unipampa

Universidade Federal do Pampa

Implementação e Análise de Desempenho de Algoritmos de Ordenação

Discentes: Davi Lemos e Heduardo Witkoski

Algoritmos e Classificação de Dados

Descrição do Escopo

Analisar **<os métodos de ordenação de dados>**
para o propósito de **<avaliação>**
com respeito ao **<tempo de execução de cada método>**
do ponto de vista do(a) **<programador>**

Introdução

Diferentes métodos de ordenação, com diversas aplicações.

Algoritmo	Melhor Caso	Caso Médio	Pior Caso
Bubblesort	O(n)	O(n²)	O(n²)
Insertionsort	O(n)	O(n²)	O(n²)
Selectionsort	O(n²)	O(n²)	O(n²)
Shellsort	O(n log n)	O(n log² n)	O(n²)
Heapsort	O(n log n)	O(n log n)	O(n log n)
Mergesort	O(n log n)	O(n log n)	O(n log n)
Quicksort	O(n log n)	O(n log n)	O(n²)
*Countingsort	O(n + k)	O(n + k)	O(n + k)
*Radixsort	O(nk)	O(nk)	O(nk)
*Bucketsort	O(n + k)	O(n + k)	O(n²)

*Lineares

Metodologia

- Implementação em Java com base nos pseudocódigos.
- Diferentes vetores aplicados nos testes:
  - Aleatório de 10^2, 10^3, 10^4 e 10^5 elementos;
  - 10^4 elementos em ordem decrescente;
  - 10^4 elementos em ordem crescente;
  - 10^4 elementos com valores repetindo 5 vezes cada.

- Análise distinta para os algoritmos lineares.
- Teste comparable -> int

Metodologia

- Coleta do tempo de 10 execuções para cada experimento.

- Ambiente de execução:
  - Microprocessador multicore AMD Ryzen 7 5700U 3,3 GHz, 8 núcleos com 16 threads;
  - Placa de vídeo integrada AMD Radeon Graphics 8;
  - 12 GB de memória DDR4;
  - Distribuição Linux Ubuntu 24.04 LTS;

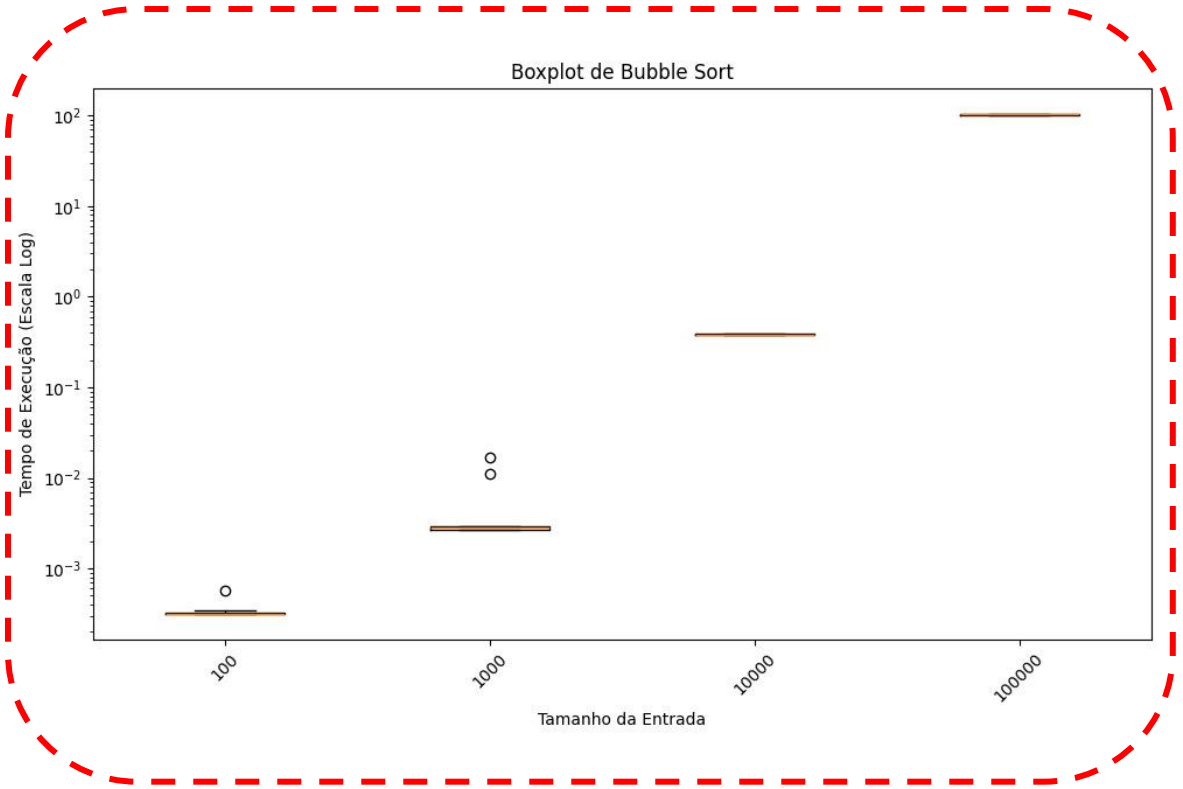
Resultados – Média dos tempos de execução (Vetor Aleatório)

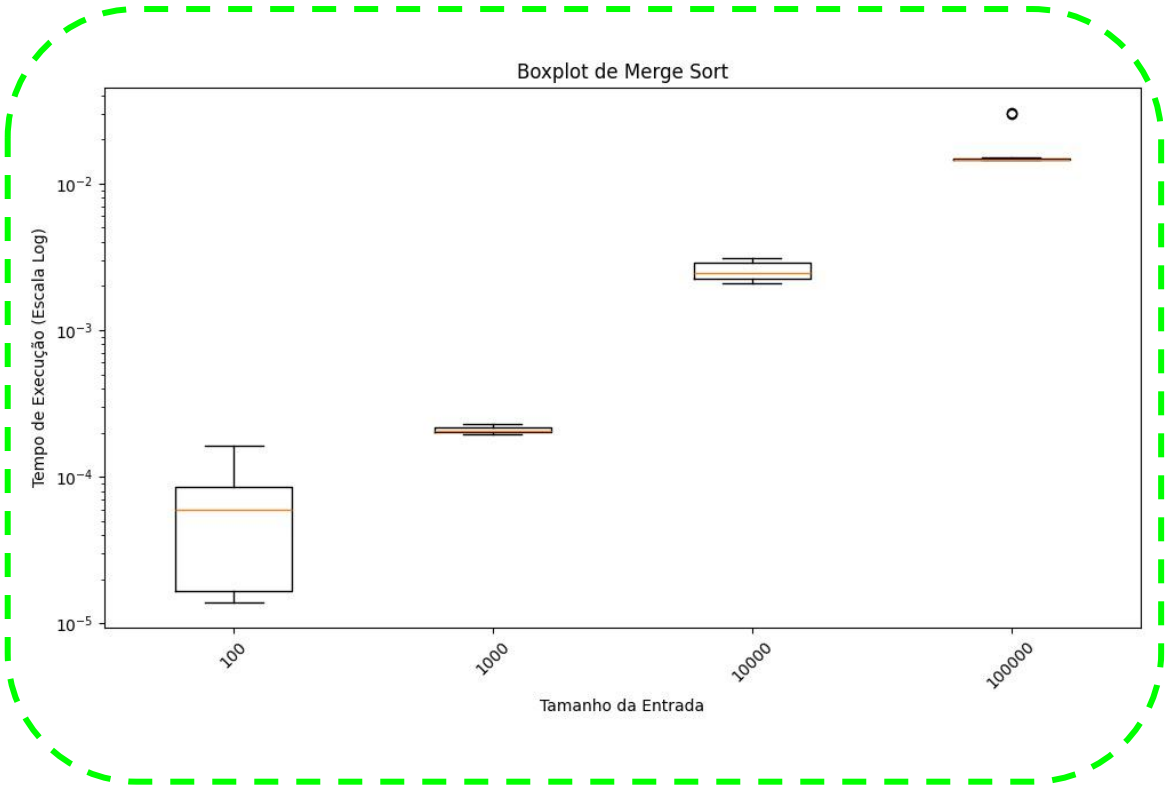
Tamanho da entrada	Tempo de execução (s)						
	Bubble Sort	Heap Sort	Insertion Sort	Merge Sort	Quick Sort	Selection Sort	Shell Sort
100	0.000346221300	0.000055491500	0.000178389100	0.000061492800	0.000095780100	0.000226193000	0.000073616500
1000	0.005010706100	0.001360851300	0.003556061200	0.000209137300	0.000215129700	0.005039750200	0.000560814000
10000	0.382374560900	0.002286988000	0.084830699000	0.002550383200	0.005628722900	0.083124567000	0.004780544000
100000	103.317613809200	0.044113279400	35.666967811000	0.017729851200	0.019331866500	45.112199601900	0.094067428900

Algoritmos Lineares:

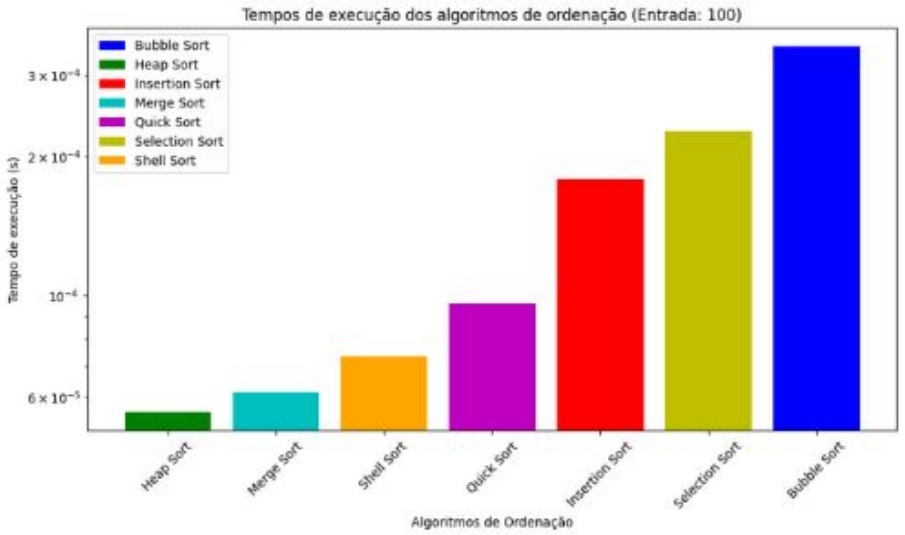
Tamanho da entrada	Tempo de execução (s)		
	Bucket Sort	Counting Sort	Radix Sort
100	0.0001354719	0.0001580994	0.0000862194
1000	0.0005679654	0.0001262816	0.000354145
10000	0.0026749743	0.0006537406	0.0017380147
100000	0.0160765804	0.0024028499	0.0075375496

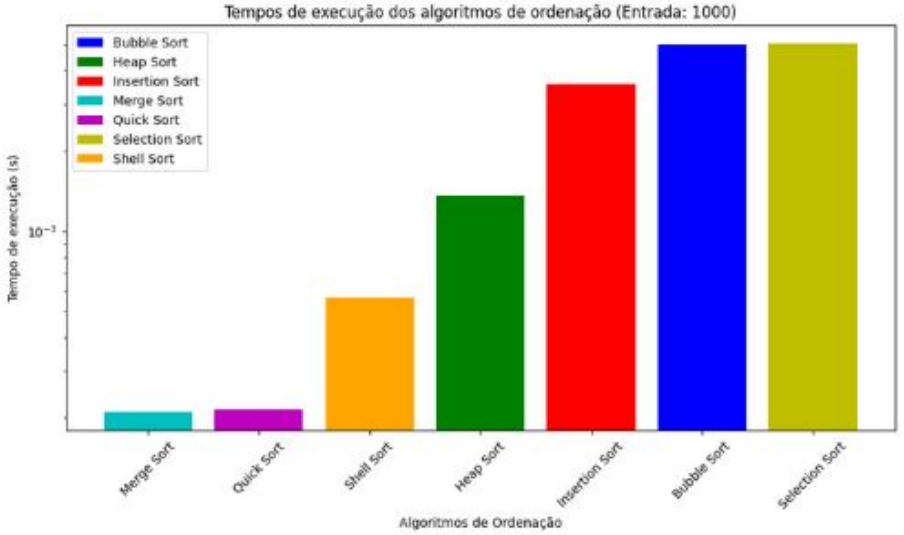
Resultados – Gráficos de caixa (*boxplots*)

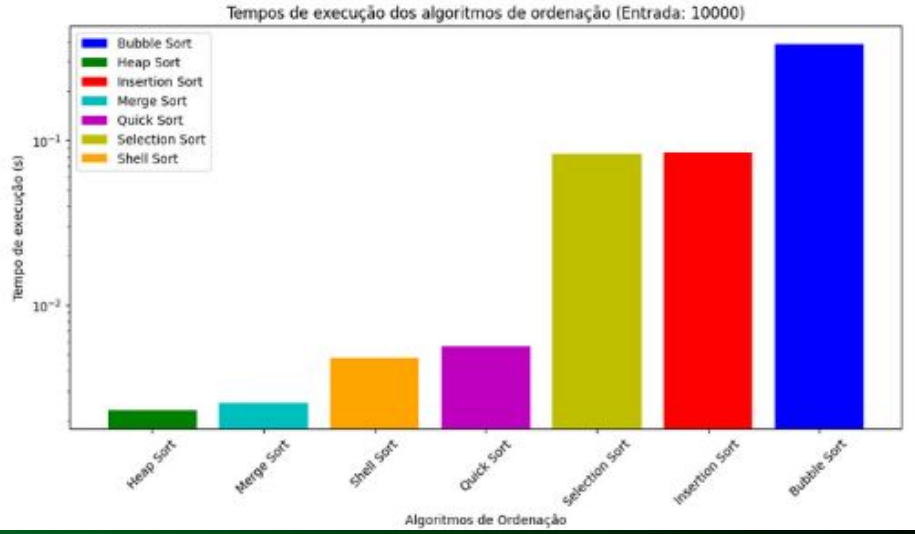


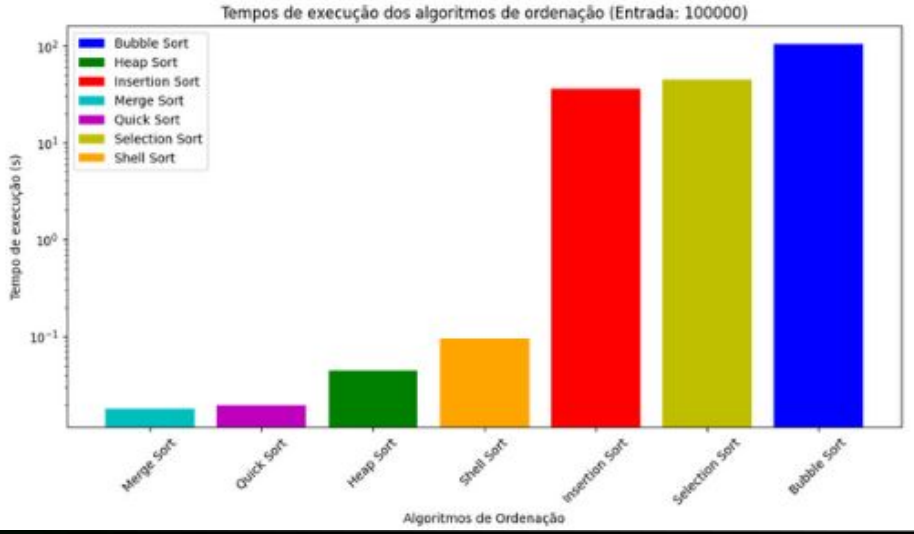


Resultados – Análise dos algoritmos não-lineares

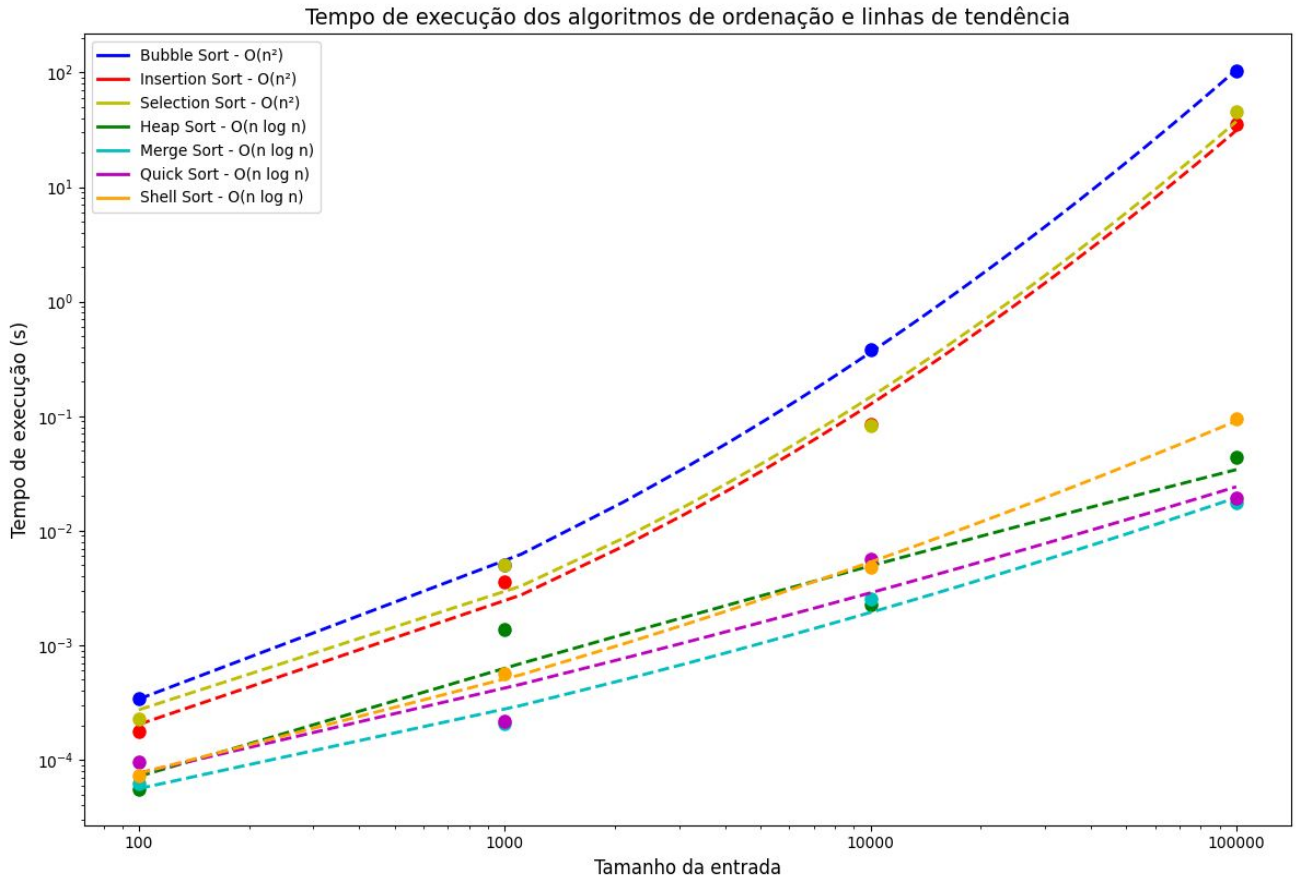


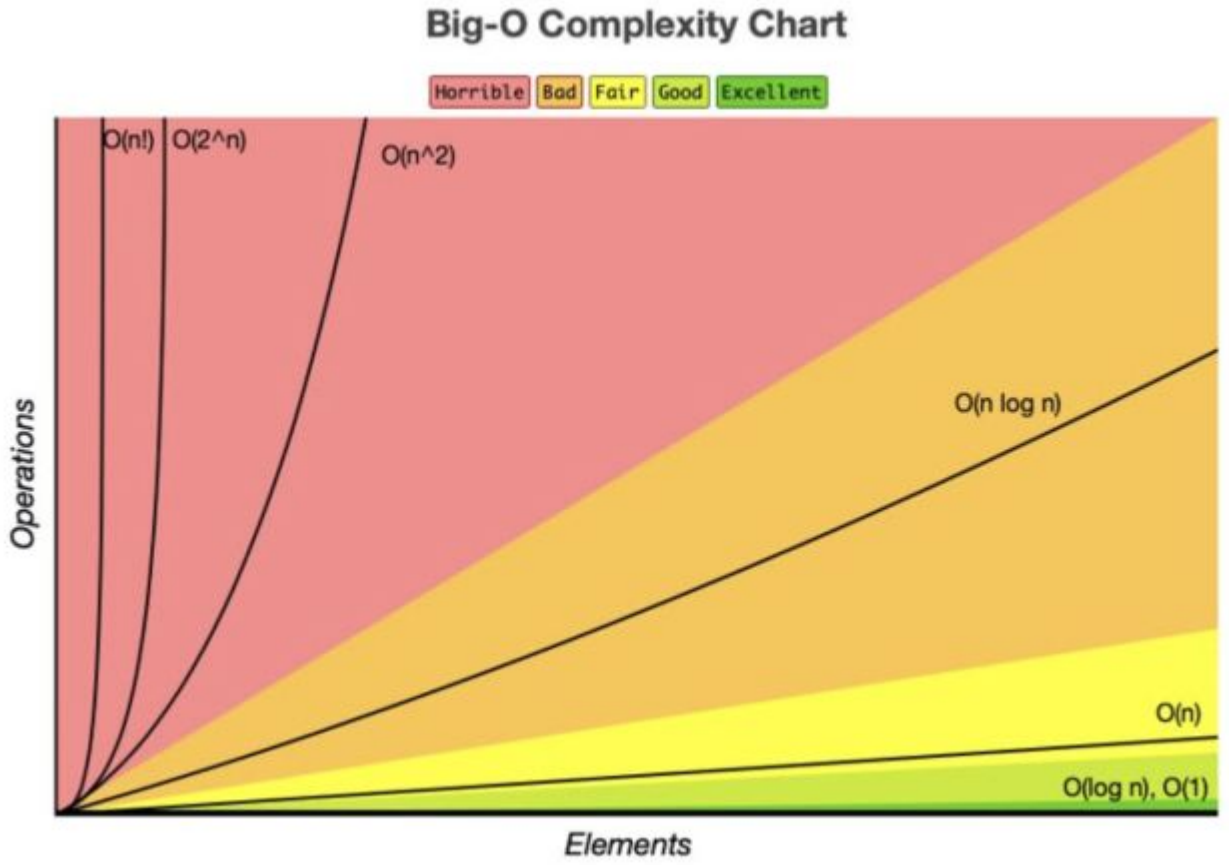




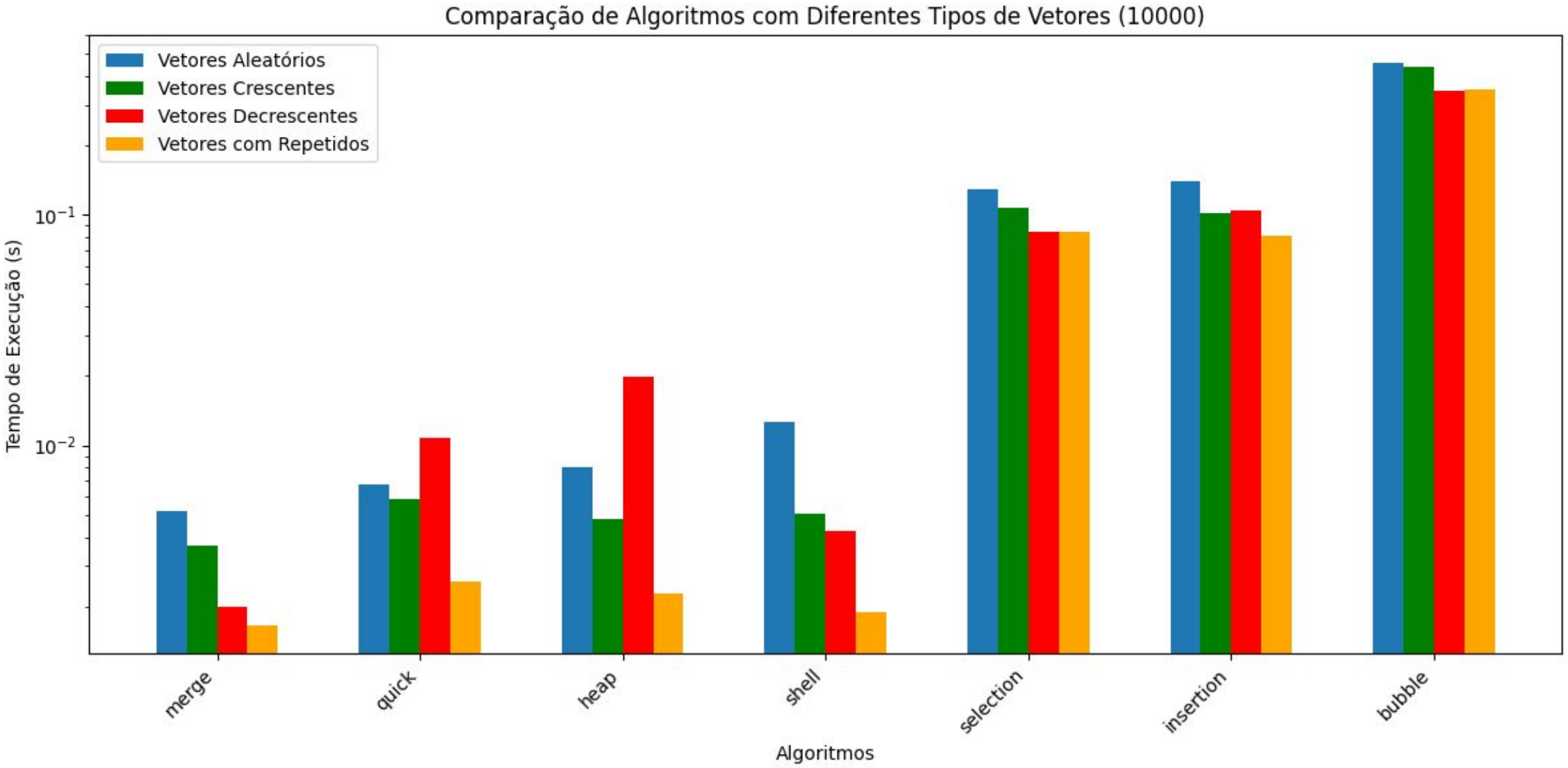


Resultados – Dispersão e tendência

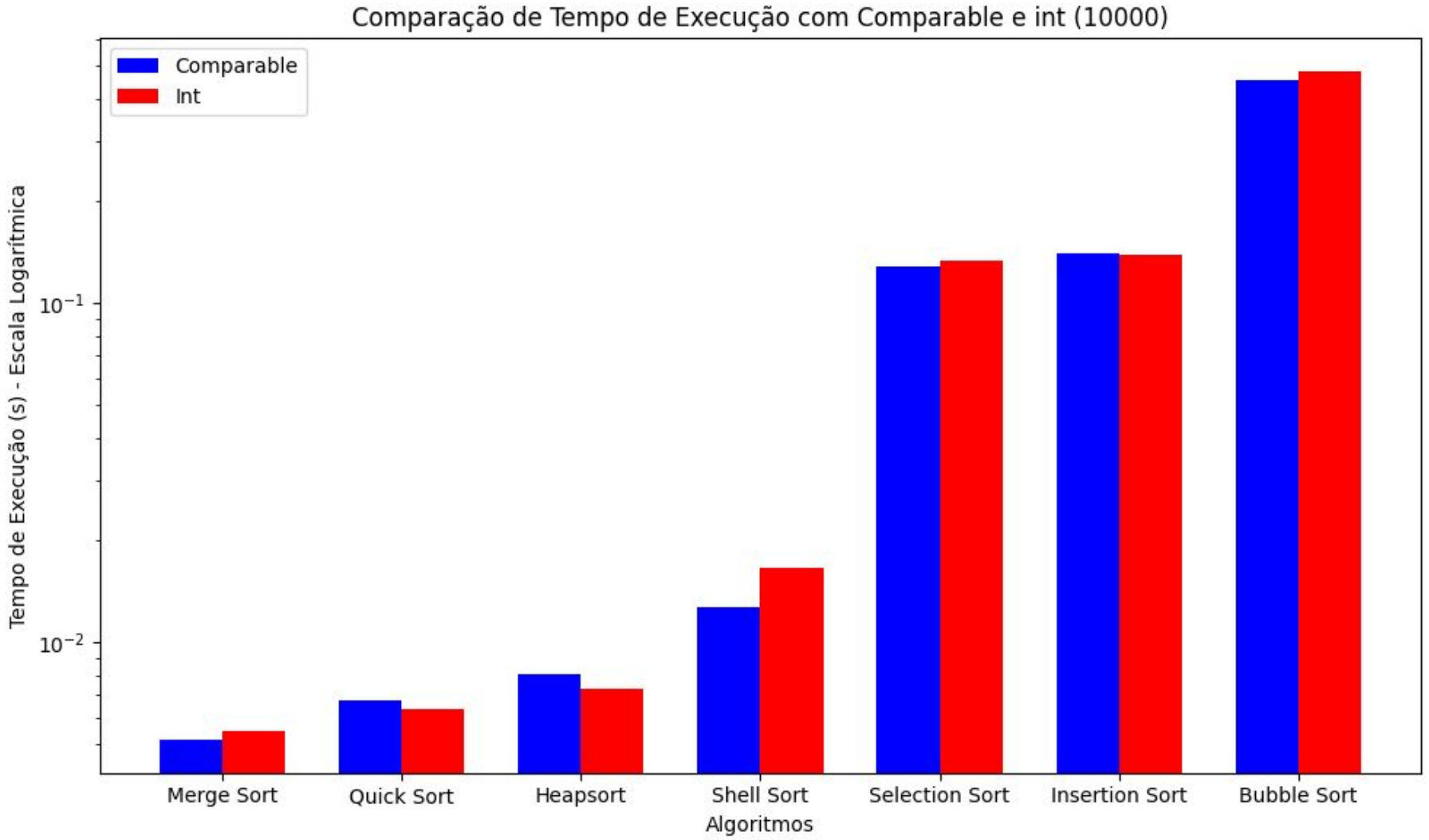




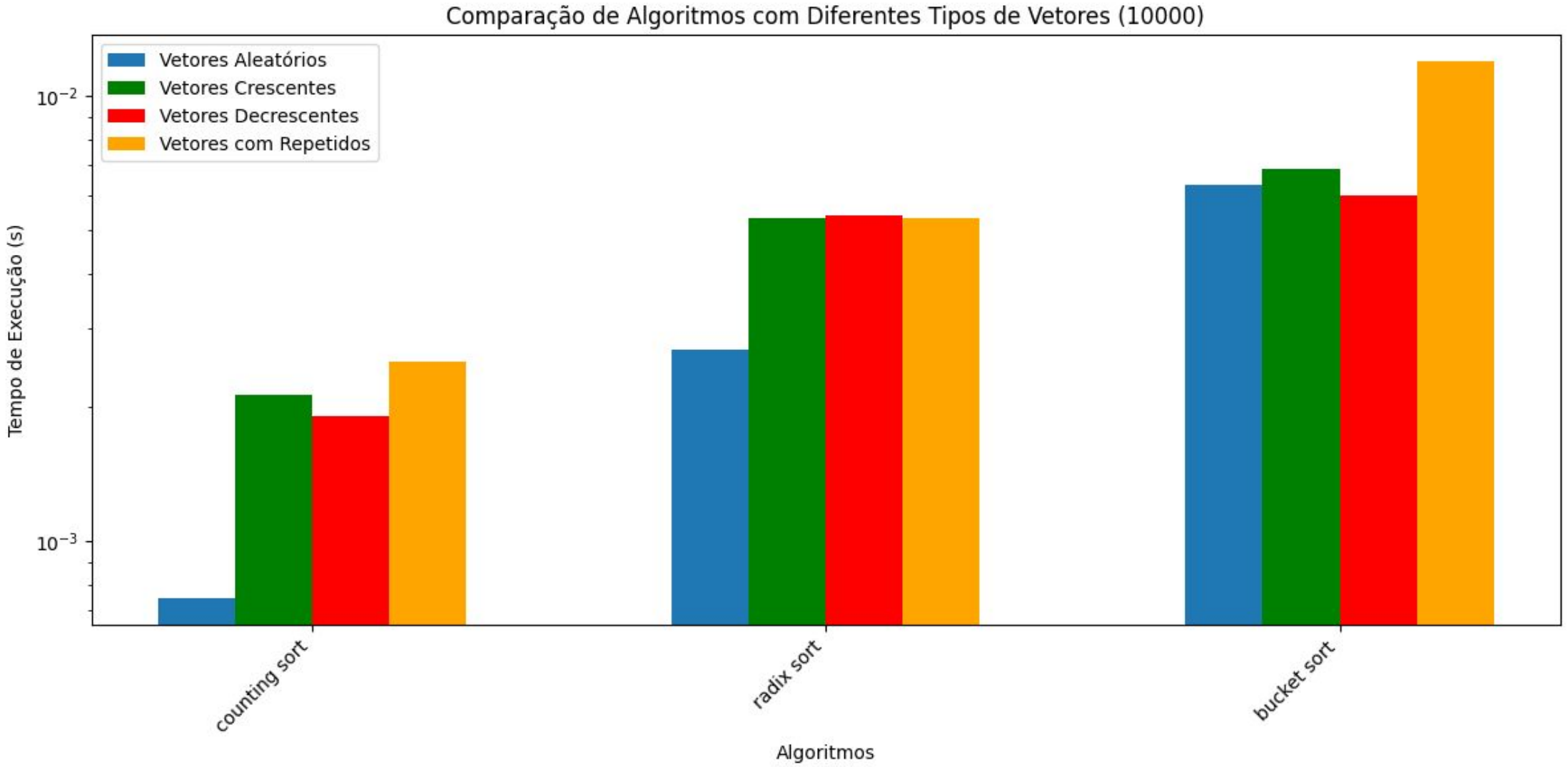
Resultados – Uso de diferentes configurações para vetor



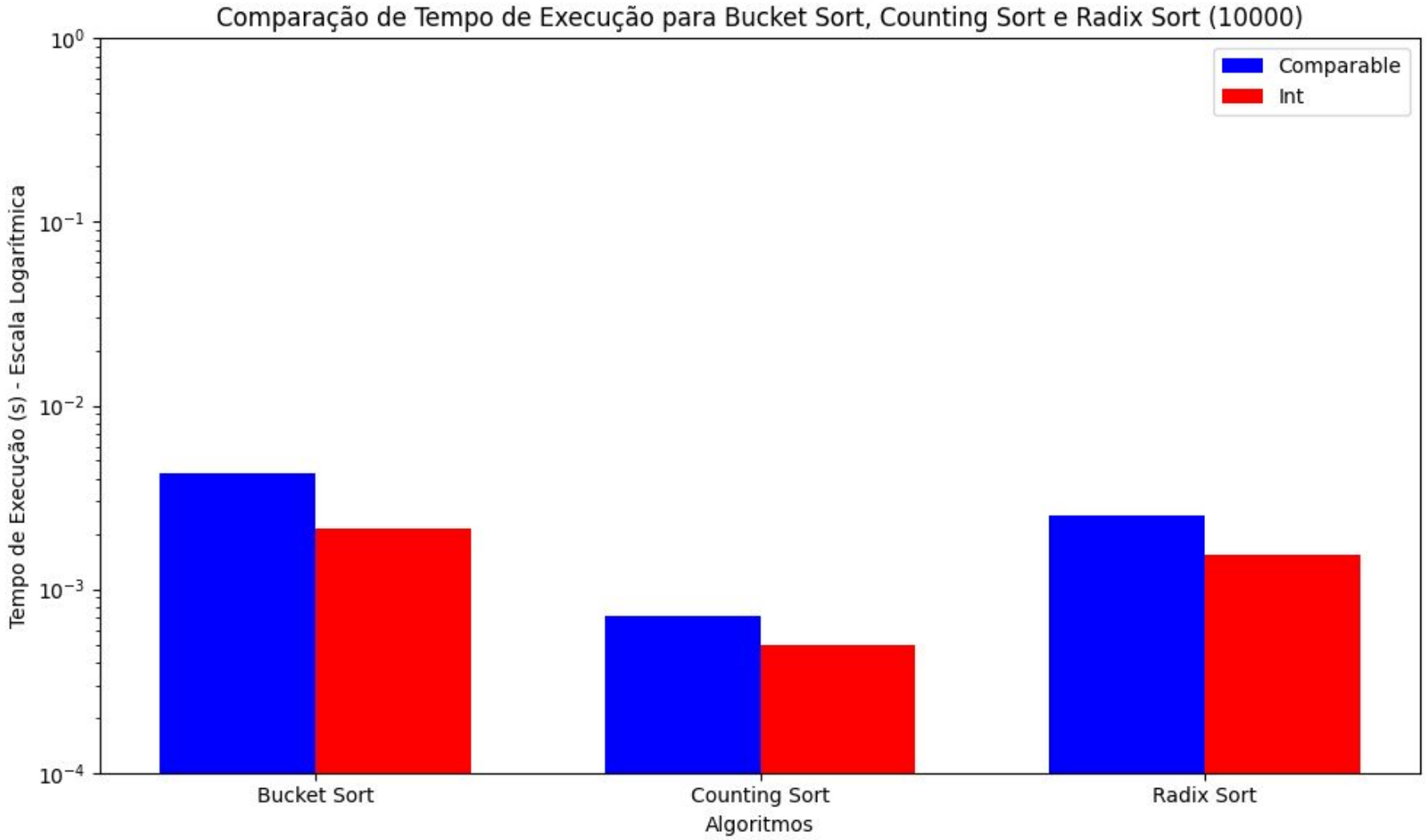
Resultados – Impactos pelo tipo de dado



Resultados – Análise dos algoritmos lineares



Resultados – Impactos pelo tipo de dado



Conclusão

- Algoritmos quadráticos tiveram tempos de execução maiores que os logarítmicos, sendo o Bubble Sort o mais lento, possivelmente devido ao alto número de trocas e iterações.

- Algoritmos logarítmicos demonstraram maior desempenho em relação aos quadráticos conforme o crescimento das entradas.

- Em relação à configuração do vetor inicial, foi observado que os algoritmos logarítmicos foram mais sensíveis à diferentes configurações em relação aos quadráticos.

- Algoritmos lineares se mostraram bastantes sensíveis à configuração inicial do vetor, pois eles trabalham com melhor desempenho em vetores com valores bem distribuídos.

- Os tipos de dados a serem ordenados não demonstraram nenhum impacto a ser considerado.