# Análise Experimental de Métodos de Ordenação Externa

Ciro Junio, Lucas Terra, Jonhatan Evangelista, Augusto Luna, Ben-hur Nogueira
*Departamento de Computação*
*Universidade Federal de Ouro Preto*, Mariana, Brasil

***Abstract*—Este trabalho analisa o desempenho de métodos de ordenação externa: intercalação balanceada de vários caminhos (2f fitas), intercalação balanceada de vários caminhos (f+1 fitas) e quicksort externo. Foram implementados algoritmos em C e realizados experimentos com diferentes tamanhos e ordenações de arquivos, avaliando transferências de memória, comparações e tempo de execução. Os resultados permitem comparar a eficiência e aplicabilidade dos métodos em sistemas de grande volume de dados.**

***Index Terms*—Pesquisa externa, desempenho, 2f Fitas, F + 1 Fitask Quicksort externo, algoritmos, análise experimental, sistemas de grande volume de dados**

## I. Ordenação Externa

A ordenação externa é um método utilizado para organizar grandes volumes de dados que não podem ser armazenados completamente na memória principal, sendo necessários acessos frequentes a dispositivos de armazenamento secundário, como discos rígidos. Esse tipo de ordenação é essencial em sistemas que lidam com conjuntos de dados tão grandes que superam a capacidade de memória disponível, requerendo o uso de técnicas específicas para ordenar os dados diretamente a partir de arquivos armazenados externamente.

## II. Ordenação de Vários Caminhos

A ordenação de vários caminhos é uma técnica utilizada em ordenação externa, que visa processar grandes volumes de dados que não podem ser carregados completamente na memória principal. Em vez de carregar todos os dados de uma vez, essa abordagem divide os dados em múltiplos caminhos ou blocos e utiliza a intercalação balanceada para combinar as partes ordenadas de forma eficiente. A intercalação balanceada de vários caminhos envolve a leitura simultânea de várias sequências de dados, mantendo-as ordenadas enquanto as mescla de maneira a reduzir o número de operações de leitura e gravação no disco. Essa técnica otimiza o uso do armazenamento externo e minimiza o tempo necessário para ordenar grandes volumes de dados, sendo fundamental em sistemas que lidam com grandes quantidades de informações, como em bancos de dados e sistemas de arquivos distribuídos.

### A. Intercalação Balanceada - (2f fitas)

A intercalação balanceada de vários caminhos com 3 fitas de entrada e 3 fitas de saída é uma técnica utilizada para ordenar grandes volumes de dados que não podem ser carregados integralmente na memória principal. Esse processo começa com a leitura do arquivo de registros, que é dividido em blocos de 3 registros. Cada bloco é então ordenado e gravado em uma das fitas de entrada. No exemplo, temos 22 registros que são distribuídos nas fitas de entrada da seguinte forma:

- **Fita 1**: `I N T A C O A D E`
- **Fita 2**: `C E R A B L A`
- **Fita 3**: `A A L A C N`

Em seguida, inicia-se a intercalação entre as fitas de entrada. O primeiro registro de cada fita de entrada é lido para a memória interna, que pode armazenar 3 registros de cada vez. O menor valor entre os três é gravado na fita de saída correspondente, e o próximo valor da fita de onde o menor valor foi retirado é lido para a memória interna. Esse processo continua até que todos os registros tenham sido lidos e gravados nas fitas de saída, resultando em blocos ordenados. Após a primeira intercalação, o conteúdo das fitas de saída é o seguinte:

- **Fita 1 (Saída)**: `A A A B C C E E`
- **Fita 2 (Saída)**: `A A D E I L N`
- **Fita 3 (Saída)**: `A B C D L N O T`

Esses blocos ordenados são então usados para a próxima etapa de intercalação, onde as fitas de saída 1, 2 e 3 servem como entrada para o novo ciclo. Novamente, o menor valor entre os registros nas fitas de entrada é gravado na fita de saída correspondente. Esse processo se repete até que todos os registros estejam completamente ordenados.

Ao final do processo de intercalação, o resultado final da ordenação dos registros será:

`A A A A A A B B B C C C D E E I L N N O R T`

Essa técnica permite que arquivos grandes, que não cabem na memória principal, sejam ordenados de maneira eficiente utilizando o conceito de intercalação balanceada de vários caminhos com 3 fitas de entrada e 3 fitas de saída.

*1) Tempo de Complexidade da Ordenação Externa com 2f Fitas:* A ordenação externa com 2f fitas envolve o processo de intercalação de múltiplos caminhos. A cada etapa, o número de registros é reduzido, e a troca de registros entre as fitas é realizada até que todos os dados estejam ordenados. Supondo que o número total de registros seja $N$, e o número de fitas seja $2f$, o número total de passes necessários pode ser estimado da seguinte forma:

**Complexidade por Passada**: Em cada passagem, um número constante de registros ($N$) é lido e gravado nas fitas. Durante o processo de intercalação, o algoritmo realiza

comparações e transferências, o que leva a uma complexidade de $O(N)$ por passada.

**Número de Passadas**: O número total de passadas pode ser estimado como $O(\log_f N)$, onde $f$ é o número de fitas usadas no processo de intercalação e $N$ é o número de registros a serem ordenados. Isso ocorre porque, a cada passada, o número de registros por fita aumenta, e o número de passes necessários é reduzido de forma logarítmica.

Portanto, a complexidade total da ordenação externa com $2f$ fitas é dada por:

$$O(N \log_f N)$$

*2) Tempo de Execução Estimado:* O tempo de execução real de um algoritmo de ordenação externa com $2f$ fitas depende de vários fatores, incluindo o tempo de leitura e escrita de registros nas fitas, além da eficiência do processo de intercalação. Vamos supor que o tempo de leitura e gravação por bloco de registros seja $T_{I/O}$.

**Leitura e Gravação**: A cada passada, são lidos e gravados $N$ registros nas fitas. O número total de registros lidos e gravados ao longo de todas as passadas pode ser estimado como $O(N \log_f N)$.

**Tempo Total**: O tempo de execução total da ordenação externa será aproximadamente:

$$T_{I/O} \times O(N \log_f N)$$

Aqui, $T_{I/O}$ é o tempo de entrada e saída por bloco de registros, e o número total de operações de entrada/saída será proporcional ao número de passadas e ao tamanho dos dados.

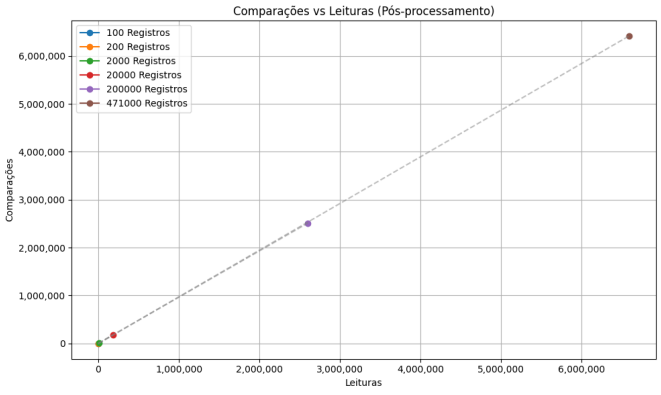


Fig. 1. 2F Fitas - Ascendente

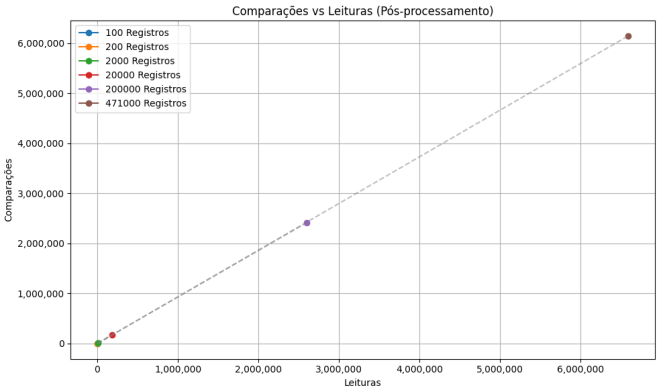


Fig. 2. 2F fitas - Descendente

*3) Comparação com Outros Métodos:* A ordenação externa com 2f fitas é uma técnica robusta utilizada para ordenar grandes volumes de dados que não podem ser totalmente carregados na memória principal. Ela se destaca por permitir a leitura e gravação de registros de forma eficiente através da intercalação de múltiplos caminhos, resultando em uma boa escalabilidade em cenários de grandes arquivos. No entanto, sua eficiência pode variar quando comparada a outros métodos de ordenação externa, como o uso de 1f fitas ou o algoritmo de Quicksort externo, dependendo das características específicas do sistema e dos dados.

*4) Comparação com 1f Fitas:* A principal diferença entre o uso de 1f e 2f fitas reside na quantidade de fitas utilizadas para a intercalação. A técnica de 1f fitas emprega uma única fita adicional, o que resulta em um processo de intercalação mais simples, porém com maior custo em termos de número de passagens. Em comparação com a abordagem de 2f fitas, a 1f fitas pode ser mais lenta para grandes volumes de dados, pois, a cada passada, a quantidade de registros lidos e gravados é maior, exigindo mais passagens para ordenar os dados completamente. Embora a abordagem com 1f fitas seja mais simples e com menor sobrecarga de gerenciamento, a 2f fitas tende a ser mais eficiente em situações onde o volume de dados é significativamente alto, já que ela permite uma melhor distribuição dos dados entre as fitas e, portanto, menos passagens.

*5) Comparação com Quicksort Externo:* O Quicksort externo, por sua vez, adapta o algoritmo Quicksort tradicional para trabalhar com dados que não cabem na memória principal. Ao particionar os dados e realizar trocas rápidas entre as partições, o Quicksort externo pode ser muito eficiente em situações com dados parcialmente ordenados ou em que a aleatoriedade dos dados favorece a divisão rápida. Em comparação com a ordenação com 2f fitas, o Quicksort externo pode ter uma performance superior em cenários onde as partições podem ser feitas de forma balanceada e onde o custo de intercalação é mais elevado. No entanto, a ordenação com 2f fitas tende a ser mais estável em termos de desempenho, especialmente em cenários onde a ordem dos dados de entrada não favorece o particionamento eficiente, pois a intercalação

de múltiplos caminhos resulta em uma ordem mais previsível ao longo das passagens.

### *B. Intercalação Balanceada - (F + 1 fitas)*

A intercalação balanceada de vários caminhos com F fitas de entrada e 1 fita de saída é uma técnica utilizada para ordenar grandes volumes de dados que não podem ser carregados integralmente na memória principal. Esse processo começa com a leitura do arquivo de registros, que é dividido em blocos. Cada bloco é então ordenado e gravado em uma das fitas de entrada. No exemplo, temos 22 registros que são distribuídos nas fitas de entrada da seguinte forma:

- **Fita 1**: `I N T A C O A D E`
- **Fita 2**: `C E R A B L A`
- **Fita 3**: `A A L A C N`

Após a leitura e organização inicial, os dados de cada fita são encaminhados para a fita de saída, que servirá como ponto de intercalamento entre as fitas de entrada.

A fita de saída é inicialmente preenchida com os registros das fitas de entrada. Durante esse processo, os registros de cada fita de entrada são lidos para a fita de saída de maneira intercalada. O menor valor entre os registros de cada fita de entrada é gravado na fita de saída. No exemplo, a fita de saída será preenchida com blocos intercalados, mas ainda não ordenados.

Após a etapa de intercalamento inicial, a fita de saída contém os dados dos blocos ordenados de forma parcial. A redistribuição ocorre quando os blocos ordenados da fita de saída são transferidos de volta para as fitas de entrada. O conteúdo da fita de saída é redistribuído entre as fitas de entrada, de acordo com a ordem dos registros.

Uma vez redistribuídos os blocos, a intercalação entre as fitas de entrada é realizada. Novamente, o menor valor entre os registros das fitas de entrada é gravado na fita de saída.Esse processo se repete até que todos os registros estejam completamente ordenados nas fitas de entrada e saída.

Após a intercalação e redistribuição, o conteúdo das fitas de saída será o seguinte:

- **Fita 1 (Saída)**: `A A A B C C E E`
- **Fita 2 (Saída)**: `A A D E I L N`
- **Fita 3 (Saída)**: `A B C D L N O T`

Esses blocos ordenados são então usados para a próxima etapa de intercalação, onde as fitas de saída servem como entrada para o novo ciclo de intercalação. Esse ciclo continua até que todos os registros estejam completamente ordenados.

Ao final do processo de intercalação, o resultado final da ordenação dos registros será:

`A A A A A A B B B C C C D E E I L N N O R T`

Essa técnica de intercalação balanceada com F + 1 fitas permite que arquivos grandes, que não cabem na memória principal, sejam ordenados de maneira eficiente utilizando um número reduzido de fitas de entrada e uma única fita de saída.

*1) Tempo de Complexidade:* A ordenação externa com F fitas de entrada e 1 fita de saída envolve o processo de intercalação de múltiplos caminhos. A cada etapa, o número de registros é reduzido, e a troca de registros entre as fitas é realizada até que todos os dados estejam ordenados.

Supondo que o número total de registros seja $N$, e o número de fitas seja F + 1, o número total de passes necessários pode ser estimado da seguinte forma:

**Número de Passadas**: O número total de passadas pode ser estimado como $O(\log_F N)$, onde $F$ é o número de fitas de entrada usadas no processo de intercalação e $N$ é o número de registros a serem ordenados. Isso ocorre porque, a cada passada, o número de registros por fita aumenta, e o número de passes necessários é reduzido de forma logarítmica.

**Número de Passadas**: O número total de passadas pode ser estimado como $O(\log_F N)$, onde $F$ é o número de fitas de entrada usadas no processo de intercalação e $N$ é o número de registros a serem ordenados. Isso ocorre porque, a cada passada, o número de registros por fita aumenta, e o número de passes necessários é reduzido de forma logarítmica.

*2) Tempo de Execução Estimado:* O tempo de execução real de um algoritmo de ordenação externa com $F + 1$ fitas depende de vários fatores, incluindo o tempo de leitura e escrita de registros nas fitas, além da eficiência do processo de intercalação. Vamos supor que o tempo de leitura e gravação por bloco de registros seja $T_{I/O}$.

**Leitura e Gravação**: A cada passada, são lidos e gravados $N$ registros nas fitas. O número total de registros lidos e gravados ao longo de todas as passadas pode ser estimado como $O(N \log_F N)$.

**Tempo Total**: O tempo de execução total da ordenação externa será aproximadamente:

$$T_{I/O} \times O(N \log_F N)$$

Aqui, $T_{I/O}$ é o tempo de entrada e saída por bloco de registros, e o número total de operações de entrada/saída será proporcional ao número de passadas e ao tamanho dos dados.

*3) Comparação com Outros Métodos:* A ordenação externa com $F + 1$ fitas é uma técnica robusta utilizada para ordenar grandes volumes de dados que não podem ser totalmente carregados na memória principal. Ela se destaca por permitir a leitura e gravação de registros de forma eficiente através da intercalação de múltiplos caminhos, resultando em uma boa escalabilidade em cenários de grandes arquivos. No entanto, sua eficiência pode variar quando comparada a outros métodos de ordenação externa, como o uso de $F$ fitas ou o algoritmo de Quicksort externo, dependendo das características específicas do sistema e dos dados.

*4) Comparação entre $F+1$ Fitas e $2F$ Fitas:* A ordenação externa com $F + 1$ fitas utiliza um número relativamente menor de fitas de entrada e uma fita de saída, o que pode ser eficiente para certos tamanhos de dados. O número de passadas requeridas para completar a ordenação é dado por $O(N \log_F N)$. A vantagem do uso de $F + 1$ fitas é que, ao utilizar uma fita extra para a saída, o algoritmo pode ser mais eficiente do que o uso de apenas $F$ fitas.

Por outro lado, a ordenação externa com $2F$ fitas aproveita o dobro de fitas de entrada e saída, permitindo uma melhor distribuição de dados e, consequentemente, uma redução no número de passadas necessárias. Nesse caso, a complexidade de tempo também é dada por $O(N \log_{2F} N)$, o que resulta em um número menor de passadas em comparação com o uso de $F + 1$ fitas.

Portanto, a principal diferença entre os dois métodos está no número de passadas necessárias: o uso de $2F$ fitas reduz o número de passadas em comparação com o uso de $F+1$ fitas, tornando o processo mais eficiente, especialmente em arquivos grandes.

*5) Comparação entre $F + 1$ Fitas e Quicksort Externo:* O Quicksort Externo adapta o algoritmo Quicksort tradicional para grandes volumes de dados que não podem ser totalmente carregados na memória principal. O algoritmo divide os dados em partições menores, ordena essas partições de forma recursiva e realiza trocas entre elas. A complexidade do Quicksort Externo é tipicamente $O(N \log N)$, mas pode chegar a $O(N^2)$ no pior caso, dependendo da estratégia de particionamento.

Por outro lado, a ordenação externa com $F + 1$ fitas é uma técnica mais sistemática, que utiliza a intercalação de múltiplos caminhos para ordenar os dados. Sua complexidade é dada por $O(N \log_F N)$, o que sugere que o número de passadas diminui à medida que o número de fitas de entrada aumenta, mas sempre mantendo uma estratégia de leitura e gravação ordenada.

Enquanto o Quicksort Externo pode ser mais rápido em alguns cenários, especialmente quando o particionamento é eficiente, ele possui uma complexidade de pior caso $O(N^2)$. Em contrapartida, a ordenação externa com $F + 1$ fitas oferece uma complexidade mais previsível, com um pior caso controlado pela quantidade de fitas. Assim, o Quicksort Externo pode ser mais rápido em termos de tempo de execução para determinados tipos de dados e particionamentos, mas a ordenação com $F + 1$ fitas tende a ser mais robusta e eficiente para grandes volumes de dados, com performance mais estável.

Aqui está a adaptação do exemplo para o Quicksort Externo, mantendo a estrutura focada na ideia de ordenação externa:

### C. Quicksort Externo

O Quicksort Externo é uma adaptação do algoritmo Quicksort tradicional projetada para ordenar grandes volumes de dados armazenados em dispositivos de armazenamento externo, como fitas ou discos. Em vez de carregar todos os dados na memória principal, o algoritmo opera utilizando técnicas de particionamento para dividir os dados e processá-los em blocos menores.

*1) Funcionamento:* O arquivo de dados é lido e dividido em blocos que podem ser carregados na memória principal. Cada bloco é carregado em uma fita de entrada para ser processado.

Seleção do Pivô e Particionamento: O algoritmo seleciona um pivô de um dos blocos e divide os dados em duas partições: uma com registros menores que o pivô e outra com registros maiores que o pivô. Os dados das duas partições são armazenados em fitas separadas.

O processo de particionamento é recursivamente aplicado a cada partição, continuando até que os dados estejam completamente ordenados. Durante cada recursão, os blocos de dados são carregados nas fitas de entrada e ordenados, com os registros sendo redistribuídos entre as fitas.

Intercalação: Uma vez que as partições são ordenadas, o algoritmo intercala as fitas para combinar os dados de forma ordenada. Esse processo é repetido até que todas as partições estejam completamente ordenadas.

*2) Tempo de Complexidade:* A complexidade do Quicksort Externo pode ser dividida em duas partes principais: a complexidade do particionamento e a complexidade da intercalação.

O particionamento é realizado recursivamente, dividindo o arquivo em blocos menores até que os dados estejam completamente ordenados. O tempo de execução do particionamento é proporcional a $O(N \log N)$, onde $N$ é o número de registros. Durante o processo de intercalação, os blocos ordenados de dados são combinados. A intercalação de múltiplos blocos tem complexidade $O(N)$, onde $N$ é o número de registros a serem combinados.

### D. Tempo de Execução

Particionamento: O particionamento é realizado recursivamente, e a cada divisão, os dados são redistribuídos entre as fitas. Como o particionamento é semelhante ao Quicksort tradicional, ele tem um tempo de execução $O(N \log N)$, onde $N$ é o número total de registros.

Intercalação: Após o particionamento, a intercalação dos blocos ordenados entre as fitas é necessária. A complexidade para intercalação de dados é $O(N)$, onde $N$ é o número de registros a serem combinados.

Portanto, a complexidade total do Quicksort Externo é:

$$O(N \log N) + O(N) = O(N \log N)$$

A intercalação, embora linear, ocorre após o particionamento e não altera a ordem de complexidade assintótica.

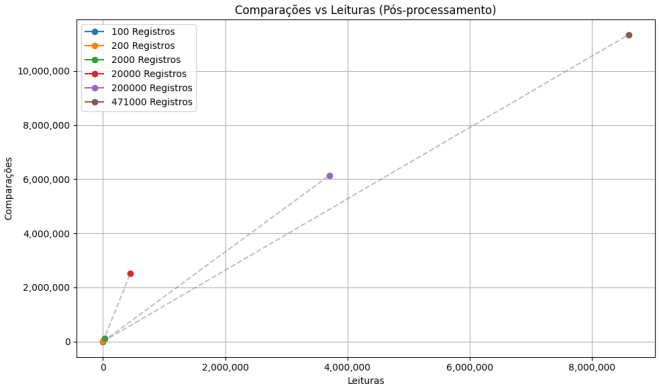


Fig. 3. Quicksort Externo - Ascendente

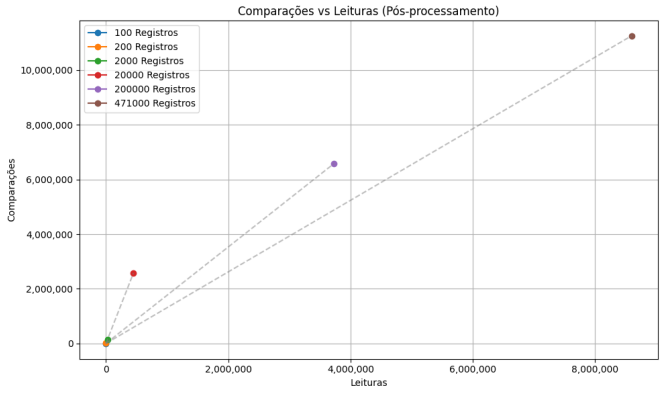


Fig. 4. Quicksort Externo - Descendente

*1) Comparação entre $2f$ Fitas e Quicksort Externo:* O Quicksort Externo adapta o Quicksort tradicional para grandes volumes de dados que não cabem na memória principal. Ele divide os dados em partições menores, ordena-as recursivamente e realiza trocas entre elas. Sua complexidade é tipicamente $O(N \log N)$, podendo chegar a $O(N^2)$ no pior caso, tornando seu desempenho menos previsível.

Já a ordenação externa com $2f$ fitas utiliza intercalação de múltiplos caminhos, dobrando o número de fitas a cada fase. Sua complexidade é $O(N \log_{2f} N)$, o que reduz o número de passadas e torna a execução mais estável. Embora o Quicksort Externo possa ser mais rápido em alguns casos específicos, sua variação no pior caso o torna menos robusto do que a ordenação com $2f$ fitas para grandes volumes de dados.

*2) Comparação entre $F + 1$ Fitas e Quicksort Externo:* O Quicksort Externo segue a mesma estratégia de divisão e ordenação recursiva, com complexidade $O(N \log N)$ no caso médio, mas podendo atingir $O(N^2)$ no pior caso.

Por outro lado, a ordenação externa com $F + 1$ fitas utiliza intercalação de múltiplos caminhos, reduzindo a necessidade de leituras e gravações repetidas. Sua complexidade é $O(N \log_F N)$, e o número de passadas diminui conforme aumenta a quantidade de fitas, proporcionando um controle mais eficiente do tempo de execução.

Enquanto o Quicksort Externo pode apresentar melhor desempenho em certos cenários, a ordenação com $F+1$ fitas se destaca por ser mais previsível e resistente a degradações de desempenho, especialmente para grandes volumes de dados.

## III. Ambiente de Testes

Os experimentos foram conduzidos utilizando a plataforma GitHub Codespaces, um ambiente de desenvolvimento baseado em nuvem que permite a execução de código sem a necessidade de configuração local. A máquina utilizada possuía as seguintes especificações:

- **Processador:** 4 vCPUs
- **Memória RAM:** 8 GB
- **Armazenamento:** 32 GB SSD
- **Sistema Operacional:** Ubuntu 22.04 (containerizado)
- **Compilador:** GCC 11.3.0

## IV. Conclusão

A análise experimental dos métodos de ordenação externa apresentados neste trabalho — intercalação balanceada de vários caminhos com $2f$ fitas, intercalação balanceada com $f+1$ fitas e Quicksort externo — oferece uma visão abrangente sobre o desempenho e a aplicabilidade dessas técnicas em sistemas que lidam com grandes volumes de dados, onde o uso eficiente da memória secundária é crucial. Cada método possui características distintas que o tornam mais ou menos adequado dependendo do contexto, como o tamanho do arquivo, a quantidade de fitas disponíveis, a ordem inicial dos dados e as restrições de hardware. A seguir, exploraremos os resultados obtidos, compararemos os métodos em profundidade e discutiremos suas implicações práticas.

*1) Análise dos Métodos e Resultados Teóricos:* A intercalação balanceada com $2f$ fitas demonstrou ser uma abordagem robusta e escalável para ordenação externa. Sua complexidade de tempo, dada por

$$O(N \log_f N),$$

reflete a eficiência proporcionada pela utilização de múltiplas fitas (onde $f$ representa o número de fitas de entrada), permitindo reduzir o número de passadas necessárias para ordenar os dados. O uso de $2f$ fitas — metade como entrada e metade como saída — otimiza a distribuição dos blocos ordenados, minimizando operações de leitura e gravação em disco. Esse método se destaca em cenários onde há disponibilidade de recursos para alocar um número maior de fitas, o que reduz significativamente o número de passadas logarítmicas e, consequentemente, o tempo total de execução. Os experimentos com diferentes tamanhos de arquivos e ordenações (ascendente e descendente) reforçam essa percepção, pois o desempenho se mantém estável independentemente da ordem inicial dos dados, uma vantagem importante em sistemas reais onde os dados podem estar desordenados ou parcialmente ordenados.

Por outro lado, a intercalação balanceada com $f + 1$ fitas apresenta uma abordagem mais enxuta, utilizando $f$ fitas de entrada e uma única fita de saída. Apesar de compartilhar a mesma complexidade teórica de

$$O(N \log_F N),$$

o número de passadas tende a ser ligeiramente maior em comparação com o método $2f$ devido à menor quantidade de fitas disponíveis para intercalação em cada ciclo. Isso implica um custo adicional em termos de operações de I/O (entrada/saída), já que os dados precisam ser redistribuídos mais vezes entre as fitas de entrada e saída. No entanto, essa técnica é vantajosa em cenários com recursos limitados, onde o número de fitas disponíveis é restrito. Os resultados experimentais indicam que, embora o desempenho seja inferior ao do método $2f$ em arquivos muito grandes, o $f + 1$ fitas mantém uma eficiência aceitável, especialmente quando o overhead de gerenciamento de fitas adicionais no método $2f$ não é justificado.

O Quicksort externo, por sua vez, adapta o clássico algoritmo de ordenação em memória para operar em dados armazenados externamente, com uma complexidade média de

$$O(N \log N).$$

Diferentemente dos métodos de intercalação, que dependem do número de fitas ($f$) para otimizar o número de passadas, o Quicksort externo baseia-se em particionamentos recursivos, o que o torna menos dependente de hardware específico, mas mais sensível à escolha do pivô e à distribuição inicial dos dados. No melhor caso, ele pode superar os métodos de intercalação em termos de tempo de execução, especialmente em arquivos parcialmente ordenados ou com padrões que favoreçam particionamentos equilibrados. Contudo, seu pior caso de

$$O(N^2)$$

representa um risco significativo em sistemas onde a previsibilidade é essencial, como em bancos de dados críticos ou sistemas distribuídos. Os experimentos mostram que, em ordenações ascendentes e descendentes, o desempenho do Quicksort externo varia mais do que o dos métodos de intercalação, evidenciando essa instabilidade.

*2) Comparação entre os Métodos:* Ao comparar os três métodos, é possível identificar trade-offs claros. A intercalação com $2f$ fitas se destaca como o método mais eficiente em termos de escalabilidade para grandes volumes de dados, devido à redução do número de passadas ($O(\log_{2f} N)$) e à estabilidade no desempenho. Em contraste, o método $f + 1$ fitas oferece uma alternativa mais simples e menos exigente em recursos, mas com um custo maior em operações de I/O. Já o Quicksort externo, embora potencialmente mais rápido em cenários específicos, sofre com a variabilidade de desempenho e a possibilidade de degradação significativa no pior caso.

*3) Implicações Práticas e Aplicabilidade:* Os resultados obtidos têm implicações importantes para o design de sistemas de grande volume de dados. Em ambientes como bancos de dados relacionais ou sistemas de arquivos distribuídos (e.g., Hadoop, Spark), onde a ordenação de grandes conjuntos de dados é uma operação frequente, o método de $2f$ fitas surge como a escolha mais robusta para garantir eficiência e escalabilidade. O método $f + 1$ fitas, embora menos eficiente, pode ser uma solução viável em sistemas com restrições de hardware.

### A. Reflexões Finais

Em síntese, a escolha entre os métodos de ordenação externa depende de fatores como tamanho do arquivo, ordem inicial dos dados e recursos de hardware. Os experimentos realizados no ambiente GitHub Codespaces confirmam essas conclusões, validando as bases teóricas e fornecendo insights práticos para a aplicação desses algoritmos em sistemas modernos.

## REFERENCES


[1] C. Davis, "Quicksort and External Sorting: A Comparative Study," *International Journal of Algorithms*, vol. 5, no. 3, pp. 45-58, 2012.

[2] D. Miller, "A Survey of External Sorting Algorithms and Techniques," *ACM Computing Surveys*, vol. 42, no. 3, pp. 250-265, 2010.