classificação e pesquisa de dados

notação assintótica

1

Revisão

- Algoritmos
- Introdução a análise de algoritmos
- Tipos de análise: empírica *versus* assintótica
- Modelo computacional RAM
- Métrica: contagem de instruções primitivas

2

Alguns princípios para análise de algoritmos

- Tempo de execução de **comandos** de atribuição, leitura ou escrita, avaliação de expressão lógica podem ser considerados constante
- Tempo de execução de uma **sequência de comandos** ⇒ **maior** tempo de execução de **qualquer** comando da sequência
- Tempo de execução de uma **estrutura condicional** ⇒ tempo dos comandos **dentro do corpo** da estrutura **mais** o tempo para avaliar a condição
- Tempo de execução de uma **estrutura de repetição** ⇒ tempo dos comandos **dentro do corpo** da estrutura **mais** o tempo para avaliar a condição, multiplicado pelo número de iterações
- Tempo de execução de **chamada** a uma **subrotina não recursiva** deve ser computado separadamente e incluído no cálculo da subrotina chamadora

3

Análise assintótica

- baseada em um modelo matemático
- emprega um computador idealizado
  - **independente** de hardware e software
- pode ser aplicada em uma representação de alto nível de um algoritmo (pseudocódigo)
  - **não é necessário** implementar o algoritmo

4

Análise assintótica

associa uma função de complexidade f a um algoritmo

$f(n)$ é a medida do tempo necessário para executar um algoritmo para um problema de tamanho n

$f(n)$ é a quantidade de memória necessária para executar um algoritmo para um problema de tamanho n

Exemplo

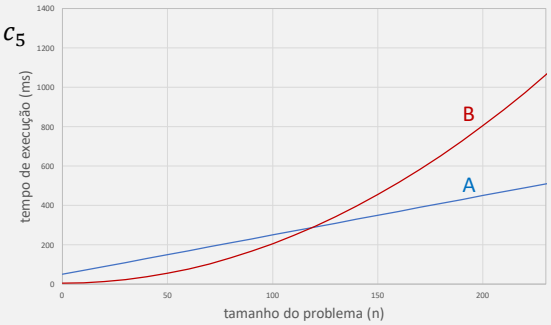
$\boldsymbol{f(n) = c_1 n + c_2}$

5

Comportamento linear (B) x quadrático (A)

A: $f(n) = c_1 n + c_2$

B: $g(n) = c_3 n^2 + c_4 n + c_5$

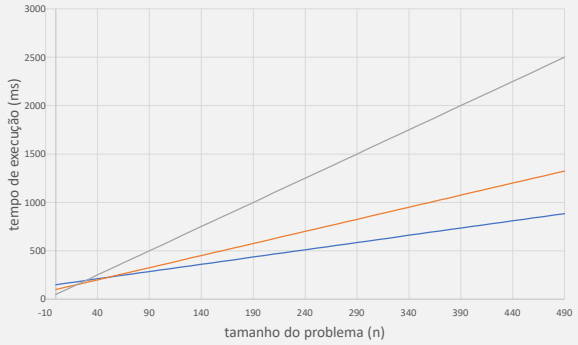


6

Classe de comportamento assintótico: **linear**

$f(n) = c_1 n + c_2$

c1	c2
5	50
2,5	150
1,5	100



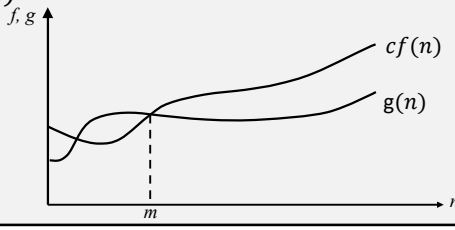
7

Notação O (Big O)

Definição: uma função $f(n)$ **domina** assintoticamente outra função $g(n)$ se

existem constantes positivas c e m tais que, para $n \geq m$, temos $0 \leq g(n) \leq cf(n)$

Limite assintótico superior

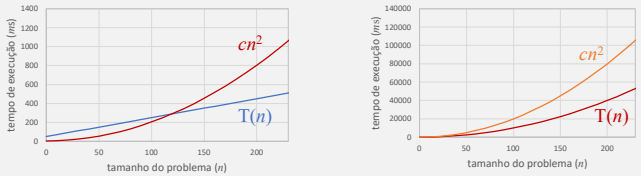


8

Notação O (Big O)

Dizer que o tempo $T(n)$ de execução de um programa é $O(n^2)$, ou seja, $\mathrm{T}(n) = O(n^2)$ significa afirmar que:

existem constantes positivas c e m tais que para valores $n \geq m$, temos $\mathrm{T}(n) \leq cn^2$



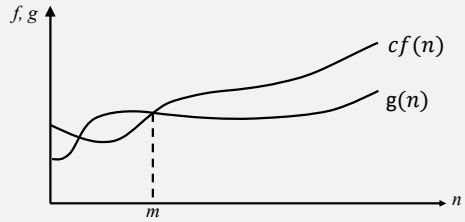
Notação O (Big O)

Usamos as descrições a seguir para expressar que $f(n)$ domina assintoticamente $g(n)$

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

$g(n)$ é $O(f(n))$

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



Notação O (Big O)

Como mostrar que $n \in O(n^2)$ é **verdadeiro**?

encontrando duas constantes positivas c e m tais que para valores $n \geq m$, temos $n \leq cn^2$

$$n \leq cn^2 \Rightarrow n \leq n^2 \Rightarrow 1 \leq 1^2 \Rightarrow 1 \leq 1$$

$c = 1$ $m = 1$

Constantes: $c = 1$ e $m = 1$

Notação O (Big O)

Como mostrar que $n \in O(n^2)$ é **verdadeiro**?

encontrando duas constantes positivas c e m tais que para valores $n \geq m$, temos $n \leq cn^2$

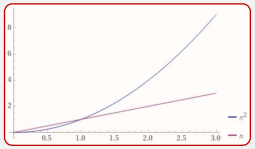
$$n \leq cn^2 \Rightarrow \frac{n}{n^2} \leq c \Rightarrow \frac{1}{n} \leq c \Rightarrow 1 \leq 1$$

$c = 1$

$m = 1$

Constantes

$c = 1$ e $m = 1$



Notação O (Big O)

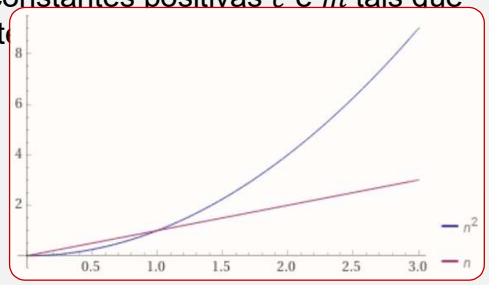
Como mostrar que $n \in O(n^2)$ é **verdadeiro**?

encontrando duas constantes positivas c e m tais que para valores $n \geq m$, t

$$n \leq cn^2 \Rightarrow \frac{n}{n^2} \leq c$$

Constantes

$c = 1$ e $m = 1$



13

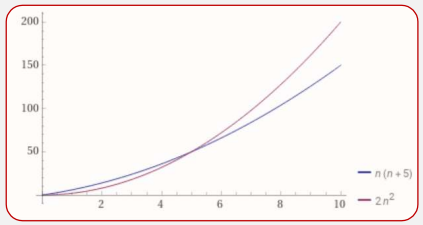
Notação O (Big O)

$n^2 + 5n \in O(n^2)$: Verdadeiro ou Falso?

$c = 2$

$$n^2 + 5n \leq cn^2 \Rightarrow \frac{n^2}{n^2} + \frac{5n}{n^2} \leq \frac{cn^2}{n^2} \Rightarrow 1 + \frac{5}{n} \leq c \Rightarrow 2 \leq 2$$

$m = 5$



Constantes

$c = 2$ e $m = 5$

14

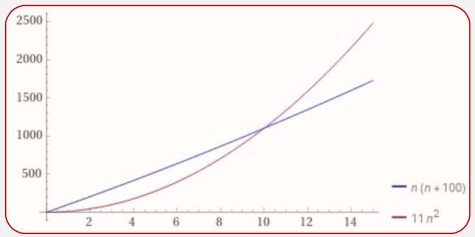
Notação O (Big O)

$n^2 + 100n \in O(n^2)$: Verdadeiro ou Falso?

$c = 11$

$$n^2 + 100n \leq cn^2 \Rightarrow \frac{n^2}{n^2} + \frac{100n}{n^2} \leq \frac{cn^2}{n^2} \Rightarrow 1 + \frac{100}{n} \leq c \Rightarrow 11 \leq 11$$

$m = 10$



Constantes

$c = 11$ e $m = 10$

15

Notação O (Big O)

$n^2 + 100n \in O(n^2)$: Verdadeiro ou Falso?

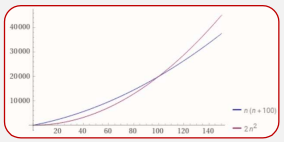
$c = 2$

$$n^2 + 100n \leq cn^2 \Rightarrow \frac{n^2}{n^2} + \frac{100n}{n^2} \leq \frac{cn^2}{n^2} \Rightarrow 1 + \frac{100}{n} \leq c \Rightarrow 2 \leq 2$$

$m = 100$

Constantes

$c = 2$ e $m = 100$



17

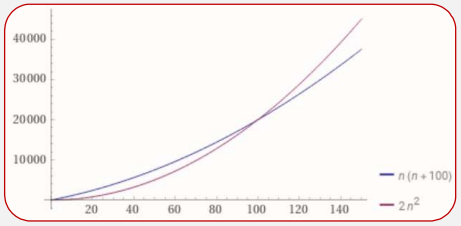
Notação O (Big O)

$n^2 + 100n \in O(n^2)$: Verdadeiro ou Falso?

$c = 2$

$$n^2 + 100n \leq cn^2 \Rightarrow \frac{n^2}{n^2} + \frac{100n}{n^2} \leq \frac{cn^2}{n^2} \Rightarrow 1 + \frac{100}{n} \leq c \Rightarrow 2 \leq 2$$

$m = 100$



Constantes

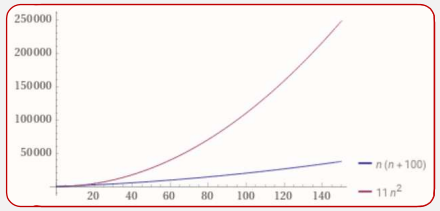
$c = 2$ e $m = 100$

Notação O (Big O)

$n^2 + 100n \in O(n^2)$: Verdadeiro ou Falso?

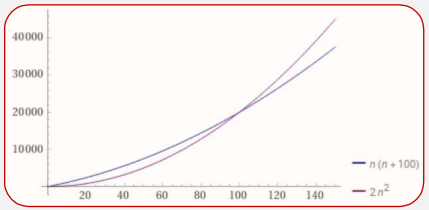
Constantes

$c = 11$ e $m = 10$



Constantes

$c = 2$ e $m = 100$



Notação O (Big O)

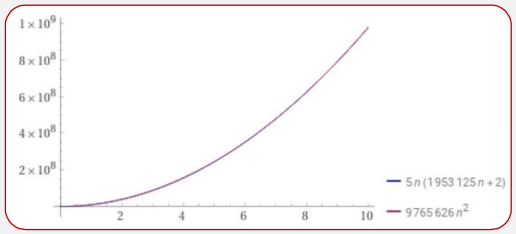
$5^{10}n^2 + 10n \in O(n^2)$: Verdadeiro ou Falso?

$$5^{10}n^2 + 10n \leq cn^2$$

$$5^{10}\frac{n^2}{n^2} + \frac{10n}{n^2} \leq \frac{cn^2}{n^2}$$

$$5^{10} + \frac{10}{n} \leq c$$

$$5^{10} + 10 \leq 5^{10} + 10$$



Constantes

$c = 5^{10} + 10$ e $m = 1$

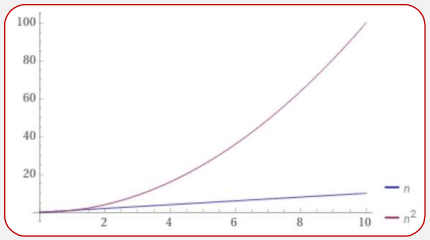
Notação O (Big O)

$n^2 \in O(n)$: Verdadeiro ou Falso?

$$n^2 \leq cn$$

$$\frac{n^2}{n} \leq c$$

$$n \leq c$$



Constantes

$\nexists\, c, m \mid n \leq c$

Notação O (Big O)

$0{,}001n^2 \in O(n)$: Verdadeiro ou Falso?

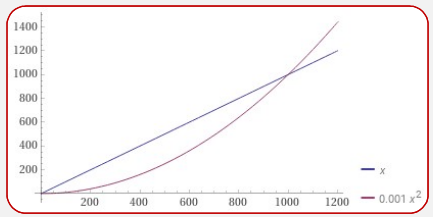
$0{,}001n^2 \leq cn$

$\frac{0{,}001n^2}{n} \leq c$

$0{,}001n \leq c$

Constantes

$\nexists\, c, m \mid n \leq c$



Notação O (Big O)

$2^{n+1} \in O(2^n)$: Verdadeiro ou Falso?

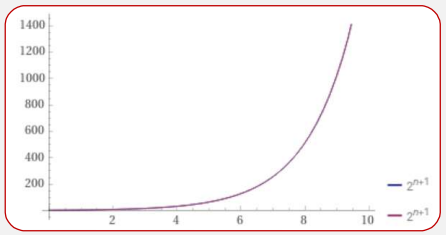
$2^{n+1} \leq c\, 2^n$

$2 \times 2^n \leq c\, 2^n$

$2 \leq c$

Constantes

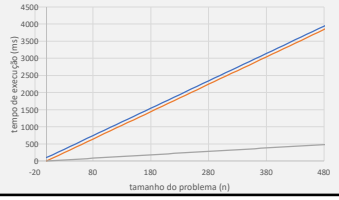
$c = 2$ e $m = 1$



Comportamento quando $n \to \infty$

		n						
		1	**10**	**100**	**1000**	**10000**	**100000**	**1000000**
T(n)	**$8n$+100**	108	180	900	8100	80100	800100	8000100
	$8n$	8	80	800	8000	80000	800000	8000000
	n	1	10	100	1000	10000	100000	1000000

Podemos desprezar fatores constantes



Comportamento quando $n \to \infty$

	n				
	1	**10**	**100**	**1000**	**10000**
$f(n) = n^2$	1	100	10000	1000000	100000000
$g(n) = n^2 + n$	2	110	10100	1001000	100010000
Δ	100%	10,00%	1,00%	0,10%	0,01%

$\Delta = \frac{g(n) - f(n)}{f(n)}$

$\Delta = \frac{(n^2 + n) - n^2}{n^2}$

Podemos desprezar termos de menor ordem

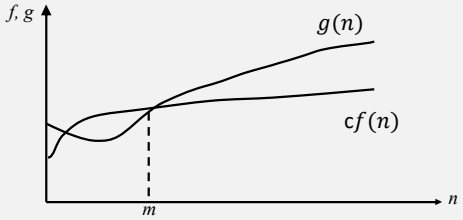
Operações com a notação O

$$f(n) = O(f(n)$$
$$c \times O(f(n)) = O(f(n))$$
$$O(f(n)) + O(f(n)) = O(f(n))$$
$$O(f(n)) + O(g(n)) = O(max(f(n), g(n)))$$
$$O(f(n)) \times O(g(n)) = O(f(n)g(n))$$
$$f(n) \times O(g(n)) = O(f(n)g(n))$$

Notação Ω (Big ômega)

Definição: uma função $g(n)$ é $\Omega(f(n))$ se existem constantes positivas c e m tais que, para $n \geq m$, temos $g(n) \geq cf(n) \geq 0$

Limite assintótico inferior



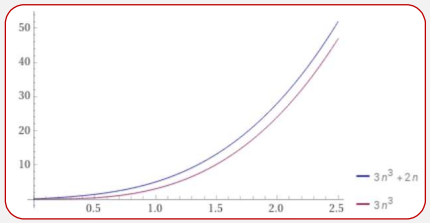
Notação Ω (Big ômega)

O que significa dizer que $3n^3 + 2n \in \Omega(n^3)$?

significa encontrar as constantes positivas c e m tais que para valores $n \geq m$, temos $3n^3 + 2n \geq \Omega(n^3)$

$$3n^3 + 2n \geq cn^3 \Rightarrow 3 + \frac{2}{n^2} \geq c$$

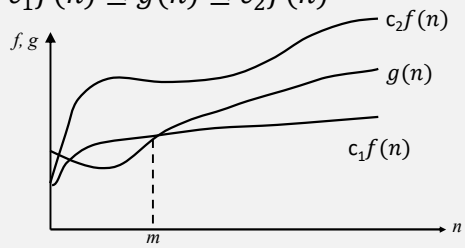
Constantes
$c = 3$ e $m = 1$



Notação Θ

Definição: uma função $g(n)$ é $\Theta(f(n))$ se existem constantes positivas c_1, c_2 e m tais que, para $n \geq m$, temos $0 \leq c_1 f(n) \leq g(n) \leq c_2 f(n)$

Limite assintótico firme (restrito)



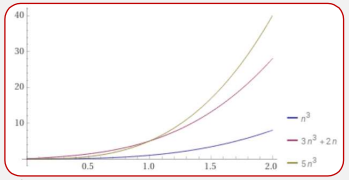
Notação Θ

O que significa dizer que $3n^3 + 2n \in \Theta(n^3)$

significa encontrar as constantes positivas c_1, c_2 e m tais que para valores $n \geq m$, temos $c_1 n^3 \leq 3n^3 + 2n \leq c_2 n^3$

$c_1 n^3 \leq 3n^3 + 2n \Rightarrow c_1 \leq 3 + \frac{2}{n^2}$

$3n^3 + 2n \leq c_2 n^3 \Rightarrow 3 + \frac{2}{n^2} \leq c_2$



Constantes: $c_1 = 1$, $c_2 = 5$ e $m = 1$

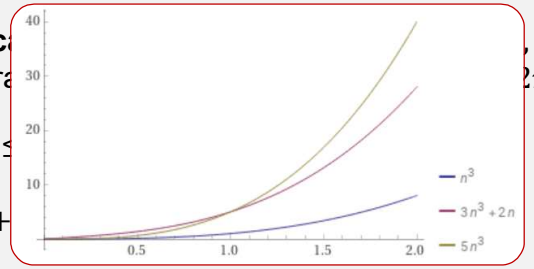
Notação Θ

O que significa dizer que $3n^3 + 2n \in \Theta(n^3)$

signific, c_2 e m tais que para
$2n \leq c_2 n^3$

$c_1 n^3 \leq$

$3n^3 +$



Constantes: $c_1 = 1$, $c_2 = 5$ e $m = 1$

Classes de comportamento assintótico

$f(n) = O(1)$

Complexidade **constante**

Independe do tamanho da entrada

Instruções do algoritmo são executas um número **constante** de vezes

Classes de comportamento assintótico

$f(n) = O(1)$

Complexidade **constante**

```
a += 1000;
```

```
int s = 0;
for(int i = 1; i <= 100; i++)
     s++;
```

Classes de comportamento assintótico

$f(n) = O(\log n)$

Complexidade **logarítmica**

Ocorre **tipicamente** em algoritmos que resolvem um problema transformando-o em problemas menores

Tempo de execução pode ser considerado **menor** do que uma constante **grande**

Classes de comportamento assintótico

$f(n) = O(\log n)$

Complexidade **logarítmica**

```
int s = 0;
for(int i = n; i >= 1; i/=2)
     s++;
```


Classes de comportamento assintótico

$f(n) = O(n)$

Complexidade **linear**

Geralmente, algumas instruções são executadas sobre **cada** elemento da entrada

Melhor situação possível para algoritmos que **necessitam** processar n itens da entrada.

Classes de comportamento assintótico

$f(n) = O(n)$

Complexidade **linear**

```
int s = 0;
for(int i = 1; i <= n; i++)
     s++;
```

Classes de comportamento assintótico

$f(n) = O(n \log n)$

Complexidade **log-linear**

Ocorre **tipicamente** em algoritmos que resolvem um problema **quebrando-o** em problemas menores, resolvendo cada um **independentemente** e depois **juntando-os**

Classes de comportamento assintótico

$f(n) = O(n \log n)$

Complexidade **log-linear**

```
int s = 0;
for(int i = 1; i <= n; i++)
  for(int j = n; j >= 1; j/=3)
     s++;
```


Classes de comportamento assintótico

$f(n) = O(n^2)$

Complexidade **quadrática**

```
int s = 0;
for(int i = 1; i <= n; i++)
  for(int j = 1; j <= n; j++)
     s++;
```


Classes de comportamento assintótico

$f(n) = O(n^3)$

Complexidade **cúbica**

Algoritmos dessa ordem de complexidade são **úteis** apenas para resolver problemas **pequenos**

Normalmente aparecem em **três** estruturas de repetição aninhadas.

Classes de comportamento assintótico

$f(n) = O(n^3)$

Complexidade **cúbica**

```
int s = 0;
for(int i = 1; i <= n; i++)
  for(int j = 1; j <= n; j++)
    for(int k = 1; k <= n; k++)
      s++;
```


Classes de comportamento assintótico

$f(n) = O(a^n)$

Complexidade **exponencial**

Algoritmos dessa ordem de complexidade normalmente **não** são úteis do ponto de vista prático

Geralmente ocorre quando se emprega **força bruta**

Classes de comportamento assintótico

$$O(1) \ll O(\log n) \ll O(n) \ll O(n \log n) \ll O(n^2) \ll O(n^3)$$

$$O(n^3) \ll O(n^a) \ll O(2^n) \ll O(a^n) \ll O(n!)$$

ZIVIANI, Nivio. Projeto de Algoritmos com Implementação em Java e C++. São Paulo: Thomson Learning, 2007.

LEISERSON, C. E.; STEIN, C.; RIVEST, R. L., CORMEN, T.H. Algoritmos: Teoria e Prática. Tradução da 2ª edição americana. Rio de Janeiro: Campus, 2002.