# 1º Teste de Análise e Desenho de Algoritmos

Departamento de Informática da FCT NOVA

16 de Abril de 2016

1. [4 valores] O vetor `uf.partition` contém o estado da partição `uf` (i.e., o estado da instância `uf` da classe *UnionFindInArray*). Assuma que se adotou reunião por tamanho e representante com compressão do caminho.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| `uf.partition:` | −5 | 0 | 0 | 0 | 3 | −4 | 5 | 5 | 7 | −2 | 9 |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

Indique o estado da partição (ou seja, o conteúdo do vetor `uf.partition`) após a execução **de cada um** dos seguintes métodos, pela ordem indicada:

(a) `uf.find(4)`
(b) `uf.union(0, 5)`
(c) `uf.find(7)`
(d) `uf.union(0, 9)`

2. [6 valores] Considere a seguinte função recursiva $f_{X,y}(i,j)$, onde:

- $X = (x_0, x_1, \ldots, x_n)$ é uma sequência de inteiros (com $n \geq 2$);
- $y$ é um inteiro positivo inferior a $n$;
- $i$ é um inteiro entre 0 e $n$ $(0 \leq i \leq n)$;
- $j$ é 0 ou 1.

$$f_{X,y}(i,j) = \begin{cases} 0, & \text{se } i < y; \\ \max\limits_{1 \leq k \leq y} \Big( f_{X,y}(i-k, 1) + x_{i-k} \Big), & \text{se } i \geq y \text{ e } j = 0; \\ \max\limits_{1 \leq k \leq y} \Big( f_{X,y}(i-k, 0) - x_{i-k} \Big), & \text{se } i \geq y \text{ e } j = 1. \end{cases}$$

Note que a sequência $X$ e o inteiro $y$ não variam entre chamadas recursivas. Por esse motivo, optou-se por escrever $f_{X,y}(i,j)$ em vez de $f(X,y,i,j)$.

Apresente um algoritmo iterativo, desenhado segundo a técnica da programação dinâmica, que recebe:

- uma sequência $X = (x_0, x_1, \ldots, x_n)$ de inteiros (com $n \geq 2$) e
- um inteiro positivo $y$ inferior a $n$

e calcula o valor de $f_{X,y}(n, 1)$. Estude (justificando) as complexidades temporal e espacial do seu algoritmo.

3. [6 valores] Para modelizar a propagação de uma infecção causada por um agente patogénico (que não é combatido), criou-se um grafo onde os vértices representam as localidades. Existe um arco entre duas localidades se as localidades forem **vizinhas**. Sabe-se que, se uma localidade estiver infectada numa determinada semana, as localidades vizinhas estarão infectadas na semana seguinte.

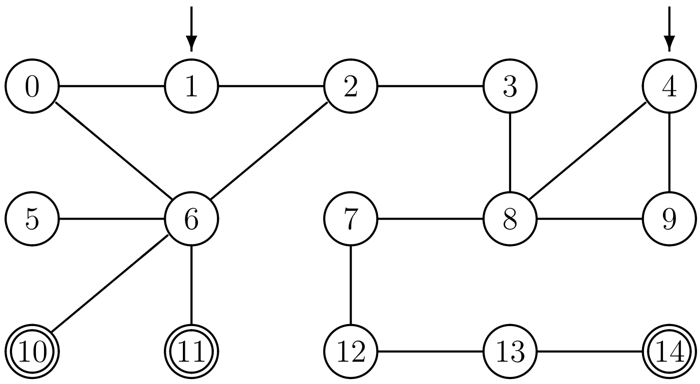


Figura 1: A infecção começa nas localidades 1 e 4. As localidades 10, 11 e 14 são nevrálgicas.

Considere o grafo esquematizado na Figura 1 e assuma que a infecção começa nas localidades 1 e 4 (ou seja, 1 e 4 são as localidades infectadas na semana zero). A infecção propaga-se da seguinte forma:

- Na semana 1, as localidades 0 e 2 (por serem vizinhas da localidade 1) e as localidades 8 e 9 (por serem vizinhas da localidade 4) também estarão infectadas.
- Na semana 2, a infecção alastrará às localidades 3, 6 e 7.
- Na semana 3, a infecção chegará às localidades 5, 10, 11 e 12.
- Na semana 4, a localidade 13 também estará infectada.
- Na semana 5, todas as localidades estarão infectadas.

Algumas localidades, onde as consequências da infecção poderão ser catastróficas, são consideradas **nevrálgicas**. Pretende-se encontrar uma das primeiras localidades nevrálgicas a ser infectada.

No nosso exemplo, há três localidades nevrálgicas: 10, 11 e 14. Como as localidades 10 e 11 serão infectadas na semana 3 e a localidade 14 será infectada na semana 5, uma das primeiras localidades nevrálgicas a ser infectada tanto pode ser 10 como 11.

Apresente uma função (em pseudo-código) que recebe:

- um grafo não orientado e não pesado,
- dois vértices distintos, `loc1` e `loc2`, onde começa a infecção e
- uma lista não vazia de vértices com as localidades nevrálgicas

e retorna uma das primeiras localidades nevrálgicas a ser infectada. Assuma (sem testar) que as localidades `loc1` e `loc2` não são nevrálgicas e que a infecção alastra sempre a alguma localidade nevrálgica. Indique (justificando) a complexidade temporal do seu algoritmo, no pior caso.

4. [4 valores] No jogo da *Pirâmide*, o jogador recebe uma pirâmide de números inteiros (como a da Figura 2a). O objetivo é efetuar um percurso do topo da pirâmide até à base que maximize a soma dos números do percurso.

   O percurso começa no topo da pirâmide. Em cada ponto do percurso (que não pertença à base da pirâmide), o jogador é obrigado a descer para o número que se encontra imediatamente à esquerda ou imediatamente à direita do ponto onde se encontra. O percurso termina assim que se chega à base da pirâmide.

$$\begin{array}{ccccccccc} & & & & 7 & & & & \\ & & & 3 & & 9 & & & \\ & & 8 & & 1 & & -1 & & \\ & 2 & & 7 & & 4 & & 4 & \\ 0 & & 5 & & 2 & & -6 & & 5 \end{array}$$

(a) A pirâmide

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 7 | | | | |
| 1 | 3 | 9 | | | |
| 2 | 8 | 1 | $-1$ | | |
| 3 | 2 | 7 | 4 | 4 | |
| 4 | 0 | 5 | 2 | $-6$ | 5 |

(b) A tabela que guarda a pirâmide

Figura 2: Exemplo de uma pirâmide

   Considere, a título de exemplo, os dois seguintes percursos na pirâmide da Figura 2a, onde os símbolos $\swarrow$ e $\searrow$ indicam que se desceu para o número, respetivamente, imediatamente à esquerda e imediatamente à direita.

   - **7** ($\searrow$) **9** ($\swarrow$) **1** ($\swarrow$) **7** ($\swarrow$) **5** O valor deste percurso é $7 + 9 + 1 + 7 + 5 = 29$.
   - **7** ($\swarrow$) **3** ($\swarrow$) **8** ($\searrow$) **7** ($\swarrow$) **5** O valor deste percurso é $7 + 3 + 8 + 7 + 5 = 30$.

   Assuma que a pirâmide é guardada numa tabela com tantas linhas e tantas colunas quantos os níveis da pirâmide, como se ilustra na Figura 2b. A linha 0 tem o elemento do topo na coluna 0; a linha 1 tem dois elementos, nas colunas 0 e 1; e assim sucessivamente, até à última linha, que tem a base da pirâmide.

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

   - numa tabela $T$ (com $n \geq 1$ linhas e $n$ colunas) onde está guardada a pirâmide,

   calcula o valor máximo dos percursos na pirâmide. Indique claramente o que representa cada uma das variáveis que utilizar e explicite a chamada inicial (a chamada que resolve o problema).