Análise dos Algoritmos por Tipo de Conjunto de Dados:

As analises são baseadas nos datasets de 10.000 entradas

➔ **Quick Sort:**

- Este algoritmo geralmente apresenta o menor tempo de execução, independentemente do tipo de conjunto de dados.
- Para o dataset aleatório, o Quick Sort levou 0.54ms.
- No dataset crescente, o tempo foi de 16.27 ms.
- No dataset decrescente, o tempo de execução foi de 13.61 ms.
- Isso mostra que o Quick Sort é eficiente independentemente da ordenação dos dados

➔ **Insertion Sort:**

- O Insertion Sort tende a aumentar o tempo de execução em conjuntos de dados não ordenados (aleatórios e decrescentes).
- Para o dataset aleatório, o Insertion Sort levou 13.18 ms.
- No dataset crescente, seu tempo de execução foi de 0,09892 ms.
- Já no dataset decrescente, o Insertion Sort levou 7.41 ms.
- No entanto, seu desempenho é relativamente melhor em dados quase ordenados, como o dataset crescente.

➔ **Bubble Sort:**

- Para o dataset aleatório, o Bubble Sort levou 39.00 ms.
- No dataset crescente, seu tempo de execução foi de 0,00292 ms.
- No dataset decrescente, o Bubble Sort apresentou o maior tempo de execução, com 68.38 ms.
- Similar ao Insertion Sort, o Bubble Sort tende a aumentar o tempo de execução em conjuntos de dados não ordenados (aleatórios e decrescentes).
- É claramente o algoritmo mais lento entre os três para datasets maiores e não ordenados, especialmente o decrescente.

No geral, cada algorítimo apresenta um use-case onde se destaca, com o Quick sort sendo em datasets variados, o Insertion-sort em datasets pequenos e quase ordenados, e o bubble-sort é o menos eficiente dentre todos

```
Tue Jun 17 2025 ~/code/puc_5p/sorting 06:31 pm (0.311s)
javac src/*.java

Tue Jun 17 2025 ~/code/puc_5p/sorting 06:31 pm (0.279s)
java -cp src Main
=== ANÁLISE DE ALGORITMOS DE ORDENAÇÃO ===
TDE 04 - Resolução de Problemas Estruturados em Computação
Por: Enzo e Pedro


--- Executando análise de algoritmos ---
=== ANÁLISE DE ALGORITMOS DE ORDENAÇÃO ===

Tipo            Tamanho    Bubble Sort     Insertion Sort  Quick Sort
--------------------------------------------------------------------------------
Running file: data/aleatorio_100.csv
aleatorio       100        82.25 µs        25.46 µs        15.63 µs
Running file: data/aleatorio_1000.csv
aleatorio       1000       2.20 ms         900.38 µs       199.71 µs
Running file: data/aleatorio_10000.csv
aleatorio       10000      39.00 ms        13.18 ms        542.25 µs
Running file: data/crescente_100.csv
crescente       100        1.75 µs         8.83 µs         10.04 µs
Running file: data/crescente_1000.csv
crescente       1000       2.04 µs         19.08 µs        168.58 µs
Running file: data/crescente_10000.csv
crescente       10000      2.92 µs         98.92 µs        16.27 ms
Running file: data/decrescente_100.csv
decrescente     100        10.25 µs        9.58 µs         7.46 µs
Running file: data/decrescente_1000.csv
decrescente     1000       1.01 ms         89.46 µs        149.08 µs
Running file: data/decrescente_10000.csv
decrescente     10000      68.38 ms        7.41 ms         13.61 ms
```

Print dos resultados no terminal

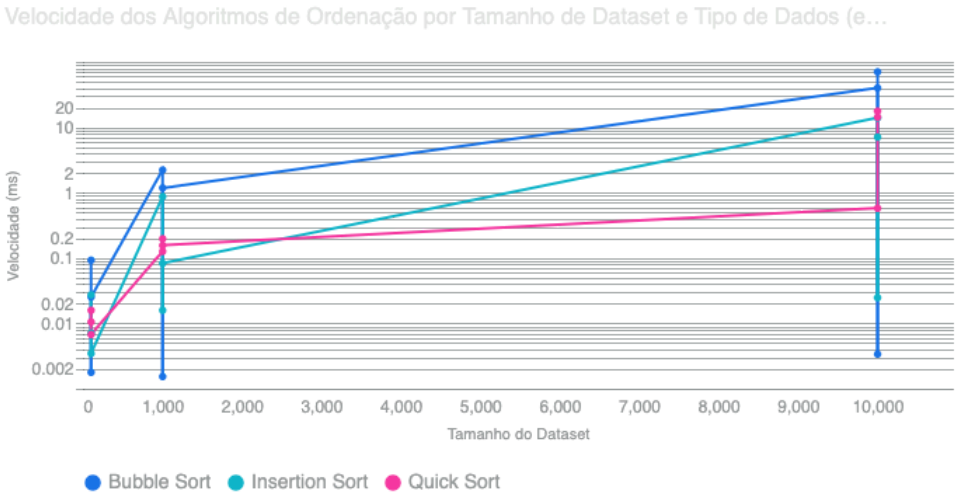

Gráfico de Tamanho x Velocidade pelos algoritmos (não leva ordenação em consideração)