Relatório de Análise de Algoritmos de Ordenação

Alunos: Felipe Kureski e Tiago Bisolo Prestes

Introdução

Este relatório compara o desempenho dos algoritmos de ordenação Bubble Sort, Insertion Sort e Quick Sort em três cenários de dados: aleatório, crescente e decrescente. Os tempos de execução estão em nanosegundos (ns) e visam ilustrar o comportamento relativo de cada algoritmo.

Arquivos aleatórios

Nome do arquivo	Insertion Sort	Bubble Sort	Quick Sort
aleatorio_100.csv	410834 ns	6667 ns	421000 ns
aleatorio_1000.csv	5559458 ns	55917 ns	10322750 ns
aleatorio_1000.csv	44377292 ns	442208 ns	147846125 ns

Arquivos crescentes:

Nome do arquivo	Insertion Sort	Bubble Sort	Quick Sort
crescente _100.csv	20875 ns	5291 ns	13750 ns
crescente _1000.csv	80959 ns	47125 ns	1217833 ns
crescente _1000.csv	821083 ns	465417 ns	134837042 ns

Arquivos decrescentes:

Nome do arquivo	Insertion Sort	Bubble Sort	Quick Sort
decrescente _100.csv	25583 ns	6666 ns	17291 ns
decrescente _1000.csv	677333 ns	43792 ns	1128209 ns
decrescente _1000.csv	70623208 ns	471125 ns	122080333 ns

Análise de Desempenho

Cenário 1: Arquivos Aleatórios

Neste cenário, que representa um caso de uso geral, o Bubble Sort apresentou os melhores tempos de execução para todos os tamanhos de arquivo. Ele foi consistentemente o mais rápido, com um tempo de 442.208 ns para 10.000 elementos, enquanto o Insertion Sort levou 44.377.292 ns e o Quick Sort, 147.846.125 ns.

Cenário 2: Arquivos Crescentes

Para dados já ordenados, o Bubble Sort novamente se destacou como o algoritmo mais eficiente, seguido de perto pelo Insertion Sort. O Bubble Sort registrou o menor tempo (465.417 ns para 10.000 elementos). O Insertion Sort também foi muito rápido (821.083 ns), o que é esperado, pois ambos os algoritmos seguem um desempenho linear neste cenário.

O Quick Sort demonstrou um desemepenho ruim (134.837.042 ns), possivelmente devido a implementação de pivô fixo (aqui no último elemento) em dados já ordenados, o que aumenta a complexidade/tempo de execução.

Cenário 3: Arquivos Decrescentes

Neste cenário, que representa o pior caso teórico para Bubble Sort e Insertion Sort, os dados indicam que o Bubble Sort, mais uma vez, foi o algoritmo mais rápido. O Bubble Sort manteve um tempo de execução baixo (471.125 ns para 10.000 elementos), o que, novamente, é um resultado inesperado para seu pior cenário.

Conforme o esperado, o Insertion Sort teve um péssimo desempenho e foi o mais lento de todos (70.623.208 ns). O Quick Sort também foi lento, mas superou o Insertion Sort neste caso específico.

Conclusão

Com base nos dados, o Bubble Sort foi considerado o algoritmo mais eficiente no contexto geral. O Insertion Sort é altamente eficaz para dados crescentes (já ordenados). Por fim, o Quick Sort, apesar de ter um desempenho fraco em dados crescentes e decrescentes neste relatório, teoricamente seria o mais indicado para os dados aleatórios de grande volume.