Divide y Vencerás

Miguel Raggi
mraggi@gmail.com

Algoritmos
Escuela Nacional de Estudios Superiores
UNAM

3 de abril de 2018

Índice:

Índice:

Ejercicios

Sean f y g funciones crecientes, y supongamos que tenemos un algoritmo A que tarda $\Theta(f)$ y otro B que tarda $\Theta(g)$.

Ejercicios

Sean f y g funciones crecientes, y supongamos que tenemos un algoritmo A que tarda $\Theta(f)$ y otro B que tarda $\Theta(g)$.

- Si corro A dos veces, una seguida de la otra, ¿cuánto me tardo?

$$(a)\ \ \Theta(f) \qquad (b)\ \ \Theta(g) \qquad (c)\ \ \Theta(f+g)$$
$$(d)\ \ \Theta(fg) \quad (e)\ \ \Theta(\text{máx}(f,g)) \quad (f)\ \ \Theta(\text{mín}(f,g))$$

Ejercicios

Sean f y g funciones crecientes, y supongamos que tenemos un algoritmo A que tarda $\Theta(f)$ y otro B que tarda $\Theta(g)$.

- Si corro A dos veces, una seguida de la otra, ¿cuánto me tardo?

$$(a)\ \Theta(f) \qquad (b)\ \Theta(g) \qquad (c)\ \Theta(f+g)$$
$$(d)\ \Theta(fg) \quad (e)\ \Theta(\text{máx}(f,g)) \quad (f)\ \Theta(\text{mín}(f,g))$$

- Si corro B catorce veces, una seguida de la otra, ¿cuánto me tardo?

$$(a)\ \Theta(f) \qquad (b)\ \Theta(g) \qquad (c)\ \Theta(f+g)$$
$$(d)\ \Theta(fg) \quad (e)\ \Theta(\text{máx}(f,g)) \quad (f)\ \Theta(\text{mín}(f,g))$$

- Si corro primero A y luego B, ¿cuánto tardo?

$$(a)\ \Theta(f) \qquad (b)\ \Theta(g) \qquad (c)\ \Theta(f+g)$$
$$(d)\ \Theta(fg) \quad (e)\ \Theta(\text{máx}(f,g)) \quad (f)\ \Theta(\text{mín}(f,g))$$

- Si corro primero A y luego B, ¿cuánto tardo?

$$(a)\ \Theta(f) \qquad (b)\ \Theta(g) \qquad (c)\ \Theta(f+g)$$
$$(d)\ \Theta(fg) \quad (e)\ \Theta(\text{máx}(f,g)) \quad (f)\ \Theta(\text{mín}(f,g))$$

- Si cambio A para que en cada paso de A corro B (por ejemplo, si A es quicksort, y B es comparar dos objetos, estoy corriendo en cada paso de A a B), ¿cuánto tardo?

$$(a)\ \Theta(f) \qquad (b)\ \Theta(g) \qquad (c)\ \Theta(f+g)$$
$$(d)\ \Theta(fg) \quad (e)\ \Theta(\text{máx}(f,g)) \quad (f)\ \Theta(\text{mín}(f,g))$$

Ejercicios

- ¿Cuánto es $O(n + \log(n))$?

Ejercicios

- ¿Cuánto es $O(n + \log(n))$?
- Si $f \in O(n^2)$ y $g \in O(\sqrt{n} \cdot n)$, ¿cuánto es $O(f + g)$?

Ejercicios

- ¿Cuánto es $O(n + \log(n))$?
- Si $f \in O(n^2)$ y $g \in O(\sqrt{n} \cdot n)$, ¿cuánto es $O(f + g)$?
- ¿Qué es más pequeño, $O(n^2 \log(n))$ o $O(n^2 + \log(n))$?

Índice:

Divide y Vencerás

- Vimos ya dos algoritmos del tipo “divide y vencerás”.

Divide y Vencerás

- Vimos ya dos algoritmos del tipo “divide y vencerás”.
- Hay muchos, muchos otros.

Divide y Vencerás

- Vimos ya dos algoritmos del tipo “divide y vencerás”.
- Hay muchos, muchos otros.
- Por ejemplo, la mejor manera de multiplicar matrices que se conoce es de este tipo.

Divide y Vencerás

- Vimos ya dos algoritmos del tipo “divide y vencerás”.
- Hay muchos, muchos otros.
- Por ejemplo, la mejor manera de multiplicar matrices que se conoce es de este tipo.
- Cuando quieres diseñar algoritmos, es buena idea intentar esta idea.

Divide, Conquista y Combina

Todos los algoritmos del tipo “divide y vencerás” tienen 3 pasos:

Divide, Conquista y Combina

Todos los algoritmos del tipo “divide y vencerás” tienen 3 pasos:

- Divide: Agarras la entrada original y la divides de algún modo en algún número de problemas más pequeños.

Divide, Conquista y Combina

Todos los algoritmos del tipo “divide y vencerás” tienen 3 pasos:

- Divide: Agarras la entrada original y la divides de algún modo en algún número de problemas más pequeños.
- Conquista: Utilizas el mismo algoritmo para tratar cada uno de los pedazos pequeños (recursión)

Divide, Conquista y Combina

Todos los algoritmos del tipo “divide y vencerás” tienen 3 pasos:

- Divide: Agarras la entrada original y la divides de algún modo en algún número de problemas más pequeños.
- Conquista: Utilizas el mismo algoritmo para tratar cada uno de los pedazos pequeños (recursión)
- Combina: Ya que el problema fue resuelto para los pedazos pequeños, utilizas la información obtenida para resolver el problema grande.

Ejemplos

En los siguientes ejemplos hay buenos algoritmos que utilizan la técnica:

Ejemplos

En los siguientes ejemplos hay buenos algoritmos que utilizan la técnica:

- Ordenamiento (quicksort, mergesort).

Ejemplos

En los siguientes ejemplos hay buenos algoritmos que utilizan la técnica:

- Ordenamiento (quicksort, mergesort).
- Multiplicar matrices: Strassen.

Ejemplos

En los siguientes ejemplos hay buenos algoritmos que utilizan la técnica:

- Ordenamiento (quicksort, mergesort).
- Multiplicar matrices: Strassen.
- Transformada rápida de Fourier (Cooley–Tukey).

Ejemplos

En los siguientes ejemplos hay buenos algoritmos que utilizan la técnica:

- Ordenamiento (quicksort, mergesort).
- Multiplicar matrices: Strassen.
- Transformada rápida de Fourier (Cooley–Tukey).
- Geometría: Encontrar envolventes convexas, etc.

Ejemplos

En los siguientes ejemplos hay buenos algoritmos que utilizan la técnica:

- Ordenamiento (quicksort, mergesort).
- Multiplicar matrices: Strassen.
- Transformada rápida de Fourier (Cooley–Tukey).
- Geometría: Encontrar envolventes convexas, etc.
- Multiplicar números grandes.

Ejemplos

En los siguientes ejemplos hay buenos algoritmos que utilizan la técnica:

- Ordenamiento (quicksort, mergesort).
- Multiplicar matrices: Strassen.
- Transformada rápida de Fourier (Cooley–Tukey).
- Geometría: Encontrar envolventes convexas, etc.
- Multiplicar números grandes.
- Eigenvalores (álgebra lineal).

El análisis

- El análisis del tiempo que tardan estos algoritmos se puede complicar:

El análisis

- El análisis del tiempo que tardan estos algoritmos se puede complicar:
- Podemos expresar el tiempo $T(n)$ que utiliza un algoritmo en una instancia de tamaño n en términos de $T(\text{cosas menores})$.

El análisis

- El análisis del tiempo que tardan estos algoritmos se puede complicar:
- Podemos expresar el tiempo $T(n)$ que utiliza un algoritmo en una instancia de tamaño n en términos de $T(\text{cosas menores})$.
- No se puede resolver en general (es decir, obtener una expresión cerrada), pero en el caso de "divide y vencerás" se tiene un teorema maestro.

El análisis

- El análisis del tiempo que tardan estos algoritmos se puede complicar:
- Podemos expresar el tiempo $T(n)$ que utiliza un algoritmo en una instancia de tamaño n en términos de $T(\text{cosas menores})$.
- No se puede resolver en general (es decir, obtener una expresión cerrada), pero en el caso de "divide y vencerás" se tiene un teorema maestro.
- Veremos el teorema y cómo utilizarlo, pero no la demostración, porque es larga y aburrida y utiliza matemáticas que aún no han visto ustedes.

El análisis

- El análisis del tiempo que tardan estos algoritmos se puede complicar:
- Podemos expresar el tiempo $T(n)$ que utiliza un algoritmo en una instancia de tamaño n en términos de $T(\text{cosas menores})$.
- No se puede resolver en general (es decir, obtener una expresión cerrada), pero en el caso de "divide y vencerás" se tiene un teorema maestro.
- Veremos el teorema y cómo utilizarlo, pero no la demostración, porque es larga y aburrida y utiliza matemáticas que aún no han visto ustedes.
- Además, se necesita un poco de probabilidad para entender bien bien cómo poner $T(n)$ en términos de los anteriores, cuando no se sabe de qué tamaño serán los problemas más chicos ("esperanza" o "promedio" de una variable aleatoria).

Algunos ejemplos

- A veces podemos encontrar la expresión de $T(n)$ directamente. Por ejemplo, si $T(n) = 2T(n-1)$, entonces $T(n) = 2^n O(1)$.

Algunos ejemplos

- A veces podemos encontrar la expresión de $T(n)$ directamente. Por ejemplo, si $T(n) = 2T(n-1)$, entonces $T(n) = 2^n O(1)$.
- Pero a veces es más difícil:

Algunos ejemplos

- A veces podemos encontrar la expresión de $T(n)$ directamente. Por ejemplo, si $T(n) = 2T(n-1)$, entonces $T(n) = 2^n O(1)$.
- Pero a veces es más difícil:
- Por ejemplo, mergesort divide el problema en términos de 2 problemas más chicos, ambos de tamaño $n/2$.

Algunos ejemplos

- A veces podemos encontrar la expresión de $T(n)$ directamente. Por ejemplo, si $T(n) = 2T(n-1)$, entonces $T(n) = 2^n O(1)$.
- Pero a veces es más difícil:
- Por ejemplo, mergesort divide el problema en términos de 2 problemas más chicos, ambos de tamaño $n/2$.
- Después los resuelve, y después los combina, tomando tiempo $O(n)$.

Algunos ejemplos

- A veces podemos encontrar la expresión de $T(n)$ directamente. Por ejemplo, si $T(n) = 2T(n-1)$, entonces $T(n) = 2^n O(1)$.
- Pero a veces es más difícil:
- Por ejemplo, mergesort divide el problema en términos de 2 problemas más chicos, ambos de tamaño $n/2$.
- Después los resuelve, y después los combina, tomando tiempo $O(n)$.
- Entonces podemos escribir algo como:

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

donde $f(n) \in \Theta(n)$.

Índice:

El Teorema Maestro

Teorema

Sean a, b *constantes y* $f : \mathbb{N} \to \mathbb{N}$. *Para una recurrencia de la forma*

$$T(n) = aT(n/b) + f(n)$$

El Teorema Maestro

Teorema

Sean a, b constantes y $f : \mathbb{N} \to \mathbb{N}$. Para una recurrencia de la forma

$$T(n) = aT(n/b) + f(n)$$

Entonces hay 3 casos para $\Theta(T)$.

El Teorema Maestro

Teorema

Sean a, b *constantes y* $f : \mathbb{N} \to \mathbb{N}$. *Para una recurrencia de la forma*

$$T(n) = aT(n/b) + f(n)$$

Entonces hay 3 casos para $\Theta(T)$. *Sea* $k = \log_b(a)$.

El Teorema Maestro

Teorema

Sean a, b *constantes y* $f : \mathbb{N} \to \mathbb{N}$. *Para una recurrencia de la forma*

$$T(n) = aT(n/b) + f(n)$$

Entonces hay 3 casos para $\Theta(T)$. *Sea* $k = \log_b(a)$.

1. *Si* $f(n) \in O(n^{k-\epsilon})$ *para alguna* $\epsilon > 0$, *entonces* $T(n) \in O(n^k)$.

El Teorema Maestro

Teorema

Sean a, b *constantes y* $f : \mathbb{N} \to \mathbb{N}$. *Para una recurrencia de la forma*

$$T(n) = aT(n/b) + f(n)$$

Entonces hay 3 casos para $\Theta(T)$. *Sea* $k = \log_b(a)$.

1. *Si* $f(n) \in O(n^{k-\epsilon})$ *para alguna* $\epsilon > 0$, *entonces* $T(n) \in O(n^k)$.
2. *Si* $f(n) \in \Theta(n^k)$, *entonces* $T(n) \in \Theta(n^k \log(n))$.

El Teorema Maestro

Teorema

Sean a, b *constantes y* $f : \mathbb{N} \to \mathbb{N}$. *Para una recurrencia de la forma*

$$T(n) = aT(n/b) + f(n)$$

Entonces hay 3 casos para $\Theta(T)$. *Sea* $k = \log_b(a)$.

1. *Si* $f(n) \in O(n^{k-\epsilon})$ *para alguna* $\epsilon > 0$, *entonces* $T(n) \in O(n^k)$.
2. *Si* $f(n) \in \Theta(n^k)$, *entonces* $T(n) \in \Theta(n^k \log(n))$.
3. *Si* $f(n) \in \Omega(n^{k+\epsilon})$, *para alguna* $\epsilon > 0$,

El Teorema Maestro

Teorema

Sean a, b *constantes y* $f : \mathbb{N} \to \mathbb{N}$. *Para una recurrencia de la forma*

$$T(n) = aT(n/b) + f(n)$$

Entonces hay 3 casos para $\Theta(T)$. *Sea* $k = \log_b(a)$.

1. *Si* $f(n) \in O(n^{k-\epsilon})$ *para alguna* $\epsilon > 0$, *entonces* $T(n) \in O(n^k)$.
2. *Si* $f(n) \in \Theta(n^k)$, *entonces* $T(n) \in \Theta(n^k \log(n))$.
3. *Si* $f(n) \in \Omega(n^{k+\epsilon})$, *para alguna* $\epsilon > 0$, *y si* $af(n/b) \leq cf(n)$ *para alguna constante* c *con* n *suficientemente grande, entonces* $T(n) \in \Theta(f(n))$.

El Teorema Maestro

Teorema

Sean a, b constantes y $f : \mathbb{N} \to \mathbb{N}$. Para una recurrencia de la forma

$$T(n) = aT(n/b) + f(n)$$

Entonces hay 3 casos para $\Theta(T)$. Sea $k = \log_b(a)$.

1. *Si $f(n) \in O(n^{k-\epsilon})$ para alguna $\epsilon > 0$, entonces $T(n) \in O(n^k)$.*
2. *Si $f(n) \in \Theta(n^k)$, entonces $T(n) \in \Theta(n^k \log(n))$.*
3. *Si $f(n) \in \Omega(n^{k+\epsilon})$, para alguna $\epsilon > 0$, y si $af(n/b) \leq cf(n)$ para alguna constante c con n suficientemente grande, entonces $T(n) \in \Theta(f(n))$.*

Nota: Aunque técnicamente hay funciones f que no caen dentro de ninguno de los tres casos, en la práctica es muy muy raro encontrarlas. Por lo tanto, este teorema resuelve (casi) completamente el análisis de los algoritmos de tipo "divide y vencerás".

Mergesort

- Para mergesort, la recurrencia es:

$$T(n) = 2T(\frac{n}{2}) + f(n) \text{ con } f(n) \in O(n)$$

Mergesort

- Para mergesort, la recurrencia es:

$$T(n) = 2T(\frac{n}{2}) + f(n) \text{ con } f(n) \in O(n)$$

- Entonces, si nos fijamos en el teorema, $a = 2$, $b = 2$, y $\log_2(2) = 1$, y $f(n) \in \Theta(O(n^1))$, así que estamos en el segundo caso, así que $T(n) \in \Theta(n^1 \log(n))$

Mergesort

- Para mergesort, la recurrencia es:

$$T(n) = 2T(\frac{n}{2}) + f(n) \text{ con } f(n) \in O(n)$$

- Entonces, si nos fijamos en el teorema, $a = 2$, $b = 2$, y $\log_2(2) = 1$, y $f(n) \in \Theta(O(n^1))$, así que estamos en el segundo caso, así que $T(n) \in \Theta(n^1 \log(n))$
- Ejercicio: Si tenemos un algoritmo que divide un problema de tamaño n en 4 subproblemas de tamaño $n/2$ cada uno, y en dividir tarda tiempo $\Theta(n)$ pero en combinar tarda tiempo $\Theta(n^2)$, ¿cuánto tarda el problema?