Universidade Federal de Ouro Preto - UFOP
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM
Ciência da Computação

Ordenação externa

BCC203 - Estrutura de Dados II

Bárbara Rodrigues Mateus, Caio Lucas Pereira da Silva, Vinicius Nunes dos Anjos
Professor: Guilherme Tavares de Assis

Ouro Preto
10 de agosto de 2023

Sumário

Lista de Códigos Fonte

1 Introdução

Este trabalho prático tem como objetivo realizar um estudo da complexidade de desempenho de diferentes métodos de ordenação externa. Os métodos que foram implementados, são os que foram ensinados em sala de aula. Sendo eles: Quicksort externo, intercalação balanceada de vários caminhos (2f fitas) utilizando, na etapa de geração dos blocos ordenados, qualquer método de ordenação interna apresentado em "Estrutura de Dados I"e outro com geração dos bolocos ordenados, pela técnica de seleção por substituição apresentada em "Estrutura de Dados II".

1.1 Considerações iniciais

Algumas ferramentas foram utilizadas durante a criação deste projeto:

- Ambiente de desenvolvimento do código fonte: Visual Studio Code. [1]
- Linguagem utilizada: C.
- Ambiente de desenvolvimento da documentação: Overleaf LaTeX. [2]

1.2 Ferramentas utilizadas

Algumas ferramentas foram utilizadas para testar a implementação, como:

- *Valgrind*: ferramentas de análise dinâmica do código.

1.3 Especificações da máquina

A máquina utilizada para a realização dos testes possuem as seguintes especificações:

- Processador: AMD Ryzen 5-5500U.
- Memória RAM: 8GB.
- Sistema Operacional: Arch Linux.

1.4 Instruções de compilação e execução

Para a compilação do projeto, basta digitar:

Compilando o projeto

```
make
```

Usou-se para a compilação as seguintes flags:

- *-std=c99*: Uso do padrão ANSI 99;
- *-Wall*: para mostrar os possíveis *warnings* do código;
- *-Werror*: para tratar os warnings como erros no código;
- *-Wextra*: para alertar sobre possíveis warnings extras do código;
- *-O3*: para otimizar o código;
- *-lm*: inclui a biblioteca de matemática no código.

Para a execução do programa basta digitar:

```
./ordena ¡método¿ ¡quantidade¿ ¡situação¿ [-p]
```

[1] Visual Studio Code está disponível em https://code.visualstudio.com
[2] Disponível em https://www.overleaf.com/

2 Desenvolvimento

2.1 Quicksort Externo

O algoritmo de ordenação Quicksort é eficiente quando aplicado a conjuntos de dados grandes, mas ele não foi originalmente projetado para trabalhar com dados externos, ou seja, dados que não cabem totalmente na memória principal. No entanto, é possível adaptar o Quicksort para ordenação externa, onde o processo de ordenação ocorre utilizando a memória principal e também fazendo leituras e escritas em uma memória secundária, como um disco rígido.

Aqui estão algumas características principais do Quicksort externo em C:

- Divisão de blocos: Em um algoritmo de Quicksort externo, o processo de ordenação é dividido em duas partes principais: a fase de classificação interna, onde os dados são classificados na memória principal (RAM), e a fase de fusão externa, onde os blocos ordenados são combinados e mesclados em um único arquivo ordenado no armazenamento externo (disco rígido).

- Divisão e conquista: O conceito central do Quicksort externo também é baseado na estratégia "dividir e conquistar". A ideia é dividir o conjunto de dados externos em blocos menores que possam ser ordenados na memória principal. Em seguida, os blocos ordenados são combinados em um processo de mesclagem externa.

- Escolha do pivô externo: Uma parte crucial do Quicksort é a escolha do pivô. No caso do Quicksort externo, a escolha do pivô externo envolve selecionar elementos-chave nos blocos de dados externos para auxiliar na divisão e ordenação.

- Fase de mesclagem: Após a fase de classificação interna, onde os blocos são ordenados individualmente na memória principal, a fase de mesclagem externa ocorre. Nessa fase, os blocos ordenados são lidos e mesclados em um único arquivo ordenado no armazenamento externo, como um arquivo em disco.

- Estratégia de leitura/escrita: Uma característica importante do Quicksort externo é como ele gerencia as operações de leitura e escrita nos dados externos. Ele precisa garantir que as leituras e escritas sejam feitas de maneira eficiente para minimizar o número de operações de entrada/saída (I/O) no armazenamento externo.

- Complexidade de tempo: A complexidade de tempo do Quicksort externo é afetada pela fase de mesclagem externa, que pode ser mais lenta do que a fase de classificação interna, especialmente quando os blocos a serem mesclados não cabem totalmente na memória principal. Portanto, a complexidade de tempo geral dependerá das características da memória externa e da eficiência da mesclagem.

2.2 Intercalação balanceada de vários caminhos (2f fitas) utilizando Merge Sort

A intercalação balanceada de vários caminhos, também conhecida como intercalação de fitas, é uma técnica utilizada para ordenar conjuntos de dados externos, onde os dados são muito grandes para caber completamente na memória principal. Ela envolve o uso de múltiplas fitas ou dispositivos de armazenamento externo para realizar a ordenação e mesclagem dos dados.

O Merge Sort é um algoritmo de ordenação eficiente que se presta bem a essa abordagem de intercalação balanceada. A ideia básica é dividir o conjunto de dados em pedaços menores, chamados de blocos, que são então ordenados na memória principal e intercalados em uma fita de saída. A intercalação ocorre entre as fitas de entrada e a fita de saída, permitindo que os dados sejam combinados e mesclados de maneira ordenada.

Aqui estão algumas características principais da intercalação balanceada de vários caminhos (2f fitas) utilizando o Merge Sort:

- Divisão em blocos: O processo começa com a divisão do conjunto de dados externos em blocos que são ordenados individualmente na memória principal. Esses blocos ordenados são então escritos nas fitas de entrada.

- Intercalação em múltiplas fitas: No Merge Sort utilizando 2f fitas, o processo de intercalação ocorre entre as fitas de entrada e a fita de saída. Duas fitas são usadas como entradas, e os blocos são combinados e mesclados de maneira ordenada na fita de saída. Isso é feito em passos iterativos até que todos os blocos tenham sido mesclados em uma única sequência ordenada.

- Balanceamento de intercalação: O termo "balanceado"refere-se ao fato de que a intercalação é distribuída de maneira uniforme entre as fitas de entrada e a fita de saída. Isso ajuda a evitar gargalos e a aproveitar ao máximo o potencial de leitura e gravação dos dispositivos externos.

- Eficiência de leitura/escrita: A intercalação balanceada busca otimizar as operações de leitura e escrita nos dispositivos externos, minimizando as operações de entrada/saída. Isso é alcançado por meio da estratégia de intercalação bem distribuída e pela ordenação eficiente dos blocos na memória principal.

- Complexidade de tempo: A complexidade de tempo da intercalação balanceada de vários caminhos depende do número de passos necessários para mesclar todos os blocos em uma única sequência ordenada. Quanto mais fitas de entrada forem usadas (2f fitas), menos passos serão necessários, tornando a intercalação mais eficiente.

2.3 Intercalação balanceada de vários caminhos (2f fitas) utilizando Seleção por Substituição

A intercalação balanceada de vários caminhos é uma estratégia eficaz para ordenar grandes conjuntos de dados externos, onde a capacidade da memória principal é insuficiente para acomodar todos os dados. Nesse método, várias fitas ou dispositivos de armazenamento externo são utilizados para realizar a ordenação e intercalação dos dados. A técnica de Seleção por Substituição é aplicada na etapa de geração dos blocos ordenados, utilizando a estrutura de dados Heap.

A ideia central é dividir o conjunto de dados em blocos menores, ordená-los na memória principal e, em seguida, realizar a intercalação dos blocos nas fitas de saída. A intercalação ocorre entre as fitas de entrada e a fita de saída, permitindo que os dados sejam mesclados de maneira ordenada. Aqui estão as principais características desse método:

- Divisão em Blocos: O processo começa com a divisão dos dados externos em blocos, os quais são ordenados individualmente na memória principal. Esses blocos ordenados são então escritos nas fitas de entrada.

- Intercalação em Múltiplas Fitas: Ao utilizar a técnica de Seleção por Substituição com 2f fitas, a intercalação ocorre entre as fitas de entrada e a fita de saída. Duas fitas são usadas como entrada, e os blocos são mesclados de forma ordenada na fita de saída. Esse processo é repetido até que todos os blocos tenham sido mesclados em uma sequência única e ordenada.

- Utilização de Heap: A estrutura de dados Heap é empregada na etapa de seleção por substituição. Ela permite que o menor item seja constantemente mantido no topo da fila de prioridades, facilitando a escolha do próximo item a ser intercalado.

- Balanceamento da Intercalação: O termo "balanceado"refere-se à distribuição uniforme da intercalação entre as fitas de entrada e a fita de saída. Isso evita gargalos e otimiza a leitura e escrita nos dispositivos externos.

- Eficiência de Leitura/Escrita: A intercalação balanceada busca otimizar as operações de leitura e escrita nos dispositivos externos, minimizando as operações de entrada/saída. Isso é alcançado por meio da estratégia bem distribuída de intercalação e da ordenação eficiente dos blocos na memória principal.

- Complexidade Temporal: A complexidade temporal da intercalação balanceada depende do número de passos necessários para mesclar todos os blocos em uma única sequência ordenada. Ao empregar um maior número de fitas de entrada (2f fitas), reduz-se a quantidade de passos necessários, resultando em um processo de intercalação mais eficiente.

Este é o texto reformulado no formato LaTeX, com a inclusão da estrutura de heap e os elementos apresentados anteriormente. Certifique-se de adaptar o texto conforme necessário para seu uso específico.

2.4 Incluindo fragmento de códigos

O Código 1 apresenta um exemplos do código do trabalho.

```
bool balanced_intercalation(char const *out_filename, Tape *tapes,
                            Performance *perf) {
  bool is_intercalated = false;

  while (should_intercalate(tapes, is_intercalated)) {
    flush_tapes(tapes);
    rewind_tapes(tapes);

    reopen_tapes(tapes, is_intercalated);

    int blocks_size = block_size(tapes, is_intercalated);

    for (int i = 0; i < blocks_size; i++) {
      intercalate(tapes, i + 1, is_intercalated, perf);
      flush_tapes(tapes);
    }

    is_intercalated = !is_intercalated;
  }

  flush_tapes(tapes);
  rewind_tapes(tapes);

  int tape_index = 0;

  if (is_intercalated)
    tape_index += HALF_TAPES_SZ;

  FILE *txt_file = fopen(out_filename, "w");

  for (int i = tape_index; i < (tape_index + HALF_TAPES_SZ); i++) {
    if (tapes[i].block_size > 0) {
      Register reg;

      rewind(tapes[i].file);

      int trash;
      fread(&trash, sizeof(int), 1, tapes[i].file);

      while (fread(&reg, sizeof(Register), 1, tapes[i].file)) {
        fprintf(txt_file, "%08zu %04.1f %s", reg.id, reg.grade, reg.content);
      }
    }
  }

  fclose(txt_file);
  close_tapes(tapes);
  return true;
}
```

Código 1: Trecho do código que realiza a intercalação balanceada dos registros nos arquivos.

3 Experimentos e Resultados

Para a realização dos experimentos e obtenção dos resultados, o código foi executando através de diferentes situações, métodos e quantidade de arquivos para que fosse possível a melhor obtenção possível de resultados não enviezádos.

3.1 Utilizando um arquivo de 100 registros:

Arquivo Ordenado Crescente:

Método	Intercalação Interna	Intercalação Externa	QuickSort
TM Ordenação	0.000083s	0.001355s	0.000527s
TM Pós-Processamento	0.001520s	0.000481s	0.000352s
TM Total	0.001603s	0.001836s	0.000879s
Comparações da Ordenação	528	668	99
Comparações da Pós Processamento	3680	0	0
Comparações Totais	4208	668	99
Leituras da Ordenação	7	120	100
Leituras do Pós Processamento	307	0	200
Leituras Totais	314	120	300
Escritas da Ordenação	10	201	100
Escritas do Pós Processamento	203	0	0
Escritas Totais	213	201	100

Arquivo Ordenado Descrescente:

Método	Intercalação Interna	Intercalação Externa	QuickSort
TM Ordenação	0.000088s	0.001324s	0.000590s
TM Pós-Processamento	0.001688s	0.000536s	0.000477s
TM Total	0.001776s	0.001860s	0.001067s
Comparações da Ordenação	528	807	2389
Comparações do Pós Processamento	3800	3800	0
Comparações Totais	4328	4607	2389
Leituras da Ordenação	7	120	100
Leituras do Pós Processamento	307	206	200
Leituras Totais	314	120	300
Escritas da Ordenação	10	201	100
Escritas do Pós Processamento	203	203	0
Escritas Totais	213	407	100

Arquivo Aleatório:

Método	Intercalação Interna	Intercalação Externa	QuickSort
TM Ordenação	0.000081s	0.000726s	0.000575s
TM Pós-Processamento	0.001430s	0.000677s	0.000453s
TM Total	0.001511s	0.002039s	0.001028s
Comparações da Ordenação	528	774	80
Comparações do Pós Processamento	3785	3797	0
Comparações Totais	4313	4571	80
Leituras da Ordenação	7	120	120
Leituras do Pós Processamento	307	205	200
Leituras Totais	314	325	300
Escritas da Ordenação	10	203	100
Escritas do Pós Processamento	203	203	0
Escritas Totais	213	406	100

3.2 Utilizando um arquivo de 1.000 registros:

Arquivo Ordenado Crescente:

Método	Intercalação Interna	Intercalação Externa	QuickSort
TM Ordenação	0.000288s	0.011321s	0.002440s
TM Pós-Processamento	0.001752s	0.000666s	0.001022s
TM Total	0.002040s	0.011987s	0.003462s
Comparações da Ordenação	4488	6968	5544
Comparações do Pós Processamento	29100	0	0
Comparações Totais	33588	6968	5544
Leituras da Ordenação	52	1020	5555
Leituras do Pós Processamento	3054	0	2000
Leituras Totais	3106	1020	7555
Escritas da Ordenação	100	2001	5555
Escritas do Pós Processamento	2005	0	0
Escritas Totais	2105	2001	5555

Arquivo Ordenado Descrescente:

Método	Intercalação Interna	Intercalação Externa	QuickSort
TM Ordenação	0.000727s	0.006344s	0.007791s
TM Pós-Processamento	0.002586s	0.001343s	0.001455s
TM Total	0.003313s	0.007687s	0.009246s
Comparações da Ordenação	4488	7797	66648
Comparações do Pós Processamento	38000	38000	0
Comparações Totais	42488	45797	66648
Leituras da Ordenação	52	1020	1890
Leituras do Pós Processamento	3054	2027	2000
Leituras Totais	3106	3047	3890
Escritas da Ordenação	100	2025	1890
Escritas do Pós Processamento	2005	2003	0
Escritas Totais	2105	4028	1890

Arquivo Aleatório:

Método	Intercalação Interna	Intercalação Externa	QuickSort
TM Ordenação	0.000764s	0.007845s	0.004959s
TM Pós-Processamento	0.003768s	0.001732s	0.000919s
TM Total	0.003313s	0.009577s	0.005878s
Comparações da Ordenação	4488	8206	55809
Comparações do Pós Processamento	37828	37942	0
Comparações Totais	42316	46148	55809
Leituras da Ordenação	52	1020	2869
Leituras do Pós Processamento	3054	2027	2000
Leituras Totais	3106	3047	4869
Escritas da Ordenação	100	2025	2869
Escritas do Pós Processamento	2005	2003	0
Escritas Totais	2105	4028	2869

3.3 Utilizando um arquivo de 10.000 registros:

Arquivo Ordenado Crescente:

Método	Intercalação Interna	Intercalação Externa	QuickSort
TM Ordenação	0.001997s	0.063440s	0.057799s
TM Pós-Processamento	0.014790s	0.003878s	0.006395s
TM Total	0.016787s	0.067318s	0.064194s
Comparações da Ordenação	44088	69968	509949
Comparações do Pós Processamento	394800	378617	0
Comparações Totais	438888	69968	509949
Leituras da Ordenação	502	10020	510050
Leituras do Pós Processamento	40528	0	20000
Leituras Totais	41030	10020	530050
Escritas da Ordenação	1000	20001	510050
Escritas do Pós Processamento	30029	0	0
Escritas Totais	31029	20001	510050

Arquivo Ordenado Descrescente:

Método	Intercalação Interna	Intercalação Externa	QuickSort
TM Ordenação	0.002886s	0.077008s	0.108239s
TM Pós-Processamento	0.020960s	0.009487s	0.007509s
TM Total	0.023846s	0.086495s	0.115748s
Comparações da Ordenação	44088	74695	766442
Comparações do Pós Processamento	560835	380000	0
Comparações Totais	604923	454695	766442
Leituras da Ordenação	502	10020	58619
Leituras do Pós Processamento	40528	20088	20000
Leituras Totais	41030	30108	78619
Escritas da Ordenação	1000	20082	58619
Escritas do Pós Processamento	30029	20088	0
Escritas Totais	31029	40089	58619

Arquivo Aleatório:

Método	Intercalação Interna	Intercalação Externa	QuickSort
TM Ordenação	0.004065s	0.070364s	0.087338s
TM Pós-Processamento	0.014535s	0.010388s	0.007281s
TM Total	0.018600s	0.080752s	0.094619s
Comparações da Ordenação	44088	83169	485090
Comparações do Pós Processamento	568259	378617	0
Comparações Totais	612347	461786	485090
Leituras da Ordenação	502	10020	65517
Leituras do Pós Processamento	40528	20251	20000
Leituras Totais	41030	30271	85517
Escritas da Ordenação	1000	20239	65517
Escritas do Pós Processamento	30029	20013	0
Escritas Totais	31029	40252	65517

3.4 Utilizando um arquivo de 100.000 registros:

Arquivo Ordenado Crescente:

Método	Intercalação Interna	Intercalação Externa	QuickSort
TM Ordenação	0.020338s	0.591614s	3.199459s
TM Pós-Processamento	0.155371s	0.039794s	0.066304s
TM Total	0.175709s	0.631408s	3.265763s
Comparações da Ordenação	440088	701704	13161872
Comparações do Pós Processamento	3251449	0	0
Comparações Totais	3691537	701704	13161872
Leituras da Ordenação	5002	100020	11557570
Leituras do Pós Processamento	405263	0	200000
Leituras Totais	410265	1020	11757570
Escritas da Ordenação	10000	200001	11557570
Escritas do Pós Processamento	300264	0	0
Escritas Totais	310264	200001	11557570

Arquivo Ordenado Decrescente:

Método	Intercalação Interna	Intercalação Externa	QuickSort
TM Ordenação	0.018130s	0.687918s	2.861352s
TM Pós-Processamento	0.151892s	0.085866s	0.081998s
TM Total	0.170022s	0.773784s	2.943350s
Comparações da Ordenação	440088	717854	5216072
Comparações do Pós Processamento	5221981	3800000	0
Comparações Totais	5662069	4517854	5216072
Leituras da Ordenação	5002	100020	1347815
Leituras do Pós Processamento	405263	200242	200000
Leituras Totais	410265	300262	1547815
Escritas da Ordenação	10000	200230	1347815
Escritas do Pós Processamento	300264	200013	0
Escritas Totais	310264	400243	1347815

Arquivo Aleatório:

Método	Intercalação Interna	Intercalação Externa	QuickSort	
TM Ordenação	0.020050s		0.670378s	1.485009s
TM Pós-Processamento	0.140857s	0.119579s	0.078860s	
TM Total	0.160907s	0.789957s	1.563869s	
Comparações da Ordenação	440088	829261	2771807	
Comparações do Pós Processamento	5678096	5688918	0	
Comparações Totais	6118184	6518179	2771807	
Leituras da Ordenação	5002	100020	1456362	
Leituras do Pós Processamento	405263	302565	200000	
Leituras Totais	410265	402585	1656362	
Escritas da Ordenação	10000	202436	1456362	
Escritas do Pós Processamento	300264	300130	0	
Escritas Totais	310264	502566	1456362	

3.5 Utilizando um arquivo de 471.705 registros:

Arquivo Ordenado Crescente:

Método	Intercalação Interna	Intercalação Externa	QuickSort
TM Ordenação	0.082823s	2.876287s	20.823405s
TM Pós-Processamento	0.740341s	0.186090s	0.339536s
TM Total	0.823164s	3.062377s	21.162941s
Comparações da Ordenação	2075496	3313374	34870036
Comparações do Pós Processamento	22076803	0	0
Comparações Totais	24152299	3313374	34870036
Leituras da Ordenação	23587	471725	19375289
Leituras do Pós Processamento	2383276	0	943410
Leituras Totais	2406863	471725	20318699
Escritas da Ordenação	47170	943411	19375289
Escritas do Pós Processamento	1887987	0	0
Escritas Totais	1935157	943411	19375289

Arquivo Ordenado Descrescente:

Método	Intercalação Interna	Intercalação Externa	QuickSort
TM Ordenação	0.096667s	3.227165s	21.215541s
TM Pós-Processamento	0.769522s	0.478524s	0.340445s
TM Total	0.866189s	3.705689s	21.555986s
Comparações da Ordenação	2075496	3351221	34874738
Comparações do Pós Processamento	32199919	26887185	0
Comparações Totais	34275415	30238406	34874738
Leituras da Ordenação	23587	471725	19375242
Leituras do Pós Processamento	2383276	1415745	943410
Leituras Totais	2406863	1887470	20318652
Escritas da Ordenação	47170	944008	19375242
Escritas do Pós Processamento	1887987	1415148	0
Escritas Totais	1935157	2359156	19375242

Arquivo Aleatório:

Método	Intercalação Interna	Intercalação Externa	QuickSort
TM Ordenação	0.103259s	3.284569s	19.869990s
TM Pós-Processamento	0.765248s	0.632459s	0.341374s
TM Total	0.868507s	3.917028s	20.211364s
Comparações da Ordenação	2075496	3900799	27354041
Comparações do Pós Processamento	35735683	35780259	0
Comparações Totais	37811179	39681058	27354041
Leituras da Ordenação	23587	471725	19349601
Leituras do Pós Processamento	2383276	1898747	943410
Leituras Totais	2406863	2370472	20293011
Escritas da Ordenação	47170	954738	19349601
Escritas do Pós Processamento	1887987	1898747	0
Escritas Totais	1935157	2842158	19349601

4 Análise dos testes

4.1 Intercalação Balanceada Interna

O método de intercalação balanceada interna é extremamente eficaz quando se trata de velocidade da ordenação em si. Este método possui o tempo médio de ordenação mais baixo de todos os 3 utilizados durante o desenvolvimento do trabalho. Isso se dá pelo fato de ele usufrir da memória interna da máquina para realizar esse processo, fazendo-o ser ótimo quando possuimos uma boa quantidade de fitas ou um bom espaço em bloco para ser utilizado.

Além disso, a Intercalação Interna demonstrou ser eficaz tanto para arquivos previamente ordenados quanto para arquivos aleatórios. Esse desempenho consistente é uma característica valiosa, uma vez que muitos conjuntos de dados na prática apresentam alguma forma de ordenação inicial ou padrões previsíveis. Portanto, ao aproveitar essa característica, a Intercalação Interna pode oferecer uma ordenação rápida e eficiente em uma variedade de contextos.

No geral, este método se demonstra bem eficaz em todas as situações às quais foi submetido, sempre apresentando um tempo de execução eficaz e consistente, além de ser menor que todos os outros na média, também é o que realiza menor número de leituras e escritas no geral.

4.2 Intercalação Balanceada Externa

A intercalação Balanceada Externa pode ser julgada como a intermediária entre as outras duas. Ela não é tão rápida quanto a intercalação balanceada interna e nem tão lenta quanto o quick sort externo.

Uma das principais vantagens é que quando o arquivo já está ordenado ascendentemente, seu número de comparações é zerado, juntamente com o número de leituras e escritas. Seu número de comparações tende a diminuir em relação a intercalação interna quando lidamos com valores maiores também. Contudo, irá continuar sendo lenda devido ao seu grande número de acessos à arquivos.

4.3 Quick Sort Externo

A principal vantagem de usar o Quick Sort Externo como método de ordenação de arquivos é a baixa necessidade de se ter memória principal. Como todas as manipulações ocorrem diretamente no arquivo a ser ordenado, mesmo com apenas um bloco de memória seria possível realizar a ordenação de um arquivo inteiro.

Uma vantagem deste método de ordenação é quando usado em arquivos com poucos registros. Nos testes realizados com 100 registros, o quick sort externo realiza o menor número de leituras e escritas em arquivos, o tornando o melhor em número de leituras e escritas quando se trata de quantidades pequenas.

Contudo, é possível perceber que na medida em que o tamanho do arquivo cresce, este método começa a ficar ineficaz. Vamos olhar para o arquivo com o número total de registros por exemplo: O quick sort demorou cerca de 20s para realizar totalmente a ordenação, enquanto os outros 2 métodos não demoram mais que 4 segundos. Isso se dá pelo fato de que manipulações diretas em arquivos são processos mais custosos que manipulações em memória interna ou em fitas. Outra desvantagem deste método é seu grande número de comparações, leituras e escritas, que aumentam significantemente com um maior número de registros

Com os dados obtidos a partir dos testes, é possível concluir que o Quick Sort Externo não é a melhor opção quando se trata de ordenação em arquivos. Por mais que haja a vantagem de não precisar de quase nenhuma memória interna, seu número de comparações, leituras e escritas é astronômico, o tornando extremamente ineficaz nessas situações.

5 Conclusão

A análise detalhada das informações apresentadas nas tabelas revela insights significativos sobre o desempenho e a eficácia de diferentes métodos de ordenação em relação a diversos cenários de dados. Observa-se que, para tamanhos menores de arquivo, o método de Intercalação Interna exibe tempos de execução mais curtos e um menor número de comparações, leituras e escritas em relação aos métodos de Intercalação Externa e QuickSort.dá pra salvar codigo dos outros pelo liveshadá pra salvar codigo dos outros pelo livesha Por fim, considerando a aplicação prática e a análise teórica, os resultados destacam a importância de escolher o método de ordenação apropriado com base no contexto e no tamanho dos dados, a fim de otimizar a eficiência do processo de ordenação.