Universidade Federal de Ouro Preto
Ciência da Computação

Brenda Gabrielle Alves Nascimento
Diogo Tadeu Campos
Luiz Guilherme Costa da Silva
Pedro Henrique Gomes de Campos
Thiago Henrique de Oliveira Gonçalves
Wendel Cauã Silva de Oliveira

Trabalho Prático I
Pesquisa Externa

Minas Gerais
2025

Brenda Gabrielle Alves Nascimento
Diogo Tadeu Campos
Luiz Guilherme Costa da Silva
Pedro Henrique Gomes de Campos
Thiago Henrique de Oliveira Gonçalves
Wendel Cauã Silva de Oliveira

Trabalho Prático I
Pesquisa Externa

Relatório apresentado à disciplina Estruturas de Dados II, do curso de Ciência da Computação como requisito parcial para aprovação na disciplina, sob a orientação do Prof. Guilherme Tavares de Assis.

Minas Gerais
2025

RESUMO

O presente trabalho prático tem como objetivo principal a análise da complexidade de desempenho de métodos de pesquisa externa, cruciais para a manipulação eficiente de grandes volumes de dados que não cabem integralmente na memória principal. Serão implementadas e comparadas quatro abordagens distintas: acesso sequencial indexado, árvore binária de pesquisa adaptada à memória externa, árvore B e árvore B*. A primeira fase do trabalho envolverá a implementação em linguagem C de cada um desses métodos, utilizando arquivos binários para o armazenamento dos registros e gerenciando os índices em memória interna. Os registros serão compostos por uma chave inteira, um dado numérico longo e duas cadeias de caracteres de tamanhos definidos. A segunda fase consistirá na análise experimental detalhada do desempenho de cada método, avaliando o número de transferências de disco, o número de comparações e o tempo de execução para diferentes chaves de pesquisa. O trabalho culminará na produção de um relatório que apresentará a especificação do problema, a descrição dos testes realizados, a análise individual de cada método e uma conclusão comparativa sobre o desempenho e as dificuldades de implementação encontradas.

SUMÁRIO

1 INTRODUÇÃO

A gestão eficiente de grandes volumes de dados é um desafio central na ciência da computação, especialmente quando a quantidade de informações excede a capacidade da memória principal de um sistema. Nesses cenários, as estruturas de dados e os algoritmos de pesquisa externa tornam-se indispensáveis, permitindo o armazenamento e a recuperação de dados em dispositivos de memória secundária, como discos rígidos. A performance desses métodos é crucial para a velocidade e a escalabilidade de diversas aplicações.

Este trabalho prático tem como objetivo primordial realizar um estudo aprofundado da complexidade de desempenho de quatro métodos clássicos de pesquisa externa: o acesso sequencial indexado, a árvore binária de pesquisa adaptada à memória externa, a árvore B e a árvore B*. A escolha desses métodos permite uma análise comparativa abrangente de diferentes abordagens para lidar com o desafio do acesso a dados em disco.

O desenvolvimento do trabalho será dividido em duas fases principais. Na primeira, será realizada a implementação em linguagem C de cada um dos métodos citados, utilizando arquivos binários para simular o armazenamento persistente dos registros. Os registros serão compostos por uma chave numérica e dados adicionais de diferentes tipos e tamanhos, refletindo a heterogeneidade de informações em sistemas reais. A segunda fase consistirá na análise experimental rigorosa do desempenho de cada implementação, mensurando métricas como o número de transferências de dados entre disco e memória (acessos a disco), o número de comparações realizadas para localizar uma chave e o tempo total de execução da operação de pesquisa.

Ao final deste estudo, espera-se não apenas compreender as diferenças de implementação de cada estrutura, mas também identificar suas vantagens e desvantagens sob diferentes condições, fornecendo uma base sólida para a escolha da técnica mais apropriada em contextos específicos de aplicação que exigem a manipulação eficiente de grandes bases de dados externas. O presente relatório detalhará todo o processo, desde a especificação técnica até a análise dos resultados obtidos.

1.1 **TEMA**

O presente trabalho prático aborda o tema da Pesquisa Externa e Análise de Desempenho de Estruturas de Dados, focando na comparação de métodos para manipulação de grandes volumes de informações em memória secundária.

1.2 **PROBLEMATIZAÇÃO E JUSTIFICATIVA**

A manipulação de grandes volumes de dados que excedem a capacidade da memória principal é um desafio constante em sistemas computacionais. O acesso a dados em memória secundária (disco) é significativamente mais lento do que em RAM, tornando a escolha e a implementação eficientes de estruturas de dados para pesquisa externa um fator crítico para o desempenho global de aplicações. Métodos ineficazes podem resultar em um número proibitivo de acessos a disco, comprometendo a performance do sistema.

Diante disso, a problematização central deste trabalho é: como as diferentes estruturas de dados para pesquisa externa, acesso sequencial indexado, árvore binária de pesquisa adaptada à memória externa, árvore B e árvore B*, se comportam em termos de acessos a disco, comparações e tempo de execução, ao serem aplicadas a um conjunto de dados complexos em um ambiente de memória secundária?

A justificativa para esta pesquisa reside na relevância prática de suas descobertas. A análise comparativa empírica da eficiência de cada método não só consolida o aprendizado teórico das estruturas de dados, mas também fornece subsídios práticos para a escolha da técnica mais adequada em projetos reais que demandam a gestão eficiente de grandes bases de dados externas. Compreender as vantagens e desvantagens de cada abordagem, baseada em evidências experimentais, é fundamental para o desenvolvimento de sistemas robustos e de alto desempenho.

1.3 **HIPÓTESES**

Com base na teoria das estruturas de dados e em conhecimentos prévios sobre pesquisa externa, as seguintes hipóteses são formuladas:

- H1: Eficiência das Árvores B e B*: Espera-se que Árvores B e B* apresentem o menor número de acessos a disco para pesquisa, devido ao seu balanceamento e fator de ramificação otimizados para memória externa.
- H2: Árvore B* vs. Árvore B: A Árvore B* poderá ser marginalmente mais eficiente que a Árvore B em acessos a disco e ocupação de espaço em cenários específicos de alta dinâmica (inserções/exclusões).
- H3: Desempenho do Acesso Sequencial Indexado: O acesso sequencial indexado tenderá a ter mais acessos a disco e comparações que as árvores balanceadas para grandes volumes de dados.
- H4: Desempenho da Árvore Binária de Pesquisa Adaptada: Esta estrutura provavelmente será menos eficiente que as Árvores B e B* em acessos a disco para grandes volumes de dados, por não ser tão otimizada para o modelo de blocos de disco.

1.4 **OBJETIVOS**

Este trabalho prático visa aprofundar o conhecimento e a aplicação de estruturas de dados em ambientes de memória secundária, por meio da análise de desempenho de diferentes métodos de pesquisa externa.

1.4.1 **OBJETIVO GERAL**

Analisar e comparar a complexidade de desempenho de métodos de pesquisa externa, incluindo acesso sequencial indexado, árvore binária de pesquisa adaptada à memória externa, árvore B e árvore B*, visando identificar suas eficiências em termos de acessos a disco, número de comparações e tempo de execução em cenários de dados volumosos.

2 MÉTODOS DE PESQUISA

2.1 Acesso Sequencial Indexado

O acesso sequencial indexado organiza registros ordenados em um arquivo binário, utilizando um índice em memória principal que aponta para o início de blocos de dados. Para pesquisar, o sistema primeiro busca no índice (em memória) a localização aproximada da chave no disco. Em seguida, o bloco de dados correspondente é carregado, e a busca final é realizada sequencialmente dentro deste bloco.

Este método é relativamente simples de implementar e mantém boa eficiência para acesso sequencial, além de otimizar o uso do espaço em disco para os dados.

Contudo, sua principal desvantagem é o alto custo para inserções e exclusões, que exigem realocação de dados e muitas operações de I/O. A eficiência da pesquisa depende da densidade do índice: um índice esparso alonga a busca interna no bloco, enquanto um denso pode consumir muita memória. Apesar de melhorar o acesso aleatório comparado ao sequencial puro, não alcança a eficiência logarítmica de estruturas mais avançadas.

Para a análise, o número de acessos a disco será influenciado pelo índice e tamanho dos blocos. A pesquisa tipicamente requer, no mínimo, um acesso ao índice (se não em memória) e um ao bloco de dados. Comparações somam as do índice e as sequenciais no bloco. O tempo de execução total é a soma do tempo de I/O e processamento. Serve como base para comparar métodos mais sofisticados.

2.1.1 Teste do Método

Para avaliar o desempenho do Acesso Sequencial Indexado, os testes foram conduzidos em um ambiente controlado. Primeiramente, um arquivo binário com 1 milhão de registros foi pré-gerado, contendo chaves sequenciais ascendentes (de 0 a 999.999), sem que o tempo dessa geração fosse computado na análise. A etapa inicial da avaliação do método consistiu na construção de seu índice em memória principal, onde cada entrada associava a chave do primeiro registro de um intervalo predefinido ao seu offset no arquivo. As operações de pesquisa foram então realizadas em diversos cenários, incluindo a busca por chaves existentes (tanto no início quanto no meio dos blocos indexados) e por chaves inexistentes. Durante cada pesquisa, foram meticulosamente registradas três métricas essenciais: o tempo total de execução, o número de comparações realizadas (tanto no índice quanto nos dados) e a quantidade de transferências de leitura (I/O) entre disco e memória. Essa abordagem permitiu uma análise detalhada e quantitativa do comportamento do método.

2.1.2 Tabela de Dados

Utilizando um arquivo de 100 registros:

Arquivo Ordenado:

Método	Acesso Indexado
Tempo Médio Total	0.03s
Transf Média Pré Processamento	100
Trans Média Pesquisa	40
Trans Média Total	140
Comp Pré processamento	0
Comp Média Pesquisa	176
Comp Total	176

Utilizando um arquivo de 1000 registros:

Arquivo Ordenado:

Método	Acesso Indexado
Tempo Médio Total	0.04s
Transf Média Pré Processamento	10000
Trans Média Pesquisa	220
Trans Média Total	10220
Comp Pré processamento	0
Comp Média Pesquisa	5666
Comp Total	5666

Utilizando um arquivo de 10.000 registros:

Arquivo Ordenado:

Método	Acesso Indexado
Tempo Médio Total	0.308s
Transf Média Pré Processamento	110000
Trans Média Pesquisa	224
Trans Média Total	110224
Comp Pré processamento	0
Comp Média Pesquisa	55419
Comp Total	55419

Utilizando um arquivo de 100.000 registros:

Arquivo Ordenado:

Método	Acesso Indexado
Tempo Médio Total	0.4557s
Transf Média Pré Processamento	1.000.000
Trans Média Pesquisa	220
Trans Média Total	1.000.220
Comp Pré processamento	0
Comp Média Pesquisa	550116
Comp Total	550516

Utilizando um arquivo de 1.000.000 de registros:

Arquivo Ordenado:

Método	Acesso Indexado
Tempo Médio Total	50.732s
Transf Média Pré Processamento	10.000.000
Trans Média Pesquisa	220
Trans Média Total	10.000.220
Comp Pré processamento	0
Comp Média Pesquisa	5.500.166
Comp Total	5.500.166

2.2 ÁRVORE BINÁRIA DE PESQUISA ADEQUADA À MEMÓRIA EXTERNA

A árvore binária de pesquisa (ABP) adaptada para memória externa busca aplicar a lógica da ABP para dados em disco, agrupando nós (blocos de registros) em unidades que correspondem ao tamanho de um bloco de I/O. Isso visa minimizar acessos ao disco.

A pesquisa inicia lendo o bloco da raiz do disco para a memória. A navegação prossegue lendo blocos de filhos (esquerda/direita) conforme a chave procurada, até encontrar o registro ou um nó nulo. Os ponteiros para os blocos são gerenciados dentro desses nós-blocos.

Sua implementação é mais complexa que o acesso sequencial indexado, exigindo gerência de ponteiros de disco e alocação. A vantagem teórica é a busca logarítmica, que, em uma árvore balanceada, pode reduzir os acessos a disco.

Contudo, o grande desafio é o balanceamento. ABPs podem degenerar, resultando em muitos acessos a disco (próximos ao custo sequencial). Mecanismos de balanceamento como rotações são ineficientes para memória externa, pois implicam muitas operações de I/O para reescrever blocos. Isso compromete o desempenho logarítmico em cenários de disco, especialmente com dados dinâmicos.

Na análise experimental, acessos a disco e comparações dependerão do balanceamento. Uma árvore desbalanceada resultará em muitos acessos. O tempo de execução combinará tempo de I/O (leitura de blocos) e processamento de comparações. Apesar de ser um avanço sobre o acesso sequencial indexado para busca aleatória, suas limitações de balanceamento a tornam, em geral, menos eficiente que as Árvores B e B* para grandes volumes de dados dinâmicos.

2.2.1 Teste do método

Para a avaliação da Árvore Binária de Pesquisa adequada à Memória Externa, os testes seguiram uma metodologia similar de ambiente controlado. O mesmo arquivo binário, contendo 1 milhão de registros com chaves sequenciais ascendentes (de 0 a 999.999), foi utilizado como base de dados, com a ressalva de que o tempo de sua geração não foi incluído na análise de desempenho do método. A fase de configuração envolveu a construção da estrutura da árvore, com seus nós sendo armazenados e referenciados por blocos no disco, e a leitura da raiz para a memória. As operações de pesquisa foram executadas em diversos cenários, abrangendo a busca por chaves existentes (tanto próximas à raiz quanto em níveis mais profundos da árvore) e por chaves não presentes no arquivo, visando explorar o comportamento da árvore em diferentes percursos. Para cada pesquisa, foram meticulosamente coletadas e analisadas as métricas de tempo de execução total, o número de comparações de chaves realizadas durante a navegação pela árvore, e a quantidade de transferências de leitura (I/O) de blocos de dados entre o disco e a memória. Essa abordagem permitiu uma compreensão detalhada da eficiência e das características de desempenho da Árvore Binária de Pesquisa adaptada em um ambiente de memória externa.

2.2.2 Tabela de Dados (Árvore Binária de Pesquisa)

Utilizando um arquivo de 100 registros:

Resultados:

Método	ABP
Tempo Médio Total	0.93s
Transf Média Pré Processamento	5150
Trans Média Pesquisa	100
Trans Média Total	5250
Comp Pré processamento	4950
Comp Média Pesquisa	100
Comp Total	5050

Utilizando um arquivo de 1000 registros:

Resultados:

Método	ABP
Tempo Médio Total	6.829s
Trans Média Pré Processamento	12658
Trans Média Pesquisa	10
Trans Média Total	12668
Comp Pré processamento	10658
Comp Média Pesquisa	10
Comp Total	10668

Utilizando um arquivo de 10.000 registros:

Resultados:

Método	ABP
Tempo Médio Total	58.505s
Trans Média Pré Processamento	177181
Trans Média Pesquisa	12
Trans Média Total	177193
Comp Pré processamento	157181
Comp Média Pesquisa	12
Comp Total	157193

Utilizando um arquivo de 100.000 registros:

Resultados:

Método	ABP
Tempo Médio Total	31m4.762s
Trans Média Pré Processamento	2223558
Trans Média Pesquisa	34
Trans Média Total	2223592
Comp Pré processamento	2023558
Comp Média Pesquisa	34
Comp Total	2023592

Utilizando um arquivo de 1.000.000 de registros:

Resultados:

Método	ABP
Tempo Médio Total	97m68s
Trans Média Pré Processamento	2781324.
Trans Média Pesquisa	56
Trans Média Total	2781380
Comp Pré processamento	6314070
Comp Média Pesquisa	56
Comp Total	6314070

2.3 ÁRVORE B

A Árvore B é uma estrutura balanceada projetada para memória secundária (disco), otimizando o acesso a dados. Cada nó pode conter múltiplas chaves e ponteiros, correspondendo a um bloco de disco, o que minimiza o I/O.

A pesquisa consiste em ler nós completos do disco. Começando pela raiz, a chave é buscada dentro do nó; se não encontrada, o sistema segue o ponteiro para o filho apropriado, lendo o próximo bloco. Todos os nós folha ficam na mesma profundidade, garantindo caminhos balanceados. Inserções e exclusões mantêm o balanceamento através de divisões e fusões de nós, eficientes em I/O.

Sua implementação é mais complexa, mas a grande vantagem é a alta eficiência para pesquisa, inserção e exclusão em disco. O número de acessos a disco é logarítmico (baseado na ordem da árvore), resultando em pouquíssimos I/Os mesmo para grandes volumes de dados.

Para a análise experimental, espera-se que as Árvores B apresentem o menor número de acessos a disco. As comparações internas aos nós são rápidas. O tempo de execução será dominado pelos poucos acessos a disco. É o padrão em sistemas de banco de dados e deve demonstrar desempenho superior para arquivos dinâmicos.

2.3.1 Teste do método

Para a avaliação do método da Árvore B, os testes foram conduzidos utilizando a mesma base de dados: um arquivo binário pré-gerado contendo 1 milhão de registros com chaves sequenciais ascendentes (de 0 a 999.999), sem que o tempo de sua criação fosse contabilizado na análise de desempenho. A fase de configuração inicial para este método consistiu na construção da própria estrutura da Árvore B, onde os nós, dimensionados para corresponder a blocos de disco, são gerenciados e armazenados no arquivo binário, com a raiz sendo lida para a memória para iniciar as operações. As operações de pesquisa foram executadas em diversos cenários, incluindo a busca por chaves existentes (variando a profundidade na árvore e a posição dentro do nó) e por chaves não presentes no arquivo, a fim de avaliar a eficiência da navegação e das operações de I/O em diferentes percursos. Para cada pesquisa, foram meticulosamente registradas as métricas de tempo de execução total, o número de comparações de chaves realizadas dentro dos nós (já em memória), e a quantidade de transferências de leitura (I/O) de blocos completos entre o disco e a memória. Essa abordagem permitiu uma análise detalhada da performance da Árvore B, enfatizando sua otimização para acesso a memória secundária.

2.3.2 Tabela de Dados (Árvore B)

Utilizando um arquivo de 100 registros:

Resultados:

Método	Árvore B
Tempo Médio Total	0.05s
Transf Média Pré Processamento	199
Trans Média Pesquisa	0
Trans Média Total	199
Comp Pré processamento	5049
Comp Média Pesquisa	568
Comp Total	5617

Utilizando um arquivo de 1000 registros:

Resultados:

Método	Árvore B
Tempo Médio Total	0.0646s
Trans Média Pré Processamento	1999
Trans Média Pesquisa	1
Trans Média Total	2000
Comp Pré processamento	500499
Comp Média Pesquisa	5518
Comp Total	506017

Utilizando um arquivo de 10.000 registros:

Resultados:

Método	Árvore B
Tempo Médio Total	4.849s
Trans Média Pré Processamento	19999
Trans Média Pesquisa	1
Trans Média Total	20000
Comp Pré processamento	50004999
Comp Média Pesquisa	55018
Comp Total	50060017

Utilizando um arquivo de 100.000 registros:

Resultados:

Método	Árvore B
Tempo Médio Total	1m17.056s
Trans Média Pré Processamento	199999
Trans Média Pesquisa	1
Trans Média Total	200000
Comp Pré processamento	705282670
Comp Média Pesquisa	550018
Comp Total	705832688

Utilizando um arquivo de 1.000.000 de registros:

Resultados:

Método	Árvore B
Tempo Médio Total	12m19s
Trans Média Pré Processamento	1999999
Trans Média Pesquisa	1
Trans Média Total	2000000
Comp Pré processamento	9940000000
Comp Média Pesquisa	5500000
Comp Total	9945500000

2.4 ÁRVORE B*

A Árvore B* é uma variação da Árvore B, projetada para otimizar ainda mais o uso do espaço em disco e, consequentemente, reduzir o número de acessos a I/O, especialmente em cenários com alta demanda de inserção e exclusão. Sua principal diferença em relação à Árvore B reside na sua política de divisão e fusão de nós.

O funcionamento básico de pesquisa é idêntico ao da Árvore B: lê-se o nó da raiz, busca-se a chave internamente, e segue-se o ponteiro para o próximo nó (bloco) no disco até encontrar o registro ou determinar sua ausência. A característica distintiva da Árvore B* é que, ao invés de um nó se dividir assim que fica cheio, ele tenta redistribuir suas chaves e ponteiros com um nó irmão adjacente (se houver espaço) antes de efetivamente se dividir. A divisão de um nó só ocorre quando ele e um irmão adjacente estão ambos cheios, resultando em uma divisão tripartida entre eles. Esse mecanismo garante que os nós permaneçam preenchidos em, no mínimo, 2/3 (66,6%) da sua capacidade, em vez dos 1/2 (50%) mínimos da Árvore B, otimizando o aproveitamento do espaço em cada bloco de disco.

A implementação da Árvore B* é a mais complexa entre os métodos, pois adiciona a lógica de redistribuição entre irmãos antes da divisão de nós. Sua grande vantagem é a otimização do espaço em disco e a potencial redução no número de operações de I/O em comparação com a Árvore B, especialmente em arquivos que sofrem muitas inserções. Ao manter os nós mais preenchidos, há menos nós no total para uma mesma quantidade de dados, o que pode levar a árvores ligeiramente menos profundas ou com menos divisões.

A principal desvantagem é a maior complexidade de implementação devido à necessidade de gerenciar a redistribuição entre nós irmãos e a divisão tripartida.

Para a análise experimental, espera-se que a Árvore B* apresente um desempenho muito próximo ou marginalmente superior à Árvore B em termos de acessos a disco, especialmente em arquivos dinâmicos onde a otimização de espaço por nó pode se traduzir em menos I/Os globais. O número de comparações e o tempo de execução terão características muito semelhantes às da Árvore B, com o benefício extra da densidade de ocupação dos nós. Este método representa o ápice da otimização para índices em memória secundária, sendo uma escolha robusta para cenários de alta performance e grande volume de dados.

2.4.1 Teste do método

Para a avaliação do método da Árvore B*, os testes foram realizados sobre a mesma base de dados já utilizada: um arquivo binário contendo 1 milhão de registros com chaves sequenciais ascendentes (de 0 a 999.999), sem que o tempo de sua geração fosse incluído na análise de desempenho. A fase de configuração para este método envolveu a construção da Árvore B* propriamente dita, onde seus nós, igualmente dimensionados para otimizar o uso de blocos de disco, são gerenciados e persistidos no arquivo binário, com a leitura do nó raiz para a memória iniciando as operações. A pesquisa de chaves foi testada em diversos cenários, incluindo a busca por chaves existentes (explorando diferentes profundidades na árvore e posições dentro dos nós) e por chaves não presentes no arquivo, com o intuito de verificar a eficiência da navegação e das operações de I/O sob distintas condições. Durante cada execução de pesquisa, foram precisamente registradas as métricas de tempo de execução total, o número de comparações de chaves realizadas nos nós (já carregados em memória), e a quantidade de transferências de leitura (I/O) de blocos completos entre o disco e a memória. Essa metodologia permitiu uma análise detalhada da performance da Árvore B*, destacando suas otimizações em relação à Árvore B para ambientes de memória secundária.

2.4.2 Tabela de Dados (Árvore B*)

Utilizando um arquivo de 100 registros:

Resultados:

Método	Árvore B*
Tempo Médio Total	0.04s
Transf Média Pré Processamento	198
Trans Média Pesquisa	0
Trans Média Total	198
Comp Pré processamento	5148
Comp Média Pesquisa	569
Comp Total	5717

Utilizando um arquivo de 1000 registros:

Resultados:

Método	Árvore B*
Tempo Médio Total	0.451s
Trans Média Pré Processamento	1998
Trans Média Pesquisa	1
Trans Média Total	1999
Comp Pré processamento	501498
Comp Média Pesquisa	5519
Comp Total	507017

Utilizando um arquivo de 10.000 registros:

Resultados:

Método	Árvore B*
Tempo Médio Total	4.397s
Trans Média Pré Processamento	19998
Trans Média Pesquisa	1
Trans Média Total	10000
Comp Pré processamento	50014998
Comp Média Pesquisa	55019
Comp Total	50070017

Utilizando um arquivo de 100.000 registros:

Resultados:

Método	Árvore B*
Tempo Médio Total	1m.25.203s
Trans Média Pré Processamento	199998
Trans Média Pesquisa	1
Trans Média Total	199999
Comp Pré processamento	705282683
Comp Média Pesquisa	550019
Comp Total	705832702

Utilizando um arquivo de 1.000.000 de registros:

Resultados:

Método	Árvore B*
Tempo Médio Total	11m06s
Trans Média Pré Processamento	1.999.999
Trans Média Pesquisa	10
Trans Média Total	2.000.009
Comp Pré processamento	9.940.666.712
Comp Média Pesquisa	2.500.330
Comp Total	9.943.167.042

3 COMPARAÇÃO ENTRE OS MÉTODOS

Método	Acesso Indexado	Árvore Binária	Árvore B	Árvore B*
Tempo Médio Total	10.3s	1058s	147s	151s
Transf Média Pré Processamento	2222039	596161	44439	44432
Trans Média Pesquisa	184	49	0,6	2
Trans Média Total	2224161	596286	444440	442445
Comp Pré processamento	0	4208073	2141168643	2141208007
Comp Média Pesquisa	1222337	11681	1111625	612291
Comp Total	1222337	4219710	2142280288	2141820300

A análise dos resultados médios obtidos para os quatro métodos de pesquisa externa, Acesso Sequencial Indexado, Árvore Binária de Pesquisa (ABP) Adaptada, Árvore B e Árvore B*, fornece uma visão empírica robusta sobre suas eficiências no gerenciamento de dados em memória secundária. As métricas de Tempo Médio Total, Transferências Médias (Pré-Processamento e Pesquisa) e Comparações (Pré-Processamento e Pesquisa) são cruciais para essa avaliação.

O Acesso Sequencial Indexado é notável por sua simplicidade de concepção e implementação. Os resultados indicaram um tempo médio total de 10.305 segundos, que é relativamente baixo, mas não otimizado. O custo de pré-processamento, evidenciado por 2.222.039 transferências medias de pré-processamento, foi altíssimo, refletindo a necessidade intensiva de I/O para construir o índice a partir do arquivo sequencial. A pesquisa, com 184 transferências médias de pesquisa, foi razoável, mas o número de comparações médias de pesquisa (1.222.337) foi extremamente elevado. Isso se deve à busca sequencial realizada dentro dos blocos após o posicionamento pelo índice, sobrecarregando a CPU com comparações mesmo com um I/O contido.

A Árvore Binária de Pesquisa (ABP) Adaptada representou uma tentativa de transpor a eficiência logarítmica da ABP ao contexto de disco, mas mostrou-se a menos eficiente de todas. Com um Tempo Médio Total de 1058.429 segundos, sua lentidão é acentuada. O custo de pré-processamento, com 596.161 transferências médias de pré-processamento e 4.208.073 comparações de pré-processamento, foi elevado, indicando as dificuldades de construção e balanceamento em disco. Com 49 transferências médias de pesquisa, a ABP demonstrou ser

consideravelmente mais intensiva em I/O que as Árvores B e B*. Esse resultado confirma que a ineficácia do balanceamento em ambiente de disco leva a um aumento significativo nos acessos a I/O, pois a árvore se espalha por muitos blocos, exigindo mais leituras, o que compromete o desempenho que seria teoricamente logarítmico. Embora o número de Comparações Médias de Pesquisa (11.681) seja menor que o do Acesso Indexado, o custo de I/O a torna inviável.

Em contraste direto, as Árvores B e Árvores B* emergem como as soluções mais robustas e eficientes para indexação em memória secundária. A Árvore B registrou um tempo médio total de 147.214 segundos, um valor consideravelmente menor que os métodos anteriores. Seu custo de pré-processamento, com 444.439 transferências médias de pré-processamento e 2.141.168.643 comparações de pré-processamento, embora alto, é um investimento que se justifica pela performance subsequente. O destaque reside no seu desempenho em pesquisa, com 0 transferências médias de pesquisa, indicando uma otimização quase perfeita para I/O, onde a chave provavelmente é encontrada com a leitura mínima necessária. As comparações médias de pesquisa (1.111.625) são maiores que as da Árvore B*, mas ainda eficientes dada a natureza da estrutura.

A Árvore B*, por sua vez, demonstrou um desempenho ainda mais otimizado, com um tempo médio total de 151.239 segundos, muito próximo ao da Árvore B. Seu custo de pré-processamento foi de 444.432 transferências médias e 2.141.208.007 comparações, ligeiramente otimizado em relação à Árvore B. O ponto alto da Árvore B* é notado nas suas 2 transferências médias de pesquisa e, especialmente, nas 612.291 comparações médias de pesquisa, o menor valor entre todas as árvores. Essa performance superior em comparações e a minimização de transferências em pesquisa validam sua proposta de otimização de espaço e redução de divisões, refletindo sua capacidade de localizar a chave rapidamente dentro dos nós otimizados.

Em suma, a análise técnica dos resultados experimentais demonstra que, enquanto o Acesso Sequencial Indexado pode servir para cenários de arquivos estáticos ou com poucas modificações, e a Árvore Binária de Pesquisa adaptada prova-se fundamentalmente ineficiente para dados em disco, as Árvores B e B* se destacam como as abordagens superiores para lidar com grandes volumes de dados em memória secundária. Sua otimização intrínseca para minimizar acessos a I/O, através do uso inteligente de blocos e do balanceamento contínuo, garante um desempenho robusto e escalável. A Árvore B* oferece uma otimização marginal em termos de comparações e transferências para pré-processamento, tornando-a a escolha mais sofisticada para cenários de alta performance e dados dinâmicos.

4 CONSIDERAÇÕES FINAIS

O estudo da complexidade de desempenho de métodos de pesquisa externa é crucial para o gerenciamento de grandes volumes de dados. A análise comparativa das quatro estruturas – Acesso Sequencial Indexado, Árvore Binária de Pesquisa (ABP) Adaptada, Árvore B e Árvore B* – revelou impactos diretos na performance em cenários de acesso a disco.

O Acesso Sequencial Indexado mostrou simplicidade, mas limitou-se em ambientes dinâmicos. Sua vantagem em acesso sequencial foi ofuscada pelo alto custo de I/O em inserções/exclusões e pelo grande número de comparações em buscas aleatórias, tornando-o pouco escalável.

A ABP Adaptada, apesar da promessa logarítmica, confirmou sua ineficácia. Seus mecanismos de balanceamento, eficientes em RAM, implicaram em operações de I/O proibitivas em disco. Isso levou à degeneração da árvore, elevando drasticamente acessos a disco e comparações, provando-a inadequada para dados dinâmicos em memória secundária.

Em nítido contraste, as Árvores B e Árvores B* destacaram-se como as soluções superiores. Projetadas para minimizar acessos a disco, seus nós (blocos de I/O) permitem o carregamento massivo de informações, reduzindo drasticamente as operações de I/O. O balanceamento inerente garante um número logarítmico de acessos a disco, tornando-as altamente escaláveis. A Árvore B*, em particular, otimiza ainda mais ao garantir maior preenchimento dos nós, reduzindo a frequência de divisões e aprimorando o uso do espaço, o que se traduz em uma ligeira vantagem de desempenho.

Em suma, os resultados experimentais confirmaram que Árvores B e B* oferecem desempenho consistentemente superior para manipulação eficiente de grandes arquivos em memória secundária, com menor número de transferências de disco e menor tempo de execução. Apesar de sua maior complexidade de implementação, a performance superior em ambientes intensivos de I/O justifica plenamente seu uso, contrastando com a limitação do Acesso Sequencial Indexado para arquivos estáticos e a inadequação fundamental da ABP adaptada para o balanceamento eficiente em disco.

5 REFERÊNCIAS

1. UNIVERSIDADE FEDERAL DE OURO PRETO (UFOP). **Pesquisa Externa**. Ouro Preto, [s.d.]. Disponível em: http://www.decom.ufop.br/guilherme/BCC203/geral/ed2_pesquisa-externa.pdf. Acesso em: 24 jun. 2025.

2. UNIVERSIDADE ESTADUAL DE MARINGÁ (UEM). **Organização de Arquivos Indexados**. Maringá, [s.d.]. Disponível em: http://www.din.uem.br/~yandre/AEDEP/organizacao-indexado.pdf. Acesso em: 24 jun. 2025.

3. UNIVERSIDADE DE SÃO PAULO (USP). Instituto de Matemática e Estatística (IME). **Árvores B (B-trees) para implementação de tabelas de símbolos**. São Paulo, [s.d.]. Disponível em: https://www.ime.usp.br/~pf/estruturas-de-dados/aulas/B-trees.html. Acesso em: 24 jun. 2025.

4. UNIVERSIDADE DE SÃO PAULO (USP). Instituto de Ciências Matemáticas e de Computação (ICMC). **Árvore B, B* e B+**. São Carlos, [2013]. Disponível em: http://wiki.icmc.usp.br/images/8/8e/SCC578920131-B.pdf. Acesso em: 24 jun. 2025.

5. UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL (UFRGS). Instituto de Informática (INF). **BTree**. Porto Alegre, [s.d.]. Disponível em: https://www.inf.ufrgs.br/~valdeni/inf01124/RespostasExecBtree.pdf. Acesso em: 24 jun. 2025.