# INF01124 - Classificação e Pesquisa de Dados - Exercício 2
# Professor João Comba

## 1 Quicksort

Em aula vimos o algoritmo de Quicksort. Nesta tarefa é solicitada a implementação de quatro versões deste algoritmo, usando duas estratégias para escolha de particionador, e duas estratégias de particionamento.

As estratégias para escolha do particionador que devem ser implementadas são:

1. Escolha 1 (Mediana de 3): Particionador deve ser a mediana entre o primeiro, último e elemento na posição média de cada sub-vetor (média entre o índice do primeiro e último elemento, arrendodada para baixo);

2. Escolha 2 (Aleatório): Particionador será um elemento aleatório do sub-vetor.

A estratégia de escolha deve primeiro definir qual será o particionador, e após feita esta escolha, colocar o particionador na primeira posição do sub-vetor, trocando com o elemento atualmente na primeira posição. Após esta troca, implemente o algoritmo de Quicksort como visto em aula usando o primeiro elemento como particionador.

Duas estratégias de particionamento devem ser implementadas, conforme vistas em aula:

1. Particionamento 1 (Lomuto)

2. Particionamento 2 (Hoare)

Para todas as quatro combinações (escolha x particionamento), teste cada uma das versões do algoritmo com o arquivo *entrada-quicksort.txt*, contendo vetores aleatórios de tamanho 100, 1000, 10000, 100000 e 1000000 em anexo. O programa deve ler os vetores de um arquivo de entrada e gravar resultados em um arquivo de saída. A entrada é o arquivo *entrada-quicksort.txt*, contendo vários vetores a serem ordenados. Cada vetor é descrito em 1 linha. O primeiro número da linha descreve o tamanho *n* do vetor, seguido por *n* números do vetor. Um exemplo que ilustra o formato do arquivo de entrada segue:

```
16 16 14 12 1 8 4 9 6 15 13 11 2 7 3 10 5
16 3 10 5 16 14 12 1 8 4 9 6 15 13 11 2 7
```

Para cada uma das 4 combinações de escolha e particionamento, execute o código com o arquivo de entrada, e armazene o número de trocas (swaps) entre elementos do vetor, o número de chamadas recursivas e o tempo de execução do algoritmo em milisegundos. Estes resultados devem ser impressos no exato formato listado abaixo:

```
tamanho,escolha-particionador,particionamento,trocas,recursoes,tempo
100,aleatorio,lomuto,999,999,9.999
100,aleatorio,hoare,999,999,9.999
100,mediana3,lomuto,999,999,9.999
100,mediana3,hoare,999,999,9.999
.
.
.
1000000,aleatorio,lomuto,999,999,9.999
1000000,aleatorio,hoare,999,999,9.999
1000000,mediana3,lomuto,999,999,9.999
1000000,mediana3,hoare,999,999,9.999
```

Salve esse arquivo com o nome resultados.txt.

## 2 Opcional: Desafio Bônus - Problema Bubbles and Buckets

Uma das formas mais interessantes de aprender a utilidade dos temas estudados é resolver desafios propostos em problemas que aparecem em maratonas de programação. Resolva o problema BEE 1259 - Even and Odd

Este é um problema de maratona de programação. O problema será considerado completo se a solução for aceita no site da BeeCrowd. O comprovante da aceitação e o código deve ser entregue junto com a solução do laboratório. A entrega correta valerá um adicional de 25% pontos.

## 3 Entrega

A solução deve ser enviada pelo Moodle dentro de um arquivo .zip, contendo os seguintes arquivos:

- **integrantes.txt**: coloque o nome dos integrantes do grupo (até 2 pessoas) , com um nome por linha
- **resultados.txt**: no exato formato listado acima
- código fonte correspondente a solução dos testes do Quicksort
- opcional: código e comprovante de aceite do problema desafio