

Fontes principais

1. Cormen T. H.; Leiserson C. E.; Rivest R.; Stein C.. *Introduction to Algorithms*, 3^a edição, MIT Press, 2009
2. Análise de algoritmo - IME/USP (prof. Paulo Feofiloff)
http://www.ime.usp.br/~pf/analise_de_algoritmos

Objetivos da análise e projeto de algoritmos

- ▷ Prova de **corretude** de um algoritmo
- ▷ Análise da quantidade de recursos para executar um algoritmo (complexidade)
- ▷ Projeto de algoritmos utilizando várias técnicas

Definição informal de algoritmos

▷ **Algoritmo** é um procedimento bem definido para resolver um problema computacional específico.

Algoritmos

▷ **Busca e ordenação** são exemplos de problemas computacionais clássicos.

Problema e instância

Um **problema** é definido por:

- Uma descrição dos parâmetros
- Uma descrição sobre as propriedades que a resposta deve satisfazer

Problema e instância

Instância: Conjunto de valores da entrada necessários para que se possa resolver o problema.

Problema e instância

Problema 1. *Dada uma sequência de n números $\langle a_1, a_2, \dots, a_n \rangle$, determinar uma permutação $\langle a'_1, a'_2, \dots, a'_n \rangle$ da sequência de entrada tal que $a_1 \leq a_2 \leq \dots \leq a_n$.*

Problema e instância

Instância: Ordenar o vetor de maneira crescente.

- Entrada: [10, 20, 5, 40, 4]
- Saída: [4, 5, 10, 20, 40]

Problema da ordenação

Qual algoritmo de ordenação é melhor?

▷ Depende:

- do tamanho da instância a ser ordenada;
- quantidade de itens a serem ordenados
- tipo de dispositivo é utilizado

Ordenação por inserção

Insertion-Sort(A, n)

```
1: para  $j = 2$  até  $n$ 
2:    $chave = A[j]$ 
3:   ▷ Insere  $A[j]$  na sequência
      ordenada  $A[1..j - 1]$ 
4:    $i = j - 1$ 
5:   enquanto  $i > 0$  e  $A[i] > chave$ 
6:      $A[i + 1] = A[i]$ 
7:      $i = i - 1$ 
8:
9:    $A[i + 1] = chave$ 
10:
```

Análise de algoritmos

Definição 2. *Análise de um algoritmo significa prever os recursos que um algoritmo requer.*

Ex.: memória, comunicação, tempo computacional

Análise de algoritmos

Objetivo geral: é encontrar uma expressão matemática que é simples para escrever e manipular, que mostre as características importantes das necessidades do algoritmo e omita os detalhes desnecessários.

Ordenação por inserção

Insertion-Sort(A, n)	<i>cost</i>	<i>times</i>
1: <u>para</u> $j = 2$ até n	c_1	n
2: $chave = A[j]$	c_2	$n - 1$
3: ▷ Insere $A[j]$ na sequência ordenada $A[1..j - 1]$	0	$n - 1$
4: $i = j - 1$	c_4	$n - 1$
5: <u>enquanto</u> $i > 0$ e $A[i] > chave$	c_5	$\sum_{j=2}^n t_j$
6: $A[i + 1] = A[i]$	c_6	$\sum_{j=2}^n (t_j - 1)$
7: $i = i - 1$	c_7	$\sum_{j=2}^n (t_j - 1)$
8:		
9: $A[i + 1] = chave$	c_8	$n - 1$
10:		

t_j é o número de vezes que o teste do while da linha 5 é executado para cada j .

Análise da ordenação por inserção

- ▷ Tamanho da entrada: n
- ▷ Tempo de execução com uma instância de tamanho n

$$T(n) = c_1n + c_2(n - 1) + c_4(n - 1) + c_5 \sum_{j=2}^n t_j \\ + c_6 \sum_{j=2}^n (t_j - 1) + c_7 \sum_{j=2}^n (t_j - 1) + c_8(n - 1)$$

Análise da ordenação por inserção

- ▷ Melhor caso: sequência ordenada, $t_j = 1, \forall j$
- ▷ A linha 5 é executada 1 vez a cada iteração de 2 até n .
- ▷ As linhas 6 e 7 não são executadas

$$T(n) = c_1n + c_2(n - 1) + c_4(n - 1) + c_5(n - 1) + c_8(n - 1)$$

Análise da ordenação por inserção

- ▷ Melhor caso: sequência ordenada, $t_j = 1, \forall j$
- ▷ A linha 5 é executada 1 vez a cada iteração de 2 até n .
- ▷ As linhas 6 e 7 não são executadas

$$T(n) = c_1n + c_2(n - 1) + c_4(n - 1) + c_5(n - 1) + c_8(n - 1)$$

$$T(n) = (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8)$$

Análise da ordenação por inserção

- ▷ Melhor caso: sequência ordenada, $t_j = 1, \forall j$
- ▷ A linha 5 é executada 1 vez a cada iteração de 2 até n .
- ▷ As linhas 6 e 7 não são executadas

$$T(n) = c_1n + c_2(n - 1) + c_4(n - 1) + c_5(n - 1) + c_8(n - 1)$$

$$T(n) = (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8)$$

$$T(n) = an + b$$

Linear em n

Análise da ordenação por inserção

▷ Pior caso: sequência em ordem decrescente, $t_j = j, \forall j$

$$T(n) = c_1n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\ c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1)$$

Lembre-se: $\sum_{j=1}^n j = \frac{n(n+1)}{2}$

Análise da ordenação por inserção

▷ Pior caso: sequência em ordem decrescente, $t_j = j, \forall j$

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\ &\quad c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\ T(n) &= (c_5 + c_6 + c_7)n^2/2 + (c_1 + c_2 + c_4 + c_8 + c_9 + \frac{c_5 - c_6 - c_7}{2})n \\ &\quad - (c_2 + c_4 + c_5 + c_8) \end{aligned}$$

Análise da ordenação por inserção

▷ Pior caso: sequência em ordem decrescente, $t_j = j, \forall j$

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\ &\quad c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\ T(n) &= (c_5 + c_6 + c_7)n^2/2 + (c_1 + c_2 + c_4 + c_8 + c_9 + \frac{c_5 - c_6 - c_7}{2})n \\ &\quad - (c_2 + c_4 + c_5 + c_8) \\ T(n) &= an^2 + bn + c \end{aligned}$$

Quadrático em n .

Notação assintótica

Notação Θ

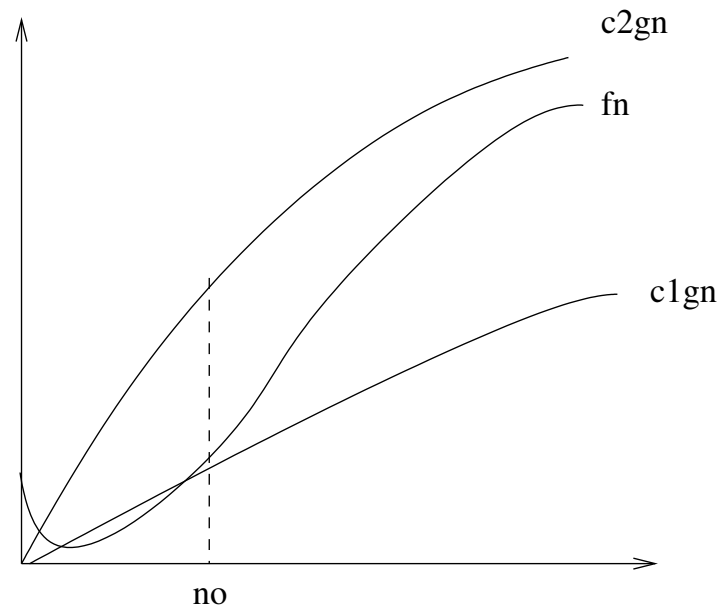
Notação Θ

Definição 3. Para uma dada função $g(n)$, denotamos por $\Theta(g(n))$ o conjunto de funções:

$$\Theta(g(n)) = \{f(n) : \exists c_1, c_2 > 0 \text{ e } n_0 > 0 \text{ tal que } c_1g(n) \leq f(n) \leq c_2g(n), \forall n \geq n_0\}$$

Notação Θ

Em vez de escrever $f(n) \in \Theta(g(n))$ escreveremos $f(n) = \Theta(g(n))$



A notação Θ é usada para delimitações exatas de funções.

Exemplo: Notação Θ

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

$$\frac{1}{2}n^2 - 3n \leq \frac{1}{2}n^2, \forall n \geq 0 \rightarrow c_2 = \frac{1}{2}, n_0^{(2)} \geq 0.$$

$$\frac{1}{4}n^2 \leq \frac{1}{2}n^2 - 3n \rightarrow c_1 = \frac{1}{4}, n_0^{(1)} \geq 12$$

$$n_0 = \max\{0, 12\} = 12$$

Portanto, $c_1 = \frac{1}{4}$, $c_2 = \frac{1}{2}$ e $n_0 = 12$

Notação O (oh grande)

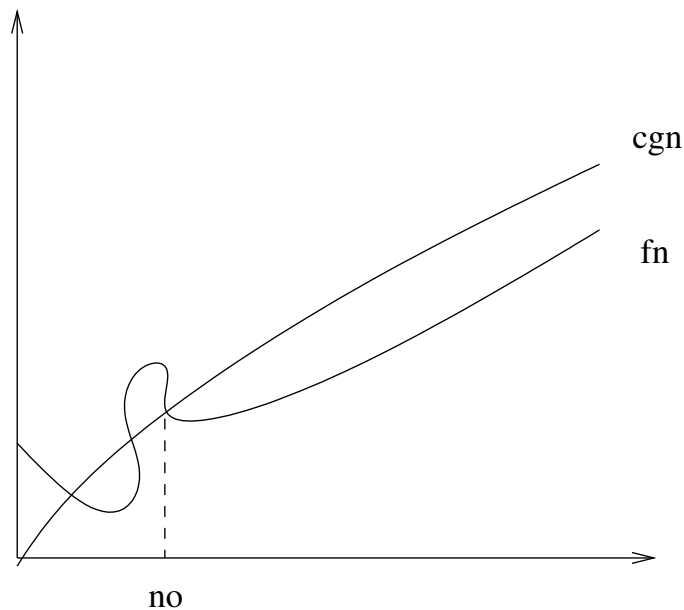
Notação O (oh grande)

Definição 4. Para uma dada função $g(n)$, denotamos por $O(g(n))$ o conjunto de funções:

$$O(g(n)) = \{f(n) : \exists c > 0 \text{ e } n_0 > 0 \text{ tal que } f(n) \leq cg(n), \forall n \geq n_0\}$$

Notação O (oh grande)

A notação O serve para atribuir delimitações superiores para a função $g(n)$.



Novamente escrevemos $f(n) = O(g(n))$ em vez de $f(n) \in O(g(n))$.

Exemplos: Notação O

(a) $n = O(n^2)$?

Sim, tome $c = 1$ e $n_0 = 1$

(b) $\frac{1}{2}n^2 - 3n = O(n^2)$?

Sim, tome $c = \frac{1}{2}$ e $n_0 = 7$

Notação Ω

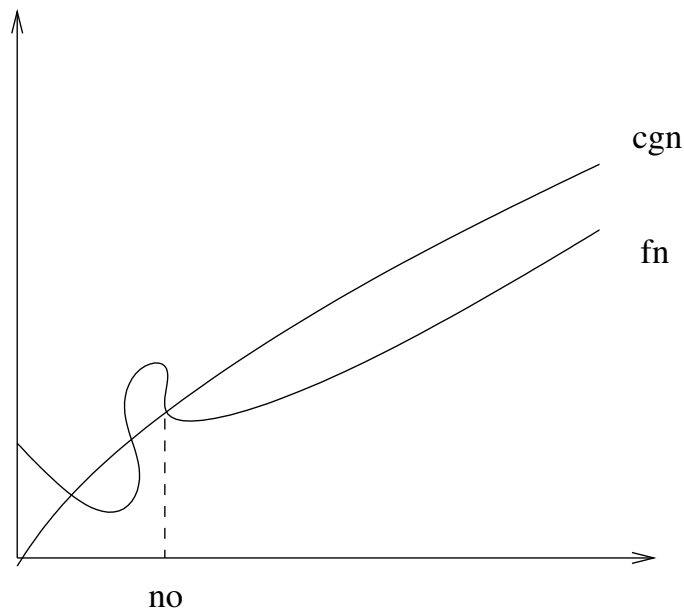
Notação Ω

Definição 5. Para uma dada função $g(n)$, denotamos por $\Omega(g(n))$ o conjunto de funções:

$$\Omega(g(n)) = \{f(n) : \exists c > 0 \text{ e } n_0 > 0 \text{ tal que } f(n) \geq cg(n), \forall n \geq n_0\}$$

Exemplos: Notação Ω

A notação Ω serve para atribuir delimitações inferiores para a função $g(n)$.



$$f(n) = \Omega(g(n))$$

Exemplos: Notação Ω

(a) $\frac{1}{3}n^2 - 3n = \Omega(n^2)$?

Sim, tome $c = \frac{1}{4}$ e $n_0 = 12$

(b) $n^2 - 2n = \Omega(n)$?

Sim , tome $c = 10$ e $n_0 = 18$

Notação o (oh pequeno)

Notação o (oh pequeno)

Para quando há apenas um **limite assintótico superior**, sem permitir $f(n) = cg(n)$.

Utiliza-se a notação o para denotar um limitante superior que é assintoticamente restrito.

Notação o (oh pequeno)

Definição 6. Para uma dada função $g(n)$, denotamos por $o(g(n))$ o conjunto de funções:

$$o(g(n)) = \{f(n) : \text{para qualquer constante positiva } c \text{ existe} \\ \text{uma constante positiva } n_0 \text{ tal que } f(n) < cg(n), \forall n \geq n_0\}$$

Intuitivamente, f é dominado por g assintoticamente.

Notação o (oh pequeno)

Por exemplo, $2n = o(n^2)$, mas $2n^2 \neq o(n^2)$

Intuitivamente, na notação o , a função $f(n)$ torna-se insignificante em relação a $g(n)$ quando vai para o infinito. Ou seja,

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

Notação o (oh pequeno)

Exemplo: $2n = o(n^2)$

$$\lim_{n \rightarrow \infty} \frac{2n}{n^2} = \lim_{n \rightarrow \infty} \frac{2}{n} = 0$$

Portanto, $2n = o(n^2)$.

Notação ω (omega pequeno)

Notação ω (omega pequeno)

Por analogia, a notação ω está para a notação Ω , assim como a notação o está para a notação O .

Então, para quando há apenas um limite assintótico inferior, sem permitir $f(n) = cg(n)$.

Utiliza-se a notação ω para denotar um limitante inferior que não é assintoticamente restrito.

Notação ω (omega pequeno)

Definição 7. Para uma dada função $g(n)$, denotamos por $\omega(g(n))$ o conjunto de funções:

$$o(g(n)) = \{f(n) : \text{para qualquer constante positiva } c \text{ existe} \\ \text{uma constante positiva } n_0 \text{ tal que } cg(n) < f(n), \forall n \geq n_0\}$$

Intuitivamente, f domina g assintoticamente.

Notação ω (omega pequeno)

Por exemplo, $2n^2 = \omega(n)$, mas $2n^2 \neq \omega(n^2)$

Intuitivamente na notação $\omega(n)$ a função $f(n)$ domina $cg(n)$ quando n vai para o infinito. Ou seja, $f(n) = \omega(g(n))$ indica que

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

Notação ω (omega pequeno)

Por exemplo, $n^2/2 = \omega(n)$.

$$\lim_{n \rightarrow \infty} \frac{n^2/2}{n} = \lim_{n \rightarrow \infty} \frac{n}{2} = \infty$$

Portanto, $n^2/2 = \omega(n)$.

Comparação de funções

Comparação de funções

▷ Transitividade

$f(n) = \Theta(g(n))$ e $g(n) = \Theta(h(n))$ então $f(n) = \Theta(h(n))$

$f(n) = O(g(n))$ e $g(n) = O(h(n))$ então $f(n) = O(h(n))$

$f(n) = \Omega(g(n))$ e $g(n) = \Omega(h(n))$ então $f(n) = \Omega(h(n))$

$f(n) = o(g(n))$ e $g(n) = o(h(n))$ então $f(n) = o(h(n))$

$f(n) = \omega(g(n))$ e $g(n) = \omega(h(n))$ então $f(n) = \omega(h(n))$

Comparação de funções

▷ Reflexividade

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

$$f(n) = O(f(n))$$

$$f(n) = \Omega(f(n))$$

$$f(n) \neq o(f(n))$$

$$f(n) \neq \omega(f(n))$$

Comparação de funções

▷ Simetria

$f(n) = \Theta(g(n))$, se somente se, $g(n) = \Theta(f(n))$

▷ Simetria transposta

$f(n) = O(g(n))$, se somente se, $g(n) = \Omega(f(n))$

$f(n) = o(g(n))$, se somente se, $g(n) = \omega(f(n))$

Analogia entre notação assintótica e números reais

Analogia entre notação assintótica e números reais

Sejam f , g funções e $a, b \in \mathbb{R}$

$$f(n) = O(g(n)) \approx a \leq b$$

$$f(n) = \Omega(g(n)) \approx a \geq b$$

$$f(n) = \Theta(g(n)) \approx a = b$$

$$f(n) = o(g(n)) \approx a < b$$

$$f(n) = \omega(g(n)) \approx a > b$$

Tricotomia

Para quaisquer dois números reais a e b , exatamente uma das seguintes sentenças deve ser verdadeira: $a < b$, $a = b$ ou $a > b$.

Apesar de quaisquer dois números poderem ser comparados, nem todas as funções são assintoticamente comparáveis. Ou seja, para duas funções $f(n)$ e $g(n)$, pode ser que nem $f(n) = O(g(n))$ nem $f(n) = \Omega(g(n))$.

Tricotomia

Exemplo: Considere as funções $f(n) = n$ e $g(n) = n^{1+\sin n}$

Neste exemplo, a expressão $1 + \sin n$ varia entre 0 e 2.

$$n^{1+\sin n} = \begin{cases} 1 & \text{se } n \in \{3\pi/2, 7\pi/2, \dots\} \\ n & \text{se } n \in \{\pi, 3\pi, \dots\} \\ n^2 & \text{se } n \in \{\pi/2, 5\pi/2, \dots\} \end{cases}$$

Não se pode afirmar que $f(n) = O(g(n))$ e nem $f(n) = \Omega(g(n))$.

Notação assintótica em expressões

Como interpretar, por exemplo, $2n^2 + n - 1 = 2n^2 + \Theta(n)$?

Existem uma função $f(n) = \Theta(n)$ tal que

$$2n^2 + n - 1 = 2n^2 + f(n).$$

Neste caso, $f(n) = n - 1$.

Notação assintótica em expressões

E a expressão $2n^2 + \Theta(n) = \Theta(n^2)$?

Para toda função $f(n) = \Theta(n)$, existe uma função $g(n) = \Theta(n^2)$ tal que $2n^2 + f(n) = g(n)$.

Por exemplo,

$$f(n) = n - \log n + 5 = \Theta(n) \text{ e}$$

$$g(n) = 2n^2 + n - \log n + 5 = \Theta(n^2).$$

Principais classes O (oh grande)

Complexidade	Nome	Eficiente
$O(1)$	Constante	sim
$O(\log n)$	Logaritmica	
$O(n)$	Linear	
$O(n \log n)$	“Linearítma”	
$O(n^2)$	Quadrática	não
$O(n^3)$	Cúbica	
$O(2^n)$	Exponencial	
$O(n!)$	Fatorial	

Obrigado