Relatório de Análise de Algoritmos de Ordenação

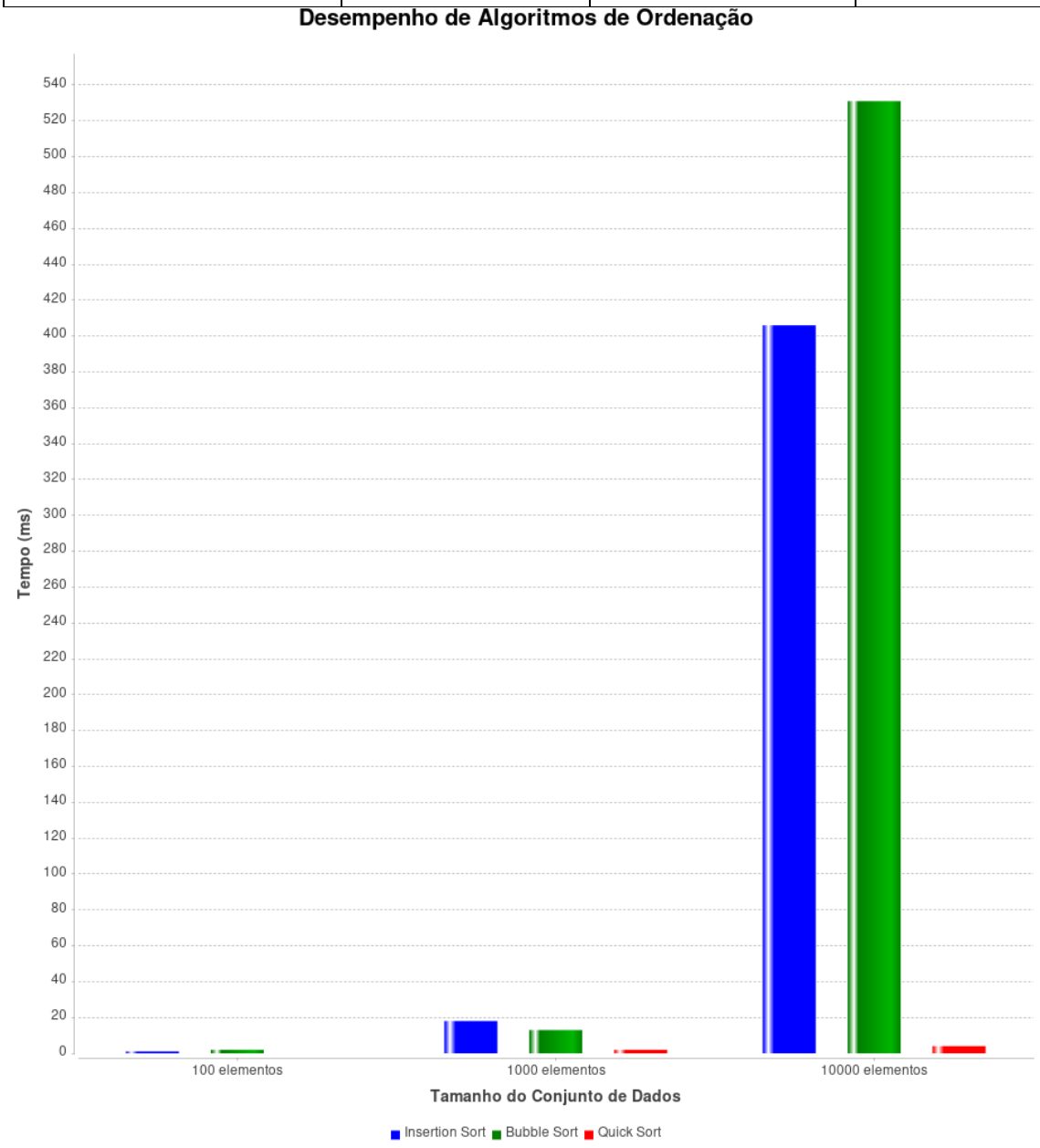
Alunos: Luis Otavio, Rafael Chicovis

Este relatório apresenta uma análise detalhada do desempenho dos algoritmos de ordenação Bubble Sort, Insertion Sort e Quick Sort, aplicados a três tipos distintos de conjuntos de dados: aleatório, ordenado em ordem crescente e ordenado em ordem decrescente.

Os tempos de execução foram medidos em milissegundos (ms) e representam dados reais obtidos por meio de experimentos práticos que serão evidenciados no decorrer do documento. Esses resultados permitem uma comparação direta entre os algoritmos, evidenciando suas características e desempenho em diferentes cenários.

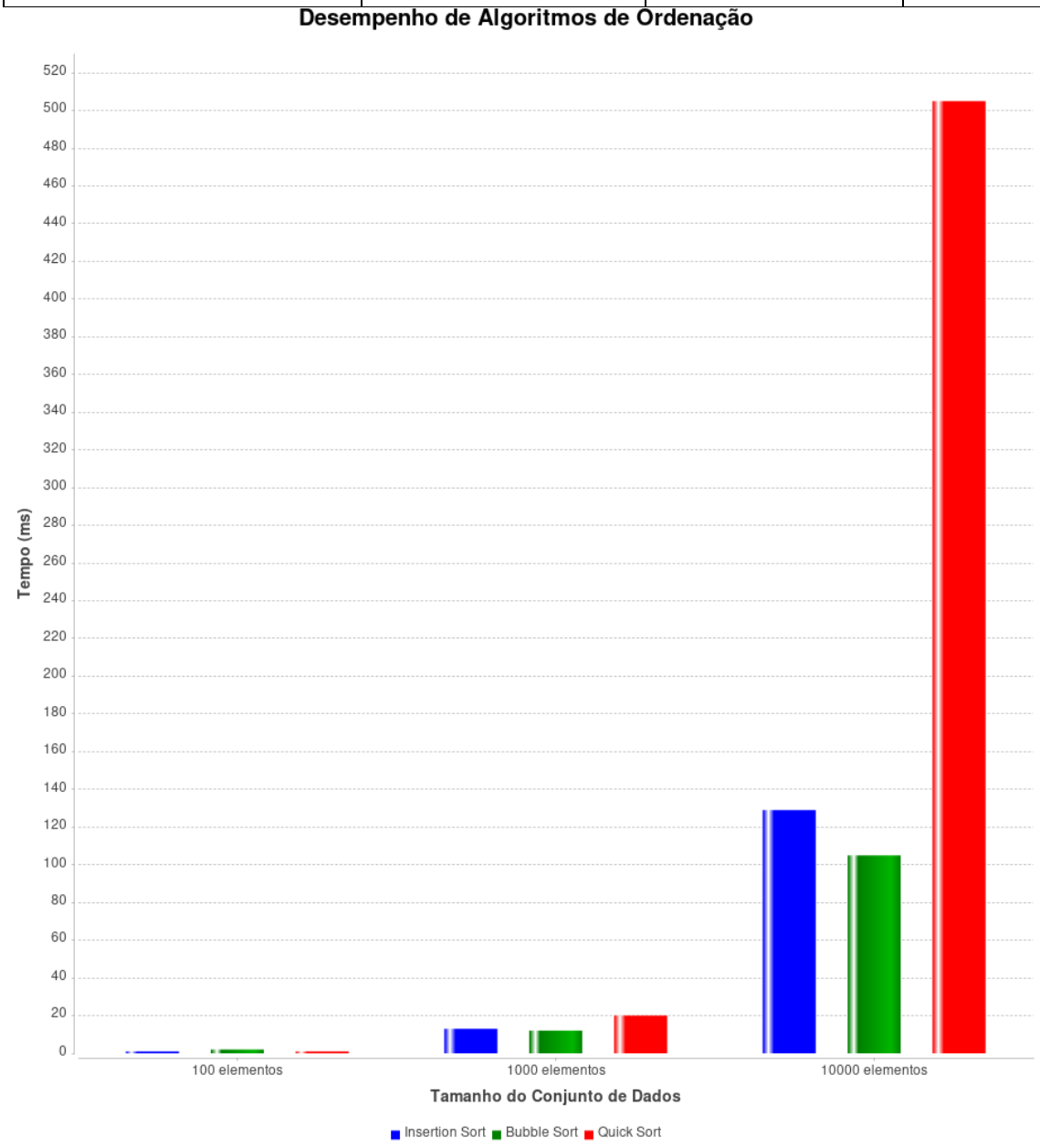
Dados em ordem aleatória:

N de dados/ Sort	Insertion (ms)	Bubble (ms)	Quick (ms)
100	1	2	0
1000	18	13	2
10000	406	531	4



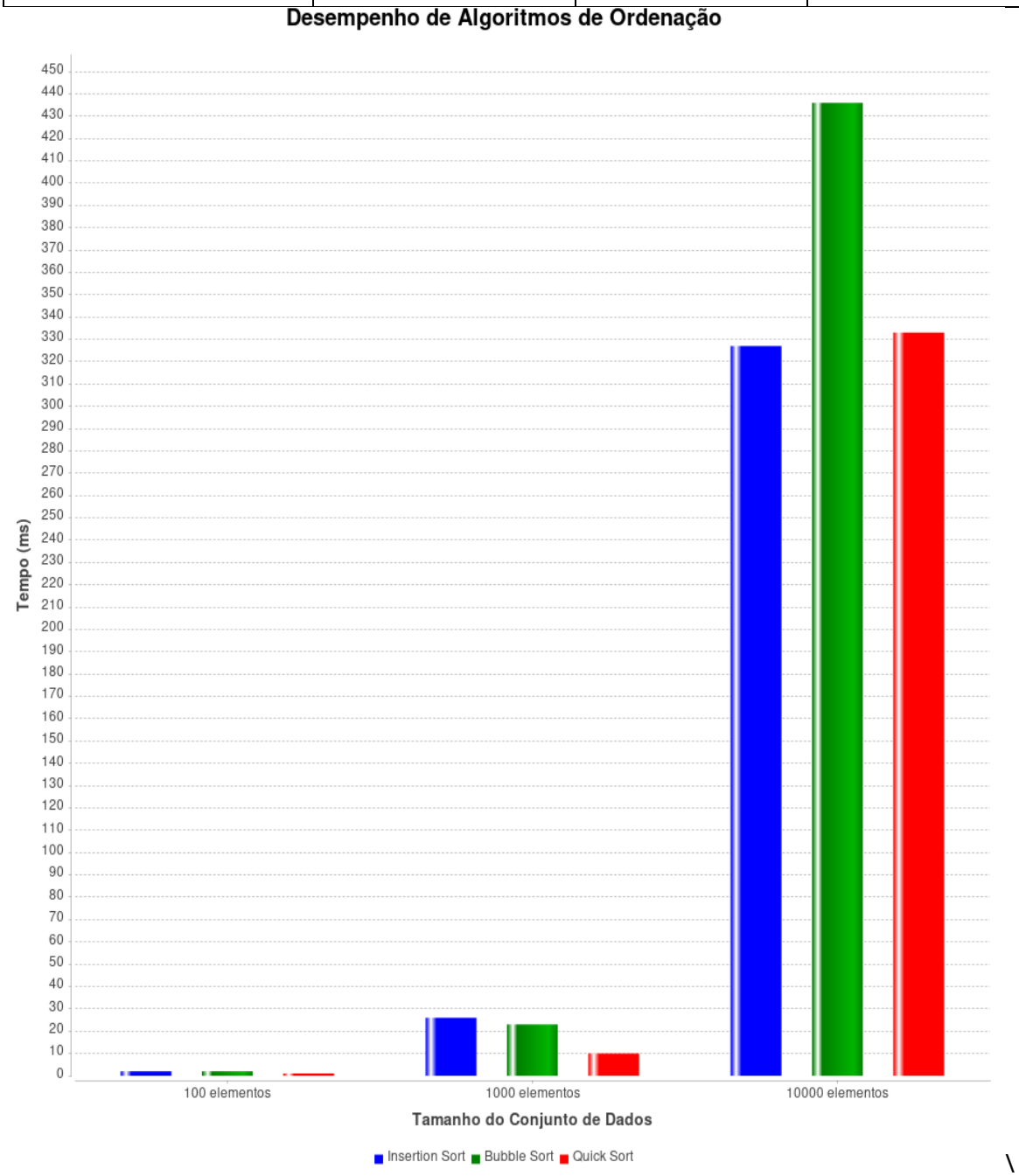
Dados em ordem crescente:

N de dados/ Sort	Insertion (ms)	Bubble (ms)	Quick (ms)
100	1	2	1
1000	20	12	20
10000	129	105	505



Dados em ordem decrescente:

N de dados/ Sort	Insertion (ms)	Bubble (ms)	Quick (ms)
100	2	2	1
1000	26	23	10
10000	327	436	333



\

Com base nos resultados que obtivemos nos testes, dá para tirar algumas lições importantes sobre o comportamento dos algoritmos de ordenação Bubble Sort, Insertion Sort e Quick Sort em diferentes tipos de conjuntos de dados.

Insertion Sort:

O Insertion Sort se saiu bem com listas pequenas (100 elementos), independentemente de estarem ordenadas ou não. Mas, à medida que o tamanho dos dados aumentou, o tempo de execução disparou, principalmente em listas em ordem decrescente.
Quando as listas estavam em ordem crescente, o Insertion Sort foi mais eficiente, o que faz sentido, já que ele é conhecido por funcionar bem em listas que já estão quase ordenadas.

Bubble Sort:

O Bubble Sort teve uma performance aceitável em listas pequenas, mas em listas grandes o tempo de execução foi muito maior, mostrando que ele não é uma boa opção para grandes volumes de dados.
Em listas que já estavam em ordem crescente, o Bubble Sort se saiu melhor, provavelmente porque ele consegue "perceber" quando a lista já está ordenada e termina mais cedo.

Quick Sort:

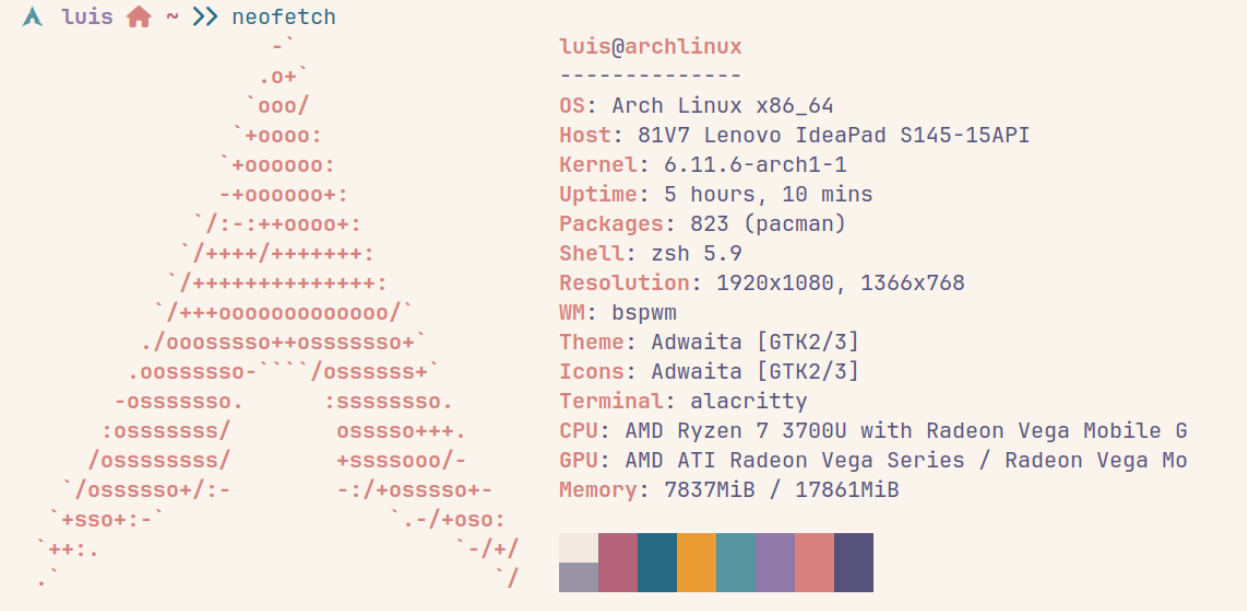
O Quick Sort foi o campeão em listas de dados aleatórios, com tempos de execução muito baixos mesmo em listas grandes. Porém, em listas já ordenadas (crescente ou decrescente), ele mostrou alguma variação no tempo de execução. Isso acontece porque o desempenho do Quick Sort depende de como o pivô é escolhido, podendo cair para $O(n^2)$ em casos ruins.
Apesar disso, o Quick Sort continuou sendo a melhor escolha em termos de eficiência, especialmente em dados aleatórios.

Outras Observações:
Em listas pequenas (100 elementos), todos os algoritmos foram mais ou menos eficientes, e a diferença de desempenho entre eles não foi tão grande.
Em listas médias (1000 elementos), as diferenças ficaram mais evidentes, com o Quick Sort mantendo a liderança.
Em listas grandes (10000 elementos), o Quick Sort brilhou, enquanto o Insertion Sort e o Bubble Sort ficaram bem atrás, especialmente com dados em ordem decrescente.

Dados complementares:

Informações complementares:

Máquina onde foi executado os algoritmos:



(Meu processador e ruim, talvez por isso os tempos de execução altos, :p)