



UNIVERSIDADE FEDERAL ALFENAS (UNIFAL)
Bacharelado em Ciência da Computação

Disciplina DCE529 - Algoritmos e Estrutura de Dados III	Método de realização Presencial	Data da prova 09/04/2025 às 08h00
Professor Iago Augusto de Carvalho (iago.carvalho@unifal-mg.edu.br)		

Prova 02

Exercício 1 (15%)

Seja um grafo bipartido, sendo que a primeira partição tem 4 vértices e a segunda tem 5 vértices. Caso ele seja não direcionado, qual é o maior número de arestas que este grafo pode ter? Desenhe o grafo resultante.

Exercício 2 (20%)

Considere o problema do caixeiro viajante, como definido para o segundo trabalho prático. Considere também que este problema seja definido em um grafo completo não direcionado com pesos nas arestas. Apresente, em formato de pseudo-código, um algoritmo de força-bruta que resolve este problema.

Exercício 3 (25%)

No problema da torre de Hanói, devemos movimentar discos de diversos tamanhos localizados em duas ou mais hastes de forma a construir uma única torre em uma haste. Neste problema, cada disco subsequente da torre tem que ter um tamanho menor que o disco anterior.

Considere que $p(n)$ seja o número mínimo de movimentos para construir a torre de Hanói utilizando n discos. Sabendo-se que só um disco deve ser movimentado por vez e que é proibido colocar um disco maior sobre um disco menor, temos que $p(1) = 1$ e que $p(n + 1) = 2p(n) + 1$

- (10%) Apresente o pseudo-código de um algoritmo recursivo para calcular $p(n)$
- (10%) Apresente o pseudo-código de um algoritmo não recursivo para calcular $p(n)$
- (5%) Dê o valor de $p(6)$

Exercício 4 (10%)

Apresente a matriz e a lista de adjacência para um grafo não direcionado completo com 5 vértices

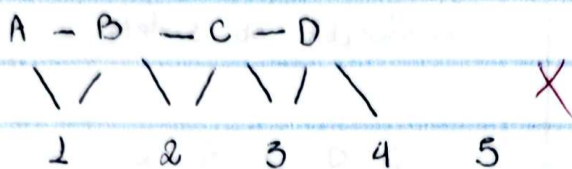
Exercício 5 (30%)

Diga se cada afirmação é verdadeira ou falsa e justifique em ambos os casos

- Um grafo não-direcionado sem ciclos com n vértices e $n - 1$ arestas pode ser desconexo
- Busca em largura e em profundidade são capazes de detectar caminhos mínimos em grafos com pesos
- Algoritmos de programação dinâmica tendem a ter um baixo consumo de memória primária (RAM)
- A complexidade de algoritmos recursivos sempre pode ser computada com o teorema mestre
- Algoritmos de programação dinâmica *bottom-up* preenchem completamente a tabela auxiliar
- O método de ordenação *mergesort* utiliza programação dinâmica

73

8) ① O número máximo de arestas é $m \times n : 4 \times 5 = 20$



② 1. Inicializa $\text{dist}[v] = \infty$ para todos vértices, exceto o origem ($\text{dist}[\text{origem}] = 0$)

2. Cria uma fila de prioridade Q com todos os vértices

3. Enquanto Q não estiver vazia:

a. Extrai u de Q com menor $\text{dist}[u]$

b. Para cada vizinho v de u :

i. Se $\text{dist}[v] > \text{dist}[u] + \text{peso}(u,v)$:

- $\text{dist}[v] = \text{dist}[u] + \text{peso}(u,v)$

- Atualiza v em Q

4. Retorna o vetor dist

25) ③ a) Função $p(n)$:

se $n == 1$:

retorne 1

senão:

retorne $2 * p(n-1) + 1$

b) Função $p(n)$:

resultado = 1

para i de 2 até n :

resultado = $2 * \text{resultado} + 1$

retorne resultado

c) $p(1) = 1$

~~$p(n) = 2^n$~~ $p(n+1) = 2p(n) + 1$

$p(n+1) = 2p(n) + 1$

$p(n) = 2^n - 1$

$p(6) = 2^6 - 1 = 63$

64

$$\begin{aligned} p(h+1) &= 2p(h) + 1 \\ &= 2(2^h - 1) + 1 \\ &= 2^{h+1} - 2 + 1 \\ &= 2^{h+1} - 1 \\ n &= h+1 \\ p(n) &= 2^n - 1 \end{aligned}$$

70

5

4

• Matriz de Adjacência

	A	B	C	D	E
A	0	1	1	1	1
B	1	0	1	1	1
C	1	1	0	1	1
D	1	1	1	0	1
E	1	1	1	1	0

• Lista de Adjacência

A : B, C, D, E

B : A, C, D, E

C : A, B, ~~A, B~~, D, E

D : A, B, C, E

E : A, B, C, D

30

5

a) Falso: Um grafo não direcionado sem ciclos com n vértices e $n-1$ arestas é uma árvore, que é conexa

b) Falso: Busca em largura e profundidade não são adequadas para encontrar caminhos mínimos em grafos com pesos. Outros são mais apropriados

c) Falso: Algoritmos de programação dinâmica geralmente usam tabelas auxiliares que podem consumir muita memória, especialmente para problemas com múltiplas dimensões

d) Falso: O teorema mestre só se aplica a recorrência que podem ser expressas na forma específica que ele cobra. Muitos algoritmos recursivos têm recorrências que não se encaixam nesse formato

e) Verdadeiro. Algoritmos bottom-up preenchem sistematicamente toda tabela antes de chegar a solução final

$T(n) = 2T(n/2) + O(n)$
 $T(n) = 2T(n/2) + O(n)$
 $T(n) = 2T(n/2) + O(n)$

5) Falso: Merge Sort utiliza divisão e conquista, não programação dinâmica.

Ele divide o problema em subproblemas independentes, enquanto a dinâmica não.

Divisão de Array	Array de Array
A: A, B, C, D, E	A B C D E
B: A, C, D, E	1 1 1 1
C: A, B, A, B, D, E	1 1 1 0 1 1
D: A, B, C, E	1 1 0 1 1
E: A, B, C, D	1 0 1 1 1
	0 1 1 1 0

6) Falso: Um dado não é dividido em subproblemas independentes, pois a solução de um subproblema depende da solução de outros subproblemas.

7) Falso: A programação dinâmica não é usada para resolver problemas de otimização, pois a solução de um subproblema depende da solução de outros subproblemas.

8) Falso: A programação dinâmica não é usada para resolver problemas de otimização, pois a solução de um subproblema depende da solução de outros subproblemas.

9) Falso: A programação dinâmica não é usada para resolver problemas de otimização, pois a solução de um subproblema depende da solução de outros subproblemas.

10) Falso: A programação dinâmica não é usada para resolver problemas de otimização, pois a solução de um subproblema depende da solução de outros subproblemas.