Divide–and–Conquer Recurrences — The Master Theorem

We assume a divide and conquer algorithm in which a problem with input size n is always divided into a subproblems, each with input size n/b. Here a and b are integer constants with $a \geq 1$ and $b > 1$.

We assume n is a power of b, say $\boldsymbol{n = b^k}$.

Otherwise at some stage we will not be able to divide the sub-problem size exactly by b.

However, the Master Theorem still holds if n is not a power of b, and the subproblem input sizes are $\lceil n/b \rceil$ or $\lfloor n/b \rfloor$

Note $\boldsymbol{k = \log_b(n)}$.

The recurrence for the running time is:

$$\boxed{T(n) = aT(n/b) + f(n), \quad T(1) = d}.$$

Here $f(n)$ represents the divide and combine time (i.e., the non-recursive time). $f(n)$ may involve Θ, e.g., $f(n) = \Theta(n^2)$.

We define $\boldsymbol{E = \log_b(a)}$.

E is called the **critical exponent**. (It strongly influences the solution.) By definition, $\boldsymbol{b^E = a}$.

Note that $\boldsymbol{a^k = n^E}$.

Why? $a^k = (b^E)^k = (b^k)^E = n^E$.

We can write down the total time to solve all sub-problems at a given depth in the recursion tree.

Depth of recursion	**Size of sub-problems**	**Number of sub-problems**	**Total (non-recursive) time at this depth is roughly proportional to**
0	n	1	$f(n)$
1	n/b	a	$a f(n/b)$
2	n/b^2	a^2	$a^2 f(n/b^2)$
3	n/b^3	a^3	$a^3 f(n/b^3)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$k-2$	n/b^{k-2}	a^{k-2}	$a^{k-2} f(n/b^{k-2})$
$k-1$	$n/b^{k-1} = b$	a^{k-1}	$a^{k-1} f(n/b^{k-1}) = \Theta(n^E)$
k	$n/b^k = 1$	$a^k = n^E$	$a^k d = O(n^E)$

$$\boldsymbol{T(n)} = \textbf{sum of terms in rightmost column above}$$
$$= f(n) + a f(n/b) + a^2 f(n/b^2) + \ldots + a^{k-1} f(n/b^{k-1}) + a^k d$$

The critical functions in determining $T(n)$ are:

i) $\boldsymbol{f(n)}$ (the non-recursive time at depth 0)

ii) $\boldsymbol{n^E}$ (the non-recursive time at depth k, or k–1).

Clearly: $T(n) \geq \Theta($ max($n^E, f(n)$)).

On the other hand, if the terms in the right hand column of the table either increase as we move down, or decrease as we move down, then : $T(n) \leq \Theta($ max($n^E, f(n)$) $\cdot \log_b(n)$).

We will see that, if one of n^E and $f(n)$ grows much more rapidly than the other, then $T(n) \leq \Theta($ more rapidly growing function).

Master Theorem:

1) ***f(n)*** **in** $\mathbf{O}(n^{E-\varepsilon})$ for fixed $\varepsilon > 0$	implies	$\boldsymbol{T(n) = \Theta(n^E)}$.
2) ***f(n)*** **in** $\Theta(n^E)$	implies	$\boldsymbol{T(n) = \Theta(n^E \log_b(n))}$.
3) ***f(n)*** **in** $\Omega(n^{E+\varepsilon})$ for fixed $\varepsilon > 0$	implies	$\boldsymbol{T(n) = \Theta(f(n))}$.

Actually, (3) requires an additional hypothesis, that typically holds.

Note none of these cases may apply. For example, if $f(n) = n^E \log_b(n)$, we are between cases (2) and (3); neither case holds.