Estruturas de Informação

Época Normal

28 de Janeiro 2016

Duração 2 horas

Resolva cada exercício em folhas separadas

1. A **média móvel simples** (**MMS**) calcula a média através dos valores mais recentes numa série de dados. Na MMS de um conjunto, o k-ésimo elemento é calculado através da média dos P elementos consecutivos do conjunto, a partir da posição k-P+1, sendo P o **período** da média móvel. A MMS dos elementos iniciais do conjunto (antes de haver suficientes para uma média) é zero. Matematicamente, dado um conjunto C, os elementos da MMS são calculados por:

$$MMS_k = \sum_{i=k-P+1}^{k} C_i$$

Como exemplo considere a média móvel do conjunto C = { 2, 4, 3, 7, 8, 10 } com período P = 3. Dado que P é 3, a média móvel dos dois primeiros elementos são zero, porque não há suficientes para uma média. O terceiro elemento da média móvel é (2+4+3)/3 = 3 e o quarto será (4+3+7)/3 = 4. A MMS do conjunto C a devolver será { 0, 0, 3, 4, 6, 8 }.

Elabore um método que recebe um conjunto de números inteiros representado numa lista ligada e um período e devolve a MMS do conjunto.

```
DoublyLinkedList<Integer> calcMMS(DoublyLinkedList<Integer> serie, Integer period)
```

Nota: Será valorizada uma resolução com complexidade linear.

2. De um modo geral, medidas de centralidade são usadas para calcular a importância dos vértices no grafo. Numa rede social, por exemplo, elas são usadas para calcular o grau de influência dos seus membros. Dado um grafo G e um vértice V de G a **centralidade do vértice** V é dada pelo comprimento do maior caminho de entre todos os caminhos mais curtos que partem do vértice V. O **raio de um grafo** G é definido pelo menor valor de centralidade dos vértices do grafo. Os vértices que apresentam essa menor centralidade são os **centros do grafo**. Utilizando a representação **map de adjacências** elabore uma **implementação Java eficiente** de um método que para um grafo determina os seus centros e devolve o seu raio:

```
Double centersGraph (Graph<V, Double> g, ArrayList<V> lstcenters)
```

Resolva cada exercício em folhas separadas

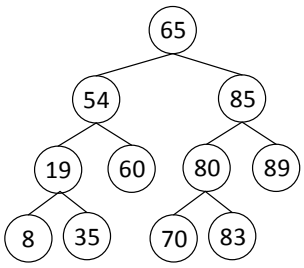
3. Considere o seguinte método implementado na **classe BST**:

```
protected Node<E> mistério (E element, Node<E> node){
  if (node == null)
      return null;
  if (element.compareTo(node.getElement())==0)
      return node;
  if (element.compareTo(node.getElement())<0)
      return misterio (element, node.getLeft());
  return misterio (element, node.getRight());
}
```

a) Explique o que faz o método acima apresentado.

b) Analise o método quanto à sua complexidade temporal. Justifique.

4. Adicione à classe TREE<E> um método que devolva uma lista com todos os nós que se encontram na base da árvore, i.e., no último nível da árvore. Para a árvore abaixo o método deve devolver a lista {8, 35, 70, 83}



5. Usando os métodos da classe HeapPriorityQueue<K,V> escreva um método **o mais eficiente possível** que devolva numa lista os N menores elementos de uma sequência.