Teorema Mestre

Referências:
- Stanford Algoriths
  videos 4.1 até 4.6
- CLRS : cap. 4
- Dasgupta et. al : cap 2

1. Introdução

O teorema mestre é uma ferramenta útil para avaliar algoritmos de divisão e conquista, que normalmente precisariam de uma análise matemática mais complexa.

Por exemplo, se quisermos comparar o algoritmo do primário para multiplicação de inteiros ($O(n^2)$) com o algortmo de Karatsuba que é baseado no paradigma de divisão e conquista.

$$x = 10^{n/2} a + b$$

$$y = 10^{n/2} c + d$$

$$x \cdot y = 10^{n} ac + 10^{n/2} (ad + bc) + bd \quad (1.1)$$

Alg#1: calcula recursivamente ac, ad, bc e bd

2. Recorrência

$T(n)$ = tempo de execução máximo para resolver um problema de tamanho n. Em uma recorrência $T(n)$ é expresso em termos do trabalho executado por suas chamada recursivas

No Alg #1.

Caso base: $T(1) \leq c$

Para $n > 1$: $T(n) \leq 4T\left(\frac{n}{2}\right) + O(n)$

Alg de karatsuba: calcula recursivamente

ac, bd, $(a+b)(c+d)$

$$ad + bc = (a+b)(c+d) - ac - bd$$

- Fazer as somas apropriadas

Caso base $T(1) \leq c$

Para $n > 1$ $T(n) \leq 3T\left(\frac{n}{2}\right) + O(n)$

Exercício 2.1: Escrever a recorrência do MergeSort.

3. O teorema mestre

- Pode ser usado como uma caixa preta para resolver recorrências.
- Só funciona quando o tamanho dos subproblemas é idêntico.
- Suponha uma recorrência da seguinte forma:

$T(n) \leq c$ para n suficientemente pequeno

$$T(n) \leq a\, T\left(\frac{n}{b}\right) + O(n^d)$$

a: número de chamadas recursivas (≥ 1)

b: razão de redução dos subproblemas (> 1)

d: expoente do tempo de execução da fase de combinação

$$t(n) = \begin{cases} O(n^d \log n) & \text{se } a = b^d \\ O(n^d) & \text{se } a < b^d \\ O(n^{\log_b a}) & \text{se } a > b^d \end{cases}$$

Exemplo 3.1: <u>MergeSort</u>

$$t(n) \leq 2\,t(n/2) + O(n)$$

$a = 2 \qquad b = 2 \qquad c = 1$

$$2 \;?\; 2^1$$

- Caso 1

$$t(n) = O(n^d \log n) = O(n \log n)$$

Exercicio 3.1) aplicar o teorema mestre na busca binaria

Exemplo 3.2: Alg #1 para multiplicação de inteiros:

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

$a = 4 \qquad b = 2 \qquad d = 1$

$4 > 2^1$

Caso 3 $\quad T(n) = O(n^{\log_b a}) = O(n^{\log_2 4})$

$= O(n^2)$

Exemplo 3.3: Alg de Karatsuba

$$T(n) = 3T(n/2) + O(n)$$

$a = 3 \qquad b = 2 \qquad d = 1$

$3 > 2^1$

Caso 3 $\quad T(n) = O(n^{\log_2 3}) \leq O(n^{1,59})$

Exemplo 3.4) Strassen

$a = 7 \qquad b = 4 \qquad d = 1$

$7 > 4^1$

caso 3

$T(n) = O(n^{\log_4 7}) = O(n^{1.4})$

$O((d^2)^{1.4}) = O(d^{2,8})$

4. Prova do teorema mestre

- Vamos considerar que n é uma potencia de b

- Essa prova não é para ser 100% rigorosa, mas é para entender porque o teorema funciona, e ajudar na dedução do teorema.

- Vamos imitar a análise do MergeSort usando uma árvore de recursão

Considere um nível j da árvore de recursão. Quanto trabalho é executado nesse nível?

\# subproblemas: a^j

Tamanho $\frac{n}{b^j}$

$$\leq a^j \cdot O\left(\left(\frac{n}{b^j}\right)^d\right) \leq a^j \cdot c \cdot \frac{n^d}{b^{j \cdot d}} = c \cdot n^d \cdot \left[\frac{a}{b^d}\right]^j$$

Somando todos o níveis

$$T(n) \leq c \cdot n^d \cdot \sum_{j=0}^{\log_b n} \left[\frac{a}{b^d}\right]^j \quad !$$

a = taxa de proliferação (RSP) rate subproblem prolif

b^d = Taxa de encolhimento de (RWS) trabalho work subproblem

Exercicio 4.1: Qual a quantidade de trabalho por nível quando:

RSP < RWS

RSP > RWS

RSP = RWS

4.2 Intuição do teorema

RSP = RWS $O(n^d \log n)$ o trabalho é igual pela árvore

RSP < RWS $O(n^d)$ o trabalho esta na raiz

RSP > RWS $O(\# folhas)$ o trabalho esta nas folhas

Concluindo a prova:

$$T(n) \le c \cdot n^d \cdot \underbrace{\sum_{j=0}^{\log_b n} \left[\frac{a}{b^d}\right]^j}_{S}$$

Seja $r = \dfrac{a}{b^d}$

Se $r = 1$ $\quad S = \log_b n$

Se $r < 1$ $\quad S \le \dfrac{1}{1-r}$

Se $r > 1$ $\quad S \le n^d \cdot r^{\log_b n} \cdot c'$

Exercício 4.2| Mostre que o teorema mestre vale para o caso 1 e 2.

No caso 3 $r > 1$ e portanto

$$T(n) \le C \cdot n^d \cdot c' \cdot r^{\log_b n}$$

$$c \cdot c' \cdot n^d \cdot \frac{a^{\log_b n}}{b^{d \cdot \log_b n}} = c \cdot c' \cdot \cancel{n^d} \cdot \frac{a^{\log_b n}}{\cancel{n^d}}$$

$$= cc' \cdot a^{\log_b n} = cc \cdot n^{\log_b a} = O\left(n^{\log_b a}\right)$$