CP Data Structures and Algorithms

Felipe Souza Carvalho – RM564779

Relatório Comparativo de Algoritmos de Ordenação

1. Metodologia

Os experimentos foram conduzidos executando repetidamente cada algoritmo sobre listas de **1000, 5000 e 10000 elementos**.
Foram considerados quatro cenários distintos de entrada:

- **Sorted list**: lista já ordenada.
- **Half sorted list**: lista parcialmente ordenada.
- **Reverse list**: lista ordenada de forma decrescente.
- **Random list**: lista em ordem aleatória.

Cada execução foi repetida diversas vezes (5) a fim de reduzir variações estatísticas no tempo de execução.
O tempo de execução (em milissegundos) foi utilizado como métrica de comparação.

---

2. Resultados

2.1 Bubble Sort

- **Desempenho geral**: apresentou os maiores tempos médios de execução em quase todos os cenários.
- **Ponto positivo**: em listas já ordenadas, o tempo foi reduzido em comparação aos outros casos, mostrando a vantagem da otimização de verificação de trocas.
- **Limitação**: cresce quadraticamente ($O(n^2)$), tornando-se impraticável para listas grandes.

2.2 Insertion Sort

- **Desempenho geral**: mostrou-se extremamente eficiente em listas já ordenadas e quase ordenadas.
- **Cenários adversos**: em listas inversamente ordenadas ou aleatórias, apresentou desempenho quadrático, embora melhor que o Bubble Sort.

2.3 QuickSort

- **Desempenho geral**: foi o mais rápido na maioria dos casos, principalmente em listas aleatórias.

- **Cenários desafiadores**: em listas já ordenadas ou reversas, o tempo de execução aumentou significativamente, indicando influência da escolha do pivô.
- **Relevância prática**: considerado o mais eficiente entre os testados, amplamente utilizado em bibliotecas padrão de linguagens de programação.

---

3. Resumo dos resultados

A comparação evidencia que:

- **Bubble Sort** é útil apenas em contextos didáticos.
- **Insertion Sort** é ótimo em listas pequenas ou quase ordenadas, podendo superar até o QuickSort nesses casos.
- **QuickSort** é, em média, o mais eficiente, mas depende de boas escolhas do pivô para evitar o pior caso.

Assim, a escolha do algoritmo deve considerar tanto o **tamanho da entrada** quanto o **grau de ordenação inicial**.

---

4. Conclusão

Os experimentos confirmam a teoria sobre complexidade de algoritmos de ordenação:

- **Bubble Sort** é o menos eficiente, mas fácil de entender.
- **Insertion Sort** é excelente em listas pequenas ou quase ordenadas.
- **QuickSort** apresenta o melhor equilíbrio entre simplicidade e desempenho, sendo o mais indicado para listas grandes e desordenadas.