RELATÓRIO – IMPLEMENTAÇÃO E ANÁLISE DE ALGORITMOS DE ORDENAÇÃO EM C

1. Introdução

Este trabalho teve como objetivo implementar e analisar cinco algoritmos de ordenação na linguagem C: Bubble Sort, Selection Sort, Insertion Sort, Quick Sort e Merge Sort.

Foram realizados testes utilizando diferentes tamanhos e estados de arrays para comparar o desempenho entre os algoritmos.

---

2. Implementação dos Algoritmos

Todos os algoritmos foram implementados em funções separadas dentro do arquivo `sorting_algorithms.c`.

Também foi criada uma função auxiliar chamada `printArray()` para exibir os elementos antes e depois da ordenação.

Para medir o desempenho, foi utilizada a função `clock()` da biblioteca `<time.h>`, permitindo calcular o tempo de execução de cada algoritmo.

Resumo das implementações:

- **Bubble Sort:** utiliza dois laços de repetição para comparar elementos adjacentes e trocá-los quando necessário. Complexidade $O(n^2)$.
- **Selection Sort:** seleciona o menor elemento e o posiciona corretamente a cada iteração. Complexidade $O(n^2)$.
- **Insertion Sort:** insere cada elemento na posição correta dentro da parte já ordenada do array. Complexidade $O(n^2)$, mas eficiente para arrays pequenos.
- **Quick Sort:** utiliza o método de divisão e conquista com pivô para organizar os elementos. Complexidade média O(n log n).
- **Merge Sort:** divide o array em partes menores e depois realiza a intercalação ordenada. Complexidade O(n log n).

---

3. Resultados dos Testes

Os algoritmos foram testados com:

- Arrays aleatórios

- Arrays ordenados crescentemente
- Arrays ordenados decrescentemente

Tamanhos utilizados:

- 10 elementos
- 1000 elementos
- 10000 elementos

Observações:

- Para arrays pequenos (10 elementos), todos os algoritmos apresentaram desempenho semelhante.
- Para arrays médios (1000 elementos), Quick Sort e Merge Sort mostraram melhor desempenho.
- Para arrays grandes (10000 elementos), os algoritmos $O(n^2)$ tornaram-se significativamente mais lentos, enquanto Quick Sort e Merge Sort mantiveram bom desempenho.

---

4. Comparação de Desempenho

Os algoritmos podem ser divididos em dois grupos principais:

Algoritmos $O(n^2)$

- Bubble Sort
- Selection Sort
- Insertion Sort

Apresentam crescimento quadrático no tempo de execução, tornando-se lentos para grandes volumes de dados.

Algoritmos O(n log n)

- Quick Sort
- Merge Sort

São mais eficientes e indicados para grandes conjuntos de dados.
O Quick Sort foi o mais rápido na maioria dos testes.
O Merge Sort apresentou desempenho estável, porém utiliza memória adicional.

---

5. Conclusão

A análise prática confirmou a teoria da complexidade dos algoritmos.

Conclui-se que:

- Algoritmos simples são adequados para pequenos conjuntos de dados.
- Para grandes volumes, Quick Sort e Merge Sort são as melhores opções.
- A escolha do algoritmo impacta diretamente no desempenho do programa.

O trabalho permitiu compreender tanto a implementação quanto a eficiência dos principais algoritmos de ordenação.