**Relatório**

**1º Algoritmo Hibrido:**

* **Descrição:** O algoritmo implementado utiliza o BucketSort como principal método de ordenação. O BucketSort divide o conjunto de dados em baldes e ordena cada balde separadamente, antes de concatená-los. No entanto, para lidar com o caso em que um balde contém muitos elementos, foi introduzido um segundo método de ordenação: o Quicksort. Quando um balde excede um determinado tamanho limite, definido pelo usuário, o Quicksort é utilizado para ordenar os elementos desse balde. Isso proporciona uma maneira eficiente de lidar com situações em que o BucketSort pode ter desempenho inferior devido à distribuição desigual dos elementos nos baldes.
* **Justificativa da escolha:**
* **BucketSort:** Escolhido como algoritmo principal devido à sua eficiência para distribuir os elementos em baldes com base em seus valores. Isso facilita a ordenação subsequente de cada balde.
* **QuickSort:** Selecionado como algoritmo secundário devido à sua rapidez e eficiência em ordenar pequenas partições de dados. É especialmente útil quando um balde do BucketSort contém muitos elementos, garantindo um desempenho satisfatório em todas as situações.
* **Comparação de Desempenho:**
* **Ordenação Interna:** Os testes de desempenho para ordenação interna foram realizados com diferentes tamanhos de array e tamanhos de balde. Os resultados são apresentados a seguir:

|  |  |  |
| --- | --- | --- |
| **Tamanho do Array** | **Tamanho do Balde** | **Desempenho(segundos)** |
| 100000 | 5 | 0.010 |
| 100000 | 100 | 0.009 |
| 1000000 | 5 | 0.0093 |
| 1000000 | 500 | 0.102 |
| 1000000 | 2 | 0.140 |

Os testes demonstram que o desempenho da ordenação interna varia significativamente com o tamanho do array e o tamanho do balde. Para arrays menores e tamanhos de balde maiores, o desempenho tende a ser mais rápido. Por outro lado, para arrays maiores e tamanhos de balde menores, o desempenho pode ser mais lento devido à sobrecarga de gerenciamento de baldes.

* **Ordenação Externa:**