S19

Materia Diseño de algoritmos

Fecha September 25, 2023

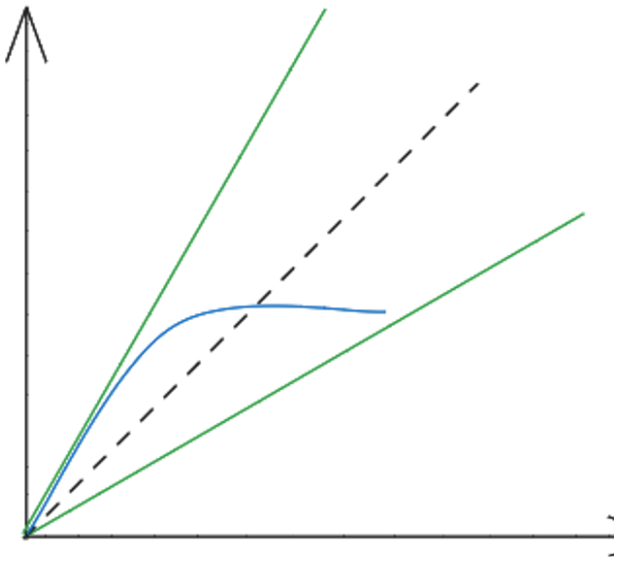
Add a property

Add a comment...

Ejemplo: $T(n) = 2T(\frac{n}{2}) + \theta(n)$

$f(n) \in \theta(n)$

No se puede usar algebra para resolverlo, vamos con el teorema maestro



1. $\exists \quad \epsilon > 0$ tal que $f(n) \in O(n^{log2-\epsilon})$

   $O(n^{1-\epsilon}) = O(1)$

   $= O(\sqrt{n})$

2. $\exists \quad k \geq 0$ tal que $f(n) \in \theta(n^{log2} log^k n)$

   Si $\exists \quad k = 0 \qquad \theta(nlog^k n) = \theta(n)$

   $T(n) \in \theta(nlogn)$

$$T(n) = 8T(\frac{n}{2}) + \theta(1)$$

1. $\exists \quad \epsilon > 0$ tal que $f(n) \in O(n^{log8-\epsilon})$ con $a = 8, \quad b = 2$

   $O(n^{3-\epsilon})$

   $f(n) \in \theta(1) \in O(n^1) \in O(n^2)$

   un $\epsilon < 3$

   $T(n) \in O(n^3)$

$$T(n) = 7T(\frac{n}{2}) + \theta(n^2)$$

1. $\exists \quad \epsilon > 0$ tal que $f(n) \in O(n^{log7-\epsilon})$ con $a = 7, \quad b = 2$

   $log7 \approx 2.8 \qquad O(n^{2.8-\epsilon})$

   un $\epsilon < 0.8$

   $T(n) \in \theta(n^{log7})$

$$T(n) = 2T(\frac{n}{2}) + \frac{n}{logn}$$

Método de sustitución con $n = 2^k$

$T(2^k) = 2T(2^{k-1}) + \frac{2^k}{log2^k}$

$T(2^k) - 2T(2^{k-1}) = \frac{2^k}{k} \qquad P(n)d = -1$ como tenemos $d < 0$ no se cumple

1. $\exists \quad \epsilon > 0$ tal que $f(n) \in O(n^{log2-\epsilon}) \quad$ con $a = 2, \quad b = 2$

   $log2 = 1 \qquad O(n^{1-\epsilon})$ no existe

2. $\exists \quad k \geq 0$ tal que $f(n) \in \theta(n^{log2-log^k n})$

   $\theta(n^{1-log^k n})$ El único valor que podría satisfacerlo es $k = -1$ por lo tanto $\nexists k$

3. $\exists \quad \epsilon > 0$ tal que $f(n) \in \Omega(n^{log2+\epsilon})$

   y además $2(\frac{\frac{n}{2}}{log\frac{n}{2}}) \leq c\frac{n}{logn}$ para cualquier $c < 1$

   $$2(\frac{\frac{n}{2}}{log\frac{n}{2}}) \leq c\frac{n}{logn}$$

   $$\frac{1}{log\frac{n}{2}} \leq \frac{c}{logn}$$

   $1 \leq C\frac{logn - 1}{logn} = c(1 - \frac{1}{logn})$ No existe c

Usando límites

$$lim_{n\to\infty}\frac{\frac{n^{1-\epsilon}}{n}}{logn}$$

$$lim_{n\to\infty}\frac{\frac{n}{logn}}{n^{1-\epsilon}}$$

Análisis de eficiencia en algoritmos de ordenamiento

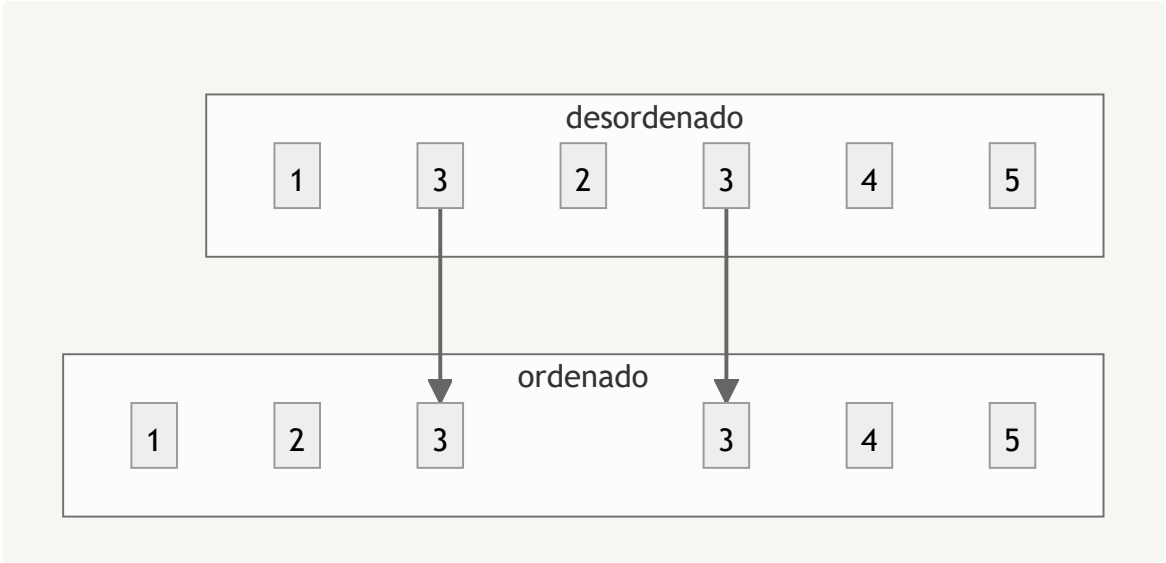
$A\{2,8,7,1,9,8,6,3,5,4\}$

$B = \{\boxdot, \triangleright, \odot, \oplus, *\} = \{a_1 \to \boxdot, a_2 \to \triangleright, a_3, a_4, \ldots.\}$

Clasificación

- Estables/Inestables

  Se dice que un algoritmo de ordenamiento es **estable**, cuando este mantiene el orden relativo de los elementos



Se mantiene el orden relativo del original (3)

- Internos/Externos

  Internos → Memoria, (almacenamiento principal)

  Externos → Almacenamiento secundario DD

- Recursivos/No recursivos

- Directos/Indirectos

  Decimos que un algoritmo es **directo** porque trabaja con el elemento cambiándolo de lugar (tienen mejor rendimiento cuando el tamaño es pequeño o relativamente grande)

  Los indirectos van a trabajar con las direcciones de memoria