# Capítulo 5

# Equações de Recorrência

O paradigma de divisão e conquista anda junto com as equações de recorrência já que estas equações fornecem uma caracterização natural para algoritmos recursivos. Nesta seção estudaremos alguns métodos que nos permitirão resolver equações de recorrência que caracterizam o tempo de execução de alguns algoritmos.

## 5.1 O Método da substituição

O primeiro método que estudaremos para resolver equações de recorrência é conhecido como o *método da substituição*. Este método consiste em duas etapas:

1. Inicialmente "adivinhamos"uma solução para a recorrência;
2. Em seguida, utilizamos indução para mostrar que a solução está correta.

Na aula anterior vimos uma aplicação deste método para provar que o algoritmo *mergesort* tem complexidade $\Theta(n.\lg n)$, onde $n$ denota o tamanho da entrada. Como outro exemplo, resolva o exercício a seguir:

**Exercício 5.1.1.** *Escreva a equação de recorrência que modela a complexidade de tempo do algoritmo de ordenação por inserção recursivo em função do número $n$ de elementos da lista de entrada, e resolva esta recorrência. Observe que a estrutura de listas é definida pela gramática: $l ::= nil \mid a :: l$ onde $nil$ representa a lista vazia, e $a :: l$ representa a lista com primeiro elemento $a$ e cauda $l$.*

## 5.2 O Método das árvores de recursão

Encontrar um bom chute para resolver uma recorrência pelo método da substituição pode ser uma tarefa difícil. Neste caso, uma árvore de recursão pode ajudar. Em uma árvore de recursão cada nó representa o custo de um subproblema, de forma que a soma dos custos em cada nível da árvore corresponde ao custo no nível correspondente da recorrência, e a soma dos custos por nível fornece o custo total em todos os níveis da recorrência. Para exemplos envolvendo este método veja a Seção 4.4 de [2].

## 5.3 O Teorema Mestre - versão 1

Nesta seção veremos um teorema que nos permite resolver diversas equações de recorrência que possuem um determinada estrutura. Este teorema, conhecido como *teorema mestre*, é bastante útil porque nos permite resolver de forma imediata as equações que satisfazem as hipóteses do teorema. Também veremos a razão pela qual o teorema funciona, isto é, veremos a prova do teorema. Iniciaremos com algumas definições sobre o crescimento de funções.

**Definição 5.3.1.** *Seja $f(n)$ uma função não-negativa definida no conjunto dos números naturais. Dizemos que $f(n)$ é* **eventualmente não-decrescente** *se existir um número inteiro $n_0$ tal que $f(n)$ é não-decrescente no intervalo $[n_0, \infty)$, ou seja,*

$$f(n_1) \leq f(n_2), \forall n_2 > n_1 \geq n_0.$$

**Definição 5.3.2.** *Seja $f(n)$ uma função não-negativa definida no conjunto dos números naturais. Dizemos que $f(n)$ é* **suave** *se for eventualmente não-decrescente e*

$$f(2.n) = \Theta(f(n))$$

Intuitivamente, uma função suave é uma função que cresce devagar. Um bom exercício é mostrar que a função logaritmo ou a função linear são suaves, enquanto que a função exponencial não é suave.

**Teorema 5.3.3.** *Seja $f(n)$ uma função suave. Então para qualquer $b \geq 2$ fixado,*

$$f(b.n) = \Theta(f(n))$$

*Demonstração.* Exercício. □

**Teorema 5.3.4** (Regra da suavização)**.** *Seja $T(n)$ uma função eventualmente não-decrescente, e $f(n)$ uma função suave. Se $T(n) = \Theta(f(n))$ para valores de $n$ que são potências de $b$ ($b \geq 2$), então*

$$T(n) = \Theta(f(n)), \forall n.$$

*Demonstração.* Exercício. □

A regra da suavização nos permite expandir a informação sobre a ordem de crescimento estabelecida para $T(n)$ de um subconjunto de valores (potências de $b$) para o domínio inteiro dos números naturais.

**Teorema 5.3.5** (Teorema Mestre - versão 1)**.** *Seja $T(n)$ uma função eventualmente não-decrescente que satisfaz a recorrência*

$$\begin{array}{l} T(n) = a.T(n/b) + f(n), \quad \text{para } n = b^k, k = 1, 2, 3, \ldots \\ T(1) = c \end{array}$$

*onde $a \geq 1, b \geq 2$ e $c > 0$ . Se $f(n) = \Theta(n^d)$, onde $d \geq 0$, então*

$$T(n) = \begin{cases} \Theta(n^d), & \text{se } a < b^d \\ \Theta(n^d . \lg n), & \text{se } a = b^d \\ \Theta(n^{\log_b a}), & \text{se } a > b^d \end{cases}$$

*Demonstração.* Considere que $f(n) = n^d$. Aplicando o método da substituição para a recorrência do teorema, obtemos:

$$T(b^k) = a^k.[T(1) + \sum_{j=1}^{k} f(b^j)/a^j]$$

Como $a^k = a^{\log_b n} = n^{\log_b a}$, podemos reescrever a equação acima como:

$$T(n) = n^{\log_b a}.[T(1) + \sum_{j=1}^{\log_b n} f(b^j)/a^j]$$

e para $f(n) = n^d$, temos:

$$T(n) = n^{\log_b a}.[T(1) + \sum_{j=1}^{\log_b n} (b^j)^d/a^j] = n^{\log_b a}.[T(1) + \sum_{j=1}^{\log_b n} (b^d/a)^j]$$

A soma acima forma uma série geométrica, e portanto:

$$\sum_{j=1}^{\log_b n} (b^d/a)^j = (b^d/a)\frac{(b^d/a)^{\log_b n} - 1}{(b^d/a) - 1}, \text{ se } b^d \neq a.$$

Quando $b^d \neq a$, temos que $\sum_{j=1}^{\log_b n} (b^d/a)^j = \log_b n$. Agora basta analisarmos cada um dos casos: $a < b^d$, $a > b^d$ e $a = b^d$. □