

FUNÇÕES DE RECORRÊNCIA E O TEOREMA MESTRE

DCE529 - Algoritmos e Estruturas de Dados III

Atualizado em: 7 de agosto de 2024

Iago Carvalho

Departamento de Ciência da Computação



Diversos algoritmos em computação são recursivos

- Sequência de fibonacci
- Algoritmos de programação dinâmica
- Fatorial
- ...

Como podemos calcular a complexidade computacional destes algoritmos?

EQUAÇÕES DE RECORRÊNCIA

Quando um algoritmo contém uma **chamadas recursivas**, seu tempo de execução pode frequentemente ser descrito por uma equação de **recorrência**

Complexidade de um *loop* (for)

$$\sum_{i=0}^n f(n)$$

Complexidade de um algoritmo recursivo

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

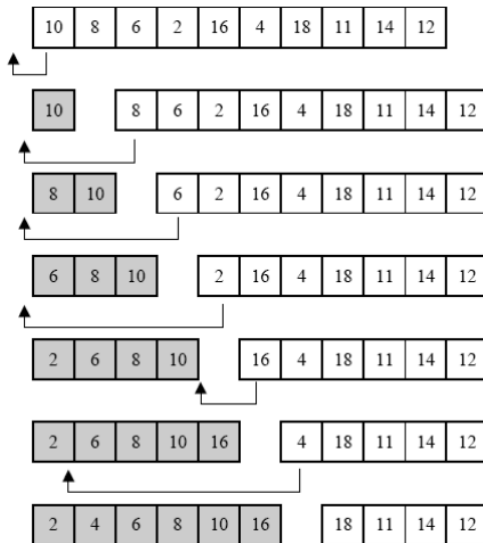
EQUAÇÕES DE RECORRÊNCIA

Uma recorrência é uma equação ou desigualdade que descreve uma função em termos de seu valor em entradas menores

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

Para cada procedimento recursivo é associada uma função de complexidade **T(n)** desconhecida, onde n mede o tamanho dos argumentos do procedimento

ORDENAÇÃO POR INSERÇÃO



ORDENAÇÃO POR INSERÇÃO

Vamos considerar o pior caso deste algoritmo. Ou seja, o número sempre é inserido na última posição verificada

1. Faz uma comparação
2. Faz duas comparações
3. Faz três comparações
4. ...

Na inserção do n -ésimo número, ele poder fazer um total de n comparações

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

$$T(n-1) = T(n-2) + 2$$

$$T(n-2) = T(n-3) + 3$$

$$T(n-3) = T(n-4) + 4$$

...

$$T(2) = T(1) + n - 2$$

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

ORDENAÇÃO POR INSERÇÃO

$$T(n) = 1 + 2 + 3 + 4 + \dots + (n-2) + (n-1)$$

$$T(n) = \sum_{i=1}^{n-1} i = \left(\sum_{i=1}^n i \right) - n = \left(\frac{n(n+1)}{2} \right) - n$$

$$T(n) = \frac{n^2 + n}{2} - n = \frac{n^2 + n - 2n}{2}$$

$$T(n) = \frac{n^2 - n}{2}$$

$$T(n) = \frac{n^2 - n}{2} \in \Theta(n^2)$$

OUTRA EQUAÇÃO DE RECORRÊNCIA

Considere o pseudo-código abaixo

- O algoritmo inspeciona n elementos de um conjunto qualquer

De alguma forma, isso permite descartar $\frac{2}{3}$ dos elementos e fazer uma chamada recursiva sobre um terço do conjunto original

```
1 algoritmo Pesquisa (vetor)
2   if vetor.size() <= 1 then
3     inspecione elemento;
4   else
5     inspecione cada elemento recebido (vetor);
6     Pesquisa(vetor.subLista(1, (vetor.size() / 3)));
7   end if
8 end.
```

OUTRA EQUAÇÃO DE RECORRÊNCIA

```
1 algoritmo Pesquisa (vetor)
2   if vetor.size() <= 1 then
3     inspecione elemento;
4   else
5     inspecione cada elemento recebido (vetor);
6     Pesquisa(vetor.subLista(1, (vetor.size() / 3)));
7   end if
8 end.
```

$\Theta(1)$ 2
 $\Theta(1)$ 3
 $\Theta(n)$ 5
Chamada recursiva 6

$$\text{Chamada recursiva} = T\left(\frac{n}{3}\right) + n$$

OUTRA EQUAÇÃO DE RECORRÊNCIA

$$T(n) = \begin{cases} 1, & \text{se } n \leq 1 \\ t\left(\frac{n}{3}\right) + n, & \text{caso contrario} \end{cases}$$

OUTRA EQUAÇÃO DE RECORRÊNCIA

$$\begin{aligned}T(n) &= n + T(n/3) \\T(n/3) &= n/3 + T(n/3/3) \\T(n/3/3) &= n/3/3 + T(n/3/3/3) \\T(n/3/3/3) &= n/3/3/3 + T(n/3/3/3/3) \\&\dots \\T(n/3/3/3/\dots/3) &= n/3/3/3/\dots/3 + T(n/3/3/3/\dots/3/3) \\T(1) &= 1\end{aligned}$$

OUTRA EQUAÇÃO DE RECORRÊNCIA

$$\begin{aligned}T(n) &= n + \cancel{T(n/3)} \\ \cancel{T(n/3)} &= n/3 + \cancel{T(n/3/3)} \\ \cancel{T(n/3/3)} &= n/3/3 + \cancel{T(n/3/3/3)} \\ \cancel{T(n/3/3/3)} &= n/3/3/3 + \cancel{T(n/3/3/3/3)} \\ &\vdots \\ \cancel{T(n/3/3/3/\dots/3)} &= n/3/3/3/\dots/3 + \cancel{T(n/3/3/3/\dots/3/3)} \\ \cancel{T(1)} &= 1\end{aligned}$$

$$T(n) = n + n/3 + n/3/3 + n/3/3/3 + \dots + n/3/3/3/\dots/3 + 1$$

OUTRA EQUAÇÃO DE RECORRÊNCIA

$$T(n) = \begin{cases} 1, & \text{se } n \leq 1 \\ T\left(\frac{n}{3}\right) + n, & \text{caso contrario} \end{cases}$$

$$T(n) = n + n/3 + n/3/3 + n/3/3/3 + \dots + n/3/3/3/\dots/3 + 1$$

$$T(n) = n \sum_{i=1}^{n-1} \left(\frac{1}{3}\right)^i + 1$$

OUTRA EQUAÇÃO DE RECORRÊNCIA

Levando em consideração que

$$\sum_{i=1}^{n-1} x^i = \frac{1}{1-x},$$

temos que

$$T(n) = n \sum_{i=1}^{n-1} \left(\frac{1}{3}\right)^i + 1 = n \left(\frac{1}{1 - \frac{1}{3}} \right) + 1 = \frac{3n}{2} + 1 = \Theta(n)$$

DIFÍCIL, NÉ?

TEOREMA MESTRE

O teorema mestre é um conjunto de regras para resolvermos equações de recorrência

Ele resolve equações de recorrência no formato

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

onde

- $a \geq 1$ e $b > 1$ são constantes
- $f(n)$ é uma função assintoticamente positiva

TEOREMA MESTRE

O teorema mestre possui 3 regras, a depender do formato da função $f(n)$

Caso 1: $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$, com $\epsilon > 0$

Caso 2: $f(n) = \mathcal{O}(n^{\log_b a})$

Caso 3: $f(n) = \Omega(n^{\log_b a + \epsilon})$, com $\epsilon > 0$, desde que $af\left(\frac{n}{b}\right) \leq cf(n)$ para alguma constante $c < 1$ e n suficientemente grande

TEOREMA MESTRE

Caso 1: $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$, então $T(n) = \Theta(n^{\log_b a})$

Caso 2: $f(n) = \mathcal{O}(n^{\log_b a})$, então $T(n) = \Theta(n^{\log_b a} \log n)$

Caso 3: $f(n) = \Omega(n^{\log_b a + \epsilon})$, então $T(n) = \Theta(f(n))$

TEOREMA MESTRE - EXEMPLOS

$$T(n) = 9T\left(\frac{n}{3}\right) + n$$

Neste caso, temos que $a = 9$, $b = 3$ e $f(n) = n$

Podemos aplicar o caso 1 do teorema mestre:

- $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$, então $T(n) = \Theta(n^{\log_b a})$
- $\epsilon = 1$

Temos que $T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$

$$T(n) = T\left(\frac{2n}{3}\right) + 1$$

Neste caso, temos que $a = 1$, $b = \frac{3}{2}$ e $f(n) = 1$

Podemos aplicar o caso 2 do teorema mestre:

- $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$, então $T(n) = \Theta(n^{\log_b a} \log n)$
- $\log_{3/2} 1 = 0$

Temos que $T(n) = \Theta(n^{\log_b a} \log n) = \Theta(\log n)$

TEOREMA MESTRE - EXEMPLOS

$$T(n) = 3T\left(\frac{n}{4}\right) + n \log_2 n$$

Neste caso, temos que $a = 3$, $b = 4$ e $f(n) = n \log_2 n$

Podemos aplicar o caso 3 do teorema mestre:

- $f(n) = \Omega(n^{\log_b a + \epsilon})$, então $T(n) = \Theta(f(n))$
- $\log_4 3 = 0,7924812504 \dots$
- $\epsilon = 1 - \log_4 3 = 1 - 0,7924812504 \dots = 0,2075187496 \dots$

Agora precisamos verificar se a condição de regularidade $af\left(\frac{n}{b}\right) \leq cf(n)$ é satisfeita

$$af\left(\frac{n}{b}\right) \leq cf(n), \quad a = 3, b = 4$$

$$af(n/b) \leq cf(n)$$

$$3 \times \frac{n}{4} \log_2 \frac{n}{4} \leq cn \log_2 n$$

$$\frac{3}{4}n (\log_2 n - \log_2 4) \leq cn \log_2 n$$

$$\frac{3}{4}n (\log_2 n - 2) \leq cn \log_2 n$$

$$\frac{3}{4}n \log_2 n - 2 \times \frac{3}{4}n \leq cn \log_2 n$$

$$\frac{3}{4}n \log_2 n - \frac{3}{2}n \leq cn \log_2 n$$

TEOREMA MESTRE - EXEMPLOS

Considerando $c = \frac{3}{4}$

$$\begin{aligned}\frac{3}{4}n \log_2 n - \frac{3}{2}n &\leq \frac{3}{4}n \log_2 n \\ -\frac{3}{2}n &\leq 0 \\ \frac{3}{2}n &\geq 0 \\ n &\geq 0\end{aligned}$$

Confirmada a condição de regularidade, temos então que

$$T(n) = \Theta(f(n)) = \Theta(n \log_2 n)$$