

Projeto e Análise de Algoritmos*

Complexidade de algoritmos

Lehilton Pedrosa

Segundo Semestre de 2017

*Criado por C. de Souza, C. da Silva, O. Lee, F. Miyazawa et al.

Notação assintótica e crescimento de funções

Notação Assintótica

- ▶ Vamos expressar complexidade através de funções em variáveis que descrevam o tamanho de instâncias do problema. Exemplos:
 - ▶ Problemas de aritmética de precisão arbitrária: número de bits (ou bytes) dos inteiros.
 - ▶ Problemas em grafos: número de vértices e/ou arestas
 - ▶ Problemas de ordenação de vetores: tamanho do vetor.
 - ▶ Busca em textos: número de caracteres do texto ou padrão de busca.
- ▶ Vamos supor que funções que expressam complexidade são sempre positivas, já que estamos medindo número de operações.

Comparação de Funções

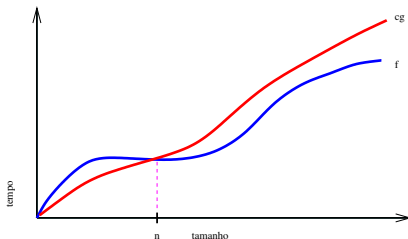
- ▶ Vamos comparar funções assintoticamente, ou seja, para valores grandes, desprezando constantes multiplicativas e termos de menor ordem.

	$n = 100$	$n = 1000$	$n = 10^4$	$n = 10^6$	$n = 10^9$
$\log n$	2	3	4	6	9
n	100	1000	10^4	10^6	10^9
$n \log n$	200	3000	$4 \cdot 10^4$	$6 \cdot 10^6$	$9 \cdot 10^9$
n^2	10^4	10^6	10^8	10^{12}	10^{18}
$100n^2 + 15n$	$1,0015 \cdot 10^6$	$1,00015 \cdot 10^8$	$\approx 10^{10}$	$\approx 10^{14}$	$\approx 10^{20}$
2^n	$\approx 1,26 \cdot 10^{30}$	$\approx 1,07 \cdot 10^{301}$?	?	?

Definição:

$O(g(n)) = \{f(n) : \text{existem constantes positivas } c \text{ e } n_0 \text{ tais que } 0 \leq f(n) \leq cg(n), \text{ para todo } n \geq n_0\}$.

Informalmente, dizemos que, se $f(n) \in O(g(n))$, então $f(n)$ cresce no máximo tão rapidamente quanto $g(n)$.



Definição:

$O(g(n)) = \{f(n) : \text{existem constantes positivas } c \text{ e } n_0 \text{ tais que } 0 \leq f(n) \leq cg(n), \text{ para todo } n \geq n_0\}.$

Informalmente, dizemos que, se $f(n) \in O(g(n))$, então $f(n)$ cresce no máximo tão rapidamente quanto $g(n)$.

Exemplo:

$$\frac{1}{2}n^2 - 3n \in O(n^2)$$

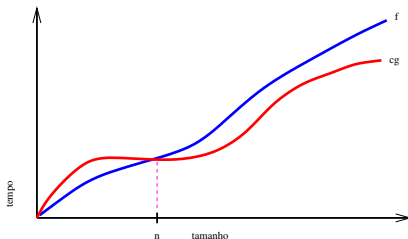
Valores de c e n_0 que satisfazem a definição são

$$c = \frac{1}{2} \text{ e } n_0 = 7.$$

Definição:

$\Omega(g(n)) = \{f(n) : \text{existem constantes positivas } c \text{ e } n_0 \text{ tais que } 0 \leq cg(n) \leq f(n), \text{ para todo } n \geq n_0\}$.

Informalmente, dizemos que, se $f(n) \in \Omega(g(n))$, então $f(n)$ cresce no mínimo tão lentamente quanto $g(n)$.



Definição:

$\Omega(g(n)) = \{f(n) : \text{existem constantes positivas } c \text{ e } n_0 \text{ tais que } 0 \leq cg(n) \leq f(n), \text{ para todo } n \geq n_0\}.$

Informalmente, dizemos que, se $f(n) \in \Omega(g(n))$, então $f(n)$ cresce no mínimo tão lentamente quanto $g(n)$.

Exemplo:

$$\frac{1}{2}n^2 - 3n \in \Omega(n^2)$$

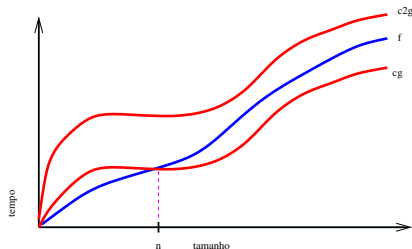
Valores de c e n_0 que satisfazem a definição são

$$c = \frac{1}{14} \text{ e } n_0 = 7.$$

Definição:

$\Theta(g(n)) = \{f(n) : \text{existem constantes positivas } c_1, c_2 \text{ e } n_0 \text{ tais que } 0 \leq c_1g(n) \leq f(n) \leq c_2g(n), \text{ para todo } n \geq n_0\}$.

Informalmente, dizemos que, se $f(n) \in \Theta(g(n))$, então $f(n)$ cresce tão rapidamente quanto $g(n)$.



Definição:

$\Theta(g(n)) = \{f(n) : \text{ existem constantes positivas } c_1, c_2 \text{ e } n_0$
tais que $0 \leq c_1g(n) \leq f(n) \leq c_2g(n)$,
para todo $n \geq n_0\}$.

Informalmente, dizemos que, se $f(n) \in \Theta(g(n))$, então $f(n)$ cresce tão rapidamente quanto $g(n)$.

Exemplo:

$$\frac{1}{2}n^2 - 3n \in \Theta(n^2)$$

Valores de c_1 , c_2 e n_0 que satisfazem a definição são

$$c_1 = \frac{1}{14}, c_2 = \frac{1}{2} \text{ e } n_0 = 7.$$

Definição:

$o(g(n)) = \{f(n) : \text{para toda constante positiva } c, \text{ existe uma constante } n_0 > 0 \text{ tal que } 0 \leq f(n) < cg(n), \text{ para todo } n \geq n_0\}.$

Informalmente, dizemos que, se $f(n) \in o(g(n))$, então $f(n)$ cresce mais lentamente que $g(n)$.

Exemplo:

$$1000n^2 \in o(n^3)$$

Para todo valor de c , um n_0 que satisfaz a definição é

$$n_0 = \left\lceil \frac{1000}{c} \right\rceil + 1.$$

Definição:

$\omega(g(n)) = \{f(n) : \text{para toda constante positiva } c, \text{ existe uma constante } n_0 > 0 \text{ tal que } 0 \leq cg(n) < f(n), \text{ para todo } n \geq n_0.\}$

Informalmente, dizemos que, se $f(n) \in \omega(g(n))$, então $f(n)$ cresce mais rapidamente que $g(n)$.

Exemplo:

$$\frac{1}{1000}n^2 \in \omega(n)$$

Para todo valor de c , um n_0 que satisfaz a definição é

$$n_0 = \lceil 1000c \rceil + 1.$$

Algumas condições necessárias:

$$f(n) \in o(g(n)) \text{ implica } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0.$$

$$f(n) \in O(g(n)) \text{ implica } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty.$$

$$f(n) \in \Theta(g(n)) \text{ implica } 0 < \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty.$$

$$f(n) \in \Omega(g(n)) \text{ implica } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} > 0.$$

$$f(n) \in \omega(g(n)) \text{ implica } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty.$$

Se também tivermos que $f(n), g(n) \geq 0$ para $n \geq n_0$, então basta calcular os limites!

Transitividade:

Se $f(n) \in O(g(n))$ e $g(n) \in O(h(n))$, então $f(n) \in O(h(n))$.

Se $f(n) \in \Omega(g(n))$ e $g(n) \in \Omega(h(n))$, então $f(n) \in \Omega(h(n))$.

Se $f(n) \in \Theta(g(n))$ e $g(n) \in \Theta(h(n))$, então $f(n) \in \Theta(h(n))$.

Se $f(n) \in o(g(n))$ e $g(n) \in o(h(n))$, então $f(n) \in o(h(n))$.

Se $f(n) \in \omega(g(n))$ e $g(n) \in \omega(h(n))$, então $f(n) \in \omega(h(n))$.

Reflexividade:

$$f(n) \in O(f(n)).$$

$$f(n) \in \Omega(f(n)).$$

$$f(n) \in \Theta(f(n)).$$

Simetria:

$$f(n) \in \Theta(g(n)) \text{ se, e somente se, } g(n) \in \Theta(f(n)).$$

Simetria Transposta:

$$f(n) \in O(g(n)) \text{ se, e somente se, } g(n) \in \Omega(f(n)).$$

$$f(n) \in o(g(n)) \text{ se, e somente se, } g(n) \in \omega(f(n)).$$

Teorema (Regra de l'Hôpital)

Sejam f e g funções diferenciáveis tal que $\lim_{n \rightarrow \infty} f(n) = \lim_{n \rightarrow \infty} g(n) = \infty$. Se o limite de $f(n)/g(n)$ existir então,

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{f'(n)}{g'(n)}.$$

Regra de l'Hôpital

Exercício:

Descreva as outras formas da Regra de l'Hôpital.

Exemplo:

Seja $f(n) = \ln n$ and $g(n) = \sqrt{n}$. Como $f'(n) = \frac{1}{n}$ e $g'(n) = \frac{1}{2\sqrt{n}}$, temos

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{f'(n)}{g'(n)} = \lim_{n \rightarrow \infty} \frac{2\sqrt{n}}{n} = \lim_{n \rightarrow \infty} \frac{2}{\sqrt{n}} = 0.$$

Portanto, $f(n) = o(g(n))$.

Quais as relações de comparação assintótica das funções:

- ▶ 2^π
- ▶ $\log n$
- ▶ n
- ▶ $n \log n$
- ▶ n^2
- ▶ $100n^2 + 15n$
- ▶ 2^n

Recorrências

Resolução de Recorrências

- ▶ Relações de recorrência expressam a complexidade de algoritmos recursivos como os algoritmos de divisão e conquista.
- ▶ É preciso saber resolver as recorrências para que possamos efetivamente determinar a complexidade dos algoritmos recursivos.

Mergesort

```
MERGE-SORT( $A, p, r$ )  
1  se  $p < r$   
2    então  $q \leftarrow \lfloor (p + r)/2 \rfloor$   
3        MERGE-SORT( $A, p, q$ )  
4        MERGE-SORT( $A, q + 1, r$ )  
5        INTERCALA( $A, p, q, r$ )
```

Qual é a complexidade de MERGE-SORT?

Seja $T(n) :=$ o consumo de tempo **máximo** (pior caso) em função de $n = r - p + 1$

Complexidade do Mergesort

```
MERGE-SORT( $A, p, r$ )  
1  se  $p < r$   
2    então  $q \leftarrow \lfloor (p + r)/2 \rfloor$   
3          MERGE-SORT( $A, p, q$ )  
4          MERGE-SORT( $A, q + 1, r$ )  
5          INTERCALA( $A, p, q, r$ )
```

linha	consumo de tempo
1	?
2	?
3	?
4	?
5	?

$T(n) = ?$

Complexidade do Mergesort

```
MERGE-SORT( $A, p, r$ )  
1  se  $p < r$   
2    então  $q \leftarrow \lfloor (p + r)/2 \rfloor$   
3        MERGE-SORT( $A, p, q$ )  
4        MERGE-SORT( $A, q + 1, r$ )  
5        INTERCALA( $A, p, q, r$ )
```

linha	consumo de tempo
1	b_0
2	b_1
3	$T(\lceil n/2 \rceil)$
4	$T(\lfloor n/2 \rfloor)$
5	an

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + an + (b_0 + b_1)$$

Resolução de recorrências

- ▶ Queremos resolver a recorrência

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

- ▶ Resolver uma recorrência significa encontrar uma **fórmula fechada** para $T(n)$.
- ▶ Não é necessário achar uma **solução exata**. Basta encontrar uma função $f(n)$ tal que $T(n) \in \Theta(f(n))$.

Resolução de recorrências

Alguns métodos para resolução de recorrências:

- ▶ substituição
- ▶ iteração
- ▶ árvore de recorrência

Veremos também um resultado bem geral que permite resolver várias recorrências diretamente: **Teorema Master**.

Método da substituição

- ▶ Idéia básica: “adivinha” qual é a solução e prove por indução que ela funciona!
- ▶ Método poderoso mas nem sempre aplicável (obviamente).
- ▶ Com prática e experiência fica mais fácil de usar!

Exemplo

Considere a recorrência:

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

Chuto que $T(n) \in O(n \lg n)$.

Mais precisamente, chuto que $T(n) \leq 3n \lg n$.

(Lembre que $\lg n = \log_2 n$.)

Exemplo

$$\begin{aligned}T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n \\&\leq 3 \left\lceil \frac{n}{2} \right\rceil \lg \left\lceil \frac{n}{2} \right\rceil + 3 \left\lfloor \frac{n}{2} \right\rfloor \lg \left\lfloor \frac{n}{2} \right\rfloor + n \\&\leq 3 \left\lceil \frac{n}{2} \right\rceil \lg n + 3 \left\lfloor \frac{n}{2} \right\rfloor (\lg n - 1) + n \\&= 3 \left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor \right) \lg n - 3 \left\lfloor \frac{n}{2} \right\rfloor + n \\&= 3n \lg n - 3 \left\lfloor \frac{n}{2} \right\rfloor + n \\&\leq 3n \lg n.\end{aligned}$$

(Yeeeeeeesssss!)

Exemplo

- ▶ Mas espere um pouco!
- ▶ $T(1) = 1$ e $3 \cdot 1 \cdot \lg 1 = 0$ e a base da indução não funciona!
- ▶ Certo, mas lembre-se da definição da classe $O()$.

Só preciso provar que $T(n) \leq 3n \lg n$ para $n \geq n_0$ onde n_0 é alguma constante.

Vamos tentar com $n_0 = 2$. Nesse caso

$$T(2) = T(1) + T(1) + 2 = 4 \leq 3 \cdot 2 \cdot \lg 2 = 6,$$

$$T(3) = T(2) + T(1) + 3 = 8 \leq 3 \cdot 3 \cdot \lg 3 \approx 14,26,$$

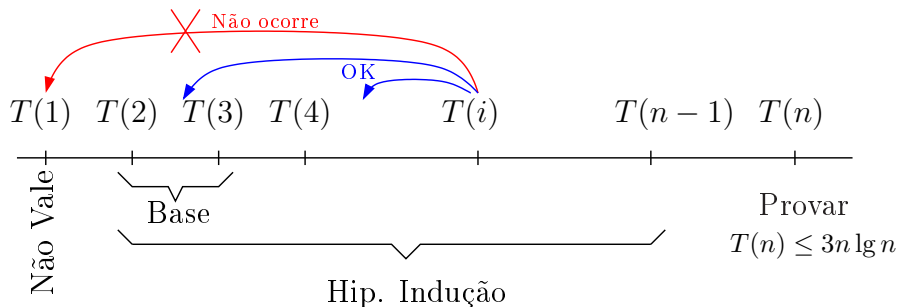
Assim, o chute vale para $n = 2$ e $n = 3$ (esta será a Base!)

Como a recorrência de $T(n)$, para $n > 3$, recai em recorrências menores até cair na base, estamos feitos.

Recorrência e Indução

$$T(1) = 1$$

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n$$



Exemplo

- ▶ Certo, funcionou quando $T(1) = 1$.
- ▶ Mas e se por exemplo $T(1) = 8$?

Então $T(2) = 8 + 8 + 2 = 18$ e $3 \cdot 2 \cdot \lg 2 = 6$.

Não deu certo...

- ▶ Certo, mas aí basta escolher uma **constante** maior. Mostra-se do mesmo jeito que $T(n) \leq 10n \lg n$, pois
 $T(2) = 18 \leq 10 \cdot 2 \cdot \lg 2 = 20$,
 $T(3) = T(1) + T(2) + 3 = 8 + 18 + 3 = 29 \leq 10 \cdot 3 \lg 3 \approx 47,55$.
- ▶ De modo geral, se o **passo de indução** funciona ($T(n) \leq cn \lg n$), é possível escolher c e a **base da indução** de modo conveniente!

Como achar as constantes?

- ▶ Tudo bem. Dá até para chutar que $T(n)$ pertence a classe $O(n \lg n)$.
- ▶ Mas como descobrir que $T(n) \leq 3n \lg n$? Como achar a constante 3?
- ▶ Eis um método simples: suponha como hipótese de indução que $T(n) \leq cn \lg n$ para $n \geq n_0$ onde c e n_0 são constantes que vou tentar determinar.

Primeira tentativa

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

(Hummm, não deu certo...)

Segunda tentativa

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

Para garantir a última desigualdade basta que $-c\lfloor n/2 \rfloor + n \leq 0$ e $c = 3$ funciona. (YeEEEEEESSSSS!)

Completando o exemplo

Mostramos que a recorrência

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

satisfaz $T(n) \in O(n \lg n)$.

Mas quem garante que $T(n)$ não é “menor”?

O melhor é mostrar que $T(n) \in \Theta(n \lg n)$.

Resta então mostrar que $T(n) \in \Omega(n \lg n)$. A prova é similar.
(Exercício!)

Como chutar?

Não há nenhuma receita genérica para adivinhar soluções de recorrências. A experiência é o fator mais importante.

Felizmente, há várias ideias que podem ajudar.

Considere a recorrência

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

Ela é quase idêntica à anterior e podemos chutar que $T(n) \in \Theta(n \lg n)$. Isto de fato é verdade. (**Exercício** ou consulte o CLRS)

Como chutar?

Considere agora a recorrência

$$\begin{aligned}T(1) &= 1 \\T(n) &= 2T(\lfloor n/2 \rfloor + 17) + n \quad \text{para } n \geq 2.\end{aligned}$$

Ela parece bem mais difícil por causa do “17” no lado direito.

Intuitivamente, porém, isto não deveria afetar a solução. Para **n grande** a diferença entre $T(\lfloor n/2 \rfloor)$ e $T(\lfloor n/2 \rfloor + 17)$ não é tanta.

Chuto então que $T(n) \in \Theta(n \lg n)$. (Exercício!)

Truques e sutilezas

Algumas vezes adivinhamos corretamente a solução de uma recorrência, mas as contas aparentemente não funcionam! Em geral, o que é necessário é fortalecer a **hipótese de indução**.

Considere a recorrência

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

Chutamos que $T(n) \in O(n)$ e tentamos mostrar que $T(n) \leq cn$ para alguma constante c .

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

(Humm, falhou...)

E agora? Será que erramos o chute? Será que $T(n) \in \Theta(n^2)$?

Truques e sutilezas

Na verdade, adivinhamos corretamente. Para provar isso, é preciso usar uma hipótese de indução mais forte.

Vamos mostrar que $T(n) \leq cn - b$ onde $b > 0$ é uma constante.

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

onde a última desigualdade vale se $b \geq 1$.
(Yeeeessss!)

Método da iteração

- ▶ Não é necessário adivinhar a resposta!
- ▶ Precisa fazer mais contas!
- ▶ Idéia: expandir (iterar) a recorrência e escrevê-la como uma somatória de termos que dependem apenas de n e das **condições iniciais**.
- ▶ Precisa conhecer limitantes para várias somatórias.

Método da iteração

Considere a recorrência

$$\begin{aligned}T(n) &= b && \text{para } n \leq 3, \\T(n) &= 3T(\lfloor n/4 \rfloor) + n && \text{para } n \geq 4.\end{aligned}$$

Iterando a recorrência obtemos

$$\begin{aligned}T(n) &= n + 3T(\lfloor n/4 \rfloor) \\&= n + 3(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor)) \\&= n + 3(\lfloor n/4 \rfloor + 3(\lfloor n/16 \rfloor + 3T(\lfloor n/64 \rfloor))) \\&= n + 3\lfloor n/4 \rfloor + 9\lfloor n/16 \rfloor + 27T(\lfloor n/64 \rfloor).\end{aligned}$$

Certo, mas quando devo parar?

O i -ésimo termo da série é $3^i \lfloor n/4^i \rfloor$. Ela termina quando $\lfloor n/4^i \rfloor \leq 3$, ou seja, $i \geq \log_4 n$.

Método da iteração

Como $\lfloor n/4^i \rfloor \leq n/4^i$ temos que

$$T(n) \leq n + 3n/4 + 9n/16 + 27n/64 + \dots + 3^j b$$

$$T(n) \leq n + 3n/4 + 9n/16 + 27n/64 + \dots + d \cdot 3^{\log_4 n}$$

$$\leq n \cdot (1 + 3/4 + 9/16 + 27/64 + \dots) + dn^{\log_4 3}$$

$$\leq n \sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i + dn^{\log_4 3}$$

$$= 4n + dn^{\log_4 3}$$

pois $3^{\log_4 n} = n^{\log_4 3}$ e $\sum_{i=0}^{\infty} q^i = \frac{1}{1-q}$ para $0 < q < 1$.

Como $\log_4 3 < 1$ segue que $n^{\log_4 3} \in o(n)$ e logo, $T(n) \in O(n)$.

Método de iteração

- ▶ As contas ficam mais simples se supormos que a recorrência está definida apenas para potências de um número, por exemplo, $n = 4^i$.
- ▶ Note, entretanto, que a recorrência deve ser provada para todo natural suficientemente grande.
- ▶ Muitas vezes, é possível depois de iterar a recorrência, **adivinhar** a solução e usar o método da substituição!

Método de iteração

$$\begin{aligned}T(n) &= b && \text{para } n \leq 3, \\T(n) &= 3T(\lfloor n/4 \rfloor) + n && \text{para } n \geq 4.\end{aligned}$$

Chuto que $T(n) \leq cn$.

$$\begin{aligned}T(n) &= 3T(\lfloor n/4 \rfloor) + n \\&\leq 3c\lfloor n/4 \rfloor + n \\&\leq 3c(n/4) + n \\&\leq cn\end{aligned}$$

onde a última desigualdade vale se $c \geq 4$.
(Yeesssss!)

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.

Árvore de recorrência

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.

Costuma-se (CLRS) usar a notação $T(n) = \Theta(1)$ para indicar que $T(n)$ é uma constante.

Árvore de recorrência

Simplificação

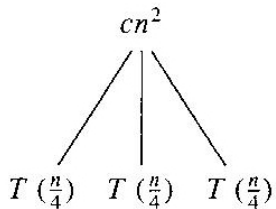
Vamos supor que a recorrência está definida apenas para potências de 4

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

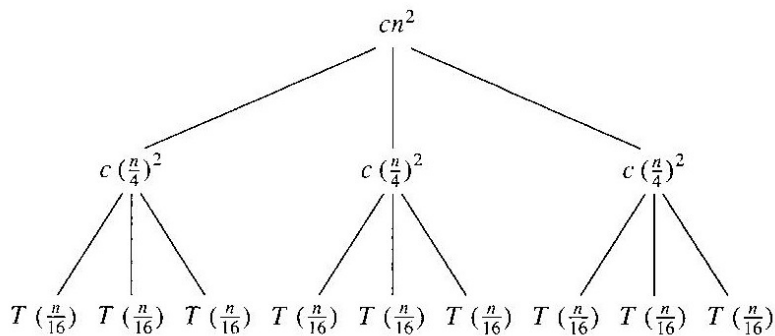
Isto permite descobrir mais facilmente a solução. Depois usamos o método da substituição para formalizar.

Árvore de recorrência

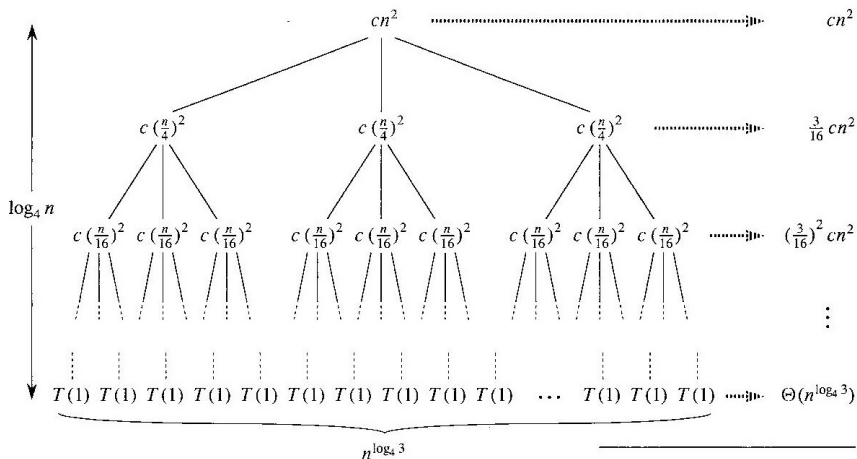
$T(n)$



Árvore de recorrência



Árvore de recorrência



Total: $O(n^2)$

Árvore de recorrência

- ▶ 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 $\Theta(n^{\log_4 3})$.

Árvore de recorrência

Logo,

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

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

Árvore de recorrência

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

Com base na 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 $d \geq (16/13)c$.
(Yeesssss!)

Árvore de recorrência

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.

Vamos tentar juntos?

Eis um exemplo um pouco mais complicado.

Vamos resolver a recorrência

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

Qual é a solução da recorrência?

Resposta: $T(n) \in O(n \lg n)$. (Resolvido em aula)

Recorrências com O à direita (CLRS)

Uma “recorrência”

$$T(n) = \Theta(1) \quad \text{para } n = 1, 2,$$

$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2) \quad \text{para } n \geq 3$$

representa todas as recorrências da forma

$$T(n) = a \quad \text{para } n = 1, 2,$$

$$T(n) = 3T(\lfloor n/4 \rfloor) + bn^2 \quad \text{para } n \geq 3$$

onde a e $b > 0$ são constantes.

As soluções exatas dependem dos valores de a e b , mas estão todas na mesma classe Θ .

A “solução” é $T(n) = \Theta(n^2)$, ou seja, $T(n) \in \Theta(n^2)$.

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

Recorrência do Mergesort

Podemos escrever a recorrência de tempo do **Mergesort** da seguinte forma

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

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

A prova é **essencialmente** a mesma do primeiro exemplo.
(**Exercício!**)

Cuidados com a notação assintótica

A notação assintótica é muito versátil e expressiva.
Entretanto, deve-se tomar alguns cuidados.

Considere a recorrência

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

É similar a recorrência do Mergesort!

Mas eu vou “provar” que $T(n) = O(n)$!

Cuidados com a notação assintótica

Vou mostrar que $T(n) \leq cn$ para alguma constante $c > 0$.

$$\begin{aligned}T(n) &= 2T(\lfloor n/2 \rfloor) + n \\ &\leq 2c\lfloor n/2 \rfloor + n \\ &\leq cn + n \\ &= O(n) \quad \leftarrow \text{ERRADO!!!}\end{aligned}$$

Por quê?

Não foi feito o passo indutivo, ou seja, não foi mostrado que $T(n) \leq cn$.

Teorema Master

- ▶ Veremos agora um resultado que descreve soluções para recorrências da forma

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

onde $a \geq 1$ e $b > 1$ são constantes.

- ▶ Isso é, convencionou-se para alguma constante n_0 , vale $T(n) = aT(n/b) + f(n)$ para todo $n \geq n_0$.
- ▶ A expressão n/b pode indicar tanto $\lfloor n/b \rfloor$ quanto $\lceil n/b \rceil$.
- ▶ O Teorema Master **não** fornece a resposta para **todas** as recorrências da forma acima.

Teorema (Master (CLRS))

Sejam $a \geq 1$ e $b > 1$ constantes, seja $f(n)$ uma função e seja $T(n)$ definida para os inteiros não-negativos pela relação de recorrência

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

Então $T(n)$ pode ser limitada assintoticamente da seguinte maneira:

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

Exemplos onde o Teorema Master se aplica:

▶ **Caso 1:**

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

$$T(n) = 4T(n/2) + n \log n$$

▶ **Caso 2:**

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

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

▶ **Caso 3:**

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

Exemplos onde o Teorema Master **não se aplica**:

- ▶ $T(n) = T(n-1) + n$
- ▶ $T(n) = T(n-a) + T(a) + n, (a \geq 1 \text{ inteiro})$
- ▶ $T(n) = T(\alpha n) + T((1-\alpha)n) + n, (0 < \alpha < 1)$
- ▶ $T(n) = T(n-1) + \log n$
- ▶ $T(n) = 2T(\frac{n}{2}) + n \log n$