# Projeto e Análise de Algoritmos (2023-1)

Avaliação Escrita

Prof. Flávio L. C. de Moura

18 de maio de 2023

- Por favor, coloque **nome** e **matrícula** em todas as folhas
- A resolução pode ser feita à lápis ou caneta, mas seja organizado.
- Esta avaliação é **individual** e **sem consulta**.
- Início: 19:00
- Término: 20:40
- Segue o enunciado do Teorema Mestre:

  **Teorema 1.** *Sejam $a \geq 1$ e $b > 1$ constantes, $f(n)$ uma função assintoticamente positiva, e $T(n)$ definida nos inteiros não-negativos pela recorrência:*

  $$T(n) = a.T(n/b) + f(n)$$

  *onde $n/b$ deve ser interpretado como $\lfloor n/b \rfloor$ ou $\lceil n/b \rceil$. Então $T(n)$ tem as seguintes cotas assintóticas:*

  1. *Se $f(n) = O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b a})$.*
  2. *Se $f(n) = \Theta(n^{\log_b a})$, então $T(n) = \Theta(n^{\log_b a}. \lg n)$.*
  3. *Se $f(n) = \Omega(n^{\log_b a + \epsilon})$ para alguma constante $\epsilon > 0$, e se $a.f(n/b) \leq c.f(n)$ para alguma constante $c < 1$, então para todo $n$ suficientemente grande, temos que $T(n) = \Theta(f(n))$.*

Resolva as seguintes questões:

1. (2.5 pontos) Sejam $f(n)$ e $g(n)$ funções não-negativas tais que $g(n) = O(f(n))$. Prove, utilizando as definições de notação assintótica, que $f(n) + g(n) = \Theta(f(n))$.

   **Solução**. Por hipótese, temos que

   $$g(n) = O(f(n)) \tag{1}$$

   e queremos mostrar que

   $$f(n) + g(n) = \Theta(f(n)) \tag{2}$$

   ou seja, precisamos encontrar constantes positivas $c_1, c_2$ e $n_0$ tais que

   $$c_1, f(n) \leq f(n) + g(n) \leq c_2.f(n), \forall n \geq n_0$$

   Da hipótese, (1) sabemos que existem constantes positivas $c'$ e $n'$ tais que

   $$g(n) \leq c'.f(n), \forall n \geq n' \tag{3}$$

   e portanto,

   $$f(n) + g(n) \leq (1 + c').f(n), \forall n \geq n'.$$

   Ou seja, $f(n) + g(n) = O(f(n))$. Adicionalmente, $f(n) + g(n) \geq f(n), \forall n$, e assim, podemos tomar $c_1 = 1$, $c_2 = 1 + c'$ e $n_0 = n'$. □

2. (2.5 pontos) Considere o problema dos elementos únicos, que checa se todos os $n$ elementos de um vetor de tamanho são distintos:

```
1 for i = 0 to n − 2 do
2 |   for j = i + 1 to n − 1 do
3 |   |   if A[i] = A[j] then
4 |   |   |   return false
5 |   |   end
6 |   end
7 end
8 return true
```

**Algoritmo 1:** EUnicos($A[0..n-1]$)

Faça a análise assintótica deste algoritmo, e justifique sua resposta.

**Solução**. Seja $T(n)$ o número de comparações executadas por este algoritmo (linha 3). O número de comparações é interrompido assim que dois elementos iguais são encontrados (linha 4), e portanto, no melhor caso apenas uma comparação é feita: por exemplo, quando os dois primeiros elementos do vetor $A$ são iguais. Neste caso, $T(n) = \Theta(1)$. No pior caso, temos $T(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2}(n-1-i) = \frac{(n-1).n}{2}$. Logo $T(n) = \Theta(n^2)$, ou seja, o algoritmo é quadrático no tamanho da entrada. □

3. Considere um algoritmo recursivo hipotético que resolve uma classe de problemas da seguinte forma:

   (a) A instância do problema de tamanho $n$ recebida como entrada é dividida em 4 subproblemas de tamanho $n/2$, que são por sua vez resolvidas recursivamente;

   (b) As soluções dos 4 subproblemas são combinadas em tempo $n^2.\lg n$, para que seja possível formar uma solução do problema original.

   Responda os itens a seguir:

   (a) (1.0 ponto) Escreva a recorrência que modela a complexidade deste algoritmo.
   **Solução**.
   $$T(n) = 4.T(n/2) + n^2.\lg n$$

   (b) Determine a complexidade assintótica deste algoritmo hipotético de acordo com os seguintes passos:

      i. (1.5 pontos) O Teorema Mestre pode ser utilizado para resolver a recorrência do item anterior? Justifique sua resposta.
      **Solução**. O teorema mestre não se aplica a esta recorrência. De fato, o caso 3 seria o candidato para resolver esta recorrência, mas para isto precisamos que $n^2.\lg n = \Omega(n^{\lg 4+\epsilon}) = \Omega(n^{2+\epsilon})$ para alguma constante $\epsilon > 0$, e isto não ocorre: de fato, para que $n^2.\lg n = \Omega(n^{2+\epsilon})$ precisam existir constantes positivas $c_0$ e $n_0$ tais que $n^2.\lg n \geq c_0.n^{2+\epsilon}, \forall n \geq n_0$ o que equivale a $\lg n \geq c_0.n^{\epsilon}, \forall n \geq n_0$, o que é um absurdo já que $\lg n = o(n^{\epsilon})$ para todo $\epsilon > 0$. Alternativamente, podemos simplesmente dizer que $n^2.\lg n$ não é *polinomialmente maior* do que $n^2$.
      Ainda que desnecessário mostrar já que o caso 3 não aplica, mas considerando que a situação apresentada no parágrafo anterior não tenha sido observada, note que a condição de regularidade do caso 3 também não é satisfeita; de fato, para que $4.(n/2)^2.\lg(n/2) \leq c.n^2.\lg n$ para alguma constante $c < 1$ e $n$ suficientemente grande, teríamos que

      $$\begin{array}{ll} n^2.((\lg n) - 1) \leq c.n^2.\lg n & \Rightarrow \\ (\lg n) - 1 \leq c.\lg n & \Rightarrow \\ \lg n \leq \frac{1}{1-c} & \text{(para } n \text{ suficientemente grande)} \end{array}$$

      o que é um absurdo, já que $\lg n$ é uma função crescente, e portanto não pode ser limitada por uma constante. □

ii. (2.5 pontos) Qual é a a complexidade assintótica deste algoritmo hipotético? Justifique sua resposta.
**Solução**. Vamos resolver esta recorrência pelo método da substituição. Para isto, vamos assumir que $n = 2^k$ para algum $k$ positivo, e que $T(1) = 0$. Assim,

$$\begin{aligned}
T(n) &= 4.T(n/2) + n^2.\lg n \\
&= 4^2.T(n/2^2) + n^2.(\lg(n/2) + \lg n) \\
&= 4^3.T(n/2^3) + n^2.(\lg(n/2^2) + \lg(n/2) + \lg n) \\
&= \dots \\
&= 4^k.T(1) + n^2.(\sum_{i=0}^{k-1} \lg(n/2^i)) \\
&= n^2.(\sum_{i=0}^{k-1} (\lg(n) - i)) \\
&= n^2.(\sum_{i=0}^{k-1} \lg(n) - \sum_{i=0}^{k-1} i) \\
&= n^2.(k.\lg(n) - \tfrac{k.(k-1)}{2}) \\
&= n^2.(\lg^2(n) - \tfrac{\lg^2(n) - \lg n}{2}) \\
&= n^2.\tfrac{\lg^2(n) + \lg n}{2}.
\end{aligned}$$

Podemos utilizar indução (forte) para verificarmos que a expressão acima é, de fato, solução da recorrência original:

$$\begin{aligned}
T(n) &= 4.T(n/2) + n^2.\lg n \\
&\overset{h.i.}{=} 4.(\tfrac{n}{2})^2.\tfrac{\lg^2(\frac{n}{2}) + \lg(\frac{n}{2})}{2} + n^2.\lg n \\
&= n^2.\tfrac{((\lg n) - 1)^2 + (\lg n) - 1}{2} + n^2.\lg n \\
&= \tfrac{n^2.\lg^2 n - 2.n^2.\lg n + n^2 + n^2.\lg n - n^2 + 2.n^2.\lg n}{2} \\
&= \tfrac{n^2.(\lg^2 n + \lg n)}{2}.
\end{aligned}$$

Logo $T(n) = \Theta(n^2.\lg^2 n)$.

Alternativamente, podemos calcular cada uma das cotas separadamente utilizando indução (forte). Inicialmente, mostraremos que $T(n) \leq n^2.\lg^2 n$, para $n$ suficientemente grande. Temos,

$$\begin{aligned}
T(n) &= 4.T(n/2) + n^2.\lg n \\
&\overset{h.i.}{\leq} 4.((\tfrac{n}{2})^2.\lg^2(\tfrac{n}{2})) + n^2.\lg n \\
&= n^2.\lg^2(\tfrac{n}{2})) + n^2.\lg n \\
&= n^2.(\lg^2 n - 2.\lg n + 1) + n^2.\lg n \\
&= n^2.\lg^2 n - n^2.((\lg n) - 1) \\
&\leq n^2.\lg^2 n, \text{ para } n \geq 2.
\end{aligned}$$

Portanto, $T(n) = O(n^2.\lg^2 n)$.
Para a cota inferior, mostraremos que $T(n) \geq \frac{n^2}{2}.\lg^2 n$, para $n$ suficientemente grande. Temos,

$$\begin{aligned}
T(n) &= 4.T(n/2) + n^2.\lg n \\
&\overset{h.i.}{\geq} 4.(\tfrac{(\frac{n}{2})^2}{2}.\lg^2(\tfrac{n}{2})) + n^2.\lg n \\
&= \tfrac{n^2}{2}.\lg^2(\tfrac{n}{2})) + n^2.\lg n \\
&= \tfrac{n^2}{2}.(\lg^2 n - 2.\lg n + 1) + n^2.\lg n \\
&= \tfrac{n^2}{2}.\lg^2 n + \tfrac{n^2}{2} \\
&\geq \tfrac{n^2}{2}.\lg^2 n
\end{aligned}$$

Logo, $T(n) = \Omega(n^2.\lg^2 n)$, e portanto $T(n) = \Theta(n^2.\lg^2 n)$. □