isep Instituto Superior de Engenharia do Porto

Estruturas de Informação
Época de Recurso

6 de Fevereiro 2018

Departamento de Engenharia Informática

Duração 2 horas

Resolva cada exercício em folhas separadas

3.5 p^tos

1. Elabore um método que recebe um **mapa de sinónimos**, onde a chave é uma palavra e o valor é uma lista de sinónimos dessa palavra. O objetivo é devolver uma lista com as palavras que aparecem como sinónimos mais do que **i vezes**. A lista de retorno não deve ter sinónimos repetidos. Considere a seguinte assinatura:

```
List<String> maisSinonimos(Map<String, List<String>> mapSyn, Integer i)
```

Nota: Será valorizada uma resolução o mais eficiente possível

3.5 p^tos

2. Considere um sistema de gestão de tarefas com o seguinte método para submissão de tarefas de acordo com a sua prioridade.

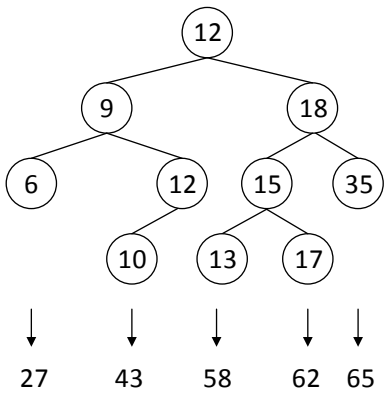
```
public static<K,V> HeapPriorityQueue<K,V> submitTask(K[] priority, V[] tasks){

  HeapPriorityQueue<K,V> heap = new HeapPriorityQueue<K,V>() ;

  for (int i=0; i < priority.length; i++)
      heap.insert(priority[i], tasks[i]);

  return heap;
}
```

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

b) Proponha uma solução mais eficiente. Justifique.

5 p^tos

3. Adicione à classe TREE<E> um método que devolve num ArrayList a soma dos conteúdos dos nós de todos os caminhos da árvore desde o nó raiz até às folhas. Por exemplo:



Resolva cada exercício em folhas separadas

5 ptos

4. O algoritmo de Visita Aleatória (*Random Walk*) é utilizado em áreas tão diversas como genética, finanças, física ou matemática, sendo também a base de algoritmos utilizados na *web* e redes sociais (e.g. pelo Google para o PageRank ou pelo Twitter para as sugestões de pessoas a seguir).
   Uma variante deste algoritmo é o de **Visita Aleatória com Reinício** (*Random Walk with Restart*), o qual realiza uma visita a partir de um determinado vértice origem. O vértice seguinte a visitar é escolhido aleatoriamente entre os vértices adjacentes do vértice atual. Em cada vértice visitado o algoritmo gera um número aleatório, voltando ao nó de origem se o valor gerado for inferior ao valor definido como probabilidade de reinício ou, se o vértice atual não tiver adjacentes. O algoritmo termina ao fim de um número máximo de reinicializações.
   Usando a representação **map de adjacência** implemente um método que, dado um grafo, um vértice de origem, uma probabilidade de reinício (prbRStart) e o número máximo de reinicializações (maxRep), devolva um map com os vértices visitados e o respetivo número de visitas. O método a desenvolver deve obedecer à interface:

```
Map<V,Integer> RandWalkWithRestart(Graph<V,E> g, V source,
                                   Double prbRStart, Integer maxRep)
```

   Considere os seguintes métodos para gerar um número aleatório:

```
public int randomInteger(int low, int high)
public double randomDouble()
```

3 ptos

5. Implemente na classe HeapPriorityQueue<K,V> um método que remova de uma heap todos os elementos com um determinado valor específico. O método deve retornar a heap resultante. Considere a seguinte assinatura:

```
public <K,V extends Comparable<V>> HeapPriorityQueue<K,V>
                          remValues(HeapPriorityQueue<K,V> heap, V value)
```