Universidade Tecnológica Federal do Paraná (UTFPR)
Departamento Acadêmico de Informática (DAINF)
Estrutura de Dados I
Professor: Rodrigo Minetto
Lista de exercícios (algoritmos básicos de ordenação)

Exercícios (seleção): necessário entregar TODOS (moodle)!

Exercício 1) Codifique e execute os algoritmo bubble-sort, para ordenar um quantidade de 10, 100, 1.000, 10.000, 100.000 e 200.000 elementos em ordem **aleatória**. Indique na tabela abaixo o tempo para cada uma das entradas acima. **Não inclua** no tempo de execução a impressão dos elementos na tela (comente a função **print** para estes testes de performance).

Algoritmo						00
Bubble-Sort						

```
PS C:\Dev\Estrutura-de-dados-1\Lista 7\Arquivos> ./bubble 10
Running time: 0.00
PS C:\Dev\Estrutura-de-dados-1\Lista 7\Arquivos> ./bubble 100
Running time: 0.00
PS C:\Dev\Estrutura-de-dados-1\Lista 7\Arquivos> ./bubble 1000
Running time: 0.00
PS C:\Dev\Estrutura-de-dados-1\Lista 7\Arquivos> ./bubble 10000
Running time: 0.29
PS C:\Dev\Estrutura-de-dados-1\Lista 7\Arquivos> ./bubble 100000
Running time: 38.31
PS C:\Dev\Estrutura-de-dados-1\Lista 7\Arquivos> ./bubble 200000
Running time: 151.66
PS C:\Dev\Estrutura-de-dados-1\Lista 7\Arquivos>
```

Para auxiliar nos programa **bubble.c** (em **arquivos.z**

Exercício 2) Implemente uma versão recursiva do algoritmo **bubble-sort**. Não é necessário eliminar todos os iteradores, basta escolher **um loop** para transformar em recursão — para isso, observe qual deles é menos acoplado ao código. Utilize o seguinte protótipo para a sua função:

```
void bubble_sort_recursive (int *A, int n);
```

Não altere o protótipo acima, ou seja, não adicione outras variáveis como argumentos de entrada na função para facilitar o projeto da recursão.

Exercício 3) Calcule a complexidade exata do algoritmo bubble-sort. Mostre os passos em como chegou até a ela.

Algorithm 1: Bubble sort

```
Data: Input array A[]
Result: Sorted A[]
int i, j, k;
N = length(A);
for j = 1 to N do
    for i = 0 to N-1 do
        if A[i] > A[i+1] then
            temp = A[i];
            A[i] = A[i+1];
            A[i+1] = temp;
        end
    end
end
```

Para cada Loop, podemos escrever:

$$\sum_{j=1}^{N} \sum_{i=0}^{j-1} 1 \quad (I)$$

exterior interior

↳ Esse é o pior caso. Array ordenado inversamente. Resolvendo:

$$\sum_{i=0}^{j-1} 1 = (j-1) - 0 + 1 = (j-1) + 1$$

Voltando para I

$$\sum_{j=1}^{N} (j-1) + 1 =$$

$$= (1-1) + 1 + (2-1) + 1 + (3-1) + 1 + \dots + (N-1-1) + 1 + (N-1) + 1$$

$$= 1 + 2 + 3 + \dots + (N-1) + N \quad II$$

Temos:

$$\sum_{i=1}^{N} i = 1 + 2 + 3 + \dots + (N-1) + N = \frac{N(N-1)}{2} \quad III$$

↳ Como II == III:

↳ $\sum_{j=1}^{N} \sum_{i=0}^{j-1} 1 = \frac{N(N-1)}{2}$ ↳ $O(N^2)$