Resolução de recorrências

Fernando Lobo

Algoritmos e Estrutura de Dados

Sumário

Recorrências

- Uma recorrência é uma função definida em termos de:
  - 1 ou + casos base.
  - a própria função, com argumentos mais pequenos.
- Aparecem naturalmente na análise de algoritmos recursivos.
- Temos de aprender a resolvê-las.

Métodos de Resolução de Recorrências

- **Método de Substituição**: método mais geral, mas é necessário adivinhar a forma da solução.
- **Método Iterativo**: método pouco formal, mas bastante intuitivo.
- **Teorema Mestre**: cobre bastantes casos, mas não todos.

Exemplo

```
FACTORIAL(n)
    if n = 0
        return 1
    else
        return n * FACTORIAL(n-1)
```

$$
\begin{aligned}
T(n) &= \begin{cases} \Theta(1) & \text{, se } n = 0 \\ T(n-1) + \Theta(1) & \text{, se } n > 0 \end{cases} \\
&= \begin{cases} a & \text{, se } n = 0 \\ T(n-1) + b & \text{, se } n > 0 \end{cases}
\end{aligned}
$$

onde a e b são constantes.

Resolução através do método iterativo ou de expansão

$$
\begin{aligned}
T(n) &= T(n-1) + b \\
&= T(n-2) + b + b \\
&= T(n-3) + b + b + b \\
&= \ldots \\
&= T(n-n) + \underbrace{b + b + \ldots + b}_{\text{n vezes}} \\
&= a + bn \\
&= \Theta(n)
\end{aligned}
$$

- Casos simples podem ser resolvidos facilmente iterando a função até chegar ao caso base.
- Para casos mais complexos (ex: MERGE-SORT) também podemos "desenrolar" a recorrência como fizemos na aula passada.

Método de Substituição

1. Adivinhar a forma da solução (normalmente após aplicar o método iterativo)
2. Verificar por indução matemática.
3. Resolver para obter as constantes.

Exemplo com a função factorial

Exemplo

$$T(n) = T(n-1) + b$$

- Vamos assumir que $T(1) = a = \Theta(1)$.
- Adivinhar que $T(n) = O(n)$, i.e., $T(n) \leq cn$ para um $c > 0, n > n_0$.
- Assumimos por hipótese que: $T(k) \leq ck$, para $k < n$
- Tentamos provar que $T(n) \leq cn$ por indução.

$$
\begin{aligned}
T(n) &= T(n-1) + b \\
&\leq c(n-1) + b \\
&= cn - c + b \\
&= \underbrace{cn}_{\text{desejado}} - \underbrace{(c-b)}_{\textit{residual}} \\
&\leq cn \quad \text{, sempre que } c - b \geq 0.
\end{aligned}
$$

Para tal basta escolher quaisquer $c \geq b$ e $n_0 \geq 1$

Teorema Mestre

- Serve para resolver grande parte das recorrências da forma:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

onde $a \geq 1, b > 1, f(n) > 0, \forall n > n_0$.

- Note que a recorrência do MERGE-SORT é um caso particular com $a = 2$, $b = 2$, $f(n) = cn$
- Ao expandirmos a recorrência do MERGE-SORT observamos que todos os níveis da árvore tinham a mesma soma cn, daí termos obtido que a soma total era $\mathrm{O}(cn \cdot h) = \mathrm{O}(n \lg n)$.
- Contudo, no caso geral, a soma em cada nível da árvore nem sempre é igual.

Teorema Mestre

- No caso geral, se expandirmos a árvore de recorrência, irá acontecer um dos seguintes 3 casos:
  - O custo é dominado pelas folhas da árvore. (Caso 1)
  - O custo é igual em cada nível da árvore. (Caso 2)
  - O custo é dominado pelo raíz. (Caso 3)
- Distingue-se qual o caso, comparando,

$$\boxed{n^{log_b a} \text{ com } f(n)}$$

Teorema Mestre: Caso 1

Caso 1: $f(n) = O\left(n^{log_b a - \epsilon}\right)$ para um $\epsilon > 0$
($f(n)$ é polinomiamente inferior a $n^{log_b a}$)

Solução:

$$\boxed{T(n) = \Theta(n^{log_b a})}$$

Intuitivamente: custo é dominado pelas folhas da árvore de recorrência.

Teorema Mestre: Caso 2

Caso 2: $f(n) = \Theta\left(n^{\log_b a}\right)$

Solução:

$$\boxed{T(n) = \Theta(n^{\log_b a} \lg n)}$$

Intuitivamente: custo é idêntico em cada nível da árvore de recorrência.

Teorema Mestre: Caso 3

Caso 3:

- $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$ para um $\epsilon > 0$
  ($f(n)$ é polinomiamente superior a $n^{\log_b a}$)

- $af\left(\frac{n}{b}\right) \leq cf(n)$ para um $c < 1$ e para todo $n > n_0$
  (Chama-se a isto *Condição de Regularidade*)

Solução:

$$\boxed{T(n) = \Theta(f(n))}$$

Intuitivamente: custo é dominado pela raíz da árvore de recorrência.

A condição de regularidade é sempre verdadeira quando $f(n) = n^k$.

Teorema Mestre

Exemplo 1

$$T(n) = \quad 4T(\tfrac{n}{2}) + n$$

$$\swarrow \quad \downarrow \quad \searrow$$

$$a \qquad b \qquad f(n)$$

Comparar $n^{\log_b a}$ com $f(n)$

$$\downarrow \qquad \searrow$$

$$n^{\log_2 4} = n^2 \qquad n$$

Caso 1: $f(n) = n = O(n^{2-\epsilon})$ para $\epsilon = 1$, logo

$$\boxed{T(n) = \Theta(n^2)}$$

Teorema Mestre

Exemplo 2

$$T(n) = \quad 4T(\tfrac{n}{2}) + n^2$$

$$\downarrow \qquad \searrow \text{f(n)}$$

$$n^{\log_b a} = n^{\log_2 4} = n^2 \qquad n^2$$

Caso 2: $f(n) = n^2 = \Theta(n^2)$, logo

$$\boxed{T(n) = \Theta(n^2 \lg n)}$$

Teorema Mestre

Exemplo 3 (recorrência de MERGE-SORT)

$$T(n) = \quad 2T(\tfrac{n}{2}) + n$$

$$n^{log_b a} \quad = n^{log_2 2} = n$$

Caso 2: $f(n) = n = \Theta(n)$, logo

$$\boxed{T(n) = \Theta(n \lg n)}$$

Teorema Mestre

Exemplo 4

$$T(n) = \qquad 4T(\tfrac{n}{2}) + n^3$$

$$\qquad\qquad\qquad\qquad\qquad\quad \searrow \text{f(n)}$$

$$n^{log_b a} \quad = n^{log_2 4} = n^2 \qquad n^3$$

Caso 3:

- $f(n) = n^3 = \Omega(n^{2+\epsilon}) \quad$ para $\epsilon = 1$
- Condição de Regularidade:

$$4\left(\frac{n}{2}\right)^3 \leq cn^3 \Leftrightarrow \frac{4n^3}{8} \leq cn^3$$

$$\Leftrightarrow \quad \frac{1}{2}n^3 \leq cn^3 \quad \text{,basta escolher } \frac{1}{2} \leq c < 1$$

  - Não era necessário mostrar a Condição de Regularidade porque $f(n)$ é polinomial.

Logo,

$$\boxed{T(n) = \Theta(n^3)}$$

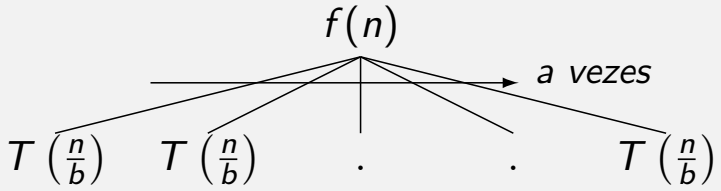
Teorema Mestre

- Demonstração formal está no livro [CLRS].
- Este formato de $T(n)$ aparece em problemas de D&C:

$$T(n) = \begin{cases} \Theta(1) & \text{, se } n = 1 \\ aT\left(\frac{n}{b}\right) + f(n) & \text{, se } n > 1 \end{cases} \qquad \text{onde } a \geq 1, b > 1$$

- a: *número de subproblemas*
- $\frac{n}{b}$: *tamanho de cada subproblema*
- $f(n)$: *trabalho não recursivo*

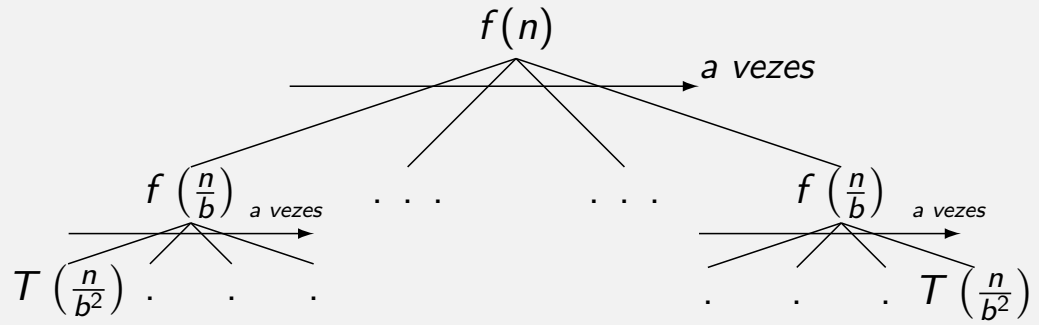
$T(n) =$



Teorema Mestre (intuição)

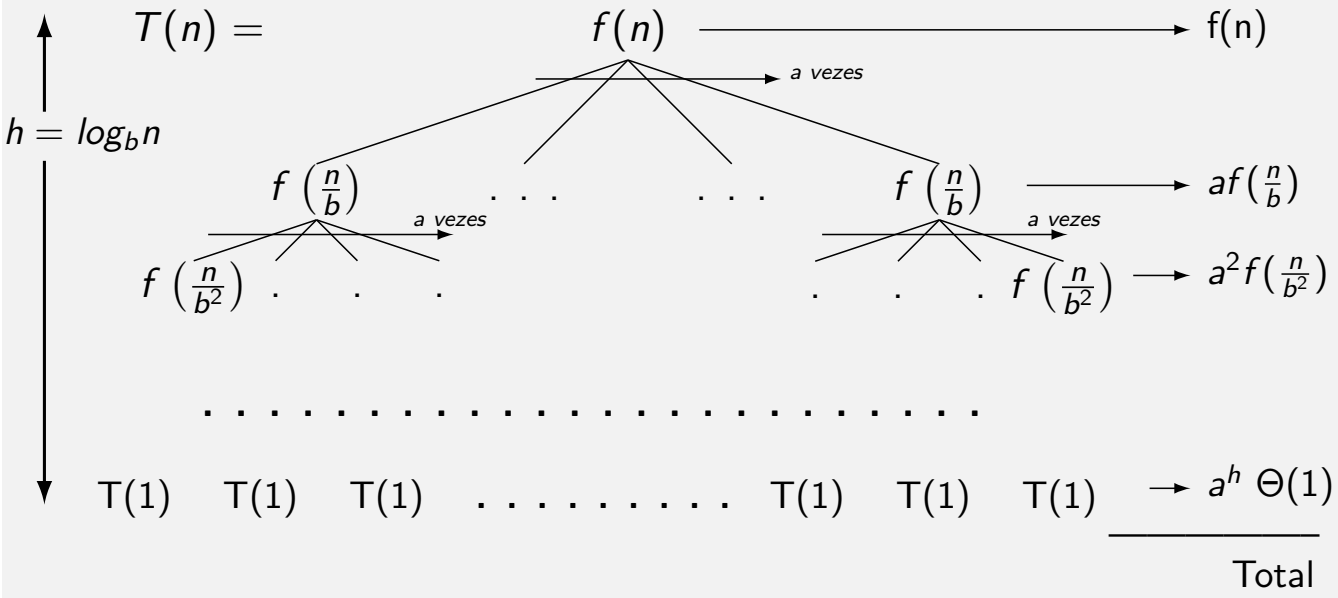
Iterando:

$T(n) =$



Teorema Mestre (intuição)

Iterando mais vezes:



- A altura da árvore é $h = \log_b n$
- No último nível temos a^h subproblemas de tamanho 1.

$$\begin{aligned} a^h\,\Theta(1) &= a^{\log_b n}\,\Theta(1) \\ &= n^{\log_b a}\,\Theta(1) \\ &= \Theta(n^{\log_b a}) \end{aligned}$$

Soma nível por nível:

$$\text{Total} = \underbrace{f(n) + af\left(\frac{n}{b}\right) + a^2 f\left(\frac{n}{b^2}\right) + .. + a^{h-1} f\left(\frac{n}{b^{h-1}}\right)}_{} + \Theta(n^{\log_b a})$$

$$T(n) = \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) + \Theta(n^{\log_b a})$$

Se $f(n)$ é polinomial, i.e., $f(n) = n^k$ para algum k fixo, resulta:

$$\begin{aligned} T(n) &= \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^k + \Theta(n^{\log_b a}) \\ &= n^k * \underbrace{\sum_{i=0}^{log_b n - 1} \left(\frac{a}{b^k}\right)^i}_{\text{progressão geométrica com } \log_b n \text{ parcelas e } r=\frac{a}{b^k}} + \Theta(n^{\log_b a}) \end{aligned}$$

Daqui obtemos:

- Se $r > 1$

$$\log_b a > k \Rightarrow T(n) = \Theta(n^{\log_b a}) \qquad \text{(Caso 1)}$$

- Se $r = 1$

$$\log_b a = k \Rightarrow T(n) = \Theta(n^k \log_b n) \quad \text{(Caso 2)}$$

- Se $r < 1$

$$\log_b a < k \Rightarrow T(n) = \Theta(n^k) \qquad \text{(Caso 3)}$$