Estruturas de Dados

Aula 12 – Árvores B e B+

Prof. Ana Carolina Sokolonski

Bacharelado em Sistemas de Informação
Instituto Federal da Bahia – Campus Feira de Santana

2026

Agenda

Árvores B (B-Trees)

Bayer & McCreight, 1972

O que é uma Árvore B?

▪ Definição

Árvore de busca **balanceada** projetada para **dispositivos de armazenamento secundário** (HD, SSD). Minimiza acessos de E/S em disco. Usada na maioria dos **Sistemas de Arquivos** e **Bancos de Dados**. [Cormen et al. 2009]

Criada por **Bayer e McCreight** em 1972. Os nós podem ter **muitos filhos** — fator de ramificação de dezenas a milhares.

▪ Por que não usar ABB ou AVL?

Em ABB/AVL, cada nó acessa **1 chave**. Para 10^9 registros, seriam $\sim$30 acessos a disco. Na Árvore B, cada nó pode ter **milhares de chaves** — apenas **2–4 acessos** ao disco!

Nó de Árvore B (ordem $t = 2$)

Propriedades da Árvore B — Estrutura do Nó

▪ Campos de cada nó x

- $n[x] \rightarrow$ número de chaves no nó
- As $n[x]$ chaves em **ordem crescente**: $k_1 \leq k_2 \leq \cdots \leq k_{n[x]}$
- $folha[x] \rightarrow$ booleano (TRUE se folha)
- $n[x] + 1$ ponteiros para filhos $c_1[x], c_2[x], \ldots, c_{n[x]+1}[x]$ (se folha = NULL)

▪ Ordenação entre subárvores

Se k_i é qualquer chave em $c_i[x]$:

$$k_1 \leq \text{chave}_1[x] \leq k_2 \leq \cdots \leq \text{chave}_{n[x]}[x] \leq k_{n[x]+1}$$

▪ Restrições de tamanho (grau mínimo $t \geq 2$)

Nó	Chaves
Raiz (não vazia)	≥ 1
Qualquer nó (não raiz)	$\geq t - 1$
Qualquer nó	$\leq 2t - 1$

Um nó com $2t - 1$ chaves é **completo**.

▪ Profundidade uniforme

Toda folha tem a mesma profundidade (altura h). Isso garante buscas com custo idêntico para qualquer chave.

Altura da Árvore B

Teorema

Se uma Árvore B de grau mínimo $t \geq 2$ tem $n \geq 1$ chaves, sua altura satisfaz:

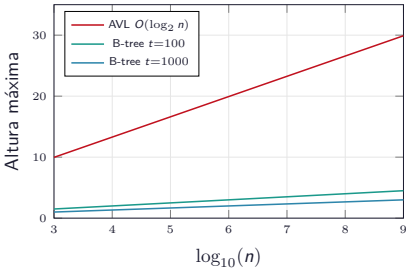
$$h \leq \log_t \frac{n+1}{2}$$

(Cormen et al., 2001)

Comparação prática

Para $n = 10^9$ chaves e $t = 1000$:

- ABB: até 10^9 comparações
- AVL: até $\sim$30 comparações
- Árvore B: apenas **3 acessos a disco**!



Árvore B — Visualização e Estrutura

```c
#define ORDEM 2

typedef struct chaveArvoreB{
    int chave;
    char nome[30];
    int folha;
}KEY;

typedef struct paginaArvoreB *POINTER;

typedef struct paginaArvoreB{
   int numPag;
   KEY listaChaves[2*ORDEM];
   POINTER ponteiros[2*ORDEM + 1];
}PAG;
```

Inserção em Árvores B

Split de nós completos

Inserção — Conceito

Ideia geral

A inserção ocorre sempre nas **folhas**. Se o nó de destino estiver **completo** ($2t - 1$ chaves), é necessário fazer um **split** antes de inserir.

Split de nó completo

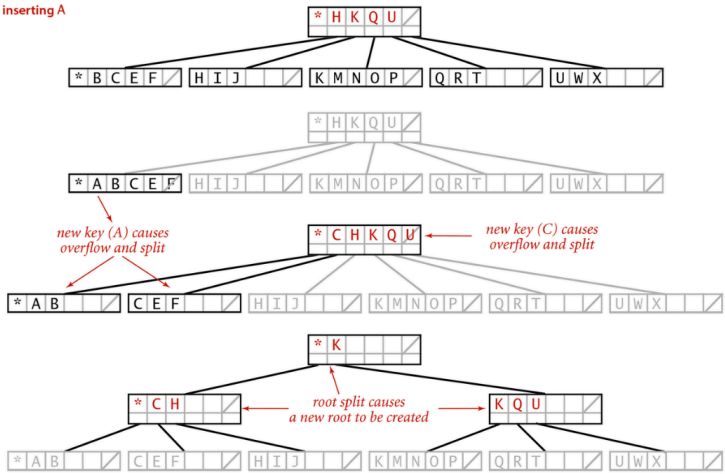
1. Divide o nó em **dois nós** com $t - 1$ chaves cada
2. A **chave mediana** sobe para o nó pai
3. O nó pai ganha um novo filho

Complexidade

Inserção: $O(t \cdot \log_t n)$

Split de nó com $2t - 1 = 5$ chaves ($t = 3$)

Inserção — Exemplo Visual



Inserting a new key into a B-tree set

Árvores B+ (B+-Trees)

Variação com dados apenas nas folhas

Árvore B+ — Definição e Diferenças

O que é Árvore B+?

Variação da Árvore B onde os **dados reais** (registros) ficam **apenas nas folhas**. Os nós internos contêm apenas **chaves de roteamento** para orientar a busca.

As folhas são ligadas em uma **lista encadeada ordenada**, permitindo varredura sequencial eficiente.

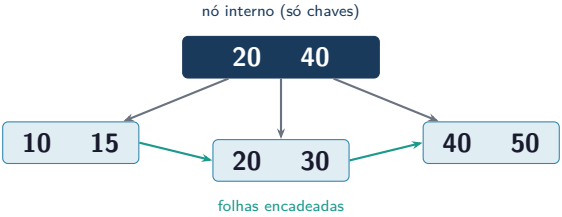
Uso predominante

A maioria dos SGBDs modernos usa **B+**, não B simples.

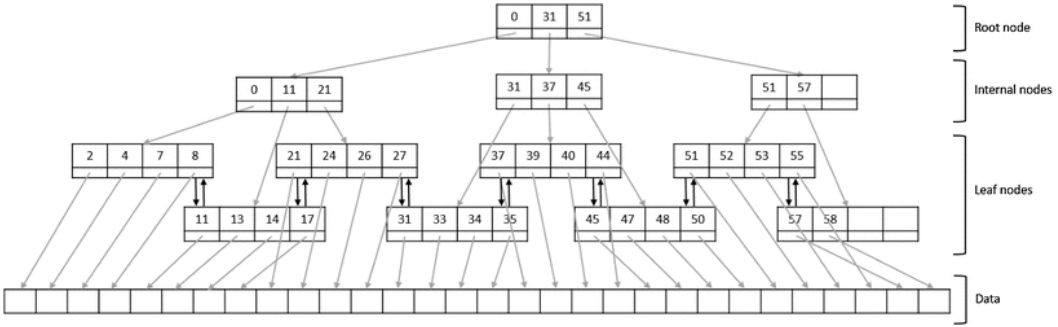
Vantagens sobre B

- Nós internos cabem **mais chaves** (sem dados) $\rightarrow$ fator de ramificação maior $\rightarrow$ árvore **mais baixa**
- Varredura sequencial em $O(n)$ pelas folhas encadeadas
- **Consultas de intervalo** muito eficientes

Árvore B+ — Visualização



Árvore B+ — Visualização



Comparação: Árvore B vs. Árvore B+

Característica	Árvore B	Árvore B+
Dados (registros)	Em **todos** os nós	Apenas nas **folhas**
Nós internos	Chaves + ponteiros + dados	Apenas chaves + ponteiros
Fator de ramificação	Menor (nós maiores)	**Maior** (nós internos menores)
Altura	Ligeiramente maior	**Menor**
Varredura sequencial	Requer percurso na árvore	**Lista encadeada** nas folhas
Consulta de intervalo	Menos eficiente	**Muito eficiente**
Busca pontual	Pode terminar nos nós internos	Sempre chega às folhas
Uso típico	Sistemas de arquivos simples	**SGBDs** (Oracle, PostgreSQL...)

Aplicações

Onde B e B+ são usados na prática

Aplicações de Árvores B e B+

Sistemas de Arquivos

- **NTFS** — Windows
- **HFS / APFS** — macOS
- **JFS / ext4** — Linux
- **Btrfs** — Linux (moderno)
- **ReFS** — Windows Server

Permitem localizar arquivos entre bilhões com poucos acessos ao disco.

Bancos de Dados

- **Oracle Database**
- **IBM DB2** e INGRES
- **Microsoft SQL Server**
- **PostgreSQL** e MySQL
- **MongoDB** (índices WiredTiger)
- **SQLite**

Índices B+ permitem buscas em $O(\log_t n)$ mesmo com terabytes de dados.

Aplicação (SGBD / SO)

Referências I

CORMEN, T. H. et al. *Introduction to Algorithms*. 2nd. ed. [S.l.]: The MIT Press, 2009. ISBN 0262032937.