Notação Assintótica: Ω, Θ

André Vignatti

DINF- UFPR

Limitantes Inferiores

Considere o seguinte trecho de código:

```
void main () {
    /* trecho que le N da entrada padrao */
    for (i = 0 ; i< N; i++)
       puzzle(i);
}

int puzzle (int N) {
   if (N == 1) return 1;
   if (N % 2 == 0) return puzzle(N/2);
   else return puzzle(3*N+1);
}
```

Testar `puzzle(3)`: $3 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$.

Limitantes Inferiores

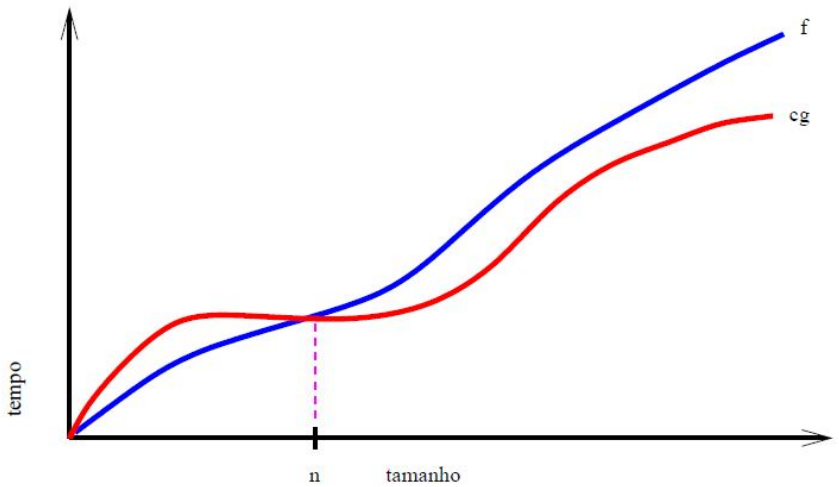
- O programa chama `puzzle` $\geq N$ vezes.
- Cada chamada de `puzzle` executa ≥ 1 comando.
- Qual o **número máximo** de comandos executado por `puzzle`?
  - Difícil responder!

Mas pode ser útil saber um limitante inferior para o número de passos de um algoritmo. Usamos então a notação Ω.

Limitantes Inferiores: Notação Ω

Definição

$f(n)$ é $\Omega(g(n))$ se existe constantes $c > 0$ e n_0 tal que $f(n) \geq cg(n)$, para todo $n \geq n_0$.



$g(n)$ é dito ser um limitante inferior assintótico para $f(n)$.

Exemplo

Seja $f(n) = 5$. É verdade que $f(n) = \Omega(1)$?

SIM. Tomando $c = 1, n_0 = 0$ então $5 \geq c$ para $n \geq n_0$.

Exemplo

Seja $f(n) = \sqrt{n}$. É verdade que $f(n) = \Omega(1)$?

SIM. Tomando $c = 1, n_0 = 1$ então $\sqrt{n} \geq c$ para $n \geq n_0$.

Exemplo

Seja $f(n) = 1/n$. É verdade que $f(n) = \Omega(1)$?

NÃO. Não existe c, n_0 pois $\frac{1}{n}$ sempre pode ser menor que uma constante fixa c.

Exemplo

As funções a seguir são $\Omega(n^2)$

- n^2
- $n^2 + n$
- $n^2 - n$
- $1000n^2 + 1000n$
- $1000n^2 - 1000n$
- n^3
- $n^{2.000000001}$
- 2^{2^n}

Algoritmos e Notações O e Ω

Definição - Tempo de Execução de um Algoritmo:

- Um algoritmo é dito $\Omega(f(n))$ se TODAS as instâncias executam em tempo $\Omega(f(n))$.
- Um algoritmo é dito $O(g(n))$, se TODAS as instâncias executam em tempo $O(g(n))$.

Exemplo

- InsertionSort é $O(n^2)$ pois TODOS os vetores passados como entrada fazem o algoritmo executar em tempo $O(n^2)$.
- InsertionSort NÃO é $O(n)$ pois existem algumas instâncias (vetor ordenado ao inverso) que executam em tempo $O(n^2)$.
- InsertionSort é $\Omega(n)$ pois TODOS os vetores passados como entrada fazem o algoritmo executar em tempo $\Omega(n)$.
- InsertionSort NÃO é $\Omega(n^2)$ pois existem algumas instâncias (vetor ordenado) que executam em tempo $\Omega(n)$.

Algoritmos e Notações O e Ω

Pergunta:

InsertionSort é $O(n^3)$?

Pergunta:

InsertionSort é $\Omega(1)$?

Ambas as respostas são SIM, mas não diz muita coisa sobre o algoritmo...

- É sempre mais interessante encontrar a menor função $f(n)$ tal que o algoritmo é $O(f(n))$.
- É sempre mais interessante encontrar a maior função $g(n)$ tal que o algoritmo é $\Omega(g(n))$.

Algoritmos e Notações O e Ω

Como O limita por cima, pode ter sido feita uma análise folgada.

Exemplo

Supondo análise $O(n^3)$ do InsertionSort (o que NÃO está errado). Podemos melhorar a análise?

- Não conseguimos encontrar vetor que gaste tempo $\Omega(n^3)$!

Como saber que a análise de $O(f(n))$ não está folgada?

Um jeito: achar uma instância que execute em $\Omega(f(n))$.

- Neste caso, não haveria "espaço" para melhorias na análise.

Algoritmos e Notações O e Ω

Exemplo

Supondo análise $O(n^2)$ e $\Omega(n)$ do InsertionSort. Podemos melhorar?

- Vetor ordenado gasta tempo $O(n) \Longrightarrow$ não dá pra aumentar $\Omega(n)$
- Vetor ordenado ao inverso gasta tempo $\Omega(n^2) \Longrightarrow$ não dá pra diminuir $O(n^2)$.

Ao comparar **melhor caso** X **pior caso** dificilmente temos uma análise justa (i.e., sem folgas).

Solução:

Solução: fazer a análise justa do melhor caso separadamente do pior caso.

Análise de Algoritmos não justa (no sentido de apertado)

InsertionSort executa em tempo $O(n^2)$ e $\Omega(n)$.

NÃO dá pra deixar a análise mais justa!

Vamos separar por melhor e pior caso:

Pior Caso

O pior caso do InsertionSort executa em $O(n^2)$ e $\Omega(n^2)$. Isto porque:

- No pior caso, **TODAS** as instâncias executam em $O(n^2)$.
- No pior caso, **TODAS** as instâncias executam em tempo $\Omega(n^2)$.

Melhor Caso

O melhor caso do InsertionSort executa em $O(n)$ e $\Omega(n)$. Isto porque:

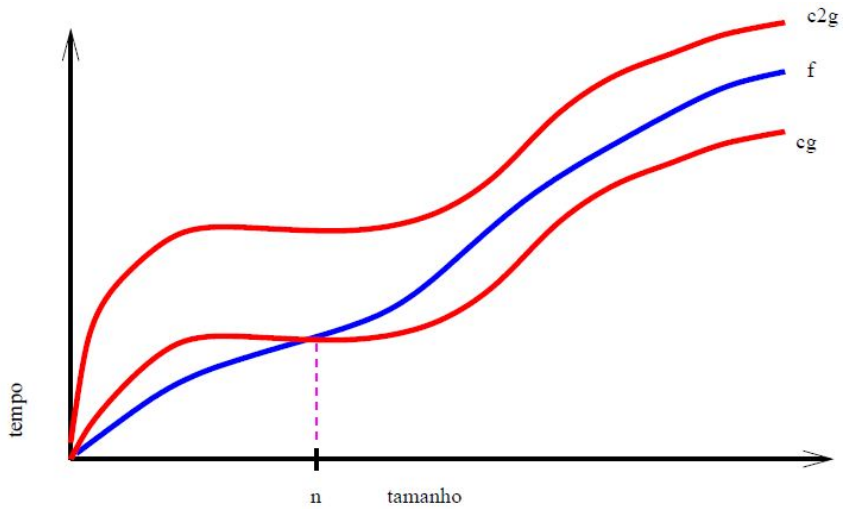
- No melhor caso, **TODAS** as instâncias executam em $O(n)$.
- No melhor caso, **TODAS** as instâncias executam em tempo $\Omega(n)$.

Notação Θ

Definição

$f(n) = \Theta(g(n))$ se $f(n) = O(g(n))$ e $f(n) = \Omega(g(n))$.

- Neste caso, dizemos que $g(n)$ é um limitante assintótico justo (apertado) para $f(n)$.



Definição Equivalente

$f(n)$ é $\Theta(g(n))$ se existe constantes $c_1, c_2 > 0$ e n_0 tal que $c_1 g(n) \leq f(n) \leq c_2 g(n)$, para todo $n \geq n_0$.

Notação Θ

Exemplo

Mostre que a soma dos n primeiros inteiros é $\Theta(n^2)$.

- **Limite Superior**: $1 + 2 + \ldots + n \leq n + n + \ldots + n = n^2$.
  Então, tomando $c = 1, n_0 = 1$, a soma é $O(n^2)$.

- **Limite Inferior**:

$$\begin{aligned} 1 + 2 + \ldots + n &\geq \lceil n/2 \rceil + (\lceil n/2 \rceil + 1) + \ldots + n \\ &\geq \lceil n/2 \rceil + \lceil n/2 \rceil + \ldots + \lceil n/2 \rceil \\ = (n - \lceil n/2 \rceil + 1)\lceil n/2 \rceil & \\ \geq (n/2)(n/2) = n^2/4, & \end{aligned}$$

  assim, tomando $c = 1/4, n_0 = 1$ a soma é $\Theta(n^2)$.

Fato útil: o termo dominante de um polinômio determina sua complexidade assintótica:

Teorema

Seja $f(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0$ *onde* $a_0, \ldots, a_k$ *são números reais, e* $a_k \neq 0$. *Então* $f(n) = \Theta(n^k)$.

Demonstração.

Exercício. □

Observação Importante:

Knuth observou: muito autores usam de forma errada O como se fosse Θ!

- Sempre se lembre disso quando você ver a notação O sendo usada!

Outro **erro clássico**: O serve para expressar o **pior caso**, Ω serve para expressar o **melhor caso**.