***RELÁTORIO DE RESULTADOS***

**Algoritmos de Ordenação**

*Emerson Boiani e Gabriel Trevisan*

***Tabela Comparativa***

|  |  |  |  |
| --- | --- | --- | --- |
| **Conjunto de Dados** | **Bubble Sort (ns)** | **Insertion Sort (ns)** | **Quick Sort (ns)** |
| aleatorio\_100 | 466560 | 301100 | 252600 |
| aleatorio\_1000 | 2774080 | 1373098 | 220590 |
| aleatorio\_10000 | 89447400 | 25584700 | 674200 |
| crescente\_100 | 2700 | 26400 | 15000 |
| crescente\_1000 | 127800 | 33000 | 386800 |
| crescente\_10000 | 1384480 | 54398 | 52754600 |
| decrescente\_100 | 3060 | 51600 | 13200 |
| decrescente\_1000 | 270810 | 300400 | 591900 |
| decrescente\_10000 | 23362380 | 37756600 | 25724100 |

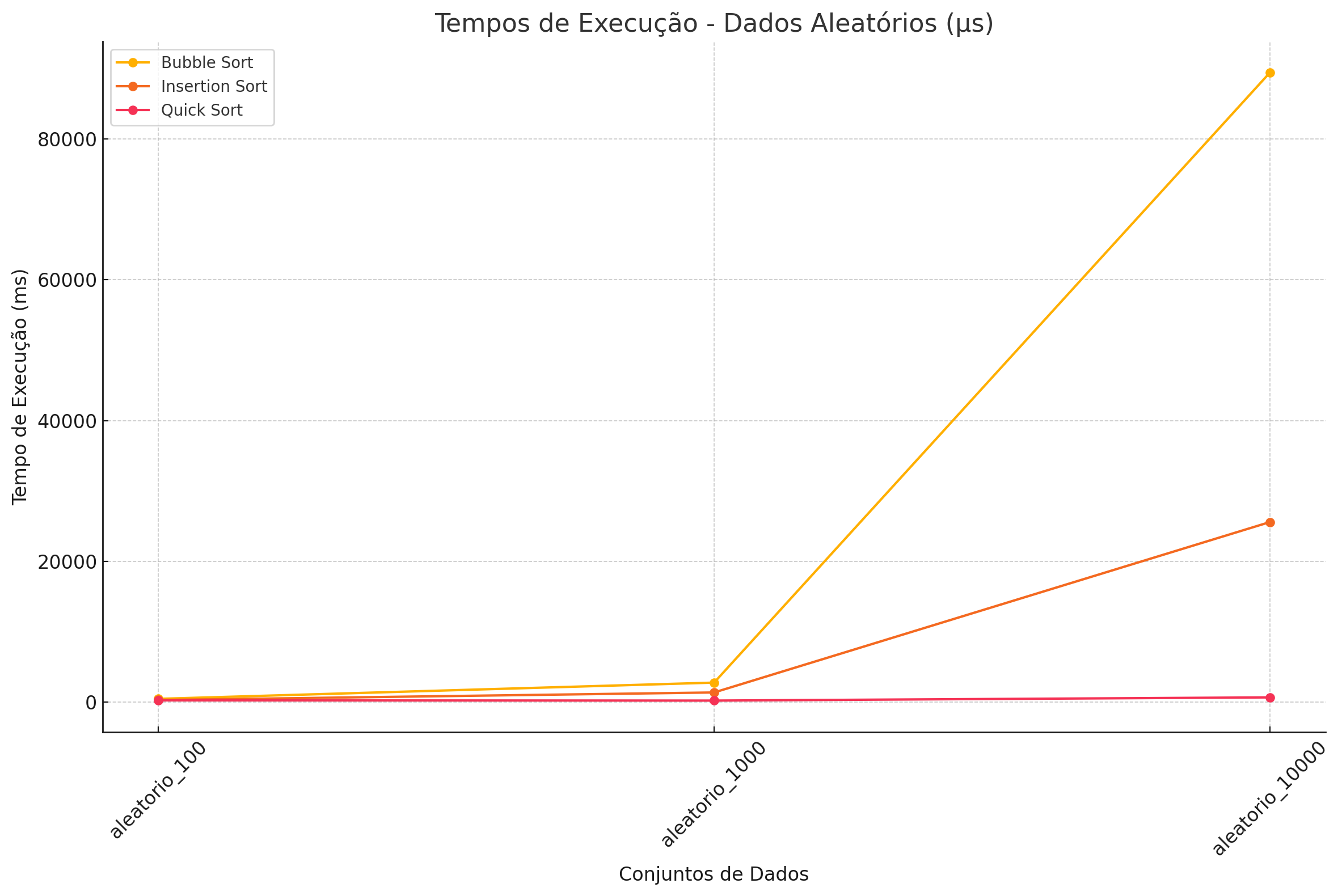
***Gráficos***

*Contém 3 gráficos, um para cada estilo de arquivo de dado;*

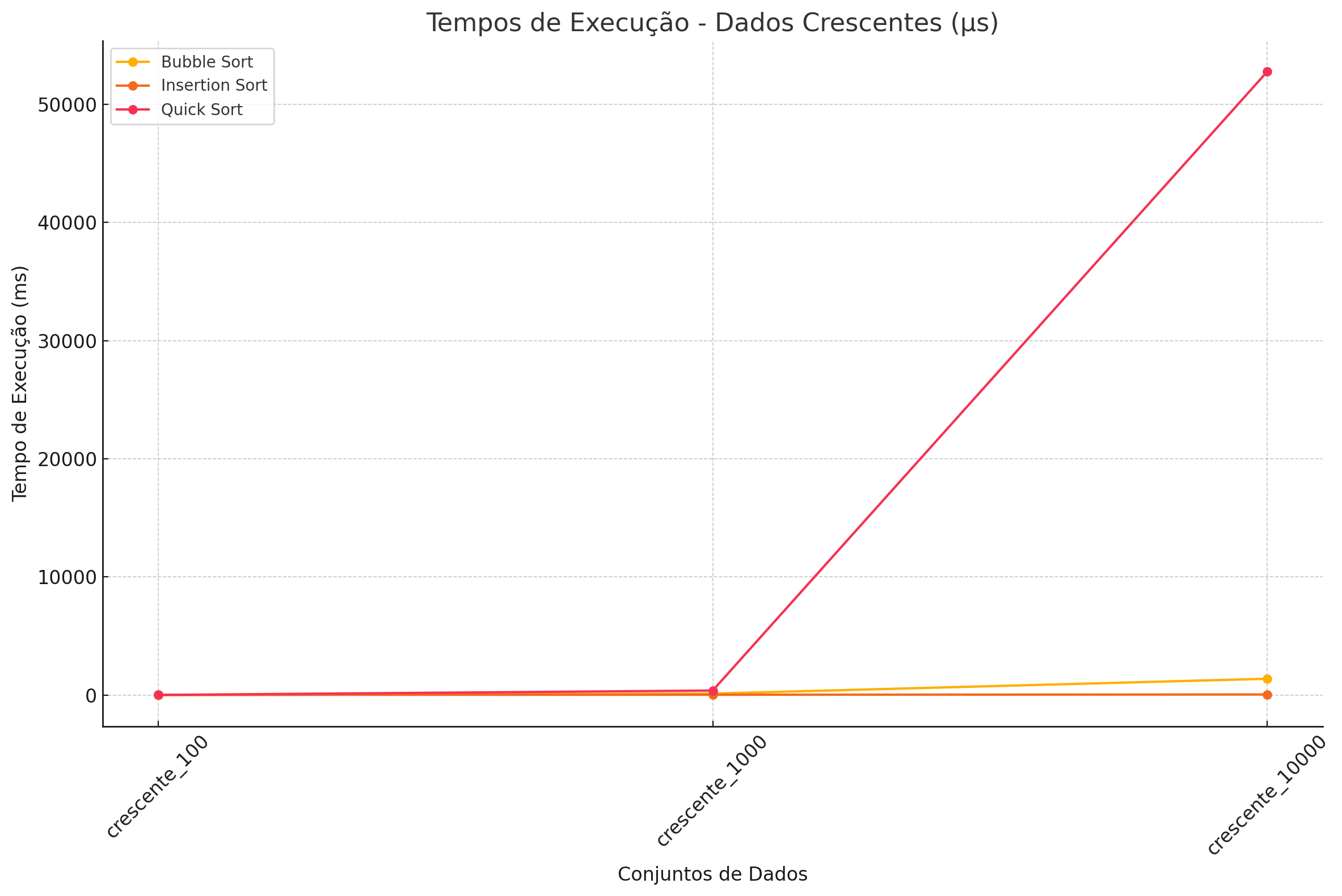
*Cada gráfico representa a ação do bubble, do insert e do quick no estilo do arquivo;*

*Gráficos em nanosegundos (1^7);*

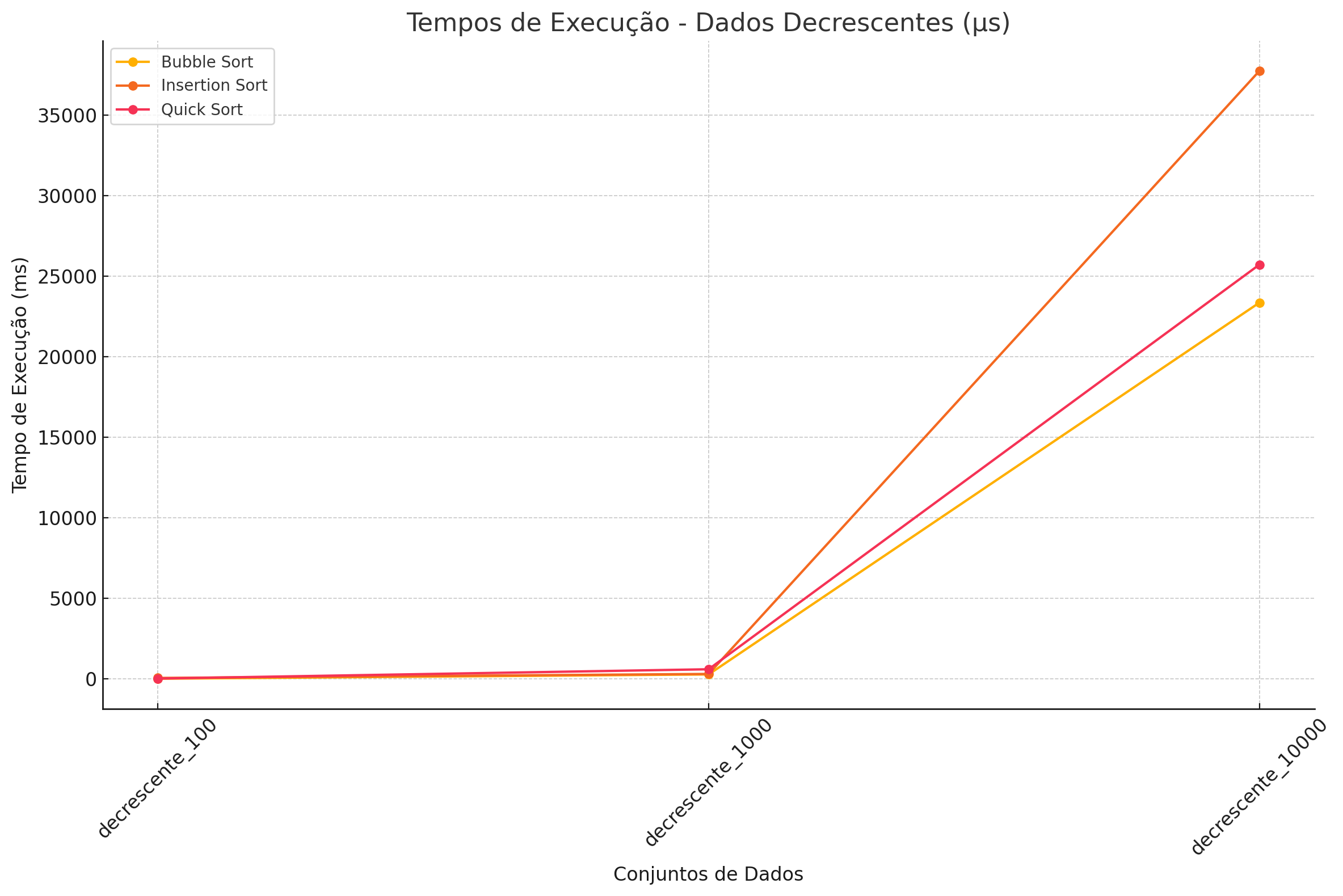
1. *Dados aleatórios*



1. *Dados Crescentes*



1. *Dados Decrescentes*



**Análise de Desempenho dos Algoritmos**

*A partir dos gráficos e dos resultados da tabela, podemos observar o desempenho de cada algoritmo de ordenação em diferentes cenários:*

1. *Dados Aleatórios*

*Quick Sort foi o mais eficiente para todos os tamanhos de dados aleatórios (100, 1000 e 10.000), o que é esperado, pois o Quick Sort tem um desempenho melhor em conjutos de dados aleatórios pela escolha do pivot, oq também torna uma desvantagem em ordenados.*

*⁡Insertion Sort apresentou tempos melhores que o Bubble Sort, especialmente nos conjuntos de dados maiores. Pois eu acredito que a divisão torna mais eficiente a organização*

*Bubble Sort foi o menos eficiente nos dados aleatórios, mostrando um tempo de execução significativamente mais alto para 10.000 elementos. Pois acredito que ler cada elemento e flutuar ele até o seu indice requer muitas comparações e leitura.*

1. *Dados Crescentes*

*Insertion Sort foi o mais eficiente nos conjuntos de dados crescentes, especialmente para os dados ordenados em 10.000 elementos, refletindo seu desempenho ótimo em dados que já estão em ordem ou quase ordenados*

*Quick Sort apresentou um tempo maior para os dados crescentes de 10.000 elementos, o que é explicado como exemplo dos dados aleatórios, pela seleção de um pivô desfavorável em um conjunto de dados já ordenado.*

*Bubble Sort também teve um bom desempenho em dados já ordenados, mas ficou atrás do Insertion Sort, creio que pelos mesmos motivos.*

1. *Dados Decrescentes*

*Quick Sort e Bubble Sort tiveram um desempenho relativamente bom nos conjuntos de dados decrescentes, com Quick Sort sendo o mais rápido para 100 e 1000 elementos.*

*Insertion Sort foi o mais lento nos dados decrescentes, especialmente em 10.000 elementos. Isso é esperado, pois o Insertion Sort tem um pior caso em dados ordenados em ordem inversa.*