

INE5403

FUNDAMENTOS DE MATEMÁTICA DISCRETA PARA A COMPUTAÇÃO

PROF. DANIEL S. FREITAS

UFSC - CTC - INE

5 - FUNÇÕES

5.1) Definições e Tipos

5.2) Crescimento de Funções

FUNÇÕES & COMPLEXIDADE

- Para cada método de solução de um problema (algoritmo), pode-se definir **funções** que relacionam:
 - o **espaço necessário** para guardar os dados
 - a **quantidade de passos** para resolvê-locom o **tamanho** (magnitude) dos dados de entrada.
- Estas funções permitem concluir sobre:
 - memória necessária para armazenar todos os dados
 - tempo de execução de algoritmos

FUNÇÕES & COMPLEXIDADE

- **Comparações de ordens de grandeza** destas funções equivalem a comparar os **custos computacionais** de diferentes métodos de solução de um problema.
- **Exemplo:** Solução de um sistema linear $n \times n$:

| método | nro de OPFs | tempo para $n = 30$ (*) |
|----------------------|-----------------|---------------------------|
| Regra de Cramer | $\sim (n + 1)!$ | 1.4×10^{12} anos |
| Eliminação gaussiana | $\sim n^3/3$ | 0.05×10^{-9} s |

(*) em um supercomputador BlueGene (~ 180 TFlops)

- A primeira função **cresce** muito mais rapidamente do que a 2a.
- Assunto relacionado com o tema **complexidade de algoritmos**.

CRESCIMENTO DE FUNÇÕES

● **Exemplo 1:** Considere o problema de determinar a transitividade de uma relação R sobre um conjunto A com n elementos.

● número de passos necessários (média) pelo método 1:

$$t(n) = \frac{1}{2}n^3 + \frac{1}{2}n^2$$

● número de passos necessários (média) pelo método 2:

$$s(n) = \frac{1}{8}n^4$$

● A tabela mostra que s “cresce mais rápido” do que t :

| n | $t(n)$ | $s(n)$ |
|-----|--------|----------|
| 2 | 6 | 2 |
| 5 | 75 | 78 |
| 10 | 550 | 1250 |
| 50 | 63750 | 781250 |
| 100 | 505000 | 12500000 |

NOTAÇÃO “BIG-O”

- Sejam f e g funções cujos domínios são subconjuntos de \mathbb{Z}^+ :
 - dizemos que f é $O(g)$ se existem constantes c e k tais que:

$$|f(n)| \leq c \cdot |g(n)|, \quad \forall n \geq k$$

- lê-se: f é “big-O” de g
- Ou seja: se f é $O(g)$, então f não cresce mais rápido do que g .
- Vantagem da notação big-O:
 - pode-se estimar o crescimento de uma função sem ligar para multiplicadores constantes ou termos de ordem menor
 - ou seja: usando notação big-O, não precisamos ligar para o hardware e o software usados para implementar um algoritmo.

NOTAÇÃO “BIG-O”

● **Exemplo 2:** A função $f(n) = \frac{1}{2}n^3 + \frac{1}{2}n^2$ é $O(g)$ para $g(n) = n^3$.

● Para ver isto, note que:

$$\frac{1}{2}n^3 + \frac{1}{2}n^2 \leq \frac{1}{2}n^3 + \frac{1}{2}n^3 \quad \text{se } n \geq 1$$

● Portanto:

$$\frac{1}{2}n^3 + \frac{1}{2}n^2 \leq 1 \cdot n^3 \quad \text{se } n \geq 1$$

● Daí, escolhendo $c = 1$ e $k = 1$, obtemos:

$$|f(n)| \leq |g(n)| \quad \forall n \geq 1$$

● O que mostra que f é $O(g)$

□

NOTAÇÃO “BIG-O”

- Note que são possíveis outras escolhas para c , k e até mesmo g .
- Note que, se $|f(n)| \leq c \cdot |g(n)|$, $\forall n \geq k$, então:
 - $|f(n)| \leq C \cdot |g(n)|$, $\forall n \geq k$, $\forall C \geq c$, e
 - $|f(n)| \leq c \cdot |g(n)|$, $\forall n \geq K$, $\forall K \geq k$
 - ou seja: quando existe um par de constantes, existem infinitos
- Agora seja novamente a função $t(n) = \frac{1}{2}n^3 + \frac{1}{2}n^2$:
 - t é $O(h)$ para $h(n) = dn^3$, se $d \geq 1$, pois:
 - $|t(n)| \leq 1 \cdot |g(n)| \leq |h(n)|$
 - Observe também que t é $O(r)$ para $r(n) = n^4$, pois:
 - $\frac{1}{2}n^3 + \frac{1}{2}n^2 \leq n^3 \leq n^4$, $\forall n \geq 1$

NOTAÇÃO “BIG-O”

- Ao analisar algoritmos, buscamos a função simples g de “crescimento mais lento” para a qual f é $O(g)$.
 - algumas vezes ela vem de um “conjunto de referência”
 - tal como as funções da forma x^n , para n dado
- É comum substituímos g em $O(g)$ pela fórmula que define g :
 - portanto, escrevemos que “ t é $O(n^3)$ ”
 - esta é a chamada notação “big-O”
- Ainda: dizemos que f e g possuem **mesma ordem** se:
 - f é $O(g)$ e
 - g é $O(f)$

NOTAÇÃO “BIG-O”

- **Exemplo 3:** As funções $f(n) = 3n^4 - 5n^2$ e $g(n) = n^4$, definidas para inteiros positivos n , possuem a mesma ordem.

- Primeiro, note que:

$$\begin{aligned} 3n^4 - 5n^2 &\leq 3n^4 + 5n^2 \\ &\leq 3n^4 + 5n^4, & \text{se } n \geq 1 \\ &= 8n^4. \end{aligned}$$

- daí, fazendo $c = 8$ e $k = 1$, temos $|f(n)| \leq c \cdot |g(n)|$, $\forall n \geq k$

- Conversamente:

$$n^4 = 3n^4 - 2n^4 \leq 3n^4 - 5n^2, \quad \text{se } n \geq 2$$

- isto ocorre porque, se $n \geq 2$, então: $2n^4 > 5n^2$

- daí, usando 1 para c e 2 para k , concluimos que g é $O(f)$.

NOTAÇÃO “BIG-O”

- Se f é $O(g)$ mas g não é $O(f)$, dizemos que:
 - f é de **ordem mais baixa** do que g ou que:
 - f **cresce mais lentamente** do que g

NOTAÇÃO “BIG-O”

● **Exemplo 4:** $f(n) = n^5$ é de ordem mais baixa do que $g(n) = n^7$.

● É claro que, se $n \geq 1$, então $n^5 \leq n^7$.

● Agora suponha que existam c e k tais que:

$$n^7 \leq cn^5, \quad \forall n \geq k$$

● então escolha um N tal que $N > k$ e $N^2 > c$

● daí: $N^7 \leq cN^5 < N^2 \cdot N^5$

● mas isto é uma contradição!

● Portanto, f é $O(g)$ mas g não é $O(f)$

● e f é de ordem mais baixa do que g

● o que, é claro, concorda com a idéia usual sobre n^5 e n^7

NOTAÇÃO “BIG-O”

- Com a ajuda da notação big-O, podemos determinar se é prático usar um certo algoritmo para resolver um problema à medida que o tamanho dos dados de entrada cresce.
- Exemplo:
 - temos dois algoritmos para resolver um problema:
 - um utiliza $100n^2 + 17n + 4$ operações
 - o outro utiliza n^3 operações
 - a notação big-O mostra que o primeiro usa muito menos operações quando n é grande
 - mas gasta menos operações para n pequeno
 - ($n = 10$, por exemplo)

NOTAÇÃO “BIG-O”

- Dica para encontrar as constantes:
 - primeiro, selecione um valor de k para o qual o tamanho de $|f(x)|$ pode ser prontamente estimado quando $x > k$
 - verificar se é possível encontrar um valor de C para o qual $|f(x)| < C|g(x)|$ para $x > k$.
 - abordagem ilustrada no exemplo a seguir
- Note que a notação big-O também funciona com funções definidas sobre os reais.

NOTAÇÃO “BIG-O”

● **Exemplo:** Mostre que $f(x) = x^2 + 2x + 1$ é $O(x^2)$

Solução:

● podemos prontamente estimar o tamanho de $f(x)$ quando $x > 1$:

● $x < x^2$ e $1 < x^2$ quando $x > 1$

● segue então que:

$$0 \leq x^2 + 2x + 1 \leq x^2 + 2x^2 + x^2 = 4x^2$$

● assim, fazendo $c = 4$ e $k = 1$, temos que $f(x)$ é $O(x^2)$, pois:

$$f(x) = x^2 + 2x + 1 < 4x^2, \text{ sempre que } x > 1$$

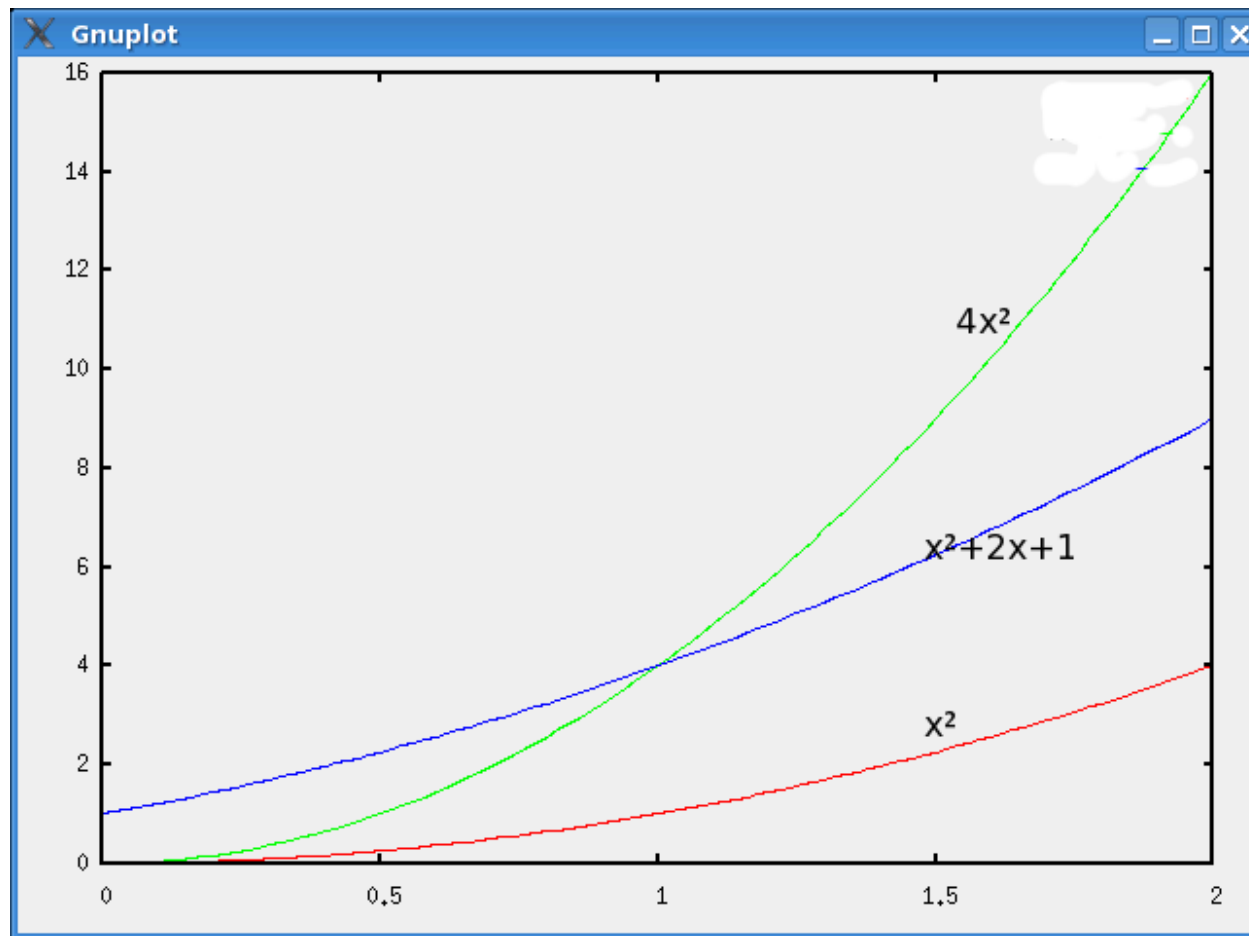
● note que $c = 3$ e $k = 2$ também serviriam, pois:

● se $x > 2$, temos que:

$$0 \leq x^2 + 2x + 1 \leq x^2 + x^2 + x^2 = 3x^2$$

NOTAÇÃO “BIG-O”

● **Exemplo (cont.):** $x^2 + 2x + 1 < 4x^2$ para $x > 1$:

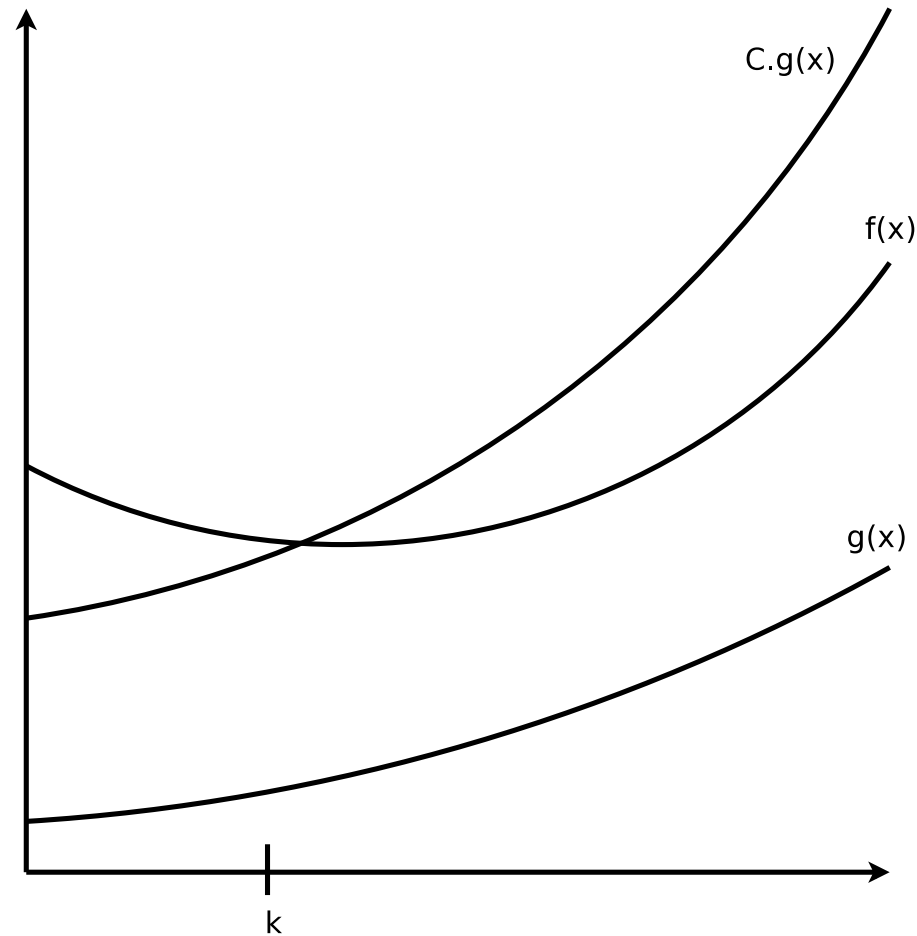


NOTAÇÃO “BIG-O”

- Observe que na relação “ $f(x)$ é $O(x)$ ”, x^2 pode ser trocada por qualquer função com valores maiores do que x^2 .
 - Exemplo:
 - $f(x)$ é $O(x^3)$
 - $f(x)$ é $O(x^2 + 2x + 1)$, etc.
- Note que $f(x) = x^2 + 2x + 1$ e $g(x) = x^2$ possuem **mesma ordem**.
- Note ainda que **não** é aceitável escrever: $f(x) = O(g(x))$
 - pois big-O significa apenas que existe uma desigualdade válida relacionando valores das funções f e g
 - para valores suficientemente grandes nos respectivos domínios
- No entanto, está correto dizer que: $f(x) \in O(g(x))$
 - pois $O(g(x))$ representa **o conjunto de todas as funções** que são $O(g(x))$

NOTAÇÃO “BIG-O”

● Ilustração de “ $f(x)$ é $O(g(x))$ ” (ou: $f(x) < c.g(x)$ para $x > k$)



NOTAÇÃO “BIG-O”

● **Exemplo:** Mostre que $7x^2$ é $O(x^3)$

Solução:

- Note que, quando $x > 7$, temos: $7x^2 < x^3$
 - (multiplicar ambos os lados de $x > 7$ por x^2)
- Logo, as constantes $c = 1$ e $k = 7$ mostram que $7x^2$ é $O(x^3)$
- Alternativamente:
 - quando $x > 1$, temos que $7x^2 < 7x^3$
 - de modo que $c = 7$ e $k = 1$ também servem

NOTAÇÃO “BIG-O”

- **Exemplo:** Mostre que n^2 **não é** $O(n)$

Solução:

- Temos que mostrar que nenhum par de constantes c e k satisfaz:

$$n^2 \leq cn, \text{ sempre que } n > k$$

- Para ver que as constantes não existem, note que, quando $n > 0$:

- pode-se dividir ambos os lados de $n^2 \leq cn$ por n

- para obter: $n \leq c$

- Note, então, que, não importa quem sejam c e k :

- a desigualdade $n \leq c$ não pode valer para todo n , com $n > k$

- note que, uma vez fixado um valor para k :

- quando n for maior do que o máximo de k e c ,
- não é verdade que $n \leq c$
- muito embora tenhamos $n > k$

NOTAÇÃO “BIG-O”

● **Exemplo:** Mostre que x^3 **não é** $O(7x^2)$

Solução:

● Temos que mostrar que nenhum par de constantes c e k satisfaz:

$$x^3 \leq c(7x^2), \text{ sempre que } n > k$$

● A desigualdade $x^3 \leq c(7x^2)$ é equivalente a: $x \leq 7c$

● Note que não existe c para o qual $x \leq 7c$ para todo $x > k$

● não importa quem seja k , pois x pode ser tornado tão grande quanto se queira

● Segue que não existem c e k para os quais exista a relação proposta.

RESULTADOS “BIG-O” IMPORTANTES

- É comum o uso de polinômios para **estimar** o crescimento de funções.
- O teorema a seguir mostra que o termo principal de um polinômio domina o seu crescimento.
- **Teorema 1:** Seja $f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$, aonde $a_0, a_1, \dots, a_{n-1}, a_n$ são números reais. Então $f(x)$ é $O(x^n)$.

RESULTADOS “BIG-O” IMPORTANTES

- **Exemplo 1(/3):** Use notação big-O para estimar a quantidade de operações envolvida na soma dos primeiros n inteiros positivos.

Solução:

- Como cada inteiro da soma é $< n$, segue que:

$$1 + 2 + \cdots + n \leq n + n + \cdots + n = n^2$$

- Então, tomando-se $c = 1$ e $k = 1$, concluimos que:

$$1 + 2 + \cdots + n \text{ é } O(n^2) \quad \square$$

RESULTADOS “BIG-O” IMPORTANTES

- **Exemplo 2(/3):** Forneça estimativas big-O para a função fatorial e para o seu logaritmo.
 - Nota: a função fatorial é definida por: $n! = 1 \cdot 2 \cdot 3 \cdots n$, ($0! = 1$)
 - Note que a função fatorial cresce rapidamente:
 $1! = 1, \quad 2! = 2, \quad 3! = 6, \quad 4! = 24, \quad \dots,$
 $20! = 2.432.902.008.176.640.000$

Solução:

- Note que cada termo no produto não excede n .
- Portanto:
$$\begin{aligned} n! &= 1 \cdot 2 \cdot 3 \cdots n \\ &\leq n \cdot n \cdot n \cdots n \\ &= n^n \end{aligned}$$
 - o que mostra que $n!$ é $O(n^n)$ (tomando $c = 1$ e $k = 1$)

RESULTADOS “BIG-O” IMPORTANTES

● **Exemplo 2 (cont.):** Estimativa big-O para o log da função fatorial:

● Tomando log de ambos os lados, obtemos:

$$\log n! \leq \log n^n = n \cdot \log n$$

● o que significa que:

$$\log n! \text{ é } O(n \cdot \log n) \text{ (tomando } c = 1 \text{ e } k = 1)$$

RESULTADOS “BIG-O” IMPORTANTES

- **Exemplo 3(/3):** No cap sobre indução veremos que, para $n \in \mathbb{Z}^+$:

$$n < 2^n$$

- Isto permite concluir que: n é $O(2^n)$ ($k = 1, c = 1$)
- Como o logaritmo é crescente, podemos tomar log desta desigualdade:

$$\log n < n$$

- Segue que $\log n$ é $O(n)$ ($k = 1, c = 1$)

NOTAÇÃO “BIG-O”

- (rel.) A notação big-O é usada para estimar o número de operações necessárias para resolver um problema usando um procedimento ou algoritmo específico.

- As funções usadas nestas estimativas comumente incluem:

$$1, \log n, n \log n, n^2, 2^n, n!$$

- Usando Cálculo, pode-se mostrar que cada função nesta lista é menor do que a seguinte.
- Ou seja: a relação entre cada função e sua sucessora tende a zero à medida que n cresce.

COMBINAÇÃO DE FUNÇÕES

- Muitos algoritmos são compostos de dois ou mais subprocedimentos.
 - Neste caso, a quantidade total de passos é a soma dos passos dos subprocedimentos.
 - Estimativas big-O para o total envolvem, portanto, a combinação das sub-estimativas.

COMBINAÇÃO DE FUNÇÕES

- Estimativas de combinações de funções exigem cuidado quando estimativas **diferentes** são combinadas.
- É frequentemente necessário estimar o crescimento de **somas** e de **produtos** de duas funções.

BIG-O DE SOMAS DE FUNÇÕES

● Suponha que:

● f_1 seja $O(g_1(x))$: existem constantes c_1 e k_1 tais que:

$$|f_1(x)| \leq c_1 |g_1(x)|, \text{ quando } x > k_1$$

● f_2 seja $O(g_2(x))$: existem constantes c_2 e k_2 tais que:

$$|f_2(x)| \leq c_2 |g_2(x)|, \text{ quando } x > k_2$$

● Estimativa para a soma de $f_1(x)$ e $f_2(x)$:

$$\begin{aligned} |(f_1 + f_2)(x)| &= |f_1(x) + f_2(x)| \\ &\leq |f_1(x)| + |f_2(x)| \end{aligned}$$

● em que foi usada a desigualdade triangular $|a + b| \leq |a| + |b|$

BIG-O DE SOMAS DE FUNÇÕES

- Então, quando x é maior do que k_1 e k_2 , temos:

$$\begin{aligned}|f_1(x)| + |f_2(x)| &\leq c_1|g_1(x)| + c_2|g_2(x)| \\ &\leq c_1|g(x)| + c_2|g(x)| \\ &= (c_1 + c_2)|g(x)| \\ &= c|g(x)|\end{aligned}$$

- onde: $c = c_1 + c_2$

- e: $|g(x)| = \max(|g_1(x)|, |g_2(x)|)$

- o que mostra que: $|(f_1 + f_2)(x)| \leq c \cdot |g(x)|$

- sempre que $x > k$

- aonde $k = \max(k_1, k_2)$

BIG-O DE SOMAS DE FUNÇÕES

● O raciocínio anterior demonstra o seguinte teorema:

● **Teorema 2:** suponha que:

● $f_1(x)$ é $O(g_1(x))$

● $f_2(x)$ é $O(g_2(x))$.

Então:

$$(f_1 + f_2)(x) \text{ é } O(\max(|g_1(x)|, |g_2(x)|))$$

● Se, por acaso, tivermos estimativas para f_1 e f_2 em termos da mesma função g , vale o seguinte:

● **Corolário:** Suponha que $f_1(x)$ e $f_2(x)$ são ambas $O(g(x))$. Então $(f_1 + f_2)(x)$ é $O(g(x))$.

BIG-O DE PRODUTOS DE FUNÇÕES

- De modo similar, quando x é maior do que $\max(k_1, k_2)$, segue que:

$$\begin{aligned} |(f_1 f_2)(x)| &= |f_1(x)| |f_2(x)| \\ &\leq c_1 |g_1(x)| \cdot c_2 |g_2(x)| \\ &\leq c_1 c_2 |(g_1 g_2)(x)| \\ &\leq c |(g_1 g_2)(x)| \end{aligned}$$

- onde: $c = c_1 c_2$

- o que mostra que $f_1(x)f_2(x)$ é $O(g_1 g_2)$, pois:

- existem constantes c e k (ou seja: $c = c_1 c_2$ e $k = \max(k_1, k_2)$)

- tais que: $|(f_1 f_2)(x)| \leq c |g_1(x) g_2(x)|$, sempre que $x > k$

BIG-O DE PRODUTOS DE FUNÇÕES

● O raciocínio anterior demonstra o seguinte teorema:

● **Teorema 3:** suponha que:

● $f_1(x)$ é $O(g_1(x))$

● $f_2(x)$ é $O(g_2(x))$.

Então:

$$(f_1 f_2)(x) \text{ é } O(g_1(x)g_2(x)).$$

ESTIMATIVAS BIG-O

- Objetivo de usar a notação big-O:
 - escolher uma função $g(x)$ que cresça lentamente o suficiente para que $f(x)$ seja $O(g(x))$.
- Os exemplos a seguir ilustram como usar os dois teoremas anteriores para fazer isto.
- Nota: este é um tipo de análise frequentemente usado na análise de tempo necessário para resolver um problema com programas computacionais.

ESTIMATIVAS BIG-O

- **Exemplo:** Forneça uma estimativa big-O para

$$f(n) = 3n \log(n!) + (n^2 + 3) \log n \quad (n \in \mathbb{Z}^+)$$

Solução:

- estimando o produto $3n \log(n!)$:
 - sabemos que: $\log(n!)$ é $O(n \log n)$
 - além disto: $3n$ é $O(n)$
 - o teorema 2, então, fornece a estimativa:

$$3n \log(n!) \text{ é } O(n^2 \log n)$$

ESTIMATIVAS BIG-O

- **Exemplo:** Forneça uma estimativa big-O para

$$f(n) = 3n \log(n!) + (n^2 + 3) \log n \quad (n \in \mathbb{Z}^+)$$

Solução (cont.):

- Vimos que: $3n \log(n!)$ é $O(n^2 \log n)$
- Estimando o produto $(n^2 + 3) \log n$:
 - uma vez que $(n^2 + 3) < 2n^2$ quando $n > 2$, segue que:
$$n^2 + 3 \text{ é } O(n^2)$$
 - logo, do teorema 3 segue que:
$$(n^2 + 3) \log n \text{ é } O(n^2 \log n)$$
- Com o teorema 2 novamente, temos que:

$$f(n) = 3n \log(n!) + (n^2 + 3) \log n \text{ é } O(n^2 \log n) \quad \square$$

ESTIMATIVAS BIG-O

- **Exemplo:** Forneça uma estimativa big-O para:

$$f(x) = (x + 1) \log(x^2 + 1) + 3x^2$$

Solução:

- estimativa big-O para $(x + 1) \log(x^2 + 1)$:

- $(x + 1)$ é $O(x)$

- além disto: $x^2 + 1 \leq 2x^2$ quando $x > 1$

- portanto, se $x \geq 2$, podemos escrever:

$$\log(x^2 + 1) \leq \log(2x^2) = \log 2 + 2\log x \leq 3\log x$$

- isto mostra que:

$$\log(x^2 + 1) \text{ é } O(\log x)$$

- então, o teorema 3 mostra que:

$$(x + 1) \log(x^2 + 1) \text{ é } O(x \log x)$$

□

ESTIMATIVAS BIG-O

- **Exemplo:** Forneça uma estimativa big-O para:

$$f(x) = (x + 1) \log(x^2 + 1) + 3x^2$$

Solução (cont.):

- vimos que: $(x + 1) \log(x^2 + 1)$ é $O(x \log x)$
- daí, uma vez que $3x^2$ é $O(x^2)$, temos, pelo teorema 2, que:
$$f(x) \text{ é } O(\max(x \log x, x^2))$$
- já que $x \log x \leq x^2$, para $x > 1$, temos: $f(x)$ é $O(x^2)$. □

CRESCIMENTO DE FUNÇÕES

- Final deste item.
- **Dica:** fazer **exercícios** sobre Crescimento de funções...