

Árvores: árvores B

Prof. Jefferson T. Oliva
Material da Profa. Luciene de Oliveira Marin

Algoritmos e Estrutura de Dados II (AE23CP)
Engenharia de Computação
Departamento Acadêmico de Informática (Dainf)
Universidade Tecnológica Federal do Paraná (UTFPR)
Campus Pato Branco

- Árvores B
 - Definições
 - Estrutura de dados
- Operações em Árvore B

- Busca em memória primária
 - Busca sequencial
 - Busca sequencial indexada
 - Busca binária
 - Busca por interpolação
 - Hashing
 - Árvore binária de busca
 - Árvore AVL
 - Árvore rubro-negra

Introdução

- Pesquisa em memória primária vs. secundária
 - Primária: acesso mais rápido, portanto, mais caro



- Secundária: mais barato, portanto, acesso mais lento

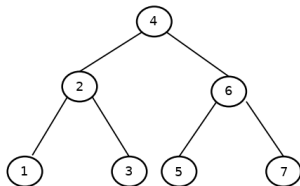


Árvores B

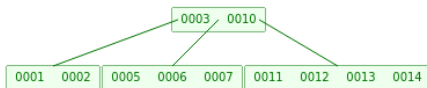
- Amplamente utilizadas para o armazenamento em memória secundária
- Diversas aplicações/ferramentas utilizam árvores B (*B-tree*)
 - Sistemas de arquivos (e.g. NTFS e ext4)
 - Banco de dados (e.g. Oracle e PostgreSQL)
- Árvores B são estruturas de dados projetadas para a indexação de dados
 - Estrutura mais avançada em relação ao arquivo sequencial indexado

Árvores B

- Árvore B é n-ária
 - Exemplo de árvore binária de busca



- Exemplo de árvore B



- Árvore binária de busca: cada nó tem um elemento e entre 0 e 2 nós filhos
- Árvore n -ária: cada nó pode ter entre 1 e $n - 1$ elementos e até n filhos

- Árvore de busca balanceada
- O nó de uma árvore B também é chamado de **página**
 - Cada nó pode ter mais de um elemento
- Ordem de árvore B: é o número máximo de páginas descendentes que uma página pode ter (Knuth)
 - Em Cormen et al. (2012) é definida como o número mínimo de páginas descendentes que uma página pode ter
 - Nesse material de aula é considerada a definição de Knuth

- Dada uma árvore B de ordem N :
 - Página raiz
 - Elementos: entre 1 (mínimo) e $N - 1$
 - Descendentes (subárvores): entre 2 (mínimo) e N
 - Página interna
 - Elementos: entre $\lceil \frac{N}{2} \rceil - 1$ (mínimo) e $N - 1$
 - Descendentes (subárvores): entre $\lceil \frac{N}{2} \rceil$ e N
 - Página folha
 - Elementos: entre $\lceil \frac{N}{2} \rceil - 1$ (mínimo) e $N - 1$
 - Não possui descendentes
 - Todas as páginas folhas estão no mesmo nível (*i.e.* possuem a mesma profundidade)

Árvores B

Definições

- Exemplos para uma árvore B de ordem 5 ($N = 5$):

Página raiz

- Elementos: entre 1 (mínimo) e $N - 1$
- Descendentes (subárvores): entre 2 (mínimo) e N

1 elemento



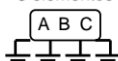
2 subárvores

2 elementos



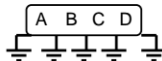
3 subárvores

3 elementos



4 subárvores

4 elementos



5 subárvores

- Quando uma página de árvore B de ordem N tem $N - 1$ elementos, diz-se que a mesma está cheia

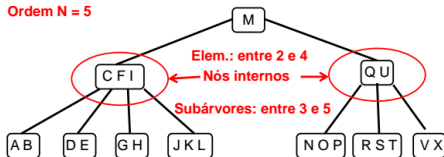
Árvores B

Definições

- Exemplos para uma árvore B de ordem 5 ($N = 5$):

Página (nó) interna

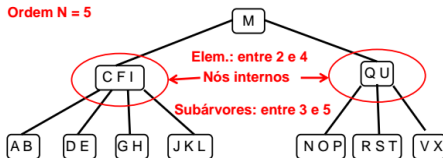
- Elementos: entre $\lceil \frac{N}{2} \rceil - 1$ (mínimo) e $N - 1$
- Descendentes (subárvores): entre $\lceil \frac{N}{2} \rceil$ e N



Árvores B

Definições

- Exemplos para uma árvore B de ordem 5 ($N = 5$):

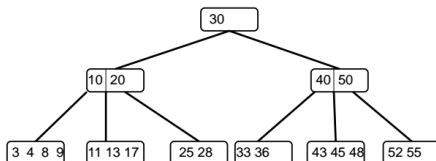


- Dada uma árvore B de ordem 5
 - Cada página interna ou folha pode ter entre 2 (mínimo) e 4 (máximo) elementos
 - Cada página interna pode ter entre 3 (mínimo) e 5 (máximo) descendentes

Árvores B

Definições

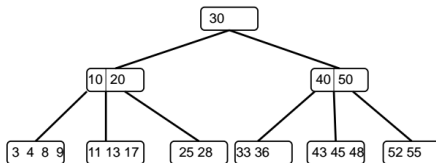
- Qual é a ordem da árvore B abaixo:



Árvores B

Definições

- Qual é a ordem da árvore B abaixo:
 - R.: $N = 5$



Árvores B

Estrutura de dados

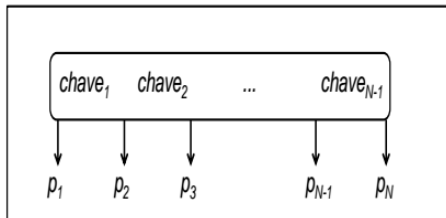
```
#define N ?

typedef struct NodeB NodeB;

struct NodeB{
    int nro_chaves;
    int chaves[N - 1];
    NodeB *filhos[N];
    int eh_no_folha;
};
```


Árvores B

Estrutura de dados



- Função para inicialização de um ponteiro do tipo NodeB (1)

```
NodeB* criar() {  
    NodeB *tree = (NodeB *) malloc(sizeof(NodeB));  
    int i;  
  
    tree->eh_no_folha = 1;  
    tree->nro_chaves = 0;  
  
    for (i = 0; i < N; i++)  
        tree->filhos[i] = NULL;  
  
    return tree;  
}
```

- Função para inicialização de um ponteiro do tipo NodeB (2)

```
NodeB* criar() {  
    NodeB *tree = (NodeB *) calloc(sizeof(NodeB));  
    int i;  
  
    tree->eh_no_folha = 1;  
  
    return tree;  
}
```

Operações em Árvore B

- Pesquisa
- Inserção
- Remoção

Pesquisa em Árvore B

- Semelhante a busca em árvores binárias de busca e AVL.

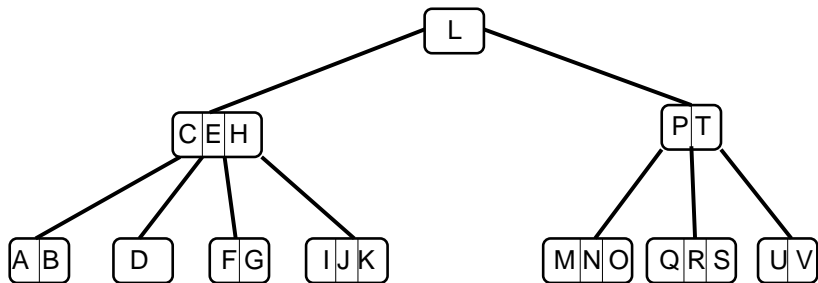
```
int busca_binaria(arvoreB *no, int info)
```

```
{  
    int meio, i, f; i = 0;  
    f = no->num_chaves-1;  
  
    while (i <= f)  
    {  
        meio = (i + f)/2;  
        if (no->chaves[meio] == info)  
            return(meio); //Encontrou. Retorna a posição em que a chave está.  
        else if (no->chaves[meio] > info)  
            f = meio - 1;  
        else i = meio + 1;  
    }  
    return(i); //Não encontrou. Retorna a posição do ponteiro para o filho.  
}
```

```
bool busca(arvoreB *raiz, int info)
```

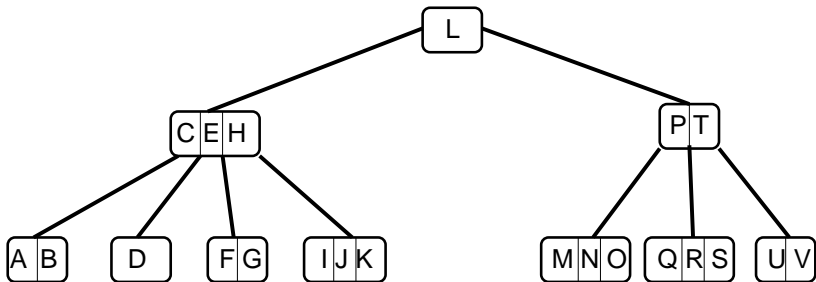
```
{  
    arvoreB *no;  
    int pos; //posição retornada pelo busca binária.  
    no = raiz;  
    while (no != NULL)  
    {  
        pos = busca_binaria(no, info);  
        if (pos < no->num_chaves && no->chaves[pos] == info)  
            return(true);  
        else no = no->filhos[pos];  
    }  
    return(false);
```

Pesquisa em Árvore B



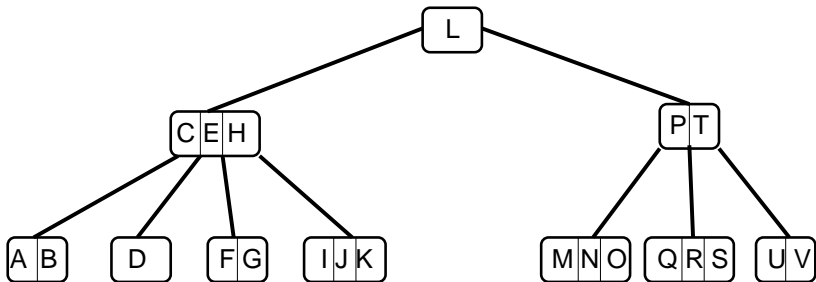
Pesquisa em Árvore B

Qual a ordem desta árvore?



Pesquisa em Árvore B

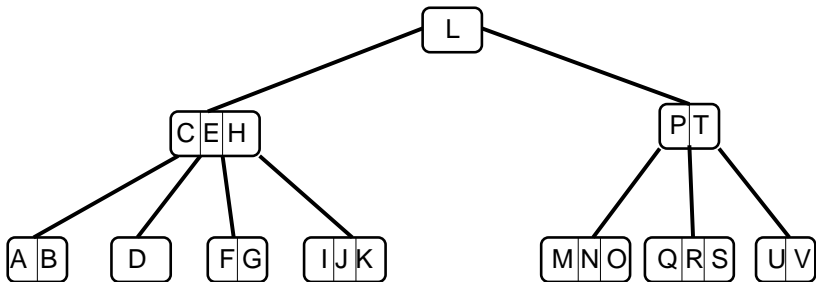
Qual a ordem desta árvore? $N = 4$



Pesquisa em Árvore B

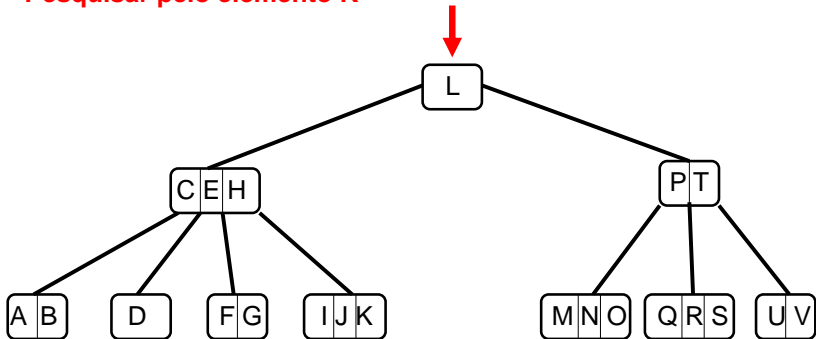
Qual a ordem desta árvore? $N = 4$

Pesquisar pelo elemento K



Pesquisa em Árvore B

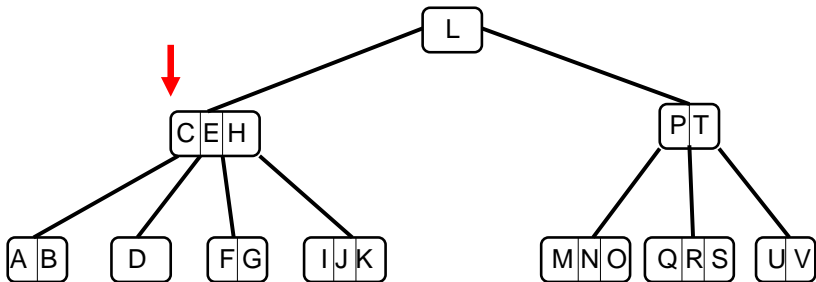
Qual a ordem desta árvore? $N = 4$
Pesquisar pelo elemento K



Pesquisa em Árvore B

Qual a ordem desta árvore? $N = 4$

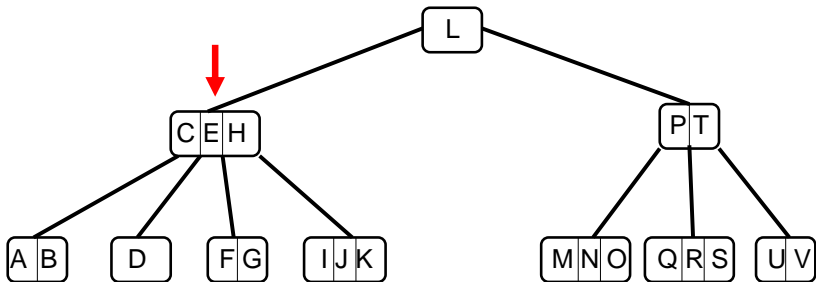
Pesquisar pelo elemento K



Pesquisa em Árvore B

Qual a ordem desta árvore? $N = 4$

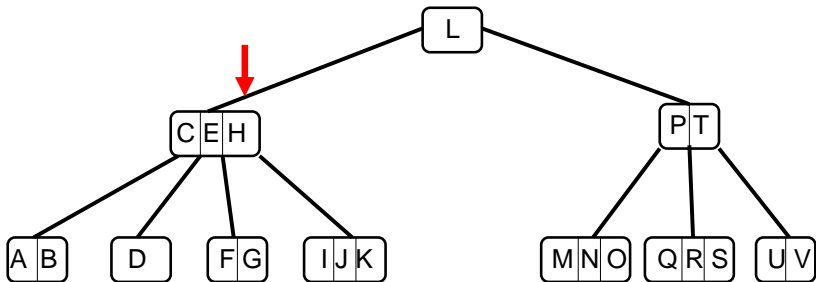
Pesquisar pelo elemento K



Pesquisa em Árvore B

Qual a ordem desta árvore? $N = 4$

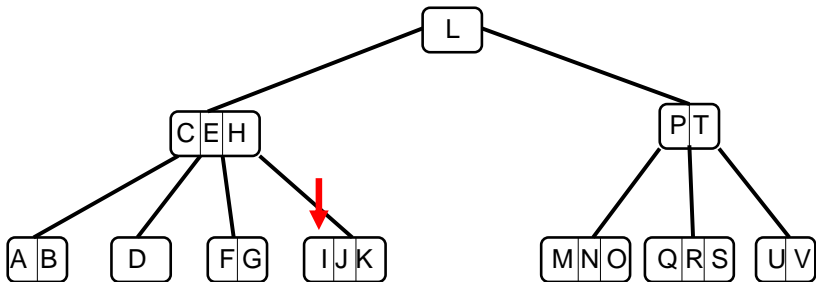
Pesquisar pelo elemento K



Pesquisa em Árvore B

Qual a ordem desta árvore? $N = 4$

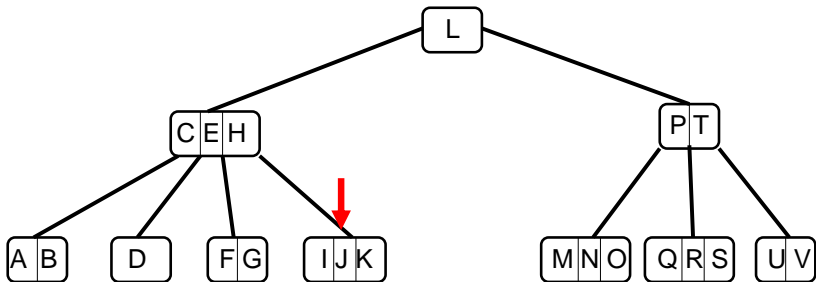
Pesquisar pelo elemento K



Pesquisa em Árvore B

Qual a ordem desta árvore? $N = 4$

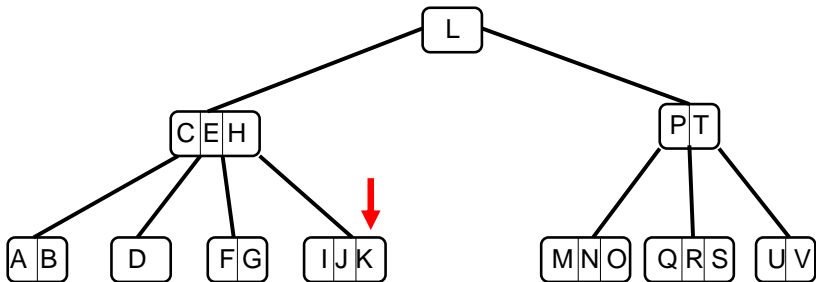
Pesquisar pelo elemento K



Pesquisa em Árvore B

Qual a ordem desta árvore? $N = 4$

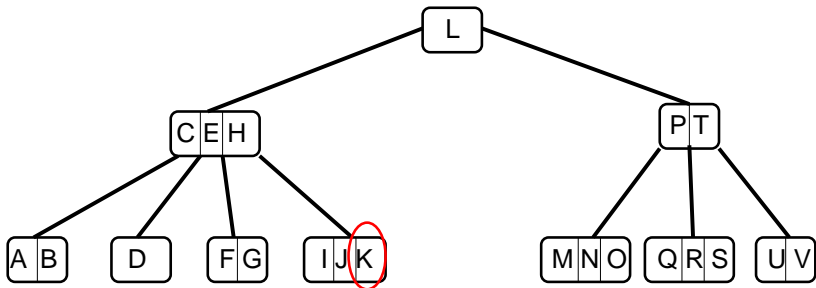
Pesquisar pelo elemento K



Pesquisa em Árvore B

Qual a ordem desta árvore? $N = 4$

Pesquisar pelo elemento K



Inserção

- Primeiro é preciso localizar a página onde o elemento deve ser inserido;
- Se a página onde o elemento deve ser inserido tiver menos de $N - 1$ elementos, este é alocado nesta página;
- Se a página já estiver cheia o processo irá provocar a criação de uma nova página.

Inserção

Criação de nova página

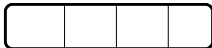
- Primeiramente escolhe-se um valor intermediário na sequência ordenada de chaves da página incluindo-se a nova chave que deveria ser inserida
- Cria-se uma nova página e os valores maiores do que a chave intermediária são armazenados nessa nova página e os menores continuam na página anterior (operação de split).
- Esta chave intermediária escolhida deverá ser inserida na página pai, na qual poderá também sofrer *overflow* ou deverá ser criada caso em que é criada uma nova página raiz. Esta série de *overflows* pode se propagar para toda a árvore B, o que garante o seu **balanceamento** na inserção de chaves.

Inserção

- Exemplo: Inserir os valores 10, 20, 30, 40 e 50 em uma árvore B de ordem 5

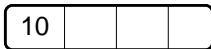
Inserção

- Exemplo: Inserir os valores 10,20,30,40 e 50 em uma árvore B de ordem 5



Inserção

- Exemplo: Inserir os valores 10,20,30,40 e 50 em uma árvore B de ordem 5



Inserção

- Exemplo: Inserir os valores 10,20,30,40 e 50 em uma árvore B de ordem 5

10	20		
----	----	--	--

Inserção

- Exemplo: Inserir os valores 10,20,30,40 e 50 em uma árvore B de ordem 5

10	20	30	
----	----	----	--

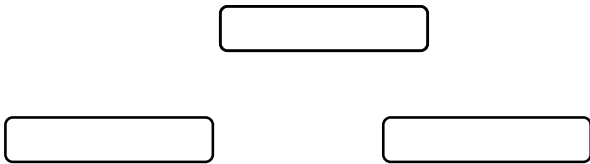
Inserção

- Exemplo: Inserir os valores 10,20,30,40 e 50 em uma árvore B de ordem 5

10	20	30	40
----	----	----	----

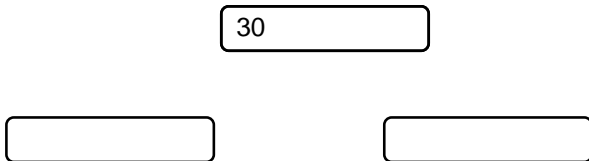
Inserção

- Exemplo: Inserir os valores 10,20,30,40 e 50 em uma árvore B de ordem 5



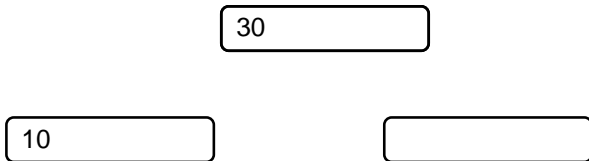
Inserção

- Exemplo: Inserir os valores 10,20,30,40 e 50 em uma árvore B de ordem 5



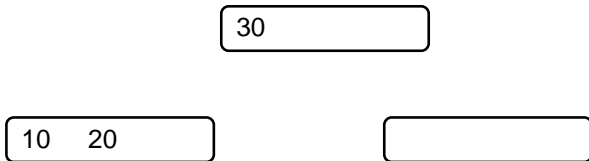
Inserção

- Exemplo: Inserir os valores 10,20,30,40 e 50 em uma árvore B de ordem 5



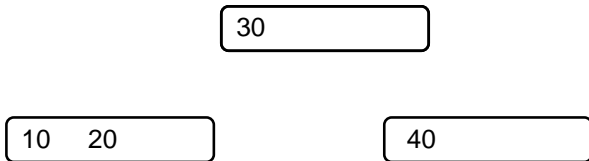
Inserção

- Exemplo: Inserir os valores 10,20,30,40 e 50 em uma árvore B de ordem 5



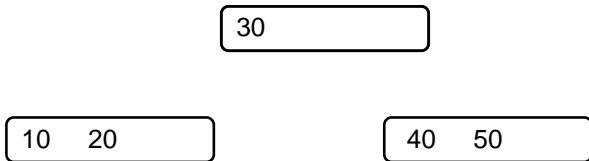
Inserção

- Exemplo: Inserir os valores 10,20,30,40 e 50 em uma árvore B de ordem 5



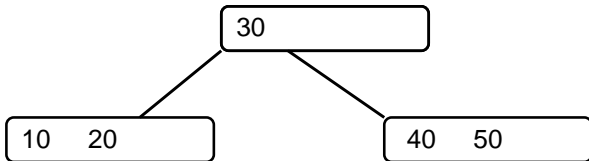
Inserção

- Exemplo: Inserir os valores 10,20,30,40 e 50 em uma árvore B de ordem 5



Inserção

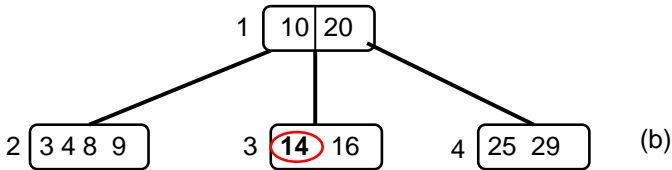
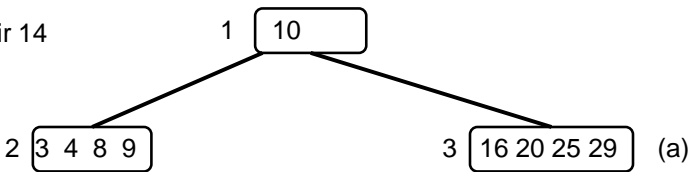
- Exemplo: Inserir os valores 10,20,30,40 e 50 em uma árvore B de ordem 5



Inserção em uma árvore B

- Ordem 5

Inserir 14



Inserção em uma árvore B de ordem 5

- 1. O registro contendo a chave 14 não é encontrado na árvore, e a página 3 (onde o registro contendo a chave 14 deve ser inserido) está cheia;
- 2. A página 3 é dividida em duas páginas, o que significa que uma nova página 4 é criada.
- 3. Os N registros (no caso são cinco registros) são distribuídos igualmente entre as páginas 3 e 4, e o registro do meio (no caso o registro contendo a chave 20) é movido para a página pai no nível acima.

Inserção

- Neste esquema de inserção, a página pai tem de acomodar um novo elemento. Se a página pai também estiver cheia, então o mesmo processo de divisão tem de ser aplicado de novo.
- No pior caso, o processo de divisão pode propagar-se até a raiz da árvore, e, neste caso, ela aumenta sua altura e um nível.
- É interessante observar que uma árvore B somente aumenta sua altura com a divisão da raiz.

Referências

- Projeto de Algoritmos – Nívio Ziviani
- Estruturas de Dados usando C – Tenenbaum
- Algoritmos – Teoria e Prática – Cormen
- B-Trees - animação
 - <https://www.cs.usfca.edu/~galles/visualization/BTree.html>

Exercício

1. Construa as seguintes árvores:

a) Ordem 3, valores: 5, 7, 14, 30, 21

b) Na a árvore obtida em a) insira: 35, 37