Trabalho Prático II – Ordenação Externa

Disciplina: Estrutura de Dados II – BCC203
Instituição: Universidade Federal de Ouro Preto – UFOP / ICEB / DECOM
Professor: Guilherme Tavares de Assis
Período: 2025/2

Grupo:

- Bernardo Fernandes Costa
- João Pedro de Sousa Cavalcante
- Maria Paula Miranda de Souza Montovani Gasparini
- Mateus Lima de Araújo
- Vitor de Sousa Costa Lopes

---

1. Introdução

A ordenação de grandes volumes de dados armazenados em **memória externa** (disco) é um problema relevante porque o custo de E/S é muito maior do que operações em memória interna. Como não é viável carregar todos os registros de um arquivo grande em RAM, utilizam-se algoritmos de **ordenação externa**, projetados para reduzir o número de leituras e escritas e, consequentemente, o tempo total de execução.

Neste trabalho, foram implementados e avaliados experimentalmente três métodos de ordenação externa discutidos em sala:

1. **Intercalação balanceada de vários caminhos (2f fitas)** com **seleção por substituição** na geração de blocos;
2. **Intercalação balanceada de vários caminhos (f+1 fitas)** com **seleção por substituição** na geração de blocos;
3. **Quicksort Externo**.

O conjunto de dados utilizado foi o arquivo **PROVAO.TXT**, contendo 471.705 alunos do Provão (2003). Cada registro possui, dentre outros campos, a **nota** (0.0 a 100.0). O objetivo dos experimentos é **ordenar ascendentemente por nota** as **n primeiras linhas** do arquivo, com $n \in \{100, 200, 2.000, 20.000, 200.000, 471.705\}$.

Além disso, foram testadas três situações de ordem inicial do arquivo:

- **Ascendente** (já ordenado por nota, crescente);
- **Descendente** (ordenado por nota, decrescente);
- **Aleatório** (arquivo original desordenado).

O programa foi executado via console no formato:

ordena <método> <quantidade> <situação> [-P]

Em cada execução foram coletadas as métricas exigidas:

- **Leituras** (transferências da memória externa → interna);
- **Escritas** (transferências da memória interna → externa);
- **Comparações** (comparações entre notas);
- **Tempo de execução** (fim − início).

---

2. Descrição dos Experimentos

Para cada método (1 a 3), foram realizadas execuções para cada n (100, 200, 2.000, 20.000, 200.000 e 471.705) nas três situações iniciais (ascendente, descendente e aleatória). Cada experimento consistiu em:

1. Selecionar o arquivo correspondente à situação inicial;
2. Considerar apenas o prefixo de tamanho n;
3. Executar o método escolhido;
4. Registrar leituras, escritas, comparações e tempo.

O parâmetro opcional **-P** foi usado em execuções de verificação para imprimir registros antes/depois e confirmar a ordenação por nota.

---

3. Método 1 – Intercalação Balanceada (2f fitas)

3.1. Implementação (resumo)

O método 1 utiliza **2f fitas** de armazenamento externo, sendo **10 fitas de entrada e 10 fitas de saída** em cada etapa de intercalação. A geração dos blocos iniciais é feita por meio da técnica de **seleção por substituição**, considerando uma memória interna com capacidade máxima de **10 registros**, conforme especificado no enunciado do TP2.

Após a geração dos blocos ordenados, são realizadas sucessivas etapas de intercalação balanceada até que reste apenas um único arquivo completamente ordenado.

3.2. Resultados experimentais

A seguir são apresentados os resultados obtidos experimentalmente para o método 1. Quando algum teste **não produziu métricas**, isso é explicitamente indicado como **não funcionou**, conforme solicitado.

3.2.1. Arquivo ascendente

n	Leituras	Escritas	Comparações	Tempo (s)
100	100	100	204	0.000217
200	200	200	404	0.000153
2.000	2.000	2.000	4.004	0.000534
20.000	20.000	20.000	40.004	0.002203
200.000	200.000	200.000	406.037	0.019063
471.705	471.705	471.705	962.617	0.051384

3.2.2. Arquivo descendente

n	Leituras	Escritas	Comparações	Tempo (s)
100	200	200	385	0.004709
200	1.100	1.100	752	0.000714
2.000	26.000	26.000	5.460	0.003882
20.000	1.520.000	1.520.000	43.722	0.134193
200.000	75.200.000	75.200.000	409.962	6.922768

471.705	944.935.230	944.935.230	960.788	89.515942

3.2.3. Arquivo aleatório (PROVAO)

n	Leituras	Escritas	Comparações	Tempo (s)
100	200	200	1214	0.002396
200	600	600	3695	0.003447
2.000	6.000	6.000	53.450	0.007538
20.000	80.000	80.000	755.933	0.038937
200.000	1.000.000	1.000.000	9.780.857	0.285773
471.705	2.830.230	2.830.230	24571355	0.731718

3.3. Discussão do desempenho (método 1)

Observa-se que o método 1 apresenta excelente desempenho para arquivos inicialmente **ordenados de forma ascendente**, com crescimento praticamente linear das métricas. Entretanto, para arquivos descendentes e aleatórios, o número de leituras e escritas cresce de forma muito acentuada, refletindo o aumento do número de blocos e de passadas de intercalação necessárias. Nos maiores valores de *n*, o método ainda conclui a execução, porém com custo de E/S significativamente maior e tempo bem superior ao caso ascendente, evidenciando a sensibilidade do método à ordem inicial da entrada..

---

4. Método 2 – Intercalação Balanceada (f+1 fitas)

Neste método, foi considerada a existência de 20 fitas, sendo **19 fitas de entrada** e **1 fita de saída** por etapa de intercalação. A geração de blocos ordenados utilizou **seleção por substituição** e uma memória interna de **até 19 registros**.

4.1. Resultados experimentais

A seguir, os resultados obtidos para o método 2, separados por situação inicial do arquivo.

4.1.1. Arquivo ascendente

n	Leituras	Escritas	Comparações	Tempo (s)
100	200	200	431	0.000170
200	400	400	831	0.000123
2.000	4.000	4.000	8.031	0.000632
20.000	40.000	40.000	80.031	0.004510
200.000	400.000	400.000	817.651	0.043802
471.705	943.410	943.410	1.941.683	0.102724

4.1.2. Arquivo descendente

n	Leituras	Escritas	Comparações	Tempo (s)
100	200	200	883	0.000244
200	400	400	1.598	0.000328
2.000	12.000	12.000	20.283	0.004335
20.000	16.000	16.000	212.109	0.023627

200.000	3.600.000	3.600.000	4.036.795	0.414738
471.705	15.094.560	15.094.560	16.107.572	1.696832

4.1.3. Arquivo aleatório (PROVAO)

n	Leituras	Escritas	Comparações	Tempo (s)
100	200	200	1.108	0.000249
200	400	400	2.250	0.000270
2.000	12.000	12.000	33.124	0.002424
20.000	1.160.000	1.160.000	1.398.640	0.145241
200.000	110.800.000	110.800.000	114.519.084	12.679156
471.705	613.216.500	613.216.500	625.389.764	70.692694

4.2. Discussão do desempenho (método 2)

- **Crescimento com n:** Para ascendente, observa-se crescimento aproximadamente linear de leituras e escritas (ordem de grandeza ~ 2n), com tempos muito baixos até 471.705. Já para descendente e principalmente aleatório, há casos com crescimento muito maior de leituras/escritas, refletindo maior número de fases e maior custo de intercalações.
- **Influência da ordem inicial:** O caso aleatório foi o mais custoso para n grandes, indicando aumento substancial de blocos e do número de passagens de intercalação.

---

5. Método 3 – Quicksort Externo

O Quicksort Externo foi implementado considerando memória interna para no máximo **10 registros**. O método realiza partições sucessivas no arquivo, utilizando uma **área interna ordenada** como estrutura de apoio, e recorre a ordenação interna (SmallSort) quando o subarquivo cabe na memória.

Para preservar os arquivos originais e manter reprodutibilidade, as execuções foram feitas sobre um arquivo temporário de trabalho (**WORK.bin**) contendo apenas os n registros necessários.

5.1. Resultados experimentais

5.1.1. Arquivo ascendente

n	Leituras	Escritas	Comparações	Tempo (s)
100	582	582	1.136	0.000288
200	1.362	1.362	2.845	0.001365
2.000	20.255	20.255	48.195	0.016263
20.000	270.111	270.111	682.481	0.163524
200.000	3.367.043	3.367.043	11.869.775	2.146239
471.705	8.506.730	8.506.730	32.135.235	5.489536

5.1.2. Arquivo descendente

n	Leituras	Escritas	Comparações	Tempo (s)
100	582	582	2.372	0.000745
200	1.362	1.362	1.598	0.001237

2.000	20.255	20.255	5.827	0.004335
20.000	270.111	270.111	1.029.668	0.165106
200.000	3.367.043	3.367.043	12.766.217	2.144126
471.705	8.506.730	8.506.730	32.369.600	5.495974

5.1.3. Arquivo aleatório (PROVAO)

n	Leituras	Escritas	Comparações	Tempo (s)
100	583	583	2.423	0.000686
200	1.426	1.426	5.624	0.001886
2.000	20.716	20.716	59.732	0.017240
20.000	272.808	272.808	602.650	0.160687
200.000	3.405.233	3.405.233	7.168.199	2.034012
471.705	8.647.284	8.647.284	18.481.805	5.345587

5.2. Discussão do desempenho (método 3)

- **Leituras/Escritas:** No Quicksort Externo, leituras e escritas crescem com n e com o número de partições realizadas. Observa-se que o método manteve valores de E/S muito menores que o método 2 para o caso aleatório em n grande.
- **Comparações:** As comparações variam com a distribuição (ordem inicial), mas permanecem em patamar compatível com o comportamento esperado para

partições recursivas. No caso aleatório, o método tende a ser mais estável do que o método 2 para n grande.
- **Tempo:** Para n = 471.705, o tempo do Quicksort ficou na faixa de ~5,3–5,5 s, muito inferior ao método 2 no caso aleatório (~70,7 s).

---

6. Conclusão

Com base nos experimentos realizados, foi possível observar diferenças significativas de desempenho entre os três métodos de ordenação externa analisados.

O método 1 (intercalação balanceada com 2f fitas) apresentou bom desempenho em situações favoráveis, especialmente quando o arquivo de entrada já se encontrava ordenado ascendentemente. Nesses casos, as transferências de leitura e escrita cresceram de forma aproximadamente linear com o aumento de n. Entretanto, para arquivos descendentes e, principalmente, aleatórios, o método mostrou uma degradação acentuada de desempenho, com crescimento elevado no número de acessos à memória externa. Para entradas aleatórias com n = 200.000 e n = 471.705, os experimentos foram concluídos, porém com custo de E/S significativamente maior e maior tempo de execução em comparação ao caso ascendente, evidenciando a sensibilidade dessa abordagem a distribuições desfavoráveis.

O método 2 (intercalação balanceada com f+1 fitas) apresentou desempenho intermediário. Em arquivos ordenados ou parcialmente ordenados, os resultados foram satisfatórios, com tempos reduzidos e crescimento controlado das métricas. No entanto, para arquivos aleatórios de grande porte, o método tornou-se bastante custoso, principalmente devido ao elevado número de leituras e escritas, o que impactou diretamente o tempo de execução.

Já o método 3 (Quicksort Externo) foi o que apresentou os melhores resultados gerais. Em praticamente todos os cenários testados, especialmente para arquivos aleatórios e valores elevados de n, o Quicksort Externo demonstrou maior eficiência, com tempos de execução significativamente menores e número de operações de E/S inferior ao observado no método 2, evidenciando sua adequação para a ordenação de grandes volumes de dados em memória externa.

Principais dificuldades encontradas

Entre as principais dificuldades na implementação dos métodos destacam-se:

- o controle rigoroso das operações de leitura, escrita e comparação, exigido para a correta coleta das métricas;
- a correta implementação da seleção por substituição e das etapas de intercalação nos métodos 1 e 2;
- no Quicksort Externo, o tratamento adequado das condições de fronteira na partição e o gerenciamento da área interna limitada;
- a necessidade de preservar os arquivos originais, resolvida com o uso de um arquivo temporário de trabalho (**WORK.bin**).

De modo geral, o trabalho permitiu uma compreensão aprofundada dos impactos da memória externa no desempenho dos algoritmos de ordenação e evidenciou a importância da escolha adequada do método de ordenação conforme o tamanho do arquivo e a distribuição inicial dos dados.
