Projeto e Análise de Algoritmos
Fábio Alves de Freitas

1. Seja $f(n)$ e $g(n)$ funções assintoticamente não negativas. Usando a definição básica de Θ, prove que $max(f(n), g(n)) = \Theta(f(n) + g(n))$.

   Resposta: Para provar a proposição, temos de encontrar constantes c_1, c_2 e n_0 que satisfaçam a inequação:

$$0 \leq c_1 * (f(n) + g(n)) \leq max(f(n), g(n)) \leq c_2 * (f(n) + g(n)), \forall n \geq n_0 \quad (1)$$

   Podemos observar que $f(n) \leq max(f(n), g(n))$, visto que:

   1. caso o $g(n) \leq f(n)$, então $f(n) = max(f(n), g(n))$
   2. caso o $g(n) > f(n)$, então $f(n) < max(f(n), g(n))$

   A mesma analogia pode ser aplicada para $g(n) \leq max(f(n), g(n))$.

   Somando os termos de ambas as inequações anteriores, temos que:

$$f(n) + g(n) \leq 2 * max(f(n), g(n))$$

$$\frac{1}{2} * (f(n) + g(n)) \leq max(f(n), g(n)) \quad (2)$$

   Visto que $f(n)$ e $g(n)$ são assintoticamente não negativas, então $f(n) \leq f(n) + g(n)$ ou $g(n) \leq f(n) + g(n)$. A partir disto podemos afirmar que:

$$max(f(n), g(n)) \leq f(n) + g(n) \quad (3)$$

   pois todas as saídas da função max pertencem ou a $f(n)$ ou a $g(n)$, cujos valores são sempre menores ou iguais a $f(n) + g(n)$, de acordo com as inequações anteriores.

   Desta forma, podemos escrever a inequação 1 como uma composição das inequações 2 e 3

$$0 \leq \frac{1}{2} * (f(n) + g(n)) \leq max(f(n), g(n)) \leq 1 * (f(n) + g(n)) \quad (4)$$

   Portanto, a proposição $max(f(n), g(n)) = \Theta(f(n) + g(n))$ é verdadeira para $c_1 = \frac{1}{2}$, $c_2 = 1$ e para n suficientemente grandes.

Fábio Alves de Freitas: fbio.alves095@gmail.com

2. Mostre que para qualquer constante real a e b, onde $b > 0$, $(n+a)^b = \Theta(n^b)$.

Resposta: Para provar a proposição, temos de encontrar contantes c_1, c_2 e n_0 que satisfaçam a inequação:

$$0 \leq c_1 * n^b \leq (n+a)^b \leq c_2 * n^b, \forall n \geq n_0 \tag{5}$$

Analisando o $n+a$, pode-se observar que:

- $n + a \leq n + |a| \leq 2n$, quando $|a| \leq n$
- $n + a \geq n - |a| \geq \frac{n}{2}$, quando $|a| \leq \frac{n}{2}$

Visto que o todos o valores presentes em de $|a| \leq \frac{n}{2}$ também estão contidos em $|a| \leq n$, então ambas as inequações anteriores são válidas para $|a| \leq \frac{n}{2}$, que pode ser reescrita como $n \geq 2|a|$.

Assumindo que $n_0 = 2|a|$, então quando $n \geq 2|a|$, temos que:

$$0 \leq \frac{n}{2} \leq n + a \leq 2n$$

Visto que $b > 0$, podemos elevar cada componente da inequação acima por b sem torná-la falsa.

$$0^b \leq (\frac{n}{2})^b \leq (n+a)^b \leq (2n)^b$$

$$0 \leq \frac{n^b}{2^b} \leq (n+a)^b \leq 2^b * n^b$$

$$0 \leq \frac{1}{2^b} * n^b \leq (n+a)^b \leq 2^b * n^b$$

Desta forma a proposição $(n+a)^b = \Theta(n^b)$ é verdadeira, pois as contantes $c_1 = \frac{1}{2^b}$, $c_2 = 2^b$ e $n \geq 2|a|$ satisfazem a inequação 5.

Fábio Alves de Freitas: fbio.alves095@gmail.com

3. Explique por que a afirmação "O tempo para rodar um algoritmo A é menor que $O(n^2)$" não tem sentido.

   Resposta: A notação O (Big O) é utilizada para para indicar o limite assintótico superior, estreito ou não estreito. Se considerarmos $T(n)$ o tempo do algoritmo A executar, então para valores de n tendendo ao infinito ele demandará $T(n) \leq O(n^2)$ para executar. Visto que a afirmação do problema só possui um sinal de menor que ("$T(n) < O(n^2)$"), então a afirmação fica sem sentido, pois ele indica que a função só é capaz de atingir o limite assintótico superior não estreito e o O exige que a função também possa atingir o limite assintótico superior estreito.

Fábio Alves de Freitas: fbio.alves095@gmail.com

4. As afirmações estão corretas? Justifique.

   (a) $2^{n+1} = O(2^n)$

   Resposta: Temos de achar um c e n_0, tal que:

$$0 \leq 2^{n+1} \leq c * 2^n, \forall n \geq n_0$$

   Se considerarmos $c = 2$ e $n_0 = 0$ a proposição torna-se verdadeira, pois:

$$0 \leq 2^{n+1} \leq 2 * 2^n, \forall n \geq 0$$

$$0 \leq 2^{n+1} \leq 2^{n+1}, \forall n \geq 0$$

   Desta forma, $2^{n+1} = O(2^n)$ é verdade, pois para $c = 2$ e $n_0 = 0$ a definição de O funciona.

   (b) $2^{2^n} = O(2^n)$

   Resposta:

   Suponha que $x = 2^n$, então a proposição seria $2^x = O(x)$, e para prová-la temos de mostrar que existem c e n_0 tais que:

$$2^x \leq c * x, \forall n \geq n_0$$

$$\log_2(2^x) \leq \log_2(c * x)$$

$$x \leq \log_2(c) + \log_2(x)$$

$$x - \log_2(x) \leq \log_2(c) \tag{6}$$

   A afirmação da equação 6 torna-se uma contradição, pois quando n tende ao infinito, então a função $x - \log_2(x)$ será maior $\log_2(c)$, independente do valor de c. Desta forma, $2^{2^n} \neq O(2^n)$

Fábio Alves de Freitas: fbio.alves095@gmail.com

5. Prove que o custo em tempo para se rodar um algoritmo é $\Theta(g(n))$ se e somente se seu pior caso é $O(g(n))$ e seu melhor caso é $\Omega(g(n))$

   Resposta: Para provar essa proposição, devemos mostrar que ambas as inequações abaixo são verdadeiras:

$$f(n) = \Theta(g(n)) \rightarrow f(n) = O(g(n)) \;\; \& \;\; f(n) = \Omega(g(n)) \tag{7}$$

$$f(n) = O(g(n)) \;\; \& \;\; f(n) = \Omega(g(n)) \rightarrow f(n) = \Theta(g(n)) \tag{8}$$

   (a) **Provando a proposição 7**:

   Pela definição de Θ temos que: $f(n) = \Theta(g(n))$, se $0 \leq c_1 * g(n) \leq f(n) \leq c_2 * g(n), \forall n \geq n_0$.

   Podemos provar que $f(n) = O(g(n))$, assumindo que $c = c_2$ (a constante superior da definição do Θ), pois a partir disso chegamos a definição de O:

$$0 \leq f(n) \leq c_2 * g(n), \forall n \geq n_0$$

   De forma semelhante, podemos provar que $f(n) = \Omega(g(n))$, assumindo que $c = c_1$ (a constante inferior da definição do Θ), pois a partir disso chegamos a definição de Ω:

$$0 \leq c_1 * g(n) \leq f(n), \forall n \geq n_0$$

   Considerando o n_0 das definições do O e Ω igual ao da definição de Θ, então a proposição 7 torna-se verdadeira.

   (b) **Provando a proposição 8**:

   Considerando as definições de Ω e O, respectivamente, temos que:

$$0 \leq k_1 * g(n) \leq f(n), \forall n \geq n_1 \tag{9}$$

$$0 \leq f(n) \leq k_2 * g(n), \forall n \geq n_2 \tag{10}$$

   Podemos os valores do n_1 e n_2 como $max(n_1, n_2)$. Se considerarmos na definição de Θ, que $n_0 = max(n_1, n_2)$ as contantes $c_1 = k_1$ e $c_2 = k_2$, então temos que:

$$0 \leq k_1 * g(n) \leq f(n) \leq k_2 * g(n) \leq, \forall n \geq max(n_1, n_2)$$

   Desta forma, pudemos chegar na definição de Θ a partir das definições de Ω e O, logo a proposição 8 é verdadeira

(c) Como ambas as proposições 7 e 8 são verdadeiras, então a proposição "$f(n) = \Theta(g(n))$ se e somente se $f(n) = O(g(n))$ & $f(n) = \Omega(g(n))$" é verdadeira.

Fábio Alves de Freitas: fbio.alves095@gmail.com