UNIVERSIDADE FEDERAL ALFENAS (UNIFAL)
Bacharelado em Ciência da Computação

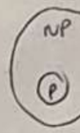
Disciplina	Método de realização	Data da prova
DCE529 - Algoritmos e Estrutura de Dados III	Presencial	12/03/2025 às 08h00

Professor
Iago Augusto de Carvalho (iago.carvalho@unifal-mg.edu.br)

Prova 01 - Complexidade

Exercício 1 (10%)

Apresente (em formato de pseudo-código) um algoritmo não determinístico que encontra o menor número de uma matriz tri-dimensional em tempo $\mathcal{O}(1)$



Exercício 2 (35%)

Diga se cada afirmação é verdadeira ou falsa e justifique

a) Se $f_1(n) = \mathcal{O}(g(n))$ e $f_2(n) = \mathcal{O}(g(n))$, então $f_1(n)f_2(n) = \mathcal{O}(g(n))$ ✓

b) Uma Máquina de Turing não determinística não é capaz de rodar algoritmos determinísticos F

c) Todo problema pertencente a P também pertence a NP √

d) Se $f(n) = \mathcal{O}(n^2)$ e $g(n) = \mathcal{O}(n)$, então $f(n) + g(n) = \mathcal{O}(n^3)$ ×

e) Problemas #P-Completos são tão difíceis quanto problemas NP-Completos

f) Um algoritmo com complexidade $(2)^{\frac{2}{n}}$ é polinomial

g) Se $f_1(n) = \mathcal{O}(g_1(n))$ e $f_2(n) = \mathcal{O}(g_2(n))$, então $f_1(n) + f_2(n) = \mathcal{O}(|g_1(n)|\,|g_2(n)|)$ ✗

Exercício 3 (15%)

Quais são as duas maneiras de mostrar que um problema pertence a NP?

Exercício 4 (30%)

Resolva as seguintes equações de recorrência utilizando o teorema mestre. Apresente qual caso do teorema mestre foi utilizado, os valores de a, b, $f(n)$, e dê, ao final, a complexidade do algoritmo

a) $T(n) = T\left(\frac{5n}{8}\right) + n$

b) $T(n) = 4T\left(\frac{n}{4}\right) + 1$

c) $T(n) = 9T\left(\frac{n}{3}\right) + n^2$

Exercício 5 (15%)

Considere um vetor de n posições contendo números inteiros. Responda verdadeiro ou falso. Se falso, justifique. Se verdadeiro, escreva o algoritmo em pseudo-código

a) Um algoritmo não-determinístico encontra o menor valor deste vetor em tempo polinomial

b) Um algoritmo determinístico lista todas as permutações deste vetor em tempo polinomial

c) Um algoritmo determinístico soma todos os valores deste vetor em tempo polinomial

78

$a f\left(\frac{n}{b}\right) \leq c f(n) \rightarrow \frac{5}{8} \leq c$

$\frac{n}{b} \leq cn$

$\frac{5n}{8} \leq cn$

30 (4) a) $T(n) = T\left(\frac{5n}{8}\right) + n$

c

$\begin{cases} a = 1 \\ b = 8/5 \\ f(n) = n \end{cases}$

$n^{\log_b a} = n^{\log_{8/5} 1} = n^0 = 1$

$n^{\log_b a}$ cresce mais lentamente que $f(n)$

↳ caso 3,,

$f(n) = \Omega(n^{\log_b a + \epsilon})$

$, \Omega(n^{0+\epsilon})$

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

$= \Theta(n)$

c b) $T(n) = 4T\left(\frac{n}{4}\right) + 1$

$\begin{cases} a = 4 \\ b = 4 \\ f(n) = 1 \end{cases}$

$n^{\log_b a} = n^{\log_4 4} = n^1$

~~$a f\left(\frac{n}{b}\right) \leq c f(n)$~~

~~$4 \cdot 1 \leq c \cdot 1$~~

~~$5 \leq c$~~

$n^{\log_b a}$ cresce mais rápido que $f(n)$

~~$T(n) = \Theta(f(n))$~~

~~$= \Theta(1)$~~

~~caso 3~~

↳ caso 1,,

$f(n) = O(n^{\log_b a - \epsilon})$

$= O(n^{1-\epsilon})$

$T(n) = \Theta(n^{\log_b a}) =$

$= \Theta(n^1)$

c c) $T(n) = 9T\left(\frac{n}{3}\right) + n^2$

$\begin{cases} a = 9 \\ b = 3 \\ f(n) = n^2 \end{cases}$

$n^{\log_b a} = n^{\log_3 9} = n^2$

$n^{\log_b a}$ cresce o mesmo que $f(n)$

↳ caso 2,,

$f(n) = \Theta(n^{\log_b a})$

~~$T(n) = (n^{\log}$~~ $f(n) = \Theta(n^2)$

$T(n) = (n^{\log_b a} \log n)$

$T(n) = (n^2 \log n)$

15 (3) 1. Apresentar um algoritmo não determinista que faça em tempo polimonial para resolver um determinado problema

2. Encontrar um algoritmo determinista polimonial para verificar que uma determinada solução é válida

10 ① Algoritmo Encontrar Menor Elemento Matriz 3D:

Entrada: Matriz tridimensional M de tamanho $n \times n \times n$

Saída: Menor número em M

1. Para cada índice (i, j, k) na matriz M:
   2. Adivinhe a posição (i, j, k) do menor elemento
   3. Verifique se M[i][j][k] é o menor número encontrado
4. Retorne o menor número encontrado

10 ⑤ ex a) Verdadeiro. Em um algoritmo não-determinista podemos "adivinhar" a posição do menor valor, o que permite que o algoritmo ~~et~~ procure de maneira instantanea todas as possibilidades

e b) Falso. O número de permutacões é $n!$ que cresce de maneira muito mais rápido que uma função polimonial. $O(n!)$ é exponencial e não polimonial

ex c) Verdadeiro. Somar todos os valores de um vetor n é uma tarefa simples, dessa forma a operação tem complexidade $\Theta(n)$ que é uma complexidade polimonial

13 ② x a) verdadeiro

x b) falso. Uma máquina não deterministica é capaz de rodar algoritmos deterministicos por terem simples complexidade

c c) verdadeiro. $P \subseteq NP$

c d) falso. $f(n) + g(n) = O(n^2)$ uma vez que se desprezam termos irrelevantes na situação, que é o caso de "n" em "$n^2 + n$"

ex e) verdadeiro

x f) falso. $2^{\frac{2}{n}}$ descreve uma função que decai rapidamente enquanto n cresce, não sendo polimonial

x g) verdadeiro