FUNDAMENTOS MATEMÁTICOS PARA COMPUTAÇÃO

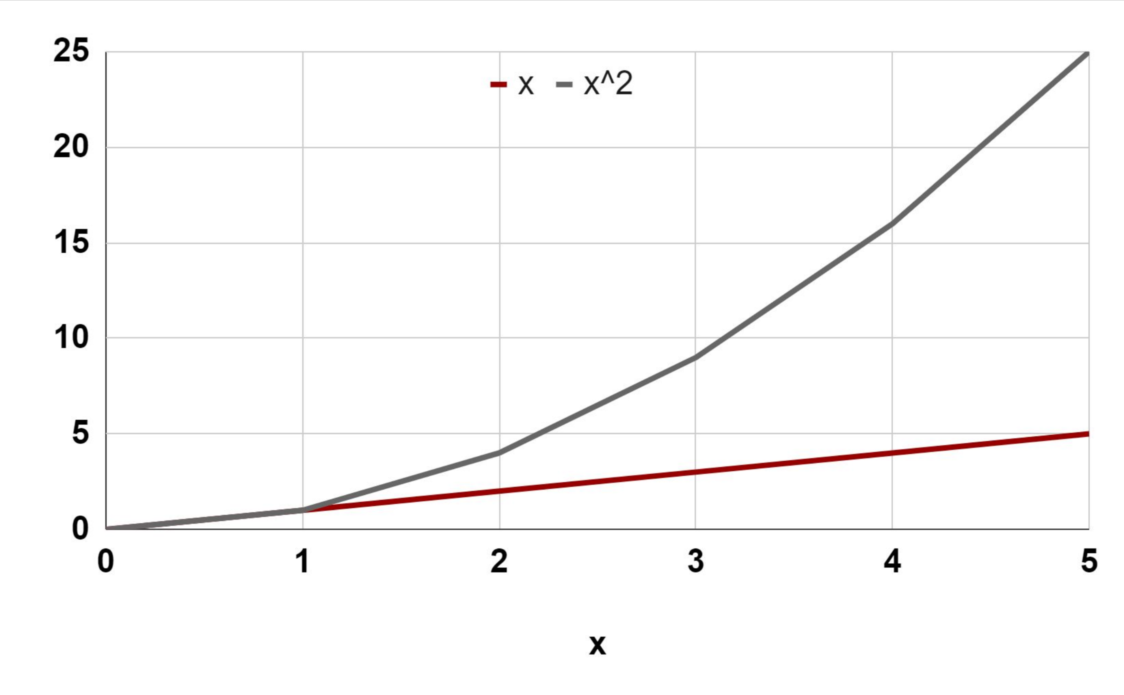
Ordem de Grandeza

SUMÁRIO

- Ordem de Grandeza
- Análise de Algoritmos
- O Teorema Mestre

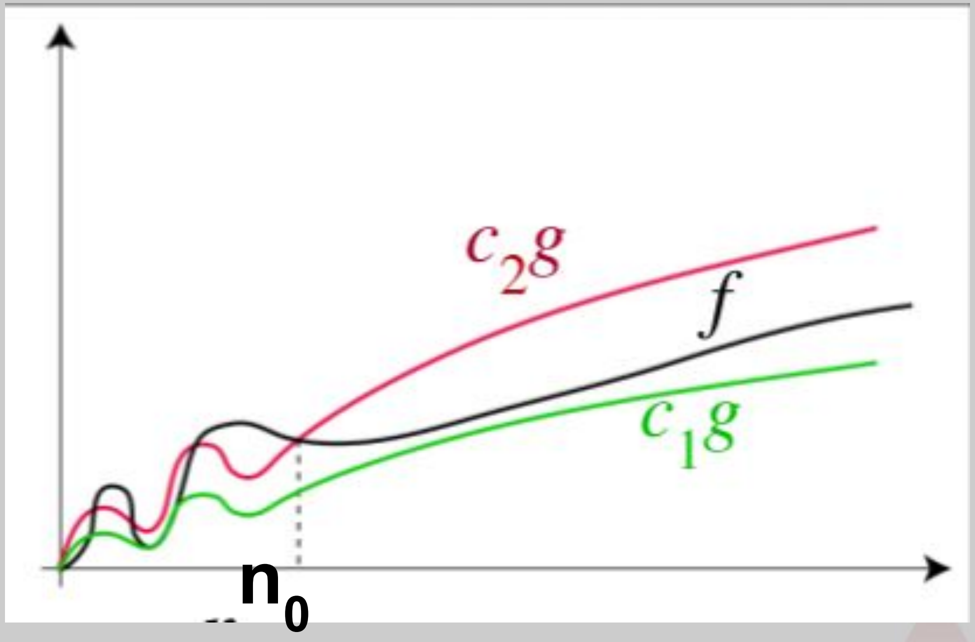
Ordem de Grandeza

A ordem de grandeza é uma maneira de comparar a "taxa de crescimento" de funções diferentes.



Funções

Sejam f e g funções em $\mathbb{R}^*_+ \rightarrow \mathbb{R}^*_+$ (reais não negativos).
A função f tem a mesma ordem de grandeza do que g, denotado por f = Θ(g), se existem constantes positivas n_0, c_1 e c_2 tais que, se $x \geq n_0$, então $c_1 g(x) \geq f(x) \geq c_2 g(x)$.



Ordem de Grandeza

Relação Binária $f \rho g$

$\leftrightarrow$

se existem constantes positivas n_0, c_1 e c_2 tais que para todo $x \geq n_0$, $c_1 g(x) \leq f(x) \leq c_2 g(x)$ ($f = \Theta(g)$)

- **A relação ρ é uma relação de equivalência.**
- **ρ estabelece classes de equivalência.**
- **$f = \Theta(g)$ quer dizer na verdade $f \in [g]$**

Ordem de Grandeza

Se $x \geq n_0$, então $c_1 g(x) \geq f(x) \geq c_2 g(x)$.

Seja $f(x) = 3x^2$ e $g(x) = 200x^2 + 140x + 7$.

$c_1(200x^2 + 140x + 7) \geq 3x^2 \geq c_2(200x^2 + 140x + 7)$

$(1)(200x^2 + 140x + 7) \geq 3x^2 \geq (1/100)(200x^2 + 140x + 7)$

$200x^2 + 140x + 7 \geq 3x^2 \geq 2x^2 + 1{,}4x + 0{,}07$

$c_1 = 1$ e $c_2 = 1/100$

Porém, $x=1$ $3(1)^2 \ngtr 2.(1)^2 + 1.4(1) + 0{,}07$ **Falso!!**

$x=2$ $3(2)^2 > 2.(2)^2 + 1.4(2) + 0{,}07$

Verdadeiro!

Logo, $x \geq 2 \Rightarrow n_0 = 2$ com $c_1 = 1$ e $c_2 = 1/100$

Ordem de Grandeza

Exemplo: Prove $f = \Theta(x^2)$ não ocorre para $f(x) = x$

Lembre-se: Se $x \geq n_0$, então $c_1 g(x) \geq f(x) \geq c_2 g(x)$.

Por absurdo, suponha que $f = \Theta(x^2)$ ocorra !

Temos $c_1 x^2 \geq x \Rightarrow c_1 x \geq 1$ **e**

$$x \geq c_2 x^2 \Rightarrow 1 \geq c_2 x$$

$$\Rightarrow c_1 x \geq 1 \geq c_2 x \quad \text{para } x \geq n_0$$

Porém, $1 \geq c_2 x \Rightarrow 1/c_2 \geq x$. **Absurdo!!**

Teremos x suficientemente grande tal que

$$x \geq 1/c_2$$

Análise de Algoritmo

- A **ordem de grandeza** é importante na **análise de algoritmos.**
- A **análise de algoritmo** identifica as **tarefas importantes** executadas por ele.
- O **número de vezes** que tais tarefas serão executadas geralmente depende do **tamanho dos dados de entrada**.
- As **funções** que expressam a quantidade de trabalho vão ter domínio $\mathbb{N}$.

Análise de Algoritmo

Exemplo: Ordenar 10 milhões de números.

Suponha que o computador A execute 1 bilhão de tarefas por segundo.
Um excelente programador implementa o algoritmo X para solucionar instâncias de tamanho n de um problema.
Esse programador utiliza linguagem de máquina no computador A e obtém uma performance de $2n^2$.

Análise de Algoritmo

Exemplo: Ordenar 10 milhões de números.

Um programador com nível mediano implementa o algoritmo Y para solucionar o problema. Ele utiliza linguagem C em um computador B que executa 10 milhões de tarefas por segundo. Essa implementação consegue solucionar as mesmas instâncias do problema com uma performance de 50nlogn.

Análise de Algoritmo

Exemplo: Ordenar 10 milhões de números ⇒ $n=10^7$

O computador A é 1000 vezes mais rápido do que o computador B

$$A: \frac{2.(10^7)^2}{10^9} = 20000 \ (>5{,}5h)$$

17 vezes superior

$$B: \frac{50.10^7(\log_2 10^7)}{10^7} \approx 1163 \ (<20 \min)$$

Análise de Algoritmo

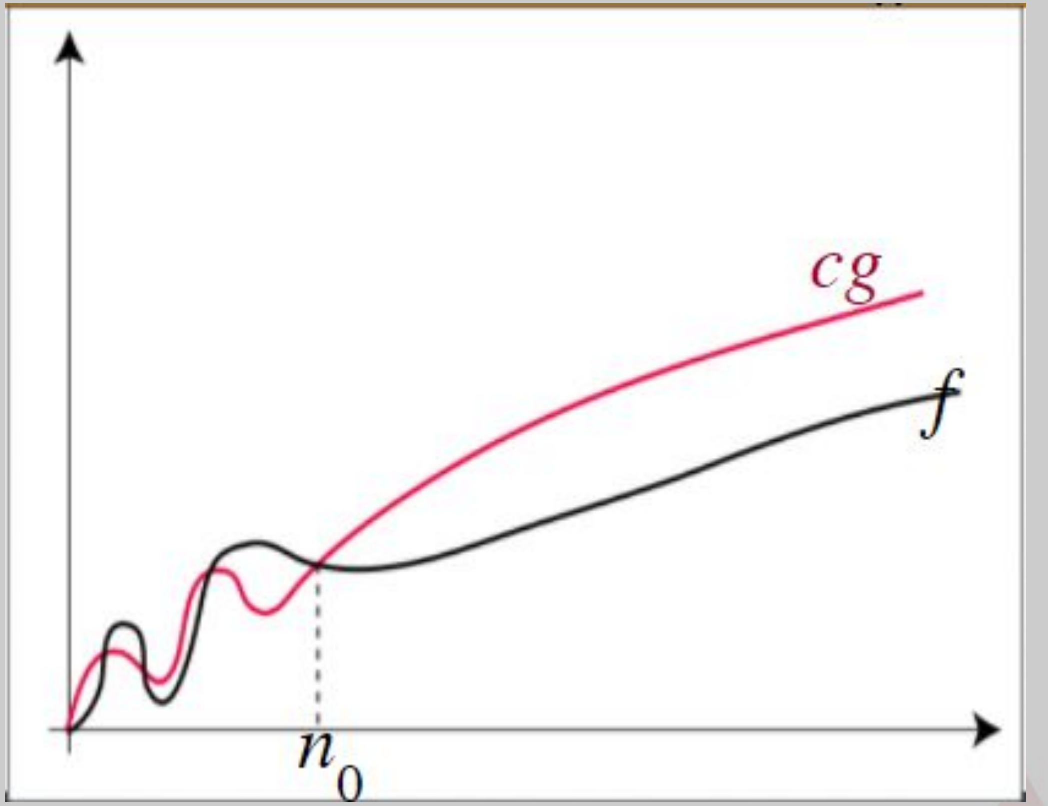
- Diferentes algoritmos que solucionam um mesmo problema podem apresentar uma diferença significativa em termos de eficiência.
- Tais diferenças podem ser mais impactantes que diferenças de hardware e software.
- A eficiência não recai sobre o tipo de linguagem utilizada para implementar o algoritmo, nem sobre o tipo de máquina na qual o algoritmo é executado.

Análise de Algoritmo

- A eficiência de um algoritmo pode ser definida pela sua complexidade de tempo.
- A complexidade de tempo é dada pelo número de instruções básicas que ele executa considerando o tamanho da entrada.

Análise de Algoritmo

Sejam f e g funções em $\mathbb{R}^*_+ \rightarrow \mathbb{R}^*_+$. A função f será O grande de g, denotado por f = O(g), se existem constantes positivas n_0 e c tais que, se $x \geq n_0$, então $f(x) \leq cg(x)$.



Análise de Algoritmo

- $f = O(g)$ diz que f cresce à mesma taxa ou a uma taxa menor do que g.
- Se soubermos que f cresce a uma taxa menor, podemos dizer que f é o pequeno de g, denotado por $f = o(g)$.
- A relação entre O grande e o pequeno é a seguinte: se $f = O(g)$, então $f = \Theta(g)$ ou $f = o(g)$.
- Isso é semelhante a dizer que para $a \leq b$ temos $a = b$ ou $a < b$.

Análise de Algoritmo

Exemplo:

```
int exp1(int a, int b) {
1.  if (b == 1)
2.      return a;
    else
3.      return a*exp1(a, b-1);
}
```

T(b): função de complexidade

b: número de vezes que teremos de multiplicar a base para obter a exponenciação.

Análise de Algoritmo

Exemplo:

```
int exp1(int a, int b) {
1. if (b == 1)                      1 Comparação
2.       return a;                  1 retorno
   else
3.       return a*exp1(a, b-1);     1 produto
}                                   1 chamada recursiva
                                      T(b-1)
```

$T(b)=T(b-1)+4$

Expandindo, supondo e verificando temos:

$$T(b)=4b-2 \text{ com } T(b)=\Theta(b)$$

Análise de Algoritmo

- Algoritmo polinomial é aquele que gasta um tempo limitado por um polinômio para solucionar instâncias do problema.

- Se existir um algoritmo polinomial para um problema computacional, o problema é classificado como um problema polinomial.

Análise de Algoritmo

- Problemas polinomiais são considerados tratáveis.

- Problemas para os quais não existe algoritmo polinomial são ditos intratáveis.

- Eles são lentos com tempo na ordem de $O(2^n)$, $O(3^n)$, $O(10^n)$

Teorema Mestre

Considere a relação de recorrência

$S(1) \geq 0$

$S(n) = aS(n/b) + n^c$

em que $n = b^m$, a e b são inteiros, $a \geq 1$, $b > 1$ e c é um número real não negativo. Então

Se $a < b^c \Rightarrow S(n) = \Theta(n^c)$

Se $a = b^c \Rightarrow S(n) = \Theta(n^c \log n)$

Se $a > b^c \Rightarrow S(n) = \Theta(n^{\log_b a})$

Teorema Mestre

Exemplo: $S(n)=4S(n/5)+n^3$ para $n \geq 2$ e $n=5^m$

Temos $a = 4$, $b = 5$ e $c = 3$ com

$a=4 < 5^3=b^c$ no caso 1 do teorema mestre.

Se $a < b^c \Rightarrow S(n)= \Theta(n^c)$

Logo, $S(n) = \Theta(n^3)$

Teorema Mestre

Exemplo: $C(n)=C(n/2)+1$ para $n \geq 2$ e $n=2^m$

Temos $a = 1$, $b = 2$ e $c = 0$ com

$a = 1 = 2^0=b^c$ no caso 2 do teorema mestre.

Se $a = b^c \Rightarrow S(n)= \Theta(n^c \log n)$

Logo, $C(n) = \Theta(n^0 \log n) = \Theta(\log n)$.

Os conceitos e exemplos apresentados nesses slides são baseados no conteúdo da seção 5.5 do material-base “Fundamentos Matemáticos para a Ciência da Computação”, J.L. Gersting, 7a edição, LTC editora.

FUNDAMENTOS MATEMÁTICOS PARA COMPUTAÇÃO

Ordem de Grandeza