# DCA0214.1 - LABORATÓRIO DE ESTRUTURAS DE DADOS
## Aula 3: Busca e ordenação em listas sequenciais (vetores)

Prof. Felipe Fernandes

22 Março de 2019

1. Implemente os seguintes algoritmos de ordenação:
   (a) InsertionSort
   (b) SelectionSort
   (c) BubbleSort
   (d) MergeSort
   (e) QuickSort

2. Escreva uma função que recebe um número inteiro $n \geq 0$, um vetor de números inteiros distintos $v[0...n-1]$ e um número inteiro $x$ e devolve $k$ no intervalo $[0, n-1]$ tal que $v[k] == x$. Se tal $k$ não existe, devolve $-1$. Faça isso para os itens abaixo.
   (a) Busca sequencial simples
   (b) Busca sequencial em lista ordenada
   (c) Busca binária iterativa
   (d) Busca binária recursiva

3. Refaça as funções de busca sequencial e busca binária assumindo que o vetor possui chaves que podem aparecer repetidas. Neste caso, a função deve retornar um outro vetor (*posicoes*) no qual constam todas as posições onde a chave foi encontrada. A função também deve retornar a quantidade de posições (*quant*) em que a chave procurada se repete.

```
void busca(int V[MAX1], int n, int x, int
posicoes[MAX1], int &quant)
```

4. Seja $V[0...n-1]$ um vetor com $n$ valores numéricos. Faça o que se pede:

   (a) Escreva uma função iterativa que encontre o segundo maior elemento de $V$. Seu algoritmo deve ser baseado em comparações.

   (b) Qual a complexidade no melhor e no pior caso?

5. Escreva uma função que recebe dois argumentos: (1) um vetor $V[0...n-1]$ com $n$ inteiros, (2) um inteiro $x$; e determina se existem dois elementos em $V$ cuja soma é exatamente $x$. A complexidade do seu algoritmo deve ser $\Theta(nlogn)$.

6. Uma forma de se obter a raiz quadrada de um número qualquer $n$ seria simulando a busca binária. Assuma que a raiz quadrada de $n$ está entre 0 e $n$ (Se o número for negativo, retorne 0). Para sabermos se um palpite y é a raiz quadrada de $n$, basta testar se $y * y$ é próximo o suficiente de $n$ ou seja, se o módulo da diferença entre eles está dentro de uma tolerância definida ($eps$). Caso contrário, podemos restringir a busca entre 0 e $y$ ou entre $y$ e $n$. Escreva a função abaixo que implemente este algoritmo.

```
double raiz_quadrada(double n, double eps) {
```