DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

# AED - 2018-2019 - 2º Semestre
# Algoritmos e Estruturas de Dados

$1^o$ Exame, 17 de Junho de 2019, 18h30m  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. Para o problema da conectividade, os algoritmos da união rápida ponderada com e sem compressão de caminho necessitam de um par de tabelas: a tabela `id[.]` e a tabela `sz[.]`. Indique qual dos pares abaixo representa um conjunto legítimo que possa ter sido produzido por algum daqueles dois algoritmos.

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A. | `id =` | 6 | 6 | 2 | 2 | 1 | 9 | 6 | 9 | 2 | 9 | `sz =` | 1 | 2 | 3 | 1 | 1 | 1 | 4 | 1 | 1 | 4 |
| B. | `id =` | 6 | 7 | 9 | 3 | 3 | 8 | 3 | 7 | 7 | 3 | `sz =` | 1 | 1 | 1 | 6 | 1 | 1 | 2 | 4 | 2 | 2 |
| C. | `id =` | 4 | 8 | 4 | 4 | 4 | 8 | 5 | 4 | 8 | 4 | `sz =` | 1 | 1 | 1 | 1 | 6 | 1 | 1 | 1 | 4 | 1 |
| D. | `id =` | 3 | 0 | 3 | 3 | 3 | 5 | 5 | 2 | 5 | 0 | `sz =` | 3 | 1 | 1 | 7 | 1 | 3 | 1 | 1 | 1 | 1 |

2. Para alocar dinamicamente uma matriz de $10 \times 10$ inteiros de 32 bits numa máquina de 64 bits gastam-se:

A. 3520 bits  B. 3840 bits  C. 6720 bits  D. 7040 bits

3. Diz-se que $g(N) \in \mathcal{O}(f(N))$ se...

A. ... para toda a constante real positiva, $c$, existir um natural $N_0$, tal que para todo o $N > N_0$, $g(N) \leq cf(N)$.
B. ... existir uma constante real positiva, $c$, e um natural $N_0$, tal que para todo o $N > N_0$, $f(N) \leq cg(N)$.
C. ... para todo o natural $N_0$ existir uma constante real positiva, $c$, tal que para todo o $N > N_0$, $g(N) \leq cf(N)$.
D. ... existir uma constante real positiva, $c$, e um natural $N_0$, tal que para todo o $N > N_0$, $g(N) \leq cf(N)$.

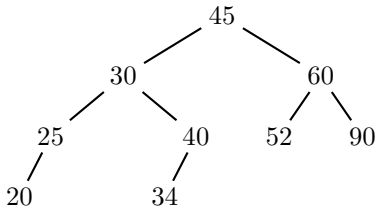
4. Considere a tabela indicada na 1a linha abaixo. As linhas seguintes correspondem, cada uma delas, ao resultado da aplicação de **dois** passos de um dado algoritmo de ordenação.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| T | 23 | 19 | 20 | 17 | 12 | 15 | 16 | 15 | 13 | 14 |
| S1 | 19 | 20 | 23 | 17 | 12 | 15 | 16 | 15 | 13 | 14 |
| S2 | 12 | 13 | 20 | 17 | 23 | 15 | 16 | 15 | 19 | 14 |
| S3 | 12 | 13 | 23 | 19 | 20 | 17 | 14 | 15 | 16 | 15 |

Indique das seguintes correspondências entre tabelas (S1, S2 e S3) e algoritmos, qual é a correcta.

| | | | |
|---|---|---|---|
| A. | S1 - *Selection*, S2 - *Insertion*, S3 - *Bubble* | B. | S1 - *Selection*, S2 - *Bubble*, S3 - *Insertion* |
| C. | S1 - *Insertion*, S2 - *Selection*, S3 - *Bubble* | D. | S1 - *Bubble*, S2 - *Insertion*, S3 - *Selection* |

5. Considere a árvore binária ordenada, balanceada AVL ao lado. Supondo que nessa árvore se procede à inserção do número 32, indique qual das afirmações é verdadeira.




A. A árvore permanece balanceada.
B. É necessário executar uma rotação simples à esquerda no vértice 30.
C. É necessário executar uma rotação dupla à esquerda no vértice 30.
D. É necessário executar uma rotação simples à direita no vértice 40.

6. Considere uma tabela de dispersão de índices livres, com resolução de colisões por procura linear e função de dispersão $f(x) = x\%100$, que se encontra parcialmente preenchida após algumas inserções. Abaixo apresenta-se o conteúdo das primeiras 10 posições, em que **E** representa posição não ocupada.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 49899 | 30400 | **E** | 2803 | 14303 | 1804 | **E** | 4107 | **E** | **E** |

Assumindo que a tabela estava originalmente vazia, qual das seguintes é uma sequência de inserção consistente com aquele preenchimento?

A. {4107; 2803; 6898; 14303; 38044; 598; 2803; 49899; 1804; 30400}
B. {6898; 2803; 38044; 4107; 598; 2803; 1804; 49899; 30400; 14303}
C. {2803; 14303; 598; 4107; 38044; 6898; 30400; 48899; 2803; 1804}
D. {598; 38044; 2803; 6898; 2803; 1804; 14303; 49899; 30400; 4107}

---

## 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. Para o problema da conectividade, o algoritmo "Quick-find" tem complexidade $\mathcal{O}(1)$ na operação abstracta de união.

8. Numa implementação de números complexos como um tipo abstracto, em que a implementação seja privada, só se pode obter o valor, por exemplo, da parte real do complexo `_t_` usando a função `Re(_t_)` (ou similar).

9. Implementando uma pilha através de lista simplesmente ligada tem como consequência que a operação abstracta `pop(.)` deixa de ser $\mathcal{O}(1)$.

10. Um algoritmo recursivo, cuja complexidade temporal seja descrita pela recorrência $C_N = C_{N-1} + \lg(N^2)$, tem complexidade $\mathcal{O}(N)$.

11. O algoritmo de ordenação "Quicksort" tem, no pior caso, complexidade $\mathcal{O}(N^2)$.

12. Se a determinação de uma MST (*Minimal Spanning Tree*) num grafo não direccionado com $V$ vértices produzir $V-1$ arestas, então a determinação de uma SPT (*Shortest Path Tree*), qualquer que seja o vértice de partida, também conterá sempre $V-1$ arestas.

13. Numa tabela de dispersão de tamanho $M$ que contenha $N$ objectos e onde a resolução de colisões se faz por dispersão em lista, saber se um dado elemento está incluído nalguma das listas dessa tabela, no pior caso, tem complexidade $\mathcal{O}(M)$.

14. Num acervo, qualquer vértice que possua dois filhos pode ter o filho mais prioritário em qualquer dos lados, esquerdo ou direito.

---

## 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.

[4.5] 15. Considere um grafo ponderado não direccionado com 12 vértices, identificados de 0 a 11, representado pela matriz de adjacências indicada abaixo. A ausência de valor para um par linha e coluna significa ausência de aresta entre esses dois vértices, i.e., custo $\infty$.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | | 14 | | 15 | | | | 12 | | | |
| 1 | | 0 | | 7 | | 20 | 3 | | | 3 | | |
| 2 | 14 | | 0 | | 24 | | 5 | | | | 18 | 10 |
| 3 | | 7 | | 0 | | 28 | | | | 11 | | |
| 4 | 15 | | 24 | | 0 | | 6 | 26 | | | | 23 |
| 5 | | 20 | | 28 | | 0 | | | | 21 | | |
| 6 | | 3 | 5 | | 6 | | 0 | | | | | |
| 7 | | | | | 26 | | | 0 | | | | |
| 8 | 12 | | | | | | | | 0 | | 31 | |
| 9 | | 3 | | 11 | | 21 | | | | 0 | | |
| 10 | | | 18 | | | | | | 31 | | 0 | |
| 11 | | | 10 | | 23 | | | | | | | 0 |

[2.0] a) Tomando o vértice 4 como ponto de partida determine a árvore de mínima de suporte. Justifique os seus cálculos de forma clara, assim como todas as decisões tomadas ao longo da determinação da árvore.

[1.0] b) Trace a árvore obtida na alínea anterior e explique se pode ou não e porquê usar essa árvore para determinar o caminho mais curto entre o vértice 4 e o vértice 5. Podendo, indique qual é o caminho e o seu custo.

[1.5] c) Tomando agora o vértice 5 como fonte determine a sequência de visitas aos vértices do grafo fazendo uso de procura em profundidade primeiro (DFS).

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

[5.0] 16. Em geral considera-se que uma *string* é um conjunto de caracteres todos diferentes do espaço em branco. No entanto é possível que uma *string* também contenha espaços em branco. Neste contexto pode ser necessário determinar qual a maior sequência de espaços em branco existente numa dada *string*, assim como poderia ser necessário determinar a maior sequência com a repetição de um outro qualquer caracter. O que torna este problema interessante é que, para uma *string* de tamanho $S$ muito grande, quanto maior for a maior sequência de brancos mais rápida deveria ser a determinação desse tamanho.

[1.5] a) Proponha um algoritmo eficiente para a determinação do comprimento da maior sequência de brancos numa *string*. O seu algoritmo deverá ser tal que se examina o menor número possível de caracteres na *string*. A sua descrição do algoritmo deverá ser clara e sem ambiguidades, assim como deverá explicar como é que o seu algoritmo fica progressivamente mais rápido à medida que aumenta o tamanho da maior sequência de brancos já encontrada.

[2.5] b) Escreva uma função em linguagem C que implemente o algoritmo que propôs. A sua função deverá ter a seguinte assinatura:

```
int max_blank_sequence(char * s1);
```

A função deverá retornar o comprimento, quantos caracteres seguidos, da maior sequência de espaços em branco. Por exemplo, se `s1` for:

```
"Às+vezes++++existem++espaços+em++++++++branco+nas+sequências+de+caracteres",
```

a função deverá retornar 8, porque entre `em` e `branco` existem 8 espaços em branco (aqui representados com o símbolo `+` para facilitar a leitura). Na sua implementação apresente também um conjunto mínimo de comentários descrevendo os aspectos mais importantes da mesma.

**Nota:** Tanto nesta como na alínea anterior a cotação máxima será atribuída à solução mais eficiente.

[1.0] c) Assumindo uma *string* de tamanho $S$ e que a maior sequência de espaços em branco tem tamanho $B << S$, discuta a complexidade temporal do seu algoritmo como função de $S$ e $B$. Note que se poderá ter de aplicar a função a *strings* de dimensão muito grande e que a utilização da função `strlen(.)` nesse contexto não pode ser considerada como uma instrução básica.

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

[2.0] 17. Dado um grafo, representado por vector de listas de adjacências, suponha que se pretende adicionar ao respectivo ADT (*Abstract Data Type*) uma função que identifica todos os vértices que não estejam mais longe que $d$ de um vértice dado.

Sem escrever código diga como procederia para resolver este problema, sendo que a sua proposta deverá permitir o cálculo desses vértices da forma mais eficiente possível. Indique, de forma devidamente justificada, qual a complexidade do algoritmo por si proposto, assim como deve ser claro por que razão (ou razões) o seu algoritmo está correcto.