ACH2002 - Introdução à Análise de Algoritmos

Notações assintóticas

Prof. Flávio Luiz Coutinho

Notações assintóticas

Notações matemáticas que permitem caracterizar a “cara” de uma função, deixando de fora detalhes desnecessários:

- $T(n) = 5n^3 - 7n^2 + 10n \longrightarrow T(n) = \Theta(n^3)$
- $5n <= T(n) <= 2n \log_2(3n) - 8 \longrightarrow T(n) = \Omega(n)$ e $T(n) = O(n\log_2(n))$

O uso (e a familiaridade) com estas notações, facilita a descrição dos aspectos comportamentais dos algoritmos:

> “O algoritmo XYZ tem complexidade assintótica $\Theta(n^2)$ em relação ao tempo de execução e consumo de memória $\Theta(n)$”

Notações assintóticas

Por que **assintóticas**?

- Porque estamos interessados em entender o comportamento de um algoritmo (isto é, a função associada ao consumo de um certo recurso) em casos limites.
- Conceito de assíntota ($n \longrightarrow 0$, $n \longrightarrow \infty$, etc).
- No contexto específico de algoritmos, estamos interessados em caracterizar o comportamento dos mesmos para valores grandes de n, ou seja $n \longrightarrow \infty$.
- Caracterizar o comportamento de um algoritmo para valores pequenos de n não possui muito valor prático.

Notações assintóticas (notação O)

$$O(g(n)) = \{\ f(n) :\ \exists\ c > 0,\ n_0 > 0 \mid 0 \leq f(n) \leq cg(n), \text{ para todo } n \geq n_0\ \}$$

O que tudo isso quer dizer???

- $O(g(n))$ define um conjunto de funções.
- $g(n)$ é uma função geradora do conjunto.
- As funções membro deste conjunto devem satisfazer uma propriedade:

$0 \leq f(n) \leq cg(n)$ [devem ser limitadas superiormente por $cg(n)$].

- $f(n)$ cresce em ritmo menor ou igual ao ritmo de crescimento de $g(n)$.

Notações assintóticas (notação Ω)

$$\Omega(g(n)) = \{ f(n) : \exists \ c > 0, n_0 > 0 \mid 0 \leq cg(n) \leq f(n), \text{ para todo } n \geq n_0 \}$$

O que tudo isso quer dizer???

- $\Omega(g(n))$ define um conjunto de funções.
- $g(n)$ é uma função geradora do conjunto.
- As funções membro deste conjunto devem satisfazer uma propriedade:

  $0 \leq cg(n) \leq f(n)$ [devem ser limitadas inferiormente por $cg(n)$].

- $f(n)$ cresce em ritmo maior ou igual ao ritmo de crescimento de $g(n)$.

Notações assintóticas (notação Θ)

Notação Θ é a combinação das duas notações anteriores:

$$\Theta(g(n)) = \{\ f(n) : \quad \exists\ c_1 > 0,\ c_2 > 0,\ n_0 > 0 \mid$$

$$0 \le c_1 g(n) \le f(n) \le c_2 g(n)\ ,\ \text{para todo } n \ge n_0\ \}$$

- Funções membro deste conjunto são limitadas inferiormente e superiormente por g(n) de forma simultânea (limite justo).
- f(n) cresce de modo similar à função g(n) [ritmo de crescimento equivalente].

Notações assintóticas

Quando $f(n) \in \Theta(g(n))$ [ou $O(g(n))$, ou $\Omega(g(n))$], dizemos que:

$$f(n) = \Theta(g(n)) \text{ [ou } f(n) = O(g(n)), \text{ ou } f(n) = \Omega(g(n)) \text{]}$$

É um **abuso de notação**, mas aceito no contexto de análise de algoritmos, e pode ser conveniente para certas manipulações matemáticas.

Notações assintóticas

Exemplos de uso das notações:

- $3n^2 - 7n + 15 = \Theta(n^2)$
- $4n^3 + 100n^2 = \Omega(n^2)$, $4n^3 + 100n^2 = O(n^4)$
- $n\log(n) = \Omega(n)$, $n\log(n) = O(n^2)$, $n\log(n) = \Theta(n\log(n))$
- *Quicksort* é $\Omega(n\log(n))$ e $O(n^2)$
- *Insertion sort* é $\Omega(n)$ e $O(n^2)$

Notações assintóticas

Mais exemplos:

$$\begin{aligned} T(n) &= 3n^2 - 7n + 15 \\ &= 3n^2 - 7n + \Theta(1) \\ &= 3n^2 - \Theta(n) \\ &= \Theta(n^2) \end{aligned}$$

Notações assintóticas

Mais exemplos:

$$T(n) = \Theta(n^3) + \Omega(n^2)$$

$$= \Omega(n^2) \qquad (1)$$

$$= \Omega(n^3) \qquad (2)$$

Qual igualdade é mais informativa? (1) ou (2)?

Notações assintóticas

Mais exemplos:

$$T(n) = \Theta(n^3) + O(n^4)$$

$$= O(n^4)$$

Notações assintóticas

Uma propriedade importante:

$$T(n) = \Theta(g(n)) \iff T(n) = \Omega(g(n)) \text{ e } T(n) = O(g(n))$$

Notações assintóticas

Como demonstrar formalmente que $f(n) = \Theta(g(n))$ [ou $\Omega(g(n))$, $O(g(n))$]?

- Devemos achar um conjunto de constantes c_1, c_2 e n_0 [ou c, n_0] que torne verdadeira a condição de pertinências para o conjunto.
- Encontrar apenas um trio (ou dupla) de constantes é suficiente!

E como demonstrar formalmente que $f(n) \neq \Theta(g(n))$ [ou $\Omega(g(n))$, $O(g(n))$]?

- Não basta encontrar um trio (ou dupla) de constantes que não satisfaça a condição de pertinência.
- É preciso demonstrar que é impossível determinar um trio (ou dupla) de constantes que satisfaça a condição de pertinência.

Notações assintóticas

Exemplo: mostrar que $f(n) = 3n^2 - 8n + 15 = \Theta(n^2)$:

- **Estratégia:** fixar c_1 e c_2 (chute "consciente") e verificar para quais valores de n as desigualdades são válidas: $c_1 = 2$, $c_2 = 3$.
- Desigualdade referente ao **limite inferior**:

$$c_1 n^2 <= 3n^2 - 8n + 15$$

$$2n^2 <= 3n^2 - 8n + 15$$

$$0 <= n^2 - 8n + 15$$

$$n^2 - 8n + 15 >= 0$$

$n = (8 \pm 2) / 2$ (raízes da função)

$n_- = 3, n_+ = 5$

Por se tratar de uma função de grau 2, a desigualdade será verdadeira para qualquer n <= 3 ou n >= 5

Notações assintóticas

Exemplo: mostrar que $f(n) = 3n^2 - 8n + 15 = \Theta(n^2)$:

- **Estratégia**: fixar c_1 e c_2 (chute "consciente") e verificar para quais valores de n as desigualdades são válidas: $c_1 = 2$, $c_2 = 3$.
- Desigualdade referente ao **limite inferior**:

$$c_1 n^2 <= 3n^2 - 8n + 15$$

$$2n^2 <= 3n^2 - 8n + 15$$

$$0 <= n^2 - 8n + 15$$

$$n^2 - 8n + 15 >= 0$$

$n = (8 \pm 2) / 2$ (raízes da função)

$n_- = 3$, $n_+ = 5$

Por se tratar de uma função de grau 2, a desigualdade será verdadeira para qualquer ~~n <= 3~~ ou n >= 5

Notações assintóticas

Exemplo: mostrar que $f(n) = 3n^2 - 8n + 15 = \Theta(n^2)$:

- **Estratégia**: fixar c_1 e c_2 (chute "consciente") e verificar para quais valores de n as desigualdades são válidas: $c_1 = 2$, $c_2 = 3$.
- Desigualdade referente ao **limite superior**:

$$3n^2 - 8n + 15 <= 3n^2$$

$$-8n + 15 <= 0$$

$$0 <= 8n - 15$$

$$8n - 15 >= 0$$

$n = 15/8$ (raiz da função)

Por se tratar de uma função de grau 1, a desigualdade será verdadeira para $n >= 15/8$ (1.875)

Notações assintóticas

Exemplo: mostrar que $f(n) = 3n^2 - 8n + 15 = \Theta(n^2)$:

- **Estratégia**: fixar c_1 e c_2 (chute "consciente") e verificar para quais valores de n as desigualdades são válidas: $c_1 = 2$, $c_2 = 3$.
- Desigualdade referente ao **limite superior**:

$$3n^2 - 8n + 15 <= 3n^2$$

$$-8n + 15 <= 0$$

$$0 <= 8n - 15$$

$$8n - 15 >= 0$$

n = 15/8 (raiz da função)

Por se tratar de uma função de grau 1, a desigualdade será verdadeira para n >= 15/8 (1.875)

Notações assintóticas

Exemplo: mostrar que $f(n) = 3n^2 - 8n + 15 = \Theta(n^2)$:

- Combinando os resultados para ambas as desigualdades, obtidas tomando-se $c_1 = 2$ e $c_2 = 3$, temos:
  - Desigualdade do limite inferior válida para $n >= 5$.
  - Desigualdade do limite superior válida para $n >= 1.875$.
  - Para que as duas desigualdades sejam simultaneamente satisfeitas, e sejam sempre satisfeitas a partir de um certo valor, tomamos $n_0 = 5$.
- Visualização gráfica.

Notações assintóticas

Exemplo: mostrar que $f(n) = 5n^3 \neq O(n^2)$:

- Não podemos fixar um valor de c, e verificar que a desigualdade não é atendida para o valor escolhido. Precisamos mostrar que qualquer que seja c, a desigualdade nunca será atendida.
- Desigualdade do limite superior:

$$5n^3 <= cn^2$$

$$5n^3 - cn^2 <= 0$$

$$n^2(5n - c) <= 0$$

$$5n - c <= 0$$

$$n <= c/5$$

Notações assintóticas

Exemplo: mostrar que $f(n) = 5n^3 \neq O(n^2)$:

- Não podemos fixar um valor de c, e verificar que a desigualdade não é atendida para o valor escolhido. Precisamos mostrar que qualquer que seja c, a desigualdade nunca será atendida.
- Desigualdade do **limite superior**:

$$5n^3 <= cn^2$$

$$5n^3 - cn^2 <= 0$$

$$n^2(5n - c) <= 0$$

$$5n - c <= 0$$

$$n <= c/5$$

Desigualdade será verdadeira quando $n <= c/5$. Mas, **independente do valor de c, esse limite sempre irá violar a condição "para todo $n >= n_0$"**. Logo, é **impossível** determinar c e n_0, que satisfaçam a definição.

Revisitando alguns resultados prévios já conhecidos

Algoritmos de ordenação (número de comparações):

- Selection sort $f(n) \sim (n^2) / 2$ $f(n) = \Theta(n^2)$
- Mergesort $f(n) <= n\log_2(n)$ $f(n) = O(n\log_2(n))$

Revisitando alguns resultados prévios já conhecidos

Max (tempo de execução):

- iterativo: $T(n) \sim an$ $T(n) = \Theta(n)$
- rec. 1 [1 subproblema n - 1]: $T(n) \sim bn$ $T(n) = \Theta(n)$
- rec. 2 [2 subproblemas n/2]: $T(n) \sim c(2n - 1)$ $T(n) = \Theta(n)$

Max (memória - apenas memória necessária no call stack):

- iterativo: 1 chamada $M(n) = \Theta(1)$
- rec. 1: n chamadas encadeadas $M(n) = \Theta(n)$
- rec. 2: $\log_2(n) + 1$ chamadas encadeadas $M(n) = \Theta(\log_2(n))$

...

...

Revisitando alguns resultados prévios já conhecidos

Algoritmos de ordenação (número de comparações):

- Algoritmo 1 (selection sort)	$f(n) \sim (n^2) / 2$	$f(n) = \Theta(n^2)$
- Algoritmo 2 (merge sort)	$f(n) <= n\log_2(n)$	$f(n) = O(n\log_2(n))$

Fibonacci:

- iterativo	$T(n) \sim an$	$T(n) = \Theta(n)$
- recursivo 1	$c(\sqrt{2}^n) <= T(n) <= c(2^n)$	$T(n) = \Omega(\sqrt{2}^n)$ e $O(2^n)$
- recursivo 2	$T(n) = c(2n - 1)$	$T(n) = \Theta(n)$

Revisitando alguns resultados prévios já conhecidos

Soma (tempo de execução):

- iterativo: $T(n) \sim an$ $T(n) = \Theta(n)$
- soma 1 [1 subproblema n - 1]: $T(n) \sim bn$ $T(n) = \Theta(n)$
- soma 2 [2 subproblemas n/2]: $T(n) \sim c(2n - 1)$ $T(n) = \Theta(n)$

Soma (memória - apenas memória necessária no call stack):

- iterativo: 1 chamada $M(n) = \Theta(1)$
- soma 1: n chamadas encadeadas $M(n) = \Theta(n)$
- soma 2: $\log_2(n) + 1$ chamadas encadeadas $M(n) = \Theta(\log_2(n))$