Relatório de Análise Comparativa de Algoritmos de Ordenação

Instituição: Pontifícia Universidade Católica do Paraná (PUCPR)
Curso: Bacharelado em Sistemas de Informação
Disciplina: Resolução de Problemas Estruturados em Computação
Professor(a): Marina de Lara
Autores: Vinicius Viana Gomes, Vinicius Da Costa Pereira, Marcos Vinicius

1. Introdução

Este trabalho apresenta uma análise de desempenho comparativa entre três algoritmos de ordenação fundamentais: **Bubble Sort**, **Insertion Sort** e **Quick Sort**. O objetivo é avaliar a eficiência de cada algoritmo ao processar conjuntos de dados com características distintas: ordem aleatória, ordem crescente e ordem decrescente. A análise foi conduzida medindo-se o tempo de execução para ordenar vetores de 100, 1.000 e 10.000 elementos em cada um dos cenários propostos.

2. Tabela de Resultados

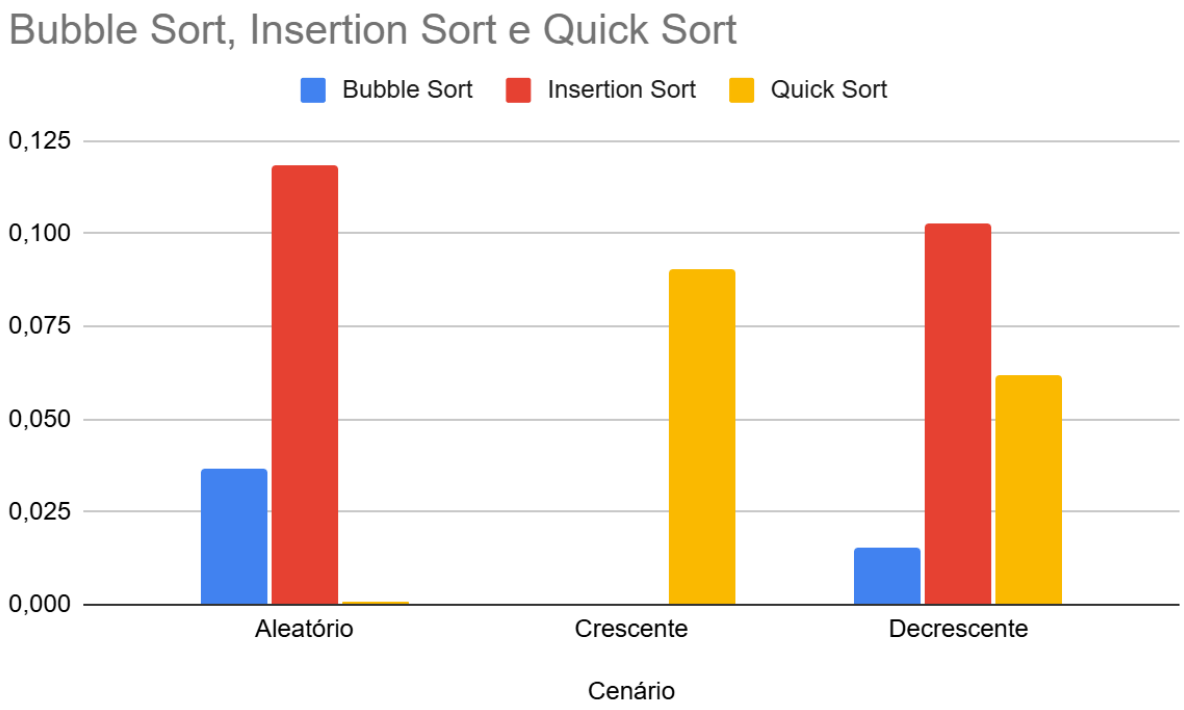
Os algoritmos foram implementados em Java e executados em ambiente local para garantir a consistência das medições. Os tempos de execução, cronometrados em segundos (s), foram compilados na tabela abaixo.

Tipo de Dados	Tamanho	Bubble Sort (s)	Insertion Sort (s)	Quick Sort (s)
Aleatório	100	0,000088	0,000197	**0,000039**
Aleatório	1.000	0,002968	0,005246	**0,000371**
Aleatório	10.000	0,036705	0,118191	**0,000927**
Crescente	100	0,000042	**0,000002**	0,000025
Crescente	1.000	0,000042	**0,000001**	0,000884
Crescente	10.000	0,000066	**0,000004**	0,090310
Decrescente	100	**0,000006**	0,000043	0,000016
Decrescente	1.000	**0,000162**	0,002050	0,000555
Decrescente	10.000	**0,015518**	0,102920	0,061717

Os tempos em negrito indicam o algoritmo mais rápido para cada teste específico.

3. Gráfico Comparativo (Dados com 10.000 elementos)

Para uma visualização mais clara do impacto do tamanho do vetor no desempenho, o gráfico abaixo compara os tempos de execução dos algoritmos para os conjuntos de dados mais volumosos (10.000 elementos).



4. Análise dos Resultados

A análise dos dados da tabela e do gráfico revela padrões de comportamento distintos para cada algoritmo, os quais estão diretamente ligados à sua complexidade computacional e à estrutura dos dados de entrada.

- Cenário 1: Dados Aleatórios (Caso Médio)
  Neste cenário, o Quick Sort demonstrou uma superioridade massiva, alinhada com sua complexidade de caso médio de O(nlogn). Para 10.000 elementos, foi aproximadamente 40 vezes mais rápido que o Bubble Sort e 127 vezes mais rápido que o Insertion Sort, provando ser a escolha mais escalável para dados desordenados.
- Cenário 2: Dados em Ordem Crescente (Melhor e Pior Caso)
  Este cenário evidenciou o melhor e o pior caso dos algoritmos. O Insertion Sort

teve um desempenho excepcional (melhor caso, O(n)), pois apenas percorreu a lista uma vez sem realizar trocas. O Bubble Sort (versão otimizada) também se beneficiou, operando em tempo linear. Em contrapartida, o Quick Sort apresentou seu pior desempenho (pior caso, O(n2)), pois a escolha de um pivô fixo em um vetor ordenado gerou partições desbalanceadas, tornando-o o mais lento dos três neste teste.

- Cenário 3: Dados em Ordem Decrescente (Pior Caso Quadrático)
  Este cenário representou o pior caso para os algoritmos de complexidade quadrática. O Insertion Sort e o Bubble Sort foram significativamente lentos, pois cada elemento necessitou do número máximo de comparações e trocas. O Quick Sort também operou em seu pior caso (O(n2)) pela mesma razão do cenário crescente. O Bubble Sort se mostrou ligeiramente mais rápido que os demais no teste com 10.000 elementos, uma particularidade que pode estar relacionada ao menor custo de suas operações de troca em comparação com o overhead da recursão do Quick Sort e das múltiplas atribuições do Insertion Sort neste estado específico.

5. Conclusão

O estudo prático confirmou a teoria de que a eficiência de um algoritmo de ordenação não é absoluta, mas sim contextual. A escolha do algoritmo mais adequado depende diretamente das características do conjunto de dados a ser processado.

O **Quick Sort** é a solução preferencial para grandes volumes de dados sem ordem pré-definida. O **Insertion Sort** destaca-se como uma ferramenta extremamente eficaz para dados que já se encontram ordenados ou parcialmente ordenados. Por fim, o **Bubble Sort**, embora de grande valor didático, prova-se ineficiente para a maioria das aplicações práticas em larga escala.

Este trabalho reforça a importância fundamental de se compreender não apenas o funcionamento, mas também a complexidade e os casos de uso de diferentes algoritmos para a resolução de problemas de forma eficiente.