Análise Assintótica

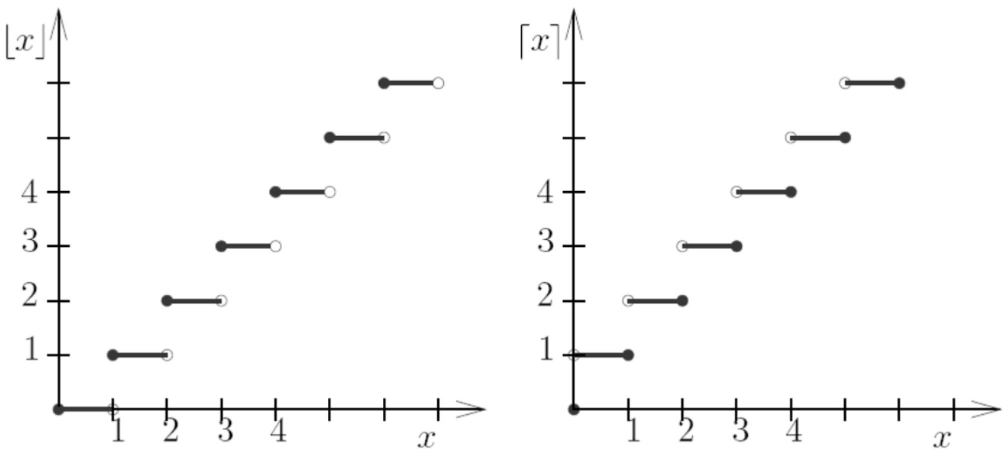
Análise Assintótica

- Objetivos:
  - Apresentar as notações assintóticas usadas para descrever a taxa de crescimento de funções de custo de algoritmos
- Tópicos:
  - Chão e Teto
  - Notação O
  - Notação Ω
  - Notação Θ
  - Classes de comportamento assintótico.
  - Algoritmo Exponencial *vs* Polinomial

Chão e Teto

$\lfloor x \rfloor$, lê-se: chão de *x*, o inteiro *i* tal que $i \leq x < i + 1$
$\lceil x \rceil$, lê-se: teto de *x*, o inteiro *j* tal que $j - 1 < x \leq j$

Exemplo: Desenhe os gráficos para as funções $\lfloor x \rfloor$ e $\lceil x \rceil$ para *x* não negativo.

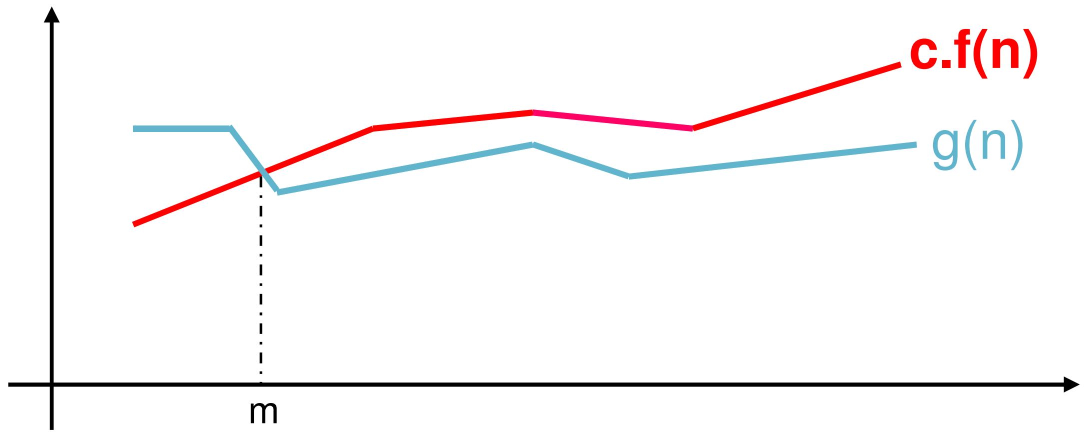


Custo Assintótico de Funções

- O custo assintótico de uma função $f(n)$ representa o limite do comportamento de custo quando n cresce.
- Geralmente, um algoritmo que é assintoticamente mais eficiente será a melhor escolha para valores grandes de n.

Dominação Assintótica

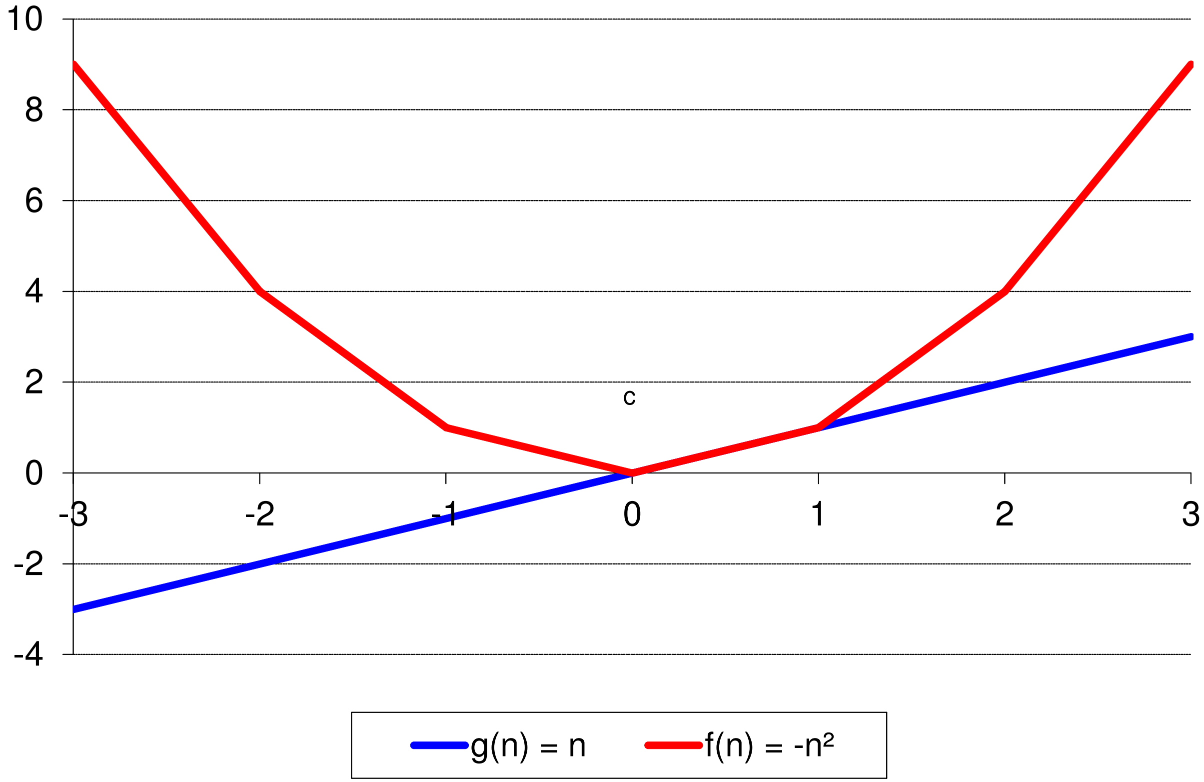
- *__Definição:__* f(n) domina assintoticamente g(n) se existem duas constantes positivas *c* e *m* tais que, para n >= m, temos |g(n)| <= c.|f(n)|



Dominação Assintótica

Exemplo: Seja $g(n) = n$ e $f(n) = -n^2$, temos que $|n| <= |-n^2|$ para todo $n \in N$.

Assim, fazendo $c = 1$ e $m = 0$ a definição é satisfeita.
Logo, *f(n)* domina assintoticamente *g(n)*.



Notação *O*

- A notação big-Oh define um limite superior para a função, dado um fator constante *c*.

- Lê-se:
  - Notação trazida da matemática por Knuth (1968): g(n) = O(f(n))
  - g(n) é de ordem no máximo f(n)
  - f(n) domina assintoticamente g(n)
  - f(n) é um limite assintótico superior para g(n)

- O(f(n)) representa o conjunto de todas as funções que são assintoticamente dominadas por uma dada função f(n).

Notação *O*

Definição: Dadas funções assintoticamente não-negativas f e g, dizemos que g está na ordem O de f, e escrevemos $g=O(f)$, se $g(n) <= c \cdot f(n)$ para algum c positivo e para todo n suficientemente grande. Em outras palavras, existe um número positivo c e um número m tais que $g(n)<=c \cdot f(n)$ para todo n maior que m.

$$g(n)=O(f(n)),\ \exists\ c>0 \text{ e } m \mid 0<=g(n)<=c.f(n),\ n>=m$$

Exemplo

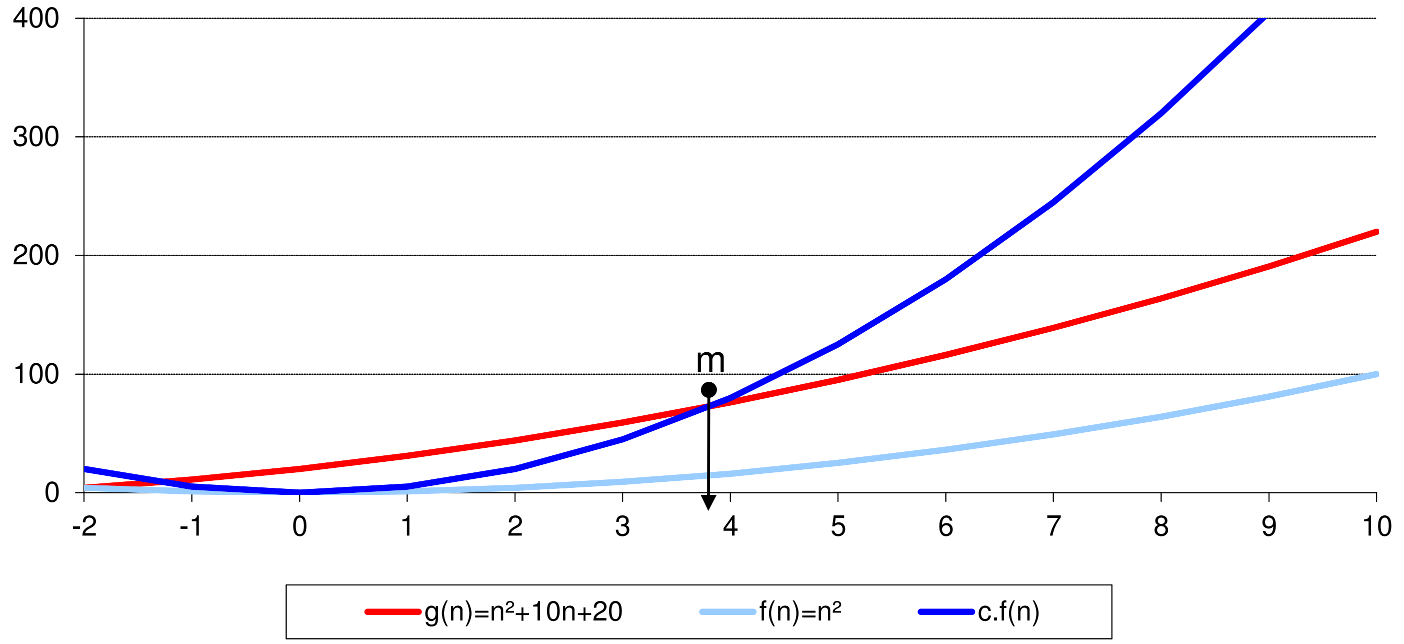
Prove que: $n^2+10n+20=O(n^2)$

- Sabemos que bigOh equivale ao conjunto de todas as funções dominadas assintóticamente por uma função, que neste caso é n^2.
- Assim, segundo a definição, temos que:
  $g=O(f) \Leftrightarrow \exists\ c>0$ e $m \mid 0<=g(n)<=c.f(n),\ n>=m$
- Então basta encontrarmos as constantes *c* e *m* que tornem a desigualdade verdadeira.

Exemplo

Prove que: $n^2+10n+20=O(n^2)$

- Se $g(n)=n^2+10n+20$ e $f(n)=n^2$:



- Então, para $c=5$ e $m=4$, $g(n)<=c.f(n)$ (c.q.d)

Notação Ω

- Ω define um limite inferior para a função, por um fator constante.

- Lê-se:
  - g(n) é de ordem no mínimo f(n)
  - f(n) é dominada assintoticamente g(n)
  - f(n) é um limite assintótico inferior para g(n)

- A notação Ω é usada para expressar o limite inferior do tempo de execução de qualquer algoritmo para resolver um problema específico

Notação Ω

<u>*Definição*</u>: Dadas funções assintoticamente não-negativas f e g, dizemos que g está na ordem Ω de f, e escrevemos $g=\Omega(f)$, se $c \cdot f(n) <= g(n)$ para algum c positivo e para todo n suficientemente grande. Em outras palavras, existe um número positivo c e um número m tais que $g(n) >= c \cdot f(n)$ para todo n maior que m.

$g(n) = \Omega(f(n)), \exists\ c>0$ e $m \mid 0 <= c.f(n) <= g(n),\ n >= m$

Exemplo

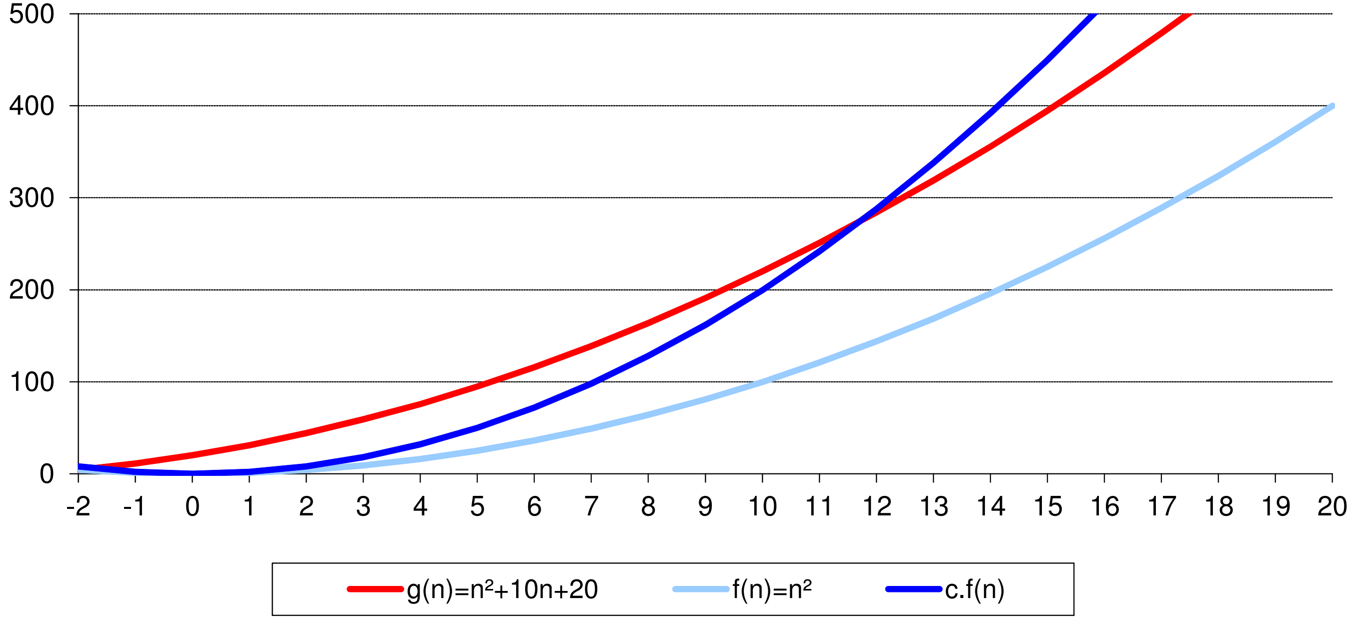
Prove que: $n^2+10n+20=\Omega(n^2)$

- Sabemos que Ômega equivale ao conjunto de todas as funções que exercem dominação assintótica sob uma função f(n), neste caso, n^2.
- Assim, segundo a definição, temos que:
  $g=\Omega(f) \Leftrightarrow \exists\ c>0$ e $m \mid 0<=c.f(n)<=g(n),\ n>=m$
- Então basta encontrarmos as constantes *c* e *m* que tornem a desigualdade verdadeira.

Exemplo

Prove que: $n^2+10n+20=\Omega(n^2)$

- Se $g(n)=n^2+10n+20$ e $f(n)=n^2$:

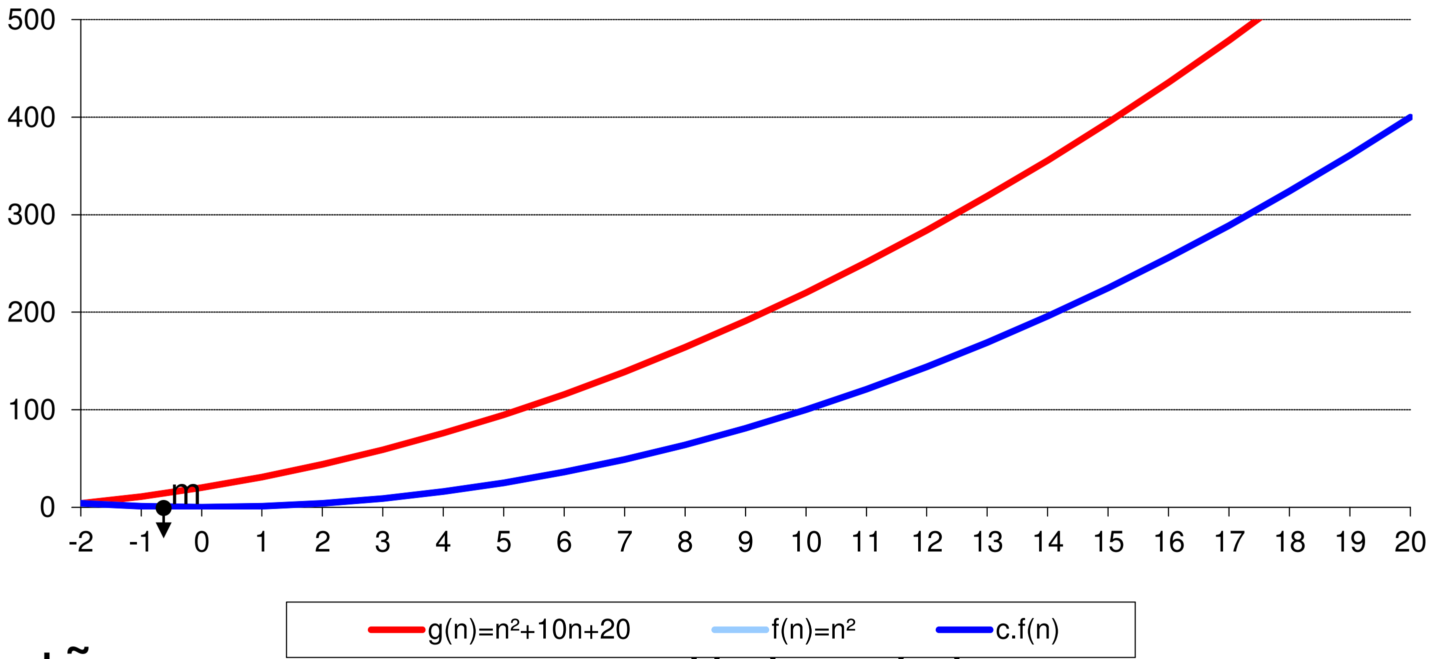


- Para c=2, c.f(n)<=g(n) somente até n≈12 !!!

Exemplo

Prove que: $n^2+10n+20=\Omega(n^2)$

- Se $g(n)=n^2+10n+20$ e $f(n)=n^2$:



- Então, para $c=1$ e $m=0$, $c.f(n)<=g(n)$ (c.q.d)

Comportamento Assintótico de Funções

- Assim, existem duas notações principais na análise assintótica de funções: O e Ω

- $O(f(n))$ – limite superior
  - depende do algoritmo
- $\Omega(f(n))$ – limite inferior
  - depende do problema

Notação Θ

- Θ limita a função, acima e abaixo, por fatores constantes.

Definição:

- Lê-se:
  - g(n) é de mesma ordem Theta que f(n)
  - f(n) é um limite assintótico firme para g(n)

- A notação Θ é usada para expressar o limite inferior e superior do tempo de execução de qualquer algoritmo para resolver um problema específico

Notação Θ

<u>Definição</u>: Dadas funções assintoticamente não-negativas f e g, dizemos que g está na ordem Θ de f, e escrevemos $g(n)=\Theta(f(n))$, se $c1 \cdot f(n) <=$ $g(n) <= c1 \cdot f(n)$ para algum $c1$ e $c2$ e m positivos, tais que para todo $n >= m$ *e* suficientemente grande.

$$g(n)= \Theta(f(n)), \exists\ c1,c2>0 \text{ e } m$$
$$\text{tal que } 0<=c1.f(n)<=g(n) <=c2.f(n),\ n>=m$$

Exemplo

Prove que: $n^2+10n+20=\Theta(n^2)$

- Sabemos que Theta equivale ao conjunto de todas as funções delimitadas assintóticamente acima e abaixo, por uma função f(n), que neste caso é n^2.
- Assim, segundo a definição, temos que:

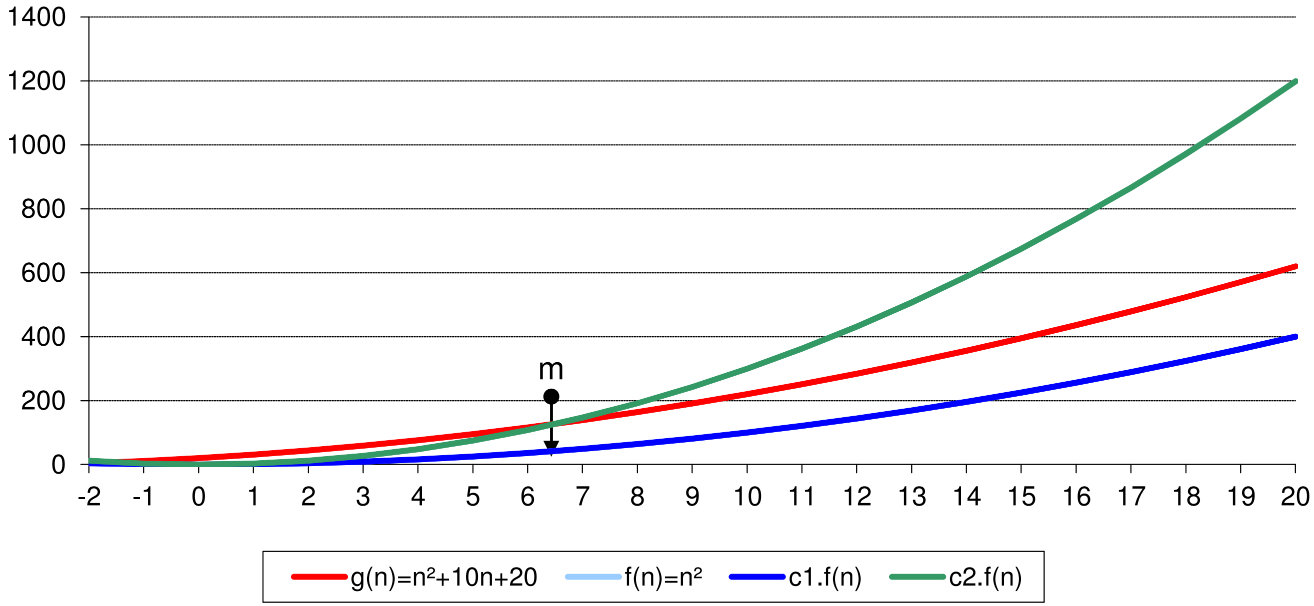
$$g=\Theta(f) \Leftrightarrow \exists\, c1,c2>0 \text{ e } m \mid$$
$$0<=c1.f(n)<=g(n)<=c2.f(n),\ n>=m$$

- Então basta encontrarmos as constantes *c1*, *c2* e *m* que tornem a desigualdade verdadeira.

Exemplo

Prove que: $n^2+10n+20=\Theta(n^2)$

Se $g(n)=n^2+10n+20$ e $f(n)=n^2$:



- Então, para $c_1=1$, $c_2=3$ e $m=7$, temos: $c_1.f(n) <= g(n) <= c_2.f(n)$ (c.q.d)

Comportamento Assintótico

- $f(n) = O(1)$ (complexidade constante)
  O uso do algoritmo independe do tamanho de n. Neste caso, as instruções do programa são executadas um número fixo de vezes.

- $f(n) = O(\log n)$ (complexidade logaritmica)
  Ocorre tipicamente em algoritmos que resolvem um problema transformando-o em problemas menores. Nestes casos, o tempo de execução pode ser considerado como sendo menor que uma constante grande . Exemplos: $n = 1000$ -- $\log_2 n \sim 10$;
  $n = 10^6 – \log_2 n \sim 20$.

- $f(n) = O(n)$ (complexidade linear)
  Em geral, um pequeno trabalho é realizado sobre cada elemento de entrada. Esta é a melhor situação possível para um algoritmo que tem que processar n elementos de entrada ou produzir n elementos de saída. Cada vez que n dobra de tamanho n dobra.

Comportamento Assintótico

- $f(n) = O(n \log n)$
  Este tempo de execução ocorre tipicamente em algoritmos resolvidos com o método dividir&conquistar. Exemplos: n = 1 milhão – n log n ~20 milhões;
  n = 2 milhões, n log n ~ 42 milhões, pouco mais que o dobro.

- $f(n) = O(n^2)$ (complexidade quadrática)
  Algoritmos desta ordem de complexidade ocorrem quando os itens são processados aos pares, em um laço dentro de outro. São úteis para resolver problemas de tamanho pequeno – métodos diretos de ordenação.

Comportamento Assintótico

- $f(n) = O(n^3)$ (complexidade cúbica)
  Algoritmos desta ordem são úteis apenas para resolver pequenos problemas.

- $f(n) = O(2^n)$ (complexidade exponencial)
  Algoritmos desta ordem geralmente não são úteis sob o ponto de vista prático. Eles ocorrem na solução de problemas quando se usa a força bruta para resolvê-los.

Exponencial vs Polinomial

- Um algoritmo cuja função de complexidade é $O(c^n)$, $c>1$, é chamado de **algoritmo exponencial de tempo de execução**.
- Um algoritmo cuja função de complexidade é $O(p(n))$, onde $p(n)$ é um polinômio, é chamado de **algoritmo polinomial de tempo de execução**.
- Um problema é considerado **intratável** se ele é tão difícil que não existe um algoritmo polinomial para resolvê-lo, enquanto um problema é considerado **bem resolvido** quando existe um algoritmo polinomial para resolvê-lo.