Ordenação Estável e Instável

Antes de falar sobre quais algoritmos são estáveis, vamos definir o que significa **ordenação estável**:

- **Ordenação Estável**: Um algoritmo de ordenação é considerado estável quando, se dois elementos possuem o mesmo valor, sua ordem relativa na lista original é mantida após a ordenação.
- **Ordenação Instável**: Um algoritmo de ordenação é instável quando a ordem relativa de elementos com valores iguais pode mudar durante a ordenação.

Exemplos de Algoritmos de Ordenação Estáveis e Instáveis

Abaixo, apresento os algoritmos que discutimos anteriormente, indicando se são estáveis ou não e demonstrando com exemplos.

Algoritmos Estáveis:

1. **Bubble Sort**: Estável
    - **Por que é estável**: O Bubble Sort compara elementos adjacentes e troca-os se estiverem na ordem errada. Se dois elementos são iguais, eles não trocam de posição, mantendo a ordem relativa.
2. **Insertion Sort**: Estável
    - **Por que é estável**: O algoritmo insere cada elemento em sua posição adequada no subarray ordenado. Se houver elementos iguais, o algoritmo mantém a ordem dos elementos iguais, pois os elementos já ordenados não são movidos, a não ser que necessário.
3. **Merge Sort**: Estável
    - **Por que é estável**: Durante a fusão das sublistas, o algoritmo coloca os elementos na ordem original se eles forem iguais, ou seja, o algoritmo nunca troca a ordem de elementos iguais.
4. **Bubble Sort** e **Insertion Sort**: São algoritmos de ordenação simples que garantem a estabilidade da ordenação.

Algoritmos Instáveis:

1. **Quick Sort**: Instável
    - **Por que é instável**: O Quick Sort pode trocar elementos com valores iguais dependendo da estratégia de escolha do pivô, o que pode alterar a ordem relativa dos elementos iguais.
2. **Selection Sort**: Instável

- **Por que é instável**: O Selection Sort pode trocar elementos iguais, já que ele seleciona o menor (ou maior) elemento e o coloca na posição final, sem considerar a ordem relativa de elementos iguais.

Exemplo Demonstrando Estabilidade:

Imaginando uma lista de tuplas com nomes e idades, onde queremos ordenar pela idade.

plaintext

Copiar código

- `[(25, "Alice"), (30, "Bob"), (25, "Charlie"), (20, "David"), (30, "Eve")]`

Ordenação Estável (Exemplo com Merge Sort): Ao ordenar essa lista por idade de forma estável, os elementos com idades iguais (25 para "Alice" e "Charlie", 30 para "Bob" e "Eve") mantêm a ordem original. Resultado:

plaintext

Copiar código

- `[(20, "David"), (25, "Alice"), (25, "Charlie"), (30, "Bob"), (30, "Eve")]`

Ordenação Instável (Exemplo com Quick Sort): Se usarmos um algoritmo instável como o Quick Sort, a ordem relativa dos elementos com idades iguais pode mudar. Por exemplo, se "Alice" e "Charlie" forem trocados de lugar, podemos ter o seguinte resultado:

plaintext

Copiar código

- `[(20, "David"), (25, "Charlie"), (25, "Alice"), (30, "Eve"), (30, "Bob")]`

Note que a ordem entre "Alice" e "Charlie" foi alterada, mesmo ambos tendo a mesma idade.

Resumo das Características de Estabilidade:

Algoritmo	Estabilidade
Bubble Sort	Estável
Insertion Sort	Estável
Merge Sort	Estável
Quick Sort	Instável
Selection Sort	Instável

Conclusão:

- **Estabilidade** em algoritmos de ordenação significa que elementos com valores iguais mantêm sua ordem relativa antes e depois da ordenação.
- Algoritmos estáveis, como o **Merge Sort**, são úteis quando a ordem original dos elementos iguais precisa ser preservada.
- Algoritmos instáveis, como o **Quick Sort**, podem ser mais rápidos, mas podem alterar a ordem de elementos iguais.
    -