---

1. **Mergesort**

   Nós nos interessamos aqui à ordenação de listas de inteiros do tipo `LinkedList<Integer>` pelo método *mergesort*.

   Um arquivo `Ex1.java` encontra-se no SIGAA. Ele contem duas classes.

   A classe `Mergesort` contem os métodos que você deve completar.

   A classe `Ex1` contem o código para a realização dos testes. Esta classe não deve ser modificada. É o método `main` desta classe que será necessário para executar e testar suas respostas.

   (a) Escreva o código do método `split` que recebe como argumentos três listas `l`, `l1` e `l2` e partilha os elementos de `l` entre as listas `l1` e `l2`. Mais precisamente, fazemos a hipótese que inicialmente as listas `l1` e `l2` estão vazias.

   Ao final, o método `split` deve assegurar que:

   - a lista `l` não foi modificada.
   - a união dos elementos de `l1` e `l2` é igual ao conjunto de elementos de `l`.
   - os tamanhos de `l1` e `l2` são iguais ou com diferença de uma unidade apenas.

   (b) Escreva o código do método `merge` que recebe como parâmetros duas listas `l1` e `l2`, supostamente ordenadas em ordem crescente, e retorna uma lista ordenada contendo o conjunto de elementos de `l1` e `l2`. As listas `l1` e `l2` podem ser modificadas.

   (c) Por fim, escreva o código do método `mergesort`.

   Faça testes com a classe `Ex1` fornecida.

2. **A bandeira holandesa de Dijkstra**

   Vamos agora ordenar um vetor de inteiros. Neste exercício, consideramos o caso particular de um vetor cujos elementos assumem apenas três valores possíveis. Este é um problema histórico elaborado por Dijkstra sob o nome de *problema da bandeira nacional holandesa*, os três valores possíveis sendo as cores azul, branco e vermelho. Aqui, considere que os três valores poossíveis são 0, 1 e 2. Desta forma, será fácil ordenar um tal vetor bastando para isso contar o número de ocorrências dos três valores: 0, 1 e 2. Entretanto, nós vamos impor aqui que o método deve usar *apenas testes e trocas*, i.e., não são permitidas comparações entre dois elementos do vetor.

   Um arquivo `Ex2.java` é fornecido. Ele contem duas classes. A classe `DutchFlag` contem dois métodos que você deve completar. A classe `Ex2` contem o código para efetuar os testes, que não pode ser alterado. É o método `main` desta classe que deverá ser executado para testar suas respostas.

   (a) Escreva o código do método `swap` que recebe como parâmetros um vetor `a`, dois índices `i` e `j`, e troca o conteúdo das células `i` e `j` de `a`.

(b) Para resolver o problema da bandeira holandesa, iremos proceder da seguinte maneira. Divida o vetor `a` em quatro zonas por meio dos índices `b`, `i` e `r` da seguinte maneira:

```
 0          b          i          r           n
+----------+----------+----------+-----------+
|    0     |    1     |     ?    |      2    |
+----------+----------+----------+-----------+
```

O intervalo `[0,b[` só contem valores 0; o intervalo `[b,i[` só contem valores 1; e o intervalo `[r,n[` só contem valores 2. O intervalo `[i,r[` corresponde a parte do vetor ainda não tratada. A notação `[x,y[` designa um intervalo fechado à esquerda e aberto à direita. Note que cada um destes intervalos pode estar vazio.

Inicialmente, temos `b=i=0` e `r=n`. Depois efetuamos um loop `while(i<r)` que examina `a[i]` e, segundo o caso, atualiza os índices e/ou faz uma troca por meio do método `swap`.

Faça testes com a classe `Ex2` fornecida.

3. **Quicksort**

Nós nos interessamos aqui à ordenação de um vetor de inteiros através do método *quicksort*.

Um arquivo `Ex3.java` é fornecido. Ele contem duas classes. A classe `Quicksort` contem os três métodos que você deve completar. A classe `Ex3` contem o código para efetuar os testes, que não pode ser alterado. É o método `main` desta classe que deverá ser executado para testar suas respostas.

(a) Escreva o código do método `swap` que recebe como parâmetros um vetor `a`, dois índices `i` e `j`, e troca o conteúdo das células `i` e `j` de `a`.

(b) Escreva o código do método `partition` que recebe como parâmetros um vetor `a`, dois índices `l` e `r` tal que `0 <= l < r < tamanho de a`, que rearranja os elementos da subsequência `a[l..r]` e enfim retorna o índice `m` neste intervalo tal que tenhamos a seguinte situação.

```
l           m           r
+----------+-+----------+
|   < v    |v|   >= v   |
+----------+-+----------+
```

ou seja, unicamente os valores inferiores a `a[m]` estão à esquerda de `m` e unicamente os valores superiores ou iguais a `a[m]` estão à direita de `m`.

(c) Escreva o código do método `quickrec` que recebe como parâmetros um vetor `a`, dois índices `r` e `l` tais que `0 <= l` e `r < a.length`, e ordena o segmento `a[l..r]` do vetor pelo método *quicksort*.

(d) Implemente o método `quicksort` que ordena o vetor inteiro.

Faça testes com a classe `Ex3` fornecida.

*Este trabalho prático é de autoria de Jean-Christophe Filli âtre (Poly, France)