For the general divide-and-conquer recurrence relation T(n), the *Master Theorem* makes it straightforward to determine divide-and-conquer (and even some decrease-and-conquer) algorithms'efficiency classes. Bear in mind what it does **not** do. It does **NOT** give you an *explicit solution* (a function in terms of n only) to the $T(n)$ recurrence relation.

The theorem states that for $T(n) = aT(n/b) + f(n)$, where $f(n)$ computes the cost of dividing the problem into smaller ones and of combining their solutions:

If

$T(1)$	$=$	c,
a	$>$	0,
b	$>$	1,
c	$>$	0,
n	$=$	$b^k, k = 0, 1, 2, \ldots$ (n is a power of b),
d	$\geq$	0, and
$f(n)$	$\in$	$\Theta(n^d)$,

then

$a < b^d \rightarrow T(n) \in \Theta(n^d)$;

$a = b^d \rightarrow T(n) \in \Theta(n^d \log n)$;

$a > b^d \rightarrow T(n) \in \Theta(n^{\log a})$;

where the logs are $\log_b$, log to the base b.