Relatório

**Algoritmos de ordenação**

Ícaro Kuchanovicz

Lucas Utrabo

**Bubble Sort**

O Bubble Sort organiza a lista passando várias vezes por ela e trocando de lugar os números que estão fora de ordem até que tudo fique em sequência.

|  |  |
| --- | --- |
| **ARQUIVO SELECIONADO** | **TEMPO DE EXECUÇÃO** |
| aleatorio\_100.csv | 14217900 nanosegundos |
| aleatorio\_1000.csv | 64980800 nanosegundos |
| aleatorio\_10000.csv | 744615400 nanosegundos |
| crescente\_100.csv | 5294100 nanosegundos |
| crescente\_1000.csv | 41851700 nanosegundos |
| crescente\_10000.csv | 220030800 nanosegundos |
| decrescente\_100.csv | 7559000 nanosegundos |
| decrescente\_1000.csv | 75243800 nanosegundos |
| decrescente\_10000.csv | 777044400 nanosegundos |

O Bubble Sort foi o mais lento em quase todos os cenários, especialmente em listas grandes e desordenadas (como decrescente\_10000.csv). Ele troca elementos repetidamente, o que torna o algoritmo ineficiente quando há muitos dados ou a lista está em ordem decrescente.

* **Bom para:** listas pequenas e quase ordenadas.
* **Ruim para:** listas grandes ou totalmente desordenadas.

**Insertion Sort**

O Insertion Sort organiza a lista pegando cada número e colocando-o no lugar correto entre os já ordenados, como se estivesse organizando cartas na mão.

|  |  |
| --- | --- |
| **ARQUIVO SELECIONADO** | **TEMPO DE EXECUÇÃO** |
| aleatorio\_100.csv | 5676200 nanosegundos |
| aleatorio\_1000.csv | 51525900 nanosegundos |
| aleatorio\_10000.csv | 325564900 nanosegundos |
| crescente\_100.csv | 3642600 nanosegundos |
| crescente\_1000.csv | 4064400 nanosegundos |
| crescente\_10000.csv | 121829100 nanosegundos |
| decrescente\_100.csv | 6254500 nanosegundos |
| decrescente\_1000.csv | 58146400 nanosegundos |
| decrescente\_10000.csv | 371802500 nanosegundos |

O Insertion Sort teve desempenho razoável, principalmente em listas pequenas ou já ordenadas. Ele é rápido para organizar listas que já estão quase em ordem, mas fica lento com listas grandes e desordenadas.

* **Bom para:** listas pequenas e ordenadas.
* **Ruim para:** listas grandes e desordenadas.

**Quick Sort**

O Quick Sort organiza a lista escolhendo um número como referência (pivô), separando os menores à esquerda e os maiores à direita, e repetindo esse processo em cada parte até que toda a lista esteja em ordem.

|  |  |
| --- | --- |
| **ARQUIVO SELECIONADO** | **TEMPO DE EXECUÇÃO** |
| aleatorio\_100.csv | 3442700 nanosegundos |
| aleatorio\_1000.csv | 9441800 nanosegundos |
| aleatorio\_10000.csv | 28114800 nanosegundos |
| crescente\_100.csv | 8177000 nanosegundos |
| crescente\_1000.csv | 4064400 nanosegundos |
| crescente\_10000.csv | 2613926800 nanosegundos |
| decrescente\_100.csv | 5073700 nanosegundos |
| decrescente\_1000.csv | 74863800 nanosegundos |
| decrescente\_10000.csv | 1503692000 nanosegundos |

O Quick Sort foi o mais rápido em quase todos os casos, principalmente em listas grandes e desordenadas (como aleatorio\_10000.csv). Ele organiza a lista dividindo-a em partes menores, o que o torna mais eficiente. Em listas grandes e decrescentes, o tempo foi maior, mas ainda melhor que os outros.

* **Bom para:** listas grandes e desordenadas.
* **Ruim para:** listas grandes e decrescentes, mas ainda é o mais eficiente.