

# Algoritmos Recursivos e Relações de Recorrência

Prof. André Vignatti

Relações de recorrência devem ser obtidas e resolvidas para determinar o tempo de execução de algoritmos recursivos.

Análise de algoritmos recursivos:

- Relações de Recorrência
- Como obtê-las?
- Como resolvê-las?

## 1 Obtendo Relações de Recorrência

Seja  $T(n)$  o tempo de execução do algoritmo para entrada de tamanho  $n$ .

Para obter relações de recorrência, a partir de um algoritmo recursivo:

- Descobrir qual é a entrada e qual o tamanho  $n$  desta entrada.
- Ver qual valor de  $n$  é usado como base da recursão. Geralmente será um único valor (por exemplo,  $n = 1$ ), ou podem ser vários valores (por exemplo,  $n \leq 1$ ). Seja  $n_0$  esse valor.
- Descubra o que é  $T(n_0)$  (geralmente fácil). Você pode geralmente usar “alguma constante  $c$ ”, mas às vezes um número específico será necessário.
- O  $T(n)$  geral será geralmente a soma de outros  $T(k)$  (as chamadas recursivas) mais o resto do trabalho executado pelo algoritmo. Geralmente as chamadas recursivas irão resolver  $a$  subproblemas de tamanho  $f(n)$ , fornecendo o termo  $aT(f(n))$  na relação de recorrência.

```

1 Algoritmo  $f(n)$ 
2   se  $n = 1$  então faz algo
3   senão
4      $f(n - 1)$ 
5      $f(n - 2)$ 
6     para  $i \leftarrow 1$  até  $n$  faça
7       faz alguma coisa

```

Esboçar uma possível relação de recorrência.

$$T_f(n) = \begin{cases} & \text{se } n \text{ está na base} \\ & \text{se } n \text{ não está na base} \end{cases}$$

```

1 Algoritmo  $g(n)$ 
2   se  $n = 1$  ou  $n = 2$  então faz algo
3   senão
4      $g(n - 1)$ 
5     para  $i \leftarrow 1$  até  $n$  faça
6       faz alguma coisa
7      $g(n - 1)$ 

```

Esboçar uma possível relação de recorrência.

$$T_g(n) = \begin{cases} & \text{se } n \text{ está na base} \\ & \text{se } n \text{ não está na base} \end{cases}$$

```

1 Algoritmo  $h(n)$ 
2   se  $n \leq 1$  então faz algo
3   senão se  $n = 2$  então faz outra coisa
4   senão
5     para  $i \leftarrow 1$  até  $n$  faça
6        $h(n - 1)$ 
7       faz alguma coisa diferente

```

Esboçar uma possível relação de recorrência.

$$T_h(n) = \begin{cases} & \text{se } n \text{ está na base} \\ & \text{se } n \text{ não está na base} \end{cases}$$

## 2 Análise da Multiplicação

```
1 Algoritmo multiplica( $y, z$ )
2   se  $z = 0$  então retorna 0
3   senão se  $z$  é ímpar então
4     retorna  $\text{multiplica}(2y, \lfloor z/2 \rfloor) + y$ 
5   senão retorna  $\text{multiplica}(2y, \lfloor z/2 \rfloor)$ 
```

Seja  $T(n)$  o tempo de execução de  $\text{multiplica}(y, z)$ , onde  $z$  é um número de  $n$  bits.

Então, para constantes  $c, d \in \mathbb{R}$ ,

$$T(n) = \begin{cases} c & \text{se } n = 1 \\ T(n-1) + d & \text{caso contrário} \end{cases}$$

## 3 Resolvendo Relações de Recorrência

Usar **substituições repetidas** – “abrir” a recorrência usando a própria definição recursiva.

Dada uma recorrência  $T(n)$ :

- Substituir algumas vezes até achar um padrão
- Partindo do padrão, escrever uma fórmula em termos de  $n$  e o número de substituições  $i$ .
- Escolher  $i$  tal que todas referências à  $T()$  se tornem referências ao caso base.
- Simplificar e resolver as somas.

Isso não irá sempre funcionar, mas funciona na maioria das vezes na prática.

## 4 Resolvendo a Recorrência da Multiplicação

Sabemos que para todo  $n \geq 1$ ,

$$T(n) = T(n-1) + d.$$

Portanto,

$$\begin{aligned}
T(n) &= T(n-1) + d \\
T(n-1) &= T(n-2) + d \\
T(n-2) &= T(n-3) + d \\
&\vdots \\
T(2) &= T(1) + d \\
T(1) &= c
\end{aligned}$$

Assim, substituindo repetidas vezes na recorrência,

$$\begin{aligned}
T(n) &= T(n-1) + d \\
&= (T(n-2) + d) + d \\
&= T(n-2) + 2d \\
&= (T(n-3) + d) + 2d \\
&= T(n-3) + 3d
\end{aligned}$$

Há um padrão se formando! Parece que, após  $i$  substituições,

$$T(n) = T(n-i) + id.$$

Agora, escolhendo  $i = n - 1$ , temos

$$\begin{aligned}
T(n) &= T(1) + d(n-1) \\
&= dn + c - d.
\end{aligned}$$

## Aviso!

Isso **NÃO** é uma demonstração. Há um “pulo” na lógica.

De onde  $T(n) = T(n-i) + id$  veio? De “achismo” e mathematical embromation!

Como podemos provar isso? Duas opções:

- Provar o **padrão** ( $T(n) = T(n-i) + id$ ) por indução em  $i$ .
- Prova o **resultado** ( $T(n) = dn + c - d$ ) por indução em  $n$

Vamos provar usando o resultado.

**Teorema.** A relação de recorrência:

$$T(n) = \begin{cases} c & \text{se } n = 1 \\ T(n-1) + d & \text{caso contrário} \end{cases}$$

tem como solução

$$T(n) = dn + c - d$$

*Demonstração.* (Indução em  $n$ )

**Base:**  $n = 1$ , temos que  $T(1) = d \cdot 1 + c - d = c$ .

**Hipótese:** Para todo  $k < n$ ,  $T(k) = dk + c - d$ .

**Passo:** Queremos provar que  $T(n) = dn + c - d$ . Pela definição da recorrência,

$$T(n) = T(n-1) + d,$$

e usando a hipótese,

$$T(n) = (d(n-1) + c - d) + d = d(n-1) + c = dn + c - d.$$

□

De fato, a **solução definitiva** para recorrências é a **prova por indução**.

- Outras estratégias servem somente para **adivinhar** a solução.