**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)** |
| 1000 | 1 | 0.001 |
| 100000 | 5 | 0.020 |
| 100000 | 100 | 0.024 |
| 1000000 | 2 | 0.193 |
| 1000000 | 5 | 0.239 |
| 1000000 | 500 | 0.199 |
| 2000000 | 12 | 0.728 |
| 20000000 | 10000 | 76.513 |

* **Análise de Desempenho:**

Os testes revelam que o desempenho da ordenação interna varia de acordo com o tamanho do array e o tamanho do balde. Observa-se que:

* Para arrays menores (1.000 elementos): O tempo de execução é muito rápido, mesmo com baldes de tamanho pequeno (1 elemento).
* Para arrays médios (100.000 elementos): O desempenho é relativamente estável e rápido. O tempo de execução aumenta ligeiramente com baldes maiores, possivelmente devido à sobrecarga de manipulação de mais elementos dentro de cada balde.
* Para arrays grandes (1.000.000 elementos): O tempo de execução varia mais significativamente com o tamanho do balde. Baldes muito pequenos (2 elementos) aumentam a sobrecarga de gerenciamento de baldes, enquanto baldes maiores (500 elementos) equilibram a carga de trabalho.
* Para arrays muito grandes (2.000.000 a 20.000.000 elementos): O tempo de execução aumenta substancialmente, refletindo a maior quantidade de dados a serem ordenados. Para um array de 20.000.000 elementos com baldes de 10.000 elementos, o tempo de execução foi significativamente maior (76.513 segundos), indicando a necessidade de otimizações adicionais para conjuntos de dados extremamente grandes.

Esses resultados indicam que há um ponto ótimo para o tamanho do balde, onde o algoritmo híbrido funciona de maneira mais eficiente. Esse ponto ótimo depende do tamanho do array e da distribuição dos dados.

* **Ordenação Externa:**