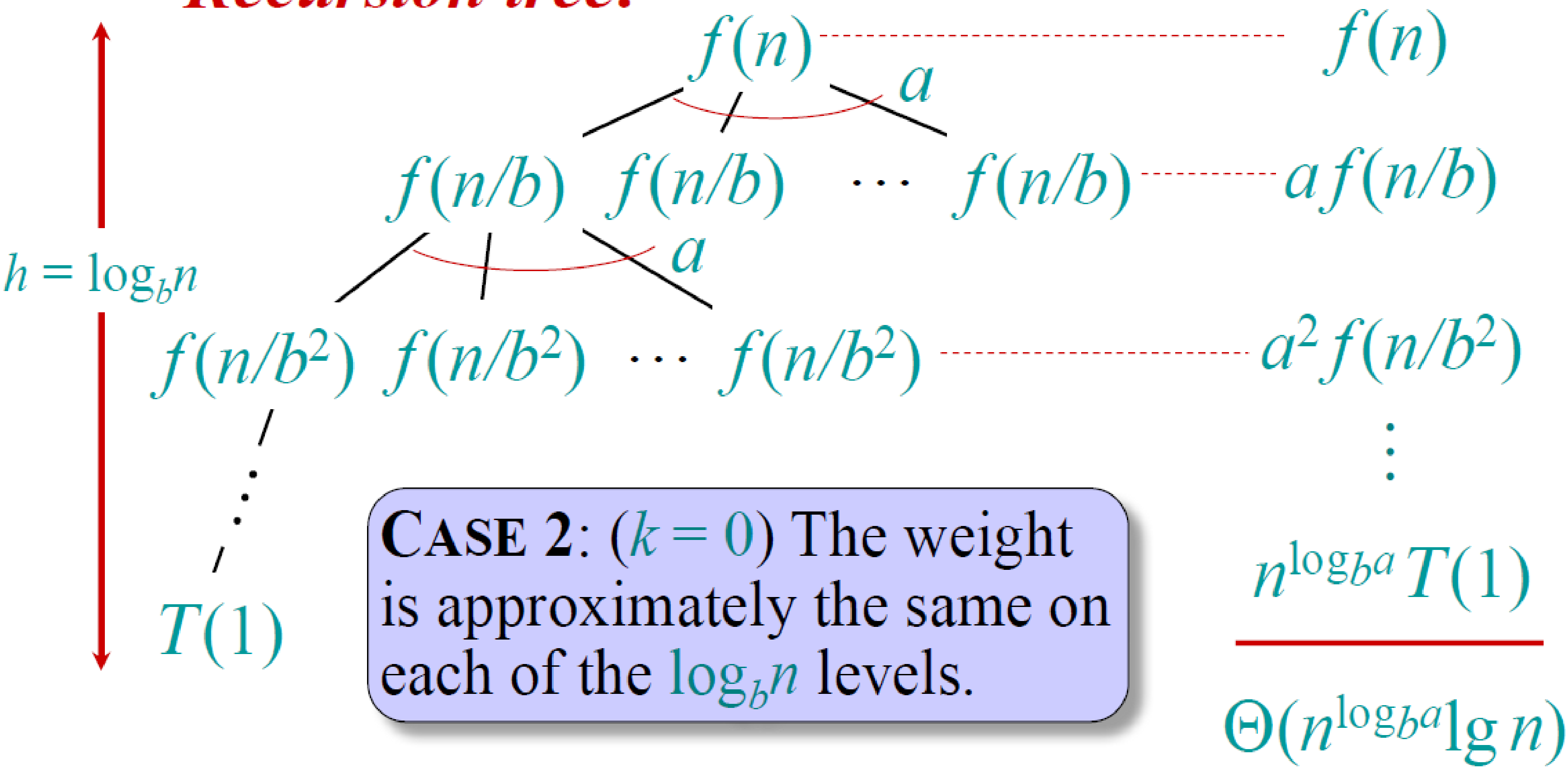
Compare $f(n)$ with $n^{\log_b a}$:

2. $f(n) = \Theta(n^{\log_b a} \lg^k n)$ for some constant $k \geq 0$.
   - $f(n)$ and $n^{\log_b a}$ grow at similar rates.

   Solution: $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.

Idea of master theorem

Recursion tree:



CASE 2: ($k = 0$) The weight is approximately the same on each of the $\log_b n$ levels.

Three common cases (cont.)

Compare $f(n)$ with $n^{\log_b a}$:

3. $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$.
   - $f(n)$ grows polynomially faster than $n^{\log_b a}$ (by an n^{ε} factor),

   and $f(n)$ satisfies the ***regularity condition*** that $a f(n/b) \leq c f(n)$ for some constant $c < 1$.

   Solution: $T(n) = \Theta(f(n))$.

Idea of master theorem

Recursion tree:

$f(n)$ .... $f(n)$

a

$f(n/b)\quad f(n/b)\quad \cdots\quad f(n/b)$ .... $a\,f(n/b)$

a

$h = \log_b n$

$f(n/b^2)\quad f(n/b^2)\quad \cdots\quad f(n/b^2)$ .... $a^2 f(n/b^2)$

$\vdots$

$T(1)$ $\qquad$ $n^{\log_b a}\,T(1)$

CASE 3: The weight decreases geometrically from the root to the leaves. The root holds a constant fraction of the total weight.

$\Theta(f(n))$

Applications

$k = \log_2 3; f(n) = n^2$
$n^2 = \Omega\left(n^{\log_2 3 + (2 - \log_2 3)}\right)$
$3 \cdot \left(\frac{n}{2}\right)^2 \leq \frac{3}{4} \cdot n^2$

$k = \log_2 1 = 0; f(n) = 2^n$
$2^n = \Omega\left(n^{0+\log 2}\right)$
$1 \cdot 2^{\frac{n}{2}} \leq \frac{1}{2} \cdot 2^n$

T(n) = 3 * T(n/2) + n^2
⇒ T(n) = $\Theta(n^2)$ (case 3)

T(n) = T(n/2) + 2^n
⇒ T(n) = $\Theta(2^n)$ (case 3)

* For sure question on this in final *

T(n) = 16 * T(n/4) + n
⇒ T(n) = $\Theta(n^2)$ (case 1)

$k = \log_4 16 = 2; f(n) = n$
$n = O(n^{2-1})$

T(n) = 2 * T(n/2) + n log n
⇒ T(n) = n $\log^2$ n (case 2)

$k = \log_2 2 = 1; f(n) = n \log n$
$n \log n = \Theta(n^1 log^1 n)$

T(n) = 2^n * T(n/2) + n^n
⇒ Does not apply!!

ex. when master theorem cannot be applied!

Examples

Ex. $T(n) = 4T(n/2) + n$

$a = 4$, $b = 2 \Rightarrow n^{\log_b a} = n^2$; $f(n) = n$.

CASE 1: $f(n) = O(n^{2-\varepsilon})$ for $\varepsilon = 1$.

$\therefore T(n) = \Theta(n^2)$.

Ex. $T(n) = 4T(n/2) + n^2$

$a = 4$, $b = 2 \Rightarrow n^{\log_b a} = n^2$; $f(n) = n^2$.

CASE 2: $f(n) = \Theta(n^2 \lg^0 n)$, that is, $k = 0$.

$\therefore T(n) = \Theta(n^2 \lg n)$.

Another way for Master's Theorem
General Divide-and-Conquer Recurrence

$T(n) = aT(n/b) + f(n)$ where $f(n) \in \Theta(n^d)$, $d \geq 0$

Master Theorem: If $a < b^d$, $T(n) \in \Theta(n^d)$

If $a = b^d$, $T(n) \in \Theta(n^d \log n)$

If $a > b^d$, $T(n) \in \Theta(n^{\log_b a})$

Note: The same results hold with O instead of Θ.

Examples: $T(n) = 4T(n/2) + n \Rightarrow T(n) \in ?$ Θ(n^2)

$T(n) = 4T(n/2) + n^2 \Rightarrow T(n) \in ?$ Θ(n^2log n)

$T(n) = 4T(n/2) + n^3 \Rightarrow T(n) \in ?$ Θ(n^3)