UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO

Disciplina: Algoritmos e Estruturas de Dados I (EDI)
Professor: Eduardo de Lucena Falcão
Exercício sobre Algoritmos de Ordenação

1) Programe na linguagem C/Go cada um dos seguintes algoritmos de ordenação:
   a) SelectionSort (in-place)
   b) BubbleSort (in-place)
   c) InsertionSort (in-place)
   d) MergeSort
   e) QuickSort (com randomização de pivô)
   f) CountingSort

2) Explique a complexidade de tempo de pior caso dos algoritmos acima utilizando o código construído. Por fim, preencha a seguinte tabela, como forma de resumo.

Algoritmo	Pior Caso	Melhor Caso
SelectionSort		
BubbleSort		
InsertionSort		
MergeSort		
QuickSort		
CountingSort		

3) Ilustre, em detalhes, o funcionamento dos seguintes algoritmos com os seguintes vetores.
   - **aleatório** = [3, 6, 2, 5, 4, 3, 7, 1, 10^9]
   - **decrescente** = [7, 6, 5, 4, 3, 3, 2, 1]
   - **crescente**=[1,2,3,3,4,5,6,7]

   a. SelectionSort (in-place)
      i. vetor para ilustrar: **aleatório**
   b. BubbleSort
      i. vetor para ilustrar: **aleatório**
   c. InsertionSort (in-place)
      i. vetor para ilustrar: **aleatório**
   d. MergeSort
      i. vetor para ilustrar: **decrescente**
   e. QuickSort (**sem randomização de pivô**)
      i. vetor para ilustrar: **crescente**
   f. QuickSort (**com randomização de pivô**)
      i. vetor para ilustrar: **crescente**

g. CountingSort
    i. vetor para ilustrar: **aleatório**

4) A seguir são apresentados 5 fatos sobre algoritmos de ordenação. Planeje, execute experimentos, e apresente resultados que evidenciem cada afirmação.
    A. Para vetores de tamanho pequeno, a performance da maioria dos algoritmos de ordenação não vai influenciar, independente da disposição dos elementos.
        a. Sugestão: um algoritmo $O(n^2)$, um algoritmo O(nlogn), e como exceção um algoritmo O(k+n)
    B. Vetor de tamanho grande, a performance do algoritmo influencia de forma significativa. Além disso, dependendo da disposição (e valores) dos elementos no vetor, podemos experimentar performances bem diferentes (melhor e pior caso).
        a. Sugestão: um algoritmo $O(n^2)$, um algoritmo O(nlogn), um algoritmo O(k+n)
    C. MergeSort tem sempre um desempenho muito bom, independente da disposição dos elementos no vetor.
    D. O pior caso do Quicksort é com o vetor ordenado de forma crescente/decrescente. O Quicksort com randomização de pivô resolve esse mau desempenho.
    E. Explique quando o CountingSort tem bom desempenho e quando tem mau desempenho mostrando os resultados através dos experimentos.

5) Perguntas com respostas rápidas:
    a) Por que SelectionSort não consegue melhorar o desempenho para cenários nos quais o vetor já está ordenado?
    b) Por que BubbleSort consegue melhorar o desempenho para cenários nos quais o vetor já está ordenado?
    c) Por que InsertionSort consegue melhorar o desempenho para cenários nos quais o vetor já está ordenado?
    d) Por que o MergeSort sempre tem o mesmo desempenho para qualquer cenário (vetor organizado de diferentes formas)?
    e) Por que o pior caso do QuickSort é $O(n^2)$?
    f) Como mitigar a probabilidade do pior caso acontecer no QuickSort?
    g) O CountingSort é melhor ou pior do que o MergeSort? E em relação ao QuickSort?