Universidade Federal Rural de Pernambuco
Departamento de Estatística e Informática
Bacharelado em Sistemas de Informação
Algoritmos e Estrutura de Dados
Prof. Tiago A. E. Ferreira

1ª Lista de Exercícios

Questão 1: Supponha que o tempo de execução de um dado algoritimo seja dado por uma função matemática que recebe como argumento o tamanho **n** de entrada do problema. Calcule os tempos de execução para ***n=35,*** considerando as funções:

$$nlg(n), n,\ n^2,\ n^3, n!, e^n$$

Questão 2: Implemente em Python uma classe para ordenamento de sequencias numéricas. Esta classe deve ser capaz de gerar ordenamento com a utilização dos algoritmos:

a. Ordenamento por Inserção;
b. Ordenamento por Intercalação.

Questão 3: Reescreva o procedimento *INSERTION-SORT* para ordenar em ordem não crescente, em vez da ordem não decrescente.

Questão 4: Use a indução matemática para mostrar que, quando **n** é uma potência exata de 2, a solução de recorrência,

$$T(n) = \begin{cases} 2, & se\ n = 2, \\ 2T\left(\frac{n}{2}\right) + n, & se\ n > 2^k, para\ k > 1 \end{cases}$$

é ***T(n)=n lg(n)***

Questão 5: O *bubblesort* é um algoritmo de ordenação popular. Elefnciona permutando repetidamente elementos adjacentes que estão fora de ordem.

```
BUBBLESORT(A)
1 for i ← 1 to comprimento[A]
2     do for j ← comprimento[A] downto i + 1
3           do if A[j] < A[j − 1]
4                 then trocar A[j] ↔ A[j − 1]
```

a. Seja A´um valor que denote a saída do *BUBBLESORT(A)*. Para provar que o *BUBBLESORT* é correto, precisamos provar que ele termina e que,
$$A'[1] \leq A'[2] \leq \cdots \leq A'[n]$$
onde n = comprimento[A]. O que mais deve ser provado para mostrar que *BUBBLESORT* realmente realiza a operação
b. Enuncie com precisão um *loop* invariante par ao *loop* **For** das linhas 2 e 4 e prove que esse *loop* invariante é válido. Sua prova deve usar a estrutura da prova do *loop* invariante apresentada.
c. Usando a condição de término do *loop* invariante lhdemonstrado no ítem (b), enucie um *loop* invariante para o *loop* das linhas 1 a 4 que lhe permita provar a desiguadade do ítem (a). Sua prova deve empregar a estrutura da prova do *loop* invariante.
d. Qual o tempo de execução do pior caso de *bubblesort*? Como ele se compara ao tempo de execução da ordenação por inserção?