UNIVERSIDADE ABERTA
www.uab.pt

Ministério da Ciência, Tecnologia e Ensino Superior

U.C. 21046

Estruturas de Dados e Algoritmos Fundamentais

15 de setembro de 2017

INSTRUÇÕES

- Leia estas instruções na totalidade antes de iniciar a resolução da prova.
- O enunciado da prova é constituído por 5 grupos de questões 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.
- A prova deve ser resolvida 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 da prova é de 150 minutos.

Grupo I [3 valores]

1.1. [1] Utilizando a definição, prove que $f(n) = n + \sqrt{n}$ é $O(n)$.

1.2. 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.

1.2.1. [0.5] $f(n) = n\sqrt{n}$, $g(n) = n\log_2(n)$

1.2.2. [0.5] $f(n) = 10000 + n$, $g(n) = n^2$

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; i++)
   for(j=1; j<=i; j++)
      for(k=1; k<=n; k++)
         a++;
```

Grupo II [5 valores]

2.1. [2] Considere uma árvore binária do tipo max Heap inicialmente vazia na qual foram inseridas as chaves 7, 3, 2, 1, 5, 6, 9, 8 pela ordem indicada. Indique na forma de vetor o Heap após cada inserção (apresente 8 vetores).

2.2. [1] Remova do Heap obtido na alínea anterior a chave de maior valor. Indique na forma de vetor o Heap resultante.

2.3. [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 3, 20, 24, 12, 8, R 3 (total de 6 desenhos).

Grupo III [4 valores]

3.1. [2] Considere o vetor [0 6 3 2 7 9 5 4 8]. Indique a sequência de passos para a sua ordenação utilizando o algoritmo Insertion sort. Justifique de um modo geral o funcionamento do algoritmo.

3.2. [2] Considere o vetor [9 2 5 3 8 6 1 7 4]. Indique a sequência de passos para a sua ordenação utilizando o algoritmo Mergesort. Justifique de um modo geral o funcionamento do algoritmo.

Grupo IV [4 valores]

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

```
// definir nó (duplamente ligado)
class IDLLNode {
public:
   // atributos
   int info;                // item
   IDLLNode *prev, *next;   // antecessor e sucessor
};

// definir lista duplamente ligada
class IDLL {
private:
   // atributos
   IDLLNode *head, *tail;   // primeiro e último nó
public:
   // construtores
   IDLL() {head= tail= 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
};

// definir árvore BST
template<class T>
class BST {
private:
   // atributos
   BSTNode<T> *root;    // raiz
public:
   // construtores
   BST() {root= 0;}     // cria BST vazia
   // métodos
   void inorder();
    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 inorder()"que efetua a travessia da árvore em ordem (inorder). 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