# DOCUMENTAÇÃO: Biblioteca do Felipe

Brian Aikau Almeida Gomes

AEDS III

## 1 Introdução

O objetivo deste trabalho é solucionar o problema de ordenação, distribuição e organização de um dado conjunto de livros em uma biblioteca com um número definido de estantes, cuja capacidade também é pré-definida.

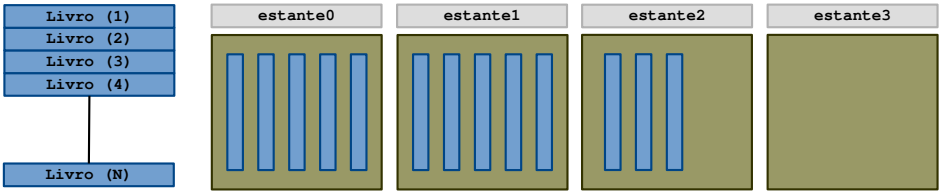



Figure 1: Ilustração dos N livros distribuídos nas estantes.

Além disso, é de interesse do bibliotecário realizar pesquisas de livros no acervo da biblioteca para verificar se algumas obras estão presentes no acervo e se estão disponíveis para o empréstimo. Para isso, todas as informações são passadas para o programa, que realiza as operações de ordenação e distribuição dos livros nas estantes para então serem usadas na busca por um livro.

## 2 Solução do Problema

### 2.1 Processamento inicial da entrada padrão:

As informações acerca do problema são dadas na entada padrão. Na primeira linha, são dados os valores: número total de livros **(N)**, capacidade da memória **(M)**, número de estantes na biblioteca **(E)**, a capacidade de armazenamento das estantes **(L)** e a quantidade de consultas a serem realizadas **(K)**. Após o registro desses valores, o programa registra as próximas informações na entrada, que, no caso, são o nomes e a disponibilidade dos livros. Os títulos e disponibilidades são escritos em um arquivo chamado *lista_de_livros*.

## 2.2 Ordenação dos livros (lista_de_livros):

O próximo processo do programa é ordenar os livros que estão na lista. Para isso, é lido um bloco de $M$ livros do arquivo *lista_de_livros*. Cada bloco lido é ordenado na memória principal e depois o bloco, já ordenado, é escrito em um dos dois arquivos iniciais do **MergeSort Externo**.

### 2.2.1 MergeSort Externo

O MergeSort Externo usa 4 dispositivos auxiliares (2 dispositivos de leitura e 2 dispositivos de escrita) para realizar a ordenação. Primeiro, blocos iniciais (de tamanho $M$) são ordenados e escritos nos dispositivos até que todos os registros tenham sido repassados para os dispositivos auxiliares.

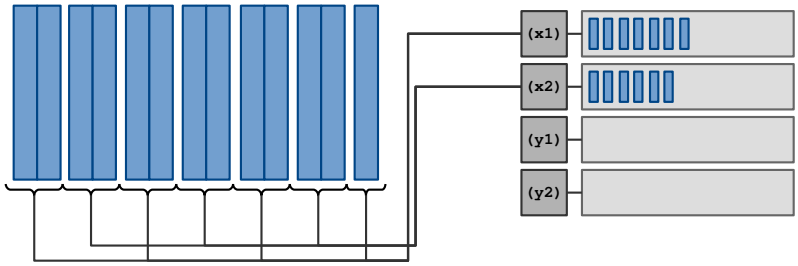



Figure 2: Preparação para o processo de intercalação com $\mathbf{(M)} = 2$.

Após esse processo, blocos de ambos dois dispositivos **(x1)** e **(x2)** são intercalados nos dispositivos **(y1)** e **(y2)**, construindo um bloco ordenado com o dobro de tamanho. Esse processo é repetido até que um dos dois dispositivos de leitura esteja vazio. O resultado desse *sorting* é um dispositivo com todos os registros ordenados.

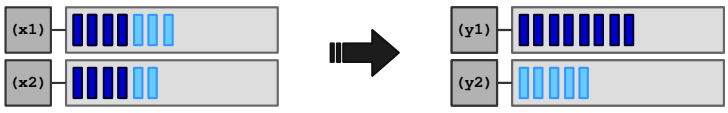



Figure 3: Ilustração de uma etapa de intercalação de bloco tamanho 4.

No programa, os dispositivos do MergeSort são arquivos temporários. No final da função de ordenação, o arquivo que contém todos os livros é renomeado para *livros_ordenados*, que será usado na distribuição dos livros na estante.

### 2.3 Distribuição dos livros nas estantes:

Os livros, já ordenados, são distribuídos nas $E$ estantes disponíveis. A regra é que toda a capacidade $L$ das estantes iniciais seja usada. Portanto, existe um ciclo para que todos os livros sejam distribuídos nas estantes.

### 2.4 Criação do arquivo índice:

O arquivo índice e criado abrindo o arquivo de cada estante e efetuando a leitura do primeiro e do último livro neste arquivo. Os dois livros lidos são guardados na memória principal e logo após são escritos no arquivo índice.

Os arquivos das estantes que estão vazios (estantes com nenhum livro) são identificados com sinal de # no índice.

### 2.5 Consultas de livros e busca binária:

Neste momento da execução do programa, as $K$ consultas são processadas. Cada título lido na entrada deve ser procurado nas estantes. Para facilitar essa busca, o arquivo índice é processado e indica a estante[1] em que o livro com o tal título está. Assim, o arquivo da estante indicada e aberto é então inicia-se o processo de busca binária no arquivo.

O processo de busca binária inicia no meio do arquivo da estante, onde o título em busca é comparado com o título no meio do bloco de busca. Se o título é o mesmo, a busca teve sucesso. Caso contrário, o título ainda é comparado para decidir o lado da subdivisão do bloco de busca.

No processo de busca, são possíveis três resultados: o livro não está no acervo, o livro está no acervo, mas está emprestado e o livro está no acervo e também está disponível para empréstimo. Nesse último caso, é indicada a estante que o livro ocupa e a posição do livro na estante.

## 3 Análise teórica de Complexidade Assintótica

### 3.1 Complexidade Assintótica de tempo:

A primeira operação do programa é registrar todos os livros do acervo em um arquivo chamado *lista_de_livros*. Seja N a quantidade de livros no acervo, a complexidade para esse procedimento é *O(N)*. O próximo procedimento é preparar os arquivos para a ordenação. São resgatados do arquivo *lista_de_livros* blocos de tamanho $M$ para a realizaao da ordenação interna.

[1] Somente se for possível que ele esteja em alguma estante, por exemplo: um livro com o título *computacaao_grafica* não estaria em um acervo com apenas uma estante que vai de *algoritmos_complexos* até *algoritmos_em_c*.

Portanto, o número de acessos a memória externa durante esse processo é dado por *O(N/M)*.

Com os arquivos de ordenação preparados, os acessos a memória externa são medidos por passagens[2] nos arquivos. Cada fase do MergeSort Externo faz uma passagem pelos dois arquivos de leitura (e a soma de todo conteúdo dos dois arquivos totaliza $N$ registros). O interessante do método é estipular a quantidade de passadas **(P)** necessárias para ordenar uma lista de livros dado tamanho $N$ registros.

Para analizar a quantidade de passagens nos arquivos, é preciso primeiro ponderar o processo de preparação dos arquivos para a intercalação. O processo entrega os dois arquivos de leitura iniciais com blocos de tamanho $M$ já ordenados. O tamanho M é considerado o tamanho do bloco inicial para a intercalação. Após cada etapa de intercalação, o tamanho do bloco é dobrado até que atinja um valor[3] maior ou igual a *N*.

A fórmula que expressa essa relação e dada por:

$$N \geq \log_2 \frac{N}{M} \approx P$$

Para a busca de livros no acervo, os livros são processados um-por-um. Cada livro é procurado no arquivo *indice*. Esse processamento inicial tem complexidade definida pelo número de estantes na biblioteca. Sendo assim, essa e uma operação *O(E)*. Depois, e se possível, é encontrada uma estante na qual seja possível que o livro esteja, a busca binária na estante, com tamanho máximo $L$ tem a odem de complexidade *O(log(L))*, no pior caso.

### 3.2 Complexidade Assintótica de espaço:

O processo de ordenação requer, no mínimo, $M = 2$ para funcionar. Seja $M$ a capacidade da de memória principal de armazenar os livros, o processo de ordenação usa $M$ no processo de preparação dos arquivos para a intercalação e o espao de dois livros para realizar a intercalação do MergeSort Externo. Portanto, a complexidade de espaço da memória principal é dada por *O(M)*.

Considerando a quantidade $N$ de livros no acervo, o custo de espaço na memória secundária para executar o processo de ordenação é *2N*, pois os arquivos temporários de leitura e de escrita têm todos os livros registrados

[2]Número de vezes que todo conteúdo de um arquivo é lido pelo programa.

[3]Quanto o tamanho do bloco a ser intercalado superar o tamanho de registros a serem ordenados, significa que já deve existir um arquivo que sozinho contenha todos os registros completamente ordenados.

neles no final de cada processo de intercalação. A complexidade é, portanto, linear e definida por *O(N)*.

Os processos no final do MergeSort interno, o arquivo *livros_ordenados* tem $N$ livros, que são distribuídos nas $E$ estantes, cujo resultado do somatório da quantidade de livros nas estantes é exatamente $N$. Seja a função *T(x)* uma função que retorne a quantidade de livros na estante $x$, então:

$$\sum_{i=0}^{E-1} T(i) = N$$

Sendo assim, no final do programa, o espaço de memória secundária necessário é de *2N*.

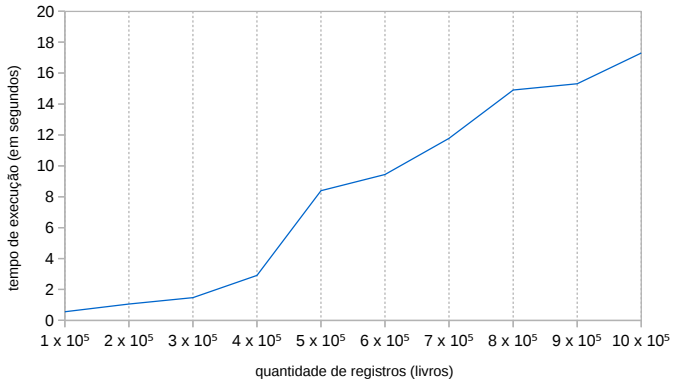
# 4 Análises Experimentais

Os testes foram realizados em um computador com processador Intel Core i3-3217U (Núcleos de 1.8GHz, 3MB de cache) e 2GB de memória RAM DDR3. As entradas usadas são identificadas pelas variáveis *N, M, E, L e K*. Primeiro, foi realizada uma bateria de testes sem o processo de busca, somente para analizar o custo de tempo para ordenar e registrar os livros nas estantes.

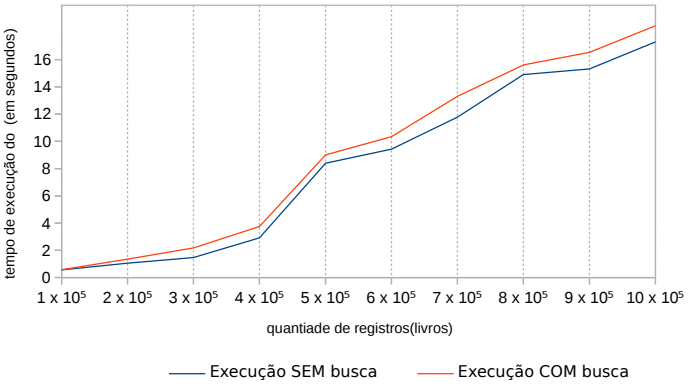
1. teste_01
    - **1000** 10 20 **60** 0
    - Tempo de execução do programa: `0m0.014s`
2. teste_02
    - **10000** 10 20 **600** 0
    - Tempo de execução do programa: `0m0.085s`
3. teste_03
    - **100000** 10 20 **6000** 0
    - Tempo de execução do programa: `0m0.565s`
4. teste_04
    - **1000000** 10 20 **60000** 0
    - Tempo de execução do programa: `17.304s`

## 4.1 Testes de ordem $10^5$

O tempo entre as entradas com $N = 10^5$ e $10^6$ foi analizado[4] com acréssimos mais granulados. Os resultados obtidos nos testes granulados podem ser visualizados no gráfico a seguir:




Foram feitos novos testes em cima dos casos analizados acima. Nos novos testes, foram inseridas buscas na ordem de $10^3$. A variação do tempo com essa busca pode ser vista no gráfico no próximo gráfico.




Com o gráfico acima, é possível examinar o tempo das duas principais etapas do programa: ordenação e busca. A ordenação consome a maior parte do tempo de execução, enquanto o processo de busca faz apenas um acréssimo ao tempo total do programa.

[4] O valor de L foi definido para 60000 para todos esses casos.