Exame de Análise e Desenho de Algoritmos

Departamento de Informática, FCT NOVA
15 de Julho de 2021
Duração: 3 horas

Caderno 1 (3 folhas e 3 perguntas)

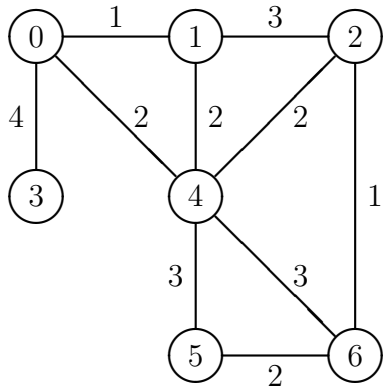
Tem de entregar os 2 cadernos.

Os cadernos não podem ser desagrafados.

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

Número: _____ **Nome:** _____

Pergunta 1 Suponha que se executa o algoritmo de Prim com o grafo G esquematizado na figura.



Assuma que a origem é o vértice 0 (ou seja, considere que o método *G.aNode()* retorna 0).

(a) [2 valores] Indique a ordem pela qual os arcos são inseridos no resultado (i.e. no vetor *mst*), representando um arco com peso p entre os vértices v e w pelo par (v, w) ou (w, v).

(b) [1 valor] Quais são os arcos que, ao serem analisados, dão origem a uma execução do método *decreaseKey*?

Pergunta 2 Considere a seguinte função recursiva, $f_S(i, j)$, onde:

- $S = (x_0, x_1, \ldots, x_n)$ é uma sequência de números inteiros positivos (com $n \geq 0$);
- i e j são números inteiros entre 0 e n $(0 \leq i \leq n$ e $0 \leq j \leq n)$.

$$f_S(i,j) = \begin{cases} \max(x_i, x_j), & \text{se } i = 0 \text{ ou } j = 0; \\ j^2 + f_S(i-1, j-1), & \text{se } i > 0 \text{ e } j > x_i; \\ \min\left(\; x_i + f_S(i-1, j),\; x_j + f_S(i, j-1) \; \right), & \text{se } i > 0 \text{ e } 1 \leq j \leq x_i. \end{cases}$$

Note que a sequência S não varia entre chamadas recursivas. Por esse motivo, optou-se por escrever $f_S(i, j)$ em vez de $f(S, i, j)$.

(a) [2.9 valores] Apresente um algoritmo iterativo, desenhado segundo a técnica da programação dinâmica e implementado em Java, que receba uma sequência de $n + 1$ inteiros positivos (guardada num vetor do tipo `int[]` com comprimento $n + 1$):

$$S = (x_0, x_1, \ldots, x_n) \qquad (\text{com } n \geq 0)$$

e calcule o valor de $f_S(n, n)$.

(b) [0.3 valores] Qual é a complexidade espacial do seu algoritmo? Justifique a sua resposta.

(c) [0.3 valores] Qual é a complexidade temporal do seu algoritmo? Justifique a sua resposta.

Pergunta 3 Na Paciência do Quadrado Latino, há uma matriz quadrada, de n por n, cujas células ou estão vazias ou têm um inteiro entre 1 e n (como se ilustra na Figura 1). O objetivo é colocar um número em cada célula vazia, garantindo que, em cada linha e em cada coluna da matriz, todos os inteiros entre 1 e n ocorrem exatamente uma vez (veja a Figura 2).

		1	
		2	
	1		

Figura 1: Paciência P

1	2	3	4
3	4	1	2
4	3	2	1
2	1	4	3

Figura 2: Solução S de P

0	0	0	0
0	0	1	0
0	0	2	0
0	1	0	0

Figura 3: Codificação C de P

No Problema do Quadrado Latino, é dada uma matriz C que codifica a paciência P (c.f. Figura 3): as células vazias de P têm zero em C e as células de P com um número têm o mesmo número em C . Pretende-se saber se existe uma *solução da paciência*, i.e., se existe uma matriz S (de n por n, tal como P e C) que satisfaz as duas seguintes propriedades:

- As células de C com um número positivo têm o mesmo número em S:
$$(\forall i, j = 1, \ldots, n)\ \ C[i][j] > 0 \ \Rightarrow\ S[i][j] = C[i][j]\,.$$
- Em cada linha e em cada coluna da matriz S, todos os inteiros entre 1 e n ocorrem exatamente uma vez.

O **Problema do Quadrado Latino** formula-se da seguinte forma. Dada uma matriz C, com n linhas e n colunas (onde $n \geq 2$), que codifica uma paciência do quadrado latino, existe uma solução da paciência?

(a) [2.7 valores] Prove que o Problema do Quadrado Latino é NP.

(b) [0.3 valores] Assuma que resolveu a alínea anterior e que há uma redução polinomial do Problema do Caixeiro Viajante para o Problema do Quadrado Latino. Pode concluir que há um algoritmo polinomial para resolver o Problema do Quadrado Latino?
(Assinale a opção correta e justifique a sua resposta de forma **extremamente concisa**.)

Sim ☐ **Não** ☐

Exame de Análise e Desenho de Algoritmos

Departamento de Informática, FCT NOVA
15 de Julho de 2021
Duração: 3 horas

Caderno 2 (5 folhas e 3 perguntas)

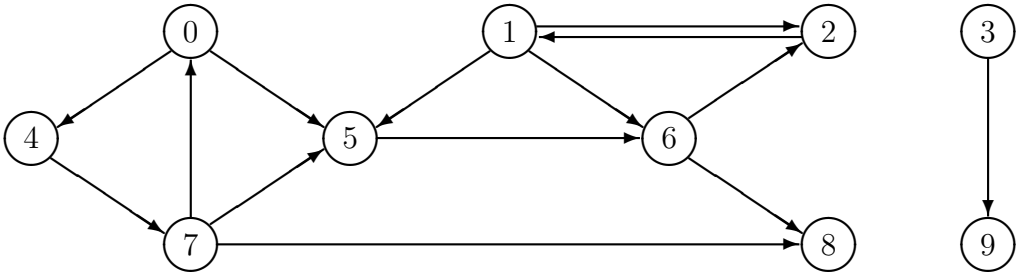
Tem de entregar os 2 cadernos.

Os cadernos não podem ser desagrafados.

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

Número: _____ **Nome:** _____

Pergunta 4 Dados um grafo orientado, dois vértices, v e w, e um número inteiro positivo, *maxValue*, pretende-se descobrir se existe algum caminho de v para w com comprimento inferior ou igual a *maxValue*.



Por exemplo, no grafo representado na figura:

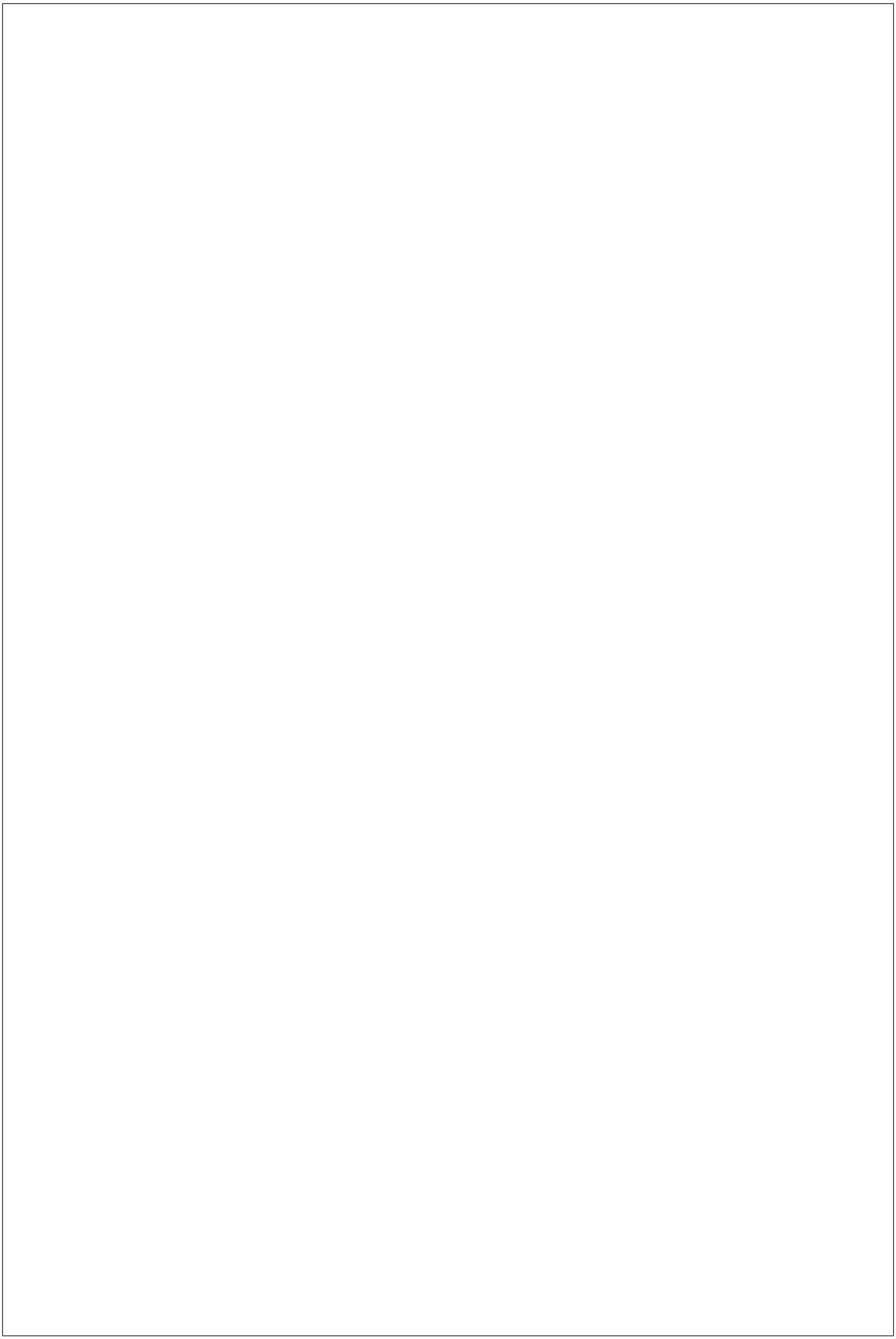
- há caminho de 4 para 8 com comprimento inferior ou igual a 2 (por exemplo, 4 7 8, que tem comprimento 2);
- há caminho de 0 para 2 com comprimento inferior ou igual a 4 (por exemplo, 0 5 6 2, que tem comprimento 3);
- não há caminho de 4 para 2 com comprimento inferior ou igual a 3 (porque todos os caminhos de 4 para 2 têm comprimento superior a 3);
- não há caminho de 1 para 9 com comprimento inferior ou igual a 20 (porque não há qualquer caminho de 1 para 9).

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

- um grafo G, orientado (e não pesado),
- dois vértices, v e w, e
- um inteiro positivo, *maxValue*,

e que retorne *true* se, e só se, existir algum caminho em G de v para w com comprimento inferior ou igual a *maxValue*.

(a) [3.4 valores] Apresente a função *existsPath* (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) [0.3 valores] 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) [0.3 valores] 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 5 O João está perdido numa ilha mágica, da qual pretende sair. A ilha tem vários tipos de terrenos, que se atravessam a velocidades diferentes, e tem rodas mágicas, que teletransportam quem as aciona para outro local na ilha. Quando se usa uma roda mágica, também se viaja no tempo, chegando-se ao destino num momento que pode pertencer ao passado, ao presente ou ao futuro. Para sair da ilha, o João tem de chegar a um local especial, chamado *a saída* da ilha.

Considere que a ilha é modelada por um grafo orientado, simplesmente conexo e pesado. Cada vértice representa um local e a existência de um arco (v, w), com peso t, indica que o João demora t unidades de tempo para se deslocar do local v para o local w. Dados dois locais, aquele onde o João se encontra e a saída da ilha, pretende-se saber o número mínimo de unidades de tempo que têm de passar para que o João saia da ilha ou que esse número não existe (caso não exista).

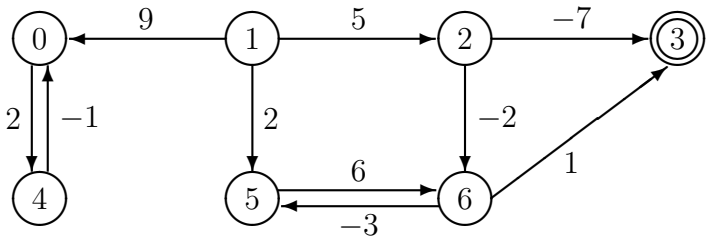


Figura 1: Ilha I

Para exemplificar, considere a ilha I, desenhada na Figura 1, cuja saída é no local 3. Se o João estiver no local 1, só consegue sair da ilha decorridas **-2** unidades de tempo, percorrendo o caminho 1 2 3. Note que há outros caminhos que o conduzem à saída, mas nenhum demora menos tempo a percorrer do que o descrito.

Pretende-se que implemente uma função, *minTime*(G, *start*, *exit*), que receba:

- um grafo G, orientado, simplesmente conexo e pesado (com pesos nos arcos),
- um vértice *start*, que é o local onde o João se encontra, e
- um vértice *exit*, que é a (única) saída da ilha,

e que retorne:

- o número mínimo de unidades de tempo que têm de passar para que o João saia da ilha, se esse número existir; e
- $-\infty$, no caso contrário.

Assuma que qualquer local é acessível a partir de *start* (existe algum caminho de *start* para o local) e que nenhum arco tem origem em *exit*. **O corpo da função *minTime* deve chamar:**

- zero, 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) [1.7 valores] **Desenhe o(s) grafo(s)** que seria(m) construído(s) durante a execução de $minTime(I, 1, 3)$, onde I é o grafo desenhado na Figura 1. Se nenhum grafo fosse construído, escreva apenas "**Nenhum**". (Desenhar um grafo é representá-lo da forma habitual.)

(b) [1.8 valores] Implemente a função *minTime* (em pseudo-código). Se a sua implementação recorrer a uma ou várias funções que constroem grafos, chame-a(s), 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 minTime( Digraph<L> graph, Node start, Node exit ) {
```

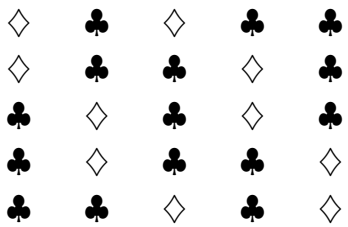
Pergunta 6 Pretende-se descobrir quantas sequências existem com as seguintes características:

- cada sequência tem (apenas) O cartas de ouros e P cartas de paus;
- em cada sequência, não há mais de x cartas de ouros consecutivas, nem mais de y cartas de paus consecutivas.

Por exemplo, se cada sequência tiver:

- $O = 2$ cartas de ouros,
- $P = 3$ cartas de paus,
- no máximo, $x = 1$ cartas de ouros consecutivas e
- no máximo, $y = 2$ cartas de paus consecutivas,

a resposta é 5. As 5 alternativas possíveis são:



Pretende-se definir **uma função matemática recursiva** que, com base em quatro números inteiros positivos:

$$O,\ P,\ x \ \text{ e } \ y \quad (\text{com } x \leq O \ \text{ e } \ y \leq P),$$

calcula o número de sequências que têm O cartas de ouros, P cartas de paus, no máximo x cartas de ouros consecutivas e no máximo y cartas de paus consecutivas.

(a) [2.7 valores] Defina a função pretendida e indique claramente o que representa cada uma das variáveis que utilizar.

(b) [0.3 valores] Indique a chamada inicial (a chamada que resolve o problema).