DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

AED - 2015-2016 - 2º Semestre
Algoritmos e Estruturas de Dados

2º Exame, 20 Junho 2016, 18:30h 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	7	8
Resposta								

[0.75] 1. Após se aplicar o algoritmo de Dijkstra a um dado grafo **não direccionado**, com 15 vértices numerados de 0 a 14, obteve-se o vector `st` que se apresenta abaixo.

2	5	7	8	8	9	7	7	6	7	5	8	9	2	0

Das seguintes opções, e de acordo com o vector `st` produzido, qual **não** constitui um caminho mais curto entre o vértices extremos da sequência apresentada.

A. 7-2-0-14 B. 5-9-7 C. 7-6-9-12 D. 6-8-3

[0.75] 2. Considere que um dado algoritmo recursivo tem a sua complexidade de memória utilizada descrita pela expressão: $C_N = 3C_{N/2} + N \lg N$. Qual das seguintes hipóteses constitui o menor majorante da memória ocupada pelo algoritmo?

A. $N \lg N$ B. $N^{\lg_2 3} \lg N$ C. $N^{\lg_2 3}$ D. $N \lg^2 N$

[0.75] 3. Considere o código abaixo à esquerda, analise-o e indique qual das expressões propostas descreve a complexidade temporal da função apresentada.

```
void silly(int n, int x, int y) {
 int i, j;
 if (x < y) {
  for (i = 0; i < n; i++)
   for (j = 0; j < n * i; j++)
    printf("y = %d", y);
 }
 else
  printf("x = %d", x);
}
```

A. $O(n)$

B. $O(n \log(n))$

C. $O(n^2)$

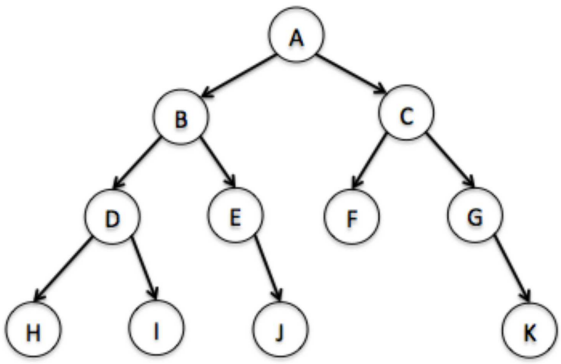
D. $O(n^3)$

[0.75] 4. Seja o problema de inserir num acervo inicialmente vazio a seguinte sequência de números: 9, 3, 2, 6, 8, 4, 1.

Considerando que a prioridade é dada aos números menores e que se pretende contabilizar exclusivamente o número de "fixUp"ou "fixDown"que produzem alteração após inserção, indique quantas destas operações são realizadas durante a inserção da sequência indicada.

A. 3	B. 4	C. 5	D. 6

[0.75] 5. Considere a seguinte árvore



Qual das seguintes sequências abaixo corresponde a um varrimento in-fixadoda árvore.

A.	A	B	D	H	I	E	J	C	F	G	K
B.	H	D	I	B	E	J	A	C	F	G	K
C.	H	D	I	B	E	J	A	F	C	G	K
D.	A	B	C	D	E	F	G	H	I	J	K

[0.75] 6. Considere a tabela de dispersão de dimensão 7 [-, 8, 15, 22, -, 33, 26]obtida após a introdução dos 5 números indicados. Para tal, utilizou-se a função de dispersão $H(k) = k \bmod 7$ e a resolução de colisões foi por procura linear. Qual das seguintes sequências de números não produz a tabela apresentada?

A. 8; 33; 15; 26; 22	B. 33; 26; 8; 15; 22	C. 8; 15; 22; 33; 26	D. 33; 8; 22; 15; 26

[0.75] 7. Suponha que se tem números entre 1 e 100 inseridos numa árvore de procura binária e se pretende encontrar (se existir) o número 45. Qual das seguintes sequências não pode ser uma sequência de nós examinados?

A.	1	2	3	4	5	6	7	8	9	10	11
B.	50	25	27	29	31	35	37	39	42	44	43
C.	50	25	30	31	32	33	34	35	36	37	38
D.	20	80	25	75	30	70	35	65	40	60	50

[0.75] 8. Suponha que se pretende construir uma árvore binária ordenada, satisfazendo o balaneamento AVL, de tal forma que a árvore deverá permanecer balanceada após cada inserção. Contando cada rotação dupla como duas rotações, para uma árvore inicialmente vazia, indique quantas rotações são realizada se forem inseridos os seguintes elementos: 62; 31; 70; 91; 25; 11; 9; 61; 73; 6.

A. 3	B. 4	C. 5	D. 6

PARTE II - Questões de Escolha Múltipla (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	9	10	11	12	13	14	15	16
Resposta								

[0.50] 9. O algoritmo de Prim calcula correctamente uma árvore mínima de suporte de um grafo que tenha algumas das arestas com pesos negativos.

[0.50] 10. Para dois algoritmos diferentes, seja $C_1(N)$ a complexidade do primeiro e $C_2(N)$ a complexidade do segundo. Se $C_1(N) \in \mathcal{O}(f(N))$ e $C_2(N) \in \mathcal{O}(f(N))$, então $C_1(N) \in \mathcal{O}(C_2(N))$ e $C_2(N) \in \mathcal{O}(C_1(N))$.

[0.50] 11. Para um acervo de n elementos distintos, o elemento mais prioritário fica na 1ª posição (índice 0), enquanto o quarto elemento mais prioritário pode ficar **apenas** nas posições 4 ou 5 (índice 3 ou 4, respectivamente).

[0.50] 12. Considere um grafo não direccionado e o problema de saber se existe algum vértice isolado. Para resolver este problema é computacionalmente mais eficiente utilizar matriz de adjacências do que lista de adjacências..

[0.50] 13. Numa fila com prioridade estática implementada através de acervo contendo apenas dois elementos de igual prioridade, se se aplicar a operação de RemoveMax o elemento retornado é o elemento que foi inserido primeiro.

[0.50] 14. Suponha que está a ordenar uma tabela de 8 números através do heapsort, e que a tabela está actualmente na seguinte configuração: 6,4,5,1,2,7,8. Pode-se então dizer que já foram realizadas duas iterações do heapsort.

[0.50] 15. Numa lista duplamente ligada com 10 nós, é necessário alterar quatro ponteiros caso se pretenda apagar um nó que não seja o primeiro.

[0.50] 16. A programação dinâmica ascendente é adequada para desenvolver código recursivo em problemas que se decompõem em sub-problemas interdependentes.

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.

[4.0] 17. Neste exercício pretende-se que desenvolva um par de funções de manipulação de listas simples ordenadas em que cada elemento é definido como está abaixo.

```
typedef struct _lista {
 int item;
 struct _lista * next;
} lista;
```

[1.75] (a) Escreva uma função em C que receba duas listas ordenadas e as junte numa só, devolvendo um ponteiro para a lista resultante. A função deve ter a seguinte assinatura:

```
lista * merge(lista *l1, lista * l2);
```

a cotação máxima será atribuída a uma implementação recursiva.

[1.50] (b) Pretende-se agora que escreva uma função em C que remova um ou mais elementos da lista. A função recebe como argumentos um ponteiro para uma lista ordenada e um inteiro, devendo devolver um ponteiro para uma lista em que todos os elementos que tenham esse inteiro sejam removidos, ou devolver a lista original se nenhum dos elementos contiver o inteiro passado como argumento.

A função deverá ter a seguinte assinatura:

```
lista * removealloccurrences(lista *l, int n);
```

na sua implementação não se deverá esquecer que a lista está ordenada. Uma implementação recursiva (correcta) receberá crédito extra.

[0.75] (c) Discuta a complexidade temporal de ambas as funções anteriores, identificando as dimensões apropriadas para caracterizar cada problema.

[4.0] 18. Considere um grafo ponderado, não direccionado, cuja matriz de adjacências se apresenta abaixo e em que os vértices estão numerados a partir de zero.

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

[0.25] (a) Produza uma lista de todas as arestas, ordenada por peso crescente, identificando os vértices que ligam.

[1.50] (b) Determine a árvore mínima de suporte, por aplicação do algoritmo de Kruskal. Justifique convenientemente os seus cálculos.

[1.50] (c) Determine a árvore mínima de suporte, por aplicação do algoritmo de Prim, usando o vértice 0 como fonte. Justifique convenientemente os seus cálculos.

[0.75] (d) Trace as árvores obtidas por ambos os algoritmos e justifique as diferenças que encontrar.

[2.0] 19. Dado um grafo ponderado e direccionado, G, define-se como a *capacidade máxima de um caminho* o valor da aresta de menor peso nesse caminho.

[1.25] a) Dados um digrafo ponderado G, dois vértices distintos s e t e um valor *threshold* T, construa um algoritmo que seja capaz de encontrar um caminho entre s e t de capacidade máxima maior ou igual a T, ou então que indique que tal caminho não existe. Sem fazer código, descreva o algoritmo de forma precisa e justificada.

[0.75] b) Assumindo V vértices e E arestas, indique qual a complexidade do algoritmo que desenhou. Justifique as suas conclusões. Na sua análise da complexidade temporal não se esqueça de indicar se as suas conclusões são as mesmas ou diferentes quer G esteja representado por matriz de adjacências ou por vector de listas de adjacências.