Aula 4: Notação O

- Definições das notações O, Ω e θ
- Manipulação de expressões em notação O
- Conceito de algoritmos ótimos

A Notação O

- Relembrando:
  - Nas complexidades são irrelevantes constantes aditivas ou multiplicativas
  - Somente valores assintóticos
- Exemplos:

$6n^3 \longrightarrow n^3$		$n^2+n \longrightarrow n^2$	
$n^2+5 \longrightarrow n^2$		$3n^2+5n-7 \longrightarrow n^2$	
$6n^2+4 \longrightarrow n^2$		$2n^3+\log n \longrightarrow n^3$	

- Objetivo: encontrar operadores matemáticos que possam representar as situações acima
- Notações: O, Ω e θ

Definição da Notação O

- Sejam f, h funções reais de variável inteira n. Digamos que f é O(h), escrevendo-se f = O(h), quando existir constante c > 0 e um valor inteiro n_0, tais que:

$$n > n_0 \Rightarrow f(n) \leq c.h(n)$$

- A função h atua como limite superior para valores assintóticos da função f.

- Exemplos:

$f=n^2-1 \Rightarrow f=O(n^2)$	$f=5 + 2 \log n + 3 \log^2 n \Rightarrow f=O(\log^2 n)$
$f=n^2-1 \Rightarrow f=O(n^3)$	$f=5n + 2 \log n + 3 \log^2 n \Rightarrow f=O(n)$
$f=403 \Rightarrow f=O(1)$	$f=3n + 5 \log n + 2 \Rightarrow f=O(n)$
$f=54 \Rightarrow f=O(1)$	$f=5.2^n + 5.n^{10} \Rightarrow f=O(2^n)$

Propriedades da Notação O

- Sejam g, h funções reais positivas e k uma constante:
  - O(g+h) = O(g) + O(h)
  - O(g.h) = O(g) . O(h)
  - O(k.g) = k.O(g) = O(g)
- Notação O: utilizada para exprimir complexidades
- Exemplos:

Algoritmo	Complexidade
inversão de uma sequência	O(n)
cálculo de fatorial	O(n)
soma de matrizes	$O(n^2)$
produto de matrizes	$O(n^3)$

Complexidade de Procedimentos Recursivos

- Um método:
  - Determinar o número total de chamadas do procedimento recursivo;
  - Determinar a complexidade de execução de uma única chamada, sem considerar as chamadas recursivas;
  - Complexidade total = número de chamadas vezes complexidade de cada chamada
- Exemplo: algoritmo de fatorial (recursivo):
  - Número de chamadas = n
  - Complexidade de cada chamada = $O(1)$
  - Complexidade = $n.O(1) = O(n)$
- Exemplo: algoritmo da Torre de Hanói:
  - Número de chamadas = $O(2^n)$
  - Complexidade de cada chamada = $O(1)$
  - Complexidade = $O(2^n).O(1) = O(2^n)$

Exercícios

Escrever as seguintes funções, em notação O:

1) $n^3 - 1$
2) $n^2 + 2.\log n$
3) $3.n^n + 5.2^n$
4) $(n-1)^n + n^{n-1}$
5) $5.3^n + 4.2^{n^2}$
6) 6.547.326
7) $5.n^7 + 3.2^n + n!$
8) $3.n + 7.m + 2$
9) $5.n^2 + 9.m + 4$
10) $3.n + 5.m + n.m$

Tempo: 3 minutos

Notação O

Solução:

1) $O(n^3)$
2) $O(n^2)$
3) $O(n^n)$
4) $O(n^n)$
5) $O(2^{n^2})$
6) $O(1)$
7) $O(n!)$
8) $O(n + m)$
9) $O(n^2 + m)$
10) $O(n.m)$

Notação θ

- Para exprimir limites superiores justos, utiliza-se a notação θ
- Sejam f, g funções reais positivas da variável n. Diz-se que f é θ(g), escrevendo-se f=θ(g), quando ambas as condições f=O(g) e g=O(f) forem verificadas.
- A notação θ exprime o fato de que duas funções possuem a mesma ordem de grandeza assintótica.
- Quando possível, a notação θ deve ser preferida à notação O.

Exemplos de Notação θ

Exemplo:

$f = n^2 - 1$

$g = n^2$

$h = n^3$

f é O(h)
g é O(f)
h não é O(f)

f é θ(g)
f não é θ(h)

Exemplo:

$f = 5+2.\log n+\log^2 n$

$g = n$

f é O(g)
g não é O(f)

f não é θ(g)
No caso,
f é $\theta(\log^2 n)$

Exercícios

- Certo ou Errado?
  Se a complexidade (de pior caso) de um algoritmo for `f`, então o número de passos que o algoritmo efetua é igual a O(`f`).
- Certo ou Errado?
  Se a complexidade (de pior caso) de um algoritmo for `f`, então o número de passos que o algoritmo efetua é igual a θ(`f`).
- Verificar se `f` é θ(`g`).
  3.1) $f = 3.n^2 + \log n$, $g = 2.n^2 + 1$
  3.2) $f = 3.2^n + 1$, $g = 250.\,n^7 + \log n$
  3.3) $f = 525$, $g = 10$

Tempo: 3 minutos

Solução

- Certo.
- Errado.
- 3.1) f é θ(g).
  3.2) f não é θ(g).
  3.3) f é θ(g).

Notação Ω

- Para exprimir limites inferiores, utiliza-se a notação Ω.
- Sejam f, h funções reais positivas da variável inteira n.
  Diz-se que f é Ω(h), escrevendo-se f = Ω(h), quando existir uma constante c > 0 e um valor inteiro n_0 tal que

$$n > n_0 \Rightarrow f(n) \geq c.h(n)$$

- Exemplos:
  $f = n^2 - 1 \Rightarrow f = O(n^2)$
  $f = n^2 - 1 \Rightarrow f = \Omega(1)$
  $f = n^2 - 1 \Rightarrow$ não vale $f = \Omega(n^3)$
- Em geral, para a notação Ω(f) procura-se um valor maior possível para f.

Exercícios

- Certo ou errado?

  Se f, g são funções tais que $f = O(g)$ e $g = \Omega(f)$, então $f = \theta(g)$.

- Certo ou errado?

  Se a complexidade de melhor caso de um algoritmo for f, então o número de passos que o algoritmo efetua é $\Omega(f)$.

- Determinar o valor $\Omega(f)$, nos seguintes casos:

  3.1) $f = 3.n^2 + 2.\log n$

  3.2) $f = n + 5.\log n$

  3.3) $f = n^3 + 2.n + 7$

Tempo: 3 minutos

Exercícios

- Errado.

- Certo.

- 3.1) $f = \Omega(\log n)$
  3.2) $f = \Omega(\log n)$
  3.3) $f = \Omega(1)$

Algoritmos Ótimos

- Noção de complexidade:
  - relacionada a um dado algoritmo específico;
  - não considera a possível existência de outros algoritmos, para o mesmo problema.
- Próximo objetivo:
  - comparar as eficiências de diferentes algoritmos, para um mesmo problema.

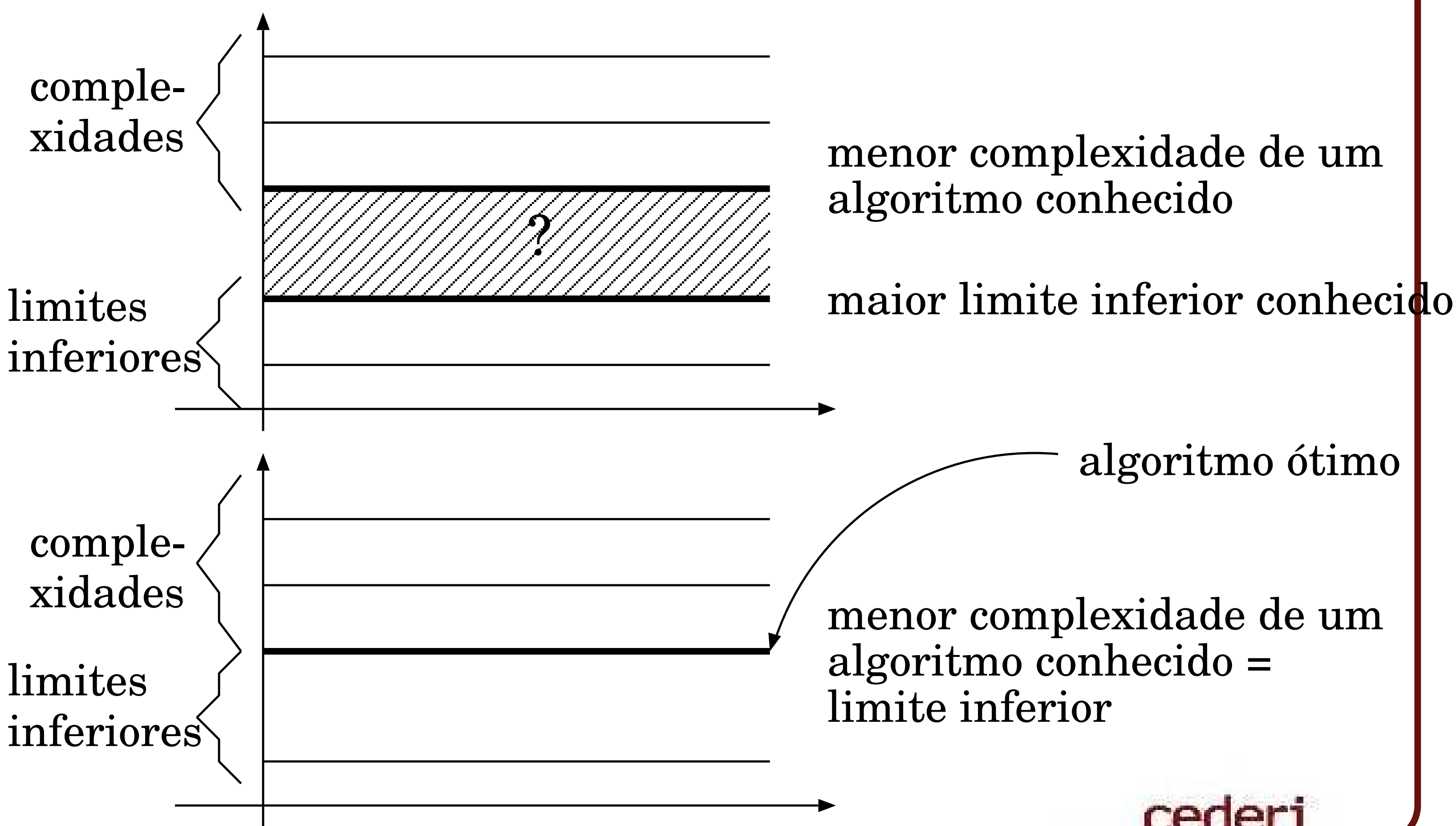
Algoritmos Ótimos

- Seja P um problema. Um limite inferior para P é uma função $\mathtt{l}$, tal que a complexidade de pior caso de qualquer algoritmo que resolva P é $\Omega(\mathtt{l})$.

- Se existir um algoritmo A, cuja complexidade seja $O(\mathtt{l})$, então A é denominado algoritmo ótimo para o problema P. Nesse caso, o limite $\Omega(\mathtt{l})$ é o melhor (maior) possível.

Algoritmos Ótimos

Um algoritmo ótimo apresenta a menor complexidade dentre todos os possíveis algoritmos para o mesmo problema.



Algoritmos Ótimos

- Exprimir complexidades: notação O
  Exprimir limites inferiores: notação Ω

- Determinação de limites inferiores:
  - pode ser de difícil tratamento matemático;
  - limite inferior natural: quantidade de dados de entrada.

- Exemplo: inversão de uma seqüência
  Complexidade do algoritmo: O (n)
  Quantidade de dados: O (n) => Limite inferior: Ω (n)

  Logo, o algoritmo é ótimo.

Exemplo: Soma de matrizes

- Complexidade do algoritmo: O(n^2)
  Limite inferior: O(n^2)
  Logo, algoritmo ótimo.

- Exemplo: Produto de matrizes
  Complexidade do algoritmo: O(n^3)
  Limite inferior: O(n^2)
  Algoritmo ótimo?

  - O algoritmo não é ótimo porque há outros de complexidade menor. Contudo, não se sabe se estes são ótimos ou não.

- Exemplo: Problema de ordenação
  Menor complexidade conhecida: O(n log n)
  Maior limite inferior conhecido: Ω(n log n)
  Logo, o algoritmo é ótimo.

Exercícios

- Certo ou errado?

  A complexidade de melhor caso de um algoritmo é necessariamente maior ou igual a qualquer limite inferior para o mesmo problema.

- A seqüência de Fibonacci é uma seqüência de elementos $f_1, \ldots, f_n$ definida do seguinte modo:

$$\begin{cases} f_1 = 0 \\ f_2 = 1 \\ f_j = f_{j-1} + f_{j-2}, \; j > 2 \end{cases}$$

  Elaborar um algoritmo para determinar o elemento f_n da seqüência, cuja complexidade seja linear em n.

Tempo: 5 minutos.

Solução

- Errado.

- <u>Algoritmo</u>: seqüência de Fibonacci

```
F(1) := 0
F(2) := 1
para j := 3, ..., n faça
   F(j) := F(j-1) + F(j-2)
```

Complexidade: O(n)