Tópicos Avançados em Algoritmos - época de recurso

O enunciado tem frente e verso / Em cada alínea indica-se a cotação / Duração da prova: 2h15m

1. [14%] Diga se cada uma das seguintes afirmações é verdadeira ou falsa (não é necessário justificar). **Notas.** Elemento de ordem i de um vector $v[1 \cdots n]$: aquele que ficaria na posição i se o vector estivesse ordenado. Os tempos considerados são tempos no pior caso. Os vectores têm n elementos.
   (a) O menor elemento de um vector já ordenado pode ser determinado em tempo $O(1)$
   (b) O menor elemento de um vector (não necessariamente ordenado) pode ser determinado em tempo $O(\sqrt{n})$
   (c) Para qualquer i, o elemento de ordem i de um vector (não necessariamente ordenado) pode ser determinado em tempo $O(\log n)$
   (d) Uma ordenação numa rede de n comparadores pode ser simulada por um algoritmo sequencial em tempo $\Theta(n)$.
   (e) É possível multiplicar 2 matrizes quadradas de dimensão $n \times n$ em tempo $O(n^4)$
   (f) É possível somar 2 matrizes quadradas de dimensão $n \times n$ em tempo $O(n\sqrt{n})$
   (g) No método do "quick sort" aleatorizado, sejam quais forem os dados, o tempo médio (relativamente à aleatorização do algoritmo) é de ordem $O(n \log n)$.

2. [7%] Como sabe, a classe **RP** é constituída pelas linguagens L para as quais existe um polinómio $l(\,)$ e uma função computável em tempo polinomial $f : \{0,1\}^n \times \{0,1\}^{l(n)} \to \{0,1\}$ tais que
$$\begin{cases} x \in L & \Rightarrow & \text{prob}_{\,y \in_u \{0,1\}^{l(n)}}[f(x,y) = 1] \geq 1/2 \\ x \notin L & \Rightarrow & \text{prob}_{\,y \in_u \{0,1\}^{l(n)}}[f(x,y) = 1] = 0 \end{cases}$$
   Mostre que **RP**$\subseteq$**NP**.

3. [4%] Defina formalmente profundidade de uma rede de comparadores.

4. Princípio 0/1
   (a) [3%] Enuncie o Princípio 0/1 relativo a redes de comparadores.
   (b) [3%] Qual o interesse do Princípio 0/1?
   (c) Suponha que se pretende mostrar que uma rede de comparadores ordena sempre correctamente n inteiros distintos.
      i. [4%] Mostre que basta mostrar que a rede ordena correctamente as $n!$ entradas que são as permutações dos inteiros 1, 2,..., n. (por exemplo, para $n = 3$ basta testar as entradas `[1,2,3]`, `[1,3,2]`, `[2,1,3]`, `[2,3,1]`, `[3,1,2]` e `[3,2,1]` para se garantir que a rede ordena correctamente qualquer entrada).
      ii. [4%] Quantas entradas é necessário testar se usarmos o Princípio 0/1? Compare (assintóticamente) com o número de testes mencionado na alínea anterior ($n!$).

5. Custo amortizado
   (a) [4%] Defina custo amortizado de uma sequência de operações.
   (b) [6%] Dê um exemplo de uma sequência de n operações em que o custo amortizado seja muito inferior ao custo máximo de uma operação, não sendo sequer da mesma ordem de grandeza exacta.

6. Ordenação estável
   (a) [4%] Defina algoritmo de ordenação estável,
   (b) [6%] Dê exemplo de um algoritmo de ordenação não estável, descrevendo-o numa linguagem informal.

7. **8%** "Quicksort"
Na demonstração (por indução) de que o número de comparações $t(n)$ efectuadas pelo algoritmo clássico "quicksort" satisfaz $t(n) \leq cn \ln n$ para determinado $c > 0$ a definir, usa-se o seguinte argumento indutivo

$$\begin{aligned}
t(n) &= (n-1) + \frac{1}{n}\sum_{i=0}^{n-1}[t(i) + t(n-1-i)] && (1)\\
&= (n-1) + \frac{2}{n}\sum_{i=1}^{n-1} t(i) && (2)\\
&\leq (n-1) + \frac{2c}{n}\sum_{i=1}^{n-1} i\ln i && (3)\\
&\leq (n-1) + \frac{2c}{n}\int_1^n x\ln x\,dx && (4)\\
&\leq (n-1) + \frac{2c}{n}\left(\frac{1}{2}n^2\ln n - \frac{n^2}{4} + \frac{1}{4}\right) && (5)\\
&\leq cn\ln n \quad \text{para} \quad c \geq 2 && (6)\\
&\in O(n\ln n) && (7)
\end{aligned}$$

Justifique cada passagem deste argumento. A sua justificação deve ter estar dividida do seguinte modo **a)** (1):... **b)** de (1) para (2):... **c)** de (2) para (3):... **d)** de (3) para (4):... **e)** de (4) para (5):... **f)** de (5) para (6):... **g)** de (6) para (7):...

8. Considere o uso da Programação Dinâmica na determinação da forma óptima de multiplicar matrizes. Sejam M_i, M_{i+1},... M_j as matrizes a multiplicar, tendo a matriz M_k dimensões $d[k] \times d[k+1]$ para $k = i,\ldots, j$.

   (a) **6%** Explique a equação $\boxed{c_{i,j} = \min_{i\leq p<j}\{c_{i,p} + d[i]d[p+1]d[j+1] + c_{p+1,j}\}}$ (8)
   Deve em particular: explicar o significado das variáveis $c_{a,b}$, i, j e p; explicar o que se passa quando $j = i+1$; deduzir a forma geral da equação (8).

   (b) **6%** Para as matrizes M_1, M_2 e M_3 de dimensões 50×100, 100×10 e 10×2, complete o quadro seguinte (na sua folha de prova) onde é sempre $j = i+m$. *Não apresente os cálculos intermédios.*

```
-------------------------------------------------------------------
m = 1:  i=1                            | i=2
        Produto matricial: M1 x M2     | Produto matricial:
        custo óptimo:      50000       | custo:
        p óptimo =         1           | p óptimo =
-------------------------------------------------------------------
m = 2:  i=1
        Produto matricial:
        custo óptimo = min{___,___} = ___
        p óptimo =
-------------------------------------------------------------------
```

   (c) **6%** Descreva numa linguagem informal um algoritmo para determinar o custo óptimo de uma multiplicação matricial; esse algoritmo deve ser baseado em (8) e obter as soluções parciais de forma "bottom-up" (começar pelos produtos matriciais com menos factores).

9. "Hash". Seja n o número de valores representado na tabela, u a dimensão do universo e a o tamanho da tabela.

   (a) **5%** Mostre que para qualquer função de "hash" h existe um conjunto de $\lceil u/a \rceil$ valores que são mapeados num mesmo índice da tabela A.

   (b) **4%** Defina "hash" perfeito.

   (c) **6%** Descreva em linhas gerais o algoritmo para a construção do "hash" perfeito com tabela de "hash" de tamanho $a = n^2$.