- The master theorem is used to determine the running time of algorithms (divide and conquer algorithms) in terms of asymptotic notations.
- a = number of subproblems in the recursion. It should be >= 1
- n/b = size of each subproblem
- f(n) = cost of work done outside the recursive calls during the merge process
- Not all recurrent relations can be solved with the use of the master theorem.
- T(n) is not monotone, e.g. T(n) = sin(n)
- F(n) is not a polynomial, e.g. T(n) = 2T(n/2) + 2^n

$$T(n) = aT(n/b) + O\left(n^d \log^k n\right)$$

Source

where n = size of the problem

a = number of subproblems in the recursion and a >= 1

n/b = size of each subproblem

b > 1, d >= 0 and k is a real number.

<table>
<tr><td colspan="4">$T(n) = aT(n/b) + O\left(n^d \log^k n\right)$</td></tr>
<tr><td rowspan="2">$b^d > a$</td><td rowspan="2">$T(n) = O\left(n^d\right)$</td><td>k >= 0</td><td>$T(n) = O(n^d \log^k n)$</td></tr>
<tr><td>k < 0</td><td>$T(n) = O\left(n^d\right)$</td></tr>
<tr><td rowspan="3">$b^d = a$</td><td rowspan="3">$T(n) = O\left(n^d \log_b(n)\right)$</td><td>k > -1</td><td>$T(n) = O\left(n^d \log^{k+1} n\right)$</td></tr>
<tr><td>k = -1</td><td>$T(n) = O\left(n^d \log(\log(n))\right)$</td></tr>
<tr><td>k < -1</td><td>$T(n) = O\left(n^d\right)$</td></tr>
<tr><td>$b^d < a$</td><td>$T(n) = O\left(n^{\log_b a}\right)$</td><td></td><td></td></tr>
</table>