Primeira Prova - Análise de Algoritmos
Prof. André Guedes
16 de outubro de 2024

"Teorema Mestre": Sejam $a \geq 1$, $b > 1$, $T, f\colon \mathbb{N} \to \mathbb{R}$ tais que $T(n) = aT(n/b) + f(n)$, então

$$T(n) = \begin{cases} \Theta(n^{\log_b a}), & \text{se } f(n) = \mathcal{O}(n^{\log_b a - \epsilon}), \text{ para algum } \epsilon > 0, \\ \Theta(n^{\log_b a} \log n), & \text{se } f(n) = \Theta(n^{\log_b a}), \\ \Theta(f(n)), & \text{se } f(n) = \Omega(n^{\log_b a + \epsilon}), \text{ para algum } \epsilon > 0 \text{ e} \\ & af(n/b) < cf(n) \text{ assintoticamente para algum } c < 1. \end{cases}$$

1. Dois algoritmos A e B para um certo problema computacional tem seus tempos de execução de pior caso expressos em função de um parâmetro n da entrada pelas funções $T_A(n)$ e $T_B(n)$. Indique se cada uma das afirmações abaixo é correta ou não, justificando sua resposta.

   (a) (10 pontos) Saber que $T_A(n) = \Omega(n^2)$ e $T_B(n) = \Omega(2^n)$ é suficiente para afirmar que o Algoritmo A é melhor que o Algoritmo B na análise de pior caso.

   (b) (10 pontos) Se o Algoritmo A é melhor que o Algoritmo B na análise de pior caso então podemos dizer que $T_A(n) = o(T_B(n))$.

   (c) (10 pontos) Se $T_A(n) = \Theta(T_B(n))$ então $T_B(n) = \Theta(T_A(n))$.

2. Sabendo que $h(n) = 5\frac{3^n}{n^4}$, indique se cada uma das afirmações abaixo é verdadeira ou falsa, justificando sua resposta.

   (a) (10 pontos) $h(n) = o(3^n)$.

   (b) (10 pontos) $h(n) = \mathcal{O}(2^n)$.

   (c) (10 pontos) $h(n) = \mathcal{O}(3^{n-\epsilon})$, para algum $\epsilon > 0$.

3. (10 pontos) Sejam f e g funções $\mathbb{N} \to \mathbb{R}$ tais que $g(n) = o(f(n))$. Considere a função $f{+}g(n) = f(n) + g(n)$. É verdade que $f{+}g(n) = \Theta(f(n))$? Justifique.

4. (15 pontos) Qual o tempo de execução, em função do número de elementos do vetor de entrada, na análise de pior caso do algoritmo abaixo. Justifique sua resposta.

```
X(l, r, v)
  // v é vetor com índices de l até r.
  Para i = l até r − 1
      Para j = i + 1 até r
          Escreva(v[i] + v[j])
```

5. (15 pontos) Considere o seguinte algoritmo que resolve o problema de potenciação (calcular y^z, para $y \neq 0$ e $z \geq 0$):

```
Potencia(y, z)
  // calcula y^z, para y ≠ 0 e z ≥ 0.
  Se z = 0
      Devolva 1
  Se z é ímpar
      Devolva Potencia(y * y,z/2) * y  // divisão inteira.
  Senão
      Devolva Potencia(y * y,z/2)
```

Expresse o tempo de execução deste algoritmo em termos assintóticos, em função de z.

1 Gabarito

1.

(a) Não porque pode ser que $T_A(n) = 3^n$ e $T_B(n) = 2^n$.

(b) Não porque se $T_B(n) = 2T_A(n)$ temos que o Algoritmo A é melhor que o Algoritmo B mas não temos que $T_A(n) = o(T_B(n))$.

(c) Sim. Como $T_A(n) = \Theta(T_B(n))$, então existem $c_1, c_2 > 0$ e $n_0 \in \mathbb{N}$ tais que

$$c_1|T_B(n)| \leq |T_A(n)| \leq c_2|T_B(n)|, \text{ para } n \geq n_0.$$

Disso podemos deduzir que

$$\frac{1}{c_2}|T_A(n)| \leq |T_B(n)| \leq \frac{1}{c_1}|T_A(n)|, \text{ para } n \geq n_0.$$

O que implica que $T_B(n) = \Theta(T_A(n))$.
Também podemos fazer usando o fato de que $f(n) \in \mathcal{O}(g(n)) \Leftrightarrow g(n) \in \Omega(f(n))$ e que $\Theta(f(n))$ é igual a $\mathcal{O}(f(n)) \cap \Omega(f(n))$.

2.

(a) Sim porque

$$\frac{h(n)}{3^n} = \frac{5\frac{3^n}{n^4}}{3^n} = \frac{5}{n^4} = o(1).$$

(b) Não porque

$$\frac{h(n)}{2^n} = \frac{5\frac{3^n}{n^4}}{2^n} = \frac{5(\frac{3}{2})^n}{n^4}$$

e

$$\lim \frac{5(\frac{3}{2})^n}{n^4} = \infty.$$

(c) Sim porque

$$\frac{h(n)}{3^{n-\epsilon}} = \frac{5\frac{3^n}{n^4}}{3^{n-\epsilon}} = \frac{5(3^\epsilon)}{n^4} = o(1).$$

(Já que $\frac{3^n}{3^{n-\epsilon}} = 3^{n-n+\epsilon} = 3^\epsilon$ é uma constante.)

3.

Como $g(n) = o(f(n))$, então $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$.

$$\lim_{n\to\infty} \frac{f+g(n)}{f(n)} = \lim_{n\to\infty} \frac{f(n)+g(n)}{f(n)} = \lim_{n\to\infty} 1 + \frac{g(n)}{f(n)}.$$

Como $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$, temos que

$$\lim_{n\to\infty} \frac{f+g(n)}{f(n)} = 1.$$

Portanto, $f+g(n) = \Theta(f(n))$.

4.

O número de elementos do vetor v é $n = r - l + 1$. A última linha custa $\Theta(1)$. O 2o laço, parametrizado por i, executa $r - (i+1) + 1$ vezes, ou seja, as duas últimas linhas juntas custam $(r-i)\Theta(1)$. O 1o laço faz o i variar de l até $r-1$, fazendo $r-l$ voltas. O custo total do algoritmo é

$$T(l,r) = \sum_{i=l}^{r-1} \sum_{i+1}^{r} \Theta(1) \stackrel{\text{(segundo somatório tem } (r-i) \text{ parcelas)}}{=}$$

$$\sum_{i=l}^{r-1} (r-i)\Theta(1) =$$

$$(\sum_{i=l}^{r-1} r - \sum_{i=l}^{r-1} i)\Theta(1) = ((r-l)r - ((r-l)\frac{r-1+l}{2}))\Theta(1) =$$

$$((r-l)(r - \frac{r-1+l}{2}))\Theta(1) = ((r-l)\frac{2r-(r-1+l)}{2}\Theta(1) =$$

$$((r-l)\frac{r-l+1}{2}\Theta(1).$$

Trocando variáveis, fazendo $n = r - l + 1$, temos que

$$T(n) = ((n-1)\frac{n}{2}\Theta(1) = \frac{n(n-1)}{2}\Theta(1) = \Theta(n^2).$$

5.

Como somente uma das duas chamadas recursivas é de fato chamada, temos que $a = 1$.

Como em cada chamada recursiva o valor de z é dividido por 2, temos que $b = 2$.

Como todo o resto de cada chamada tem custo $\Theta(1)$ (testar se z é 0 (zero), ou par, ou ímpar, multiplicações e divisões inteiras), temos que $f(n) = \Theta(1)$.

Logo, se $T(z)$ expressa o tempo de execução deste algoritmo, temos que $T(z) = aT(z/b) + f(n) = T(z/2) + \Theta(1)$.

Como $z^{\log_b a} = z^0 = 1$, pelo Teorema Mestre, caímos no segundo caso e, portanto, $T(z) = \log_2 z = \Theta(\log z)$.