Complexidade Assintótica

Professores:

Norton T. Roman

Fátima L. S. Nunes

SISTEMAS DE INFORMAÇÃO

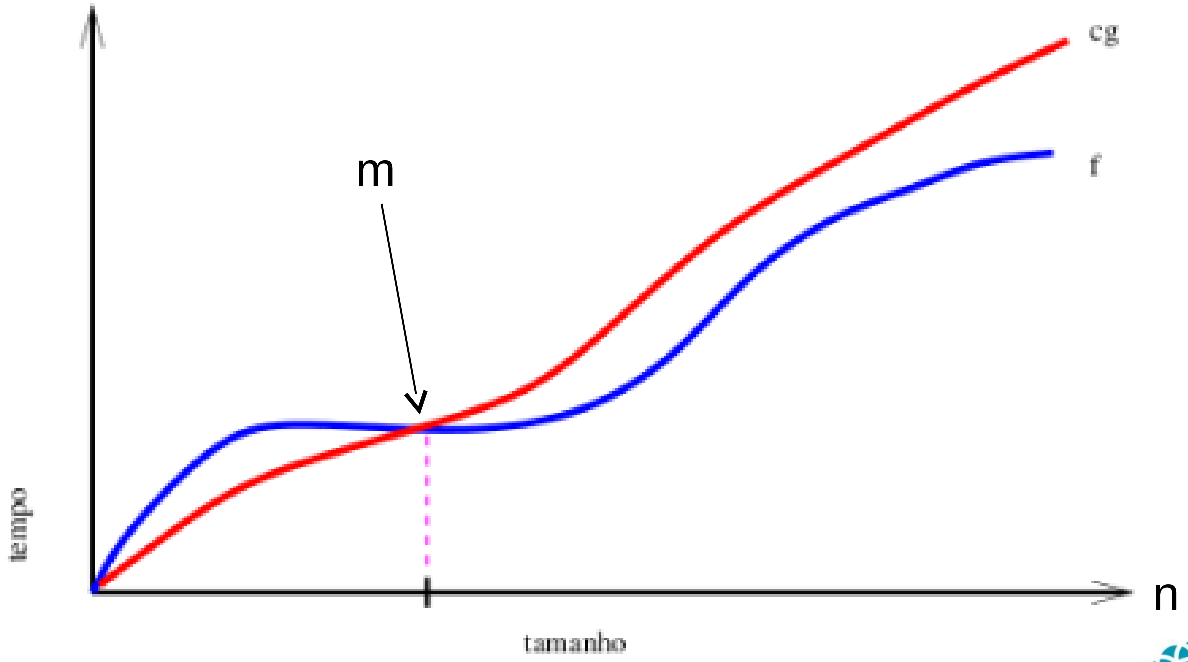
EACH

Comportamento Assintótico – Resumindo...

- Se ***f(n)*** é a função de complexidade de um algoritmo A
  - O comportamento assintótico de f (n) representa o limite do comportamento do custo (complexidade) de A quando ***n*** cresce.
- A análise de um algoritmo (função de complexidade)
  - Geralmente considera apenas algumas operações elementares ou mesmo uma operação elementar (e.g., o número de comparações).
- A complexidade assintótica relata crescimento assintótico das operações elementares.

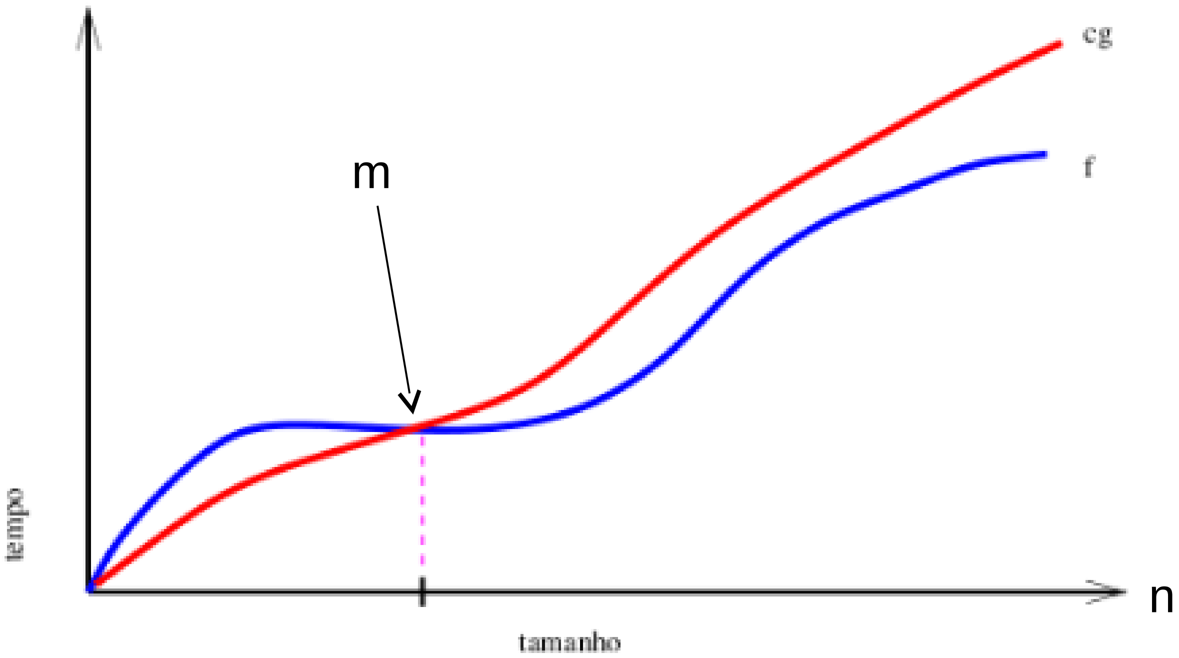
Relacionamento Assintótico

- Definição:
  - Uma função ***g(n)*** domina assintoticamente outra função ***f(n)*** se existem duas constantes positivas ***c*** e ***m*** tais que, para $n \geq m$, tem-se $|f(n)| \leq c \cdot |g(n)|$.



Relacionamento Assintótico

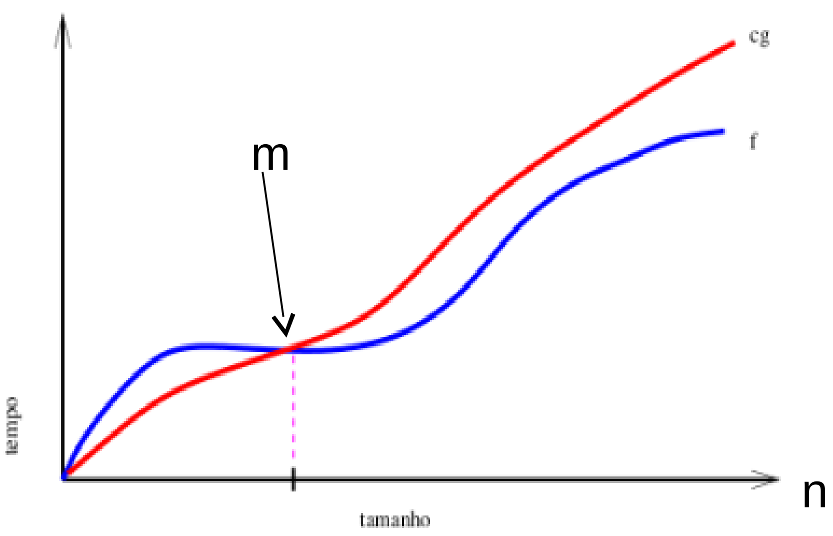
- Definição:
  - Uma função ***g(n)*** domina assintoticamente outra função ***f(n)*** se existem duas constantes positivas ***c*** e ***m*** tais que, para $n \geq m$, tem-se $|f(n)| \leq c \cdot |g(n)|$.



- Exemplo:
- $g(n) = n$ e $f(n) = n^2$
- Alguém domina alguém?
  - ???

Relacionamento Assintótico

- Definição:
  - Uma função ***g(n)*** domina assintoticamente outra função ***f(n)*** se existem duas constantes positivas ***c*** e ***m*** tais que, para $n \geq m$, tem-se $|f(n)| \leq c \cdot |g(n)|$.



- Exemplo:
- $g(n) = n$ e $f(n) = n^2$
- Alguém domina alguém?
  - $|n| \leq |n^2|$ para todo $n \in N$
- Para $c = 1$ e $m = 1 \Rightarrow |g(n)| \leq |f(n)|$
- Portanto, f (n) domina assintoticamente g(n).

Relacionamento Assintótico

- $g(n) = n$ e $f(n) = -n^2$
- Alguém domina alguém?
  - ???

Relacionamento Assintótico

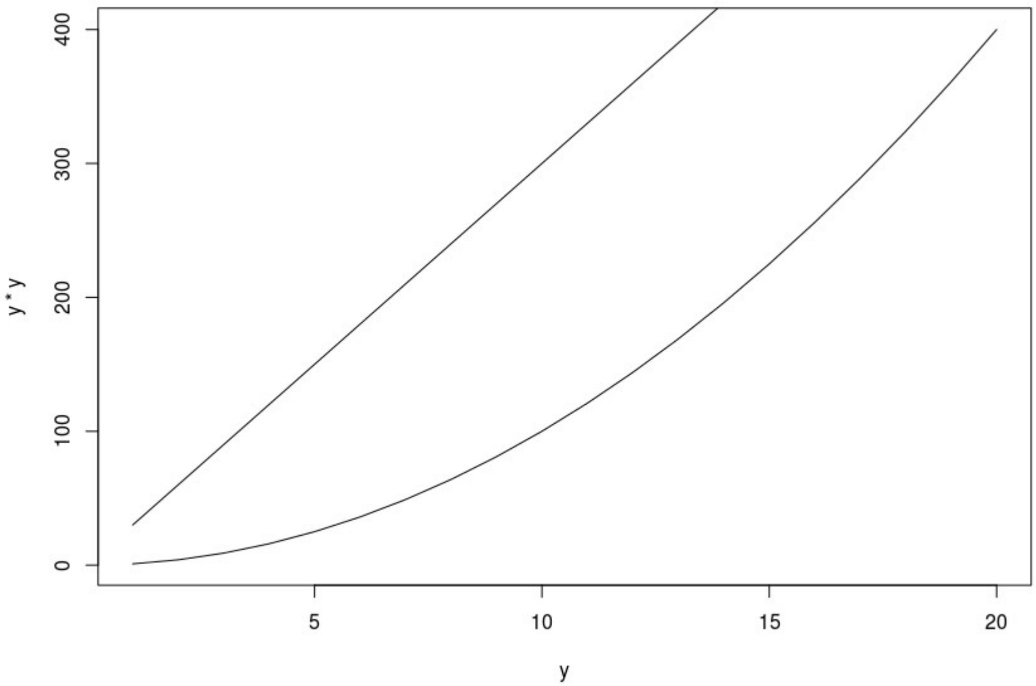
- $g(n) = n$ e $f(n) = -n^2$
- Alguém domina alguém?
  - $|n| \leq |-n^2|$ para todo $n \in N$.
    - Por ser módulo, o sinal não importa
  - Para $c = 1$ e $m = 1 \Rightarrow |g(n)| \leq |f(n)|$.
- Portanto, f (n) domina assintoticamente g(n).

Relacionamento Assintótico

- $g(n) = 30n$ e $f(n) = n^2$
- Alguém domina alguém?
  - ???

Relacionamento Assintótico

- $g(n) = 30n$ e $f(n) = n^2$
- Alguém domina alguém?

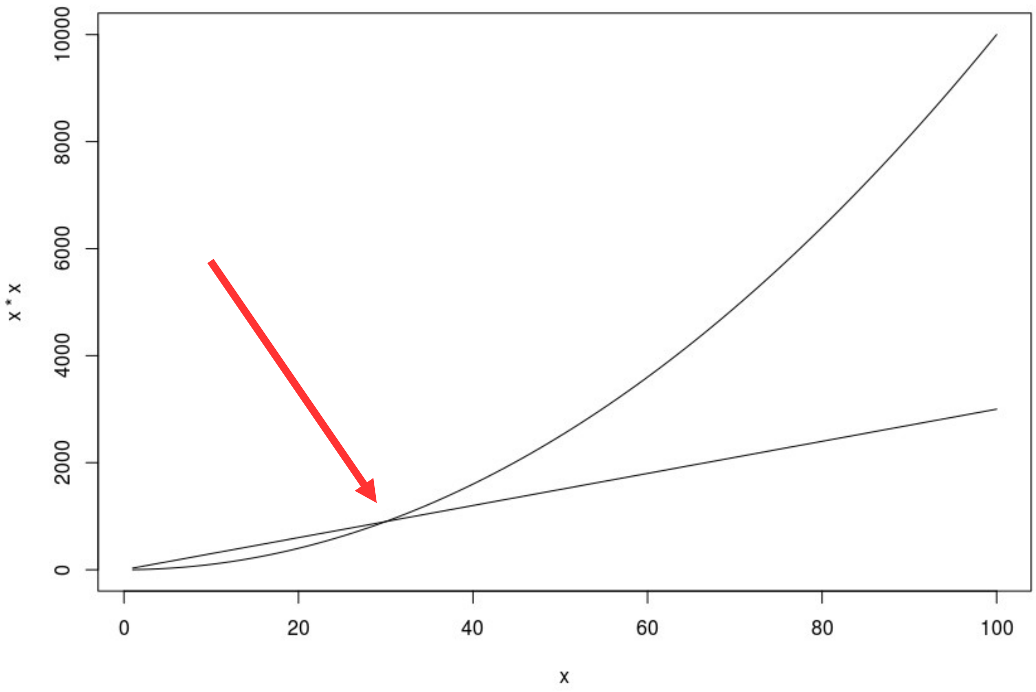


SISTEMAS DE INFORMAÇÃO

EACH

Relacionamento Assintótico

- $g(n) = 30n$ e $f(n) = n^2$
- Alguém domina alguém?

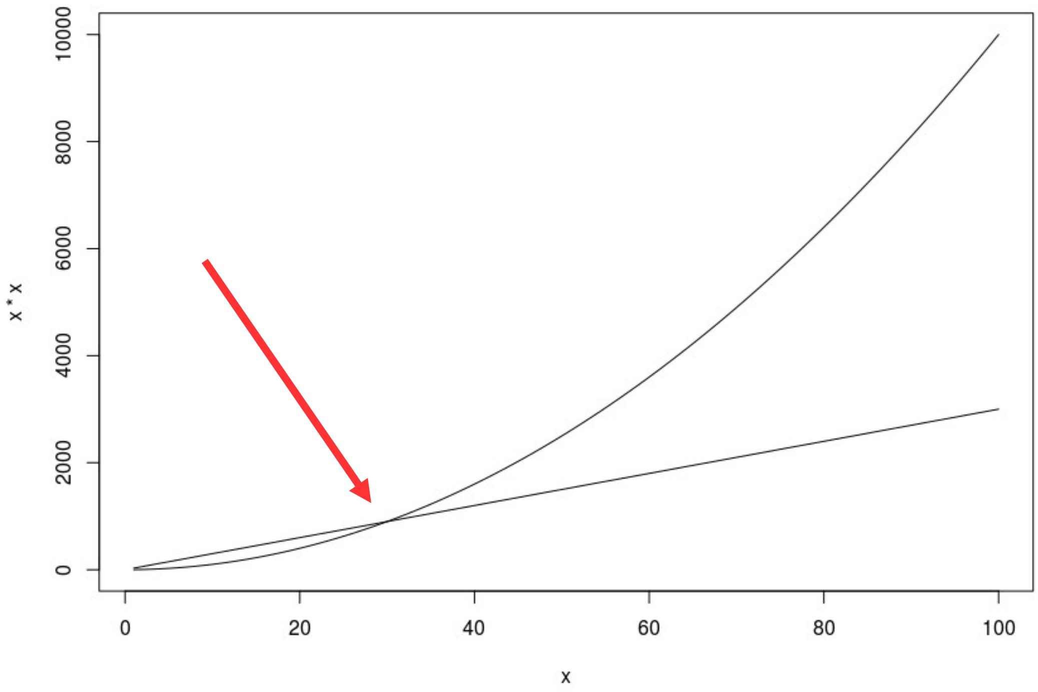


SISTEMAS DE INFORMAÇÃO

EACH

Relacionamento Assintótico

- $g(n) = 30n$ e $f(n) = n^2$
- Alguém domina alguém?



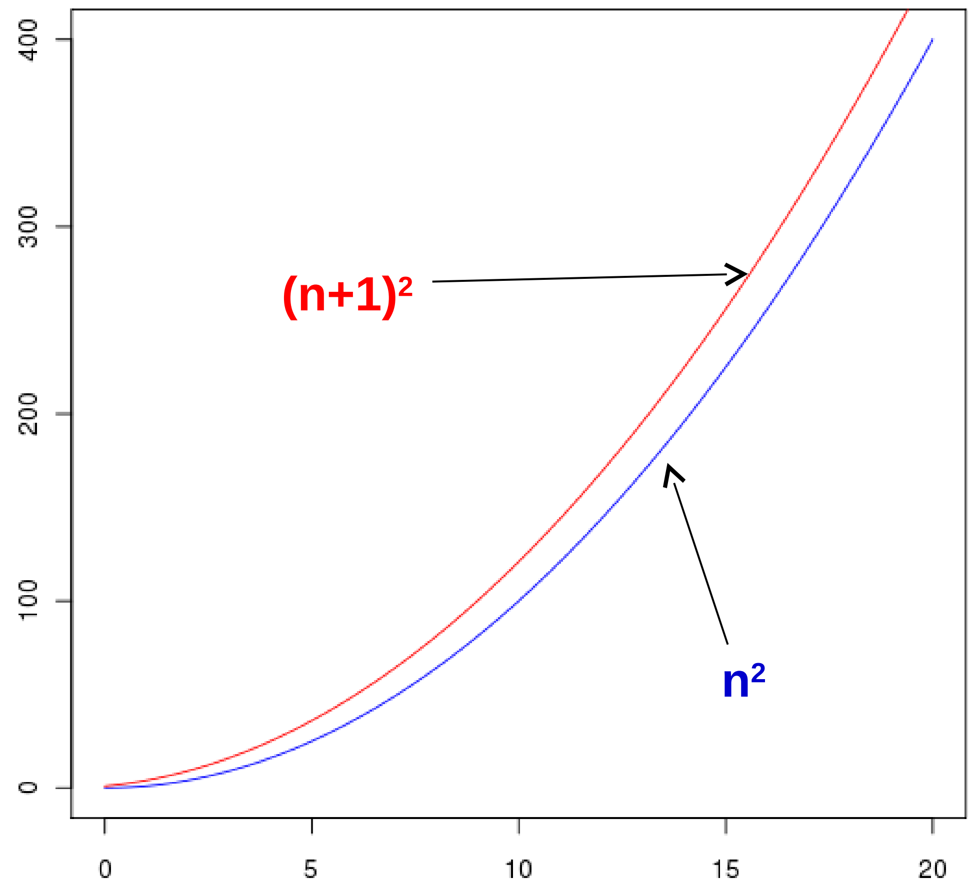
m = 30
c = 1
$|n^2| \geq |30n|$

Relacionamento Assintótico

- $g(n) = (n+1)^2$ e $f(n) = n^2$
- Alguém domina alguém?
  - ???

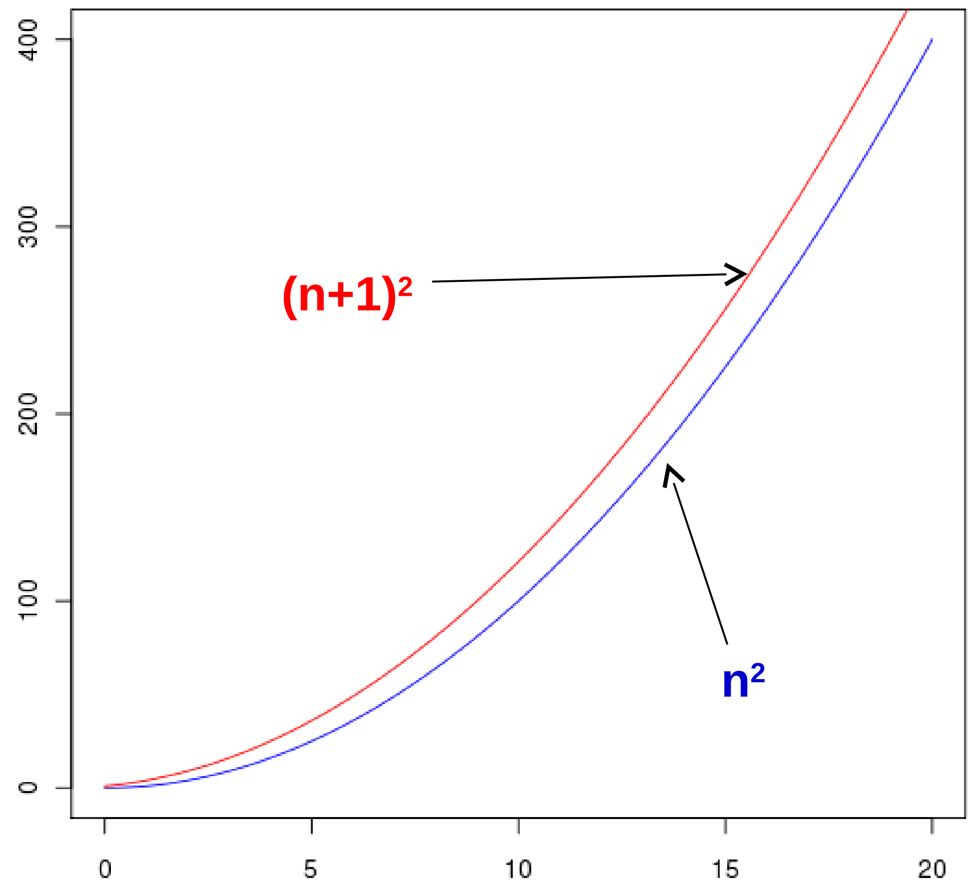
Relacionamento Assintótico

- $g(n) = (n+1)^2$ e $f(n) = n^2$
- Alguém domina alguém?
  - Vamos colocar em um gráfico



Relacionamento Assintótico

- $g(n) = (n+1)^2$ e $f(n) = n^2$
- Alguém domina alguém?
  - Vamos colocar em um gráfico
  - Para m = 0, c = 1
  - $|n^2| \leq |(n+1)^2|$, para $n \geq 0$
  - g(n) domina f(n)



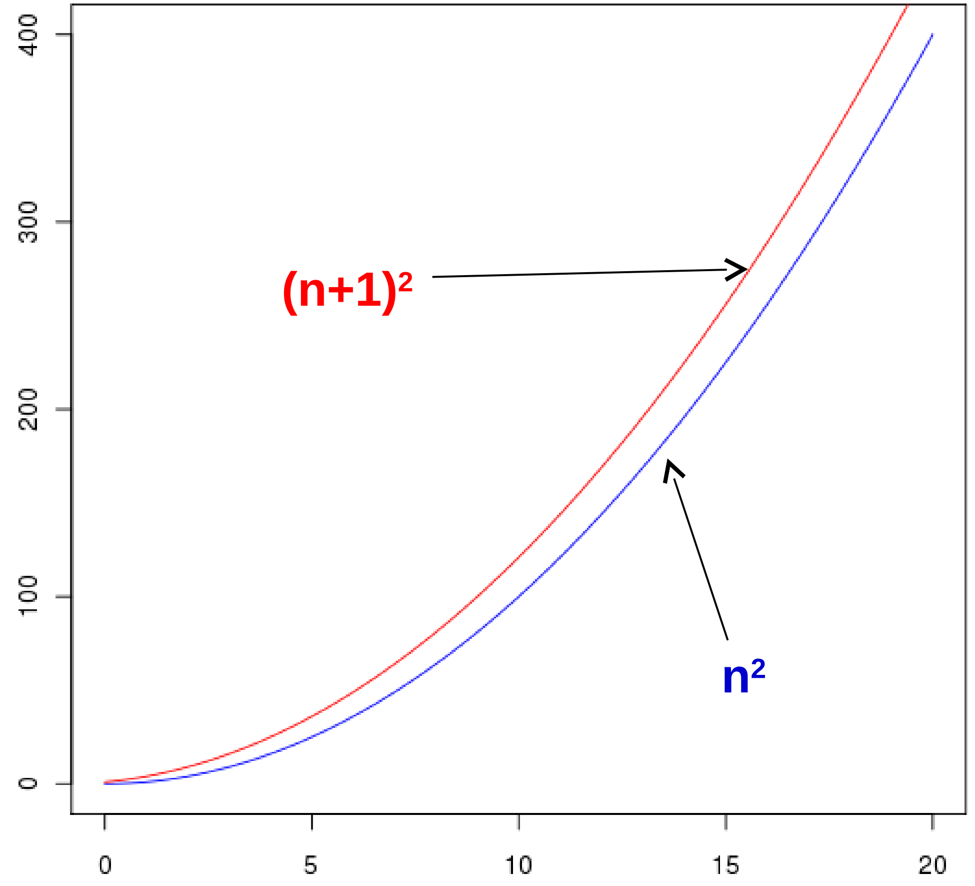
SISTEMAS DE INFORMAÇÃO

EACH

Relacionamento Assintótico

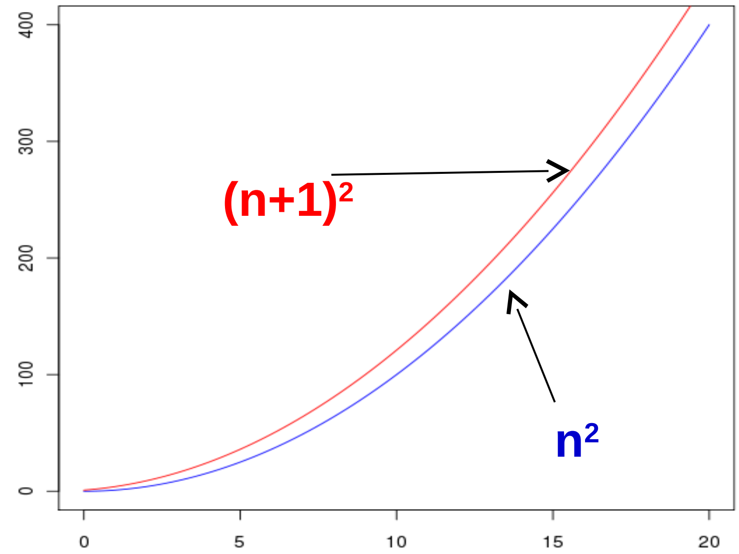
- $g(n) = (n+1)^2$ e $f(n) = n^2$
- Alguém domina alguém?
  - Vamos colocar em um gráfico
  - Para $m = 0$, $c = 1$
  - $|n^2| \leq |(n+1)^2|$, para $n \geq 0$
  - g(n) domina f(n)

Que mais????



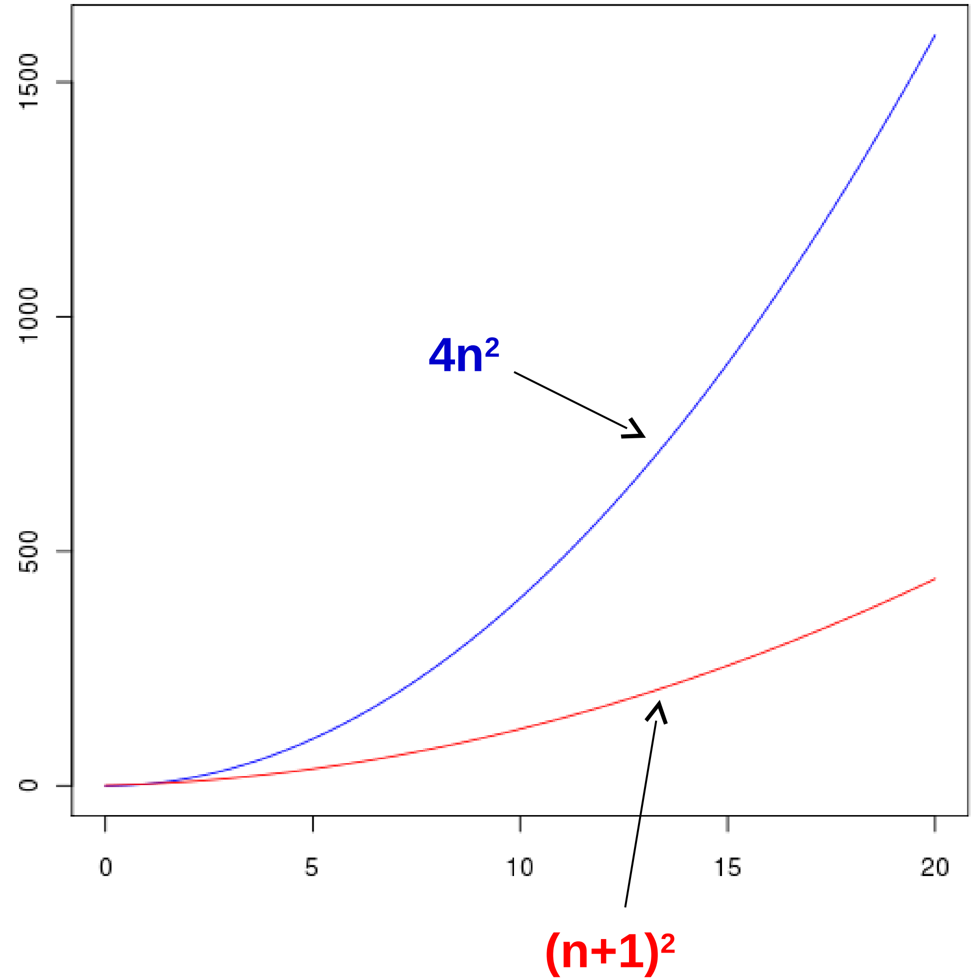
Relacionamento Assintótico

- $g(n) = (n+1)^2$ e $f(n) = n^2$
- Alguém domina alguém?
  - Não há como f(n) dominar g(n)?
    - Lembre que a definição envolve também uma constante.
    - Suponha que queremos $g(n) \leq cf(n)$
    - Então $|(n+1)^2| \leq |cn^2|$
    - Mas, para isso, basta que $|(n+1)^2| \leq |(\sqrt{c}\, n)^2|$,
      - ou $|n+1| \leq |\sqrt{c}\, n|$
    - Se $\sqrt{c} = 2$, ou seja, $c=4$, isso é verdade, para $n \geq 1$



Relacionamento Assintótico

- $g(n) = (n+1)^2$ e $f(n) = n^2$
- Alguém domina alguém?
  - $|(n+1)^2| \leq |4n^2|$, para $n \geq 1$
  - f(n) domina g(n), para $n \geq 1$
- Nesse caso, dizemos que f(n) e g(n) dominam assintoticamente uma a outra.



Notação *O*

- Knuth(1971) * criou a notação ***O*** (lê-se "O grande") para expressar que ***g(n)*** domina assintoticamente ***f(n)***
  - Escreve-se ***f(n) = O(g(n))*** e lê-se: "***f(n) é da ordem no máximo g(n)***".
- Para que serve isto para o Engenheiro de Computação?

---

*Knuth, D.E. (1971) "Mathematical Analysis of Algorithms". *Proceedings IFIP Congress 71, vol. 1, North Holland, Amsterdam, Holanda, 135-143.*

Notação O

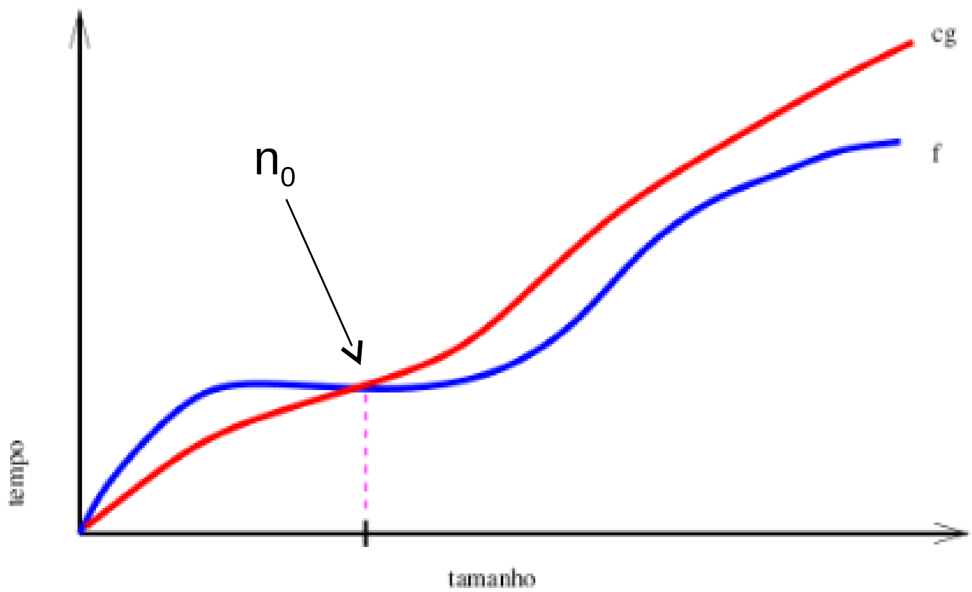
- Knuth(1971) * criou a notação ***O*** (lê-se "O grande") para expressar que ***g(n)*** domina assintoticamente ***f(n)***
  - Escreve-se ***f(n) = O(g(n))*** e lê-se: "***f(n) é da ordem no máximo g(n)***".
- Para que serve isto ?
  - Muitas vezes calcular a função de complexidade **g(n)** de um algoritmo A é complicado.
  - É mais fácil determinar que ***f (n)*** é ***O(g(n))***, isto é, que assintoticamente ***f(n)*** cresce no máximo como ***g(n)***.

---

*Knuth, D.E. (1971) "Mathematical Analysis of Algorithms". *Proceedings IFIP Congress 71, vol. 1, North Holland, Amsterdam, Holanda, 135-143.*

Notação *O*

- Definição:
  - $O(g(n)) = \{f(n)$: existem constantes positivas $\boldsymbol{c}$ e $\boldsymbol{n_0}$ tais que $0 \leq f(n) \leq cg(n)$, para todo $\boldsymbol{n} \geq \boldsymbol{n_0}\}$
- Informalmente, dizemos que, se $f(n) \in O(g(n))$, então f(n) cresce no máximo tão rapidamente quanto g(n).



Notação *O*

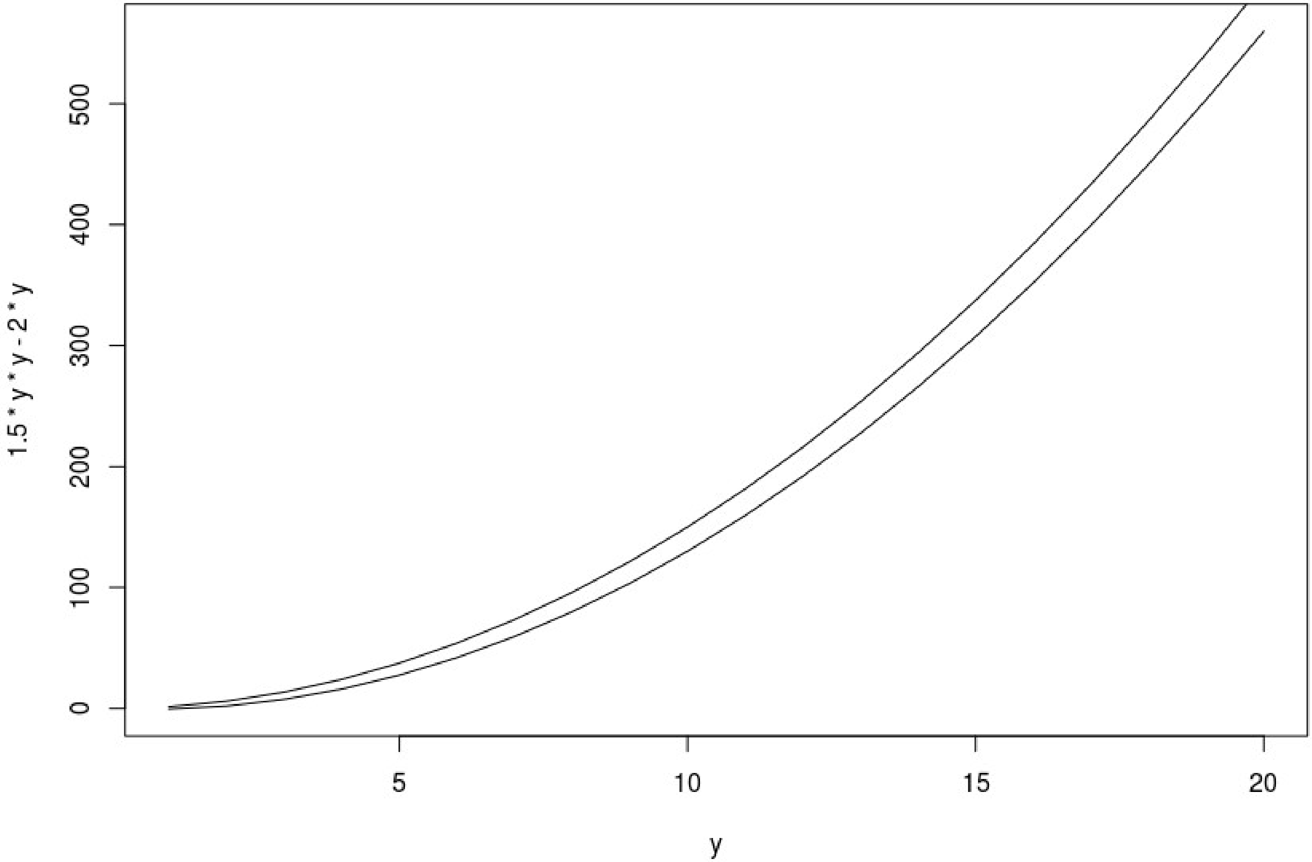
- Definição:
  - $O(g(n)) = \{f(n)$: existem constantes positivas $\boldsymbol{c}$ e $\boldsymbol{n_0}$ tais que $0 \leq f(n) \leq cg(n)$, para todo $\boldsymbol{n} \geq \boldsymbol{n_0}\}$
  - $\frac{3}{2}n^2 - 2n \in O(n^2)$?
    - ???

Notação *O*

- Definição:
  - O(g(n)) = {f(n): existem constantes positivas $\boldsymbol{c}$ e $\boldsymbol{n_0}$ tais que $0 \leq f(n) \leq cg(n)$, para todo $\boldsymbol{n} \geq \boldsymbol{n_0}$}
  - $\frac{3}{2}n^2 - 2n \in O(n^2)$?
    - Fazendo c = 3/2, teremos $\frac{3}{2}n^2 - 2n \leq \frac{3}{2}n^2$, para $n_0 \geq 0$
    - Outras constantes podem existir, mas o que importa é que <u>existe alguma escolha para as constantes</u>

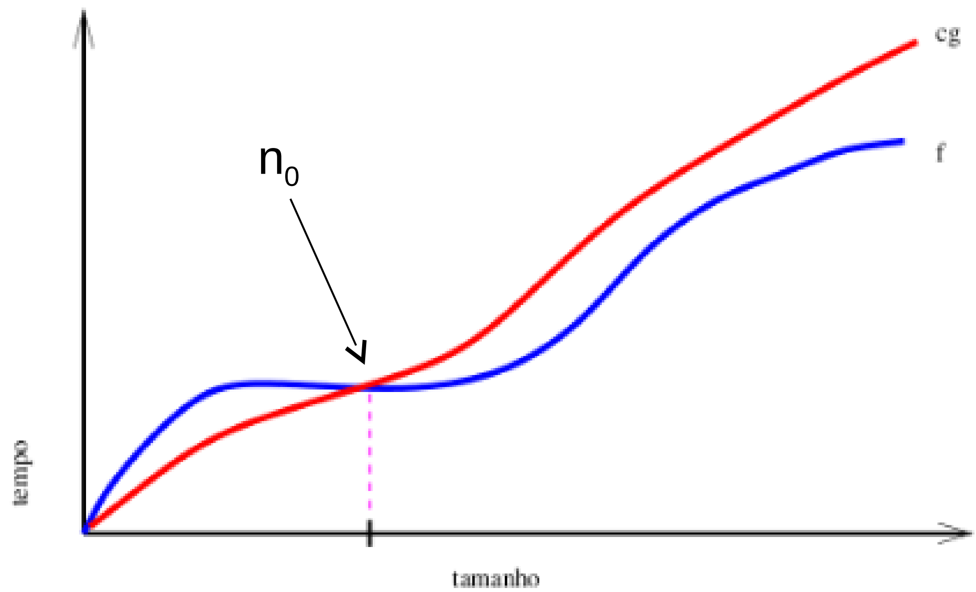
Notação *O*

- Defin
  - O(g ... n_0 tais que $0 \leq$
  - $\frac{3}{2}n$
    - ... ara $n_0 \geq 0$
    - (... que <u>existe</u> <u>alguma escolha para as constantes</u>



Notação *O*

- Usamos a notação *O* para dar um limite superior sobre uma função, dentro de um fator constante.



Notação *O*

- Com a notação ***O*** podemos descrever frequentemente o tempo de execução de um algoritmo apenas inspecionando a estrutura global do algoritmo.
  - Exemplo:
    - estrutura de laço duplamente aninhado no algoritmo *insertion-sort* (visto anteriormente) produz um limite superior $O(n^2)$ no pior caso:
      - custo do laço interno é limitado na parte superior por $O(1)$ (constante)
      - índices i e j são no máximo n
      - laço interno é executado no máximo uma vez para cada um dos n^2 pares de valores correspondentes a i e j

Notação O – o pior caso

- Como a notação O dá um limite superior, quando empregado ao pior caso...

Notação O – o pior caso

- Como a notação O dá um limite superior, quando empregado ao pior caso...
  - indica que esse limite vale para **qualquer** instância daquele algoritmo.

Notação O – o pior caso

- Como a notação O dá um limite superior, quando empregado ao pior caso...
  - indica que esse limite vale para **qualquer** instância daquele algoritmo.
- Assim, o limite $O(n^2)$ do pior caso do insertion sort também se aplica a qualquer entrada
- Veremos que o mesmo não é verdadeiro para a notação Θ

Operações com a notação *O*

$$\begin{aligned}
f(n) &= O(f(n)) \\
c \times f(n) &= O(f(n)), \text{ } c \text{ é uma constante} \\
O(f(n)) + O(f(n)) &= O(f(n)) \\
O(O(f(n))) &= O(f(n)) \\
O(f(n)) + O(g(n)) &= O(max(f(n), g(n))) \\
O(f(n))O(g(n)) &= O(f(n)g(n)) \\
f(n)O(g(n)) &= O(f(n)g(n))
\end{aligned}$$

SISTEMAS DE INFORMAÇÃO

EACH

Operações com a notação *O*

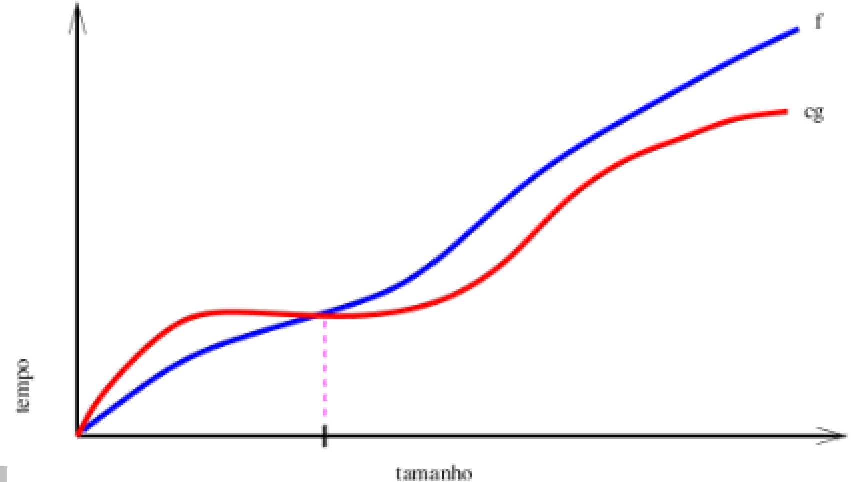
- A regra $O(f(n)) + O(g(n)) = O(\max(f(n),g(n)))$ pode ser usada para calcular o tempo de execução de uma sequência de trechos de um programa
  - Suponha 3 trechos: $O(n)$, $O(n^2)$ e $O(n\log n)$
  - Qual o tempo de execução do algoritmo como um todo?
    - ???

Operações com a notação O

- A regra $O(f(n)) + O(g(n)) = O(\max(f(n),g(n)))$ pode ser usada para calcular o tempo de execução de uma sequência de trechos de um programa
  - Suponha 3 trechos: $O(n)$, $O(n^2)$ e $O(n\log n)$
  - Qual o tempo de execução do algoritmo como um todo?
    - Lembre-se que o tempo de execução é a soma dos tempos de cada trecho
    - $O(n) + O(n^2) + O(n\log n) = \max(O(n), O(n^2), O(n\log n)) = O(n^2)$

Notação Ω

- Definição:
  - $\Omega(g(n)) = \{f(n)$: existem constantes positivas $\boldsymbol{c}$ e $\boldsymbol{n_0}$ tais que $0 \leq cg(n) \leq f(n)$, para todo $\boldsymbol{n} \geq \boldsymbol{n_0}\}$
- Informalmente, dizemos que, se $f(n) \in \Omega(g(n))$, então f(n) cresce no mínimo tão lentamente quanto g(n).
  - Note que se $f(n) \in O(g(n))$ define um limite superior para f(n), $\Omega(g(n))$ define um limite inferior



Notação Ω

- Definição:
  - Ω(g(n)) = {f(n): existem constantes positivas c e n_0 tais que $0 \leq cg(n) \leq f(n)$, para todo $n \geq n_0$}
  - $\frac{3}{2}n^2 - 2n \in \Omega\left(n^2\right)$?
    - ???

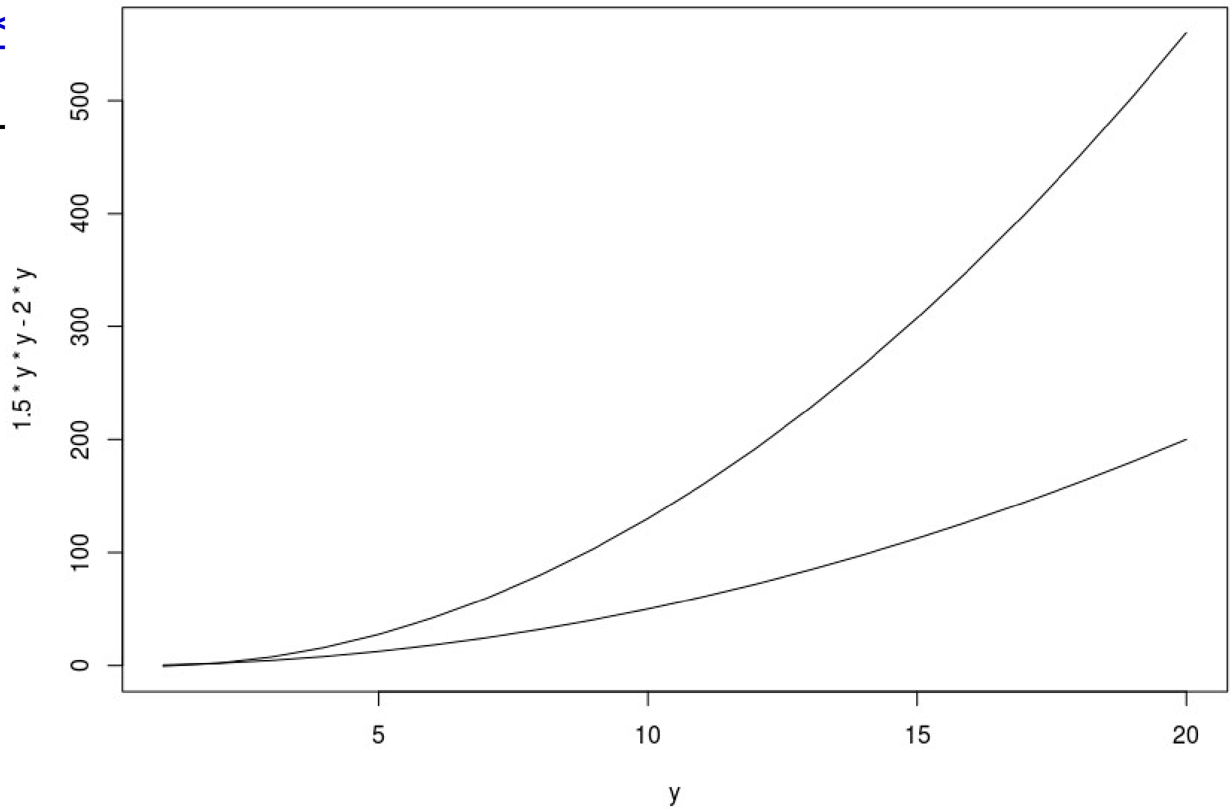
Notação Ω

- Definição:
  - Ω(g(n)) = {f(n): existem constantes positivas ***c*** e $\boldsymbol{n_0}$ tais que 0 ≤ cg(n) ≤ f(n), para todo ***n*** ≥ $\boldsymbol{n_0}$}
  - $\frac{3}{2}n^2 - 2n \in \Omega(n^2)$?
    - Fazendo c = 1/2, teremos $\frac{3}{2}n^2 - 2n \geq \frac{1}{2}n^2$, para $n_0 \geq 2$

Notação *Ω*

- Definição:
  - *Ω*(… ***c*** e ***n₀*** tais que

    0 …
  - $\frac{3}{2}$

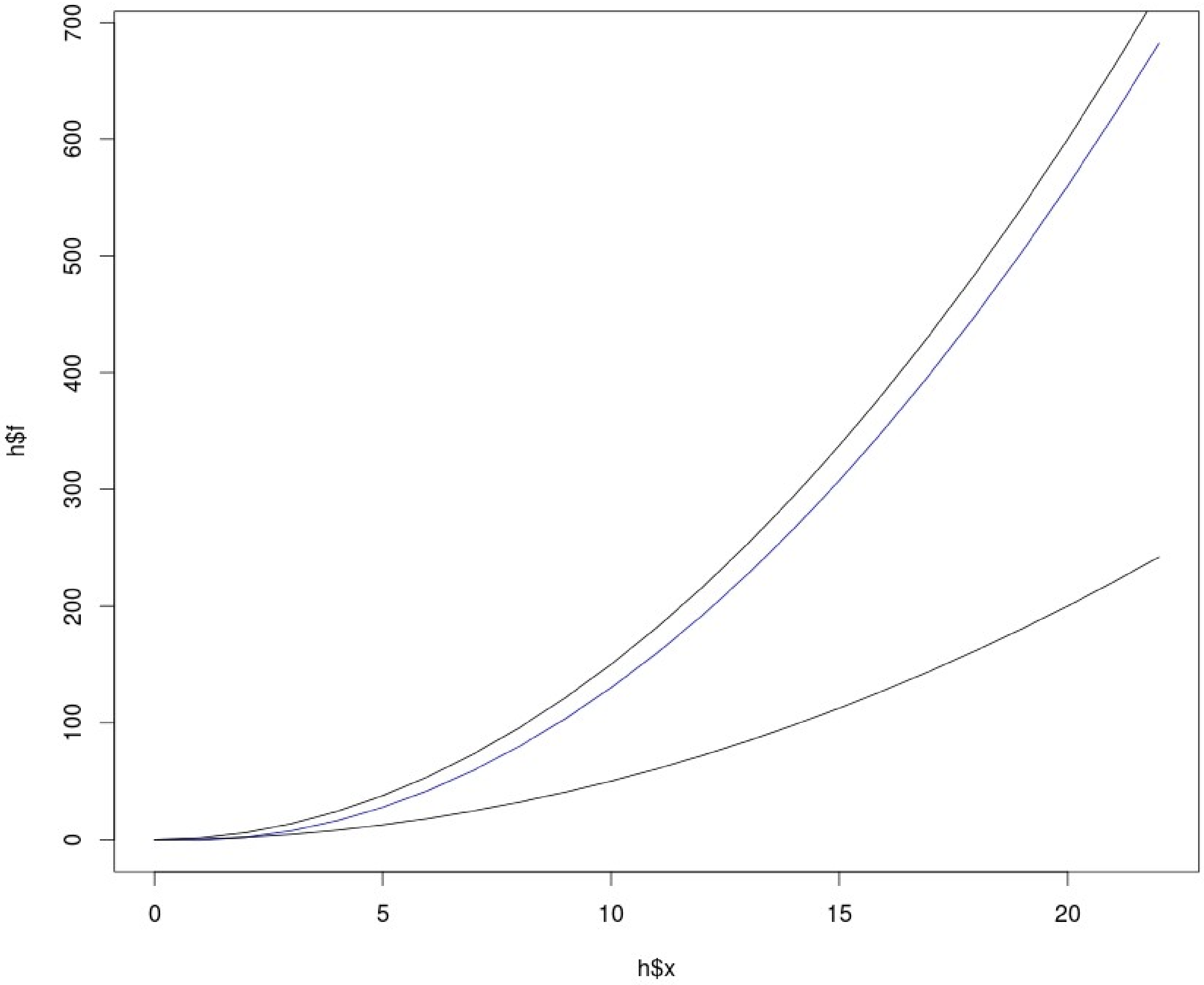


para $n_0 \geq 2$

SISTEMAS DE INFORMAÇÃO

EACH

Notação O e Ω



$3/2\ n^2 - 2n$

$3/2\ n^2$

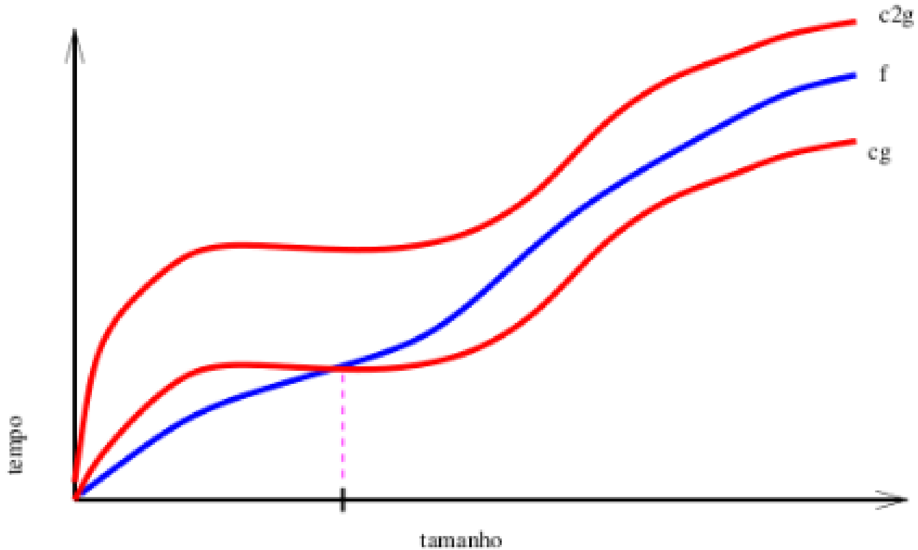
$1/2\ n^2$

SISTEMAS DE INFORMAÇÃO

EACH

Notação Θ

- Definição:
  - $\Theta(g(n)) = \{f(n)$: existem constantes positivas c_1, c_2 e n_0 tais que $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$, para todo $n \ge n_0\}$
- Informalmente, dizemos que, se $f(n) \in \Theta(g(n))$, então f(n) cresce tão rapidamente quanto g(n).



Notação *Θ*

- Definição:
  - $\Theta(g(n))$ = {f(n): existem constantes positivas $\boldsymbol{c_1}$, $\boldsymbol{c_2}$ e $\boldsymbol{n_0}$ tais que $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$, para todo $\boldsymbol{n} \geq \boldsymbol{n_0}$}
  - $\frac{3}{2}n^2 - 2n \in \Theta(n^2)$?
    - ???

SISTEMAS DE INFORMAÇÃO

EACH

Notação Θ

- Definição:
  - Θ(g(n)) = {f(n): existem constantes positivas $\boldsymbol{c_1}$, $\boldsymbol{c_2}$ e $\boldsymbol{n_0}$ tais que $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$, para todo $\boldsymbol{n \geq n_0}$}
  - $\frac{3}{2}n^2 - 2n \in \Theta(n^2)$?
    - Fazendo $c_1 = 1/2$ e $c_2 = 3/2$ teremos $$|\frac{1}{2}n^2| \leq |\frac{3}{2}n^2 - 2n| \leq |\frac{3}{2}n^2|$$

para $n_0 \geq 2$

Notação Θ

- Mas, já vimos que:
  - $\frac{3}{2}n^2 - 2n \in O(n^2)$
  - $\frac{3}{2}n^2 - 2n \in \Omega(n^2)$ e ...
  - $\frac{3}{2}n^2 - 2n \in \Theta(n^2)$
  - Será coincidência?
    - ???

Notação Θ

- Mas, já vimos que:
  - $\frac{3}{2}n^2 - 2n \in O(n^2) \rightarrow |\frac{3}{2}n^2 - 2n| \leq |\frac{3}{2}n^2|$

    $\frac{3}{2}n^2 - 2n \in \Omega(n^2) \rightarrow |\frac{1}{2}n^2| \leq |\frac{3}{2}n^2 - 2n|$ e ...

  - $\frac{3}{2}n^2 - 2n \in \Theta(n^2) \rightarrow |\frac{1}{2}n^2| \leq |\frac{3}{2}n^2 - 2n| \leq |\frac{3}{2}n^2|$

  - Será coincidência?
    - ???

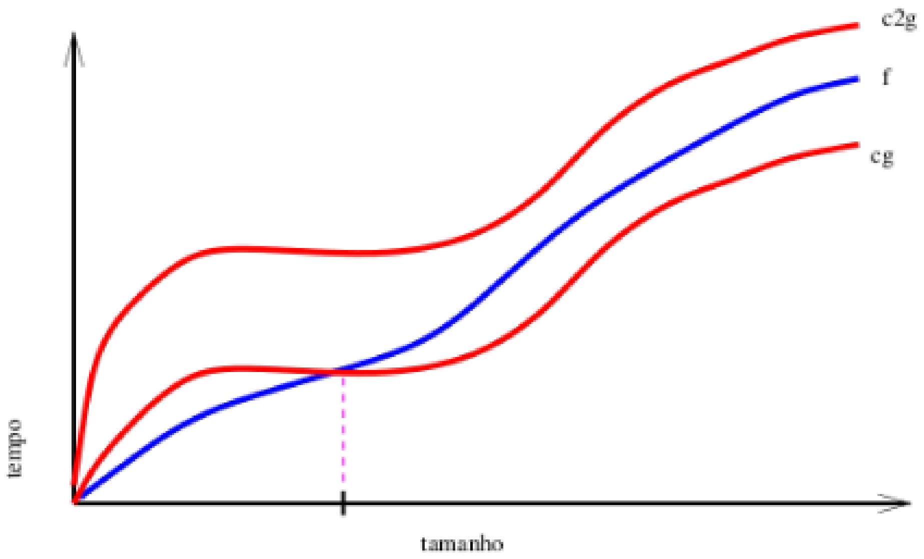
Notação Θ

- Mas, já vimos que:
  - $\frac{3}{2}n^2 - 2n \in O(n^2) \quad \rightarrow \quad |\frac{3}{2}n^2 - 2n| \le |\frac{3}{2}n^2|$
  - $\frac{3}{2}n^2 - 2n \in \Omega(n^2) \quad \rightarrow \quad |\frac{1}{2}n^2| \le |\frac{3}{2}n^2 - 2n|$ e ...
  - $\frac{3}{2}n^2 - 2n \in \Theta(n^2) \quad \rightarrow \quad |\frac{1}{2}n^2| \le |\frac{3}{2}n^2 - 2n| \le |\frac{3}{2}n^2|$

  - Será coincidência?
    - Não!
    - Se f(n) ∈ O(g(n)) e f(n) ∈ Ω (g(n)), então f(n) ∈ Θg(n))

Notação Θ

- Mas:
  - Será coincidência?
    - Não!
    - Se $f(n) \in O(g(n))$ e $f(n) \in \Omega(g(n))$, então $f(n) \in \Theta(g(n))$



Notação Θ – pior caso

- O tempo limite de $\Theta(n^2)$ para o pior caso *do insertion sort*
- Não implica um tempo $\Theta(n^2)$ para qualquer entrada
- Por exemplo, se pegarmos o melhor caso, vemos que ele tem $\Theta(n)$

Notação Θ – pior caso

- O tempo limite de $\Theta(n^2)$ para o pior caso *do insertion sort*
- Não implica um tempo $\Theta(n^2)$ para qualquer entrada
- Por exemplo, se pegarmos o melhor caso, vemos que ele tem $\Theta(n)$

==> $O(n^2)$
=/=> $\Theta(n^2)$
=/=> $\Omega(n^2)$

Notação *o*

- Definição:
  - $o(g(n)) = \{f(n)$: para toda constante positiva c, existe uma constante $n_0 > 0$ tal que $0 \le f(n) < cg(n)$, para todo $n \ge n_0\}$
- Informalmente, dizemos que, se $f(n) \in o(g(n))$, então $f(n)$ cresce mais lentamente que $g(n)$.
  - Intuitivamente, na notação ***o***, a função f(n) torna-se insignificante em relação a g(n) quando ***n*** tende para o infinito
  - $\lim_{n \to \infty}(f(n)/g(n)) = 0$

Notação *o*

- $1000n^2 \in o\left(n^3\right)$?
  - ???

Notação *o*

- $1000n^2 \in o\left(n^3\right)$?
  - Para todo valor de ***c***, um $\boldsymbol{n_0}$ que satisfaz a definição é:

$$n_0 = \left|\frac{1000}{c}\right| + 1$$

Notação *o*

- Qual a diferença entre ***O*** e ***o***?
  - ***O***: **existem** constantes positivas c e n_0 tais que $0 \le f(n) \le cg(n)$, para todo $n \ge n_0$
    - A expressão $0 \le f(n) \le cg(n)$ é válida para alguma constante $c>0$
  - ***o***: **para toda** constante positiva c, existe uma constante $n_0 > 0$ tal que $0 \le f(n) < cg(n)$, para todo $n \ge n_0$
    - A expressão $0 \le f(n) < cg(n)$ é válida para toda constante $c>0$
    - O limite superior não é assintoticamente justo

Notação *o*

- Qual a diferença entre ***O*** e ***o***?
  - O limite superior não é assintoticamente justo
  - Por exemplo:
    - $2n = o(n^2)$
    - $2n^2 \neq o(n^2)$

SISTEMAS DE INFORMAÇÃO

EACH

Notação ω

- Definição:
  - $\omega(g(n)) = \{f(n)$: para toda constante positiva $\boldsymbol{c}$, existe uma constante $n_0 > 0$ tal que $0 \leq cg(n) < f(n)$, para todo $\boldsymbol{n} \geq \boldsymbol{n_0}$ $\}$
- Informalmente, dizemos que, se $f(n) \in \omega(g(n))$, então f(n) cresce mais rapidamente que g(n).
  - Intuitivamente, na notação ω, a função f(n) tem crescimento muito maior que g(n) quando $\boldsymbol{n}$ tende para o infinito

Notação ω

- ω está para $\boldsymbol{\Omega}$, da mesma forma que $\boldsymbol{o}$ está para $\boldsymbol{O}$
  - $\boldsymbol{O}$ e $\boldsymbol{\Omega}$ são chamados de assintoticamente firmes
- $|\frac{1}{1000}n^2| \in \omega(n)$?
  - ???

Notação ω

- ω está para ***Ω***, da mesma forma que ***o*** está para ***O***
  - ***O*** e ***Ω*** são chamados de assintoticamente firmes
- $\frac{1}{1000}n^2 \in \omega(n)$?
  - Para todo valor de ***c***, um n_0 que satisfaz a definição é:

$$n_0 = \lfloor 1000c \rfloor + 1$$

Propriedades das Classes

Reflexividade:

$f(n) \in O(f(n))$.
$f(n) \in \Omega(f(n))$.
$f(n) \in \Theta(f(n))$.

Simetria:

$f(n) \in \Theta(g(n))$ se, e somente se, $g(n) \in \Theta(f(n))$.

Simetria Transposta:

$f(n) \in O(g(n))$ se, e somente se, $g(n) \in \Omega(f(n))$.
$f(n) \in o(g(n))$ se, e somente se, $g(n) \in \omega(f(n))$.

SISTEMAS DE INFORMAÇÃO

EACH

Propriedades das Classes

Transitividade:

Se $f(n) \in O(g(n))$ e $g(n) \in O(h(n))$, então $f(n) \in O(h(n))$.
Se $f(n) \in \Omega(g(n))$ e $g(n) \in \Omega(h(n))$, então $f(n) \in \Omega(h(n))$.
Se $f(n) \in \Theta(g(n))$ e $g(n) \in \Theta(h(n))$, então $f(n) \in \Theta(h(n))$.
Se $f(n) \in o(g(n))$ e $g(n) \in o(h(n))$, então $f(n) \in o(h(n))$.
Se $f(n) \in \omega(g(n))$ e $g(n) \in \omega(h(n))$, então $f(n) \in \omega(h(n))$.

Analogia

- Analogia com números reais a e b:
  - $f(n) = O(g(n))$ é como $a \leq b$
  - $f(n) = \Omega(g(n))$ é como $a \geq b$
  - $f(n) = (g(n))$ é como $a = b$
  - $f(n) = o(g(n))$ é como $a < b$
  - $f(n) = \Omega(g(n))$ é como $a > b$

Exercício

Quais as relações de comparação assintótica (O, Ω, Θ) das funções:

$$f_1(n) = 2^\pi$$
$$f_2(n) = 2^n$$
$$f_3(n) = n \log n$$
$$f_4(n) = \log n$$
$$f_5(n) = 100n^2 + 150000n$$
$$f_6(n) = n + \log n$$
$$f_7(n) = n^2$$
$$f_8(n) = n$$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_1	Θ							
f_2		Θ						
f_3			Θ					
f_4				Θ				
f_5					Θ			
f_6						Θ		
f_7							Θ	
f_8								Θ

Referências

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest & Clifford Stein. Algoritmos - Tradução da 2a. Edição Americana. Editora Campus, 2002 (Capítulo 3).
- Michael T. Goodrich & Roberto Tamassia. Estruturas de Dados e Algoritmos em Java. Editora Bookman, 4a. Ed. 2007 (Capítulo 4).
- Nívio Ziviani. Projeto de Algoritmos com implementações em C e Pascal. Editora Thomson, 2a. Edição, 2004 (Seção 1.3).
- Notas de aula dos professores Marcos Chaim, Cid de Souza, Cândida da Silva e Delano M. Beder.

Complexidade Assintótica

Professores:

Norton T. Roman

Fátima L. S. Nunes