UNIVERSIDADE AbERTA www.uab.pt

Ministério da Ciência, Tecnologia e Ensino Superior

U.C. 21046

Estruturas de Dados e Algoritmos Fundamentais

27 de setembro de 2018

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 $f(n) = 2^{n-1} + n^2$ é $\Omega(2^n)$.

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] $f(n) = (n+1)^2$, $g(n) = n \log_2 [(n+10)^4]$

1.2.2. [0.5] $f(n) = n^4 + 5n^3 + 2n^2 + 10$, $g(n) = n^4 + 100$

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=2; i<=n; i*=2)
   for(j=1; j<=n; ++j)
      a++;
```

Grupo II [5 valores]

2.1. Considere uma árvore de pesquisa binária (BST) inicialmente vazia.

2.1.1. [1] Insira na árvore as chaves 15, 6, 5, 3, 9, 12, 21, 18, 28, 8 pela ordem indicada. Desenhe a árvore obtida após efetuadas todas as inserções. Nas alíneas seguintes considere a árvore obtida como a árvore "original".

2.1.2. [1] Remova da árvore original a chave 6 utilizando o algoritmo de remoção por fusão (Deletion by Merging), escolhendo a opção em que o nó removido é substituído pelo seu filho direito. Desenhe a árvore obtida.

2.1.3. [1] Efetue na árvore original uma rotação à esquerda de 21 em torno de 15. Desenhe a árvore obtida.

2.2. [2] Considere uma árvore B^+ (B^+-Tree) de ordem 3 inicialmente contendo apenas o nó raiz com as chaves 1, 15. Desenhe a árvore após cada uma das seguintes operações de inserção (I) e remoção (R) pela ordem indicada: I 12, 20, 17, 3, 7; R 15; (total de 6 desenhos).

Grupo III [4 valores]

3. Considere o vetor [7 6 3 2 1 9 5 4 8]. Ordene o vetor utilizando o algoritmo de ordenação indicado. Indique a sequência de vetores obtida correspondente às iterações principais ou ciclo externo do algoritmo. Justifique de um modo geral o funcionamento do algoritmo.

3.1. [2] Algoritmo de ordenação por seleção (Selection Sort).

3.2. [2] Algoritmo de ordenação Comb Sort.

Grupo IV [4 valores]

4. Pretende-se conceber em C++ uma estrutura de dados tipo lista circular simplesmente ligada (circular single linked list) em que os itens são inteiros. A lista deve suportar as operações de inserir itens no final da lista (insert_end) e remover (delete) itens da lista.

4.1. [1] Defina as classes que entender necessárias para a implementação da lista. Defina apenas atributos e métodos, não inclua código para os métodos.

4.2. [1.5] Implemente o método "insert_end"que insere um item no final da lista.

4.3. [1.5] Implemente o método "delete"que remove um item da lista. Admita que o item está na lista.

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.

```
template<class T>   // definir nó
class BSTNode {
public:
   T info;                  // item
   BSTNode *left, *right;   // filhos
};

template<class T>   // definir árvore BST
class BST {
private:
   BSTNode<T> *root;    // raiz
public:
   void preorder();
    int deleteByCopying(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 preorder()"que efetua a travessia da árvore em pré-ordem (preorder). Visitar um nó significa imprimir o nó (atributo info).

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

FIM