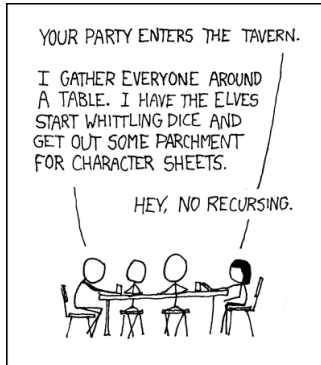


Resolviendo Recurrencias

Mauro Jaskelioff

20/03/2023



Análisis de Algoritmos

- ▶ Queremos poder evaluar la performance de **algoritmos**.
- ▶ Para esto utilizamos
 - ▶ Notación Asintótica
 - ▶ Modelo de Costo basado en Lenguaje
 - ▶ Analizamos trabajo (W) y profundidad (S)
- ▶ Al plantear el trabajo y profundidad de algoritmos recursivos surgen recurrencias.
- ▶ ¿Cómo resolver las recurrencias?
 - ▶ Veremos varias técnicas...

Método de Substitución

- ▶ Seguimos el siguiente procedimiento
 1. Adivinar la forma de la solución.
 2. Probar que la forma es correcta usando inducción matemática.
- ▶ Ejemplo (recordar mergesort)

$$W(0) = c_0$$

$$W(1) = c_1$$

$$W(n) = 2W(\lfloor n/2 \rfloor) + c_2 n$$

- ▶ Adivinamos que $W(n) \in O(n \lg n)$
- ▶ Probamos que $\exists c$ tal que $W(n) \leq cn \lg n$.

- Reemplazamos nuestra solución en la recurrencia

$$\begin{aligned}W(n) &\leq 2(c \lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor) + c_2 n \\&\leq cn \lg(n/2) + c_2 n \\&= cn(\lg n - \lg 2) + c_2 n \\&= cn \lg n - cn + c_2 n \\&\leq cn \lg n\end{aligned}$$

siempre y cuando $c \geq c_2$.

- Faltan chequear los casos bases. ¿ $W(0) \leq c \cdot 0 \cdot \lg 0$?
- ¿ $W(1) \leq c \cdot 1 \cdot \lg 1$?
- La notación O sólo requiere $W(n) \leq cn \lg n$ para $n \geq N$.
Tomamos $N = 2$

$$W(2) = 2W(1) + 2c_2 = 2c_1 + 2c_2 \leq c \cdot 2 \lg 2 = 2c$$

$$W(3) = 2W(1) + 3c_2 = 2c_1 + 3c_2 \leq c \cdot 3 \lg 3$$

- Elegimos c suficientemente grande.

Acerca de la adivinanza

- ▶ Como adivinanza de una solución $O(n \lg n)$ utilizamos una función $f(n) = cn(\lg n)$ para una constante arbitraria c .
- ▶ Es obvio que $f(n) \in O(n \lg n)$.
- ▶ Podemos utilizar **cualquier** función $O(n \lg n)$. Por ejemplo:
 - ▶ $f(n) = cn(\lg n) + c_1 n$,
 - ▶ $f(n) = cn(\lg n) - c_2$, o
 - ▶ $f(n) = cn(\lg n) - c_1 n + c_2$.
- ▶ El sumar o restar elementos de menor orden puede ser crucial para poder terminar la prueba.

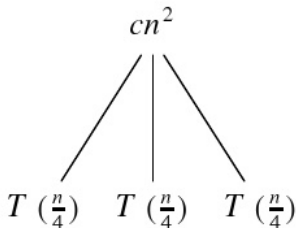
Árboles de recurrencia

- ▶ Una técnica general para resolver recurrencias son los *árboles de recurrencia*
- ▶ Los ilustramos usando la recurrencia

$$T(1) = c_1$$

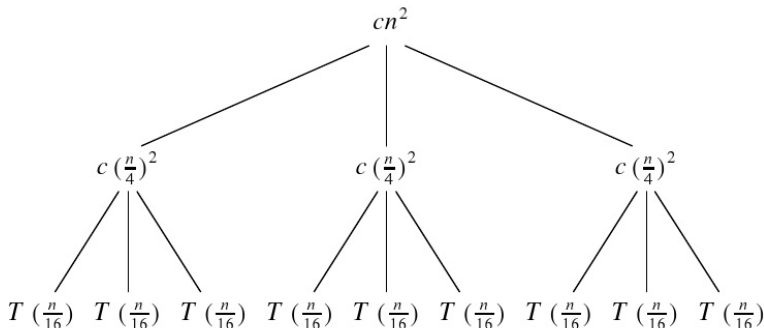
$$T(n) = 3T(n/4) + cn^2$$

- ▶ Hacemos un árbol que en su raíz tiene el costo para el n inicial, con ramas indicando cada una de las llamadas recursivas.



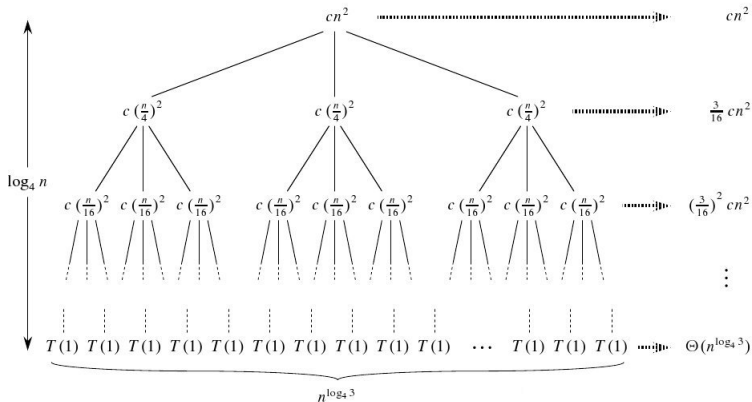
Árboles de recurrencia (cont.)

- Las ramas correspondientes a las llamadas recursivas se expanden, ahora tienen el costo correspondiente a cada llamada y generan nuevas hojas.



Árboles de recurrencia (cont.)

- ▶ Se sigue expandiendo el árbol hasta llegar a un caso base.
- ▶ En cada nivel se suma el total de operaciones por nivel.
El costo total es la suma de todos los niveles.



Árboles de recurrencia (cont.)

- ▶ La suma obtenida se manipula algebraicamente para llegar al resultado.

$$\begin{aligned}T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + kn^{\log_4 3} \\&= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + kn^{\log_4 3} \\&= \frac{(3/16)^{\log_4 n} - 1}{(3/16) - 1} cn^2 + kn^{\log_4 3} \\&\in O(n^2)\end{aligned}$$

- ▶ Si somos prolijos, el método nos da una solución exacta.
- ▶ Si no, al menos nos da un candidato para usar el método de substitución.

Ejercicio

Verificar usando el método de la substitución que la cota obtenida es ajustada. Es decir, probar que

$$T(n) = 3T(n/4) + cn^2 \in \Theta(n^2)$$

Nota: De aquí en más, si no se menciona caso base, suponerlo(s) constante(s).

Simplificando Recurrencias

- ▶ Para el análisis de mergesort, nos olvidamos de los $\lfloor \cdot \rfloor$ y los $\lceil \cdot \rceil$
- ▶ ¿Cuándo es válido hacer estas aproximaciones?
- ▶ Si $n = b^k$ entonces $n/b = \lfloor n/b \rfloor = \lceil n/b \rceil$.
- ▶ Notar que no alcanza que n sea múltiplo de b .
- ▶ Podría suceder que la solución asintótica para entradas $n = b^k$ sea la solución para cualquier entrada.

Funciones suaves

- ▶ Una función $f : \mathbb{N} \rightarrow \mathbb{R}^+$ es *eventualmente no decreciente* si

$$\exists N \in \mathbb{N}. f(n) \leq f(n+1) \text{ para todo } n \geq N$$

- ▶ Una función $f : \mathbb{N} \rightarrow \mathbb{R}^+$ es *b-suave* (smooth) si
 1. es eventualmente no decreciente y
 2. $f(bn) \in O(f(n))$.
- ▶ Una función es *suave* si es *b-suave* para todo b .
- ▶ Propiedad: si f es *b-suave* para un $b \geq 2$, entonces es suave para todo b .
- ▶ Ejemplos de funciones suaves: n^2 , n^r (para todo r), $n \lg n$.
- ▶ Ejemplos de funciones no suaves: $n^{\lg n}$, 2^n , $n!$.
- ▶ Ejercicio: verificar que n^2 es suave y que 2^n no lo es.

Regla de suavidad

- *Regla de suavidad:* Sea f suave y sea g eventualmente no decreciente. Para todo $b \geq 2$,

$$g(b^k) \in \Theta(f(b^k)) \Rightarrow g(n) \in \Theta(f(n))$$

- Ejemplo de uso: Considere la siguiente recurrencia

$$W(1) = c$$

$$W(n) = W(\lceil n/2 \rceil) + W(\lfloor n/2 \rfloor) + kn$$

- Si sólo consideramos potencias de 2 obtenemos

$$W'(1) = c$$

$$W'(n) = 2W'(n/2) + kn$$

que es $\Theta(n \lg n)$.

Uso de la Regla de suavidad

- ▶ $W'(n) \in \Theta(n \lg n)$
- ▶ Pero $W'(n) = W(n)$ para potencias de 2.
- ▶ Por lo tanto $W(2^k) = W'(2^k) \in \Theta(2^k \lg(2^k))$.
- ▶ Como $n \lg n$ es suave, por regla de suavidad,

$$W(n) \in \Theta(n \lg n)$$

- ▶ Las recurrencias que aparecen en algoritmos “Divide & Conquer” usualmente tienen la forma:

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

- ▶ En general, podemos obtener el orden de estas recurrencias directamente usando una resolución general.

“El Teorema Maestro”

El Teorema Maestro

- Dados $a \geq 1$ y $b > 1$ y la recurrencia

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

entonces

$$T(n) \in \begin{cases} \Theta(n^{\lg_b a}) & \text{si } \exists \epsilon > 0. f(n) \in O(n^{(\lg_b a) - \epsilon}) \\ \Theta(n^{\lg_b a} \lg n) & \text{si } f(n) \in \Theta(n^{\lg_b a}) \\ \Theta(f(n)) & \begin{array}{l} \text{si } \exists \epsilon > 0. f(n) \in \Omega(n^{(\lg_b a) + \epsilon}) \\ \text{y } \exists c < 1, N \in \mathbb{N}. \forall n > N. \\ af(n/b) \leq cf(n) \end{array} \end{cases}$$

- Para $W(n) = 2W(\lfloor n/2 \rfloor) + c_2 n$ estamos en el 2do caso ya que $\lg_2 2 = 1$ y $g(n) \in \Theta(n^1)$, y por lo tanto $W(n) = \Theta(n \lg n)$

Comentarios acerca del Teorema Maestro

- ▶ Los casos se deciden comparando $f(n)$ y $n^{\lg_b a}$.
 - ▶ Caso 1: $n^{\lg_b a}$ es mas grande.
 - ▶ Caso 2: $f(n)$ y $n^{\lg_b a}$ tienen el mismo orden.
 - ▶ Caso 3: $f(n)$ es mas grande.
- ▶ En realidad no basta con que una sea mas grande que la otra, sino que debe serlo polinomialmente.
 - ▶ $f(n) = n \lg n$ no es polinomialmente mas grande que $g(n) = n$.
- ▶ Los tres casos **no** cubren todas las posibilidades

Ejercicios

Resolver usando el teorema maestro.

1. $S(n) = 2S(n/4) + 3n^2$
2. $W(n) = 2W(n/2) + 14n$
3. $T(n) = 2T(n/2) + 14$

Resolución de recurrencias usando:

- ▶ Substitución
- ▶ Árbol de recurrencias.
- ▶ Regla de suavidad
- ▶ Teorema Maestro

- ▶ Introduction to Algorithms. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein
- ▶ Fundamentals of Algorithmics. Gilles Brassard, Paul Bratley