RELATÓRIO - SORT PROJECT

Alunos: Vinicius Dorneles e Adriano Vale

INTRODUÇÃO

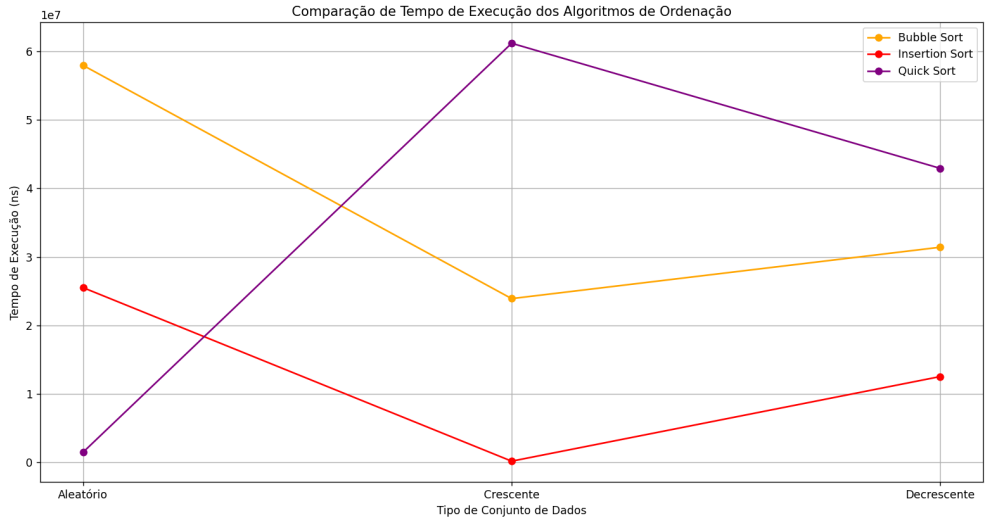
Comparamos a eficiência dos algoritmos de ordenação Bubble Sort, Insertion Sort e Quick Sort em diferentes conjuntos de dados, variando em ordem (aleatória, crescente e decrescente) e tamanho (100, 1.000 e 10.000 elementos). A implementação foi feita em Java, com medição precisa do tempo de execução usando `System.nanoTime()`. Os resultados obtidos permitem analisar o desempenho de cada algoritmo em termos de tempo, auxiliando na compreensão de suas vantagens e limitações em diferentes contextos de uso.

RESULTADO DOS TEMPOS DE EXECUÇÃO:

Conjunto de Dados	Bubble Sort (ns)	Insertion Sort (ns)	Quick Sort (ns)
Aleatório 100	268,800	85,600	36,600
Aleatório 1.000	5,921,200	2,381,300	421,200
Aleatório 10.000	51,751,400	23,047,200	1,053,200
Crescente 100	7,300	38,500	26,900
Crescente 1.000	331,100	67,600	809,300
Crescente 10.000	23,577,000	92,500	60,333,600
Decrescente 100	7,200	7,800	20,200
Decrescente 1.000	280,900	115,300	374,400
Decrescente 10.000	31,124,400	12,419,000	42,526,000

RESULTADOS RESUMIDOS:

Tipo de Conjunto de Dados	Bubble Sort (ns)	Insertion Sort (ns)	Quick Sort (ns)
Aleatório	57,941,400	25,514,100	1,511,000
Crescente	23,915,400	198,600	61,169,800
Decrescente	31,412,500	12,542,100	42,920,600



CONCLUSÃO:

Os resultados mostram que o **Quick Sort** é o mais eficiente em dados **aleatórios**, mas seu desempenho piora em conjuntos **crescente** e **decrescente**. O **Insertion Sort** é o melhor em dados **crescente**, sendo eficiente para conjuntos ordenados. Já o **Bubble Sort** é o mais lento em todos os cenários, especialmente em dados desordenados, devido à sua complexidade elevada. Em resumo, o Quick Sort é ideal para dados aleatórios, o Insertion Sort para dados ordenados, e o Bubble Sort é o menos eficiente em todos os casos.