Departamento de Ciência da Computação — UnB

Projeto e Análise de Algoritmos — 2026/1

Primeira Avaliação Escrita

Nome: _____ Matrícula: _____

Instruções:

1. Esta avaliação é **individual** e **sem consulta**;
2. Esta prova contém 3 questões totalizando 10,0 pontos;
3. Justifique **todas** as suas respostas com rigor matemático.

Teorema .1. *Seja $T(n)$ uma função eventualmente não-decrescente que satisfaz a recorrência*

$$\begin{cases} T(n) = a.T(n/b) + \Theta(n^d), & \text{para } n = b^k, k = 1, 2, 3, \ldots \\ T(1) = c \end{cases}$$

onde a é um inteiro positivo ($a \geq 1$), b é uma constante real maior do que 1 e $c, d \geq 0$ constantes reais. Então

$$T(n) = \begin{cases} \Theta(n^{\log_b a}), & \text{se } a > b^d \\ \Theta(n^d . \lg n), & \text{se } a = b^d \\ \Theta(n^d), & \text{se } a < b^d \end{cases}$$

Teorema .2. *Sejam a um inteiro positivo e b uma constante real maior do que 1, $f(n)$ 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 $T(n) = \Theta(f(n))$.*

Teorema .3. *Sejam a um inteiro positivo e b uma constante real maior do que 1, $f(n)$ 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 $f(n) = \Theta(n^{\log_b a} . \lg^k n)$ para alguma constante $k \geq 0$, então $T(n) = \Theta(n^{\log_b a} . \lg^{k+1} n)$;*

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* $T(n) = \Theta(f(n))$.

Questão 1 (2.5 pontos)

Sejam $f(n)$ e $g(n)$ funções dos inteiros não-negativos nos reais positivos. Prove, utilizando **apenas as definições** de notação assintótica, que:

$$f(n) = O(g(n)) \text{ e } g(n) = O(f(n)) \implies f(n) = \Theta(g(n)).$$

Gabarito

Por hipótese, $f(n) = O(g(n))$, ou seja, existem constantes $c_1 > 0$ e $n_1 > 0$ tais que

$$f(n) \leq c_1\, g(n), \quad \forall\, n \geq n_1. \tag{1}$$

Analogamente, $g(n) = O(f(n))$ significa que existem constantes $c_2 > 0$ e $n_2 > 0$ tais que

$$g(n) \leq c_2\, f(n), \quad \forall\, n \geq n_2. \tag{2}$$

Queremos mostrar que $f(n) = \Theta(g(n))$, ou seja, que existem constantes positivas $c_{\min}$, $c_{\max}$ e n_0 tais que

$$c_{\min}\, g(n) \leq f(n) \leq c_{\max}\, g(n), \quad \forall\, n \geq n_0.$$

Tome $n_0 = \max\{n_1, n_2\}$. Para todo $n \geq n_0$ valem simultaneamente (1) e (2):

- **Cota superior:** da equação (1), $f(n) \leq c_1\, g(n)$.
- **Cota inferior:** da equação (2), $g(n) \leq c_2\, f(n)$, o que equivale a $\dfrac{1}{c_2}\, g(n) \leq f(n)$.

Portanto, tomando $c_{\min} = 1/c_2$ e $c_{\max} = c_1$, temos

$$\frac{1}{c_2}\, g(n) \leq f(n) \leq c_1\, g(n), \quad \forall\, n \geq n_0,$$

o que demonstra que $f(n) = \Theta(g(n))$. □

Questão 2 (5,0 pontos)

Considere o seguinte algoritmo recursivo que verifica se o vetor $A[1..n]$ está ordenado em ordem não-decrescente:

```
1 if n ≤ 1 then
2 |   return verdadeiro;
3 end
4 if A[n − 1] > A[n] then
5 |   return falso;
6 end
7 return Ordenado(A[1..n − 1]);
```

Algoritmo 1: Ordenado($A[1..n]$)

(a) **(2.0 ponto)** Prove por indução (em n) que Ordenado($A[1..n]$) retorna *verdadeiro* **se e somente se** $A[1..n]$ está ordenado em ordem não-decrescente.

(b) **(1.0 ponto)** Escreva a recorrência que modela o número de comparações realizadas no pior caso;

(c) **(2.0 ponto)** Resolva a recorrência do item anterior e determine a complexidade assintótica do algoritmo.

Gabarito

(a) Prova por indução em n.

Base ($n \leq 1$): Um vetor com 0 ou 1 elemento está trivialmente ordenado, e o algoritmo retorna *verdadeiro*. ✓

Passo indutivo ($n > 1$): Suponha (hipótese de indução) que o algoritmo é correto para vetores de tamanho $n - 1$.

(⇒) Suponha que $A[1..n]$ está ordenado. Então $A[n-1] \leq A[n]$, logo a segunda condição não é ativada. Como $A[1..n-1]$ é prefixo de um vetor ordenado, está ordenado. Pela hipótese de indução, Ordenado($A[1..n-1]$) retorna *verdadeiro*, e portanto o algoritmo retorna *verdadeiro*. ✓

(⇐) Suponha que Ordenado($A[1..n]$) retorna *verdadeiro*. Então a segunda condição não foi ativada, logo $A[n-1] \leq A[n]$. Além disso, Ordenado($A[1..n-1]$) retornou *verdadeiro*, e pela hipótese de indução $A[1..n-1]$ está ordenado. Como $A[1..n-1]$ está ordenado e $A[n-1] \leq A[n]$, conclui-se que $A[1..n]$ está ordenado. ✓

□

(b) O pior caso ocorre quando o vetor está ordenado em ordem não-decrescente (todas as $n-1$ chamadas recursivas chegam à base sem retornar *falso* antecipadamente). Cada chamada realiza exatamente 2 comparações ($n \leq 1$ e $A[n-1] > A[n]$) além de uma chamada recursiva. Portanto o número de comparações satisfaz:

$$T(n) = T(n-1) + \Theta(1), \qquad T(1) = \Theta(1).$$

(c) Aplicando o método da substituição com $T(n) = T(n-1) + c$:

$$T(n) = T(n-1) + c = T(n-2) + 2c = \cdots = T(1) + (n-1)c = \Theta(n).$$

Portanto $T(n) = \Theta(n)$.

Questão 3 (2.5 pontos)

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

1. A instância do problema de tamanho n recebida como entrada é dividida em 3 subproblemas de tamanho $n/4$, que são por sua vez resolvidos recursivamente;
2. As soluções dos subproblemas são combinadas em tempo $n.\lg n$, 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;

(b) **(1.5 pontos)** Resolva a recorrência do item anterior, e determine a complexidade assintótica deste algoritmo.

Gabarito

(a) A descrição corresponde à recorrência:

$$T(1) \in \Theta(1), \qquad T(n) = 3\,T(n/4) + n \lg n.$$

(b) Solução via o Teorema Mestre .2:

Identificamos $a = 3$, $b = 4$ e $f(n) = n \lg n$. Calculamos $n^{\log_b a} = n^{\log_4 3}$.

Como $\log_4 3 < 1$, temos $n^{\log_4 3} = o(n)$, e portanto $f(n) = n \lg n = \Omega(n^{\log_4 3 + \varepsilon})$ para qualquer $0 < \varepsilon < 1 - \log_4 3$ (por exemplo, $\varepsilon = \frac{1-\log_4 3}{2}$, pois $n \lg n \geq n = \Omega(n^{\log_4 3 + \varepsilon})$).

Verificamos a condição de regularidade do caso 3:

$$a \cdot f\left(\frac{n}{b}\right) = 3 \cdot \frac{n}{4} \lg \frac{n}{4} = \frac{3n}{4}(\lg n - 2) \leq \frac{3}{4}\, n \lg n = c \cdot f(n), \quad c = \tfrac{3}{4} < 1.$$

Portanto, pelo caso 3 do Teorema Mestre .2:

$$\boxed{T(n) = \Theta(f(n)) = \Theta(n \lg n).}$$

Solução alternativa: **Método da árvore de recorrência.**

No nível j da árvore há 3^j subproblemas, cada um de tamanho $n/4^j$. O custo de combinação em cada subproblema desse nível é $f(n/4^j) = (n/4^j)\,\lg(n/4^j) = (n/4^j)(\lg n - 2j)$. O custo *total* do nível j é portanto:

$$3^j \cdot \frac{n}{4^j}\,(\lg n - 2j) = n\left(\frac{3}{4}\right)^j (\lg n - 2j).$$

A árvore tem altura $\log_4 n$ (o subproblema tem tamanho 1 quando $n/4^j = 1$, i.e., $j = \log_4 n$). O custo total é:

$$T(n) = \sum_{j=0}^{\log_4 n} n\left(\tfrac{3}{4}\right)^j (\lg n - 2j).$$

Cota superior. Como $\lg n - 2j \leq \lg n$ para todo $j \geq 0$:

$$T(n) \leq n \lg n \sum_{j=0}^{\infty} \left(\tfrac{3}{4}\right)^j = n \lg n \cdot \frac{1}{1 - 3/4} = 4n \lg n = O(n \lg n).$$

Cota inferior. O termo $j = 0$ sozinho já dá $n \lg n$, logo:

$$T(n) \geq n \lg n = \Omega(n \lg n).$$

Das duas cotas, concluímos:

$$\boxed{T(n) = \Theta(n \lg n).}$$

Observe que o custo é dominado pela raiz da árvore (nível $j = 0$), pois a série geométrica de razão $3/4 < 1$ converge: os níveis inferiores contribuem com um fator constante apenas. □