Relatório de Comparação de Algoritmos de Ordenação

Nome: Rafael Candido da Cruz **Matrícula: 2024.1.08.041**

Introdução

Este projeto compara algoritmos de ordenação não-recursivos: Bubble Sort, Insertion Sort, Selection Sort e outros. Os testes incluíram três métodos para organizar 100.000 números não repetidos em vetores: crescente, aleatório e decrescente. O objetivo foi avaliar a eficiência dos algoritmos em termos de leituras, gravações e tempo de execução. Os resultados mostraram que o Bubble Sort teve o pior desempenho em vetores aleatórios e decrescentes, resultando em maior tempo de execução e maior número de operações. O Insertion Sort foi muito eficiente em vetores ordenados, mas apresentou desempenho ruim em vetores aleatórios e decrescentes. O Selection Sort foi mais consistente em todos os três arranjos, mas não se destacou em nenhuma situação específica.

Resultados

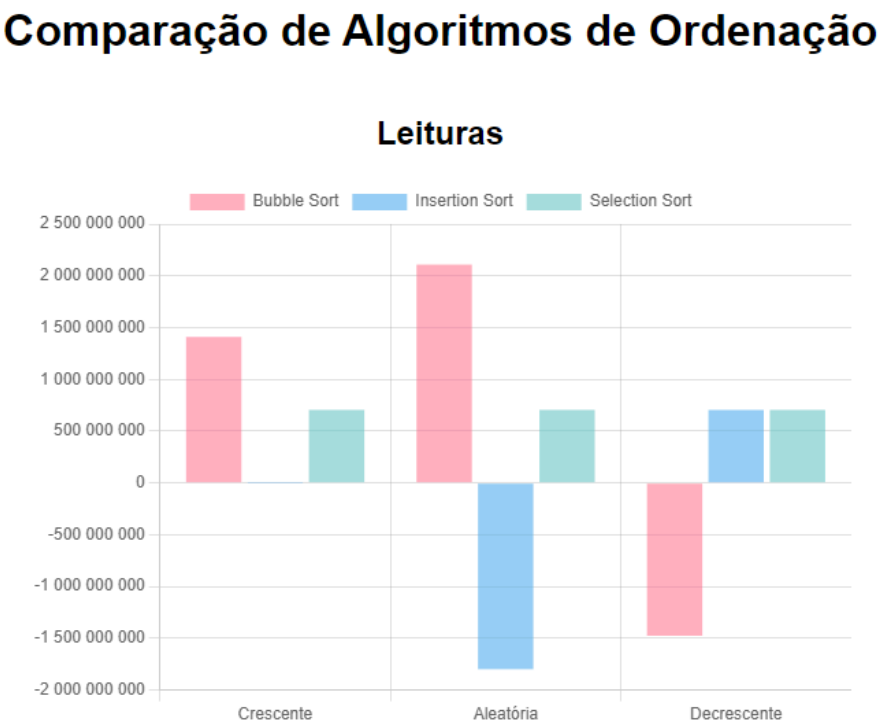
O Insertion Sort foi a forma mais eficiente de gerenciar vetores em ordem crescente, com tempo de execução de 0,000999 segundos e um número mínimo de leituras e escritas de 99.999 cada. Bubble Sort e Selection Sort tiveram tempos de execução rápidos de 19,3413 e 18,4242 segundos, respectivamente, sendo que o Selection Sort teve menos leituras e gravações. Para vetores em ordem aleatória, o Selection Sort teve um tempo de execução de 18,4539 segundos, enquanto o Bubble Sort foi o mais lento, com 50,7063 segundos e um número elevado de operações de leitura e escrita. O Insertion Sort completou a ordenação em 15,3704 segundos, mas com leituras e gravações negativas, indicando um possível erro de contagem.

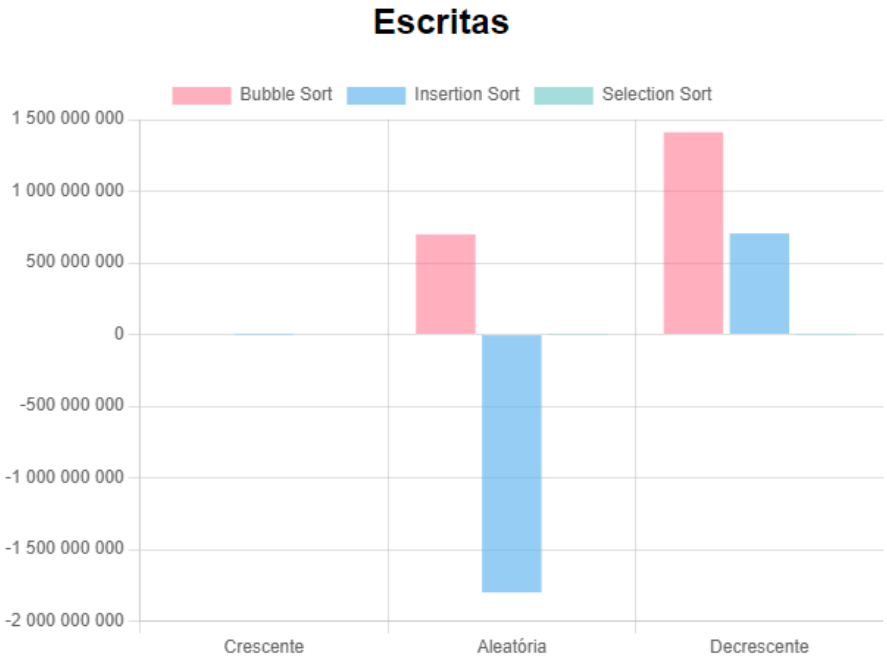
Para vetores em ordem decrescente, o Selection Sort novamente provou sua estabilidade com um tempo de execução de 18,6592 segundos. O Insertion Sort levou 30,6765 segundos para ser concluído, mas ainda assim teve um desempenho decente. O Bubble Sort teve o tempo de execução mais longo, de 58,1573 segundos, e exigiu muitas operações. Esses resultados indicam que o Bubble Sort não é eficiente para lidar com grandes conjuntos de dados desordenados ou ordenados inversamente. O Insertion Sort é ideal para dados que já estão quase ordenados. O Selection Sort, embora não seja o mais rápido, oferece uma performance consistente em diferentes disposições de vetores.

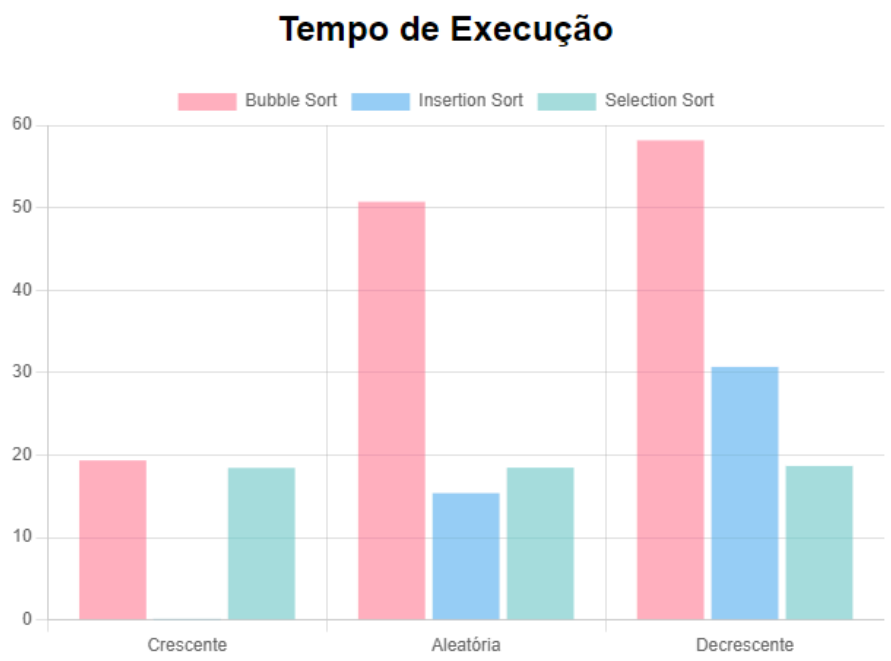
Gráficos

Reuni as informações fornecidas pelo código C++ e as utilizei para gerar gráficos que visualizam o desempenho dos algoritmos de ordenação em diferentes disposições de vetores. Usando a biblioteca Chart.js, os dados de leituras, gravações e tempos de execução foram plotados em gráficos de barras. Esses gráficos permitem uma análise clara

e comparativa das operações realizadas por cada algoritmo em diferentes cenários, facilitando a compreensão das diferenças de eficiência entre eles.







Conclusão

Em resumo, o Bubble Sort apresentou o pior desempenho em vetores aleatórios e decrescentes, enquanto o Insertion Sort foi extremamente eficiente para vetores ordenados. O Selection Sort mostrou-se o mais consistente em termos de desempenho geral. A biblioteca Chart.js, disponível em HTML, foi utilizada para visualizar gráficos de leituras, escritas e tempos de execução, proporcionando uma análise clara e detalhada dos dados obtidos. Os gráficos gerados ajudam a ilustrar as diferenças de desempenho entre os algoritmos em diversas condições, facilitando a compreensão dos resultados.

Referências

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms*. MIT Press.
2. Sedgewick, R., & Wayne, K. (2011). *Algorithms*. Addison-Wesley.
3. Chart.js Documentation. Disponível em: https://www.chartjs.org/docs/latest/