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, segunda época (2h30m)

I (4 v)

1. (1,5) Considere os algoritmos de ordenação estudados.

   1.1. Indique um algoritmo cujo custo, quaisquer que sejam os valores presentes na sequência a ordenar, pertence a $O(n^2)$ mas não pertence a $\Theta(n^2)$.

   1.2. Indique um algoritmo cujo custo, quaisquer que sejam os valores presentes na sequência a ordenar, pertence a $O(n \log n)$ e pertence a $\Theta(n \log n)$.

   Justifique as suas escolhas.

2. (1,5) Considere uma aplicação que lê do *standard input* uma sequência de palavras e escreve no *standard output* as palavras que ocorrem mais do que uma vez. Descreva um algoritmo para esta aplicação. Notas: as palavras repetidas não estão necessariamente seguidas; uma palavra deve aparecer no *standard output* no máximo uma vez.

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):

   $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

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. (3) Seja `v` um *array* de inteiros constituído pela concatenação de duas sequências de inteiros não vazias, em que a primeira é estritamente crescente e a segunda é estritamente decrescente. A dimensão destas sequências não é conhecida.

   Realize um método que, recebendo este *array*, retorna o valor do maior elemento nele presente. O custo deste método deve pertencer $O(\log n)$, em que n é a dimensão do *array*.

2. (3,5) Realize o método estático

   ```
   public static int intersection(int[] dest, int[] h1, int h1Len, int[] h2, int h2Len);
   ```

   que preenche o *array* `dest` com o *min-heap* resultante da intersecção dos elementos presentes nos *min-heaps* representados pelos *arrays* `h1` e `h2`. A dimensão destes *min-heaps* é, respectivamente, `h1Len` e `h2Len`. O método retorna a dimensão do *min-heap* resultante. Caso a dimensão de `dest` não seja suficiente para armazenar o `min-heap` com a intersecção, o método deve retornar -1.

   Nota: os *min-heaps* representados por `h1` e por `h2` não têm de ser preservados.

3. (3,5) Considere a classe `HashTable<E>` com a implementação de tabelas de dispersão com encadeamento externo. Realize o método de instância

   ```
   public boolean hasDuplicates()
   ```

   que retorna `true` se e só se existirem elementos duplicados na tabela.

4. (3) Realize o método estático

   ```
   public static boolean contains(Node root, Integer l, Integer r)
   ```

   que retorna `true` se e só se a árvore binária de pesquisa com raíz `root` contém algum elemento no intervalo $[l, r]$. 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.

5. (3) Realize o método estático

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

   que retorna um array contendo os k menores elementos presentes na árvore binária de pesquisa com raíz `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 Fevereiro de 2009.