Algoritmos

Pedro Hokama

Fontes

- [*clrs*] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest e Clifford Stein.
- [*timr*] Algorithms Illuminated Series, Tim Roughgarden
- Desmistificando Algoritmos, Thomas H. Cormen.
- Algoritmos, Sanjoy Dasgupta, Christos Papadimitriou e Umesh Vazirani
- Stanford Algorithms
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V`
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZl7A5HEK6`
- Conjunto de Slides dos Professores Cid C. de Souza, Cândida N. da Silva, Orlando Lee, Pedro J. de Rezende
- Conjunto de Slides do Professores Cid C. de Souza para a disciplina MO420

Qualquer erro é de minha responsabilidade.

Análise Assintótica

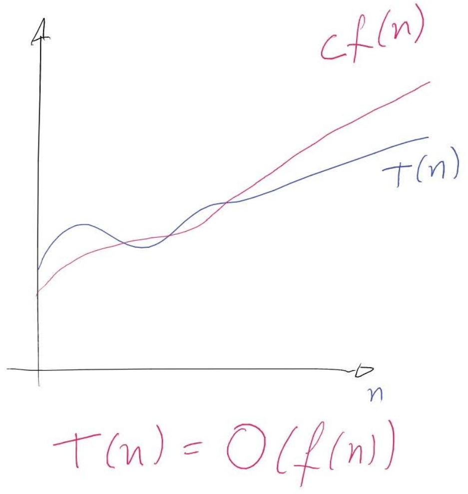
Ideia geral

Suprimir fatores constantes e termos de ordens inferiores.

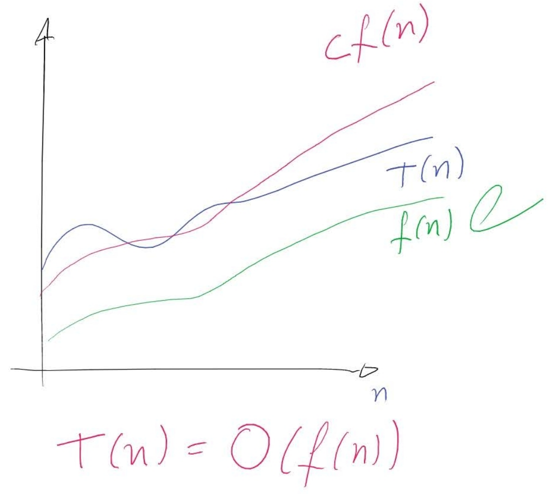
- Por exemplo em:
  $$6n \log_2 n + 6n$$
  6n é um termo de ordem inferior, 6 é constante então resultaria em:
  $$n \log n$$
  Exercício: A base do log também não importa. Por que?
- Então quando dizemos que o tempo de execução do MergeSort é $O(n \log n)$, ou de maneira geral quando dizemos que um algoritmo é $O(f(n))$. Estamos dizendo que depois de eliminar os termos de ordem inferior e constantes acabamos apenas com $f(n)$.

Notação O

- Big Oh, Ózão, O grande.
- Notação O é utilizada em funções definidas nos inteiros positivos.
- Seja $T(n)$ uma função sobre $n = 1, 2, 3, \ldots$
- $T(n) : \mathbb{Z}_+^* \to \mathbb{R}$
- Pergunta: Quando podemos dizer que $T(n) = O(f(n))$?
- Resposta: Se eventualmente, para um n suficientemente grande, $T(n)$ é limitado superiormente por uma alguma constante vezes $f(n)$.



- $T(n) = O(f(n))$ se quanto multiplicado por alguma constante c tal que $c \cdot f(n)$ está sempre acima de $T(n)$



- É válido mesmo se $f(n)$ ou $c' \cdot f(n)$ esteja sempre abaixo de $T(n)$, o importante é existir uma constante que a deixe sempre acima.

Formalizando o O

Definição

$T(n) = O(f(n))$ se e somente se exitem constantes c, $n_0 > 0$ tais que

$$T(n) \leq c.f(n)$$

para todo $n \geq n_0$.

- Então para provar que $T(n) = O(f(n))$ você precisa exibir c e n_o que provem que $T(n) \leq c.f(n)$ para todo $n \geq n_0$.
- Dizer que c e n_0 são constantes, significa que não dependem de n. (Tudo bem se c depender de n_0 e vice versa).
- (muito) Formalmente $O(f(n))$ é um conjunto de funções que podem ser limitadas por $f(n)$, então o correto seria dizer que $T(n) \in O(f(n))$. Mas dizer que $T(n) = O(f(n))$ é um abuso de notação comumente aceito e que facilita a manipulação dessas funções.

Exemplo

Teorema

Se $T(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0$ *então* $T(n) = O(n^k)$

- Para provar precisamos exigir constantes c e $n_0 > 0$ tais que $T(n) \leq c.f(n)$ para todo $n \geq n_0$.
- Vamos então escolher $n_0 = 1$ e $c = |a_k| + |a_{k-1}| + \ldots + |a_1| + |a_0|$.

$$\begin{aligned}
T(n) &= a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 \\
&\leq |a_k| n^k + |a_{k-1}| n^{k-1} + \ldots + |a_1| n + |a_0| \\
&\leq |a_k| n^k + |a_{k-1}| n^k + \ldots + |a_1| n^k + |a_0| n^k \text{ (vál. para } n > n_0) \\
&= (|a_k| + |a_{k-1}| + \ldots + |a_1| + |a_0|) n^k \\
&= cn^k
\end{aligned}$$

portanto, $T(n) \leq cn^k$ logo, $T(n) = O(n^k)$ $\square$

Teorema

Para todo $k \geq 1$, n^k não é $O(n^{k-1})$.

- Podemos provar por contradição, ou seja, assumimos que a premissa é verdadeira porém a conclusão é falsa e chegamos a algum absurdo.
- Supomos que existe um $k \geq 1$ e constantes c, $n_0 > 0$ tal que $n^k = O(n^{k-1})$, ou seja,

$$\begin{aligned} n^k &\leq cn^{k-1} & \forall n \geq n_0 \\ nn^{k-1} &\leq cn^{k-1} & \forall n \geq n_0 \\ n\cancel{n^{k-1}} &\leq c\cancel{n^{k-1}} & \forall n \geq n_0 \\ n &\leq c & \forall n \geq n_0 \end{aligned}$$

Como n pode ser todos os naturais maiores do que n_0, não pode existir tal constante c que seja maior do que qualquer natural. Portanto chegamos a um absurdo e provamos que todo $k \geq 1$, n^k não é $O(n^{k-1})$. □

Notação Ω e Θ

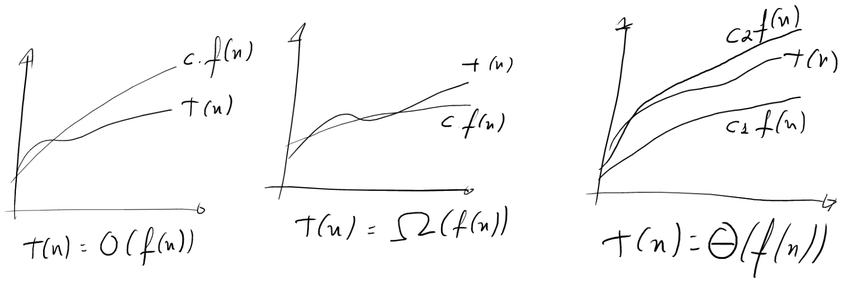
Analogia

- Intuitivamente, você pode comparar a notação O com uma relação de $\leq$. Já que dizer que $T(n)$ ser $O(f(n))$, quer dizer que: (existe constante c e n_0 tal que para todo n maior que n_0)

$$T(n) \leq cf(n)$$

- Nessa comparação a notação Ω seria como uma relação de $\geq$
- E a notação Θ seria como uma relação de $=$

Notação Ω e Θ



Notação Ω

A notação Ω que define um limitante inferior. Formalmente:

Definição

$T(n) = \Omega(f(n))$ se e somente se exitem constantes c, $n_0 > 0$ tais que

$$T(n) \geq c.f(n)$$

para todo $n \geq n_0$.

Notação Θ

A notação Θ indica que uma função $T(n)$ é limitada inferiormente e superiormente por uma função $f(n)$. Formalmente:

Definição

$T(n) = \Theta(f(n))$ se e somente se exitem constantes c_1, c_2 e $n_0 > 0$ tais que

$$c_1.f(n) \leq T(n) \leq c_2.f(n)$$

para todo $n \geq n_0$.

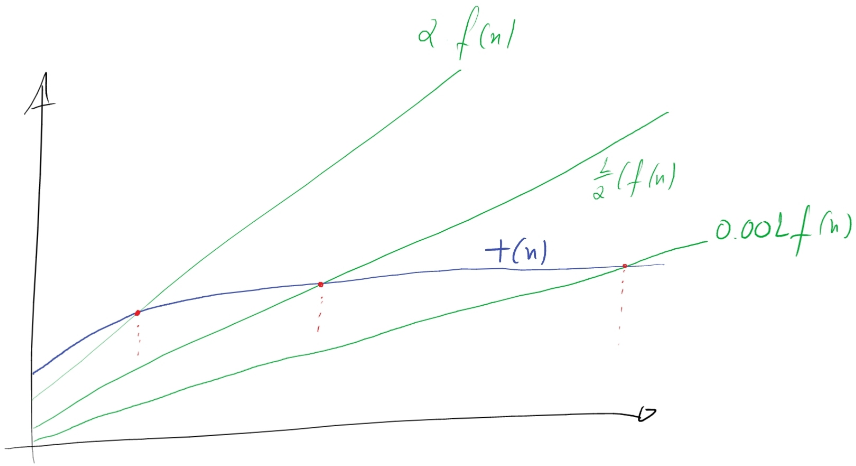
Definição

$T(n) = \Theta(f(n))$ se e somente se $T(n) = O(f(n))$ e também se $T(n) = \Omega(f(n))$.

o e ω

- A notação o (Little-Oh, ózinho, ó pequeno) é semelhante a notação O porém não assintoticamente justo.
- Informalmente pode-se pensar que se O está para uma relação do tipo $\leq$, enquanto o o é uma relação do tipo $<$.
- A notação ω (Little-omega, omeguinha, omega pequeno) é semelhante a notação Ω porém não assintoticamente justo.
- Informalmente pode-se pensar que Ω está para uma relação do tipo $\geq$, enquanto a ω é uma relação do tipo $>$.

o e ω



Notação o

Definição

$T(n) = o(f(n))$ se e somente se para toda constante $c > 0$, existe um $n_0 > 0$ tal que

$$T(n) < c.f(n)$$

para todo $n \geq n_0$.

Exercício: Provar que para todo $k \geq 1$, $n^{k-1} = o(n^k)$.

Notação ω

Definição

$T(n) = \omega(f(n))$ se e somente se para toda constante $c > 0$, existe um $n_0 > 0$ tal que

$$T(n) > c.f(n)$$

para todo $n \geq n_0$.

Mais exemplos de Notação Assintótica

Exemplo

Provar que $2^{n+10} = O(2^n)$.

Precisamos exibir c e $n_0 > 0$ tais que

$$2^{n+10} \leq c2^n \text{ para todo } n \geq n_0.$$

Como escolher c e n_0?

$$\begin{aligned} 2^{n+10} &\leq c2^n \\ 2^{10}2^n &\leq c2^n \\ 1024 \cdot 2^n &\leq c2^n \end{aligned}$$

Quando isso é verdade?
Escolhemos então algum $c \geq 1024$, e $n_0 \geq 1$.

Exemplo

Provar que $2^{n+10} = O(2^n)$.

Precisamos exibir c e $n_0 > 0$ tais que

$$2^{n+10} \leq c2^n \text{ para todo } n \geq n_0.$$

Escolhemos então $c = 1024$, e $n_0 = 1$ e vamos mostrar que:

$$\begin{aligned} 2^{n+10} &\leq 1024 \cdot 2^n \text{ para todo } n \geq 1 \\ 2^{10}2^n &\leq 1024 \cdot 2^n \text{ para todo } n \geq 1 \\ 1024 \cdot 2^n &\leq 1024 \cdot 2^n \text{ para todo } n \geq 1 \end{aligned}$$

logo $2^{n+10} = O(2^n)$. □

Exemplo

Provar que 2^{10n} não é $O(2^n)$.

Suponha por absurdo (por contradição) que $2^{10n} = O(2^n)$ e portanto $2^{10n} \leq c2^n$ para alguma constante c e $n \geq n_0$. Então

$$\begin{aligned} 2^{10n} &\leq c2^n \\ 2^{9n}2^n &\leq c2^n \\ 2^{9n}\cancel{2^n} &\leq c\cancel{2^n} \\ 2^{9n} &\leq c \end{aligned}$$

obviamente não existe uma constante que seja maior que 2^{9n} para todos os naturais n, portanto chegamos a uma contradição. Logo 2^{10n} não pode ser $O(2^n)$. □

Exemplo

Para quaisquer dois pares de funções positivas, $f(n)$ e $g(n)$, provar que $\max\{f(n), g(n)\} = \Theta(f(n) + g(n))$.

Para mostrar que a afirmação é verdadeira, podemos escolher $c_1 = 1/2$, $c_2 = 1$ e $n_0 = 1$ e verificamos que:

$$\frac{1}{2}(f(n) + g(n)) \leq \max\{f(n), g(n)\} \leq f(n) + g(n) \quad \forall n \geq n_0$$

Usando limites

$$T(n) \in o(f(n)) \text{ se } \lim_{n\to\infty} \frac{T(n)}{f(n)} = 0.$$

$$T(n) \in \omega(f(n)) \text{ se } \lim_{n\to\infty} \frac{T(n)}{f(n)} = \infty.$$

$$T(n) \in O(f(n)) \text{ se } \lim_{n\to\infty} \frac{T(n)}{f(n)} < \infty.$$

$$T(n) \in \Omega(f(n)) \text{ se } \lim_{n\to\infty} \frac{T(n)}{f(n)} > 0.$$

$$T(n) \in \Theta(f(n)) \text{ se } 0 < \lim_{n\to\infty} \frac{T(n)}{f(n)} < \infty.$$

Exemplo

$T(n) = \ln n$ e $f(n) = n^e$.

$$\lim_{n\to\infty} \frac{\ln n}{n^e} = \lim_{n\to\infty} \frac{1/n}{e.n^{e-1}} = 0.$$

portanto $\ln n = o(n^e)$. Obviamente $\ln n = O(n^e)$ também.