Comparação dos algoritmos de ordenação: Bolha, Seleção e Inserção

Introdução

Este relatório apresenta uma análise comparativa dos algoritmos de ordenação Bolha, Seleção e Inserção, com base nos dados fornecidos para os casos melhor, médio e pior. Os critérios de comparação incluem o número de comparações e movimentações realizadas por cada algoritmo em diferentes tamanhos de entrada.

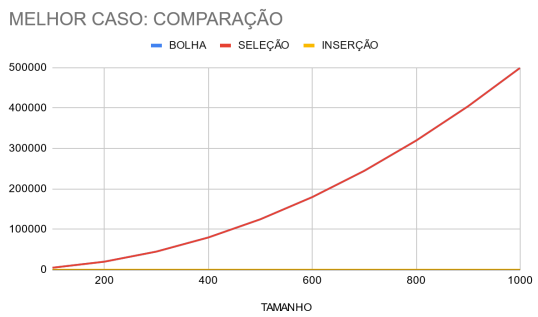
Algoritmos Analisados

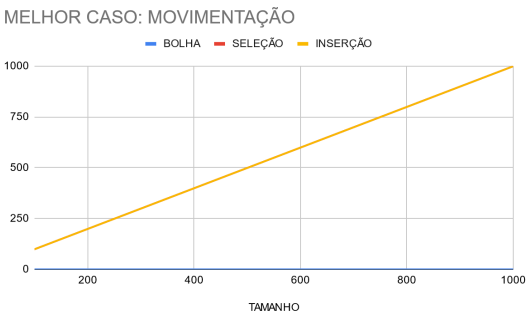
1. **Bolha (Bubble Sort)**: Algoritmo simples que compara elementos adjacentes e os troca se estiverem na ordem errada. O processo é repetido até que a lista esteja ordenada.
2. **Seleção (Selection Sort)**: Algoritmo que divide a lista em duas partes: a sub-lista de itens já ordenados e a sub-lista de itens restantes a serem ordenados. Encontra o menor elemento da sub-lista não ordenada e o troca com o primeiro elemento da sub-lista não ordenada.
3. **Inserção (Insertion Sort)**: Algoritmo que constrói a lista ordenada um item de cada vez, pegando cada elemento e inserindo-o na posição correta em relação aos elementos já ordenados.

Comparações

Melhor Caso

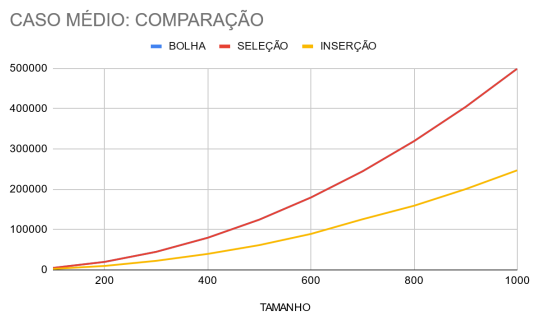
- **Bolha**: O melhor caso ocorre quando a lista já está ordenada. O algoritmo faz n−1 comparações em cada iteração até concluir que a lista está ordenada. Não há movimentações.
- **Seleção**: O melhor caso ainda requer n(n−1)/2 comparações, pois cada elemento é comparado com todos os outros elementos restantes na lista. Movimenta-se n vezes para confirmar a posição correta de cada elemento.
- **Inserção**: No melhor caso, a lista já está ordenada, e cada inserção é imediata, resultando em zero comparações adicionais e zero movimentações.

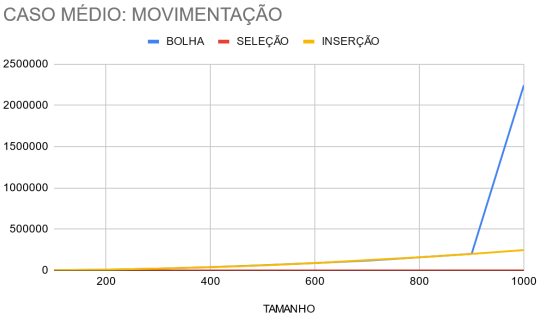




Caso Médio

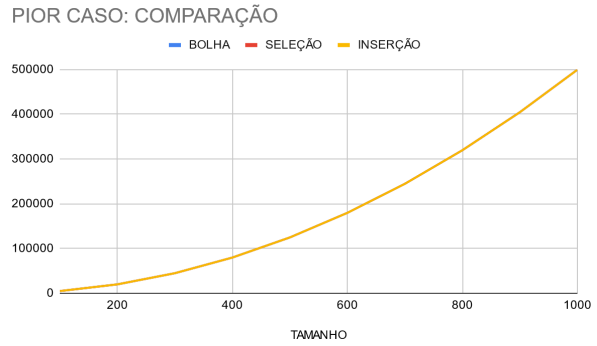
- **Bolha**: Realiza aproximadamente n(n−1)/2 comparações e movimentações adicionais conforme necessário para ordenar a lista.
- **Seleção**: O número de comparações permanece n(n−1)/2 e as movimentações são n, pois o algoritmo deve percorrer toda a lista para cada elemento.
- **Inserção**: O número de comparações e movimentações é proporcional ao quadrado do número de elementos, pois, em média, cada novo elemento é comparado com metade dos elementos já ordenados.

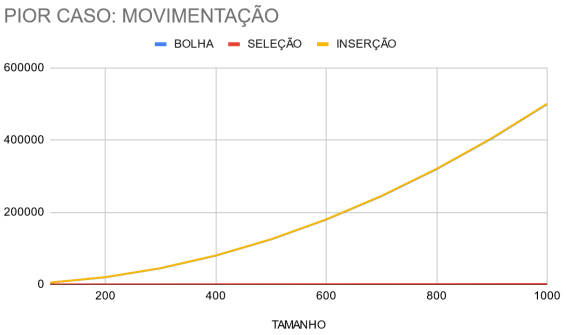




Pior Caso

- **Bolha**: O pior caso ocorre quando a lista está em ordem reversa, exigindo o máximo de comparações e movimentações n(n−1)/2.
- **Seleção**: O número de comparações é n(n−1)/2 independentemente da ordem inicial, mas as movimentações aumentam conforme a posição correta é encontrada para cada elemento.
- **Inserção**: O pior caso ocorre quando a lista está em ordem reversa, resultando no máximo número de comparações e movimentações.





Conclusão

1. **Bolha**:
    - Desempenho pior nos casos médio e pior, com muitas comparações e movimentações.
    - Melhor caso não requer movimentações, mas ainda realiza n(n−1)/2 comparações.
2. **Seleção**:
    - Número de comparações constante (n(n−1)/2 em todos os casos.

    - Movimentações são minimizadas no melhor e médio casos, mas aumentam no pior caso.
3. **Inserção**:
    - Excelente desempenho no melhor caso com zero comparações extras e movimentações proporcionais a n.
    - Desempenho intermediário no caso médio, com comparações e movimentações menores que o bolha, mas superiores ao seleção.
    - Pior caso resulta em comparações e movimentações comparáveis ao bolha.

Para aplicações práticas, a escolha do algoritmo de ordenação depende do contexto específico:

- **Bolha**: Ideal apenas para listas quase ordenadas.
- **Seleção**: Utilizável quando o custo de comparações é menos crítico que o de movimentações.
- **Inserção**: Efetivo para listas pequenas ou quase ordenadas.

A decisão final deve considerar o tamanho da lista e a distribuição inicial dos elementos para escolher o algoritmo mais eficiente.