

# Divide & Conquer

Fernando Schapachnik<sup>1,2</sup>

<sup>1</sup>Basada en trabajo previo de Esteban Feuerstein

<sup>2</sup>Departamento de Computación, FCEyN,  
Universidad de Buenos Aires, Buenos Aires, Argentina

Algoritmos y Estructuras de Datos II,  
segundo cuatrimestre de 2018

## (2) Técnicas algorítmicas

- Qué es y qué no es una técnica algorítmica.
- D&C es una de ellas.
- Se basa en dividir un problemas en varios subproblemas más chicos,
- resolverlos,
- y combinar las soluciones.

### (3) D&C

- ¿Cuáles de estos son D&C?
  - Pintar una pared: sí
  - Construir una casa: no
  - Buscar al máximo en una matriz recursivamente: sí

## (4) D&C (cont.)

- Se trata de:
  - Dividir un problema en subproblemas.
  - Resolver los problemas más pequeños.
  - Combinar las soluciones.
- Algunas características de algoritmos D&C:
  - Las subpartes tienen que ser más pequeñas.
  - Y ser el mismo tipo de tarea.
  - Dividir y combinar pueden no ser nulas, pero no tienen que ser demasiado costosas.

## (5) Forma general de D&C

- $F(X)$ 
  - Si  $X$  es suficientemente chico o simple, solucionar de manera ad hoc.
  - Si no,
    - Dividir a  $X$  en  $X_1, X_2, \dots, X_k$
    - $\forall i \leq k$ , hacer  $Y_i = F(X_i)$
    - Combinar los  $Y_i$  en un  $Y$  que es una solución para  $X$ .
    - Devolver  $Y$

## (6) ¿Y cuánto tarda?

- ¿Cómo calculamos la complejidad de un algoritmo D&C?
- El costo de un algoritmo D&C de tamaño  $n$  se puede expresar como  $T(n)$ , que debe considerar:
  - Resolver  $a$  subproblemas de tamaño máximo  $n/c$ , siempre que  $n/c > n_0$ , para algún  $n_0$ .
  - El costo de efectivamente hacer la subdivisión y luego unir los resultados.
  - Si ese costo es  $b$  cuando  $n = 1$ , entonces podemos expresarlo como  $b n^d$ , para algún  $d$ .
- Es decir,  $T(n) = a T(n/c) + b n^d$ .

## (7) ¿Y cuánto tarda? (cont.)

- Supongamos que  $n = c^k$  para algún  $k$  y analicemos la recurrencia.
- $T(n) = a T(n/c) + b n^d$
- $= T(c^k) = a T(c^{k-1}) + b (c^k)^d$
- $= a(a T(c^{k-2}) + (b c^{(k-1)d})) + bc^{kd}$
- $= a^2 T(c^{k-2}) + abc^{(k-1)d} + bc^{kd}$
- $= a^3 T(c^{k-3}) + a^2 b(c^{k-2})^d + abc^{(k-1)d} + bc^{kd}$
- ...
- $= a^j T(c^{k-j}) + \sum_{i=0}^{j-1} a^i bc^{(k-i)d}$
- $= a^j T(c^{k-j}) + b \sum_{i=0}^{j-1} a^i c^{(k-i)d}$

## (8) ¿Y cuánto tarda? (cont.)

- $T(c^k) = a^j T(c^{k-j}) + b \sum_{i=0}^{j-1} a^i c^{(k-i)d}$
- Seguimos teniendo una recurrencia. ¿Cuál es el caso base?  
 $T(1)$
- $c^{k-j} = 1$ , es decir  $c^k / c^j = 1$ , es decir, hasta que  
 $j = k = \log_c n$
- $= a^k T(1) + b \sum_{i=0}^{k-1} a^i c^{(k-i)d}$
- Si tenemos en cuenta que  $T(1) = b$ , nos queda
- $= a^k b + b \sum_{i=0}^{k-1} a^i c^{(k-i)d}$
- $= b \sum_{i=0}^k a^i c^{(k-i)d}$
- $= b \sum_{i=0}^k a^i c^{dk-di}$
- $= b c^{dk} \sum_{i=0}^k a^i c^{-di}$
- ¿Y esto cuánto es?



## (9) Análisis por caso

- $T(n) = a T(n/c) + b n^d$
- $= bc^{dk} \sum_{i=0}^k a^i c^{-di}$
- Si  $a = 1$  y  $d = 0$ , es decir, 1 subproblema, combinar tiene costo constante:  $b \sum_{i=0}^{\log_c n} 1 = O(\log_c n)$
- Si  $d = 1$ , es decir, división + unión tiene costo lineal.
- $T(n)$  queda como  $bn \sum_{i=0}^{\log_c n} (a/c)^i$ 
  - Si  $a < c$  ("pocos subproblemas"),  $a/c < 1$ , por ende, la serie converge:
    - Cuando  $n \rightarrow \infty$ :
    - $bn \sum_{i=0}^{\log_c n} (a/c)^i \rightarrow bn \text{ cte} = O(n)$
  - Si  $a = c$ :
    - $bn \sum_{i=0}^{\log_c n} 1 = O(n \log_c n)$
  - Si  $a > c$  ("muchos subproblemas")
    - Recordemos que  $\sum_{i=0}^x y^i = \frac{y^{x+1}-1}{y-1}$  (1)

## (10) Análisis por caso (cont.)

- Caso  $d = 1$ ,  $a > c$ ,  $T(n) = bn \sum_{i=0}^{\log_c n} (a/c)^i$
- Usando (1) nos queda
- $T(n) = bn \frac{(a/c)^{\log_c n + 1} - 1}{a/c - 1}$
- Aplicando  $O()$  queda como  $O(n(\frac{a}{c})^{\log_c n})$
- $= O(n \frac{a^{\log_c n}}{c^{\log_c n}})$
- $= O(n \frac{a^{\log_c n}}{n})$
- $= O(a^{\log_a n \cdot \log_c a})$
- $= O((a^{\log_a n})^{\log_c a})$
- $= O(n^{\log_c a})$

## (11) Teorema Maestro

- Permite resolver relaciones de recurrencia de la forma:

$$T(n) = \begin{cases} a T(n/c) + f(n) & \text{si } n > 1 \\ 1 & \text{si } n = 1 \end{cases}$$

- Si  $f(n) = O(n^{\log_c a - \epsilon})$  para  $\epsilon > 0$ , entonces  $T(n) = \Theta(n^{\log_c a})$
- Si  $f(n) = \Theta(n^{\log_c a})$ , entonces  $T(n) = \Theta(n^{\log_c a} \log n)$
- Si  $f(n) = \Omega(n^{\log_c a + \epsilon})$  para  $\epsilon > 0$  y  $af(n/c) < c'f(n)$  para  $c' < 1$  y  $n$  suficientemente grande, entonces  $T(n) = \Theta(f(n))$
- Intuitivamente se trata de comparar  $f(n)$  contra  $n^{\log_c a}$  y ver quién gana.
- La demo la pueden ver en el Cormen, cap. 4.

## (12) Multiplicación entera

- ¿Cuál es la complejidad de multiplicar dos números enteros?
- Si tienen  $n$  dígitos en base  $b$  la complejidad es  $O(n^2)$ .
- ¿Puedo hacer algo mejor?
- Podemos expresarlos como una suma donde cada sumando tiene la mitad de los dígitos (aprox).
- $x = x_1 b^{n/2} + x_0$  y  $y = y_1 b^{n/2} + y_0$ .
- Entonces  $xy$  es  $x_1 y_1 b^n + (x_0 y_1 + x_1 y_0) b^{n/2} + x_0 y_0$
- Todavía no gané nada, pero qué pasa si defino:
- $m_1 = x_0 y_0$ ,  $m_2 = x_1 y_1$  y  $m_3 = (x_0 - x_1)(y_1 - y_0)$
- La multiplicación se vuelve
$$xy = m_2 b^n + (m_1 + m_2 + m_3) b^{n/2} + m_1$$
- Esto se llama algoritmo de Karatsuba.

## (13) Algoritmo de Karatsuba

- $xy = m_2 b^n + (m_1 + m_2 + m_3) b^{n/2} + m_1$
- Pensémoslo algorítmicamente.
- Karatsuba( $x, y$ )
  - 1) Si son suficientemente chicos, multiplicarlos “a mano” y retornar.
  - 2) Separar a  $x$  en  $x_1$  y  $x_0$ .
  - 3) Separar a  $y$  en  $y_1$  y  $y_0$ .
  - 4) Calcular  $m_1$ ,  $m_2$  y  $m_3$  mediante llamadas recursivas.
  - 5) Sumar  $m_2$  desplazado  $n$   $b$ -bits +  $(m_1 + m_2 + m_3)$  desplazado  $n/2$   $b$ -bits +  $m_1$ .
  - 6) Retornar esa suma.
- ¿Cuál es la complejidad?
- Separaciones, sumas y desplazados son lineales en  $n$ .
- Hay 3 llamadas recursivas.
- $T(n) = 3T(\lceil n/2 \rceil) + c'n + c''$
- $T(n) = 3T(\lceil n/2 \rceil) + f(n)$  con  $f(n) = O(n^{\log_2 3 - \epsilon})$
- Que por el Teorema Maestro tiene  $\Theta(n^{\log_2 3})$ , es decir aprox.  $\Theta(n^{1.59})$ .