



RESOLVENDO RECORRÊNCIAS: ÁRVORE DE RECORRÊNCIA

Prof. André Vignatti

ÁRVORE DE RECORRÊNCIA

Método da Árvore de Recorrência:

- Permite visualizar melhor o que acontece quando a recorrência é iterada
- É mais fácil organizar as contas
- Útil para recorrências de algoritmos de divisão-e-conquista

Considere a recorrência

$$\begin{aligned} T(n) &= \Theta(1) && \text{para } n = 1, 2, 3, \\ T(n) &= 3T(\lfloor n/4 \rfloor) + cn^2 && \text{para } n \geq 4, \end{aligned}$$

onde $c > 0$ é uma constante.

Considere a recorrência

$$\begin{aligned} T(n) &= \Theta(1) && \text{para } n = 1, 2, 3, \\ T(n) &= 3T(\lfloor n/4 \rfloor) + cn^2 && \text{para } n \geq 4, \end{aligned}$$

onde $c > 0$ é uma constante.

Simplificação: supor recorrência definida apenas para potências de 4

$$\begin{aligned} T(n) &= \Theta(1) && \text{para } n = 1, \\ T(n) &= 3T(n/4) + cn^2 && \text{para } n = 4, 16, \dots, 4^i, \dots \end{aligned}$$

Considere a recorrência

$$\begin{aligned} T(n) &= \Theta(1) && \text{para } n = 1, 2, 3, \\ T(n) &= 3T(\lfloor n/4 \rfloor) + cn^2 && \text{para } n \geq 4, \end{aligned}$$

onde $c > 0$ é uma constante.

Simplificação: supor recorrência definida apenas para potências de 4

$$\begin{aligned} T(n) &= \Theta(1) && \text{para } n = 1, \\ T(n) &= 3T(n/4) + cn^2 && \text{para } n = 4, 16, \dots, 4^i, \dots \end{aligned}$$

A simplificação facilita a solução.

Considere a recorrência

$$\begin{aligned} T(n) &= \Theta(1) && \text{para } n = 1, 2, 3, \\ T(n) &= 3T(\lfloor n/4 \rfloor) + cn^2 && \text{para } n \geq 4, \end{aligned}$$

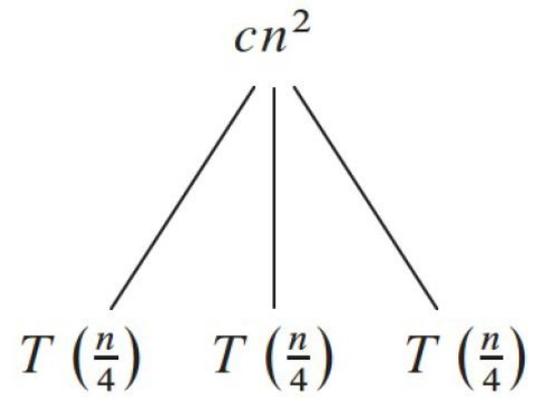
onde $c > 0$ é uma constante.

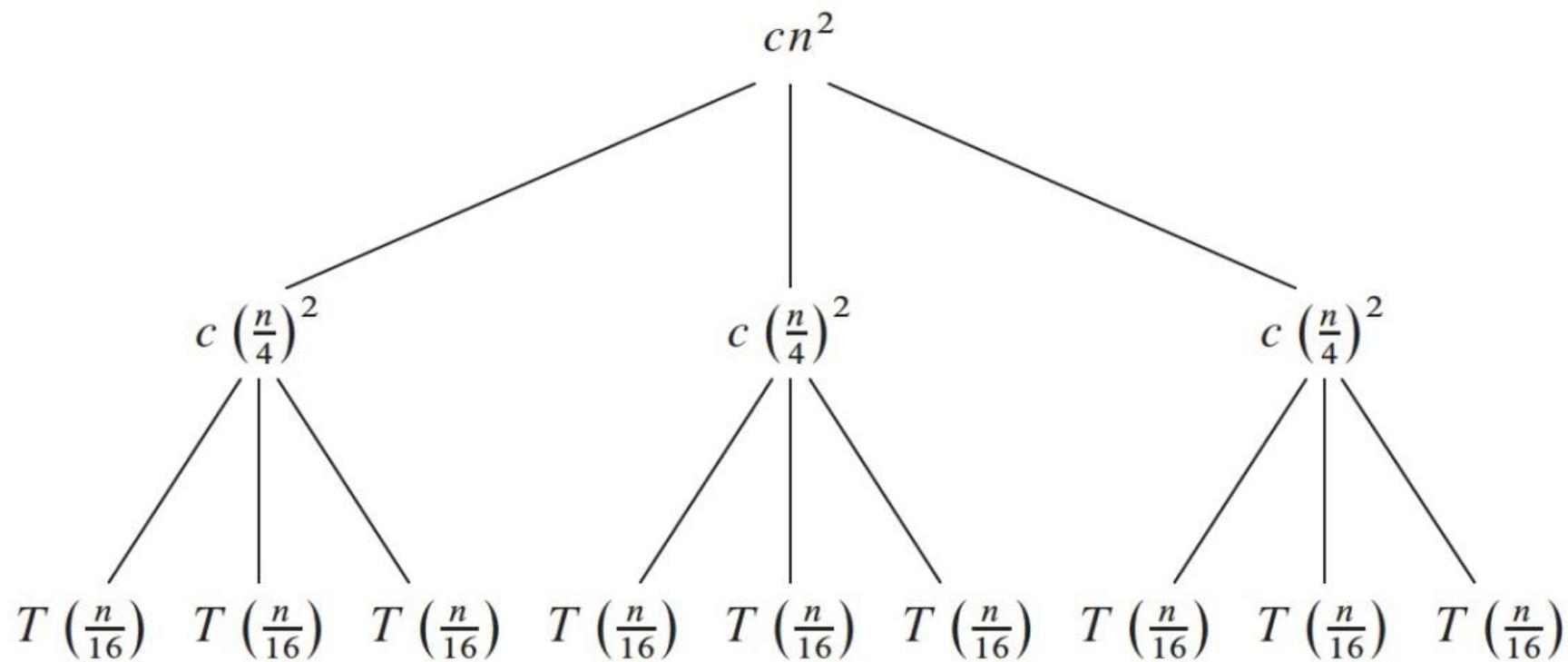
Simplificação: supor recorrência definida apenas para potências de 4

$$\begin{aligned} T(n) &= \Theta(1) && \text{para } n = 1, \\ T(n) &= 3T(n/4) + cn^2 && \text{para } n = 4, 16, \dots, 4^i, \dots \end{aligned}$$

A simplificação facilita a solução.

Depois usamos o método da substituição para formalizar.





- O número de níveis é $\log_4 n + 1$

- O número de níveis é $\log_4 n + 1$
- No nível i o tempo gasto (sem contar as chamadas recursivas) é $(3/16)^i cn^2$

- O número de níveis é $\log_4 n + 1$
- No nível i o tempo gasto (sem contar as chamadas recursivas) é $(3/16)^i cn^2$
- No último nível há $3^{\log_4 n} = n^{\log_4 3}$ folhas. Como $T(1) = \Theta(1)$ o tempo gasto é $\Theta(n^{\log_4 3})$.

- O número de níveis é $\log_4 n + 1$
- No nível i o tempo gasto (sem contar as chamadas recursivas) é $(3/16)^i cn^2$
- No último nível há $3^{\log_4 n} = n^{\log_4 3}$ folhas. Como $T(1) = \Theta(1)$ o tempo gasto é $\Theta(n^{\log_4 3})$.

Logo,

$$T(n) = cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3})$$

- O número de níveis é $\log_4 n + 1$
- No nível i o tempo gasto (sem contar as chamadas recursivas) é $(3/16)^i cn^2$
- No último nível há $3^{\log_4 n} = n^{\log_4 3}$ folhas. Como $T(1) = \Theta(1)$ o tempo gasto é $\Theta(n^{\log_4 3})$.

Logo,

$$\begin{aligned}
 T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3}) \\
 &= cn^2 \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i + \Theta(n^{\log_4 3})
 \end{aligned}$$

- O número de níveis é $\log_4 n + 1$
- No nível i o tempo gasto (sem contar as chamadas recursivas) é $(3/16)^i cn^2$
- No último nível há $3^{\log_4 n} = n^{\log_4 3}$ folhas. Como $T(1) = \Theta(1)$ o tempo gasto é $\Theta(n^{\log_4 3})$.

Logo,

$$\begin{aligned}
 T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3}) \\
 &= cn^2 \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i + \Theta(n^{\log_4 3}) \\
 &\leq cn^2 \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i + \Theta(n^{\log_4 3})
 \end{aligned}$$

- O número de níveis é $\log_4 n + 1$
- No nível i o tempo gasto (sem contar as chamadas recursivas) é $(3/16)^i cn^2$
- No último nível há $3^{\log_4 n} = n^{\log_4 3}$ folhas. Como $T(1) = \Theta(1)$ o tempo gasto é $\Theta(n^{\log_4 3})$.

Logo,

$$\begin{aligned}
 T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3}) \\
 &= cn^2 \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i + \Theta(n^{\log_4 3}) \\
 &\leq cn^2 \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i + \Theta(n^{\log_4 3}) \\
 &= \frac{16}{13}cn^2 + \Theta(n^{\log_4 3}),
 \end{aligned}$$

- O número de níveis é $\log_4 n + 1$
- No nível i o tempo gasto (sem contar as chamadas recursivas) é $(3/16)^i cn^2$
- No último nível há $3^{\log_4 n} = n^{\log_4 3}$ folhas. Como $T(1) = \Theta(1)$ o tempo gasto é $\Theta(n^{\log_4 3})$.

Logo,

$$\begin{aligned}
 T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3}) \\
 &= cn^2 \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i + \Theta(n^{\log_4 3}) \\
 &\leq cn^2 \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i + \Theta(n^{\log_4 3}) \\
 &= \frac{16}{13}cn^2 + \Theta(n^{\log_4 3}),
 \end{aligned}$$

e $T(n) \in O(n^2)$.

PROVANDO O RESULTADO OBTIDO

Mas $T(n) \in O(n^2)$ é realmente a solução da recorrência original (sem simplificação)?

PROVANDO O RESULTADO OBTIDO

Mas $T(n) \in O(n^2)$ é realmente a solução da recorrência original (sem simplificação)?

Pela árvore de recorrência, chutamos que $T(n) \leq dn^2$ para alguma constante $d > 0$

PROVANDO O RESULTADO OBTIDO

Mas $T(n) \in O(n^2)$ é realmente a solução da recorrência original (sem simplificação)?

Pela árvore de recorrência, chutamos que $T(n) \leq dn^2$ para alguma constante $d > 0$

$$\begin{aligned} T(n) &= 3T(\lfloor n/4 \rfloor) + cn^2 \\ &\leq 3d\lfloor n/4 \rfloor^2 + cn^2 \\ &\leq 3d(n/4)^2 + cn^2 \\ &= \frac{3}{16}dn^2 + cn^2 \\ &\leq dn^2 \end{aligned}$$

onde a última desigualdade vale se $d \geq (16/13)c$ (**Yeeesss!!**).

RESUMO

- O número de nós em cada nível da árvore é o número de chamadas recursivas.

RESUMO

- O número de nós em cada nível da árvore é o número de chamadas recursivas.
- Em cada nó indicamos o “tempo” ou “trabalho” gasto naquele nó que **não** corresponde a chamadas recursivas.

RESUMO

- O número de nós em cada nível da árvore é o número de chamadas recursivas.
- Em cada nó indicamos o “tempo” ou “trabalho” gasto naquele nó que **não** corresponde a chamadas recursivas.
- Na coluna mais à direita indicamos o **tempo total** naquele nível que **não** corresponde a chamadas recursivas.

RESUMO

- O número de nós em cada nível da árvore é o número de chamadas recursivas.
- Em cada nó indicamos o “tempo” ou “trabalho” gasto naquele nó que **não** corresponde a chamadas recursivas.
- Na coluna mais à direita indicamos o **tempo total** naquele nível que **não** corresponde a chamadas recursivas.
- Somando ao longo da coluna determina-se a solução da recorrência.

RECORRÊNCIA COM NOTAÇÃO ASSINTÓTICA

Uma “recorrência”

$$\begin{aligned} T(n) &= \Theta(1) && \text{para } n = 1, 2, \\ T(n) &= 3T(\lfloor n/4 \rfloor) + \Theta(n^2) && \text{para } n \geq 3 \end{aligned}$$

representa todas as recorrências da forma

$$\begin{aligned} T(n) &= a && \text{para } n = 1, 2, \\ T(n) &= 3T(\lfloor n/4 \rfloor) + bn^2 && \text{para } n \geq 3 \end{aligned}$$

onde a e $b > 0$ são constantes.

$$\begin{aligned} T(n) &= a && \text{para } n = 1, 2, \\ T(n) &= 3T(\lfloor n/4 \rfloor) + bn^2 && \text{para } n \geq 3 \end{aligned}$$

Soluções exatas dependem dos valores de a e b

mas estão todas na mesma classe Θ

A “solução” é $T(n) = \Theta(n^2)$

As mesmas observações valem para as classes O, Ω, o, ω

RECORRÊNCIA DO MERGESORT

Podemos escrever a recorrência de tempo do Mergesort como

$$\begin{aligned} T(1) &= \Theta(1) \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) \quad \text{para } n \geq 2 \end{aligned}$$

A solução da recorrência é $T(n) = \Theta(n \lg n)$ (Exercício!)