Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Algoritmos e Estruturas de Dados

Semestre de Verão 2007/08

Teste final, segunda época (2h30m)

I (6 v)

1. (2) Os *amontoados binários* (*heaps*) e as *árvores binárias de pesquisa* são casos particulares de árvores binárias. Contudo são usados para finalidades diferentes. Indique quais são estas finalidades.

2. (1) Considere o algoritmo de inserção em $B-Trees$ apresentado nas aulas. A inserção dum elemento numa $B-Tree$, constituída por apenas um nó (página) totalmente cheio, resulta numa $B-Tree$ com quantos nós?

3. (1,5) Quais as vantagens e desvantagens relativas entre os algoritmos de ordenação *quick sort* e *merge sort*?

4. (1,5) Considere o amontoado binário definido pelo *array* $[10; 5; 9; 3; 4; 1; 8; 2; 1]$. Apresente a sequência de alterações realizadas sobre o amontoado durante a operação de remoção do maior elemento.

II (14 v)

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

1. (3,5) Realize o método estático

   ```
   public static int partitionPoint(int[] v, int len, int val)
   ```

   que dado o *array segmentado* `v`, com dimensão `len`, retorna o índice do primeiro elemento maior ou igual que `val`. O *array* diz-se *segmentado* se todos os elementos menores que `val` estiverem antes que os elementos maiores ou iguais que `val`. Caso todos os elementos sejam menores que `val`, deve ser retornado o índice `len`.

   Valorizam-se soluções com custo assimptótico $O(\log(n))$, em que n é a dimensão do *array*.

   Nota: o *array* não está necessariamente ordenado.

2. (3,5) Realize o método estático

   ```
   public static <E extends Comparable<E>> Node<E> partition(Node<E> head, E val)}
   ```

   que segmenta a lista simplesmente ligada sem sentinela referenciada por `head`, de acordo com o seguinte critério: todos os valores menores que `val` devem ficar antes dos valores maiores ou iguais que `val`. Os nós da lista são representados pela classe `Node<E>`, que possui dois campos públicos: `value`, com o elemento presente no nó; e `next` com a referência para o próximo nó

   O método retorna a referência para o primeiro nó da lista resultante.

3. (3,5) Considere a seguinte classe:

   ```
   public class Pair<E1,E2>{
      public E1 first;
      public E2 second;
      public Pair(E1 first, E2 second){
         this.first = first;
         this.second = second;
      }
   }
   ```

   Realize o método estático

   ```
   public static <E> Pair<E,Integer>[] histogram(E[] v)
   ```

   que dada a sequência representada pelo *array não ordenado* de elementos E, retorna um *array* de pares com o histograma da sequência. O algoritmo implementado deve usar uma tabela de dispersão, de dimensão constante, para o cálculo do histograma.

4. (3,5) Realize o método estático

   ```
   public static <E extends Comparable<E>> Node<E> getListOfLeafs(Node<E> t)}
   ```

   que dada a árvore binária de pesquisa referenciada por `t`, retorna uma lista simplesmente ligada com os valores presentes nos nós folha de `t`. A lista retornada deve estar ordenada e a árvore deve permanecer inalterada.

   Assuma que cada nó da árvore tem três campos: um `E val` e duas referências, `left` e `right`, para os descendentes respectivos. A lista simplesmente ligada usa o campo `right` da classe `Node<E>` para referir o próximo nó da lista.

ISEL, 15 de Julho de 2008.