DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

AED - 2017-2018 - 1° Semestre
Algoritmos e Estruturas de Dados

2^o Exame, 30 de janeiro de 2018, 11h30m Duração: 3 horas

Prova escrita, individual e sem consulta

NOME:_____ NÚMERO:_____

PARTE I - Questões de Escolha Múltipla (A/B/C/D)

Preencha as respostas na tabela (usando apenas letras maiúsculas). Se nenhuma opção servir, escreva **NENHUMA**. Se pretender alterar a sua resposta, risque e escreva ao lado a sua nova opção. Todas as questões de escolha múltipla seguintes valem 0.75 valores. Estas questões de escolha múltipla não respondidas são cotadas com 0 valores, mas por cada resposta errada são descontados 0.75/4 valores.

Questão	1	2	3	4	5	6
Resposta						

1. Uma das seguintes tabelas não pode representar um acervo. Qual?

   A. 8-12-10-18-16-14-13-24-20-32-28-40 B. 8-10-12-16-18-13-14-20-24-28-32-40
   C. 8-12-10-18-16-13-14-24-20-32-38-40 D. 8-12-10-16-18-14-13-24-15-32-28-40

2. Após a aplicação de **3** passos de um dado algoritmo de ordenação numa tabela de inteiro, o estado da tabela é o seguinte:

0	2	8	4	7	9	13	10

   Qual dos algoritmos pode ter sido aplicado?

   A. *ShellSort (h=4,2,1)* B. *InsertionSort* C. *BubbleSort* D. *SelectionSort*

3. Da análise de complexidade temporal de um algoritmo concluiu-se ser $\mathcal{O}(N^3)$. Sabendo-se que o algoritmo é recursivo, qual das recorrências abaixo melhor descreve esse algoritmo?

   A. $C_N = 3C_{N-1} + 1$ B. $C_N = 8C_{N/2} + N^3$
   C. $C_N = C_{N-1} + N^2$ D. $C_N = 3C_{N/3} + N^2$

4. Aplicou-se DFS a um grafo não direccionado representado por matriz de adjacências para determinação da existência (ou não) de caminho simples entre o vértice 6 e o vértice 7. O caminho obtido foi o seguinte:

   6-4-8-1-0-3-7

   Qual dos seguintes caminhos nunca poderia ter sido obtido numa outra DFS?

   A. 3-4-8-1 B. 3-0-1-2 C. 7-8-1-0 D. 8-1-0-3

5. Assuma um grafo ponderado em que todas as arestas têm pesos distintos. Indique qual das seguintes afirmações é **falsa**

   A. A aresta de menor peso do grafo faz sempre parte da sua MST.
   B. A aresta de menor peso do grafo faz sempre parte da SPT iniciada num dos nós adjacentes a essa aresta.
   C. A aresta de menor peso do grafo faz sempre parte da SPT qualquer que seja o nó de partida.
   D. A aresta de maior peso do grafo pode fazer parte da sua MST

6. Suponha um conjunto de N números inteiros (N muito grande), organizados numa **tabela** sobre a qual é implementado o seguinte algoritmo: dado um número k, esse número é pesquisado na **tabela**; o algoritmo retorna TRUE ou FALSE respectivamente, consoante encontrou ou não o número k na **tabela**. Em média, **quantas comparações** são efectuadas durante a execução do algoritmo, admitindo que ele é o mais eficiente possível?

   A. $\mathcal{O}(k)$ se a **tabela** estiver ordenada e a pesquisa se iniciar pelo último elemento.
   B. $\mathcal{O}(\log N)$ se a **tabela** estiver ordenada.
   C. $\mathcal{O}(\log N)$ se não houver números repetidos na **tabela**.
   D. $\mathcal{O}(N)$ independentemente da **tabela** estar ordenada ou não.

PARTE II - Questões de Escolha Binária (V/F)

Preencha as respostas na tabela (usando apenas letras maiúsculas – **V**(erdadeira) ou **F**(alsa)). Todas as questões de escolha múltipla seguintes valem 0.50 valores. Estas questões de escolha múltipla não respondidas ou erradas são cotadas com 0 valores.

Questão	7	8	9	10	11	12	13	14
Resposta								

7. Ordenar por ordem decrescente os N elementos de uma lista ordenada por ordem crescente, nunca pode ser feito apenas em $\mathcal{O}(N)$ operações.

8. Se o caminho mais curto entre um par de vértices, v e u, passar por algum outro vértice, w, então a SPT em causa permite determinar o caminho mais curto entre v e w, mas não entre w e u.

9. Para efeitos da contabilização da complexidade temporal, define-se como instrução básica toda a instrução que se expresse numa só linha de código.

10. No problema da conectividade, usando o algoritmo da união rápida ponderada com compressão de caminho, quando os pares introduzidos são sempre raízes (de conjuntos diferentes), a compressão não produz qualquer efeito.

11. Dos algoritmos de ordenação estudados, o *SelectionSort()* é sempre o que faz o menor número de trocas.

12. Numa árvore binária ordenada podem existir elementos à esquerda da raíz que possuam chave maior que a da raíz.

13. A operação abstracta de procura por um objecto, cuja chave de prioridade seja dada, para uma colecção de objectos armazenados num acervo de acordo com essa chave tem, no pior caso, complexidade logarítmica no número de objectos presentes.

14. Ordenar por ordem decrescente os N elementos de uma tabela ordenada por ordem crescente, pode ser feito em $\mathcal{O}(N)$ operações.

PARTE III - Questões de Desenvolvimento

Responda a cada uma das questões de desenvolvimento em **folhas de exame separadas** e devidamente **identificadas com nome e número.**

[5.0] 15. Dada uma sequência de números inteiros positivos, dizemos que um número é dominante se ele é maior do que a soma dos elementos que aparecem depois dele na sequência. Assuma que os números da sequência estão guardados numa tabela, `sequence[]`. Considere a seguinte função para identificar que números na sequência representada por `sequence[]` são dominantes. Assuma que a tabela `domin[]`, que contém booleanos, foi inicializada a $FALSE$ antes da chamada à função. No final da execução, as posições de `domin[]` que estiverem a $TRUE$ indicam os números que são dominantes.

```
void dominante (int sequence[], bool domin[], int N)
{
  int i, j, sum;

  for (i = 0 ; i < N ; i++) {
    sum = 0;
    for (j = i+1 ; j < N ; j++)
      sum = sum + sequence[j];

    if (sequence[i] > sum) domin[i] = TRUE;
    else domin[i] = FALSE;
  }
}
```

[1.0] a) Identifique, justificando, a complexidade do algoritmo descrito.

[1.5] b) Sem escrever código descreva um algoritmo mais eficiente do que o apresentado, utilizando a mesma memória. Identifique, justificando, a complexidade do algoritmo que descreveu.

[2.5] c) Escreva o código em C que implementa o algoritmo que descreveu na alínea anterior assumindo que a assinatura da função é a mesma que a dada em a). Note que no seu algoritmo pode usar qualquer outro algoritmo estudado na disciplina de AED desde que identifique claramente qual o algoritmo e a sua complexidade.

Responda em folhas separadas. Não se esqueça de mudar de folha

---

[4.5] 16. Considere o grafo ponderado e não direccionado representado pela matriz de adjacências abaixo. Ausência de aresta é representada por ausência de entrada na matriz. Por exemplo, não existe aresta de 3 para 9, por isso a linha do vértice 3 não possui entrada na coluna do vértice 9.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
0	0	6	3	4	9							
1	6	0			6	10						
2	3		0	7							2	8
3	4		7	0	4			12				
4	9	6		4	0	7	3	8				
5		10			7	0	7		11			
6					3	7	0	3	4			
7				12	8		3	0			9	7
8						11	4		0	5	4	
9									5	0	3	
10			2					9	4	3	0	6
11			8					7			6	0

[2.5] a) Calcule a árvore mínima de suporte (MST) do grafo, usando o **algoritmo de Prim**, tomando o vértice 5 como ponto de partida. Justifique detalhadamente os seus cálculos.

[1.0] b) Trace a MST calculada e indique o seu custo total.

[1.0] c) Suponha que para um grafo de 8 vértices, numerados de 0 a 7, se calculou uma SPT e uma MST. Para a SPT obteve-se os seguintes dois vectores

`st` – [7 2 4 1 -1 0 2 4] `wt` – [8 9 3 15 0 13 7 6]

Indique justificando, se os dois seguintes vectores poderiam ter sido obtidos para a MST do mesmo grafo

`st` – [3 -1 1 1 2 0 2 1] `wt` – [4 0 6 6 3 5 4 3]

Sugestão: Comece por traçar as árvores associadas com cada um dos pares de vectores. Justifique esse traçado.

Responda em folhas separadas. Não se esqueça de mudar de folha

---

[2.0] 17. Suponha que se pretende desenvolver um algoritmo eficiente para, dada uma tabela `vec` contendo N números inteiros positivos, produzir na saída a mesma tabela mas em que os números pares aparecem todos antes dos números ímpares. Adicionalmente, com base num segundo parâmetro, pode ser pedido que os números pares e os números ímpares apareçam ordenados entre si. A assinatura da função seria então

```
void SeparateEvenOdd(int *vec, int N, int sortWithin),
```

em que quando o 3^o argumento, `sortWithin`, for 1, os elementos devem ser ordenados conforme indicado, e quando for diferente de 1 os elementos podem ser apresentados por qualquer ordem (mas sempre os pares primeiro do que os ímpares).

Nota: todo o procedimento tem de ser feito *in-place*, ou seja apenas pode ser utilizada memória de dimensão constante (variáveis escalares) para além do espaço ocupado por `vec`.

[1.25] a) **Sem escrever código**, descreva, **em menos de 15 linhas**, um algoritmo que resolva o problema quando `vec` $\neq 1$. Indique, justificando, qual a complexidade do algoritmo descrito.

[0.75] a) **Sem escrever código**, descreva, **em menos de 15 linhas**, um algoritmo que resolva o problema quando `vec` $= 1$. Indique, justificando, qual a complexidade do algoritmo descrito.