UNIVERSIDADE ABERTA
www.uab.pt

Ministério da Educação e Ciência

U.C. 21046

Estruturas de Dados e Algoritmos Fundamentais

22 de setembro de 2016

INSTRUÇÕES

- Leia estas instruções na totalidade antes de iniciar a resolução do teste.
- O enunciado do teste é constituído por 5 grupos de questões, tem 4 páginas e termina com a palavra FIM.
- Se o seu exemplar não estiver completo ou nele se verificar qualquer outra deficiência, por favor dirija-se ao professor vigilante.
- O teste deve ser resolvido na sua totalidade em folhas de respostas.
- Nas respostas, tenha a preocupação de utilizar uma letra legível.
- Todas as respostas devem ser escritas unicamente com caneta azul ou preta.
- O teste é SEM CONSULTA. Todos os elementos necessários à resolução são fornecidos no enunciado.
- É permitido utilizar máquina de calcular.
- As cotações são indicadas por grupo e nas próprias questões.
- Nas questões de escrita de programas, a sua correção terá em conta critérios de proficiência e compreensibilidade do código (legibilidade, indentação, estrutura, comentários e explicação geral).
- O não cumprimento das instruções implica a anulação das respetivas questões.
- O tempo de realização do teste é de 150 minutos.

Grupo I [3 valores]

1.1. [1] Utilizando a definição, prove que $f(n) = n^2 - 2n + 8$ é $\Omega(n^2)$.

1.2. [1] Para cada um dos seguintes pares de funções $f(n)$ e $g(n)$, indique se $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$, $f(n) = \Theta(g(n))$ ou nenhum dos casos.

1. $f(n) = n^3 + 10n^2 + \sqrt{n}$, $g(n) = n^3 - n + 1000$
2. $f(n) = (n+3)^2$, $g(n) = n\log_2(n) + 10\sqrt{n} + 2$

1.3. [1] Considere a complexidade do seguinte segmento de código em termos do nº $f(n)$ de operações aritméticas realizadas na variável a. Determine a expressão de $f(n)$ e indique a sua complexidade na notação $O(.)$.

```
for(a=0,i=1; i<=n*n; i++)
   for(j=i; j<=n*n; j++)
      a++;
```

Grupo II [5 valores]

2.1. [1] Considere uma árvore de pesquisa binária inicialmente vazia na qual foram inseridas as chaves 7, 3, 2, 1, 5, 6, 9, 8, 10, 4 pela ordem indicada. Desenhe a árvore obtida após efetuadas todas as inserções. Nas alíneas seguintes considere a árvore obtida como a árvore "original".

2.2. [1] Remova da árvore original a chave 3 utilizando o algoritmo de remoção por fusão (Deletion by Merging). Desenhe a árvore obtida.

2.3. [1] Efetue na árvore original uma rotação à direita de 3 em torno de 7. Desenhe a árvore obtida.

2.4. [2] Considere uma árvore B^+-Tree de ordem 3 inicialmente vazia. Desenhe a árvore após cada uma das seguintes operações de inserção (I) e remoção (R) pela ordem indicada: I 14, 18, 32, 17, 15, R 14 (total de 6 desenhos).

Grupo III [4 valores]

3.1. [2] Considere uma tabela T de dispersão (hash) com dimensão 9 e função de hash $h(x) = x \bmod 9$. As colisões são resolvidas com sondagem (probing) linear. Considerando a tabela inicialmente vazia, determine o conteúdo final da tabela após a inserção das chaves 4, 9, 13, 20, 14, 11, 2 pela ordem apresentada. Justifique os cálculos efetuados para cada inserção.

3.2. [2] Considere o vetor [3 6 1 7 2 5 8 9 4]. Indique a sequência de passos para a sua ordenação utilizando o algoritmo Quicksort.

Grupo IV [4 valores]

4. Considere as seguintes classes para implementação de uma estrutura de dados tipo lista simplesmente ligada (Single Linked List) em que os itens são inteiros.

```
// definir nó
class ISLLNode {
public:
   // atributos
   int info;              // item
   ISLLNode *next;        // sucessor
   // construtores
   ISLLNode() {next=0;}
};

// definir lista simplesmente ligada
class ISLL {
private:
   // atributos
   ISLLNode *head;      // primeiro nó
public:
   // construtores
   ISLL() {head=0;}     // cria lista vazia
   // métodos
   void insertTail(int x);
   void deleteHead();
    int insertPos(int pos, int x);
};
```

Na resolução das alíneas seguintes pode criar outros métodos e/ou construtores que achar convenientes. Explique em termos gerais o funcionamento do código e indique situações especiais ou casos particulares que necessitem de ser considerados.

4.1. [1] Implemente o método "void insertTail(int x)"que insere um item no fim da lista.

4.2. [1] Implemente o método "void deleteHead()"que remove o primeiro item da lista.

4.3. [2] Implemente o método "int insertPos(int pos, int x)"que insere um item na lista na posição pos. Considera-se que as posições começam em 0. O método deve retornar 0 em caso de sucesso e -1 se a posição indicada não for válida.

Grupo V [4 valores]

5. Considere as seguintes classes para implementação de uma estrutura de dados tipo árvore de pesquisa binária (Binary Search Tree) em que os itens são genéricos.

```
// definir nó
template<class T>
class BSTNode {
public:
   // atributos
   T info;                  // item
   BSTNode *left, *right;   // filhos
   // construtores
   BSTNode() {left=right=0;}
};

// definir árvore BST
template<class T>
class BST {
private:
   // atributos
   BSTNode<T> *root;    // raiz
public:
   // construtores
   BST() {root= 0;}     // cria BST vazia
   // métodos
   void postorder();
    int deleteByMerging(const T& x);
};
```

Na resolução das alíneas seguintes pode criar outros métodos e/ou construtores que achar convenientes. Explique em termos gerais o funcionamento do código e indique situações especiais ou casos particulares que necessitem de ser considerados.

5.1. [1] Implemente o método "void postorder()"que efetua a travessia da árvore em pós-ordem (postorder). Visitar um nó significa imprimir o nó (atributo info).

5.2. [3] Implemente o método "int deleteByMerging(const T& x)"que remove um item da árvore utilizando o algoritmo de remoção por fusão. O método deve retornar 0 em caso de sucesso e -1 se o item indicado não for encontrado.

FIM