Análise de Algoritmos

Solução de recorrências

Recorrências

Uma recorrência é uma expressão que dá o valor de uma função em termos dos valores "anteriores" da mesma função.

Recorrências

Uma recorrência é uma expressão que dá o valor de uma função em termos dos valores "anteriores" da mesma função.

Para analisar o consumo de tempo de um algoritmo recursivo é necessário resolver uma recorrência.

Recorrências

O exemplo clássico de recorrência, provavelmente o mais famoso, é a fórmula de Fibonacci:

$$F(n) = \begin{cases} 0 & \text{se } n = 0 \\ 1 & \text{se } n = 1 \\ F(n-1) + F(n-2) & \text{se } n \geq 2 \end{cases}$$

Recorrências

O exemplo clássico de recorrência, provavelmente o mais famoso, é a fórmula de Fibonacci:

$$F(n) = \begin{cases} 0 & \text{se } n = 0 \\ 1 & \text{se } n = 1 \\ F(n-1) + F(n-2) & \text{se } n \geq 2 \end{cases}$$

Uma "fórmula fechada" para $F(n)$ é dada por:

$$F(n) = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^n$$

Recorrências

Um outro exemplo clássico de recorrência é:

$$F(n) = \begin{cases} 1 & \text{se } n = 0 \\ n.F(n-1) & \text{se } n \geq 1 \end{cases}$$

Recorrências

Um outro exemplo clássico de recorrência é:

$$F(n) = \begin{cases} 1 & \text{se } n = 0 \\ n.F(n-1) & \text{se } n \geq 1 \end{cases}$$

Uma fórmula fechada para $F(n)$ é dada por:

$$\begin{aligned} F(n) &= n.F(n-1) \\ &= n.(n-1).F(n-2) = \cdots = \\ &= n.(n-1).(n-2)\ldots 2.1.1 \\ &= n! \end{aligned}$$

Recorrências

Resolver uma recorrência é encontrar uma "fórmula fechada" que dê o valor da função diretamente em termos do seu argumento.

Recorrências - busca binária

Vamos analisar novamente o consumo de tempo do algoritmo da busca binária.

Recorrências - busca binária

Vamos analisar novamente o consumo de tempo do algoritmo da busca binária. Em cada iteração, o algoritmo descarta metade do vetor.

Recorrências - busca binária

Vamos analisar novamente o consumo de tempo do algoritmo da busca binária. Em cada iteração, o algoritmo descarta metade do vetor. Se denotamos por $T(n)$ o número máximo de iterações realizadas pela busca binária sobre um vetor com n elementos,

Recorrências - busca binária

Vamos analisar novamente o consumo de tempo do algoritmo da busca binária. Em cada iteração, o algoritmo descarta metade do vetor. Se denotamos por $T(n)$ o número máximo de iterações realizadas pela busca binária sobre um vetor com n elementos, a função $T(n)$ pode ser expressa pela seguinte recorrência:

$$T(n) = T(\left\lfloor \frac{n}{2} \right\rfloor) + 1, \quad T(1) = 1$$

Recorrências - busca binária

Para obter uma solução, supomos inicialmente que $n = 2^k$.

Recorrências - busca binária

Para obter uma solução, supomos inicialmente que $n = 2^k$.

$$
\begin{aligned}
T(n) &= T\left(\frac{n}{2}\right) + 1 \\
&= \left(T\left(\frac{n}{2^2}\right) + 1\right) + 1 = T\left(\frac{n}{2^2}\right) + 2 \\
&= \left(T\left(\frac{n}{2^3}\right) + 1\right) + 2 = T\left(\frac{n}{2^3}\right) + 3 \\
&= \cdots \\
&= T\left(\frac{n}{2^k}\right) + k = k + 1 = \log n + 1
\end{aligned}
$$

Recorrências - busca binária

Podemos agora tentar mostrar que

$T(n) \leq \log n + 1, \quad \forall n \geq 1$, por indução em n.

Recorrências - busca binária

Podemos agora tentar mostrar que

$T(n) \leq \log n + 1, \quad \forall n \geq 1$, por indução em n.

- Para $n = 1$, $T(1) = 1 = \log 1 + 1$

Recorrências - busca binária

Podemos agora tentar mostrar que

$T(n) \leq \log n + 1, \quad \forall n \geq 1$, por indução em n.

- Para $n = 1$, $T(1) = 1 = \log 1 + 1$
- Para $n > 1$, temos

Recorrências - busca binária

Podemos agora tentar mostrar que

$T(n) \leq \log n + 1, \quad \forall n \geq 1$, por indução em n.

- Para $n = 1$, $T(1) = 1 = \log 1 + 1$
- Para $n > 1$, temos

$$\begin{aligned} T(n) &= T(\left\lfloor \frac{n}{2} \right\rfloor) + 1 \\ &\leq (\log(\left\lfloor \frac{n}{2} \right\rfloor) + 1) + 1 \\ &\leq \log\left(\frac{n}{2}\right) + 2 = (\log n - 1) + 2 \\ &= \log n + 1 \end{aligned}$$

Recorrências - Exemplo

Consideremos a recorrência

$$T(n) = 2T(\left\lfloor \frac{n}{2} \right\rfloor) + n - 1, \quad T(1) = 1$$

Recorrências - Exemplo

Consideremos a recorrência

$$T(n) = 2T(\left\lfloor \frac{n}{2} \right\rfloor) + n - 1, \quad T(1) = 1$$

Podemos conjecturar que $T(n) \leq n^2$ e tentar provar por indução em n.

Recorrências - Exemplo

Consideremos a recorrência

$$T(n) = 2T(\left\lfloor \frac{n}{2} \right\rfloor) + n - 1, \quad T(1) = 1$$

Podemos conjecturar que $T(n) \leq n^2$ e tentar provar por indução em n.

- $T(1) = 1 \leq 1^2$

Recorrências - Exemplo

Consideremos a recorrência

$$T(n) = 2T(\left\lfloor \frac{n}{2} \right\rfloor) + n - 1, \quad T(1) = 1$$

Podemos conjecturar que $T(n) \leq n^2$ e tentar provar por indução em n.

- $T(1) = 1 \leq 1^2$
- $T(2) = 2T(1) + 1 = 3 < 2^2$

Recorrências - Exemplo

- Para $n \geq 3$, temos

Recorrências - Exemplo

- Para $n \geq 3$, temos

$$
\begin{aligned}
T(n) &= 2T(\left\lfloor \frac{n}{2} \right\rfloor) + n - 1 \\
&\leq 2 \left\lfloor \frac{n}{2} \right\rfloor^2 + n - 1 \\
&\leq 2 \left(\frac{n}{2} \right)^2 + n - 1 \\
&\leq \frac{n^2}{2} + n - 1 \\
&\leq \frac{n^2}{2} + \frac{n^2}{2} = n^2, \quad \forall n > 0
\end{aligned}
$$

Recorrências - Exemplo

Será n^2 uma boa estimativa para $T(n)$?

Recorrências - Exemplo

Será n^2 uma boa estimativa para $T(n)$?

Será que conseguimos provar que $T(n) \leq cn$, para alguma constante $c > 0$ e $n \geq n_0$?

Recorrências - Exemplo

Será n^2 uma boa estimativa para $T(n)$?

Será que conseguimos provar que $T(n) \leq cn$, para alguma constante $c > 0$ e $n \geq n_0$? Vamos tentar.

Recorrências - Exemplo

$$
\begin{aligned}
T(n) &= 2T(\left\lfloor \frac{n}{2} \right\rfloor) + n - 1 \\
&\leq 2c \left\lfloor \frac{n}{2} \right\rfloor + n - 1 \\
&\leq 2c \left(\frac{n}{2} \right) + n - 1 \\
&\leq cn + n - 1 \not\leq cn
\end{aligned}
$$

Para qualquer $c > 0$.

Recorrências - Exemplo

$$
\begin{aligned}
T(n) &= 2T(\left\lfloor \frac{n}{2} \right\rfloor) + n - 1 \\
&\leq 2c \left\lfloor \frac{n}{2} \right\rfloor + n - 1 \\
&\leq 2c \left(\frac{n}{2} \right) + n - 1 \\
&\leq cn + n - 1 \not\leq cn
\end{aligned}
$$

Para qualquer $c > 0$. Logo,

$$T(n) \not\leq cn$$

Recorrências - Exemplo

E que tal $T(n) \leq n \log_2 n + 1$?

Recorrências - Exemplo

E que tal $T(n) \leq n \log_2 n + 1$?

- $T(1) = 1 = 1 \log 1 + 1$

Recorrências - Exemplo

E que tal $T(n) \leq n \log_2 n + 1$?

- $T(1) = 1 = 1 \log 1 + 1$
- $T(2) = 2T(1) + 1 = 3 = 2 \log 2 + 1$

Recorrências - Exemplo

E que tal $T(n) \leq n \log_2 n + 1$?

- $T(1) = 1 = 1 \log 1 + 1$
- $T(2) = 2T(1) + 1 = 3 = 2 \log 2 + 1$

Vamos então tentar mostrar que

$$T(n) \leq n \log_2 n + 1$$

para $n \geq 1$.

Recorrências - Exemplo

$$
\begin{aligned}
T(n) &= 2T(\left\lfloor \frac{n}{2} \right\rfloor) + n - 1 \\
&\leq 2(\left\lfloor \frac{n}{2} \right\rfloor \log \left\lfloor \frac{n}{2} \right\rfloor + 1) + n - 1 \\
&\leq 2(\left(\frac{n}{2}\right) \log \left(\frac{n}{2}\right) + 1) + n - 1 \\
&= n(\log n - 1) + n + 1 = n \log n + 1
\end{aligned}
$$

Recorrências - Exemplo

$$
\begin{aligned}
T(n) &= 2T(\left\lfloor \frac{n}{2} \right\rfloor) + n - 1 \\
&\leq 2(\left\lfloor \frac{n}{2} \right\rfloor \log \left\lfloor \frac{n}{2} \right\rfloor + 1) + n - 1 \\
&\leq 2(\left(\frac{n}{2}\right) \log \left(\frac{n}{2}\right) + 1) + n - 1 \\
&= n(\log n - 1) + n + 1 = n \log n + 1
\end{aligned}
$$

Portanto,

$$T(n) \leq n \log n + 1$$

para $n \geq 1$.

Recorrências

Os exemplos anteriores mostram que a solução de uma recorrência pode ser obtida:

1. Desenvolvendo (ou “desenrolando”) a recorrência.
2. Estimando a solução e provando o resultado por indução.

Estas duas formas conjuntas recebem o nome de método da substituição.

Recorrências - Exemplo

Consideremos a recorrência

$$T(n) = T(\left\lfloor \frac{n}{2} \right\rfloor) + T(\left\lceil \frac{n}{2} \right\rceil) + 1, \quad T(1) = 1$$

Recorrências - Exemplo

Consideremos a recorrência

$$T(n) = T(\left\lfloor \frac{n}{2} \right\rfloor) + T(\left\lceil \frac{n}{2} \right\rceil) + 1, \quad T(1) = 1$$

Podemos conjecturar que $T(n) \leq cn$ e tentar provar por indução em n.

Recorrências - Exemplo

Consideremos a recorrência

$$T(n) = T(\left\lfloor \frac{n}{2} \right\rfloor) + T(\left\lceil \frac{n}{2} \right\rceil) + 1, \quad T(1) = 1$$

Podemos conjecturar que $T(n) \leq cn$ e tentar provar por indução em n. É comum partimos logo para analisar o caso geral, pois a base geralmente pode ser acertada.

Recorrências - Exemplo

$$\begin{aligned} T(n) &= T(\left\lfloor \frac{n}{2} \right\rfloor) + T(\left\lceil \frac{n}{2} \right\rceil) + 1 \\ &\leq c \left\lfloor \frac{n}{2} \right\rfloor + c \left\lceil \frac{n}{2} \right\rceil + 1 \\ &= cn + 1 \end{aligned}$$

O que não implica que $T(n) \leq cn$.

Recorrências - Exemplo

$$\begin{aligned} T(n) &= T(\left\lfloor \frac{n}{2} \right\rfloor) + T(\left\lceil \frac{n}{2} \right\rceil) + 1 \\ &\leq c \left\lfloor \frac{n}{2} \right\rfloor + c \left\lceil \frac{n}{2} \right\rceil + 1 \\ &= cn + 1 \end{aligned}$$

O que não implica que $T(n) \leq cn$.

Importante: precisamos provar a forma exata da hipótese indutiva.

Recorrências - Exemplo

Uma forma de contornar este problema é subtrair um termo de ordem menor. Por exemplo, tomemos agora $T(n) \leq cn - 1$.

Recorrências - Exemplo

Uma forma de contornar este problema é subtrair um termo de ordem menor. Por exemplo, tomemos agora $T(n) \leq cn - 1$.

$$\begin{aligned} T(n) &= T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + 1 \\ &\leq (c \lfloor \frac{n}{2} \rfloor - 1) + (c \lceil \frac{n}{2} \rceil - 1) + 1 \\ &\leq cn - 1 \end{aligned}$$

para qualquer c positivo.

Recorrências - Exemplo

Uma forma de contornar este problema é subtrair um termo de ordem menor. Por exemplo, tomemos agora $T(n) \leq cn - 1$.

$$\begin{aligned} T(n) &= T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + 1 \\ &\leq (c \lfloor \frac{n}{2} \rfloor - 1) + (c \lceil \frac{n}{2} \rceil - 1) + 1 \\ &\leq cn - 1 \end{aligned}$$

para qualquer c positivo. Tomando $c = 2$, temos $T(1) = 1 \leq 2(1) - 1$. Logo, $T(n) \leq 2n - 1$.

Recorrências - Exercícios

1. Resolva a recorrência $T(n) = T(\lfloor \frac{n}{2} \rfloor) + 7n + 2$, $T(1) = 1$.
2. Mostre que a solução de $T(n) = T(n-1) + n$ é $O(n^2)$.
3. Mostre que a solução de $T(n) = T(\lceil \frac{n}{2} \rceil) + 1$ é $O(\log n)$.
4. Mostre que a solução de $T(n) = 2T(\lfloor \frac{n}{2} \rfloor + 17) + n$ é $O(n \log n)$.
5. Resolva a recorrência $T(n) = 3T(\lfloor \frac{n}{2} \rfloor) + n, \quad T(1) = 1$.
6. Resolva a recorrência $T(n) = 3T(\lfloor \frac{n}{3} \rfloor) + 1, \quad T(1) = 1$.
7. Mostre que a solução de $T(n) = 3T\left(\frac{n}{2}\right) + n$ é $O(n^{\log_2 3})$.
8. Mostre que a solução de $T(n) = 4T\left(\frac{n}{2}\right) + n$ é $O(n^2)$.

Análise de Algoritmos

Identidades úteis

Identidades úteis

Apresentamos a seguir algumas igualdades e desigualdades que são úteis na análise algoritmos.

Identidades úteis

Apresentamos a seguir algumas igualdades e desigualdades que são úteis na análise algoritmos.

Série aritmética: Se $a_i = a_{i-1} + c$, onde c é uma constante, então

$$a_1 + a_2 + a_3 + \cdots + a_n = \frac{n(a_n + a_1)}{2}$$

Série geométrica

Se $a_i = ca_{i-1}$, onde $c \neq 1$ é uma constante, então

$$a_1 + a_2 + a_3 + \cdots + a_n = \frac{a_1(c_n - 1)}{c - 1}$$

Se $0 < c < 1$, então

$$\sum_{i=1}^{\infty} a_i = \frac{a_1}{1 - c}$$

Soma de quadrados

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Logaritmos

$$\log_b a = \frac{\log_c a}{\log_c b}$$

$$a^{\log_a x} = x$$

Aproximação de Stirling

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n (1 + O(\frac{1}{n}))$$

Em particular, a aproximação de Stirling implica que

$$\log(n!) = \Theta(n \log n)$$

Análise de Algoritmos

Árvore de recursão

Árvore de recursão

Árvore de recursão é um método muito útil para estimar a solução de uma recorrência. Uma vez obtida a solução, utilizamos o método anterior para prová-la formalmente.

A idéia da árvore de recursão é tentar representar o desenvolvimento da recorrência por um diagrama (geralmente uma árvore) onde cada nó representa um subproblema, e somar os custos por nível.

Exemplo 1

Consideremos a recorrência $T(n) = 2T\left(\lfloor \frac{n}{2} \rfloor\right) + n$

Exemplo 2

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

Exemplo 3

$$T(n) = 3T\left(\lfloor \tfrac{n}{2} \rfloor\right) + n^2$$

Exercícios

Use árvore de recursão para determinar um limite superior para as recorrências abaixo, e depois prove o resultado usando indução ou substituição.

1. $T(n) = T(n/4) + T(n/2) + n^2$
2. $T(n) = 3T\left(\lfloor \frac{n}{4} \rfloor\right) + n$
3. $T(n) = 4T\left(\frac{n}{2} + 2\right) + n$
4. $T(n) = T(n-a) + T(a) + cn$

Análise de Algoritmos

Método mestre

Método mestre

O método mestre fornece uma receita para a solução de recorrências da forma

$$T(n) = aT(n/b) + f(n)$$

onde $a \geq 1$ e $b > 1$ são constantes e $f(n)$ é uma função assintóticamente positiva.

Método mestre

Considere a recorrência

$$T(n) = aT(n/b) + f(n)$$

onde $a \geq 1$ e $b > 1$ são constantes, $f(n)$ é uma função assintóticamente positiva, e n/b pode ser $\lfloor n/b \rfloor$ ou $\lceil n/b \rceil$. Então

1. Se $f(n) = O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$ então $T(n) = \Theta(n^{\log_b a})$

Método mestre

Considere a recorrência

$$T(n) = aT(n/b) + f(n)$$

onde $a \geq 1$ e $b > 1$ são constantes, $f(n)$ é uma função assintóticamente positiva, e n/b pode ser $\lfloor n/b \rfloor$ ou $\lceil n/b \rceil$. Então

1. Se $f(n) = O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$ então $T(n) = \Theta(n^{\log_b a})$
2. Se $f(n) = \Theta(n^{\log_b a})$ então $T(n) = \Theta(n^{\log_b a} \log n)$

Método mestre

3. Se $f(n) = \Omega(n^{\log_b a+\epsilon})$ para alguma constante $\epsilon > 0$, e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$ e n suficientemente grande, então $T(n) = \Theta(f(n))$

Método mestre

$$T(n) = aT(n/b) + f(n)$$

$T(n)$	Se
$\Theta(n^{\log_b a})$	$f(n) = O(n^{\log_b a - \epsilon})$
$\Theta(n^{\log_b a} \log n)$	$f(n) = \Theta(n^{\log_b a})$
$\Theta(f(n))$	$f(n) = \Omega(n^{\log_b a + \epsilon})$ e $af(n/b) \leq cf(n)$, $c < 1$

Método mestre - Exemplo 1

$$T(n) = 9T(n/3) + n$$

Método mestre - Exemplo 1

$$T(n) = 9T(n/3) + n$$

- $a = 9$, $b = 3$ e $f(n) = n$

Método mestre - Exemplo 1

$$T(n) = 9T(n/3) + n$$

- $a = 9$, $b = 3$ e $f(n) = n$
- $n^{\log_b a} = n^{\log_3 9} = n^2$

Método mestre - Exemplo 1

$$T(n) = 9T(n/3) + n$$

- $a = 9$, $b = 3$ e $f(n) = n$
- $n^{\log_b a} = n^{\log_3 9} = n^2$
- Logo, $f(n) = O(n^{\log_b a - \epsilon})$, onde $\epsilon = 1$

Método mestre - Exemplo 1

$$T(n) = 9T(n/3) + n$$

- $a = 9$, $b = 3$ e $f(n) = n$
- $n^{\log_b a} = n^{\log_3 9} = n^2$
- Logo, $f(n) = O(n^{\log_b a - \epsilon})$, onde $\epsilon = 1$
- Pelo Caso 1, $T(n) = \Theta(n^2)$.

Método mestre - Exemplo 2

$$T(n) = T(2n/3) + 1$$

Método mestre - Exemplo 2

$$T(n) = T(2n/3) + 1$$

- $a = 1$, $b = 3/2$ e $f(n) = 1$

Método mestre - Exemplo 2

$$T(n) = T(2n/3) + 1$$

- $a = 1$, $b = 3/2$ e $f(n) = 1$
- $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$

Método mestre - Exemplo 2

$$T(n) = T(2n/3) + 1$$

- $a = 1$, $b = 3/2$ e $f(n) = 1$
- $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$
- Logo, $f(n) = 1 = \Theta(n^{\log_b a})$

Método mestre - Exemplo 2

$$T(n) = T(2n/3) + 1$$

- $a = 1$, $b = 3/2$ e $f(n) = 1$
- $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$
- Logo, $f(n) = 1 = \Theta(n^{\log_b a})$
- Pelo Caso 2, $T(n) = \Theta(\log n)$.

Método mestre - Exemplo 3

$$T(n) = 3T(n/4) + n \log n$$

Método mestre - Exemplo 3

$$T(n) = 3T(n/4) + n \log n$$

- $a = 3$, $b = 4$ e $f(n) = n \log n$

Método mestre - Exemplo 3

$$T(n) = 3T(n/4) + n \log n$$

- $a = 3$, $b = 4$ e $f(n) = n \log n$
- $n^{\log_b a} = n^{\log_4 3}$

Método mestre - Exemplo 3

$$T(n) = 3T(n/4) + n \log n$$

- $a = 3$, $b = 4$ e $f(n) = n \log n$
- $n^{\log_b a} = n^{\log_4 3}$
- Logo, $f(n) = n \log n = \Omega(n^{\log_4 3 + \epsilon})$

Método mestre - Exemplo 3

$$T(n) = 3T(n/4) + n \log n$$

- $a = 3$, $b = 4$ e $f(n) = n \log n$
- $n^{\log_b a} = n^{\log_4 3}$
- Logo, $f(n) = n \log n = \Omega(n^{\log_4 3 + \epsilon})$
- $af(n/b) = 3(n/4) \log(n/4) \leq (3/4) n \log n$

Método mestre - Exemplo 3

$$T(n) = 3T(n/4) + n \log n$$

- $a = 3$, $b = 4$ e $f(n) = n \log n$
- $n^{\log_b a} = n^{\log_4 3}$
- Logo, $f(n) = n \log n = \Omega(n^{\log_4 3 + \epsilon})$
- $af(n/b) = 3(n/4) \log(n/4) \leq (3/4) n \log n$
- Pelo Caso 3, $T(n) = \Theta(n \log n)$.

Método mestre - Exemplo 4

$$T(n) = 2T(n/2) + n \log n$$

Método mestre - Exemplo 4

$$T(n) = 2T(n/2) + n \log n$$

- $a = 2$, $b = 2$ e $f(n) = n \log n$

Método mestre - Exemplo 4

$$T(n) = 2T(n/2) + n \log n$$

- $a = 2$, $b = 2$ e $f(n) = n \log n$
- $n^{\log_b a} = n^{\log_2 2} = n$

Método mestre - Exemplo 4

$$T(n) = 2T(n/2) + n \log n$$

- $a = 2$, $b = 2$ e $f(n) = n \log n$
- $n^{\log_b a} = n^{\log_2 2} = n$
- Caso 3

Método mestre - Exemplo 4

$$T(n) = 2T(n/2) + n \log n$$

- $a = 2$, $b = 2$ e $f(n) = n \log n$
- $n^{\log_b a} = n^{\log_2 2} = n$
- Caso 3
- Mas, $f(n) = n \log n \neq \Omega(n^{\log_2 2 + \epsilon})$

Método mestre - Exemplo 4

$$T(n) = 2T(n/2) + n \log n$$

- $a = 2$, $b = 2$ e $f(n) = n \log n$
- $n^{\log_b a} = n^{\log_2 2} = n$
- Caso 3
- Mas, $f(n) = n \log n \neq \Omega(n^{\log_2 2+\epsilon})$
- Portanto, o método mestre não se aplica.

Método mestre - Exemplo 5

$$T(n) = 2T(n/2) + cn$$

Método mestre - Exemplo 5

$$T(n) = 2T(n/2) + cn$$

- $a = 2$, $b = 2$ e $f(n) = cn$

Método mestre - Exemplo 5

$$T(n) = 2T(n/2) + cn$$

- $a = 2$, $b = 2$ e $f(n) = cn$
- $n^{\log_b a} = n^{\log_2 2} = n$

Método mestre - Exemplo 5

$$T(n) = 2T(n/2) + cn$$

- $a = 2$, $b = 2$ e $f(n) = cn$
- $n^{\log_b a} = n^{\log_2 2} = n$
- Logo, $f(n) = cn = \Theta(n) = \Theta(n^{\log_b a})$

Método mestre - Exemplo 5

$$T(n) = 2T(n/2) + cn$$

- $a = 2$, $b = 2$ e $f(n) = cn$
- $n^{\log_b a} = n^{\log_2 2} = n$
- Logo, $f(n) = cn = \Theta(n) = \Theta(n^{\log_b a})$
- Pelo Caso 2, $T(n) = \Theta(n \log n)$.

Método mestre - Exemplo 6

$$T(n) = 8T(n/2) + cn^2$$

Método mestre - Exemplo 6

$$T(n) = 8T(n/2) + cn^2$$

- $a = 8$, $b = 2$ e $f(n) = cn^2$

Método mestre - Exemplo 6

$$T(n) = 8T(n/2) + cn^2$$

- $a = 8$, $b = 2$ e $f(n) = cn^2$
- $n^{\log_b a} = n^{\log_2 8} = n^3$

Método mestre - Exemplo 6

$$T(n) = 8T(n/2) + cn^2$$

- $a = 8$, $b = 2$ e $f(n) = cn^2$
- $n^{\log_b a} = n^{\log_2 8} = n^3$
- Logo, $f(n) = cn^2 = O(n^{\log_b a - \epsilon})$, onde $\epsilon = 1$

Método mestre - Exemplo 6

$$T(n) = 8T(n/2) + cn^2$$

- $a = 8$, $b = 2$ e $f(n) = cn^2$
- $n^{\log_b a} = n^{\log_2 8} = n^3$
- Logo, $f(n) = cn^2 = O(n^{\log_b a - \epsilon})$, onde $\epsilon = 1$
- Pelo Caso 1, $T(n) = \Theta(n^3)$.

Método mestre - Exemplo 7

$$T(n) = 7T(n/2) + cn^2$$

Método mestre - Exemplo 7

$$T(n) = 7T(n/2) + cn^2$$

- $a = 7$, $b = 2$ e $f(n) = cn^2$

Método mestre - Exemplo 7

$$T(n) = 7T(n/2) + cn^2$$

- $a = 7$, $b = 2$ e $f(n) = cn^2$
- $n^{\log_b a} = n^{\log_2 7}$

Método mestre - Exemplo 7

$$T(n) = 7T(n/2) + cn^2$$

- $a = 7$, $b = 2$ e $f(n) = cn^2$
- $n^{\log_b a} = n^{\log_2 7}$
- Logo, $f(n) = cn^2 = O(n^{\log_b a - \epsilon})$, onde $\epsilon > 0$

Método mestre - Exemplo 7

$$T(n) = 7T(n/2) + cn^2$$

- $a = 7$, $b = 2$ e $f(n) = cn^2$
- $n^{\log_b a} = n^{\log_2 7}$
- Logo, $f(n) = cn^2 = O(n^{\log_b a - \epsilon})$, onde $\epsilon > 0$
- Pelo Caso 1, $T(n) = \Theta(n^{\log_2 7})$.

Método mestre - Exercícios

1. $T(n) = 2T(n/4) + 1$
2. $T(n) = 2T(n/4) + \sqrt{n}$
3. $T(n) = 2T(n/4) + n$
4. $T(n) = 2T(n/4) + n^2$
5. $T(n) = T(n/2) + 1$
6. $T(n) = 4T(n/2) + n$
7. $T(n) = 4T(n/2) + n^2$
8. $T(n) = 4T(n/2) + n^3$