Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Algoritmos e Estruturas de Dados

Semestre de Inverno 2009/10

Primeira série de exercícios

Observações:

- Data de entrega: **16 de Novembro de 2009**.

1. Realize a classe `BinaryHeaps` contendo os seguintes métodos estáticos.

   1.1. `public static void printByDepth(int[] v, int count);`

   Este método apresenta na consola os inteiros presentes no amontoado binário representado pelos primeiros `count` inteiros do *array* `v`. Esta apresentação deve ter a seguinte organização:

   - Apresentar na mesma linha da consola os elementos com a mesma profundidade (distância à raíz).
   - Apresentar primeiro a linha com a raíz e por último a linha com os elementos de maior profundidade.
   - Em cada linha, os elementos são apresentados da esquerda para a direita.

   1.2. `public int countMaxInMaxHeap(int[] v, int count)`

   Este método, dado o *max-heap* representado pelos primeiros `count` inteiros do *array* `v`, retorna o número de ocorrências do maior elemento presente nesse *max-heap*.

   1.3. `public static int largestSubArrayThatIsAMaxHeap(int[] v);`

   Este método retorna a dimensão do maior *sub-array* de v, com início no índice 0, que representa um *max-heap*.

2. Realize a classe `AedLinkedList<E>` com a mesma funcionalidade da classe `java.util.LinkedList<E>`. A classe realizada pode extender a classe `java.util.AbstractCollection<E>`.

3. Realize a classe `Iterables` contendo os seguintes métodos estáticos:

   3.1. `public static <E> Iterable<E> concat(Iterable<E> iter1, Iterable<E> iter2)`

   que retorna um objecto com a interface `Iterable<E>`, representando a concatenação das sequências `iter1` e `iter2`. Não é necessário implementar o método `remove`, pertencente à interface genérica `Iterator<E>`.

   A implementação deste método deve minimizar o espaço ocupado pelo iterador.

   3.2. `public static <E> Iterable<E> takeWhile(Iterable<E> iter, Predicate<T> pred){`

   que retorna um objecto com a interface `Iterable<E>`, representando o maior prefixo da sequência `iter` em que todos os elementos satisfazem o predicado `pred`. Não é necessário implementar o método `remove`, pertencente à interface genérica `Iterator<E>`

   A implementação deste método deve minimizar o espaço ocupado pelo iterador.

   Use a seguinte definição da interface `pred`.

```
public interface Predicate<E> {
    boolean eval(E e);
}
```

4. Considere a representação de redes rodoviárias através de *grafos não dirigidos com pesos* (ver capítulo 22 e apêndice B.4 do livro de referência):

   - Cada intersecção entre ligações da rede corresponde a um nó do grafo.
   - Cada ligação na rede corresponde a um arco do grafo. Este arco tem associado a distância da ligação.

   Considere também os ficheiros com redes rodoviárias presentes em

   `http://www.dis.uniroma1.it/~challenge9/download.shtml`

   Os formatos destes ficheiros estão descritos em

   `http://www.dis.uniroma1.it/~challenge9/format.shtml`

   Realize um programa que, dado um ficheiro com a definição dum grafo, produza outro ficheiro com a definição do mesmo grafo mas com os arcos ordenados por ordem crescente do seu custo.

   Assuma que o conteúdo do grafo excede a dimensão de memória disponível na máquina virtual. Utilize o algoritmo de *ordenação externa* descrito na secção 11.4 do livro R. Sedgewick, "Algorithms in Java", 3ª edição, Addison-Wesley, 2003.

ISEL, 19 de Outubro de 2009.