Notação Assintótica

Letícia Rodrigues Bueno

Notação Assintótica

- Para valores suficientemente pequenos de n, qualquer algoritmo custa pouco para ser executado, mesmo os algoritmos ineficientes;

Notação Assintótica

- Para valores suficientemente pequenos de n, qualquer algoritmo custa pouco para ser executado, mesmo os algoritmos ineficientes;
- A análise de algoritmos é realizada para valores grandes de n considerando-se o comportamento assintótico das funções de custo;

Notação Assintótica

- Para valores suficientemente pequenos de n, qualquer algoritmo custa pouco para ser executado, mesmo os algoritmos ineficientes;
- A análise de algoritmos é realizada para valores grandes de n considerando-se o comportamento assintótico das funções de custo;
- O comportamento assintótico de $f(n)$ representa o limite do comportamento do custo quando n cresce.

Notação Assintótica: Notação O

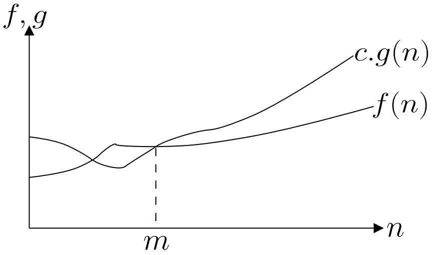
- Uma função $f(n)$ é $O(g(n))$, **notação** $f(n) = O(g(n))$, se existem duas constantes positivas c e m tais que $f(n) \leq c.g(n)$, para todo $n \geq m$;

Notação Assintótica: Notação O

- Uma função $f(n)$ é $O(g(n))$, **notação** $f(n) = O(g(n))$, se existem duas constantes positivas c e m tais que $f(n) \leq c.g(n)$, para todo $n \geq m$;
- $f(n) = O(g(n))$: $g(n)$ é limite superior para $f(n)$;

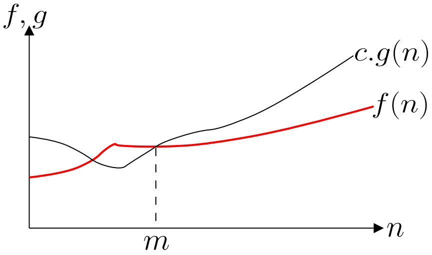
Notação Assintótica: Notação O

- Uma função $f(n)$ é $O(g(n))$, **notação** $f(n) = O(g(n))$, se existem duas constantes positivas c e m tais que $f(n) \leq c.g(n)$, para todo $n \geq m$;
- $f(n) = O(g(n))$: $g(n)$ é limite superior para $f(n)$;



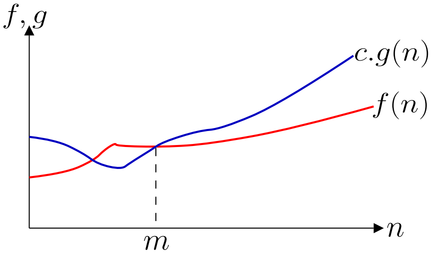
Notação Assintótica: Notação O

- Uma função $f(n)$ é $O(g(n))$, **notação** $f(n) = O(g(n))$, se existem duas constantes positivas c e m tais que $f(n) \leq c.g(n)$, para todo $n \geq m$;
- $f(n) = O(g(n))$: $g(n)$ é limite superior para $f(n)$;



Notação Assintótica: Notação O

- Uma função $f(n)$ é $O(g(n))$, **notação** $f(n) = O(g(n))$, se existem duas constantes positivas c e m tais que $f(n) \leq c.g(n)$, para todo $n \geq m$;
- $f(n) = O(g(n))$: $g(n)$ é limite superior para $f(n)$;



Notação Assintótica: Notação Ω

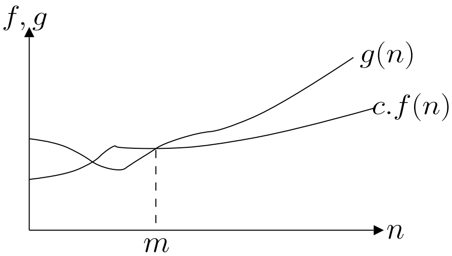
- Uma função $g(n)$ é $\Omega(f(n))$, **notação** $g(n) = \Omega(f(n))$, se existem duas constantes positivas c e m tais que $g(n) \geq c.f(n)$, para todo $n \geq m$;

Notação Assintótica: Notação Ω

- Uma função $g(n)$ é $\Omega(f(n))$, **notação** $g(n) = \Omega(f(n))$, se existem duas constantes positivas c e m tais que $g(n) \geq c.f(n)$, para todo $n \geq m$;
- $g(n) = \Omega(f(n))$: $f(n)$ é limite inferior para $g(n)$ (tempo mínimo do algoritmo);

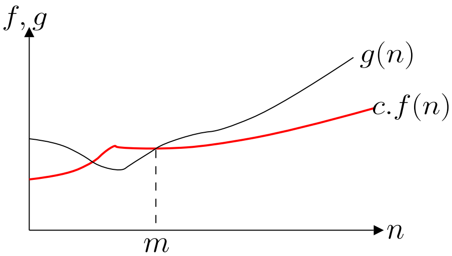
Notação Assintótica: Notação Ω

- Uma função $g(n)$ é $\Omega(f(n))$, **notação** $g(n) = \Omega(f(n))$, se existem duas constantes positivas c e m tais que $g(n) \geq c.f(n)$, para todo $n \geq m$;
- $g(n) = \Omega(f(n))$: $f(n)$ é limite inferior para $g(n)$ (tempo mínimo do algoritmo);



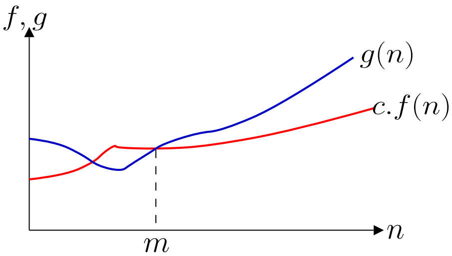
Notação Assintótica: Notação Ω

- Uma função $g(n)$ é $\Omega(f(n))$, **notação** $g(n) = \Omega(f(n))$, se existem duas constantes positivas c e m tais que $g(n) \geq c.f(n)$, para todo $n \geq m$;
- $g(n) = \Omega(f(n))$: $f(n)$ é limite inferior para $g(n)$ (tempo mínimo do algoritmo);



Notação Assintótica: Notação Ω

- Uma função $g(n)$ é $\Omega(f(n))$, **notação** $g(n) = \Omega(f(n))$, se existem duas constantes positivas c e m tais que $g(n) \geq c.f(n)$, para todo $n \geq m$;
- $g(n) = \Omega(f(n))$: $f(n)$ é limite inferior para $g(n)$ (tempo mínimo do algoritmo);



Notação Assintótica: Notação Θ

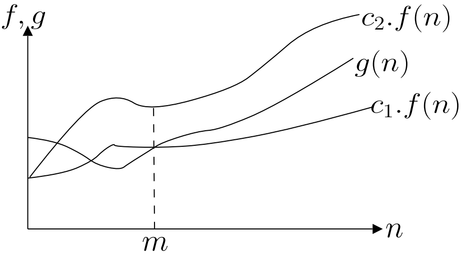
- Uma função $g(n)$ é $\Theta(f(n))$, **notação** $g(n) = \Theta(f(n))$, se existem duas constantes positivas c_1, c_2 e m tais que $0 \leq c_1.f(n) \leq g(n) \leq c_2.f(n)$, para todo $n \geq m$;

Notação Assintótica: Notação Θ

- Uma função $g(n)$ é $\Theta(f(n))$, **notação** $g(n) = \Theta(f(n))$, se existem duas constantes positivas c_1, c_2 e m tais que $0 \leq c_1.f(n) \leq g(n) \leq c_2.f(n)$, para todo $n \geq m$;
- $g(n) = \Theta(f(n))$: $f(n)$ é limite restrito para $g(n)$ ou limite assintótico firme;

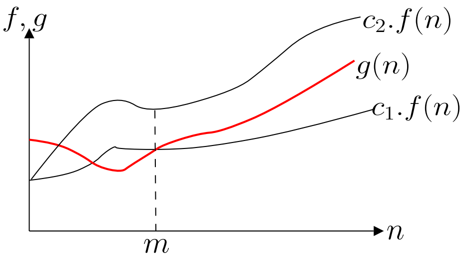
Notação Assintótica: Notação Θ

- Uma função $g(n)$ é $\Theta(f(n))$, **notação** $g(n) = \Theta(f(n))$, se existem duas constantes positivas c_1, c_2 e m tais que $0 \leq c_1.f(n) \leq g(n) \leq c_2.f(n)$, para todo $n \geq m$;
- $g(n) = \Theta(f(n))$: $f(n)$ é limite restrito para $g(n)$ ou limite assintótico firme;



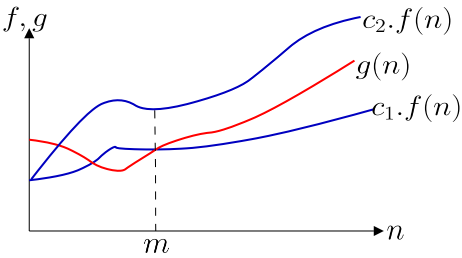
Notação Assintótica: Notação Θ

- Uma função $g(n)$ é $\Theta(f(n))$, **notação** $g(n) = \Theta(f(n))$, se existem duas constantes positivas c_1, c_2 e m tais que $0 \leq c_1.f(n) \leq g(n) \leq c_2.f(n)$, para todo $n \geq m$;
- $g(n) = \Theta(f(n))$: $f(n)$ é limite restrito para $g(n)$ ou limite assintótico firme;



Notação Assintótica: Notação Θ

- Uma função $g(n)$ é $\Theta(f(n))$, **notação** $g(n) = \Theta(f(n))$, se existem duas constantes positivas c_1, c_2 e m tais que $0 \leq c_1.f(n) \leq g(n) \leq c_2.f(n)$, para todo $n \geq m$;
- $g(n) = \Theta(f(n))$: $f(n)$ é limite restrito para $g(n)$ ou limite assintótico firme;



Notação Assintótica: Notações o e ω

- Uma função $g(n)$ é $o(f(n))$, **notação** $g(n) = o(f(n))$, se, para qualquer constante $c > 0$, temos $0 \leq g(n) < c.f(n)$, para todo $n \geq m$;

Notação Assintótica: Notações o e ω

- Uma função $g(n)$ é $o(f(n))$, **notação** $g(n) = o(f(n))$, se, para qualquer constante $c > 0$, temos $0 \leq g(n) < c.f(n)$, para todo $n \geq m$;
- Uma função $g(n)$ é $\omega(f(n))$, **notação** $g(n) = \omega(f(n))$, se, para qualquer constante $c > 0$, temos $0 \leq c.f(n) < g(n)$, para todo $n \geq m$;

Notação Assintótica: Notações o e ω

- Uma função $g(n)$ é $o(f(n))$, **notação** $g(n) = o(f(n))$, se, para qualquer constante $c > 0$, temos $0 \leq g(n) < c.f(n)$, para todo $n \geq m$;
- Uma função $g(n)$ é $\omega(f(n))$, **notação** $g(n) = \omega(f(n))$, se, para qualquer constante $c > 0$, temos $0 \leq c.f(n) < g(n)$, para todo $n \geq m$;

Comparação de Funções

Muitas das propriedades relacionais de números reais também se aplicam a comparações assintóticas. No caso das propriedades abaixo, suponha $f(n)$ e $g(n)$ assintoticamente positivas.

1. **Transitividade (válido também para O, Ω, o e ω):**
   $f(n) = \Theta(g(n))$ e $g(n) = \Theta(h(n))$ então $f(n) = \Theta(h(n))$.
2. **Reflexividade (válido também para O e Ω):**
   $f(n) = \Theta(f(n))$
3. **Simetria:**
   $f(n) = \Theta(g(n))$ se e somente se $g(n) = \Theta(f(n))$
4. **Simetria de transposição:**
   $f(n) = O(g(n))$ se e somente se $g(n) = \Omega(f(n))$
   $f(n) = o(g(n))$ se e somente se $g(n) = \omega(f(n))$

Regras de Simplificação

1. Se $f_1(n) = O(g_1(n))$ e $f_2(n) = O(g_2(n))$, então $f_1(n) + f_2(n)$ está em $O(\max(g_1(n), g_2(n)))$.
2. Se $f_1(n) = O(g_1(n))$ e $f_2(n) = O(g_2(n))$, então $f_1(n)f_2(n)$ está em $O(g_1(n)g_2(n))$.

2^a Lista de Exercícios - Entrega: 14/10/2011

1. Implemente os algoritmos de ordenação: (a) por inserção; (b) por seleção; (c) bolha. Contabilizar e comparar o número de operações gastas por cada algoritmo no melhor caso e no pior caso.

Bibliografia

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L. e STEIN, C. *Introduction to Algorithms*, 3ª edição, MIT Press, 2009.

SZWARCFITER, J. L. e MARKENZON, L. Estruturas de Dados e seus Algoritmos, LTC, 1994.

ZIVIANI, N. Projeto de Algoritmos: com implementações em Pascal e C, 2ª edição, Cengage Learning, 2009.