# 1º Teste de Análise e Desenho de Algoritmos

Departamento de Informática, FCT NOVA

23 de Abril de 2018

Duração: 1 hora e 45 minutos.

O teste tem **3** páginas e **4** perguntas.

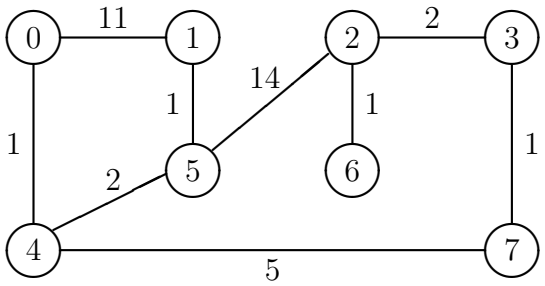
Responda a **perguntas** diferentes em **folhas** diferentes.

Se precisar de folhas, peça ao docente.

Nº de **folhas** entregues (excluindo o enunciado)

**Número:** _____ **Nome:** _____

1. [3 valores] Suponha que se executa o algoritmo de Kruskal com o grafo esquematizado na figura e que a classe da partição de vértices (a classe da variável `nodesPartition`) implementa **reunião por nível** e **representante com compressão do caminho**.




Assuma que a primeira extremidade de um arco é sempre o vértice menor e que, quando o nível das árvores é igual, o primeiro representante é a raiz da nova árvore. Por exemplo, se o primeiro arco a ser analisado for $(0, 4)$, a primeira extremidade é 0 e a segunda extremidade é 4. Como as duas árvores têm nível 1, `union(0,4)` cria uma árvore cuja raiz é 0.

Indique o estado da partição (o conteúdo do vetor `nodesPartition.partition`) imediatamente após o método `union` ter sido executado pela quarta vez, pela sexta vez e pela sétima vez.

**Após a 4ª reunião:**

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

**Após a 6ª reunião:**

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

**Após a 7ª reunião:**

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

2. [6 valores] Considere a seguinte função recursiva, $f(n)$, onde $n$ é um inteiro não negativo.

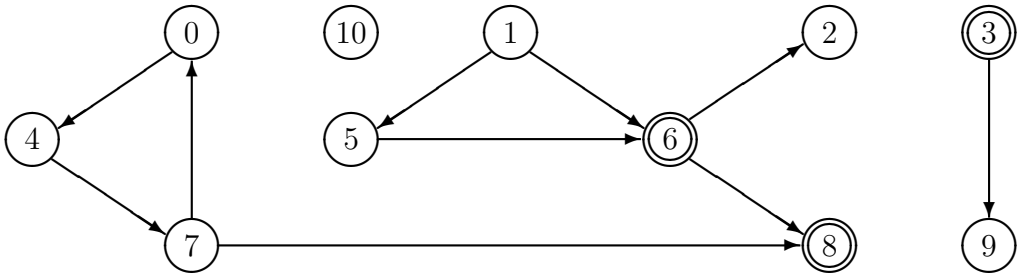
$$f(n) = \begin{cases} 1, & \text{se } n = 0; \\ \sum_{i=0}^{n-1} \Big( \; f(i) \times f(n-i-1) \; \Big), & \text{se } n \geq 1. \end{cases}$$

Apresente um algoritmo iterativo, desenhado segundo a técnica da programação dinâmica e implementado em Java, que recebe um número inteiro positivo, $k$, e calcula o valor de $f(k)$. Estude (justificando) as complexidades temporal e espacial do seu algoritmo.

3. MIKA é uma paciência jogada com $n$ varetas vermelhas e pretas. Cada vareta é monocromática (é toda vermelha ou é toda preta) e tem um número gravado (entre 0 e $n-1$) que indica o seu valor. Não há duas varetas com o mesmo valor.

   Antes de se começar a jogar, colocam-se as varetas todas na vertical sobre uma mesa, segurando-as com uma mão, que se abre para que as varetas caiam livremente. Em cada jogada, o jogador tenta retirar uma vareta vermelha da mesa sem tocar nas outras varetas. Se tocar noutra vareta, o jogo termina. O jogo também termina quando se retira uma vareta preta (por engano) ou quando já não houver varetas vermelhas na mesa. A *pontuação* obtida é a soma dos valores das varetas vermelhas retiradas com sucesso (sem ter sido tocada outra vareta para além da que se estava a retirar).

   A configuração inicial das varetas pode ser representada por um grafo orientado e uma coloração das varetas. Cada vértice representa uma vareta e o seu identificador (um inteiro entre 0 e $|V|-1$) corresponde ao valor da vareta. A existência de um arco do vértice $v$ para o vértice $w$ indica que a vareta com valor $v$ deve ser retirada antes de se retirar a vareta com valor $w$ (porque é impossível retirar $w$ com $v$ na mesa sem lhe tocar). A coloração das varetas diz quais são as varetas vermelhas e quais são as pretas. Dada uma configuração inicial, pretende-se obter uma *sequência ótima de jogadas*, ou seja, uma sequência de varetas vermelhas que, se forem retiradas com sucesso por essa ordem, permitem obter a maior pontuação possível.




   Para exemplificar, considere a configuração inicial esquematizada na figura, onde os círculos duplos representam varetas pretas. Qualquer uma das seguintes sequências é uma sequência ótima de jogadas (que permitem obter 16 pontos):

   10 1 5, 1 10 5 e 1 5 10

   (a) [5 valores] Apresente uma função (em pseudo-código) que recebe:
   - um grafo orientado, $G = (V, A)$, com a informação sobre o número de varetas, os seus valores e as suas dependências, e
   - uma lista de vértices, $L$, com as varetas vermelhas (sem repetições). Pode assumir que há sempre, pelo menos, uma vareta de cada cor (ou seja, $1 \leq |L| < |V|$). (No exemplo, $L$ teria os números 0, 1, 2, 4, 5, 7, 9 e 10, por uma ordem qualquer.)

   A função retorna uma qualquer sequência ótima de jogadas, implementada em lista de vértices.

   (b) [1 valor] Que estruturas de dados usaria para implementar o grafo? Não escreva código, mas pode ilustrar a sua resposta com o grafo do exemplo.

   (c) [1 valor] Estude (justificando) a complexidade temporal do seu algoritmo, no pior caso, assumindo que o grafo está implementado como indicou na alínea anterior.

4. [4 valores] A Maria recebeu um ramo com 10 rosas e decidiu distribuí-las por três jarras, para colocar uma jarra na sala, outra no quarto e a outra no escritório. Mas a distribuição tem restrições:

   - Na sala, que é a divisão maior, quer que haja muitas rosas: entre 4 e 7 rosas.
   - No quarto, quer apenas 1 ou 2 rosas, para não ter de as retirar à noite.
   - No escritório quer ter entre 1 e 3 rosas.

   De quantas maneiras pode a Maria distribuir as rosas, satisfazendo todas estas restrições?

   É fácil verificar que só há 5 distribuições das 10 rosas que satisfazem todas as restrições. Cada uma dessas distribuições corresponde a uma linha da tabela.

| Sala<br>4 . . . 7 | Quarto<br>1 . . . 2 | Escritório<br>1 . . . 3 |
|---|---|---|
| 5 | 2 | 3 |
| 6 | 1 | 3 |
| 6 | 2 | 2 |
| 7 | 1 | 2 |
| 7 | 2 | 1 |

   Apresente **uma função matemática recursiva** que, com base:

   - no número (inteiro positivo) $R$ de rosas,
   - no número (inteiro positivo) $D$ de divisões da casa,
   - em duas sequências de $D$ números inteiros positivos,

     $$M = (m_1\ m_2\ \cdots\ m_D) \quad \text{e} \quad N = (n_1\ n_2\ \cdots\ n_D),$$

     que indicam que a divisão $k$ da casa tem de ter entre $m_k$ e $n_k$ rosas (onde $m_k < n_k$, para $k = 1, \ldots, D$),

   calcula o número de distribuições das $R$ rosas que satisfazem todas as restrições. Indique claramente o que representa cada uma das variáveis que utilizar e explicite a chamada inicial (a chamada que resolve o problema).