Simulado 2 CIC110

Algoritmo 1: Matrix-Chain-Order(p)

```
1.1  n = p.comprimento − 1
1.2  para i = 1 até n faça
1.3      m[i, i] = 0
1.4  para l = 2 até n faça
1.5      para i = 1 até n − l + 1 faça
1.6          j = i + l − 1
1.7          m[i, j] = ∞
1.8          para k = i até j − 1 faça
1.9              q = m[i, k] + m[k + 1, j] + p_{i−1} p_k p_j
1.10             se q < m[i, j] então
1.11                 m[i, j] = q
1.12                 s[i, j] = k
1.13 retorna m e s
```

Algoritmo 2: Mochila(c, w, W, n)

```
2.1  para d = 0 até W faça
2.2      z[0, d] = 0
2.3  para k = 1 até n faça
2.4      z[k, 0] = 0
2.5  para k = 1 até n faça
2.6      para d = 1 até W faça
2.7          z[k, d] = z[k − 1, d]
2.8          se w_k ≤ d e c_k + z[k − 1, d − w_k] > z[k, d] então
2.9              z[k, d] = c_k + z[k − 1, d − w_k]
2.10 retorna z[n, W]
```

1. Considere o problema da parentização de matrizes: Temos uma sequência $< A_1, A_2, \ldots, A_n >$ de n matrizes para multiplicar. Desejamos descobrir como parentizar essa sequência de forma a minimizar o número de multiplicações necessárias. Para tanto é fornecido um vetor $p = (p_0, p_1, \ldots p_n)$ de $n + 1$ inteiros, contendo as dimensões das matrizes. Sendo que a matriz A_i tem dimensões $p_{i-1} \times p_i$. Dica: representando por $A_{i..k}$ o produtos das matrizes $A_i, \ldots, A_k$, o número de multiplicações necessárias para calcular o produto $A_{i..k}A_{k+1..j}$ é $p_{i-1}p_kp_j$.

   a)(1 ponto) Mostre que esse problema tem subestrutura ótima.

   b)(1 ponto) Seja $m[i, j]$ o número de multiplicações de uma parentização ótima. Faça uma definição recursiva desse valor.

   c) (1 ponto) Um algoritmo de Programação Dinâmica calcula as subsequências de tamanhos menores e usa essas para calcular as sequências maiores. Um algoritmo para desse tipo é apresentado no Algoritmo 1. Aplique esse algoritmo, preenchendo as tabelas m e s para descobrir a parentização ótima de matrizes com as seguintes dimensões:

matriz:	A_1	A_2	A_3	A_4
dimensão	1×1	1×2	2×3	3×1

m	1	2	3	4
1				
2	———			
3	———	———		
4	———	———	———	

s	1	2	3	4
1	———			
2	———	———		
3	———	———	———	
4	———	———	———	———

   d) (1 ponto) De acordo com a suas tabelas, qual a parentização ótima das matrizes:

2. Considere o problema da mochila, em que recebemos n itens, cada item i um com peso $w[i]$ e valor $c[i]$, e uma mochila com uma capacidade representada por um inteiro W. O objetivo é selecionar um subconjunto dos itens que maximize o valor, mas cujo a soma dos pesos não ultrapasse a capacidade da mochila.

   Aplique o Algoritmo 2 na seguinte instância: $c = \{10, 7, 25, 34\}$, $w = \{2, 1, 6, 5\}$ e $W = 7$.

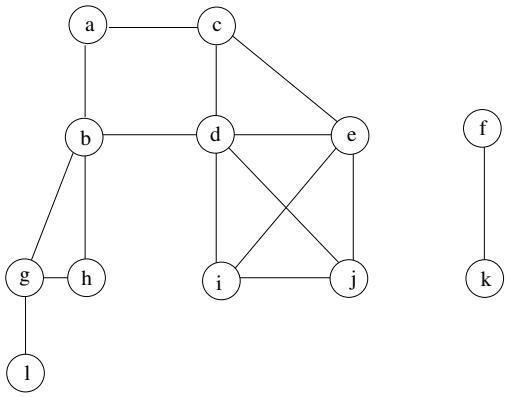
   a)(1 ponto) Complete a tabela:

	0	1	2	3	4	5	6	7
0								
1								
2								
3								

   b)(1 ponto) De acordo com a sua tabela, diga o valor da solução ótima:

   c)(1 ponto) De acordo com a sua tabela, diga quais itens fazem parte da sua solução ótima:

3. Considere o seguinte grafo:



a) (1 ponto) Em que ordem os vértices seriam visitados em uma busca em largura começando da raiz a? Considere que as adjacências são exploradas em ordem alfabética e que você imprime no momento que pinta o vértice de cinza.

b) (1 ponto) E em uma busca em profundidade?

4. Considere o texto **abcdeabcab**. Uma possível codificação de comprimento fixo poderia ser:

Cod.	a	b	c	d	e
Fixa	000	001	010	011	100

Nesse caso a codificação do texto seria:

000 001 010 011 100 000 001 010 000 001

o que gasta 30 bits para ser representado.

a)(1 ponto) Escreva uma codificação **ótima** de comprimento variável livre de prefixo.

Cod.	a	b	c	d	e
Variável					

b)(1 ponto) Escreva o texto usando a sua codificação.

c)(1 ponto) Quantos bits gastou sua representação do texto?