CI1057: Algoritmos e Estruturas de Dados III
ISAM e Árvore B+

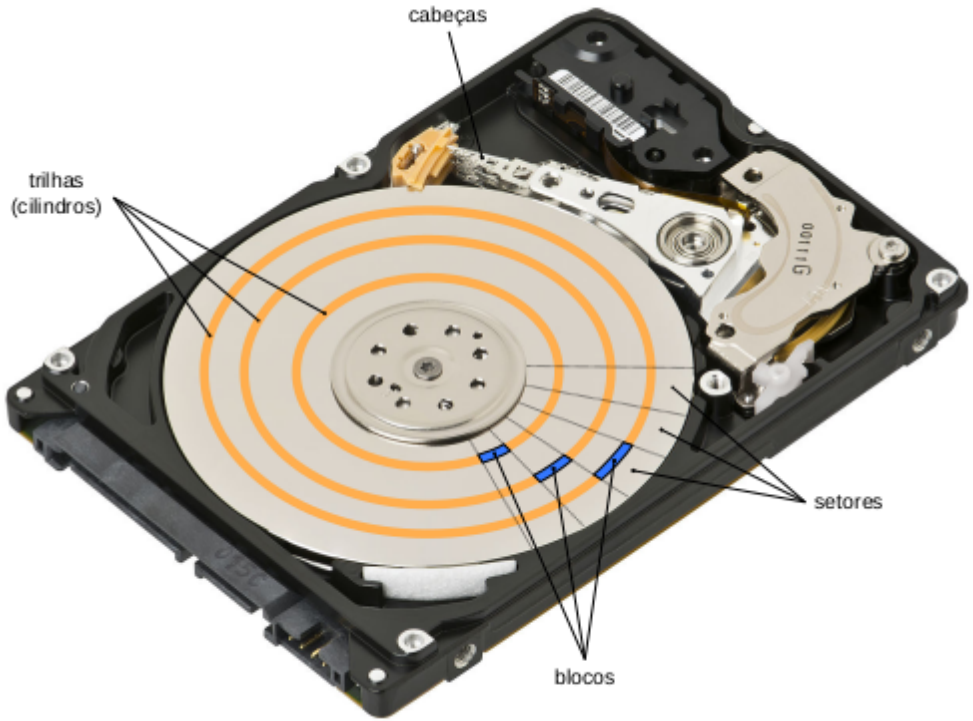
Profa. Carmem Hara

Departamento de Informática/UFPR

3 de julho de 2024

ISAM e Árvore B+

- Motivação: indexação de dados em memória secundária (discos)
- Acessar um registro em disco é algumas ordens de grandeza maior que o processamento na memória primária

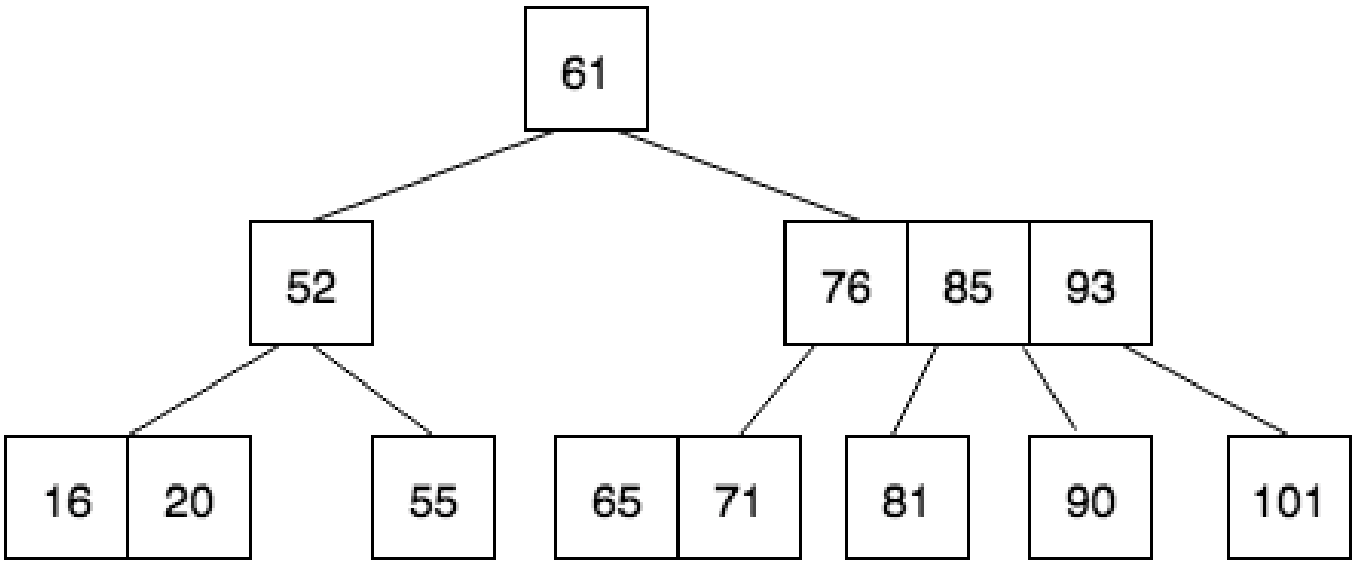


Relembrando: Árvore B

Uma árvore B de ordem t ($t \geq 2$) é uma árvore n-ária na qual:

1. os nodos internos (páginas), exceto a raiz, contém no minimo $t-1$ registros (e t descendentes) e no máximo $2t-1$ registros ($2t$ descendentes)
2. a raiz pode conter entre 1 e $2t-1$ registros
3. todos os nodos folha estão no mesmo nivel.

Árvore B de ordem 2



Árvore B - Estrutura de dados

Todos os nodos da árvore tem a **mesma** estrutura.

```
1  #define T 3
2  #define MIN T-1
3  #define MAX 2*T-1
4  #define MEIO T-1
5
6  typedef int ItemArv;
7
8  typedef struct nodo *ApNodo;
9  typedef struct nodo {
10   ItemArv item[MAX];
11   ApNodo ap[MAX+1];
12   int numItem;
13   ApNodo pai;
14 } Nodo;
15 typedef ApNodo ArvB;
16
17 typedef struct posicao{
18   ApNodo ap;
19   int ind;
20 } Posicao;
```

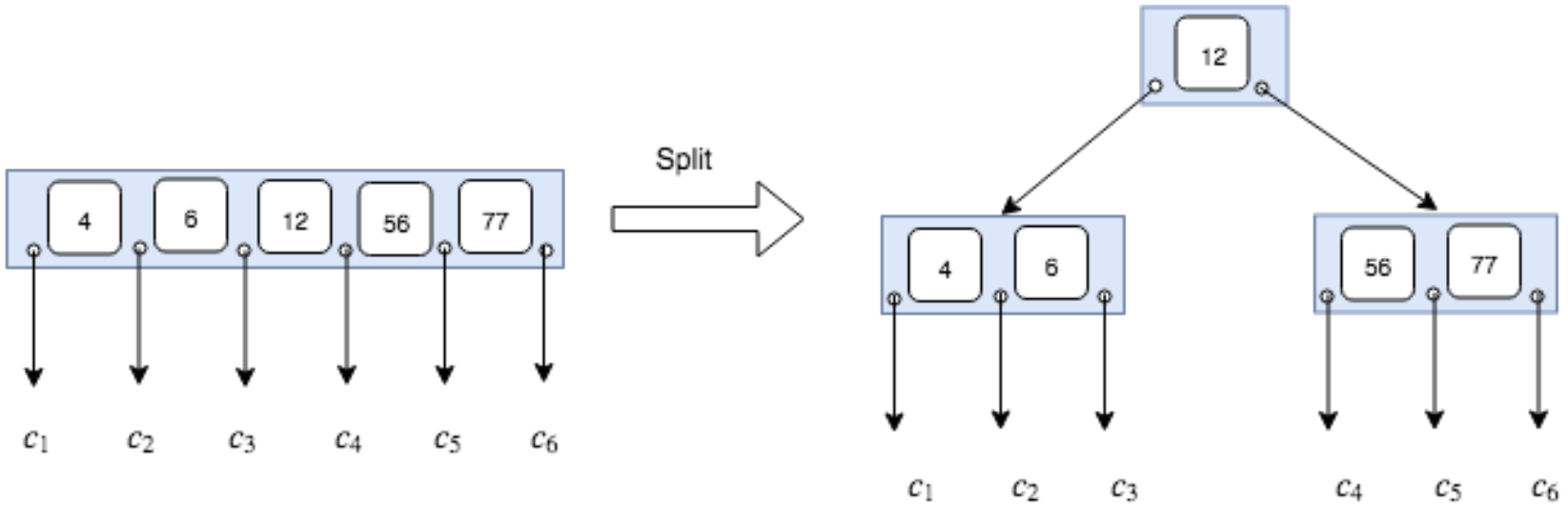
Busca

```
1  void buscaArvB( ItemArv v, ArvB p, Posicao *res ){
2    int i;
3
4    if( p == NULL ){
5      res->ap= NULL;
6      return;
7    }
8    i= 0;
9    while( i < p->numItem && lt( p->item[i], v ))
10     i++;
11   if( eq( v, p->item[i] ) ){
12     res->ap= p;
13     res->ind= i;
14     return;
15   }
16   /* readFile( p->ap[i] ) */
17   return buscaArvB( v, p->ap[i], res );
18 }
```

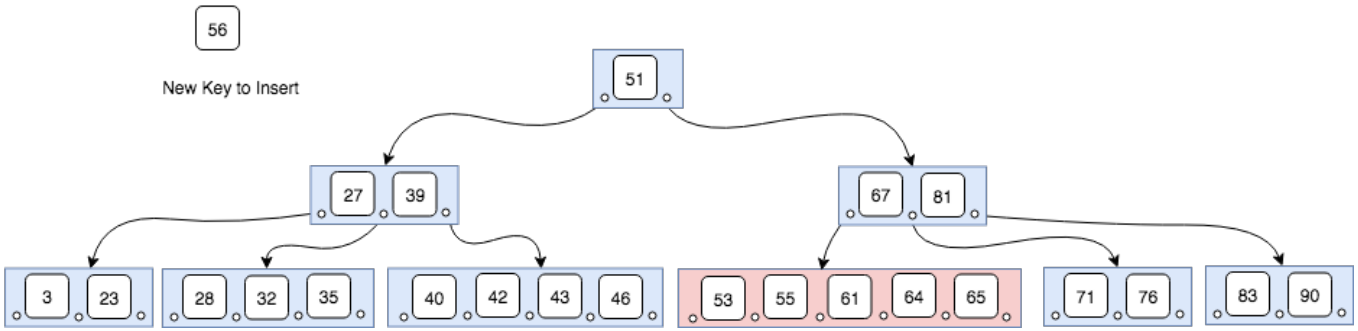
Inserção

- Inserção sempre na folha
- Balanceamento com duas abordagens:
  - Top-down: *split* dos nodos cheios no caminho da raiz até a folha
  - **Bottom-up:** *split* da folha quando necessário, com propagação do valor do meio para o pai

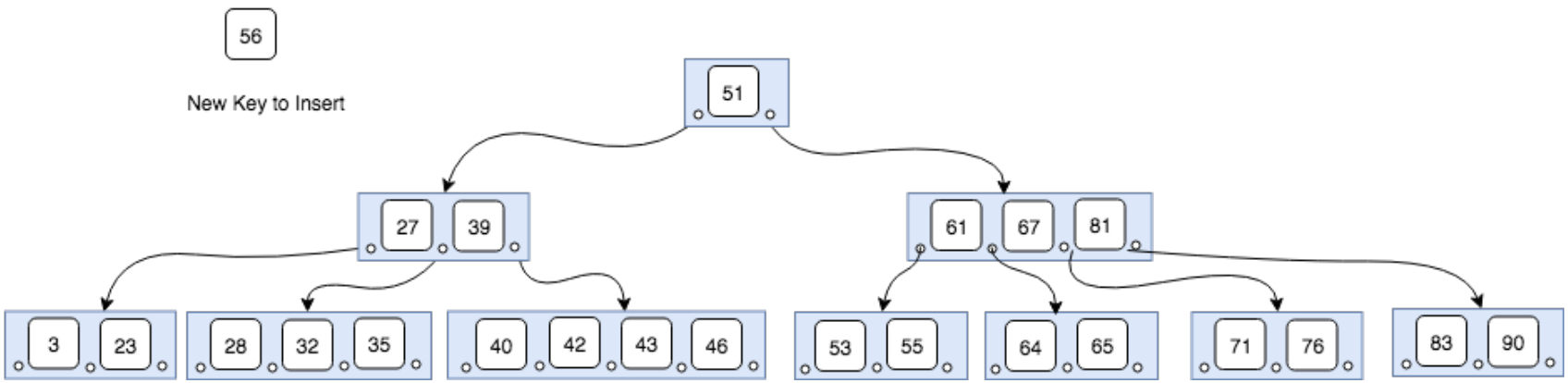
Split em Árvore B de ordem 3



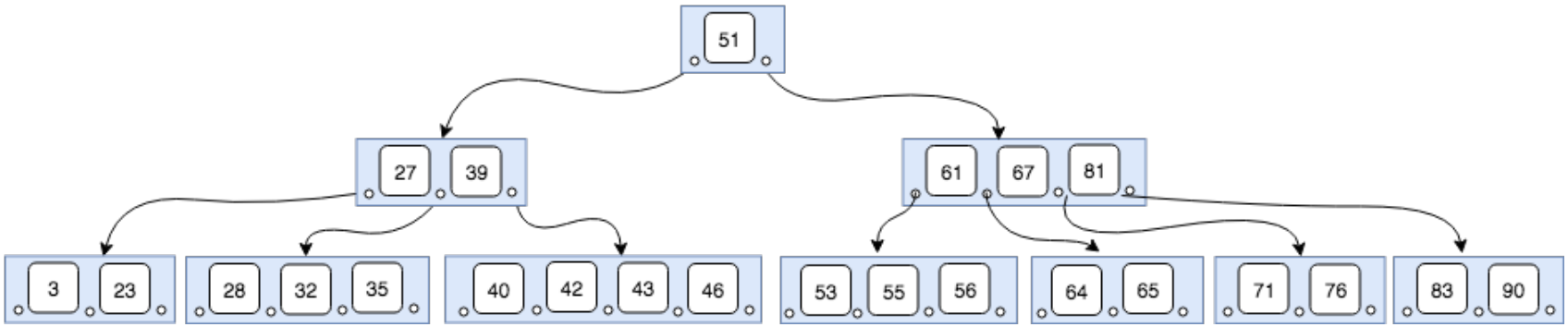
Exemplo - Inserção de 56 na Árvore B de ordem 3



Depois do Split

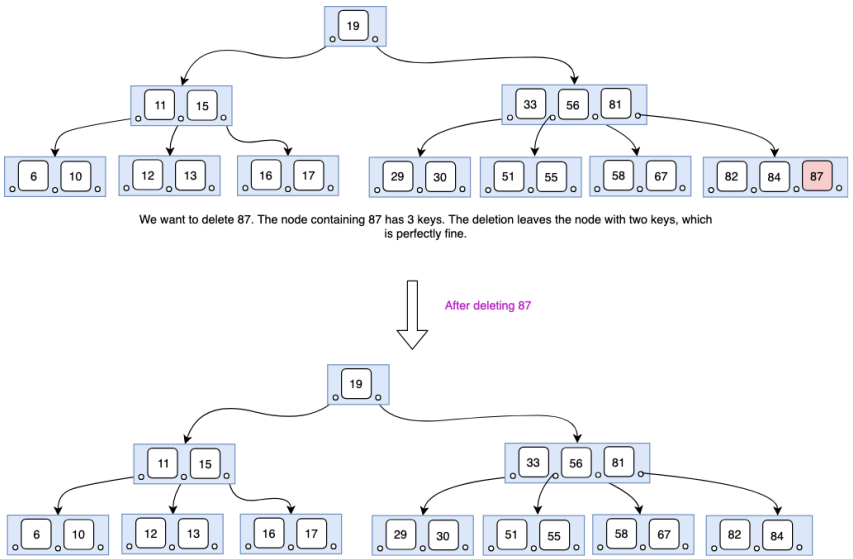


Árvore Resultante Após a Inserção



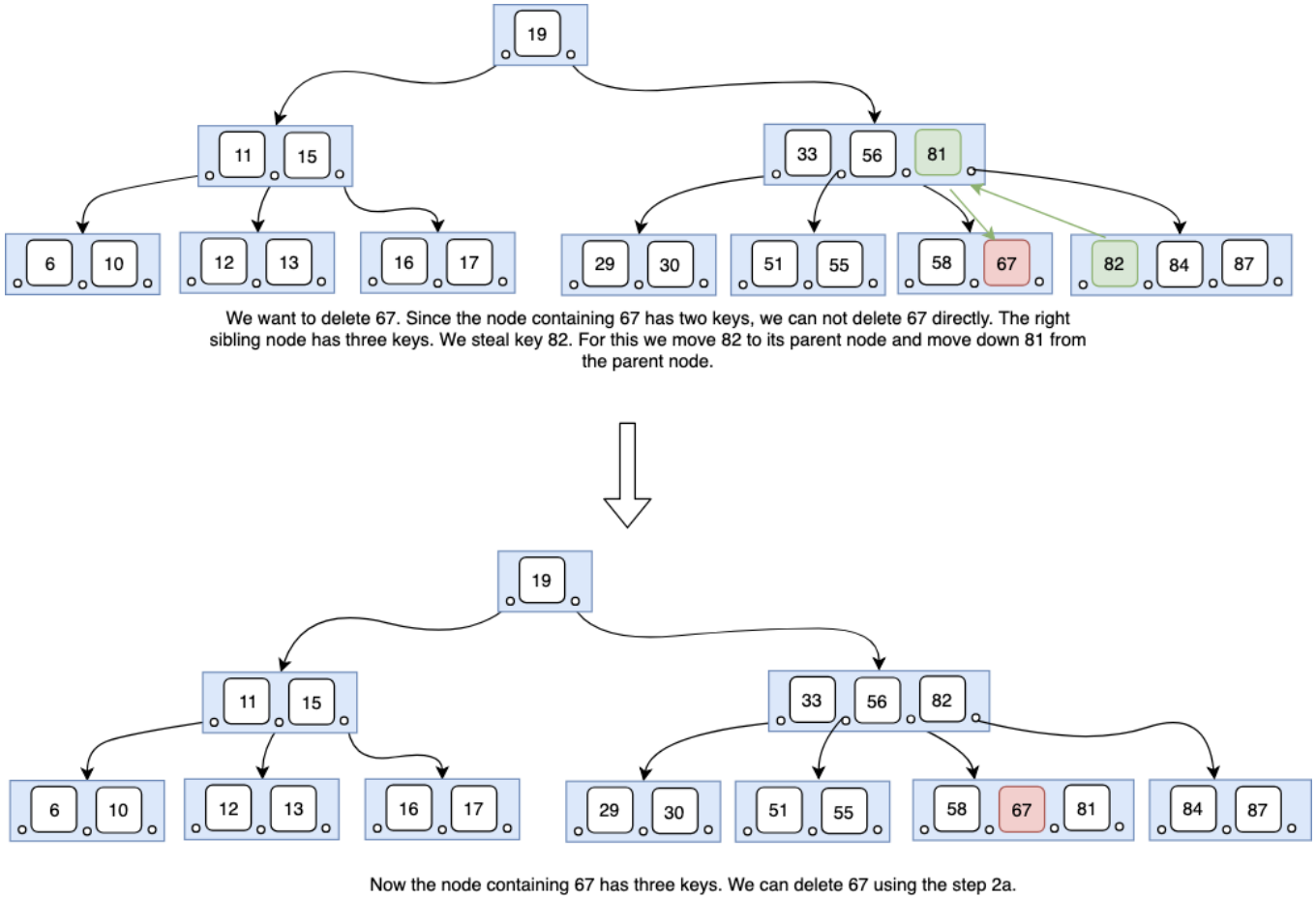
Remoção

1. Busque o nodo N que contém o valor V a ser removido
2. Se N não é um nodo folha, remover o sucessor de V ($remV$), localizado em uma folha ($nodoRem$) e depois substituir V por $remV$



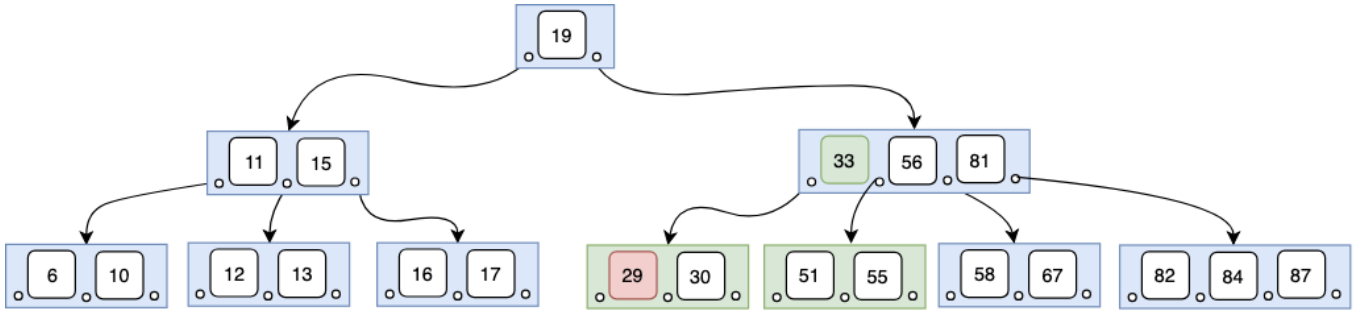
Remoção - "empréstimo do irmão"

4. Se o nodo contiver menos que $t - 1$ valores, "empresta" um valor do irmão

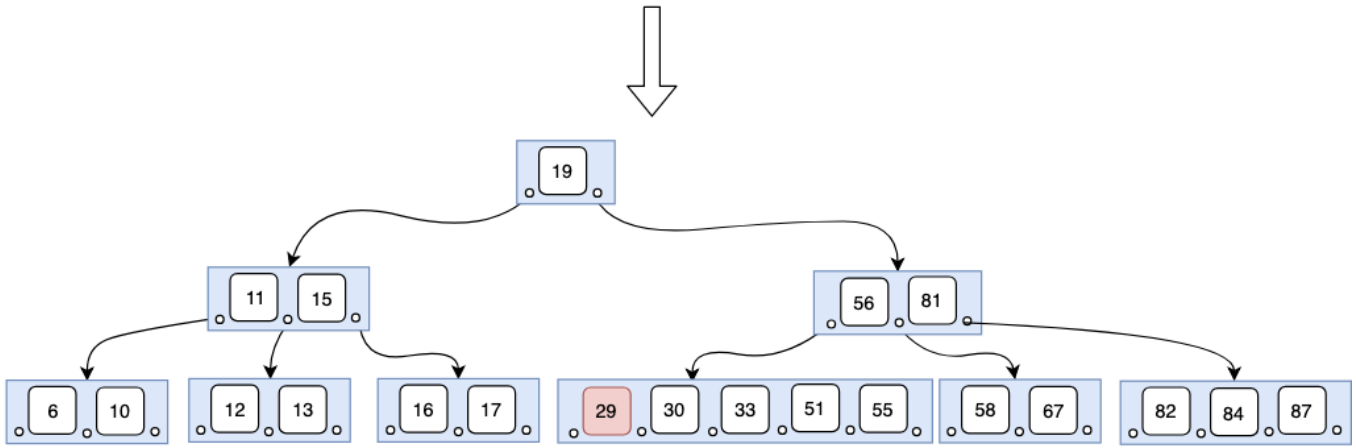


Remoção- "merge com o pai"

5. Se os irmãos tem somente $t-1$ valores, faz o "merge" com um valor do pai

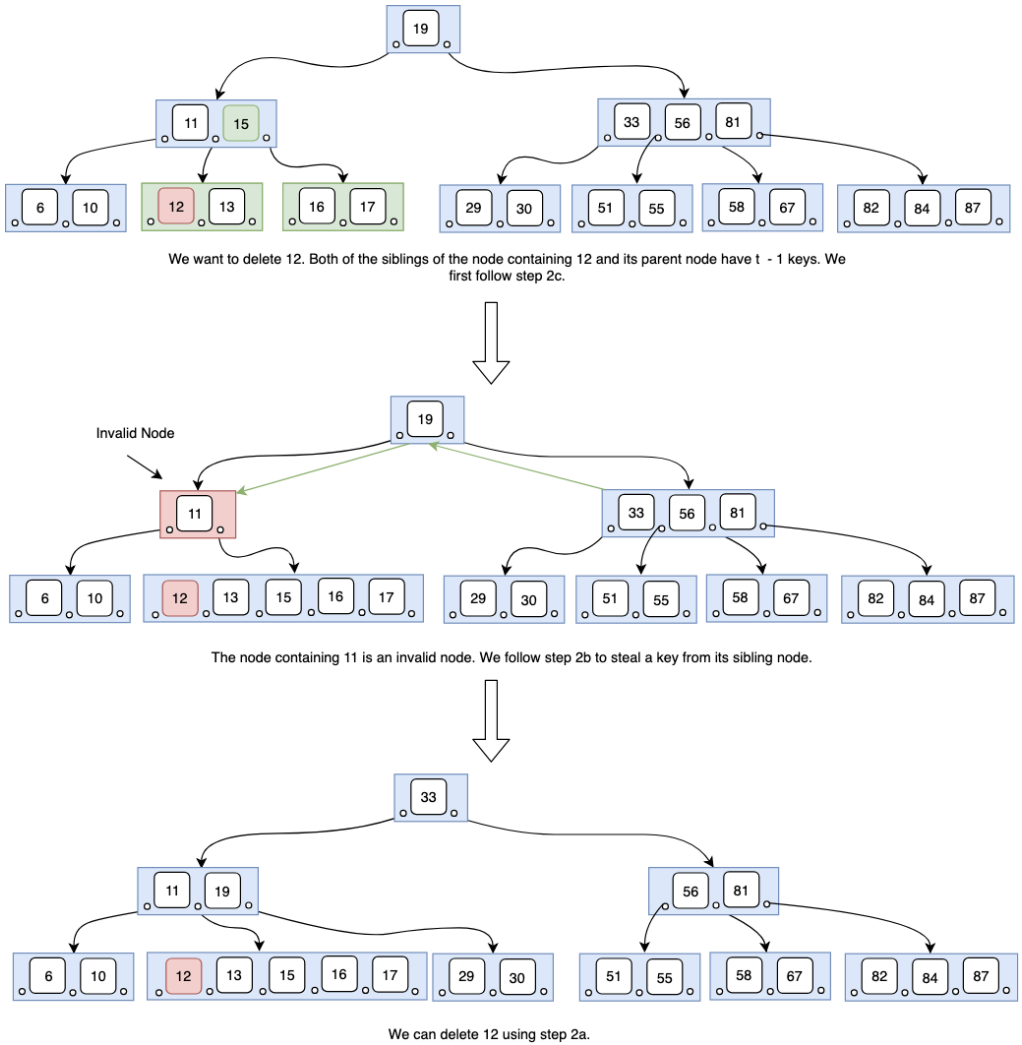


We want to delete 29. Immediate sibling of the node containing 29 has only t - 1 keys. In this case, we merge the node containing 29 with its immediate sibling and one key from the parent node.

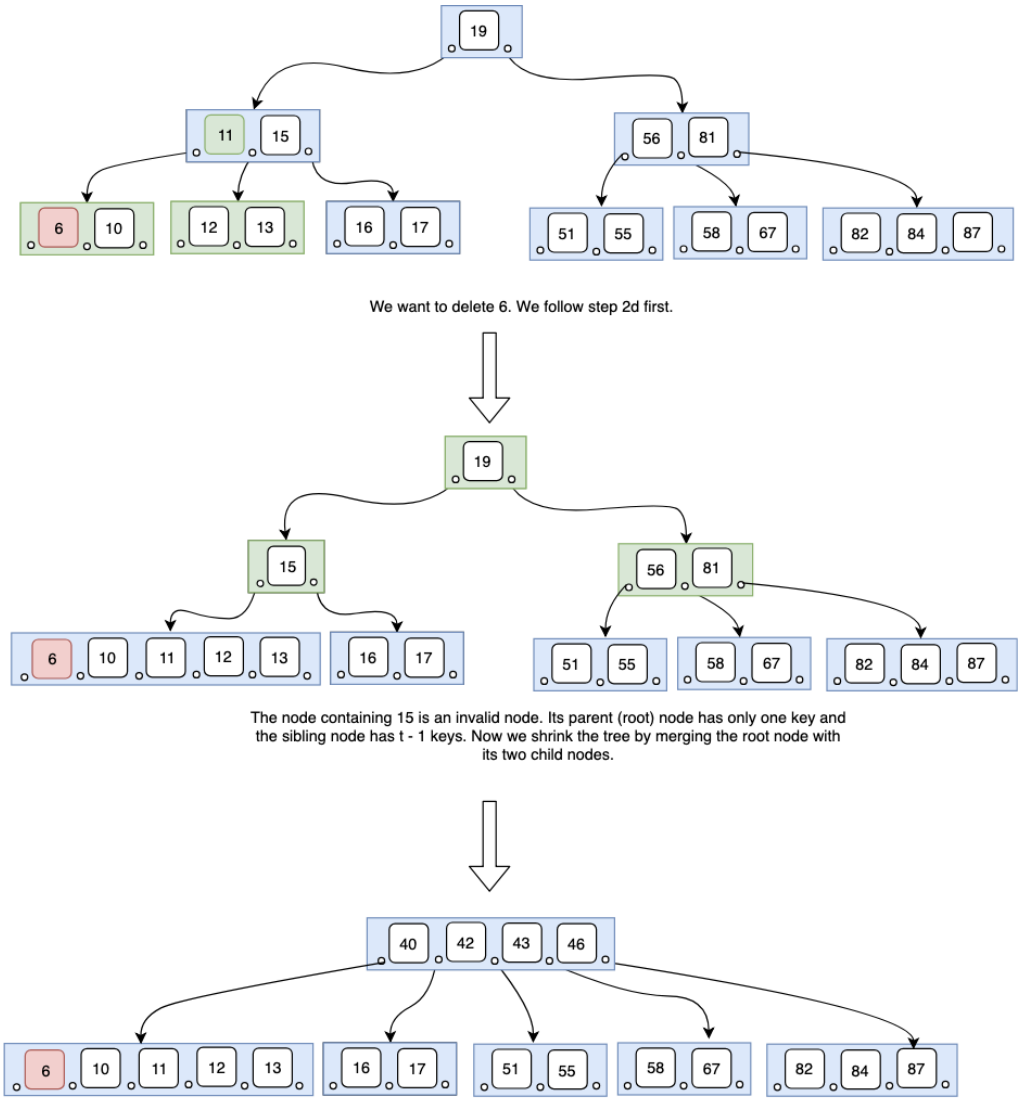


Now we can delete 29 using step 2a.

Remoção- propagação do “merge”

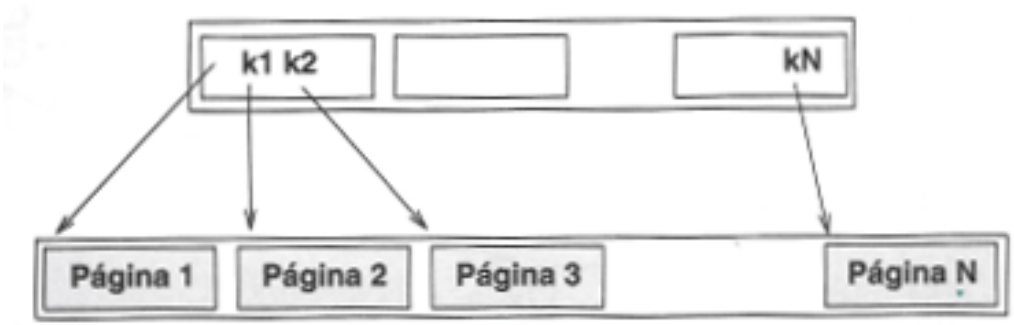


Remoção- propagação do "merge"

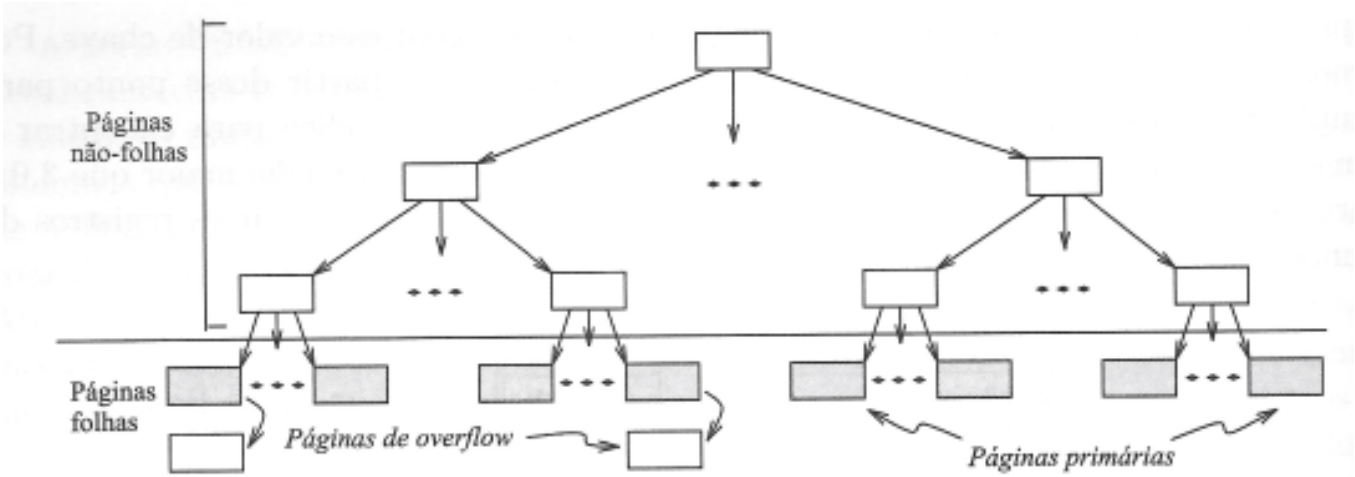


Estruturas ISAM e Árvore B+

- Há 2 tipos de nodos (ou páginas):
  - Páginas de índice (em nodos não folha)
  - Páginas de dados (nos nodos folha)
- Motivação
  - Facilita a busca sequencial (e de intervalo)
  - Operações mais simples
  - Permite maior concorrência no acesso à estrutura de indexação



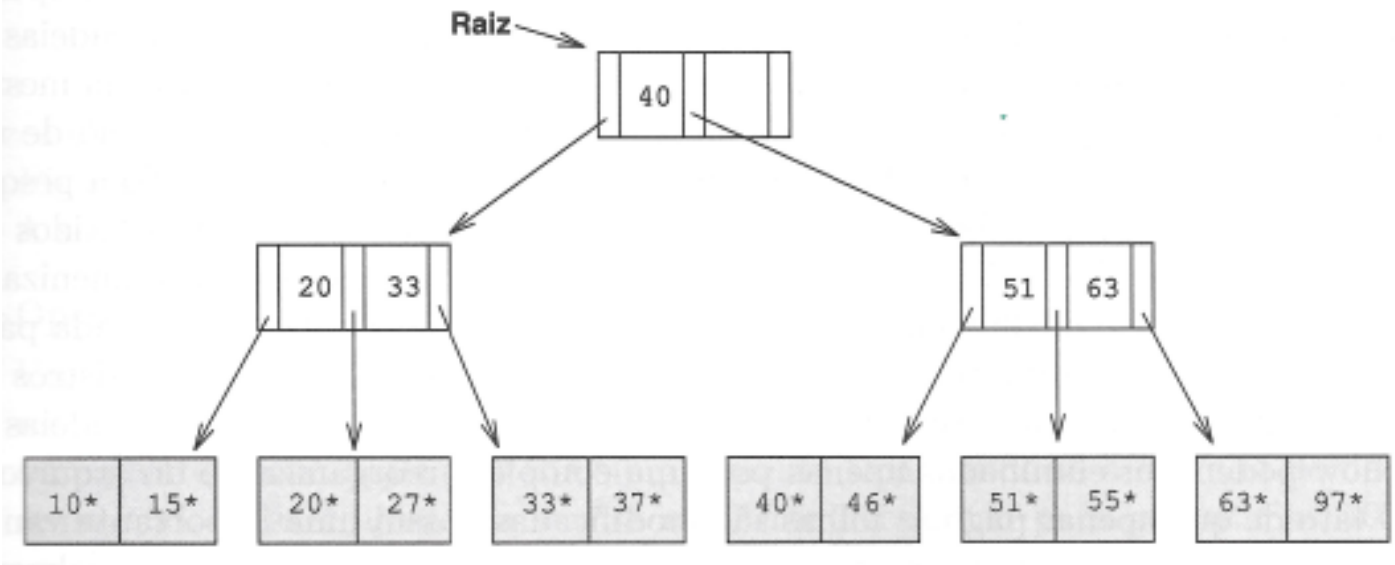
Indexed Sequential Acess Method (ISAM)



- Quando a estrutura é criada, todas as páginas folha (com os dados) são alocadas sequencialmente no disco, ordenadas pela chave
  - Estas são as *páginas primárias*
- As páginas de índice são **estáticas** e criadas sobre os páginas de dados
- A inserção de novos dados pode criar páginas de *overflow*

ISAM - Exemplo

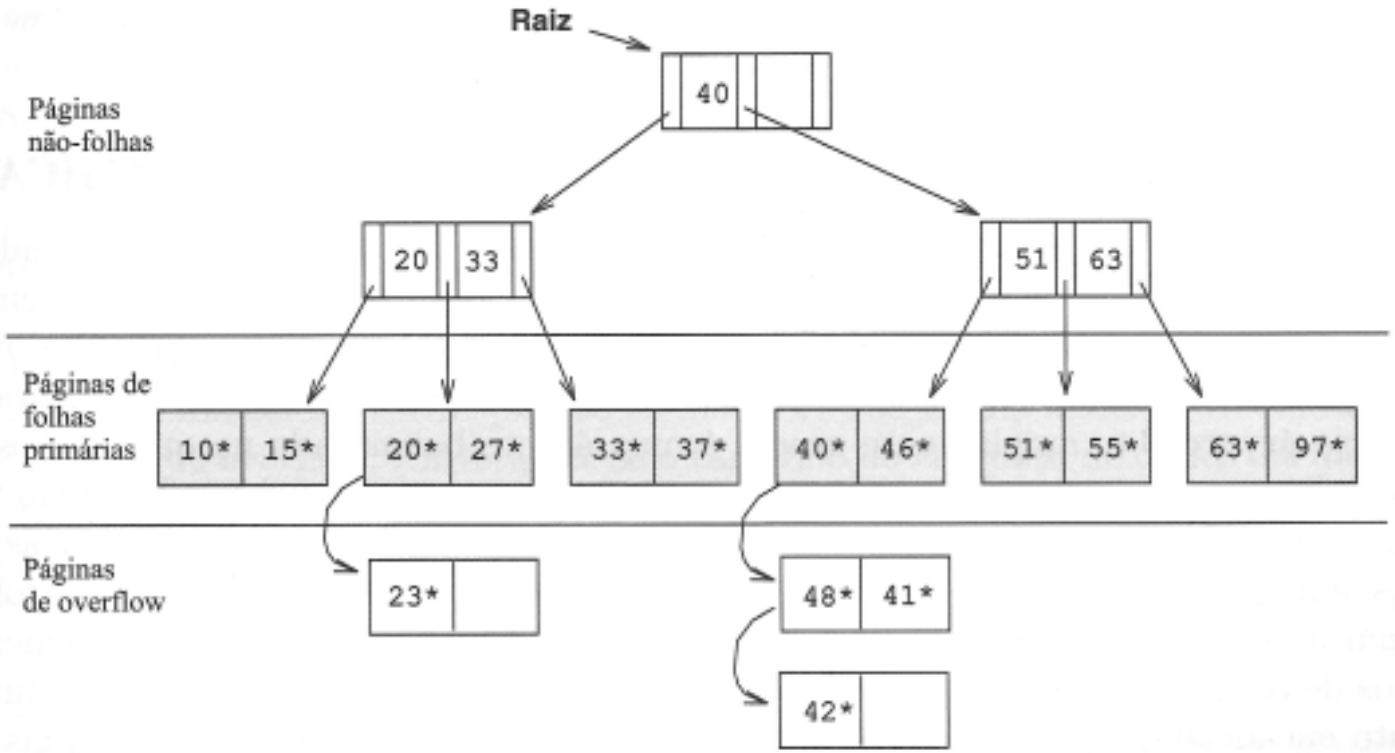
- Quantidade de páginas primárias e de índice são FIXAS.
- Para minimizar páginas de overflow: páginas primárias são criadas com 20% de espaço livre



ISAM - Inserção

Inserções e remoções são executadas apenas nas páginas de dados (primárias e de overflow).

- Inserção de 23, 48, 41, 42



ISAM - Remoção

- Se uma página de overflow ficar vazia - página removida
- Se uma página primária ficar vazia - ela é mantida

ISAM

Vantagens:

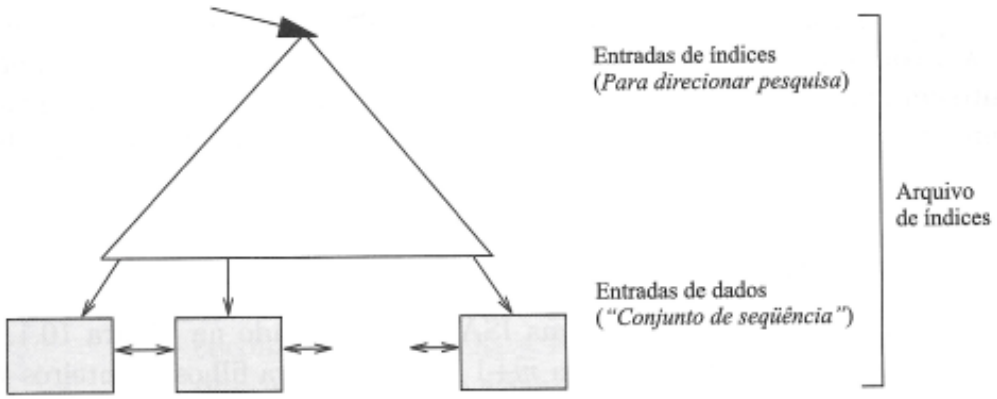
- O controle de acesso concorrente é simples porque apenas paǵinas de dados precisam ser *bloqueadas*.
- Varredura por intervalos grandes é em geral rápido porque nodos folha são alocados em sequência

Desvantagens:

- O desempenho da estrutura piora à medida que se criam longas cadeias de páginas de overflow.

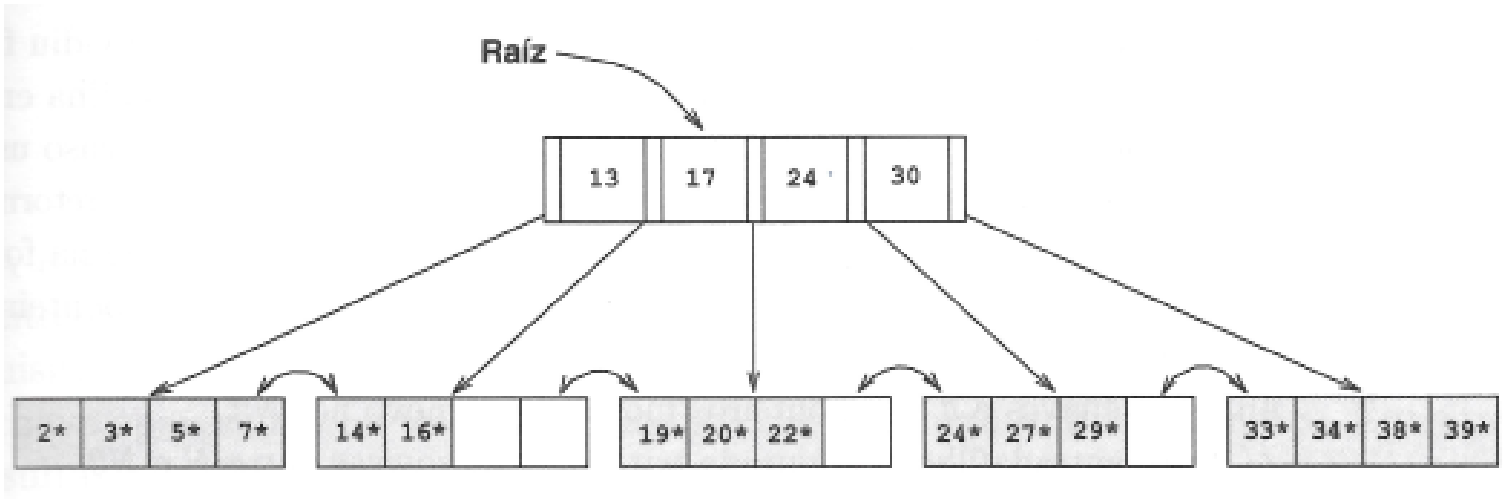
Árvore B+

- Possui páginas de índice **dinâmicas**
- Páginas de índice são similares a uma árvore B
- Páginas de dados formam um *conjunto de sequência*
  São páginas duplamente encadeadas
- A ordem t dos nodos folha e não-folha podem ser diferentes
- Assim com nas árvores B, todos os nodos devem ter no mínimo $(t-1)$ entradas e no máximo $(2t-1)$ entradas



Árvore B+: Busca

- Similar a busca em uma árvore B
- A busca SEMPRE termina em um nodo folha



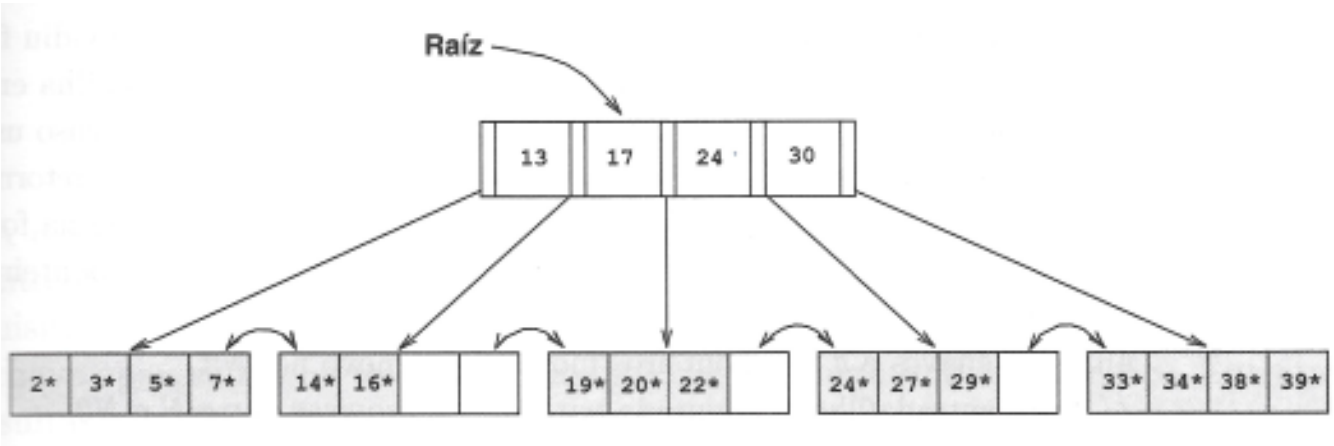
Árvore B+: Busca

```
1  void buscaArvB( ItemArv v, ArvB p, Posicao *res ){
2    int i;
3
4    if( p == NULL ){
5      res->ap= NULL;
6      return;
7    }
8    i= 0;
9    while( i < p->numItem && lt( p->item[i], v ))
10     i++;
11   if( eq( v, p->item[i] ) ){
12     res->ap= p;
13     res->ind= i;
14     return;
15   }
16   /* readFile( p->ap[i] ) */
17   return buscaArvB( v, p->ap[i], res );
18 }
```

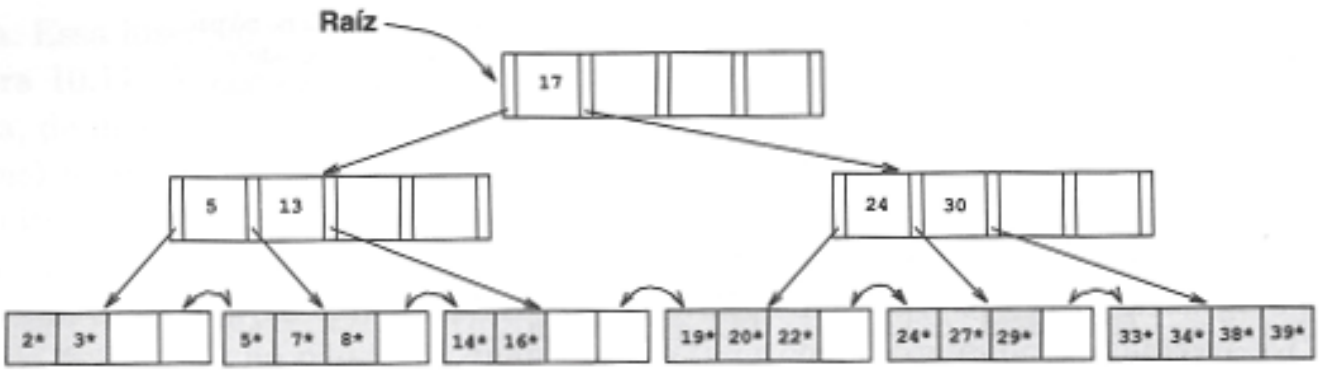
Árvore B+: Inserção

- Inserção sempre na folha
- Duas abordagens para inserir em folha cheia:
  - Split do nodo
  - Redistribuição

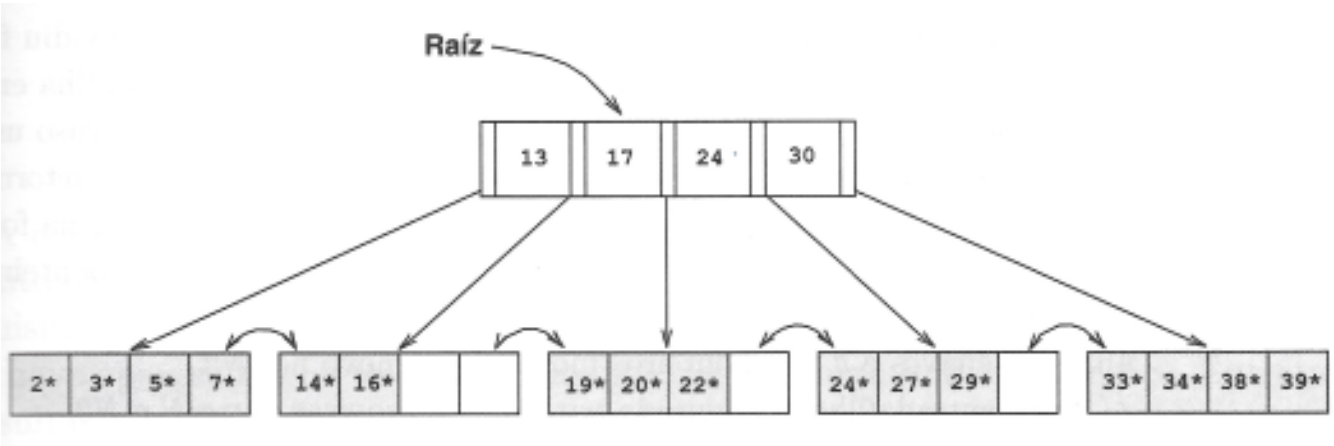
Inserção com Split



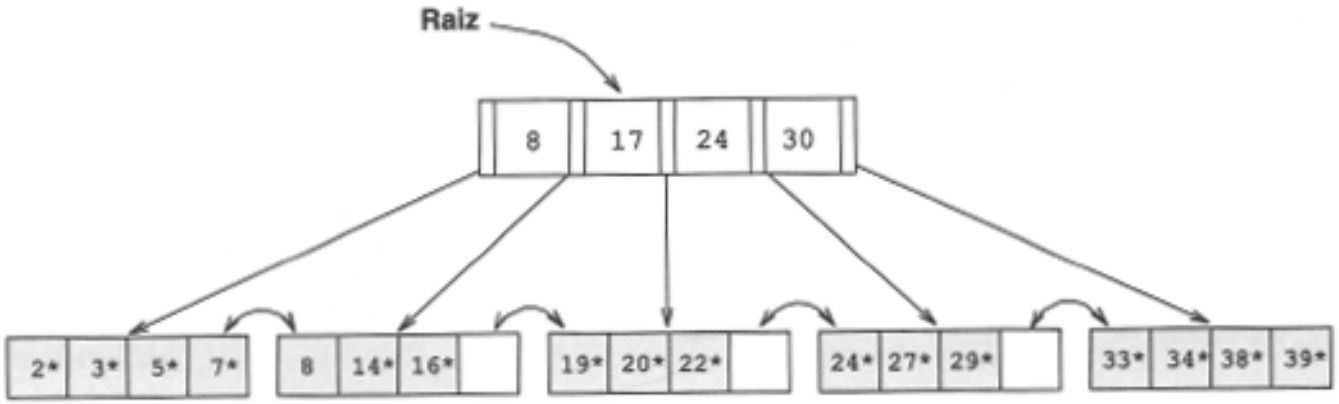
Inserção de 8:



Inserção com Redistribuição



Inserção de 8:



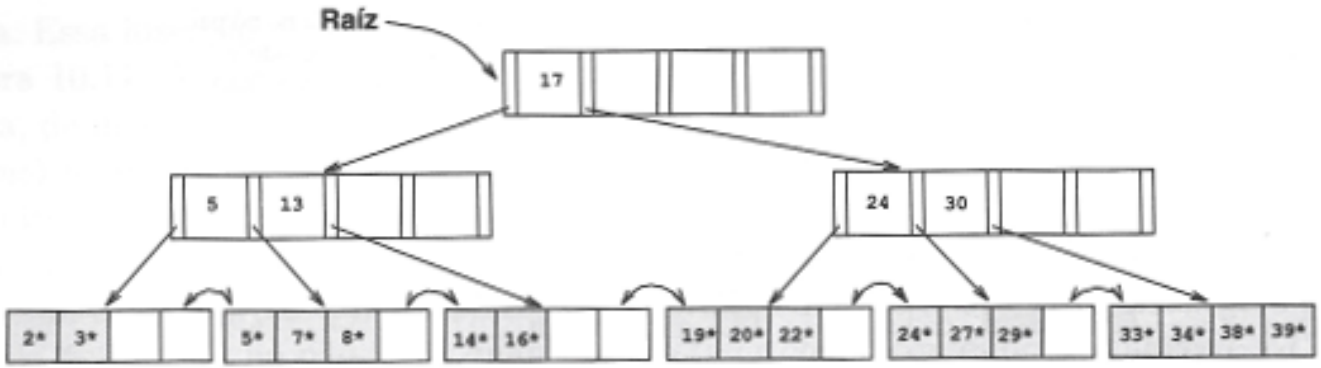
Inserção - Comparação das Abordagens

- Split
  - É possível propagar até a raiz, mas este caso é infrequente
- Redistribuição
  - Para saber se é possível, é preciso recuperar um ou mais irmãos
  - Se não for possível é preciso fazer o split
- Heurística Recomendada
  - Não redistribuir nodos não folha
  - Para nodos folha: redistribuir apenas com nós vizinhos de mesmo pai; caso contrário, fazer o split

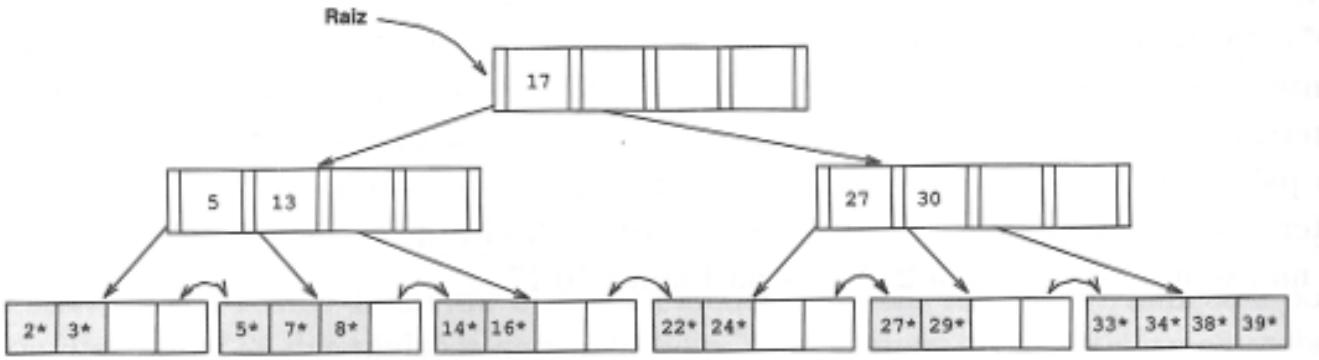
Árvore B+:- Exclusão

- Remoção sempre na folha
- Duas abordagens para folhas com menos de $t - 1$ entradas:
  - Merge com o vizinho
  - Redistribuição

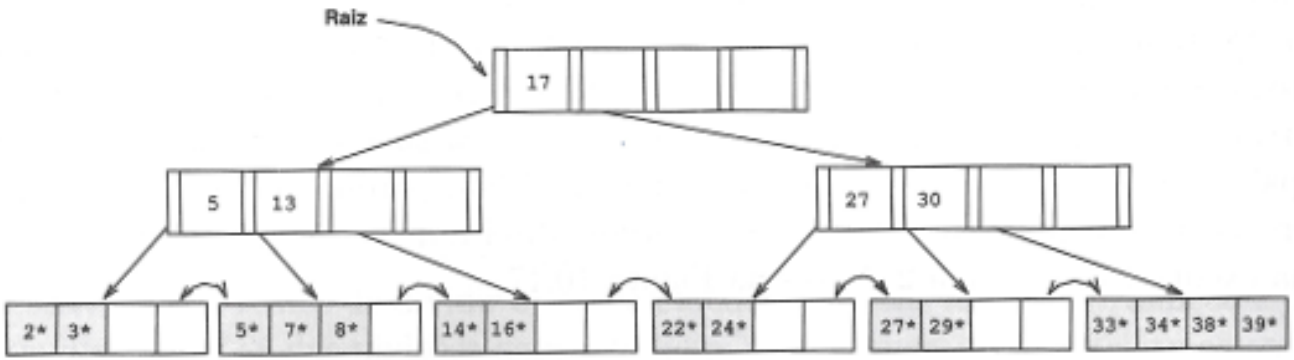
Exemplo - Exclusão com Redistribuição



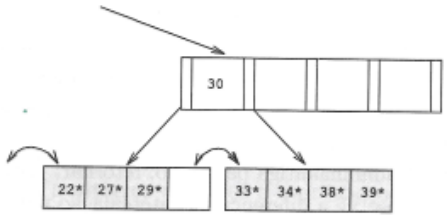
Exclusão de 19 e 20:

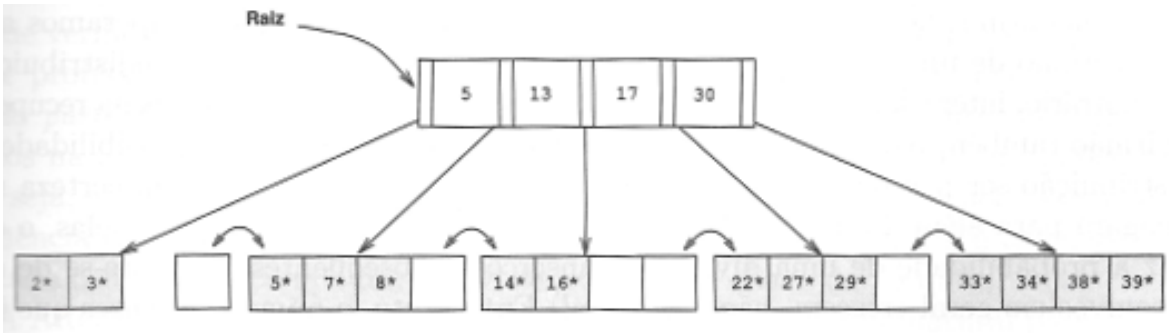


Exemplo - Exclusão com Merge

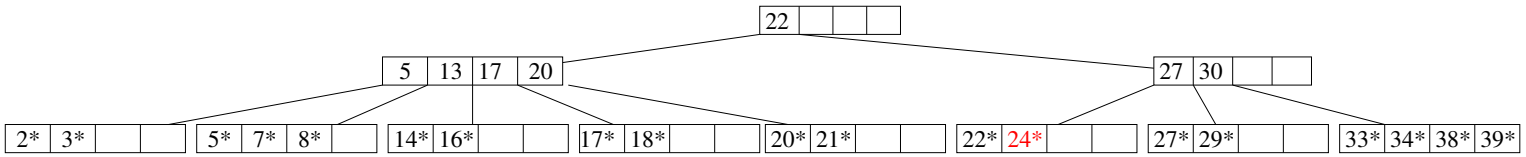


Exclusão de 24:

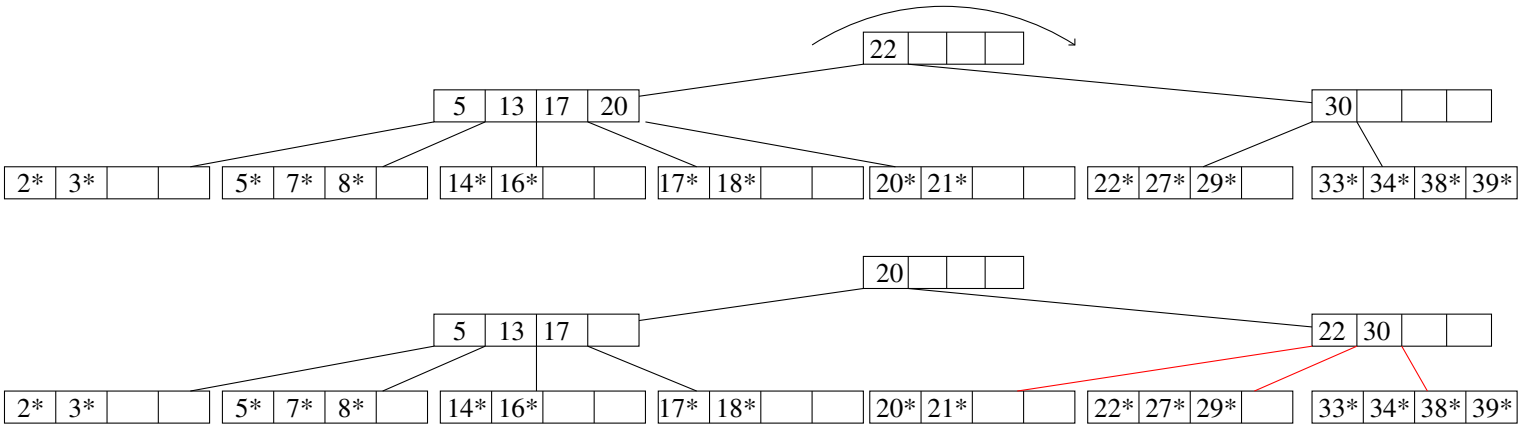




Exemplo - Redistribuição de Nodo Não-Folha



Exclusão de 24:



Exclusão - Heurística Recomendada

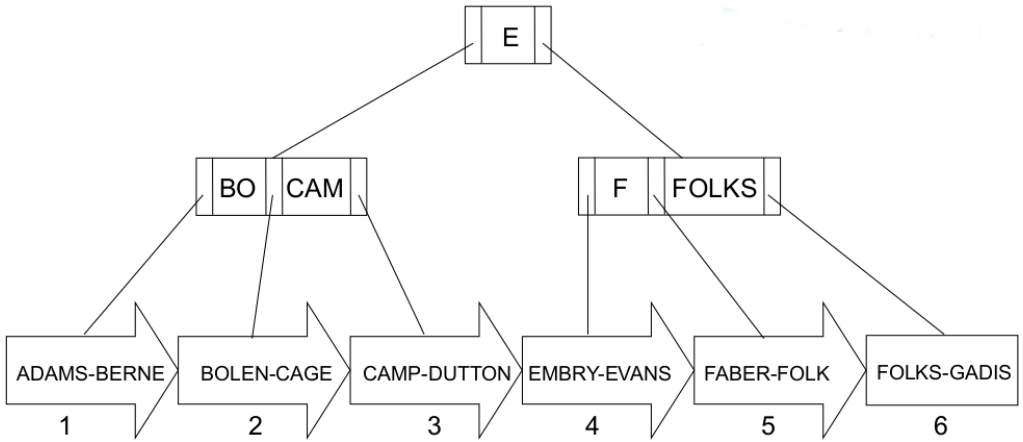
Usar sempre redistribuição para manter os nodos mais vazios, uma vez que os arquivos em geral crescem.

Abordagens para Chaves Duplicadas

1. Usar páginas de overflow (como o ISAM)
2. Usar um identificador como parte da chave. A chave na árvore passa a ser $(chave, id)$.

Compressão de Chaves de Prefixo

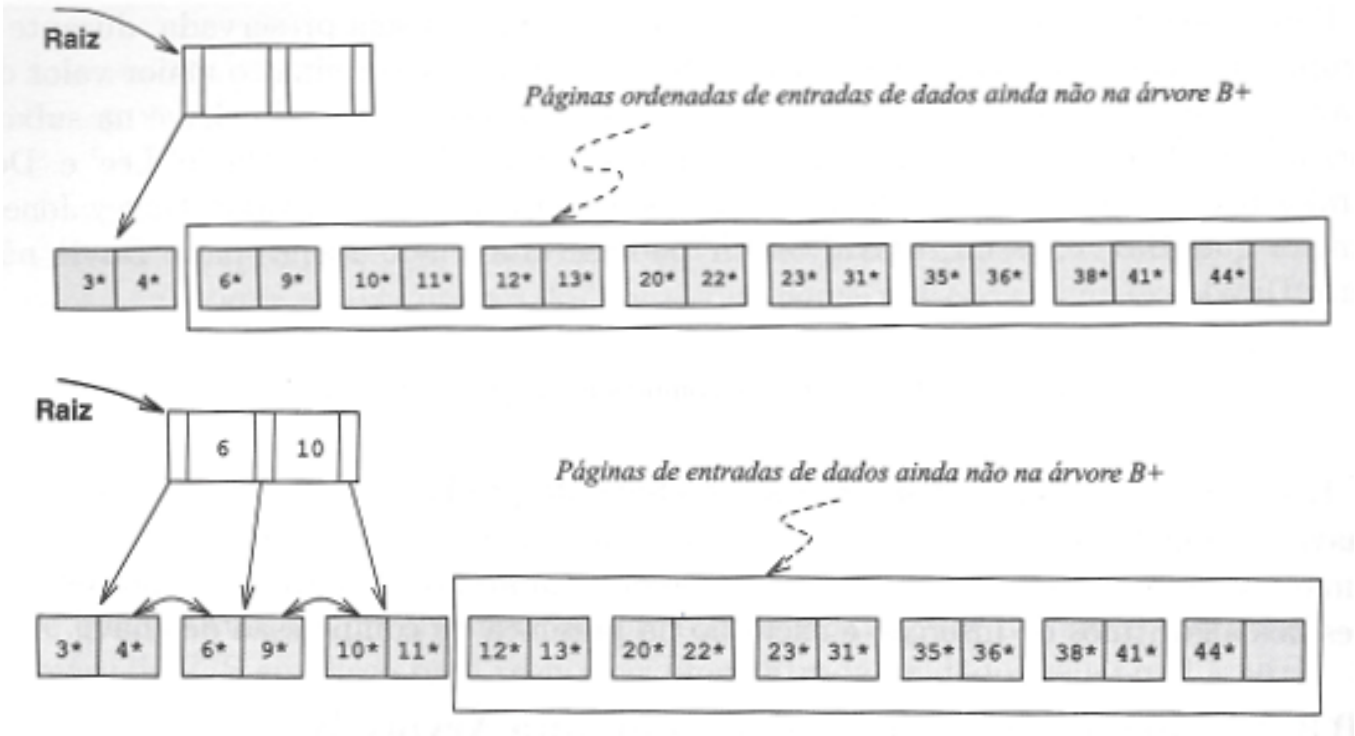
- Bastante útil para casos em que a chave é longa (p.ex. strings)
- Utiliza um prefixo das chaves como "separadores" nos nodos índice



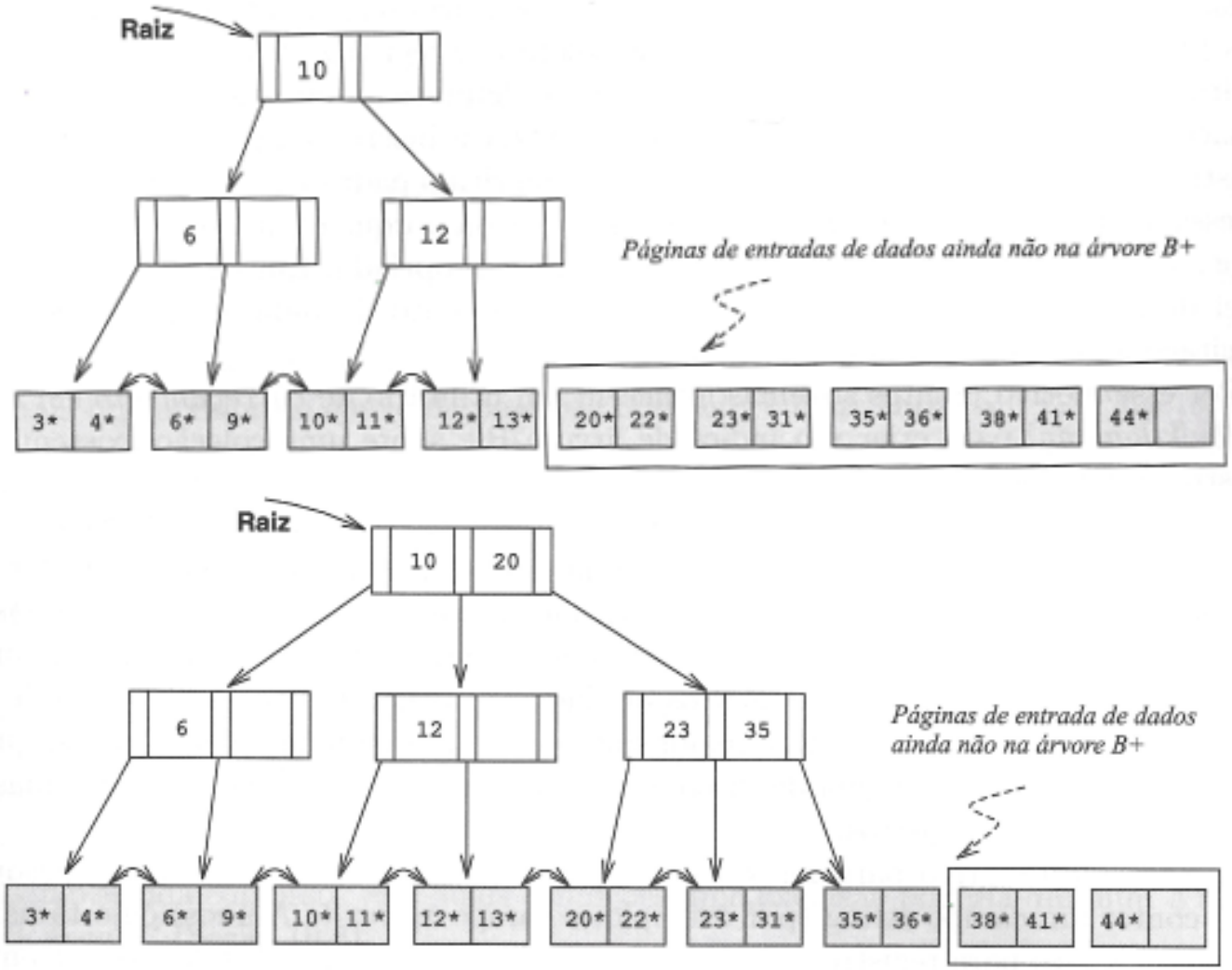
Carregamento em Massa - *Bulkloading*

- Criação de uma estrutura de indexação sobre dados já existentes
- Inserção de cada elemento individualmente é muito custosa
- Carregamento em massa:
  1. ordenação pela chave
  2. construção do índice a partir das entradas ordenadas

Carregamento em Massa - Exemplo



Carregamento em Massa - Exemplo (cont.)



Árvore B+ - Acesso Concorrente

- Podem ocorrer problemas se um processo estiver lendo a estrutura e outro inserindo uma nova chave que causa divisão de um nodo
- Uma página é segura se não há possibilidade de mudança na estrutura da árvore devido à inserção ou remoção na página
  - Inserção: página é segura se $\#chaves < 2t - 1$
  - Remoção: página é segura de $\#chave > t - 1$
- Protocolo de bloqueio
  - `lock-R`: bloqueio compartilhado para leitura
  - `lock-W`: bloqueio exclusivo para atualização

Bloqueio da estrutura para Leitura

1. lock-R(raiz)
2. read(raiz) e torne-a página corrente
3. enquanto a página corrente não for uma folha:
   - lock-R(descendente)
   - unlock(página corrente)
   - read(descendente) e torne-a a página corrente

Bloqueio da estrutura para Atualização

1. lock-W(raiz)
2. read(raiz) e torne-a a página corrente
3. enquanto a página corrente não for uma folha
   - lock-W(descendente)
   - read(descedente) e torne-a a página corrente
   - se a página corrente for segura:
     - unlock(p) para todos os nodos p antecedentes que tenham sido bloqueados

Referências

- Sistemas de Gerenciamento de Banco de Dados, 3ed Ramakrishnan & Gerke, Cap.10
- livro Nivio Ziviani, Cap. 6, Seção 6.3