Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Algoritmos e Estruturas de Dados

Semestre de Inverno 2008/09

Teste final, época especial (2h30m)

I (4 v)

1. (1,5) Resolva a seguinte recorrência, usando notação assimptótica para representar o resultado na forma de uma função explícita.

   $C(n) = 3C(n/3) + O(n)$

2. (1,5) Considere uma aplicação que lê do *standard input* uma sequência de palavras e escreve no *standard output* as palavras que ocorrem apenas uma vez. Descreva um algoritmo para esta aplicação.

   Nota: as palavras repetidas não estão necessariamente seguidas.

3. (1) Esquematize a inserção da seguinte sequência de elementos numa *B-Tree* com $M = 2T - 1 = 3$ (número máximo de chaves por nó/página):

   $\{0, 10, 20, 2, 3, 21, 22, 4, 23, 5, 24\}$.

II (16 v)

Nota: a resolução das questões deste grupo pode utilizar métodos ou classes auxiliares. Contudo, o seu código tem de ser completamente apresentado.

1. (4) Realize o método estático

   ```
   public static int findLastIndex(int[] v, int value);
   ```

   que retorna o índice da última ocorrência de `value` no *array* ordenado (de forma crescente) `v`; ou -1 caso `value` não esteja presente em `v`. O custo do algoritmo implementado deve ser logarítmico.

2. (4) Realize o método estático

   ```
   public static void sortMinHeap(int[] v);
   ```

   que ordena *de forma crescente* o *array* `v`, assumindo que inicialmente `v` representa um *min-heap*.

3. (4) Considere a classe `HashTable<E>` com a implementação de tabelas de dispersão com encadeamento externo e dimensão N, onde N é uma constante *igual para todas as instâncias da classe*. Esta classe usa como função de dispersão o método `hashCode` definido em `Object`.

   Realize o método de instância

   ```
   public HashTable<E> unionWith(HashTable<E> otherTable)
   ```

   que retorna uma nova tabela de dispersão com a união dos elementos presentes em `this` e em `otherTable`.

4. (4) Realize o método estático

   ```
   public static Integer[] firstKElements(Node root, int k)
   ```

   que retorna um *array* com os primeiros k inteiros presentes na árvore binária de pesquisa com raiz `root`. Assuma que cada objecto do tipo `Node` tem 3 campos: um `value` do tipo `Integer` e duas referências, `left` e `right`, para os descendentes respectivos.

ISEL, 2 de Abril de 2009.