Relatório - Aplicação de Algoritmos de Ordenação em C

O objetivo desse trabalho foi implementar e avaliar o desempenho de diversos algoritmos de ordenação usando a linguagem C. Seis algoritmos clássicos foram implementados: bubble sort, selection sort, insertion sort, quick sort, merge sort e shell sort, o código ficou mais organizado e mais fácil de ler e manter, pois cada um deles foi implementado em uma função distinta, além das funções de ordenação, foi adicionada uma função auxiliar denominada printArray() para mostrar os elementos do vetor antes e depois da ordenação. Uma função para copiar o vetor original também foi usada, garantindo que todos os algoritmos fossem testados com o mesmo conjunto de dados, isso foi muito bom para garantir uma comparação justa de desempenho, a função clock() da biblioteca foi usada para medir o tempo de execução, permitindo calcular quanto tempo cada algoritmo levou para ordenar o vetor.
Durante os testes, os algoritmos foram executados com vetores de tamanhos variados, preenchidos com números criados aleatoriamente. Nos vetores pequenos, a variação de tempo entre os métodos não foi tão clara, no entanto, à medida que o tamanho do vetor aumentava, ficou claro que certos algoritmos levavam muito mais tempo do que outros, pouca coisa, os algoritmos bubble sort, selection sort e insertion sort mostraram um aumento grande no tempo de execução ao serem aplicados em vetores de maior tamanho. Isso acontece porque possuem complexidade quadrática, ou seja, o número de operações cresce rapidamente conforme o tamanho do vetor aumenta, por outro lado, o quick sort e o merge sort apresentaram um desempenho significativamente melhor nos testes com grandes volumes de dados, mantendo tempos muito menores em relação aos outros. Isso acontece porque esses algoritmos têm uma complexidade média de O(n log n), o que os torna mais eficientes para grandes volumes de dados, o shell sort teve um desempenho intermediário, sendo mais veloz que os algoritmos mais simples, porém, na maioria das vezes, não superando o quick sort e o merge sort.

De modo geral, a atividade permitiu observar na prática aquilo que é estudado na teoria sobre complexidade de algoritmos, ficou claro que a escolha do método de ordenação influencia diretamente no desempenho do programa, principalmente quando se trabalha com grandes quantidades de dados, a experiência ajudou a compreender melhor como cada algoritmo se comporta e reforçou a importância de escolher a estratégia adequada para cada situação