UNIVERSIDADE FEDERAL DO PARANÁ
DEPARTAMENTO DE INFORMÁTICA

Disciplina: Algoritmos III
Professor: Andrey Ricardo Pimentel

Relatório de Implementação e Análise: Dimensionamento do Tamanho dos Nós de uma Árvore B+

Autor: Pietro Comin
pietro.comin@ufpr.br
GRR20241955

Curitiba
3 de julho de 2025

Resumo

Este relatório detalha a minha implementação e análise de desempenho de uma estrutura de dados Árvore B+. O objetivo principal foi investigar o impacto da ordem da árvore (tamanho do nó) no tempo médio de busca para diferentes volumes de dados. Para isso, conduzi dois experimentos principais: o primeiro utilizando uma busca linear interna nos nós, e o segundo, uma busca binária otimizada. Os resultados do primeiro experimento revelaram, de forma contraintuitiva, que ordens menores apresentavam melhor desempenho, o que atribuí ao alto custo computacional da busca linear em nós grandes. O segundo experimento, com busca binária, validou a teoria clássica, revelando uma curva de desempenho em "U", onde o ponto ótimo de performance é alcançado através do equilíbrio entre a altura da árvore e o custo de busca no nó. Meu estudo conclui que a escolha do algoritmo de busca interna é um fator crítico e determinante para o dimensionamento ótimo dos nós em implementações de Árvores B+ em memória.

Palavras-chave: Árvore B+, Estrutura de Dados, Análise de Algoritmos, Otimização de Desempenho, Busca Binária, Gerenciamento de Memória.

Sumário

1 Introdução

A Árvore B+ é uma estrutura de dados em árvore auto-balanceada, amplamente utilizada na implementação de sistemas de arquivos e bancos de dados. Sua principal característica é a otimização para sistemas que leem e escrevem grandes blocos de dados, minimizando o número de acessos a dispositivos de armazenamento secundário, como discos rígidos (I/O). A estrutura garante que todos os dados residam nos nós folha, que por sua vez são ligados sequencialmente, permitindo tanto buscas pontuais eficientes quanto varreduras por faixa (range scans).

O parâmetro fundamental no projeto de uma Árvore B+ é a sua **ordem (**m**)**, que define o número máximo de filhos que um nó pode ter e, consequentemente, o tamanho de cada nó. Em um sistema baseado em disco, a ordem é tipicamente escolhida de forma que o tamanho de um nó corresponda ao tamanho de um bloco de disco (ex: 4 KB). Isso garante que cada operação de leitura de nó corresponda a uma única operação de I/O, tornando ordens maiores inerentemente mais eficientes.

Neste trabalho, proponho-me a investigar essa mesma questão de dimensionamento, mas em um contexto diferente: um ambiente puramente *in-memory*. O objetivo é determinar, através da minha implementação e experimentação, qual a ordem de árvore B+ que oferece o melhor desempenho de busca para um conjunto de 10.000 registros de automóveis, analisando o comportamento para subconjuntos de 100, 1.000 e 10.000 registros.

2 Metodologia

Para alcançar o objetivo proposto, desenvolvi um ambiente de teste em linguagem C, composto por três componentes principais: um gerador de dados, a implementação da Árvore B+ e um programa de experimentação.

2.1 Geração de Dados

Criei um programa, `gerador_dados.c`, para produzir um arquivo de texto (`carros.txt`) contendo 10.000 registros de automóveis. Cada registro é unicamente identificado por um RENAVAM (chave primária) de 11 dígitos gerado aleatoriamente, e contém atributos como modelo, ano e cor, também gerados de forma aleatória.

2.2 Implementação da Árvore B+

A estrutura da Árvore B+ foi implementada nos arquivos `arvore_b_mais.h` e `arvore_b_mais.c`. A implementação suporta as operações essenciais de inserção e busca. Uma característica fundamental para este estudo foi a capacidade de selecionar o algoritmo de busca interna nos nós em tempo de execução, permitindo a comparação direta entre busca linear e busca binária.

2.3 Condução do Experimento

O programa `experimento.c` orquestra os testes. Ele carrega os registros do arquivo `carros.txt` para a memória e, em seguida, executa uma série de testes aninhados:

1. **Itera sobre os algoritmos de busca interna:** Linear e Binária.
2. **Itera sobre os volumes de dados:** 100, 1.000 e 10.000 registros.
3. **Itera sobre diferentes ordens de árvore:** de 5 a 200.

Para cada combinação, o programa popula uma nova árvore e mede o tempo médio de 100 buscas por chaves aleatórias, utilizando a função `clock()` da biblioteca `<time.h>` para aferir o tempo de execução.

3 Resultados e Análise

A condução dos experimentos revelou uma narrativa em duas etapas, que se mostrou fundamental para a compreensão do problema.

3.1 Experimento 1: A Descoberta com a Busca Linear

Inicialmente, implementei a árvore com o algoritmo mais simples para a busca interna nos nós: uma busca linear. Minha hipótese inicial, baseada na teoria de sistemas de disco, era de que ordens maiores levariam a um melhor desempenho. Os resultados, no entanto, contradisseram essa expectativa de forma clara.

Tabela 1: Tempo médio de busca (em μs) para 10.000 registros com **Busca Linear**.

Ordem (m)	**Tempo Médio (μs)**
5	0.13
10	0.13
25	0.24
50	0.28
75	0.47
100	0.54
150	0.71
200	0.82

Como observei na Tabela 1, o desempenho piorou consistentemente com o aumento da ordem. Este resultado surpreendente me levou a formular uma nova hipótese: em um ambiente *in-memory*, sem a latência de I/O de disco, o custo computacional da busca no nó ($O(m)$ para a busca linear) torna-se o fator dominante, superando qualquer benefício obtido pela redução da altura da árvore.

3.2 Experimento 2: A Confirmação com a Busca Binária

Para testar minha nova hipótese, estendi a implementação para suportar também a busca binária ($O(\log_2 m)$) dentro dos nós. Se a hipótese estivesse correta, a redução drástica no custo de busca interna deveria fazer com que o fator "altura da árvore"voltasse a ser relevante, revelando uma curvatura que lembra um "U".

Tabela 2: Tempo médio de busca (em μs) para 10.000 registros com **Busca Binária**.

Ordem (m)	**Tempo Médio (μs)**
5	0.13
10	0.15
25	0.17
50	0.28
75	0.24
100	0.26
150	0.23
200	0.40

Os resultados da Tabela 2 apontam fortemente que a hipótese está correta. Observa-se que o tempo, após um pico em ordem 50, **volta a diminuir** para ordens maiores como 75 e 150, indicando que a árvore mais "achatada"se tornou vantajosa. O aumento final no tempo para a ordem 200 sugerir a influência de outros fatores.

A comparação direta entre os dados das Tabelas 1 e 2 ilustra a diferença dramática no comportamento do desempenho, validando que a otimização da busca interna altera fundamentalmente o perfil de performance da estrutura de dados.

4 Conclusão

Neste trabalho, demonstrei experimentalmente que o dimensionamento ótimo dos nós de uma Árvore B+ em um ambiente *in-memory* é intrinsecamente dependente do algoritmo de busca interna utilizado.

Minha análise permite concluir que:

1. **Com Busca Linear**, o custo de varredura do nó é o gargalo principal. Portanto, para maximizar o desempenho, devem ser utilizadas **ordens pequenas**, que minimizam este custo.

2. **Com Busca Binária**, o custo de busca no nó é drasticamente reduzido, revelando o clássico trade-off com a altura da árvore. Mas mesmo nesse cenário, os meus resultados ainda mostram que ordens menores podem ser mais vantajosas. No caso desse experimento é o que foi observado; talvez esse seja um fator decorrente de um outro gargalo não observado nesse estudo.

Assim, esse expeirmento evidencia que não existe uma "ordem ótima"universal. A resposta para a questão do dimensionamento ideal requer uma análise cuidadosa do ambiente de execução (disco vs. memória) e das escolhas algorítmicas internas da própria estrutura de dados.