Análise Assintótica

Pedro Henrique Del Bianco Hokama
8 de Agosto de 2019

Referências:

- Notas de aulas fortemente baseadas no curso: Stanford Algorithms by Tim Roughgarden
  - `https://www.youtube.com/playlist?list=PLEAYkSg4uSQ37A6_NrUnTHEKp6EkAxTMa`
  - Vídeos: 2.1 até 2.5
- CLRS: Cap 3, 4

1 Análise Assintótica

É o vocabulário usado por todo programador e cientista da computação para discutir em alto nível o desempenho de algoritmos.

- suficientemente simples para ser feita
- poderosa o suficiente para ter um poder preditivo.
- capaz de distinguir entre algoritmos melhores e piores.

Como funciona a análise assintótica?

- suprime constantes e termos de ordem menor (irrelevantes)

Por exemplo dizemos que $6n \log_2 n + 6n$ é $O(n \log n)$ (Big-Oh, Ó Grande, Ózão, ou simplesmente Ó)

Exemplos:

1. (Um laço) Suponha o seguinte algoritmo: Dado um arranjo de n números e um valor t, percorrer cada posição do arranjo procurando por t, devolve Verdadeiro se t está no arranjo e falso caso contrário. Qual é o tempo de execução?
2. (Dois laços em sequencia) Considere ou problema e algoritmo: Dado dois Arranjos A e B de n números e um valor t, percorrer cada posição do arranjo A procurando por t depois percorrer cada posição do arranjo B procurando por t, devolve Verdadeiro se t está em qualquer um dos arranjos e falso caso contrário. Qual é o tempo de execução?
3. (Dois laços aninhados) Considere que são dados dois arranjos A e B e desejamos saber se existe um valos que seja igual em ambos. Usaremos o seguinte algoritmos, para cada posição i percorrer todo arranjo B procurando por $A[i]$. Devolver verdadeiro se existir e falso caso contrário. Qual é o tempo de execução?
4. (Dois laços aninhados) Considere que é dado um arranjo A de n elementos e desejamos saber se existe um valos repetido. Usaremos o seguinte algoritmos, para cada posição i percorrer o arranjo A da posição $i + 1$ até n procurando por $A[i]$. Devolver verdadeiro se existir e falso caso contrário. Qual é o tempo de execução?

1.1 Formalizando o O

Seja $T(n)$ uma função em $n = 1, 2, 3, \ldots$. Quando podemos dizer que $T(n)$ é $O(f(n))$? Se eventualmente para todos os valores de n suficientemente grandes, $T(n)$ é limitado superiormente por uma constante vezes $f(n)$, formalmente:

Definição 1.1: $T(n) = O(f(n))$ se e somente se exitem constantes c, $n_0 > 0$ tais que

$$T(n) \leq c.f(n)$$

para todo $n \geq n_0$.

Exemplos:

1. Provar que $T(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0$ então $T(n) = O(n^k)$

   Para provar precisamos exigir constantes c e $n_0 > 0$ tais que $T(n) \leq c.f(n)$ para todo $n \geq n_0$. Vamos então escolher $n_0 = 1$ e $c = |a_k| + |a_{k-1}| + \ldots + |a_1| + |a_0|$.

$$\begin{aligned} T(n) &= a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 \\ &\leq |a_k| n^k + |a_{k-1}| n^{k-1} + \ldots + |a_1| n + |a_0| \\ &\leq |a_k| n^k + |a_{k-1}| n^k + \ldots + |a_1| n^k + |a_0| n^k \text{ (válido para } n > n_0) \\ &= (|a_k| + |a_{k-1}| + \ldots + |a_1| + |a_0|) n^k \\ &= c n^k \end{aligned}$$

   portanto,

$$T(n) \leq c n^k$$

   logo,

$$T(n) = O(n^k)$$

2. Provar que para todo $k \geq 1$, n^k não é $O(n^{k-1})$.

   Podemos provar por contradição, ou seja, assumimos que a premissa é verdadeira porém a conclusão é falsa e chegamos a algum absurdo. Então supomos que existe um $k \geq 1$ e constantes c, $n_0 > 0$ tal que $T(n) = n^k = O(n^{k-1})$, ou seja,

$$\begin{aligned} n^k &\leq c n^{k-1} && \forall n \geq n_0 \\ n n^{k-1} &\leq c n^{k-1} && \forall n \geq n_0 \\ n \cancel{n^{k-1}} &\leq c \cancel{n^{k-1}} && \forall n \geq n_0 \\ n &\leq c && \forall n \geq n_0 \end{aligned}$$

   Como n pode ser todos os naturais maiores do que n_0, não pode existir tal constante c que seja maior do que qualquer natural. Portanto chegamos a um absurdo e provamos que todo $k \geq 1$, n^k não é $O(n^{k-1})$.

1.2 Ω e Θ

Similar à notação O, que define um limitante superior, temos a notação Ω que define um limitante inferior. Formalmente:

Definição 1.2: $T(n) = \Omega(f(n))$ se e somente se exitem constantes c, $n_0 > 0$ tais que

$$T(n) \geq c.f(n)$$

para todo $n \geq n_0$.

A notação Θ indica que uma função $T(n)$ é limitada inferiormente e superiormente por uma função $f(n)$, ou seja, se $T(n) = O(f(n))$ e também se $T(n) = \Omega(f(n))$. Formalmente:

Definição 1.3: $T(n) = \Theta(f(n))$ se e somente se exitem constantes c_1, c_2 e $n_0 > 0$ tais que

$$c_1.f(n) \leq T(n) \leq c_2.f(n)$$

para todo $n \geq n_0$.

1.3 o e ω

A notação o (Little-Oh, ózinho, ó pequeno) é semelhante a notação O porém não assintoticamente justo. Informalmente pode-ser pensar que O está para uma relação do tipo $\leq$, enquanto o o é uma relação do tipo $<$.

Definição 1.4: $T(n) = o(f(n))$ se e somente se para toda constante $c > 0$, existe um $n_0 > 0$ tal que

$$T(n) < c.f(n)$$

para todo $n \geq n_0$.

Exercício: Provar que para todo $k \geq 1$, $n^{k-1} = o(n^k)$.

Analogamente, a notação ω (Little-omega, omeguinha, omega pequeno) é semelhante a notação Ω porém não assintoticamente justo. Informalmente pode-ser pensar que Ω está para uma relação do tipo $\geq$, enquanto a ω é uma relação do tipo $>$.

Definição 1.5: $T(n) = \omega(f(n))$ se e somente se para toda constante $c > 0$, existe um $n_0 > 0$ tal que

$$T(n) > c.f(n)$$

para todo $n \geq n_0$.

Exemplos:

(a) Provar que $2^{n+10} = O(2^n)$.
Para provar essa afirmação basta exibir c e $n_0 > 0$ tais que $2^{n+10} \leq c.2^n$ para todo $n \geq n_0$. Note que $2^{n+10} = 2^{10}2^n$, então se escolhemos $c = 2^{10}$ e $n_0 = 1$, fica evidente que $2^{n+10} \leq 2^{10}.2^n$ para todo $n \geq 1$, e portanto a afirmação é verdadeira.

(b) Provar que 2^{10n} não é $O(2^n)$.
Suponha por absurdo (por contradição) que $2^{10n} = O(2^n)$ e portanto $2^{10n} < c2^n$ para alguma constante c e $n \geq n_0$. Então

$$\begin{aligned} 2^{10n} &< c2^n \\ 2^{9n}2^n &< c2^n \\ 2^{9n}\cancel{2^n} &< c\cancel{2^n} \\ 2^{9n} &< c \end{aligned}$$

obviamente não existe uma constante que seja maior que 2^{9n} para todos os naturais n, portanto chegamos a uma contradição. Logo 2^{10n} não pode ser $O(2^n)$.

(c) Para quaisquer dois pares de funções positivas, $f(n)$ e $g(n)$, provar que $\max\{f(n), g(n)\} = \Theta(f(n) + g(n))$.

Para mostrar que a afirmação é verdadeira, podemos escolher $c_1 = 1/2$, $c_2 = 1$ e $n_0 = 1$ e verificamos que:

$$\frac{1}{2}(f(n) + g(n)) \leq \max\{f(n), g(n)\} \leq f(n) + g(n) \qquad \forall n \geq n_0$$

1.4 Usando limites

$f(n) \in o(g(n))$ se $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

$f(n) \in O(g(n))$ se $\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$.

$f(n) \in \Theta(g(n))$ se $0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$.

$f(n) \in \Omega(g(n))$ se $\lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$.

$f(n) \in \omega(g(n))$ se $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$.

Exemplo: $f(n) = \ln n$ e $g(n) = n^e$.

$$\lim_{n\to\infty} \frac{\ln n}{n^2} = \frac{1/n}{e.n^{e-1}} = 0.$$