Disciplina: Estruturas de Dados 1

Aluno(a):_____ Nota: _____

Prof. Tiago Pessoa Ferreira de Lima

Questão 1: (2,5 pontos)

Descreva, passo a passo, como o algoritmo **Bubble Sort** ordenaria a seguinte lista: [77, 26, 44, 17]. Mostre a configuração da lista **após cada passagem completa do algoritmo.**

```python
def bubble_sort(lista):
    """
    Ordena a lista utilizando Bubble Sort não otimizado.
    Algoritmo estável, in-place, com complexidade O(n²) no pior caso.
    """
    n = len(lista)  # Tamanho da lista
    # Percorre a lista
    for i in range(n):  # Executa n vezes
        for j in range(0, n - 1):  # Executa n - 1 vezes
            # Compara elementos adjacentes
            if lista[j] > lista[j + 1]:
                # Troca os elementos de posição
                lista[j], lista[j + 1] = lista[j + 1], lista[j]
```

Questão 2: (2,5 pontos)

Implemente a função **bubble_sort_otimizado(lista)**. Esta versão deve incluir a otimização que interrompe a execução caso uma passagem completa pela lista seja feita sem que nenhuma troca de elementos ocorra.

Questão 3: (2,5 pontos)

O algoritmo **Bubble Sort** pode ser implementado de duas formas:

- **Versão original:** sempre percorre toda a lista em cada passagem, independentemente de trocas (questão 1).
- **Versão otimizada:** interrompe a execução se, em uma passagem completa, nenhuma troca for realizada (questão 2).

a) Qual é a complexidade de tempo no **melhor caso** para a versão original? Justifique.

b) Qual é a complexidade de tempo no **melhor caso** para a versão otimizada? Justifique.

Questão 4: (2,5 pontos)

Um algoritmo de ordenação é dito **estável** quando mantém a ordem relativa de elementos iguais na lista original.

a) O **Bubble Sort** é estável? Justifique sua resposta.

b) Dê um exemplo de uma lista com elementos repetidos em que a estabilidade (ou a falta dela) possa ser observada.

Rascunho