UNIVERSIDADE ESTADUAL DE MATO GROSSO DO SUL UNIDADE UNIVERSITÁRIA DE DOURADOS
CURSO DE BACHARELADO DE CIÊNCIA DA COMPUTAÇÃO

IGOR MONTEIRO NUNES

ALGORITMOS DE ORDENAÇÃO

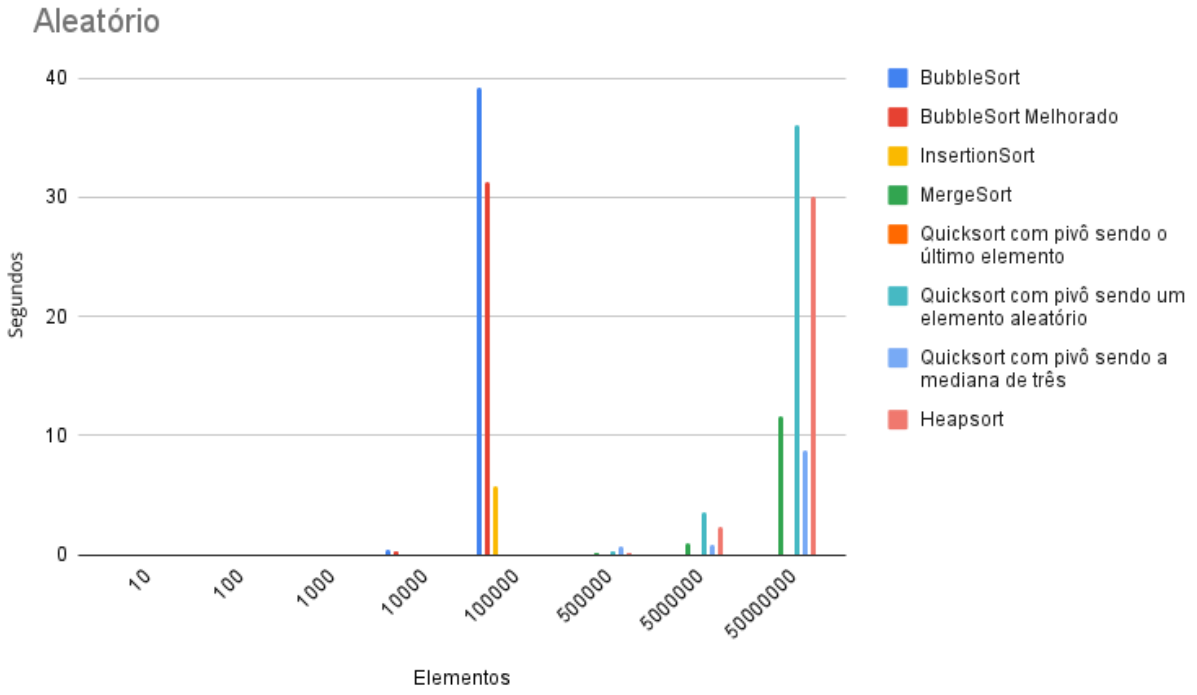
Dourados,
MS 2024

INTRODUÇÃO

Este relatório apresenta a análise de desempenho de diversos algoritmos de ordenação, incluindo Bubble-sort (nas versões original e melhorada), Insertion-sort, Mergesort, Heapsort, e três variações do Quicksort, utilizando diferentes estratégias para a escolha do pivô: último elemento, elemento aleatório e mediana de três. Foram realizados três testes para cada algoritmo de acordo com a quantidade de elementos fornecidos e foram selecionados os melhores tempos de execução entre os três, com o objetivo de avaliar seu comportamento em diferentes situações. Os experimentos foram conduzidos na máquina L4M10, permitindo uma comparação consistente entre as implementações.

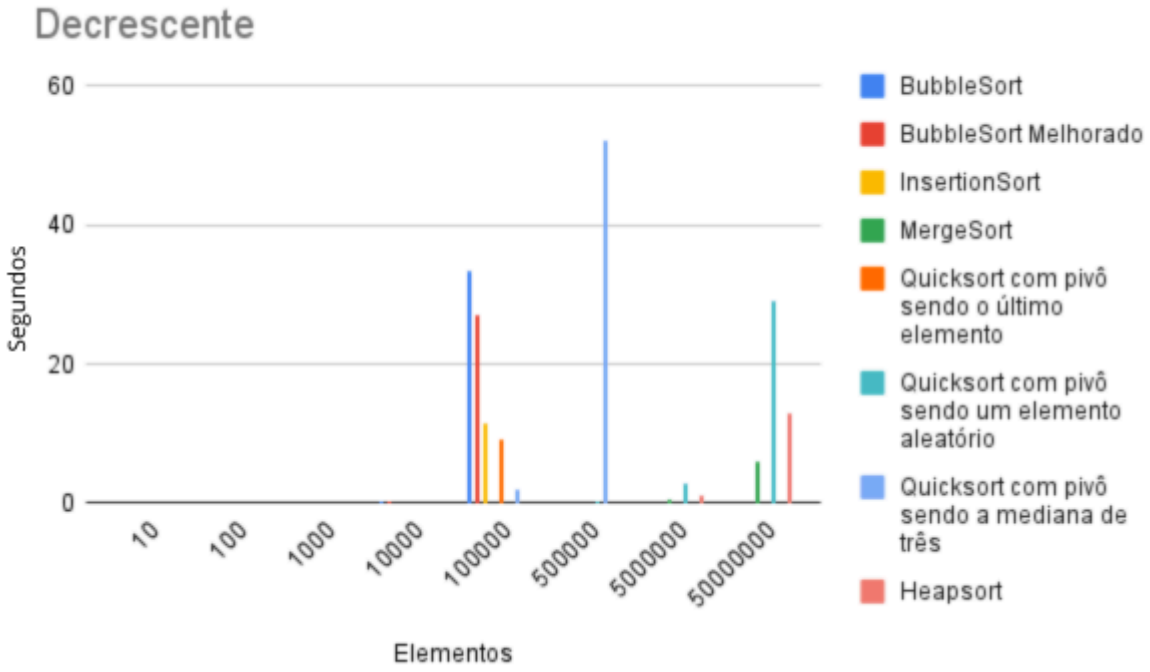
ALEATÓRIO

Elementos	BubbleSort	BubbleSort Melh	InsertionSort	MergeSort	Quicksort com p	Quicksort com p	Quicksort com p	Heapsort
10	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0
1000	0,002	0	0	0	0,001	0,003	0	0,001
10000	0,189	0	0	0,003	0,088	0,007	0,001	0,008
100000	19,046	0	0,001	0,011	8,669	0,061	0,007	0,019
500000				0,046		0,282	0,023	0,108
5000000				0,527		2,981	0,228	1,172
50000000				5,923		29,731	2,299	13,018



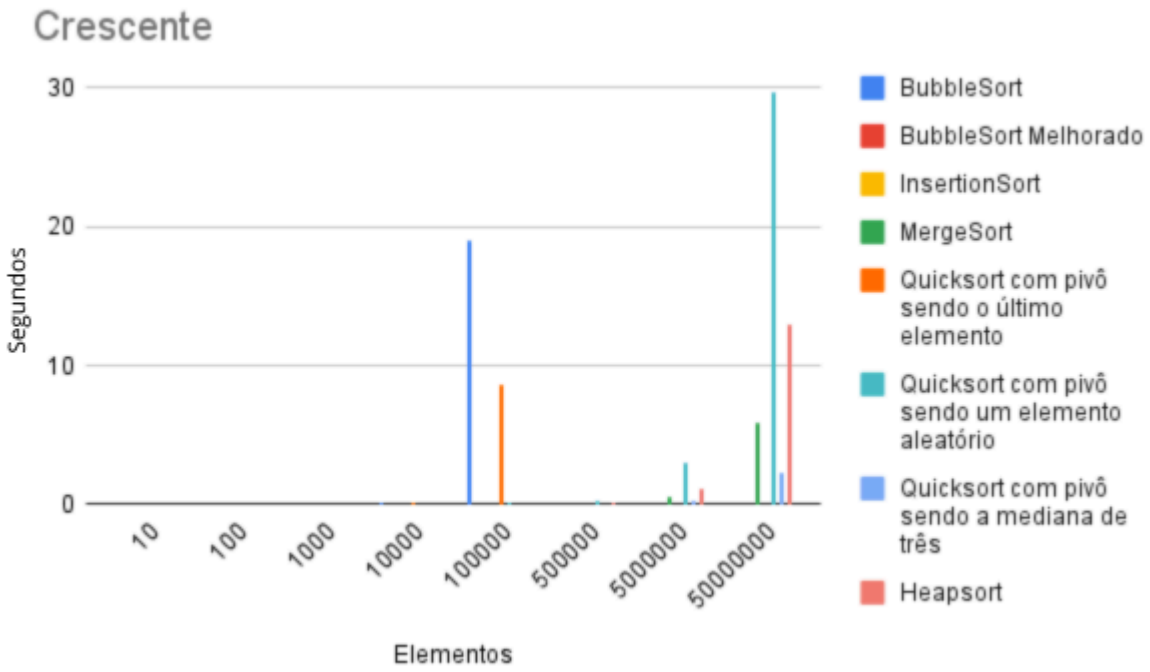
DECRESCENTE

Elementos	BubbleSort	BubbleSort Melh	InsertionSort	MergeSort	Quicksort com p	Quicksort com p	Quicksort com p	Heapsort
10	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0
1000	0,004	0,004	0,006	0	0,003	0,002	0,001	0
10000	0,329	0,27	0,114	0,001	0,095	0,008	0,031	0,003
100000	33,451	27,097	11,615	0,011	9,345	0,055	2,091	0,019
500000				0,047		0,287	52,161	0,102
5000000				0,547		2,815	Indefined	1,171
50000000				6,08		29,033	Indefined	12,886



CRESCENTE

Elementos	BubbleSort	BubbleSort Melh	InsertionSort	MergeSort	Quicksort com p	Quicksort com p	Quicksort com p	Heapsort
10	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0
1000	0,002	0	0	0	0,001	0,003	0	0,001
10000	0,189	0	0	0,003	0,088	0,007	0,001	0,008
100000	19,046	0	0,001	0,011	8,669	0,061	0,007	0,019
500000				0,046		0,282	0,023	0,108
5000000				0,527		2,981	0,228	1,172
50000000				5,923		29,731	2,299	13,018



DISCUSSÃO ACERCA DO ASSUNTO

Nos casos com poucas entradas (10, 100 e 1000 elementos), todos os algoritmos mostraram tempos insignificantes. Conforme o número de elementos aumenta, a disparidade no desempenho dos algoritmos torna-se mais evidente.

O BubbleSort original e o BubbleSort Melhorado apresentaram tempos de execução cada vez maiores conforme o número de elementos aumenta, destacando-se como algoritmos ineficientes para grandes volumes de dados. Aos 100.000 elementos, o BubbleSort original demorou mais de 19 segundos, e o melhorado teve tempos elevados também.

O InsertionSort, que é eficiente para entradas menores, mostrou boa performance até 10.000 elementos, mas apresentou tempos altos a partir de 100.000, chegando a mais de 11 segundos.

O MergeSort e o Heapsort mantiveram tempos mais baixos, mesmo com grandes volumes de dados, mostrando boa escalabilidade.

O Quicksort com pivô sendo o último elemento e Quicksort com pivô aleatório apresentaram boa performance até 5 milhões de elementos. No entanto, o Quicksort com pivô sendo a mediana de três enfrentou um problema de performance nos testes com 5 milhões de elementos, onde o tempo de execução ultrapassou o limite de tempo razoável, resultando em um status "Indefinido".

CONCLUSÃO

Com base nos testes realizados, foi possível observar comportamentos distintos entre os algoritmos de ordenação, especialmente em relação à eficiência e escalabilidade conforme o tamanho das entradas aumentava. Algoritmos simples, como o BubbleSort e o InsertionSort, demonstraram limitações evidentes com grandes volumes de dados, sendo inadequados para entradas acima de 10.000 elementos. Por outro lado, algoritmos mais avançados como MergeSort, Heapsort, e as variações do Quicksort provaram ser mais eficientes para grandes entradas, com destaque para o Quicksort com pivô aleatório e o MergeSort, que mantiveram tempos de execução baixos até 50 milhões de elementos

De maneira geral, os resultados mostram que a escolha do algoritmo de ordenação deve ser feita com base no volume de dados e na estrutura do problema, com algoritmos como MergeSort e Heapsort sendo indicados para grandes volumes, enquanto os mais simples podem ser usados eficientemente em casos menores.

.