Lista de Exercícios - Algoritmos de Ordenação

Parte 1: BubbleSort e SelectionSort

1. **(Teórico)** Explique o funcionamento dos algoritmos **BubbleSort** e **SelectionSort**. Qual deles geralmente executa menos trocas de elementos?
2. **(Implementação)** Implemente o algoritmo **BubbleSort** em sua linguagem de preferência.
3. **(Implementação)** Implemente o algoritmo **SelectionSort** e compare sua saída com o BubbleSort para um mesmo conjunto de dados.
4. **(Complexidade)** Demonstre a análise de complexidade do **pior, melhor e caso médio** do BubbleSort e SelectionSort.
5. **(Comparação)** Explique, com exemplos, como o número de trocas realizadas pelo **BubbleSort** pode ser reduzido utilizando a versão otimizada (early stopping).
6. **(Teste de Performance)** Compare o tempo de execução do BubbleSort e SelectionSort para entradas de **100, 1000, 5000 e 10000 elementos aleatórios**.
7. **(Variação)** Modifique o **SelectionSort** para ordenação decrescente e implemente a solução.
8. **(Exercício Prático)** Dado um vetor com números repetidos e únicos, utilize **BubbleSort** para ordená-lo e exiba a posição inicial e final de cada número.
9. **(Algoritmo Estável?)** O BubbleSort e o SelectionSort são algoritmos **estáveis**? Justifique sua resposta com um exemplo prático.

Parte 2: InsertionSort e QuickSort

10. **(Teórico)** Explique o funcionamento do algoritmo **InsertionSort** e descreva sua aplicação em cenários reais.
11. **(Implementação)** Implemente o algoritmo **InsertionSort** e teste-o com um conjunto de números ordenados de forma decrescente.
12. **(Otimização)** Em que situação o **InsertionSort** pode ser mais eficiente do que o **BubbleSort** e o **SelectionSort**? Justifique com exemplos.
13. **(Complexidade)** Demonstre a análise de complexidade do **InsertionSort** para os casos **pior, melhor e caso médio**.
14. **(Implementação)** Implemente o **QuickSort** usando a estratégia do **pivô como primeiro elemento** e teste a execução para diferentes tamanhos de entrada.
15. **(Variação)** Implemente uma variação do QuickSort escolhendo o pivô como **mediana de três elementos** e compare o desempenho com a implementação tradicional.

16. **(Comparação)** Execute o **InsertionSort** e o **QuickSort** para vetores de tamanhos variados. Para qual tamanho de entrada o QuickSort se torna claramente superior?
17. **(Complexidade)** Explique por que o **QuickSort** tem complexidade média de **O(n log n)** e qual a condição que o faz ter um **pior caso de $O(n^2)$**.
18. **(Estabilidade)** O **QuickSort** é um algoritmo **estável**? Justifique sua resposta e forneça um exemplo onde isso faz diferença.
19. **(Exercício Prático)** Em um sistema de ranking de jogadores, onde os jogadores são ordenados por pontos, implemente um algoritmo que utiliza o **InsertionSort** para adicionar um novo jogador em um ranking ordenado.
20. **(Desafio)** Implemente um sistema que utilize o **QuickSort** para organizar uma lista de palavras em ordem alfabética e teste seu desempenho com um dicionário de mais de 10.000 palavras.