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, primeira época (2h30m)

I (6 v)

1. (2) Os *amontoados binários* (*heaps*) e as *árvores binárias de pesquisa* são casos particulares de árvores binárias, contudo diferem nas formas de representação normalmente usada. Descreva estas formas de representação e justifique as diferenças.

2. (1) Considere o algoritmo de inserção em $B - Trees$ apresentado nas aulas. Em que situações a inserção dum novo elemento resulta no aumento da altura da árvore?

3. (1,5) As tabelas de dispersão com encadeamento externo usam normalmente listas simplesmente ligadas para armazenar os elementos com o mesmo valor de *hash*. Justifique a escolha desta estrutura de dados em detrimento de outras, tais como: *arrays* ordenados ou árvores binárias de pesquisa.

4. (1,5) Compare o desempenho qualitativo dos algoritmos de ordenação *selection sort* e *insertion sort* em *arrays quase* ordenados.

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. (1,5) Realize o método estático

   ```
   public static int heapHeight(int[] v, int len)
   ```

   que retorna a altura do amontoado binário de inteiros representado pelo *array* `v` com dimensão `len`. Na implementação deste método não pode usar métodos da classe `Math`, pertencente à biblioteca normalizada da linguagem Java.

2. (3,5) Realize o método estático

   ```
   public static int kGreatest(int[] v, int len, int k)
   ```

   que devolve o k-ésimo maior elemento do amontoado binário de inteiros representado pelo *array* `v` com dimensão `len`. O *array* `v` não pode ser alterado.

   Usando a notação O, diga qual o custo da implementação realizada.

3. (2,0) Realize o método estático

   ```
   public static <E> Node<E> rotateLeft(Node<E> head, int count)
   ```

   que *roda* a lista simplesmente ligada sem sentinela e *não circular*, referida por `head`, de `count` nós para a esquerda. Rodar `count` nós para a esquerda significa que os primeiros `count` nós são colocados no final da lista, mantendo a ordem relativa. Por exemplo, a rotação de 3 elementos da lista $[0; 1; 2; 3; 4; 5; 6; 7; 8; 9]$ resulta na lista $[3; 4; 5; 6; 7; 8; 9; 0; 1; 2]$.

   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 (quando existe). Assuma que `count` é menor do que a dimensão da lista.

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

4. (3,5) Acrescente, à tabela de dispersão com encadeamento externo realizada nas aulas, o método de instância

   ```
   public void reduceSize(int newSize)
   ```

   que reduz a dimensão da tabela para a dimensão `newSize` (estritamente menor que a dimensão actual). Os nós da lista de encadeamento externo são representados pela classe `Node<E>`, que possui três campos públicos: `value`, com o elemento presente no nó; `hashCode` com o valor de *hash* do elemento (ainda sem ter em conta a dimensão da tabela) e `next` com a referência para o próximo nó (quando existe).

5. (3,5) Realize o método estático

   ```
   public static <E> int countNodesWithHeight(Node<E> t, int h)
   ```

   que, dada a árvore binária referenciada por `t`, retorna o número de nós de `t` com altura `h`. Note que a altura do nó raiz é 0.

   Assuma que cada nó da árvore tem três campos: um `value`, do tipo `E`, e duas referências, `left` e `right`, para os descendentes respectivos.

ISEL, 25 de Junho de 2008.