2º Teste de Análise e Desenho de Algoritmos

Departamento de Informática, FCT NOVA

22 de Junho de 2021

Duração: 2 horas

Caderno 1 (2 folhas e 2 perguntas)

Tem de entregar os 2 cadernos.

Os cadernos não podem ser desagrafados.

Identifique os cadernos com o seu número e o seu nome.

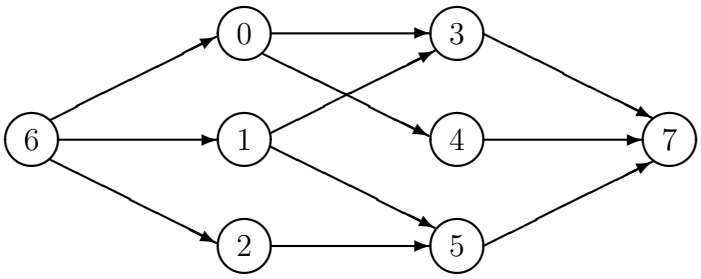
Número: _____ **Nome:** _____

Pergunta 1 Suponha que se pretende encontrar um emparelhamento máximo no grafo não orientado $G = (\{0,1,2,3,4,5\}, \{(0,3),(0,4),(1,3),(1,5),(2,5)\})$ e que, para o obter, se realizam os três seguintes passos:

(1) Constrói-se o grafo G' orientado e pesado, esquematizado na figura, cujos arcos têm peso 1;

(2) Executa-se o algoritmo de Edmonds-Karp com o grafo G', a fonte 6 e o dreno 7;

(3) Computa-se um emparelhamento máximo de G com o fluxo máximo obtido no passo (2).



(a) [3 valores] Indique a sequência de caminhos da fonte para o dreno que são encontrados pelo algoritmo de Edmonds-Karp, assumindo que o método *outIncidentEdges* itera sempre os arcos por ordem crescente de vértice destino. Por exemplo, $G'.outIncidentEdges(6)$ produz os arcos $(6,0)$, $(6,1)$ e $(6,2)$, por esta ordem, todos com peso 1.

(b) [1 valor] Quais são os arcos do emparelhamento máximo computado?

Pergunta 2 No Problema da Concatenação, há duas sequências não vazias de palavras sobre um alfabeto:

$$X = (x_1,\, x_2,\, \ldots,\, x_n) \quad \text{e} \quad Y = (y_1,\, y_2,\, \ldots,\, y_n).$$

Uma *ordem de concatenação* é uma sequência I de inteiros entre 1 e n (usados como índices):

$$I = (i_1, i_2, \ldots, i_k) \qquad (\text{com } 1 \leq k \leq n).$$

Quando se concatenam palavras de X pela ordem definida por I, obtém-se a palavra:

$$P(X, I) = x_{i_1}\, x_{i_2}\, \cdots\, x_{i_k}\,.$$

Quando se concatenam palavras de Y pela ordem definida por I, obtém-se a palavra:

$$P(Y, I) = y_{i_1}\, y_{i_2}\, \cdots\, y_{i_k}\,.$$

Queremos saber se existe uma ordem de concatenação tal que $P(X, I) = P(Y, I)$.
Vejamos um exemplo onde X e Y têm 5 palavras:

$$\begin{array}{lccccccccc} X = (& a\,a\,b & , & a\,a & , & a & , & b\,a\,b & , & a\,b &) \\ Y = (& b\,b & , & a\,a\,a\,a & , & a & , & a\,b & , & b\,a &). \\ & \scriptstyle 1 & & \scriptstyle 2 & & \scriptstyle 3 & & \scriptstyle 4 & & \scriptstyle 5 & \end{array}$$

A ordem de concatenação $I = (2, 1, 4)$ satisfaz a propriedade pretendida porque $x_2\, x_1\, x_4 = y_2\, y_1\, y_4$:

$$\underbrace{a\,a}_{x_2}\underbrace{a\,a\,b}_{x_1}\underbrace{b\,a\,b}_{x_4} \;=\; \underbrace{a\,a\,a\,a}_{y_2}\underbrace{b\,b}_{y_1}\underbrace{a\,b}_{y_4}$$

O **Problema da Concatenação** formula-se da seguinte forma. Dadas duas sequências não vazias de palavras sobre um alfabeto, $X = (x_1, \ldots, x_n)$ e $Y = (y_1, \ldots, y_n)$, existe uma ordem de concatenação I tal que $P(X, I) = P(Y, I)$?

(a) [3.7 valores] Prove que o Problema da Concatenação é NP.

(b) [0.3 valores] Assuma que resolveu a alínea anterior e indique, de forma **extremamente concisa**, o que seria necessário provar para concluir que o Problema da Concatenação é NP-completo. (Não prove nada; limite-se a dizer o que teria de ser provado.)

2º Teste de Análise e Desenho de Algoritmos

Departamento de Informática, FCT NOVA

22 de Junho de 2021

Duração: 2 horas

Caderno 2 (4 folhas e 2 perguntas)

Tem de entregar os 2 cadernos.

Os cadernos não podem ser desagrafados.

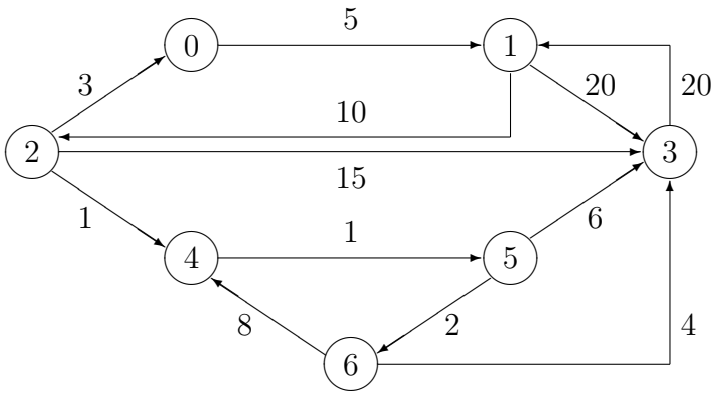
Identifique os cadernos com o seu número e o seu nome.

Número: _____ **Nome:** _____

Pergunta 3 Considere um grafo orientado e pesado. Dados dois caminhos p e q, com a mesma origem e o mesmo destino, diz-se que p é *mais equilibrado* do que q se o maior peso dos arcos de p é menor que o maior peso dos arcos de q.

Dados um grafo orientado e pesado, cujos arcos têm peso positivo, e dois vértices v e w, pretende-se encontrar um caminho *ótimo* de v para w. Um caminho é *ótimo* se satisfizer as duas seguintes propriedades:

- é um caminho mais curto (o seu comprimento pesado é o menor possível);
- de entre os caminhos mais curtos de v para w, é um dos mais equilibrados.



No grafo representado na figura, há dois caminhos mais curtos do vértice 2 para o vértice 3. Só um deles é ótimo (o caminho 2 4 5 6 3) porque é mais equilibrado do que o outro.

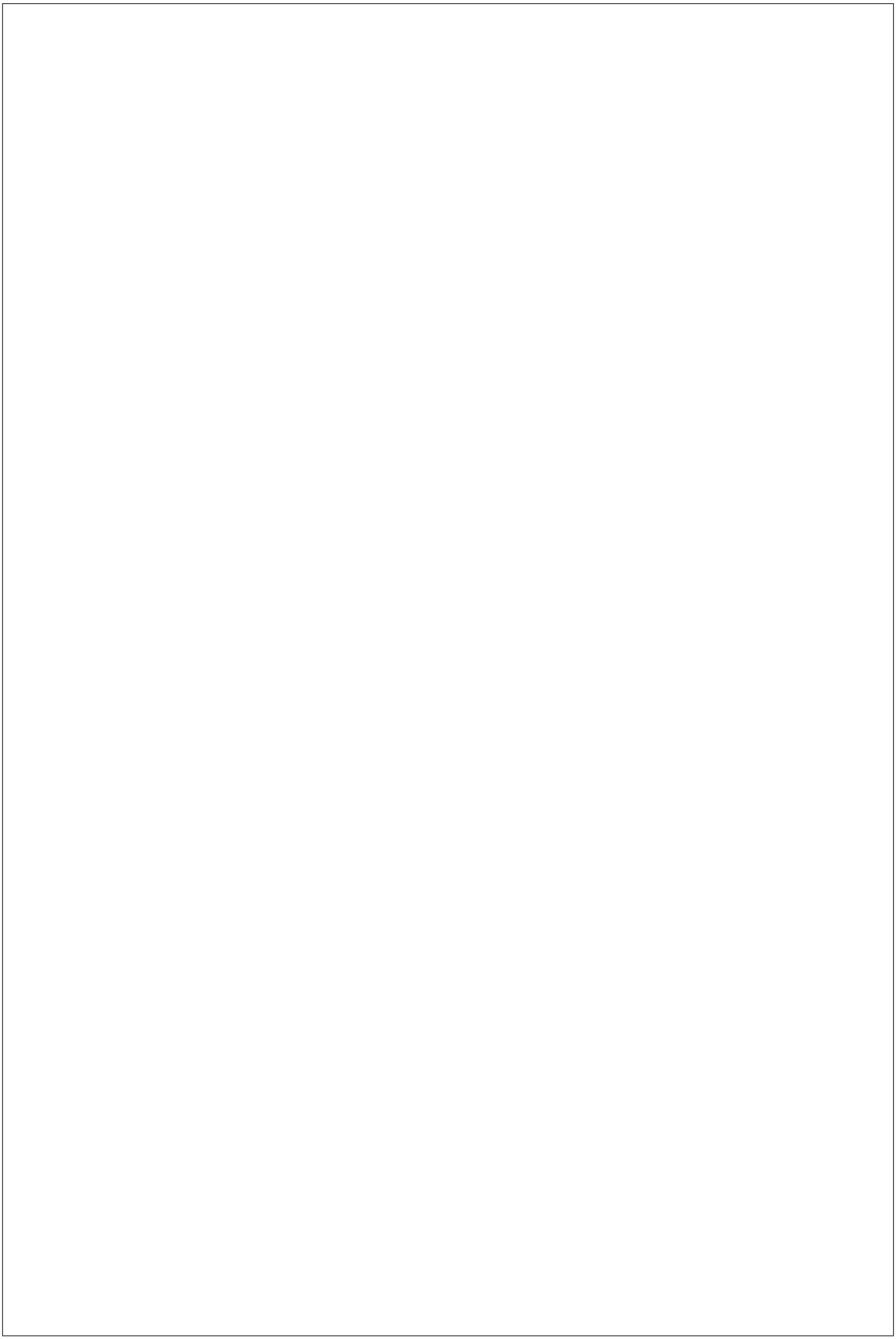
Caminho	**Comprimento Pesado**	**Maior Peso dos Arcos**
2 4 5 3	8	$6 = \max(1, 1, 6)$
2 4 5 6 3	8	$4 = \max(1, 1, 2, 4)$

Pretende-se que implemente uma função, *optPath*(G, v, w), que receba:

- um grafo G, orientado e pesado (com pesos positivos nos arcos) e
- dois vértices, v e w,

e que retorne um caminho ótimo de v para w. Assuma que há algum caminho de v para w.

(a) [5 valores] Apresente a função *optPath* (em pseudo-código). Se quiser usar algum método dos slides, não o copie; chame-o. Se quiser alterar algum método dos slides, escreva a sua nova definição integralmente.



(b) [1 valor] Que estruturas de dados escolheria para implementar o grafo? Não escreva código, mas ilustre a sua resposta com o grafo do exemplo. Como o grafo é grande, ilustre apenas com uma parte.

(c) [1 valor] Qual é a complexidade temporal do seu algoritmo, no pior caso, assumindo que o grafo está implementado como indicou na alínea anterior? Justifique a sua resposta.

Pergunta 4 O estudo das redes sociais permite compreender e prever comportamentos. Neste exercício, pretende-se calcular a afabilidade de uma rede (em tempos de crescente agressividade).

Uma rede social é modelada por um grafo não orientado, conexo e pesado. Cada vértice representa um indivíduo e a existência de um arco entre dois vértices indica que os dois indivíduos são "amigos". O peso do arco caracteriza a *cordialidade* entre eles. Valores de cordialidade elevados denotam verdadeira amizade; valores negativos denotam crispação.

Uma *sub-rede representativa* da rede social é um sub-grafo (não orientado e pesado) da rede que tem todos os vértices da rede e que é conexo e acíclico. A *afabilidade* de uma sub-rede representativa é a soma dos pesos dos seus arcos. A *afabilidade* da rede é a maior afabilidade das suas sub-redes representativas.

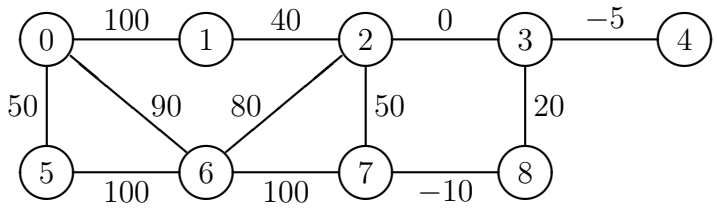


Figura 1: Rede social R

Para exemplificar, considere a rede social R, desenhada na Figura 1, e a sub-rede representativa R_s de R, desenhada na Figura 2. A afabilidade de R_s é 485, porque a soma dos pesos dos arcos de R_s é 485. A afabilidade da rede R é 485, porque há uma sub-rede representativa de R cuja afabilidade é 485 e nenhuma sub-rede representativa de R tem afabilidade superior a 485.

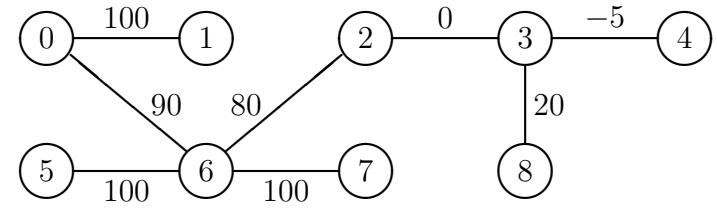


Figura 2: Sub-rede representativa R_s de R

Pretende-se que implemente uma função, *affability*(G), que receba:

- um grafo G, não orientado, conexo e pesado (com pesos nos arcos)

e que retorne a afabilidade da rede G. **O corpo da função *affability* deve chamar:**

- uma ou várias funções que constroem grafos;
- um ou vários algoritmos de grafos estudados, como se eles estivessem numa biblioteca (mesmo que esses algoritmos retornem resultados que não interessam para resolver este problema e sejam menos eficientes do que poderiam ser para este caso).

(a) [2 valores] **Desenhe o(s) grafo(s)** que seria(m) construído(s) durante a execução de *affability*(R), onde R é o grafo desenhado na Figura 1. (Desenhar um grafo é representá-lo da forma habitual.)

(b) [3 valores] Implemente a função *affability* (em pseudo-código). Chame a ou as várias funções que constroem grafos, mas não a(s) implemente (porque já ilustrou o(s) grafo(s) criado(s) na alínea anterior). Não implemente nenhum algoritmo de grafos; recorra aos algoritmos estudados, chamando as respetivas funções, sem as alterar.

```
L affability( UndiGraph<L> graph ) {
```