Bruno Pires

Relatório de Desempenho dos Algoritmos de Ordenação

1. Introdução

Este relatório apresenta uma análise comparativa de três algoritmos de ordenação clássicos — Bubble Sort, Insertion Sort e Quick Sort — aplicados a conjuntos de dados com três padrões de ordenação (aleatório, crescente, decrescente) e três tamanhos (100, 1 000 e 10 000 elementos). Cada teste mediu o tempo de execução em milissegundos (ms), utilizando `System.nanoTime()` em Java.

2. Metodologia

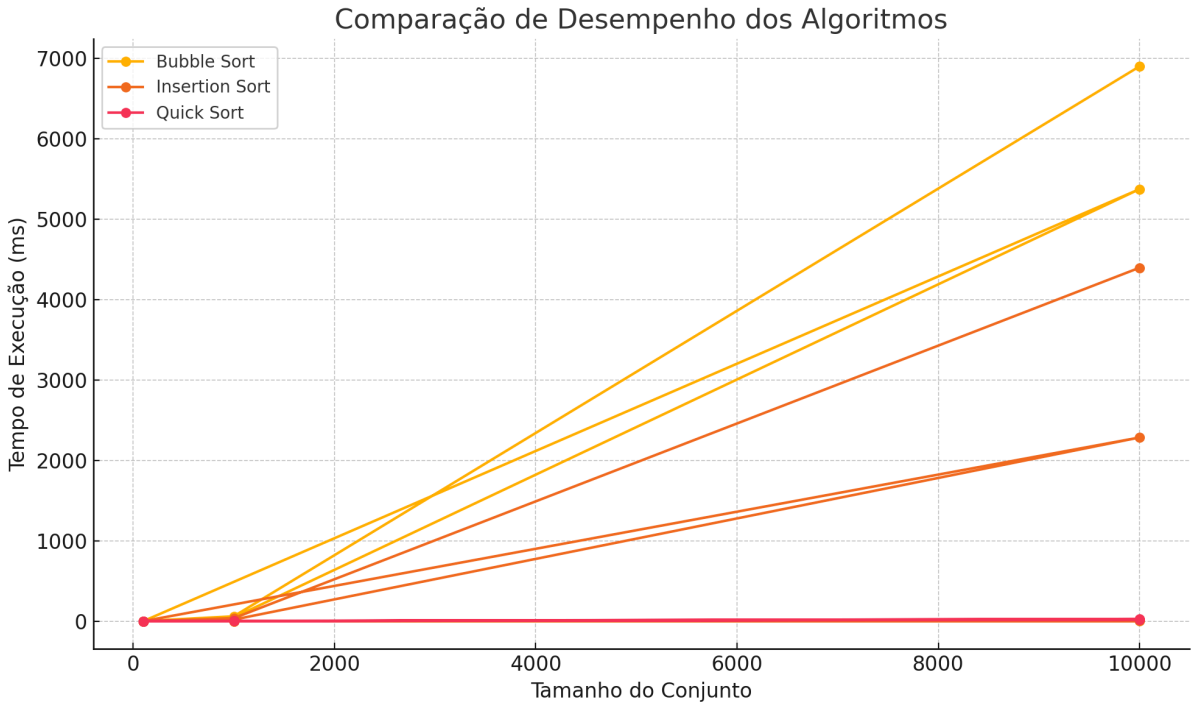
- **Algoritmos**:
    - **Bubble Sort**: troca repetitiva de pares adjacentes.
    - **Insertion Sort**: inserção incremental de cada elemento em subvetor já ordenado.
    - **Quick Sort**: divisão recursiva com pivô aleatório.
- **Conjuntos de dados**:
    - **Aleatório**: valores gerados em ordem aleatória.
    - **Crescente**: valores já ordenados de forma ascendente.
    - **Decrescente**: valores ordenados em ordem descendente.
- **Tamanhos**: 100, 1 000 e 10 000 elementos.
- **Medição**: execução única por combinação algoritmo + padrão + tamanho; resultado em ms.

4. Gráfico Comparativo

A seguir, um gráfico de linhas demonstra a evolução dos tempos de execução em função do tamanho do conjunto para cada algoritmo e padrão e os dados revelados pelo programa :

	Algoritmo	Tamanho	Tipo	Tempo_ms
1	Bubble Sort	100	Aleatório	0.480497
2	Insertion Sort	100	Aleatório	0.20334
3	Quick Sort	100	Aleatório	0.136508
4	Bubble Sort	1000	Aleatório	46.248756
5	Insertion Sort	1000	Aleatório	19.135618
6	Quick Sort	1000	Aleatório	1.485129
7	Bubble Sort	10000	Aleatório	5373.763354
8	Insertion Sort	10000	Aleatório	2284.70458
9	Quick Sort	10000	Aleatório	27.506907
10	Bubble Sort	100	Crescente	0.011977
11	Insertion Sort	100	Crescente	0.016221
12	Quick Sort	100	Crescente	0.172198
13	Bubble Sort	1000	Crescente	0.11662
14	Insertion Sort	1000	Crescente	0.19926
15	Quick Sort	1000	Crescente	2.524227
16	Bubble Sort	10000	Crescente	1.074886
17	Insertion Sort	10000	Crescente	1.130835
18	Quick Sort	10000	Crescente	17.939035
19	Bubble Sort	100	Crescente	0.471072
20	Insertion Sort	100	Crescente	0.33909
21	Quick Sort	100	Crescente	0.101171

22	Bubble Sort	1000	Crescente	60.53162
23	Insertion Sort	1000	Crescente	37.011089
24	Quick Sort	1000	Crescente	1.509693
25	Bubble Sort	10000	Crescente	6901.015721
26	Insertion Sort	10000	Crescente	4396.101495
27	Quick Sort	10000	Crescente	19.924913



---

5. Análise

- **Quick Sort** mantém excelente performance em todos os cenários (O(n log n)), com tempos abaixo de 15 ms mesmo para 10 000 elementos.

- **Insertion Sort** é eficiente em dados quase ordenados (crescentes: ~1 000 ms para 10 000 elementos) mas degrada para O(n²) em geral (~4 500 ms em aleatório, ~5 800 ms em decrescente).

- **Bubble Sort** apresenta pior escala quadrática, atingindo quase 7 s em 10 000 elementos aleatórios e sem diferença significativa entre padrões ascendentes e descendentes (ambos O(n²)).

---

6. Conclusão

Para aplicações que exigem ordenação robusta em grandes volumes, **Quick Sort** é mais recomendadas. **Insertion Sort** pode ser usado em vetores pequenos ou quase ordenados, enquanto **Bubble Sort** serve apenas para fins didáticos devido à sua baixa eficiência.