# Lista 3 - Algoritmos e Estrutura de Dados II

Profs. Marcos Castilho, André Vignatti e Daniel Oliveira

**Exercício 1.** O algoritmo **BubbleSort** é um algoritmo de ordenação cuja ideia é percorrer o vetor diversas vezes, e a cada passagem fazer "flutuar" para o topo o maior elemento da sequência. Essa flutuação lembra bolhas em um tanque de água, e por isso vem o nome do algoritmo. Seu pseudo-código iterativo é mostrado abaixo:

```
BubbleSort(v, a, b)
Para i ← a até b − 1
    Para j ← a até b − i
        Se v[j] > v[j + 1] então
            Troca(v, j, j + 1)
```

Usando a mesma ideia, escreva uma versão **inteiramente recursiva** do BubbleSort. (Obs: talvez seja necessário usar mais de uma função.)

**Exercício 2.** Mostre como o InsertionSort ordena o vetor "EASYQUESTION".

**Exercício 3.** Como é um vetor $v$ de tamanho $n$ onde o InsertionSort faz o menor número possível de comparações com elementos de $v$?

**Exercício 4.** Qual é o maior e menor número de **trocas** feitas pelo InsertionSort?

**Exercício 5.** Mostre como o SelectionSort ordena o vetor "EASYQUESTION".

**Exercício 6.** Qual é o maior e menor número de **trocas** feitas pelo SelectionSort?

**Exercício 7.** Como é um vetor $v$ de tamanho $n$ onde o SelectionSort faz o menor número de comparações com elementos de $v$?

**Exercício 8.** Explique como modificar o SelectionSort para que o número de trocas entre elementos do vetor no melhor caso seja 0.

**Exercício 9.** Mostre como o MergeSort executa no vetor $v = (3, 41, 52, 26, 38, 57, 9, 49)$.

**Exercício 10.** A seguinte versão do Intercala (Merge) pode ser usada pelo MergeSort?

```
Intercala(A, p, q, r)
Para i ← p até q
    B[i] ← A[i]
Para j ← q + 1 até r
    B[r + q + 1 − j] ← A[j]
i ← p
j ← r
Para k ← p até r
    Se B[i] ≤ B[j] então
        A[k] ← B[i]
        i ← i + 1
    Senão
        A[k] ← B[j]
        j ← j − 1
```

**Exercício 11.** Re-escreva o MergeSort para que este passe a ordenar um vetor em ordem decrescente.

**Exercício 12.** Mostre como o QuickSort executa no vetor "EASYQUESTION"

**Exercício 13.** Mostre como o algoritmo Particiona executa no vetor $v = (13, 19, 9,\ 5, 12, 8,\ 7, 4, 21,\ 2, 6, 11)$.

**Exercício 14.** Qual o valor retornado pelo algoritmo Particiona se todos os elementos do vetor tiverem valores iguais?

**Exercício 15.** Modifique o QuickSort para que ele escolha como pivô o primeiro elemento do vetor (e não o último, como visto em aula).

**Exercício 16.** Como você modificaria o QuickSort para ordenar em ordem não-crescente?

**Exercício 17.** Considere o QuickSort executando em um vetor de tamanho $n$. Qual o maior número de vezes que o maior elemento pode ser movido?

**Exercício 18.** Considere o Quicksort com a seguinte modificação: em toda chamada de Particiona, o pivô é sempre escolhido aleatoriamente (ao invés de ser sempre o elemento da última posição). Intuitivamente falando (sem precisar mostrar contas), essa modificação traria alguma mudança para o pior caso do Quicksort?