Divide Et Impera
Teorema Master

$T(n) = a * T(n / b) + O(n^d)$

a = nr de subprobleme	n^d	, daca $a < b^d$
b = dim unei subprobleme	$n^d \log n$	, daca $a = b^d$
d = n^d complexitatea ^^	$n^{\log b\ (a)}$	, daca $a > b^d$

- Exemple:
  - Mergesort:

    $a = 2$ $\quad 2 = 2^1$

    $b = 2$ $\quad$ => Th Master

    $d = 1$ $\quad T(n) = O(n \log n)$

    Cautare binara:

    $a = 1$ $\quad 1 = 2^0$

    $b = 2$ $\quad$ => Th Master

    $d = 0$ $\quad T(n) = O(\log n)$

Teorema Master

- Puteti afla complexitatea unui algoritm care foloseste aceasta metoda folosind formula:

  $T(n) = a * T(n / b) + O(n^d)$, unde

  a = numarul de **subprobleme**

  b = dimensiunea unei **subprobleme** (*factorul*)

  d = complexitatea unei **subprobleme**

Teorema Master

$T(n) = a * T(n / b) + O(n^d)$

$$T(n) = \begin{cases} O(n^d) & \text{, daca } a < b^d \\ O(n^d * \log n) & \text{, daca } a = b^d \\ O(n^{\log b\,(a)}) & \text{, daca } a > b^d \end{cases}$$