Comparação entre os métodos de ordenação

Universidade Federal de Alfenas
Ciência da Computação

Pedro Augusto de Souza Finochio
RA: 2024.1.08.020

Introdução

A eficiência e a performance de algoritmos de ordenação são cruciais em diversas aplicações computacionais. Neste relatório, investigamos e comparamos três métodos clássicos de ordenação de vetores: Bubblesort, Insertionsort e Selectionsort. Cada um desses algoritmos possui características únicas que afetam diretamente seu desempenho em diferentes cenários de entrada de dados. O objetivo deste estudo é avaliar como esses métodos se comportam em termos de tempo de execução e complexidade computacional, destacando suas vantagens e limitações práticas. Ao final da análise comparativa, esperamos fornecer insights valiosos para a escolha do método mais adequado em diferentes contextos de aplicação.

Referencial teórico

→ **Bubblesort:**
O Bubble Sort é um algoritmo de ordenação baseado na comparação e troca de elementos adjacentes. Ele recebe este nome porque elementos maiores "flutuam" para o topo da lista (final do array) em cada iteração, similar ao comportamento das bolhas em um líquido.
→ **Selectionsort:**
O algoritmo divide a lista em duas partes: uma sub-lista de itens já ordenados e uma sub-lista de itens ainda não ordenados. A cada iteração, o menor (ou maior) elemento da sub-lista não ordenada é selecionado e trocado com o primeiro elemento da sub-lista não ordenada.
→ **Insertionsort:**
O Insertion Sort ordena uma lista dividindo-a em duas sub-listas: uma sub-lista ordenada e outra não ordenada. Inicialmente, a sub-lista ordenada contém apenas o primeiro elemento, e a sub-lista não ordenada contém o restante dos elementos. O algoritmo então pega cada elemento da sub-lista não ordenada e o insere na posição correta da sub-lista ordenada.

Material Utilizado

→Para realizar esse projeto de comparação entre os métodos de ordenação de vetores, foi utilizado o NetBeans como ambiente de desenvolvimento integrado na linguagem C/C++, o sistema operacional Linux além de diversos websites como Wikipedia e StackOverflow.

Métodos Implementados

Contadores de Utilização do Vetor: Cada método de ordenação foi modificado para incluir contadores que registram cada operação de leitura e escrita no vetor. Isso permite uma análise detalhada do desempenho de cada algoritmo não apenas em termos de tempo de execução, mas também em termos de eficiência no acesso aos dados.

Comparação de Desempenho:

Para avaliar o desempenho dos métodos de ordenação, os vetores foram gerados com tamanhos variados, conforme especificado nos requisitos do projeto (de 100 até 100.000 elementos). Para cada

tamanho de vetor e para cada forma de disposição (crescente, aleatória e decrescente), os métodos foram executados e os contadores de utilização foram registrados.

Visualização dos Resultados: Os resultados obtidos foram analisados e visualizados graficamente. Os gráficos gerados mostram como os métodos de ordenação se comportam em diferentes condições de entrada (vetor crescente, aleatório e decrescente) e como variam os contadores de utilização dos vetores em função do método utilizado.

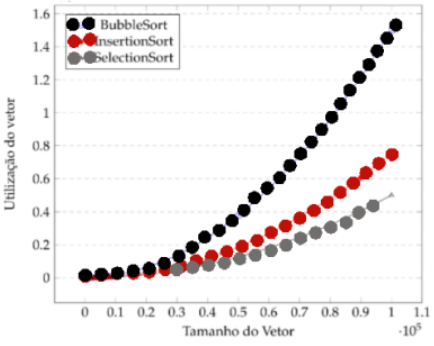
Desenvolvimento

Para que seja possível demonstrar a diferença entre os 3 métodos, o professor Paulo Bressan nos solicitou um projeto que visa demonstrar na prática quais os casos que favorecem cada tipo de método de ordenação:

→ **Descrição do professor Paulo Bressan**
Desenvolver um projeto baseado na linguagem C/C++ que ordene vetores com números não-repetidos e dispostos de três formas: crescente, aleatória e decrescente. Nos códigos de ordenação devem ser inseridos contadores de utilização do vetor, ou seja, toda vez que o vetor for utilizado (escrita ou leitura) deve-se contabilizar o seu uso para comparação entre os métodos. A comparação deve ser feita entre os três métodos implementados e variando o tamanho do vetor em intervalos regulares, de 100 em 100 unidades até 1.000, 1.000 em 1.000 unidades até 10.000, e de 10.000 em 10.000 unidades até 100.000. Os valores obtidos devem ser plotados em gráficos e fazer parte do relatório que será entregue. Na conclusão do relatório devem ser indicados quais métodos se destacaram e em quais condições.

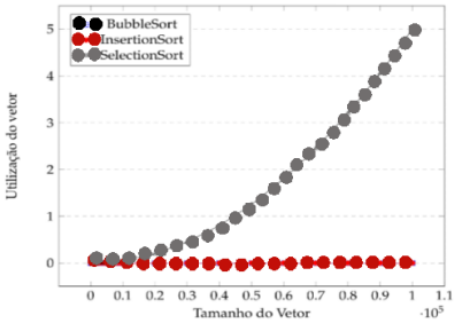
Resultados Obtidos:

Utilização dos métodos em vetores aleatórios



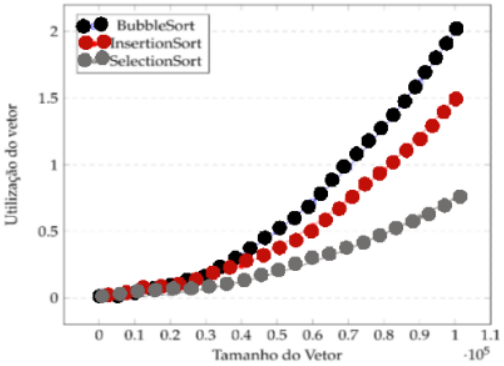
O gráfico acima mostra a utilização dos métodos de ordenação em vetores aleatórios, podemos observar que o bubblesort foi o método que mais teve que utilizar ou comparar as posições dos vetores , isso demonstra que o bubblesort utilizou mais tempo e processamento do computador para ordenar o determinado vetor.

Utilização dos métodos em vetores crescentes



Resultados evidenciam que utilizar tanto o bubblesort quanto o selectionsort para ordenar vetores em ordem crescente é inviável em projetos que requerem otimização. Nesse contexto, o método que se destaca como a opção mais recomendada é o insertionsort. O insertionsort demonstrou um desempenho superior em relação aos seus concorrentes, apresentando um contador muito menor de operações. Isso indica que, quando se trata de ordenar vetores em ordem crescente, o insertionsort é, de fato, o melhor método a ser utilizado.

Utilização dos métodos em vetores decrescentes



Com base nas estatísticas coletadas, o bubblesort demonstrou ser o método mais ineficiente, exigindo mais tempo e trocas de valores no vetor. Em contraste, o selectionsort se destacou como o melhor método para ordenar vetores crescentes, com um número significativamente menor de operações. Assim, recomenda-se o uso do selectionsort em projetos que envolvam essa necessidade específica.

***Note que em todos os gráficos o eixo y está numa escala de 1 para 1000000000**

Conclusão

→**Bubblesort**: Embora o bubblesort seja um algoritmo simples de entender e implementar, seu desempenho não é ideal para ordenar vetores grandes ou em condições adversas, como vetores decrescentes e aleatórios.
→**Insertionsort**: O insertionsort obteve vantagem para ordenar vetores decrescentes e aleatórios em relação ao bubblesort, mas mesmo assim não é o mais indicado por realizar mais operações que o selectionsort.
→**Selectionsort**: Dentre os 3 métodos de ordenação, o selectionsort foi o que teve o melhor desempenho nos vetores decrescente e aleatório, mas no vetor crescente foi o que mais levou tempo para ordenar.