Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Algoritmos e Estruturas de Dados
Semestre de Verão 2016/2017
Segunda série de exercícios

Observações:

- Data de entrega: **31 de Maio de 2017**.
- Não é permitida a utilização de algoritmos e estruturas de dados já existentes na biblioteca base da plataforma Java.

1. Exercícios

1. Realize o método estático da classe **`Utils`**,

   ```
   public static boolean verifyPairing(String str)
   ```

   que recebe a `string str` e verifica se os parêntesis curvos, parêntesis retos e chavetas (`'(', '[', '{'`), presentes em `str`, se encontram emparelhados e aninhados corretamente. O método retorna `true` em caso afirmativo e `false` caso contrário. Exemplos:

   - `"...(...{... )...}...[...]..."` retorna `false`;
   - `"...(...)...{...[...]...}...[...]..." retorna true.`

   Indique, justificando, a complexidade do algoritmo.

2. Realize na classe **`ListUtils`**

   2.1. O método,

   ```
   public static <E>
   void removeAfterIntersectionPoint(Node<E> list1,Node<E> list2,Comparator<E> cmp)
   ```

   que, dadas duas listas duplamente ligadas, circulares e com sentinela, referenciadas por `list1` e `list2`, remove de `list1` os nós a partir do *nó de interseção* de ambas, caso exista. O *nó de interseção* de duas listas é o primeiro nó após o qual ambas as listas partilham a mesma sequência de valores segundo o comparador `cmp`.

   Por exemplo, no caso das listas serem: `list1 = {3,5,2,7,4}` e `list2 = {9,3,10,8,2,7,4}`, existe o nó de interseção entre as duas listas que é o que contém o valor 2. Neste caso, após a execução do método a `list1` ficará `list1 = {3,5}`.

   2.2. O método,

   ```
   public static <E> Node<E> merge(Node<E>[] lists, Comparator<E> cmp)
   ```

   que dado um `array` de listas duplamente ligadas, não circulares e sem sentinela, ordenadas pelo comparador `cmp`, retorna uma lista duplamente ligada, circular e com sentinela, resultante da junção ordenada segundo o comparador, das listas presentes em `lists`. A lista resultante deve reutilizar os nós presentes em `lists`, ficando vazias todas as listas.

   2.3. O método,

   ```
   public static <E> Node<E> interleaved(Node<Node<E>> list)
   ```

   que retorna uma lista duplamente ligada circular e com sentinela, resultante da junção, de modo intercalado, das listas duplamente ligadas circulares e com sentinelas, presentes em **`list`**. Por exemplo, se estiverem presentes em **`list`** as listas `[1; 2; 3; 4]`, `[4; 5; 6]` e `[7; 8; 9; 10]`, a lista resultante será `[1; 4; 7; 2; 5; 8; 3; 6; 9; 4; 10]`. A lista **`list`** deverá ficar vazia. Note que **`list`** também é uma lista duplamente ligada circular e com sentinela, devendo os nós das suas listas, serem reaproveitados na lista resultante.

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

   3.1. O método estático,

   ```
   public static Iterable<Integer>
   getValuesBetween(Iterable<Integer> src, int l,int r)
   ```

   que retorna um iterável com os elementos presentes na sequência `src` que pertençam ao intervalo fechado `[l,r]`. A sequência `src` está ordenada de modo crescente segundo a ordem natural dos inteiros. A implementação deste método deve minimizar o espaço ocupado pelo iterável. O iterador associado ao iterável retornado não suporta o método `remove`.

   3.2. O método estático,

   ```
   public static Iterable<String>
   getPhrasesStart(Iterable<Iterable<String>> phrases, String prefix)
   ```

   que retorna um iterável com as frases que ocorram na sequência de frases equivalente à concatenação das sequências presentes em **`phrases`**, e que contenham **`prefix`** como primeira palavra. A implementação deste método deve minimizar o espaço ocupado pelo iterável. O iterador associado ao iterável retornado não suporta o método **`remove`**. Por exemplo, se **`phrases`** contiver as seguintes sequências de palavras: `[["O","rato","roeu","a","rolha","da","garrafa","do","rei","da","Rússia"], ["Fui", "ao", "mar", "colher", "cordões", "vim", "do", "mar", "cordões", "colhi"], ["O","original","nunca","se","desoriginou","nem","nunca","se","desoriginalizará"], ["Três","pratos","de","trigo","para","três","tristes","tigres"]]` e se **`prefix`**=`"O"`, então o objeto retornado deve representar a sequência: `["O rato roeu a rolha da garrafa do rei da Rússia", "O original nunca se desoriginou nem numca se desoriginalizará"]`.

2. Problema: utilização de Mapas em semelhança de documentos

Pretende-se desenvolver uma aplicação que permita inferir a semelhança entre dois documentos. Como medida de semelhança, a aplicação compara se as palavras que ocorrem em ambos documentos são ou não as mesmas e se têm ou não o mesmo número de ocorrências.
Esta aplicação retorna um inteiro positivo, que exprime o grau de semelhança. O valor de retorno zero expressa o maior grau de semelhança e neste caso significa que ambos os documentos contêm as mesmas palavras e com o mesmo número de ocorrências. Por cada palavra distinta ou por cada palavra cuja ocorrência não é a mesma em ambos os documentos, o valor a retornar é incrementado de um. Deste modo, a aplicação **DocumentsSimilarity** tem as seguintes características:

- recebe como parâmetro dois ficheiros de texto;
- retorna um inteiro que expressa a semelhança entre os dois ficheiros recebidos por parâmetro.

No contexto do desenvolvimento da aplicação **DocumentsSimilarity**, será necessário implementar um mapa **`AEDMap<K,V>`**, em que mapeia chaves a valores. Este tipo de dados deverá ter as seguintes operações:

- **`public V put(K key, V value) -`** Associa o valor **`value`** à chave **`key`** no mapa. Se o mapa já contiver um mapeamento para a chave **`key`**, substitui-o pelo novo valor **`value`**. Esta operação deverá retornar o valor que se encontrava associado à chave **`key`**, ou **`null`** caso não existisse mapeamento para a chave.
- **`public V get(K key)-`** Retorna o valor ao qual a chave **`key`** está mapeada, ou **`null`**, se este mapa não contém mapeamento para esta chave.
- **`public V remove(K key)`** - Remove o mapeamento para uma chave **`key`** deste mapa, se este mapeamento existir. Retorna o valor que estava associado à chave **`key`**, ou **`null`** se não existir mapeamento para esta chave.
- **`public int size()`** - Retorna o número de mapamentos chave-valor presentes neste mapa.
- **`public boolean isEmpty()`** - Retorna **`true+`** se este mapa não contém mapeamentos chave-valor.
- **`Public Collection<V> values()+`** - Retorna uma vista da coleção dos valores contidos neste mapa.
- **`public Set<K> keySet() -`** Retorna uma vista do conjunto das chaves contidas neste mapa.
- **`public Set<Map.Entry<K,V>> entrySet()-`** Retorna uma vista do conjunto dos pares chaves-valor contidos neste mapa. Um mapeamento chave-valor é especificado através do tipo **`java.util.Map.Entry<K,V>`**, através das seguintes operações:
  - **`K getKey() -`** Retorna a chave correspondente a esta entrada.
  - **`V getValue() -`** Retorna o valor correspondente a esta entrada.

Observação: para a implementação deste tipo de dados, poderá utilizar interfaces definidas em **`java.util`** e também os seguintes tipos definidos em **`java.util: java.util.AbstractSet`** e **`java.util.AbstractCollection.`**

Funcionalidades a implementar

As funcionalidades a implementar são as seguintes

- A aplicação **documentsSimilarity**;
- Uma implementação do tipo de dados **AEDMap,** tendo em conta que a sua utilização será na aplicação que infere a semelhança de documentos.

Parâmetros de Execução da aplicação DocumentsSimilarity
Para iniciar a execução da aplicação **DocumentsSimilarity**, terá de se executar

```
java documentsSimilarity document1.txt document2.txt
```

durante a sua execução, a aplicação deverá processar os seguintes comandos:

- `allWords`, que lista todas as palavras que ocorram em pelo menos um dos ficheiros de *input*;
- `wordsWithTheSameOccurrence,` que lista as palavras que estejam presentes em ambos os ficheiros de *input* e com o mesmo número de ocorrências;
- `similarity,` que retorna o grau de semelhança entre os dois ficheiros;
- `exit,` que termina a aplicação.

Avaliação Experimental

Realize uma avaliação experimental da aplicação que infere a semelhança entre dois documentos. Apresente os resultados graficamente, utilizando uma escala adequada.