División y Conquista: Presentación

Teoría de Algoritmos I (75.29 / 95.06)

Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Dīvide et īmpera

Máxima política y militar

atribuida al emperador romano Julio Cesar.

Utilizado en algoritmos matemáticos

desde hace siglos (Euclides, Gauss, ...)

División y conquista

Conjunto de técnicas algorítmicas

en el que se divide el problema en subproblemas de igual naturaleza y menor tamaño

se los conquista (resuelve) en forma recursiva (hasta un caso base)

y se combina los resultados en una solución general

Generalmente se pueden aplicar a problemas

donde la solución por fuerza bruta ya tiene una complejidad polinómica

Analizar su complejidad requiere resolver una relación de recurrencia

Relación de recurrencia

Ecuación que define una secuencia recursiva

Cada término de la secuencia es definido como una función de términos anteriores

$$T_n = F(T_{n-1}, T_{n-2}, \ldots)$$

Existe uno o varios términos base o iniciales desde los cuales se calculan los siguientes.

Plantilla básica

1. Dividir el problema en “q” subproblemas de tamaño reducido al original.
2. Resolver cada subproblema por separado mediante recursión
3. Combinar el resultado de los subproblemas.

Ejemplo: MergeSort

Algoritmo de ordenamiento

utiliza el divide y vencerás

Propuesto por John von Neumann en 1945

(Según Knuth en “Art of computing Programming”)

MERGESORT(A)

Si size A == 2 → comparar y devolver ordenado

A1 = (size A)/2 primeros elementos de A

A2 = (size A)/2 últimos elementos de A

Retornar MERGE (MERGESORT(A1), MERGESORT(A2))

MergeSort – Análisis de Complejidad (cont.)

Se debe resolver la relación de recurrencia

para poder calcular la complejidad.

3 formas básicas de resolverlas:

“Desenrollarla”

“Adivinar” y verificar: Inducción

Método del maestro

MergeSort – Análisis de Complejidad

Sea T(n) el peor caso de tiempo de ejecución

para la resolución del problema de “n” elementos.

$F(N) = O(1) \leq c$ (para un $c > 0$)

Sea DIV el proceso de dividir el problema en 2 subproblemas.

Sea UNI el proceso de unir el resultado de de los 2 subproblemas

$MERGE \longrightarrow F(N) = O(N) \leq c\,n$ (para un $c > 0$)

$T(n) \leq 2^*T(n/2) + DIV + UNI$ y $T(2) \leq c$

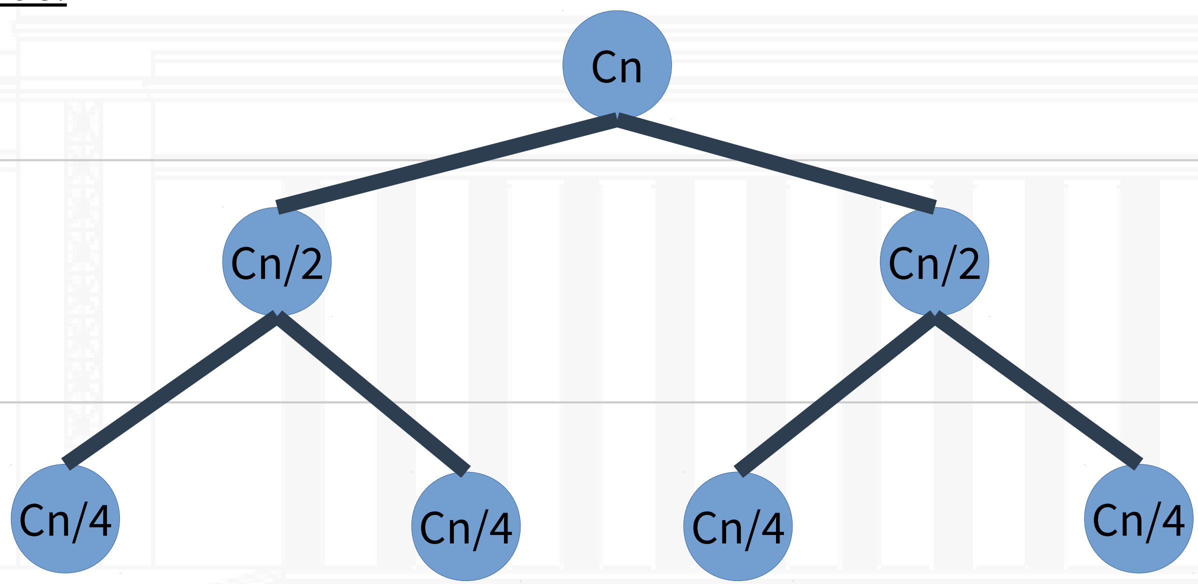
$$\begin{cases} T(n) \leq 2^*T(n/2) + cn & n>2 \\ T(2) \leq d & n=2 \end{cases}$$

Desenrollando la recurrencia

Analizar los primeros niveles de la misma

Cada nivel "consume" cn operaciones

<u>Subproblemas:</u>



1 — *Nivel 0: Cn*

2 — *Nivel 1: 2*Cn/2=cn*

4 — *Nivel 2: 4*Cn/4=cn*

.
.
.

En el nivel j tengo problemas con 2 elementos

¿Cuántos niveles hay?

2^j — $2^j = n/2 \rightarrow j*\log_2 2 = \log_2 n / \log_2 2 \rightarrow j = \log_2 n$ ➡ $O(cn*\log_2 n)$ ⬅ *Nivel j:* $2^j*Cn/2^j=cn$

Validación del resultado

Probar en la recurrencia la validez del resultado

$T(2) \le c$	$T(n) \le 2*T(n/2) + cn$
$T(n{=}2) = cn \log2 n = 2c$ $T(2) \le d < 2c$	$T(n) \le 2c(n/2) \log2(n/2) + cn$ $= cn[(\log2 n) - 1] + cn$ $= (cn \log2 n) - cn + cn$ $= cn \log2 n.$

Desenrollando - Matemáticamente

Relación de recurrencia

T(n) = 2T(n/2) + cn

T(2) = d

T(n/2) = 2T(n/4) + n/2

T(n/4) = 2T(n/8) + n/4

. . .

T(n) = 2 [2 (2 { T(n/16) + n/8 } + n/4) + n/2] + n

$$T(n) = 2^k * 1 + \sum_{i=0}^{k} \frac{2^i * n}{2^i}$$

Desenrollando – Matemáticamente (cont.)

Continuemos

vemos que k= logn

$$a^{\log_2 n} = n^{\log_2 a}$$

$$T(n) = 2^k * 1 + \sum_{i=0}^{k} \frac{2^i * n}{2^i}$$

$$T(n) = 2^{\log(n)} + n * \sum_{i=0}^{\log(n)} \left(\frac{2}{2}\right)^i$$

$$T(n) = 2^{\log(n)} + n * \sum_{i=0}^{\log(n)} 1^i$$

$$T(n) = 2^{\log(n)} + n \log(n)$$

$$T(n) = n^{\log(2)} + n \log(n) \quad \Rightarrow \quad T(n) = O(n \log n)$$

“Adivinar” y verificar

Podemos calcular T(n) probando…

Ej: $T(n) = O(n)$, $O(n^2)$, $O(n \log n)$

$$T(n) \leq 2*T(n/2) + cn \quad n>2$$

$O(n)$: $kn \leq 2* kn/2 + cn \leq (k+c)n$ sii $c=0$ X

$O(n^2)$: $kn^2 \leq 2* kn^2/2 + cn \leq kn^2 + cn$ para todo $c \geq 0$ √

$O(n \log n)$: $kn \log_2 n \leq 2k(n/2) \log_2 (k/2) + cn =$

$= kn*(\log_2 n -1) + cn = kn \log_2 n - kn + cn$

Mejor!

$kn \log_2 n \leq kn \log_2 n + (c-k)n$ sii $k \leq c$ √

Corolario

Cualquier función que satisfaga:

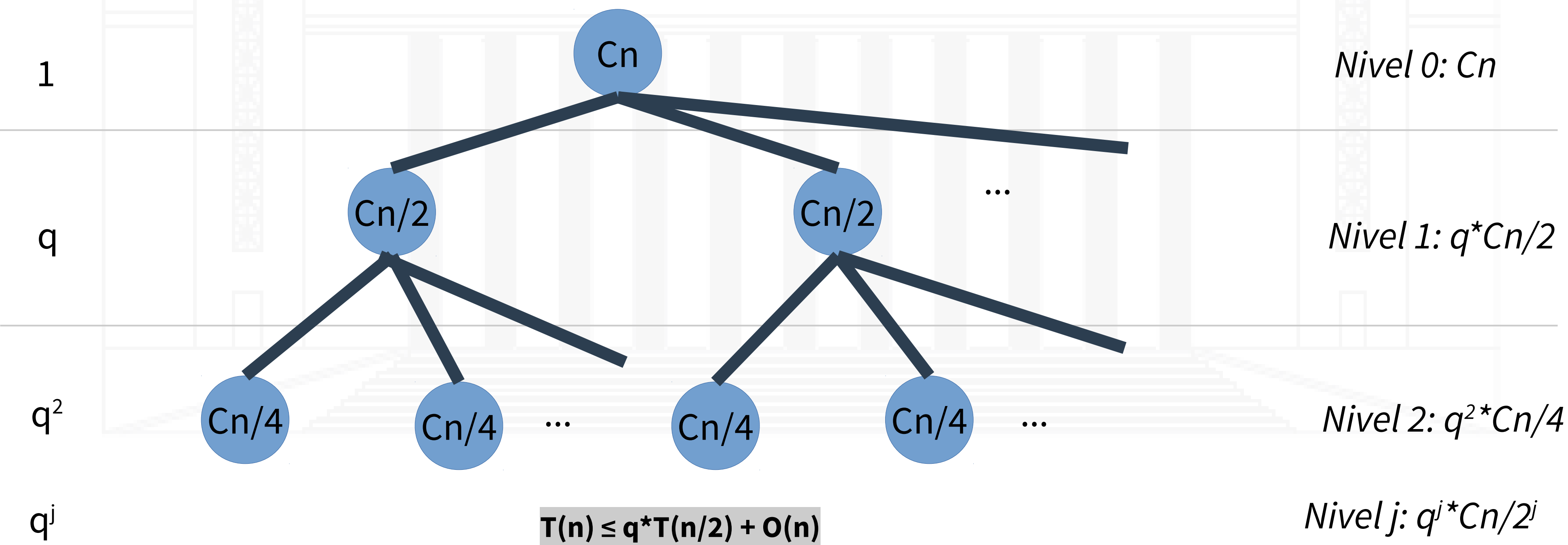
$$T(n) \leq 2*T(n/2) + cn \quad n>2$$
$$T(2) \leq c$$

cumple con O(n log n) cuando n>1

Subdivisión en mas de 2 partes

Que pasa si en vez de dividir el problema en 2 subproblemas se hace en “q”?

Si q>2



$T(n) \leq q^*T(n/2) + O(n)$

Análisis de complejidad

Podemos explorar la recurrencia como:

r = q/2 > 1 → podemos usar la suma geométrica

$$\sum_{i=0}^{x} r^i = \frac{1-r^{(x+1)}}{1-r}$$

$$T(n) \leq \sum_{j=0}^{\log_2 n - 1} \left(\frac{q}{2}\right)^j cn$$

$$n^{\log_2 r}$$

$$T(n) \leq cn \frac{\left(1-r^{\log_2 n}\right)}{1-r} \leq cn \frac{\left(r^{\log_2 n}\right)}{r-1}$$

Si q=3 → O($n^{1,59}$)

Si q=4 → O(n^2)

$$T(n) \leqslant \frac{c}{\frac{q}{2}-1} n\, n^{\log_2 \frac{q}{2}} = \frac{c}{\frac{q}{2}-1} n^{1+\log_2 q - 1}$$

$$O\left(n^{\log_2 q}\right)$$

Subdivisión en solo 1 problema

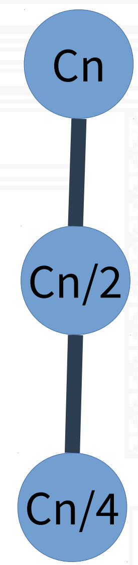
Si q=1

$$T(n) \leqslant \sum_{j=0}^{\log_2 n - 1} \frac{cn}{2^j}$$

podemos usar la suma geométrica

T(n) ≤ 2 cn

$$\boldsymbol{O(n)}$$



Nivel 0: Cn

Nivel 1: Cn/2

Nivel 2: Cn/4

T(n) ≤ q*T(n/2) + O(n)

El método del maestro

Presentado por Jon Bentley, Dorothea Haken y James B. Saxe

en el paper “A General Method for Solving Divide-and-Conquer Recurrences ” en 1980.

Nombrado originalmente "unifying method"

Popularizado como "Master Theorem" por Cormen, Leiserson, Rivest, and Stein

Presentación realizada en Septiembre de 2020