Lista de Exercícios 3 **Alg. e Estruturas de Dados I**

ORDENAÇÃO

EXERCÍCIO 45
Faça uma comparação entre todos os métodos de ordenação estudados em aula com relação a estabilidade (preservar ordem lexicográfica), ordem de complexidade levando em consideração as comparações e movimentações.

EXERCÍCIO 46
Dada a sequência de números: 3 4 9 2 5 8 2 1 7 4 6 2 9 8 5 1, ordene-a em ordem crescente segundo os seguintes algoritmos, apresentando a sequência obtida após cada passo do algoritmo:

a) ShellSort
b) MergeSort
c) QuickSort
d) HeapSort

EXERCÍCIO 47
João diz ter desenvolvido um algoritmo que é capaz de ordenar qualquer conjunto de n números reais, fazendo apenas $O(n^{3/2})$ comparações. Você compraria este algoritmo? Justifique.

EXERCÍCIO 48
Dado um conjunto de n inteiros distintos e um inteiro positivo $k \leq n$:

a) Proponha um algoritmo que imprime os k -1 elementos do conjunto (em qualquer ordem) em tempo O(n).
b) Suponha agora que queremos imprimir os k-1 em ordem crescente. É ainda possível fazer isso em tempo O(n) para quaisquer valores $k \leq n$?

EXERCÍCIO 49
Reescreva a função BubbleSort apresentada em aula com sucessivas passagens em direções opostas.

EXERCÍCIO 50
Intercalação de listas encadeadas. Digamos (para efeito deste exercício) que uma LEDsc é uma lista encadeada (sem cabeça) que contém uma sequência crescente de números inteiros. Escreva uma função que intercale duas LEDsc, produzindo assim uma terceira. Sua função não deve alocar novas células na memória, mas reaproveitar as células das duas listas dadas.

EXERCÍCIO 51
Escreva uma versão do algoritmo Mergesort com *cutoff* para vetores pequenos: quando o vetor a ser ordenado tiver menos que M elementos, a ordenação passa a ser feita pelo algoritmo de inserção. O valor de M pode ficar entre 10 e 20. Compare os tempos de execução com o Mergesort apresentado em sala.

> **cutoff**: Esse truque é usado na prática porque o algoritmo de inserção é mais rápido que o Mergesort puro quando o vetor é pequeno. O fenômeno é muito comum: algoritmos sofisticados são tipicamente mais lentos que algoritmos simplórios quando o volume de dados é pequeno.

EXERCÍCIO 52
Uma **ordenação por contagem** de um vetor x de tamanho n é executada da seguinte forma: declare um vetor count e defina count[i] como o número de elementos menores que x[i]. Em seguida, coloque x[i] na posição count[i] de um vetor de saída (leve em consideração a possibilidade de elementos iguais). Escreva uma função para ordenar um vetor x de tamanho n usando esse método.

EXERCÍCIO 53
Presuma que um vetor contém inteiros entre a e b, inclusive, com vários números repetidos diversas vezes. Uma **ordenação por distribuição** (BucketSort) ocorre da seguinte maneira: declare um vetor number de tamanho b – a + 1, defina number[i - a] como o número de vezes que o inteiro i aparece no vetor e, em seguida, redefina os valores no vetor concomitantemente, Escreva uma função para ordenar um vetor x de tamanho n contendo inteiros entre a e b, inclusive, com esse método.

EXERCÍCIO 54
A ordenação por transposição de par-impar ocorre da seguinte maneira. Percorra o vetor várias vezes. Na primeira passagem compare x [i] com x [i +1] para todo i ímpar. Na segunda passagem compare x [i] com x [i +1] para todo i par. Toda vez que x [i] > x [i +1] troque os dois. Continue alternando dessa maneira até que o vetor esteja ordenado.

a) Qual a condição para o término da ordenação?
b) Escreva uma função para implementar essa ordenação?
c) Qual é o custo médio dessa ordenação?

EXERCÍCIO 55
Modifique a função partição do método QuickSort de modo que o valor do meio (mediano) de x [menor], x[maior] e x[meio] (onde meio = (maior + meio)/2) seja usado para particionar o vetor.

EXERCÍCIO 56
Escreva uma versão do algoritmo Quicksort com **cutoff** para vetores pequenos: quando o vetor a ser ordenado tiver menos que M elementos, a ordenação passa a ser feita pelo algoritmo de inserção. O valor de M pode ficar entre 10 e 20. Compare os tempos de execução com o Quicksort apresentado em sala e o quicksort do exercício anterior.

BOM ESTUDO!