PUCRS
Pontifícia Universidade Católica
do Rio Grande do Sul

ESCOLA
POLITÉCNICA

Árvores B

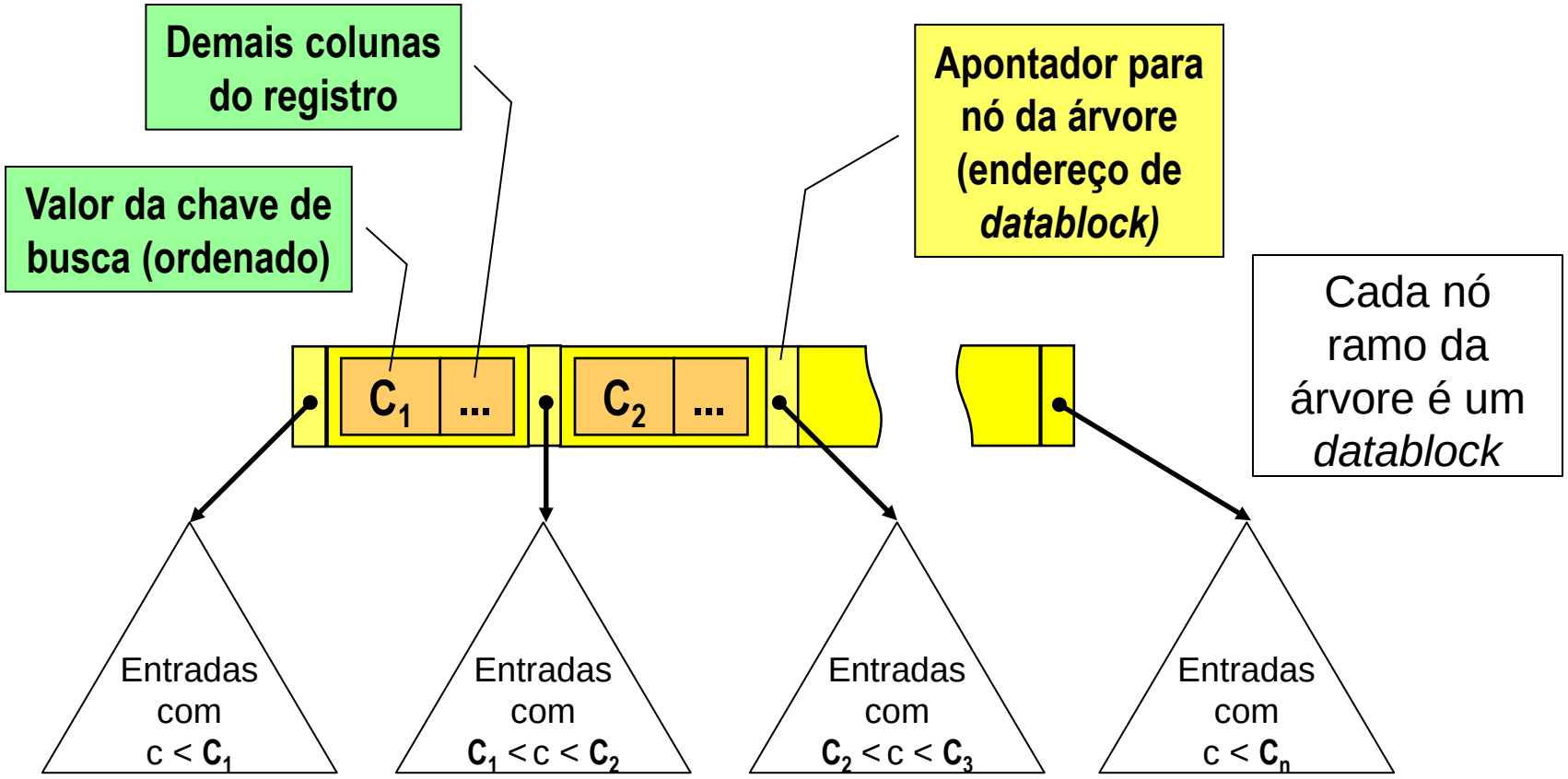
98H00-04 - Infraestrutura para Gestão de Dados

Prof. Msc. Eduardo Arruda
eduardo.arruda@pucrs.br

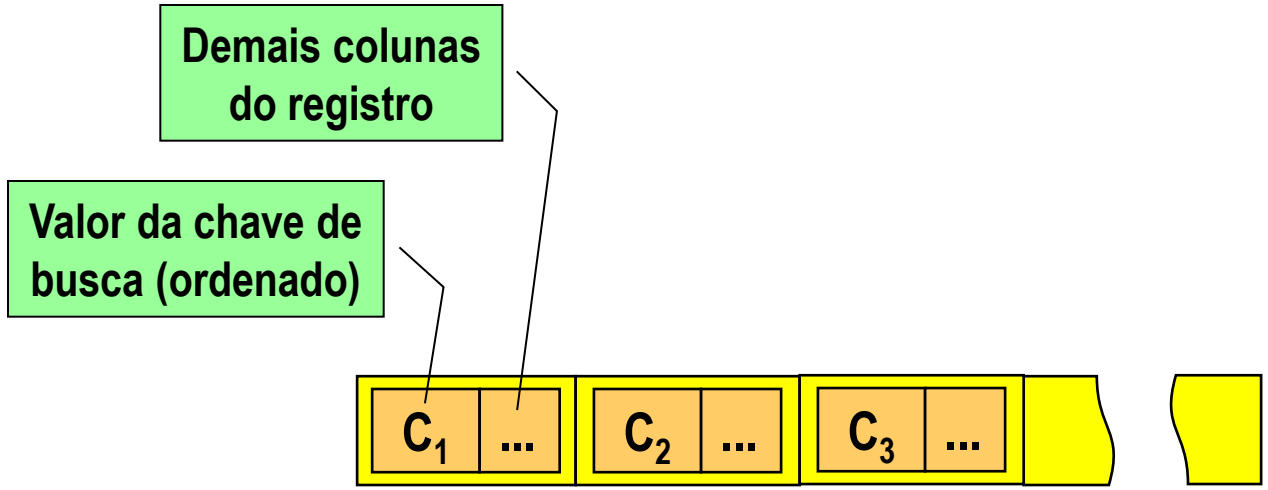
Árvores B

- Não necessita sequencialidade dos dados
- Mantém automaticamente a quantidade de níveis adequada ao tamanho do arquivo que está sendo indexado
  - B = *Balanced* (todos os ramos tem o mesmo comprimento)
  - Garante escalabilidade quase linear
- Balanceamento → aumento do desempenho
- Gerenciam o espaço nos blocos, de modo a não haver blocos de estouro para o índice
  - Algoritmo mantém o espaço nos blocos entre meio cheio e completamente cheio
- Podem ser aplicadas na implementação de índices primários e secundários

Nó interno (ramo ou raiz) padrão de árvores B como índice primário

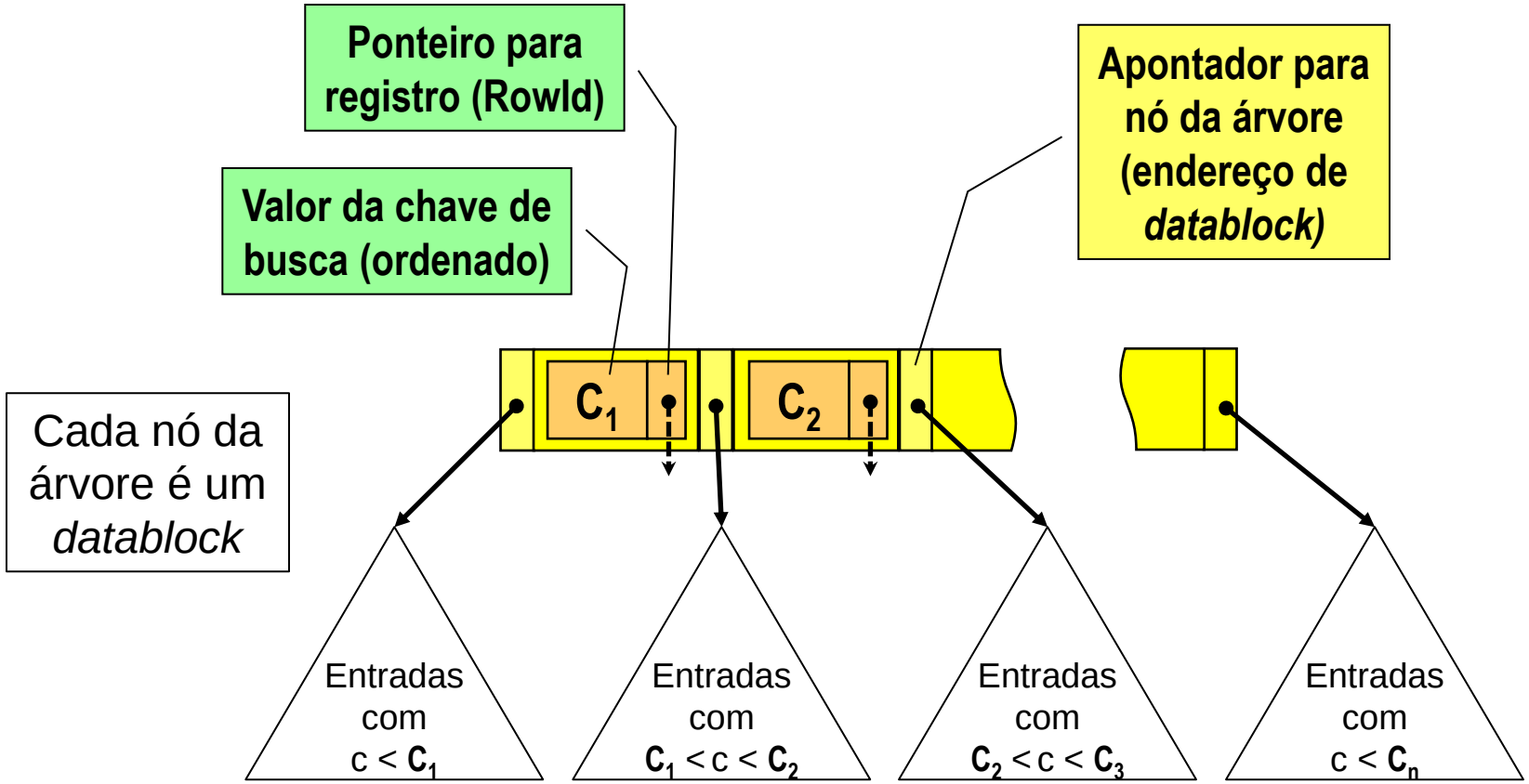


Nó folha padrão de árvores B como índice primário

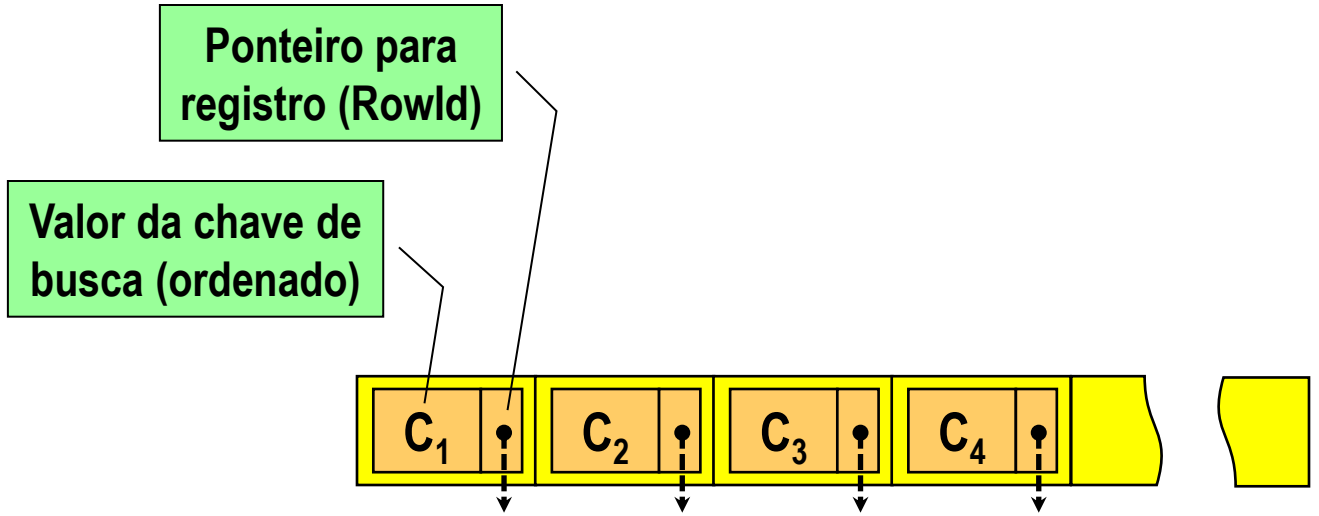


Cada nó folha da árvore também é um *datablock*

Nó interno (ramo ou raiz) padrão de árvores B como índice secundário



Nó folha padrão de árvores B como índice secundário



Níveis em árvores B

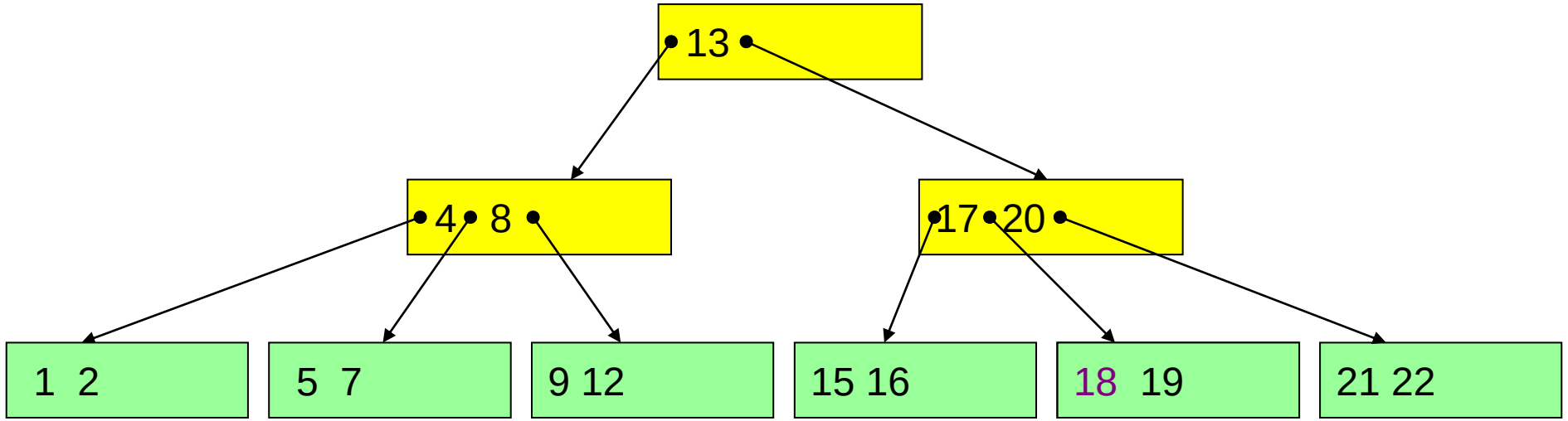
- Uma árvore B tem uma ordem **n**
- Cada nó, exceto o nó raiz, obrigatoriamente deve ter entre $\lfloor n/2 \rfloor$ e n entradas
  - O nó raiz pode ter apenas 1 entrada (ou nenhuma, se a árvore estiver vazia)
- Cada nó interno é da forma $P_0R_1P_1R_2P_2\ldots R_nP_{n+1}$, onde:
  - P_i é um ponteiro para outro nó da árvore, contendo apenas entradas de índice cujos valores são maiores que C_i e menores que C_{i-1}
  - R_i pode ser:
    - Se for um índice primário, um registro
    - Se for um índice secundário, uma estrutura do tipo C_iP_i, onde:
      - C_i é o valor da chave de busca
      - P_i é o ponteiro (RowId) para o registro

Busca em árvores B

- Seja X o valor buscado
  1. Nó raiz é lido
  2. Faz-se uma busca nas k chaves do nó
     2.1. Se X for encontrado, o algoritmo termina retornando Verdadeiro e o registro buscado
     2.2. Senão, X corresponde ao intervalo de algum P_i
        2.2.1 Se P_i é nulo, o algoritmo termina retornando Falso
        2.2.2 Senão, o nó apontado por P_i é lido e retorna-se ao passo 2

Exemplo de busca em árvores B

X = 18



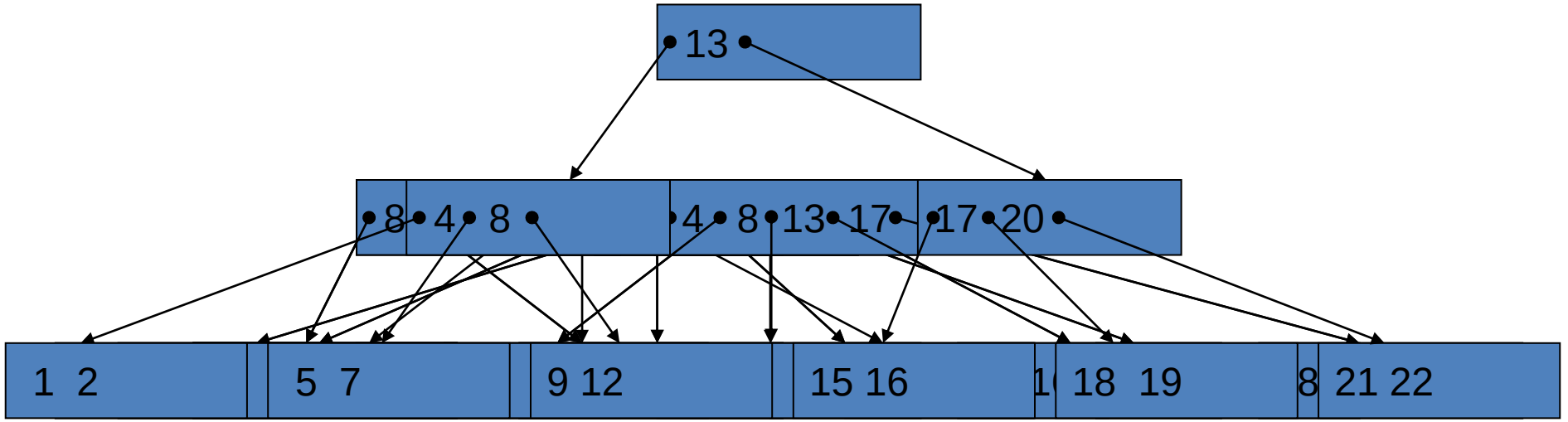
Inserção em árvores B

- Buscar o nó onde o valor X deve ser inserido
- Se o nó ainda tem lugar, algoritmo termina (insere ordenado)
- Senão, dividir as n+1 chaves em três grupos:
  [1] as $\lfloor n/2 \rfloor$ menores chaves,
  [2] a chave mediana e
  [3] as $\lceil n/2 \rceil$ maiores chaves
- O grupo [1] permanece no nó e o grupo [3] vai formar um novo nó
- A chave mediana, é inserida (recursivamente se necessário) no nó pai
  - Se não existe nó pai, uma nova raiz é criada

Inserção em árvores B

N=4

Ordem de Inserção: 8 5 15 12 7 16 13 9 1 2 4 17 19 18 21 22 20



Exclusão em árvores B

- O algoritmo de exclusão de uma chave é um pouco mais complexo do que o de inserção
- Há vários casos a serem considerados:
  - Nó que contém a chave é interno ou folha?
  - Há risco de "*underflow*" (nó fica com chaves de menos) ou não?

Exclusão em árvores B

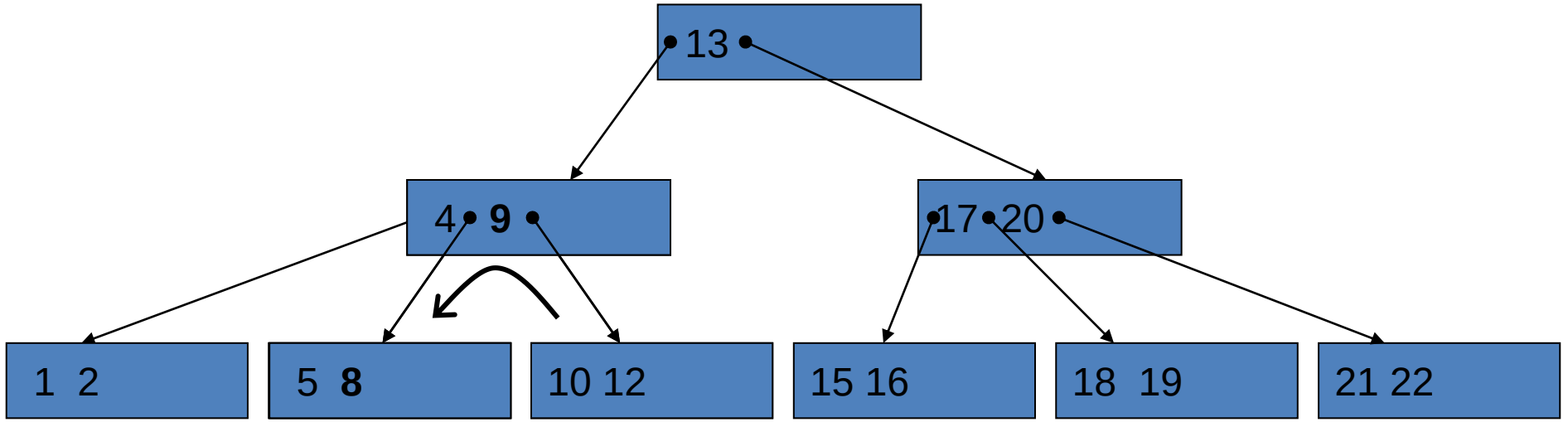
- Se a chave está num nó folha
  - Se o nó não corre risco de *underflow*, a chave é removida e o algoritmo termina
  - Senão, 2 situações são possíveis:

    (1) O nó vizinho à esquerda ou à direita têm uma chave para emprestar, isto é, não correm o risco de *underflow*
    - Neste caso, fazer operação de **empréstimo**

    (2) Ambos os vizinhos têm o número mínimo de chaves
    - Neste caso fazer a operação de **combinação** do nó com um de seus vizinhos

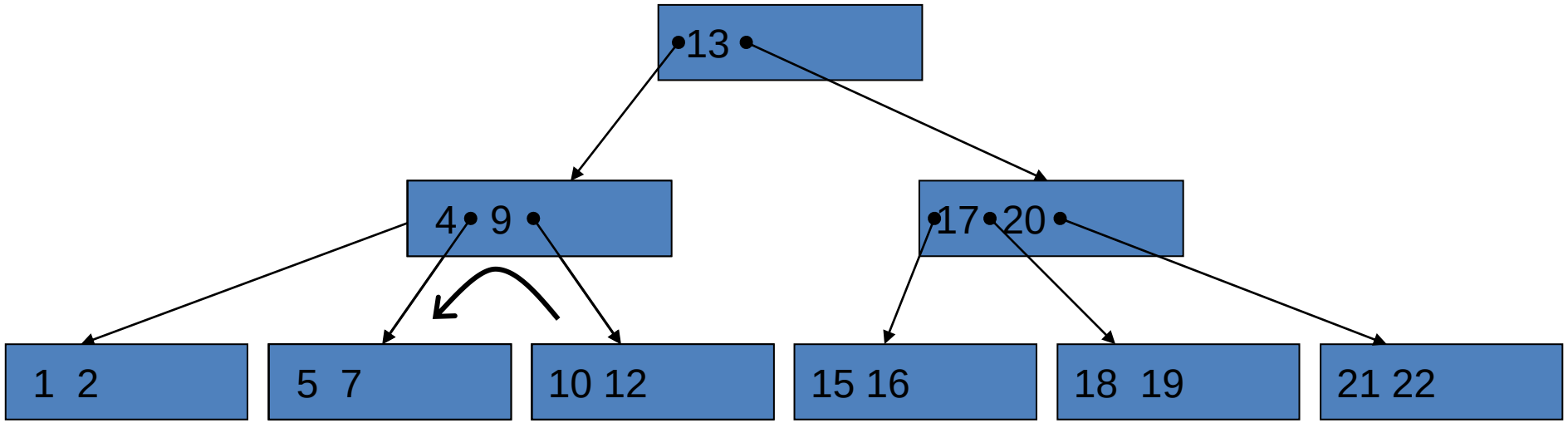
Operação de “empréstimo”

Remoção da chave 7



Operação de “empréstimo”

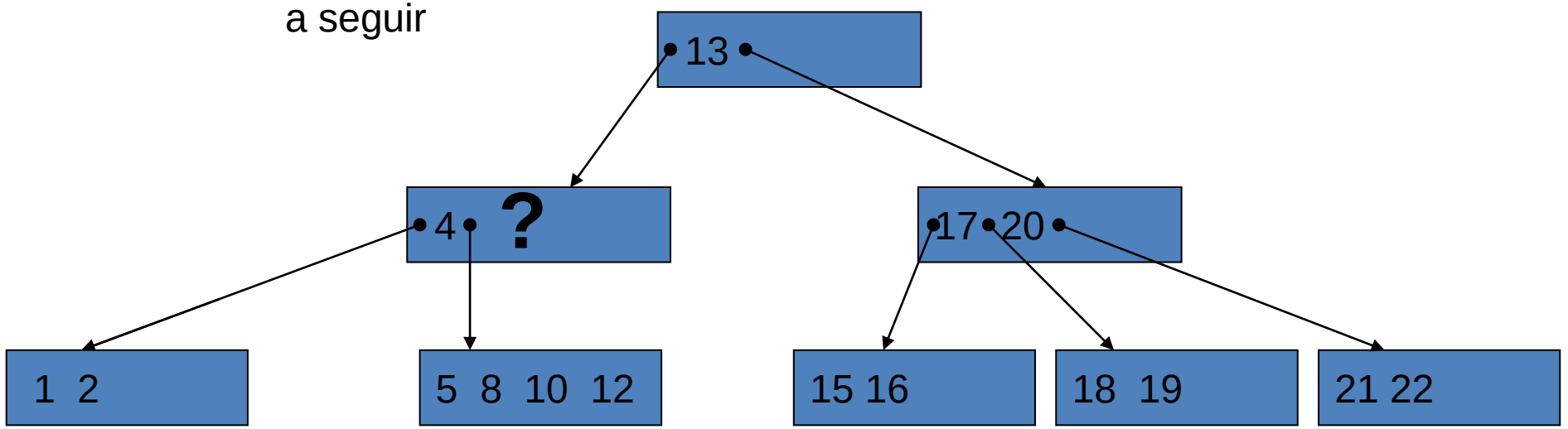
- Se a chave está num nó ramo
  - Se é possível fazer uma operação de empréstimo entre os nós filhos à esquerda e à direita, então fazê-la e remover a chave do filho que a recebeu



Operação de Combinação

Remoção da chave 7

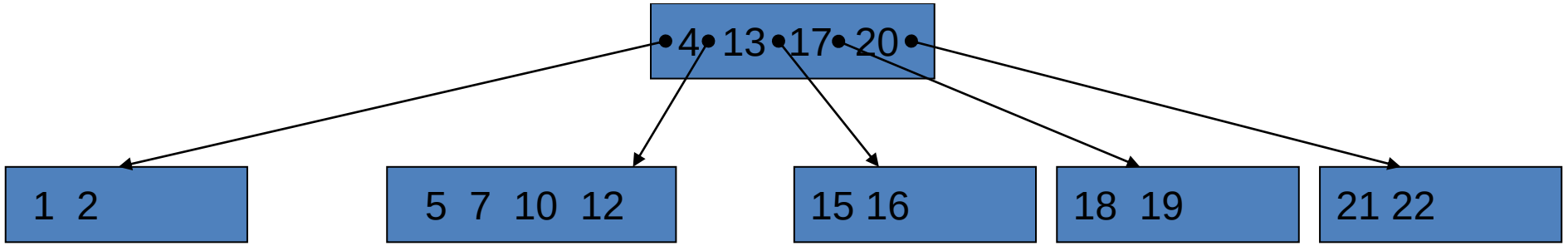
Solução é vista
a seguir



Remoção em árvores B

- Se a chave está num nó interno e não é possível fazer uma operação de empréstimo entre os nós filhos à esquerda e à direita:
  - Combine-os (junto com a chave) e remova a recursivamente a chave do nó filho resultante.
  - Se o nó interno resultante estiver com underflow, aplicar um empréstimo com um nó vizinho se possível, ou então uma combinação e assim sucessivamente.

Exemplo de Remoção

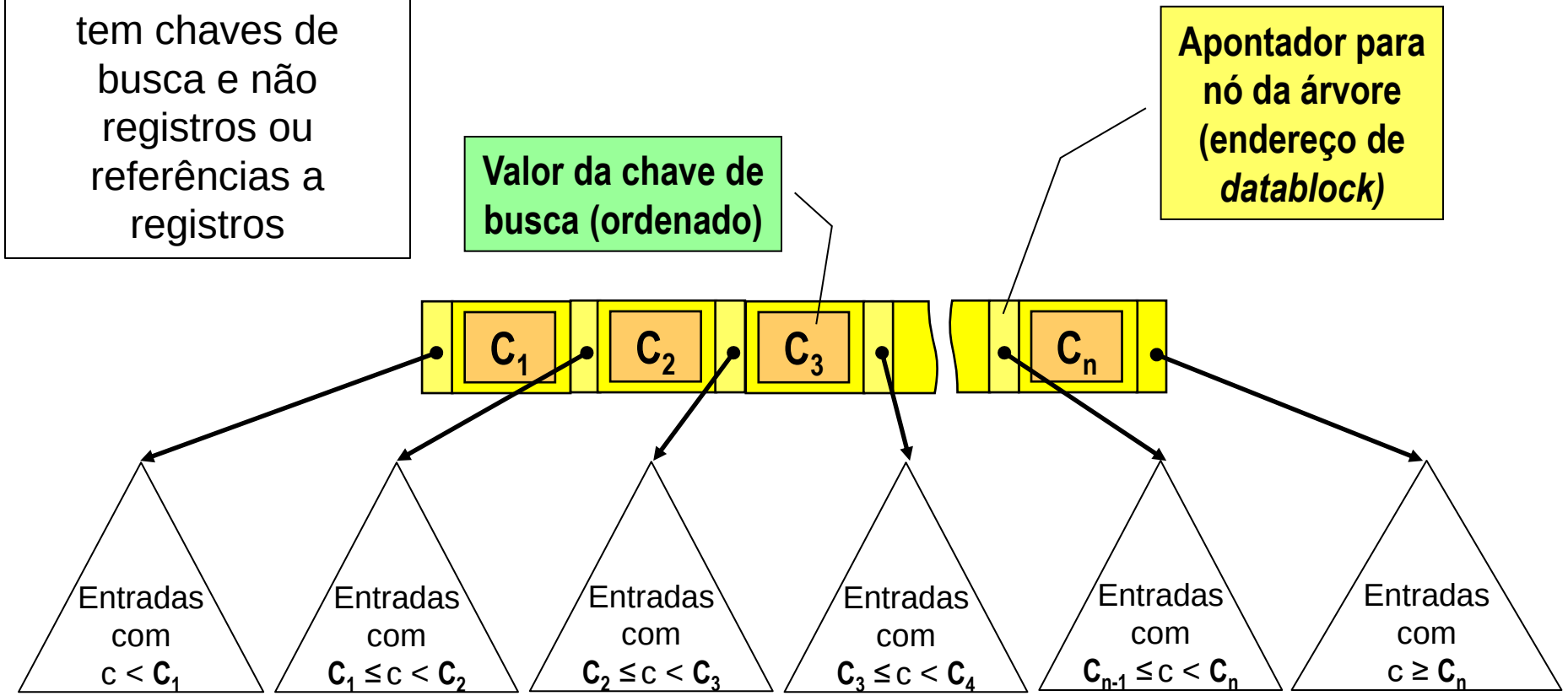


Árvores B+

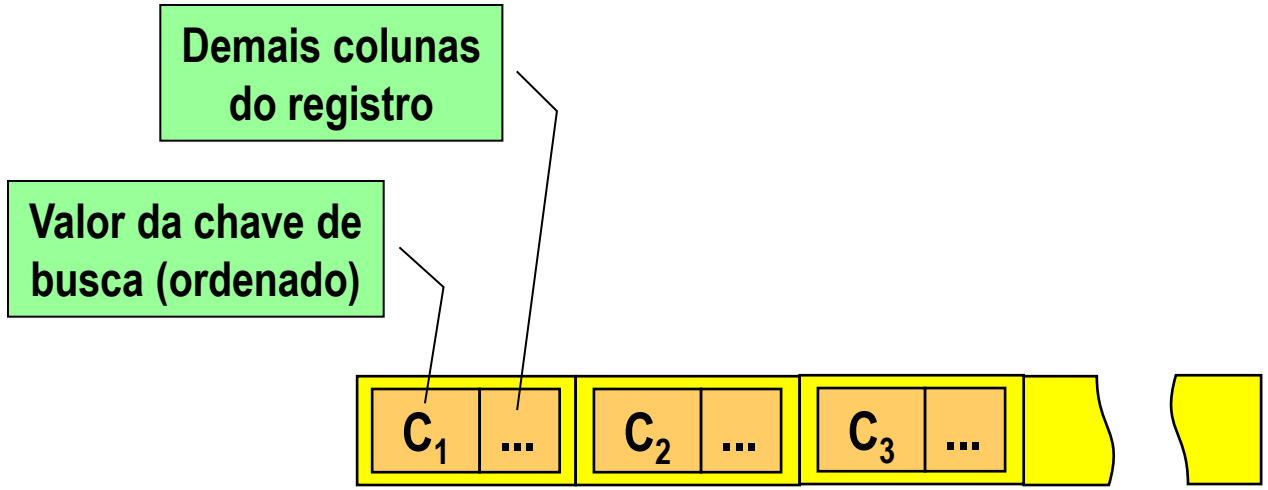
- São uma variante das árvores B
- Diferentemente das árvores B, nas árvores B+ os registros de dados são referenciados somente nos nós folha
- Todos os nós folhas são ligados entre si, proporcionando acesso sequencial aos registros ordenados

Nó interno padrão de árvores B+

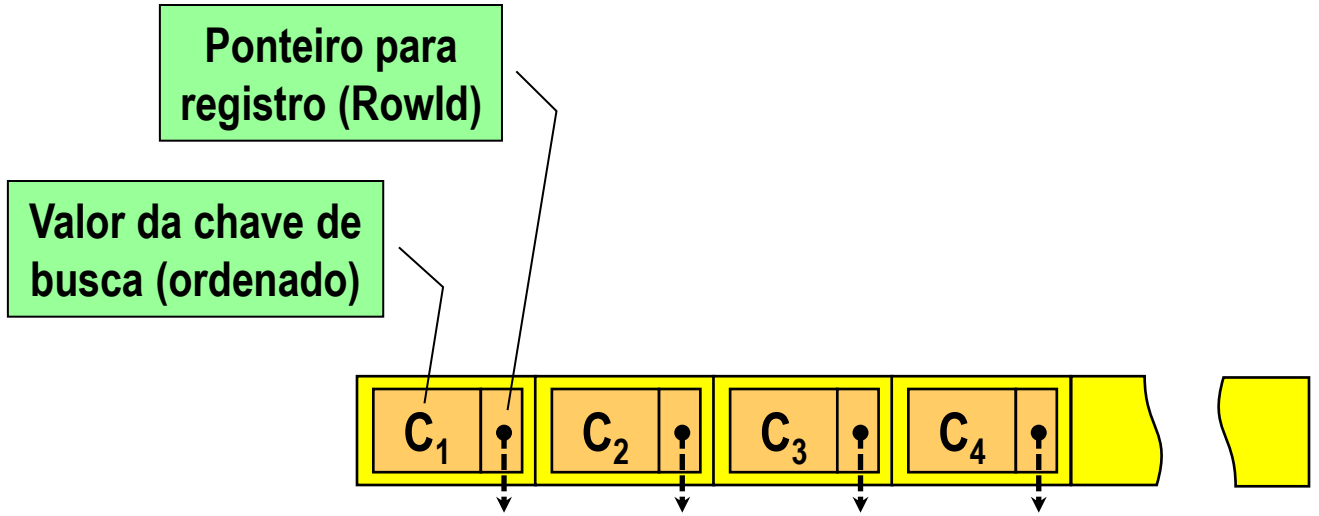
Os nós internos só tem chaves de busca e não registros ou referências a registros



Nó folha padrão de árvores B como índice primário

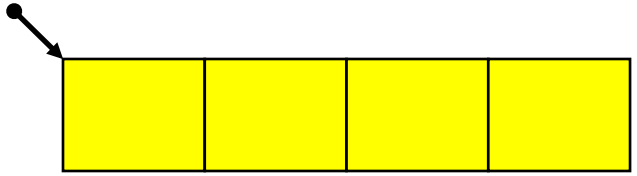


Nó folha padrão de árvores B como índice secundário

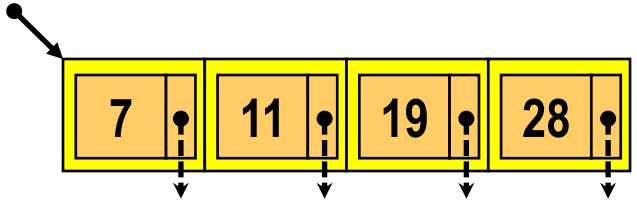


Exemplo de árvore B+

- Seja uma árvore de ordem n = 2 inicialmente vazia

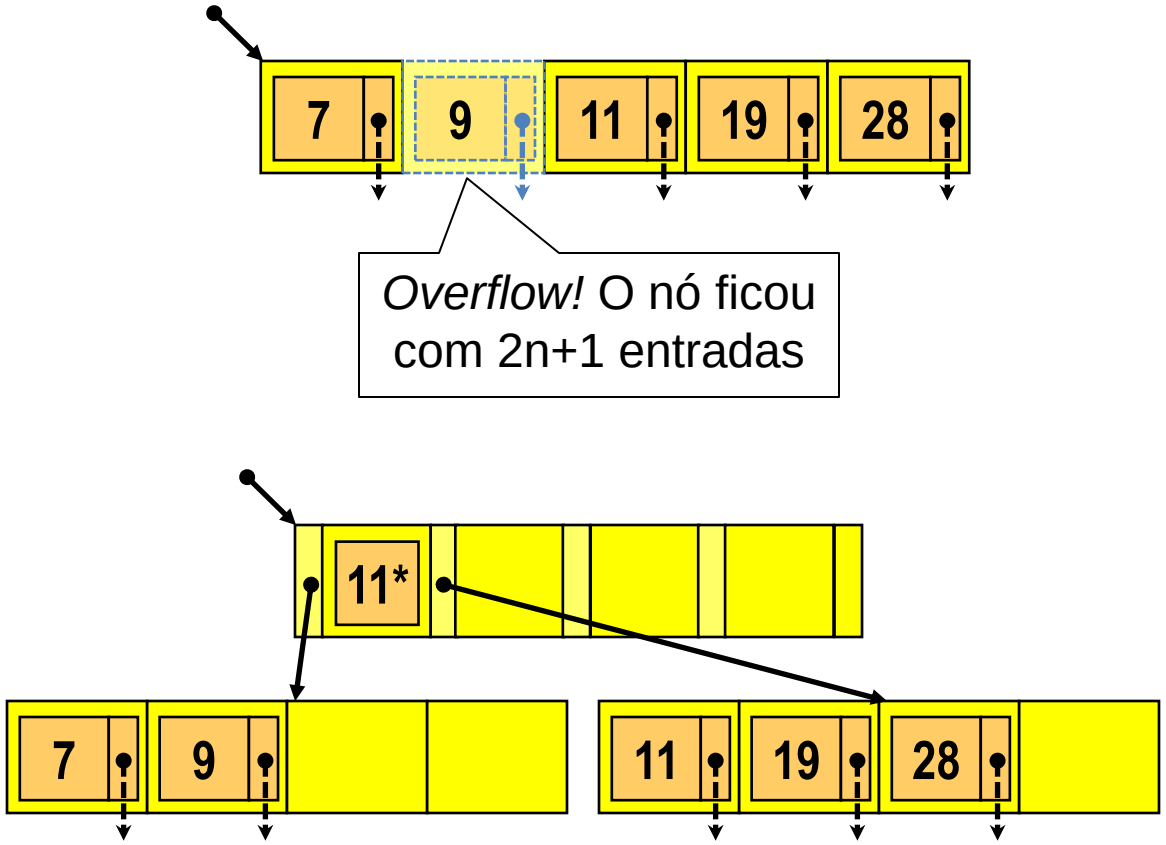


- Considere a inserção de registros com as chaves de busca 7, 11, 19 e 28



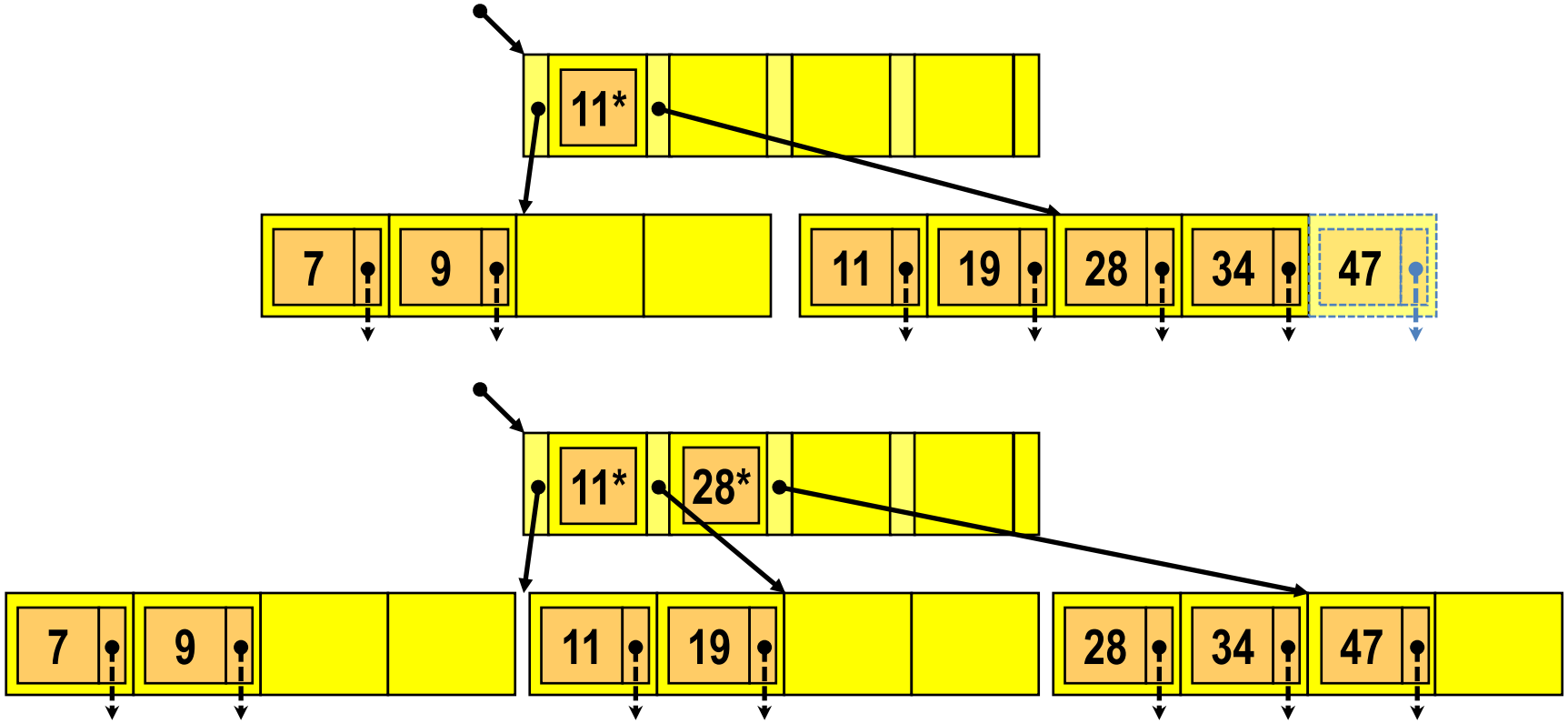
Exemplo de árvore B+

- Considere a inserção dos registro com a chave de busca 9



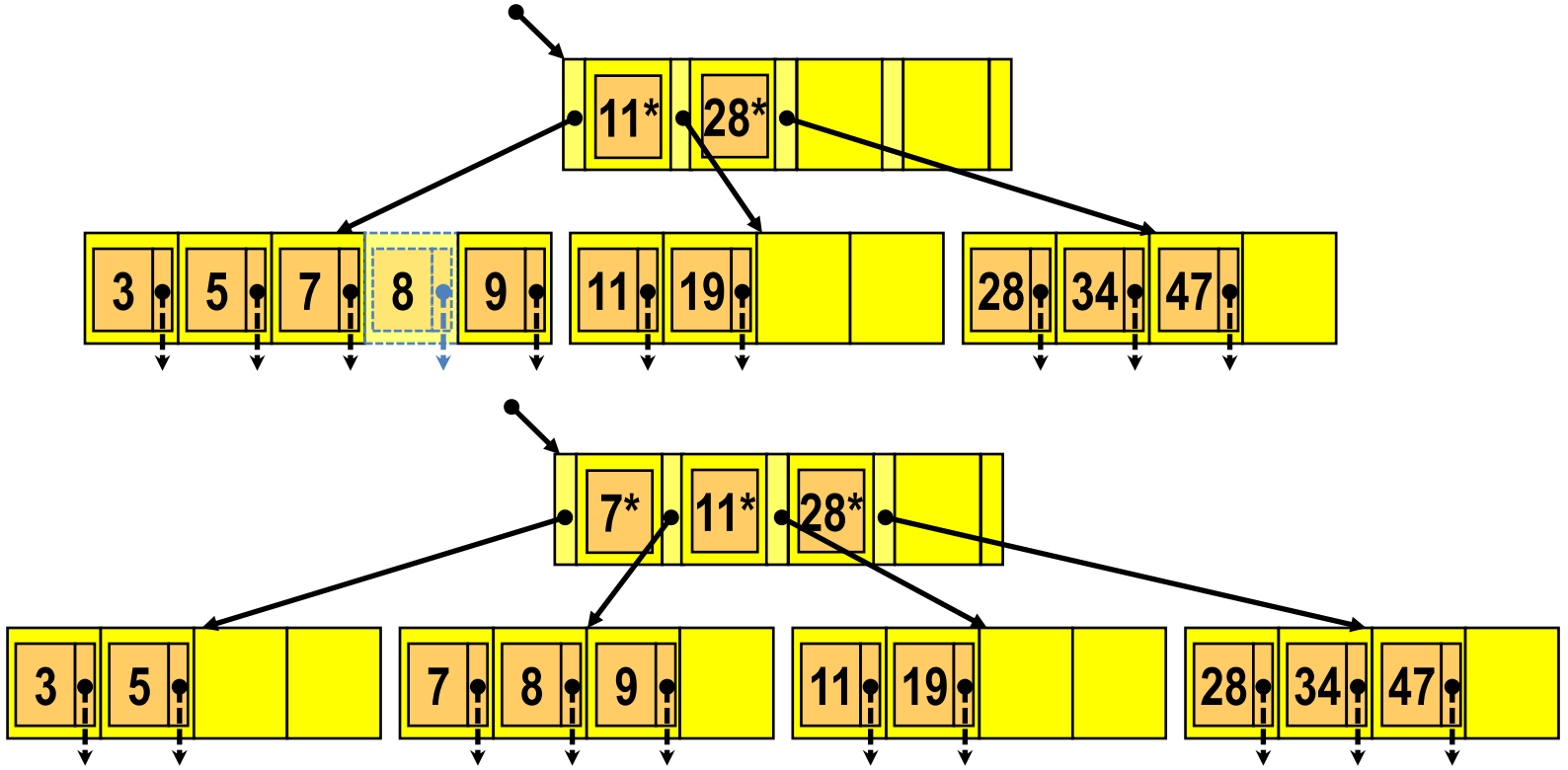
Exemplo de árvore B+

- Considere a inserção dos registros com as chaves de busca 34 e 47



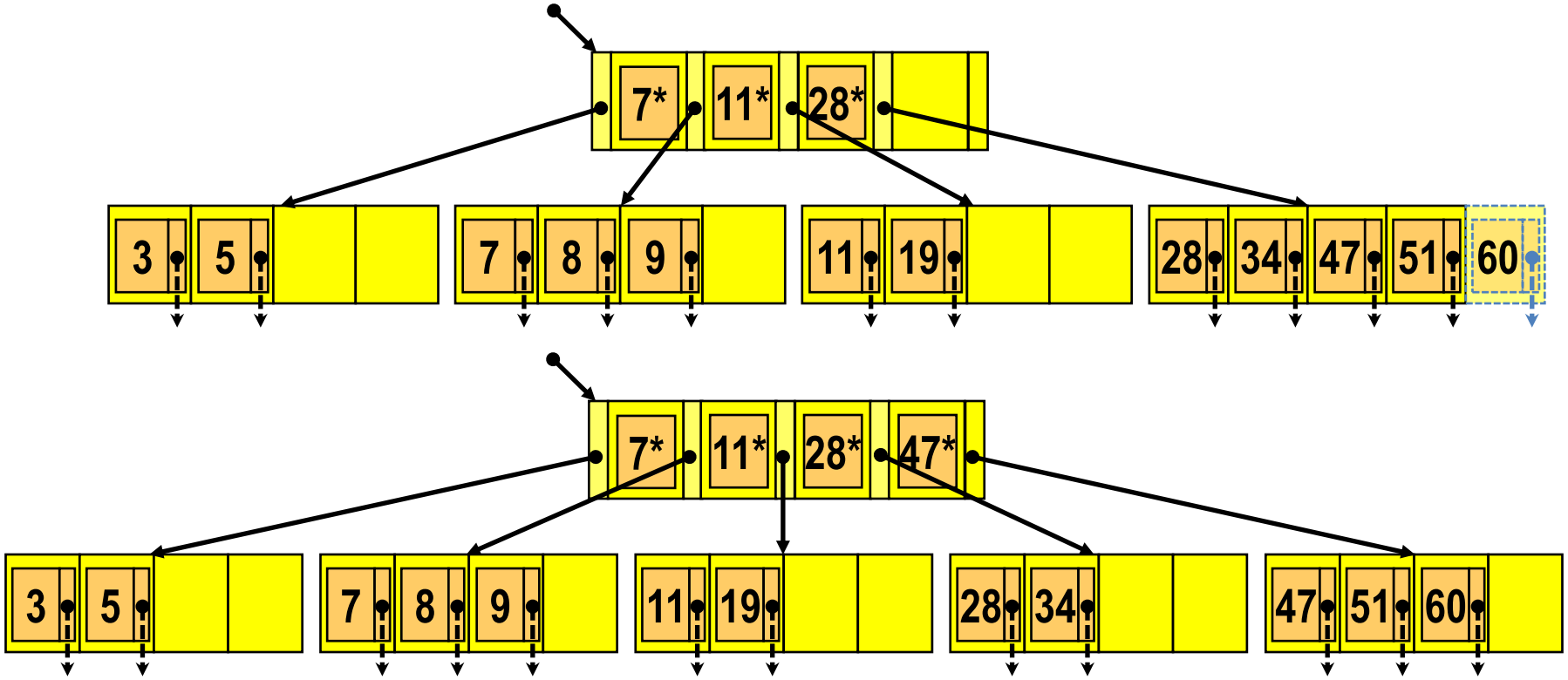
Exemplo de árvore B+

- Considere a inserção dos registros com as chaves de busca 3, 5 e 8



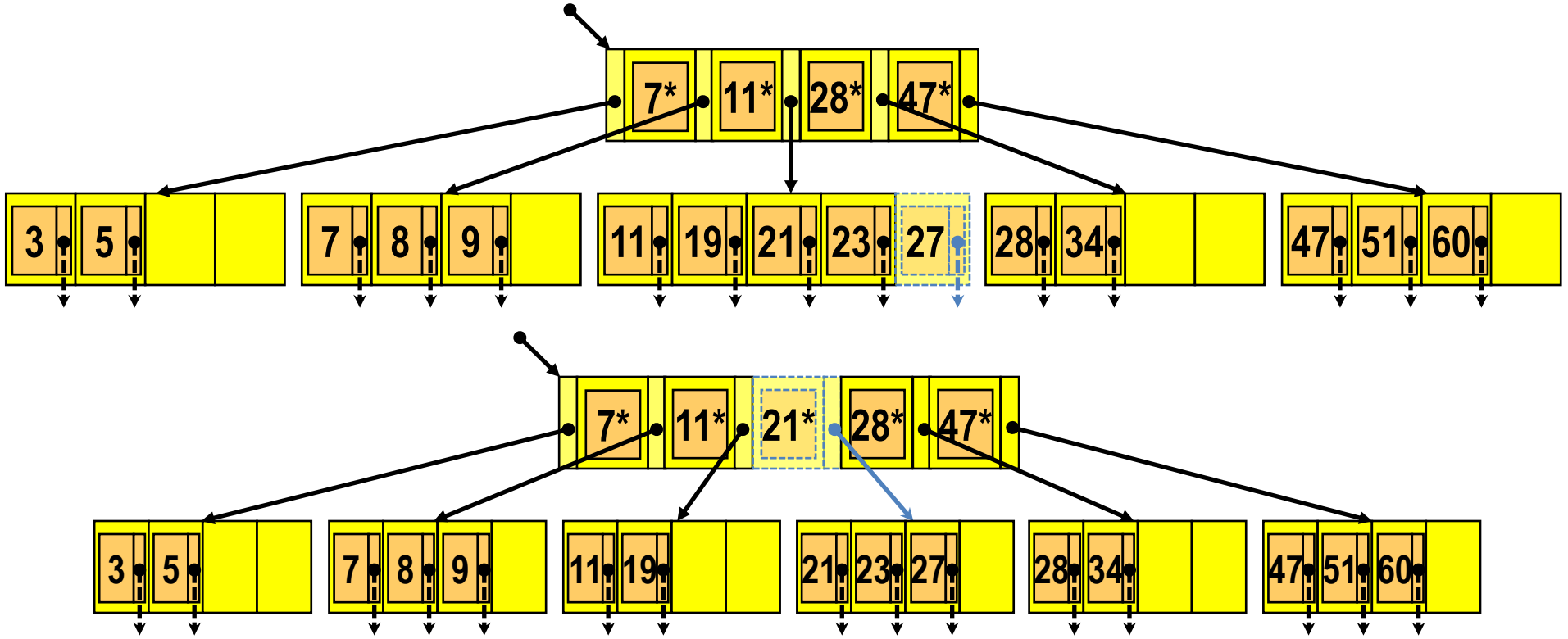
Exemplo de árvore B+

- Considere a inserção dos registros com as chaves de busca 51 e 60

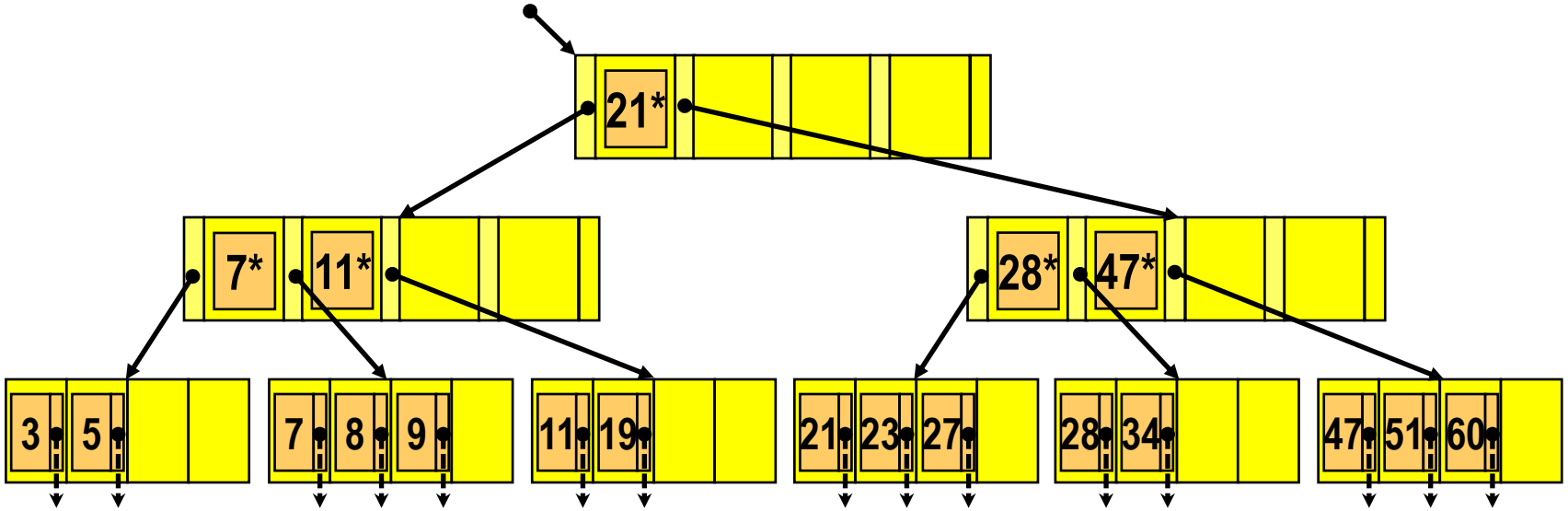


Exemplo de árvore B+

- Considere a inserção dos registros com as chaves de busca 21, 23 e 27



Exemplo de árvore B+



Árvores B X Árvores B+

Árvore B:

- Recupera mais rápido registros dos nós internos
- Nós folha e internos possuem tamanhos diferentes
- Deleção é mais complicada

Árvore B+:

- Acesso igual para qualquer chave
- Permite pesquisa por faixas de forma mais fácil
- Permite pesquisa diretamente pelo nível folha do índice

Árvore B+ → MELHOR PARA BD!

Operações de I/O em árvores B+

- Um dado relevante sobre o uso de Árvores B+ em bancos de dados é o número de níveis, ou seja, o número de operações de entrada e saída no acesso
- Considere
  - BD com *datablock* de 8KB (8192B)
  - Tabela com registros de 200B e chave de busca de 8B
  - *Overhead* médio de 10% com o *header* dos *datablocks*
  - *Rowid* de 8B
- Neste caso
  - Ordem dos nós internos do índice =
    - (datablock – overhead – 1 Rowid) / (1 Chave + 1 Ponteiro)
    - (8192 – 819 – 8B) / (8B + 8B) ≈ 460 entradas em nós internos → ordem = **230**
  - Ordem dos nós folha do índice =
    - (datablock – overhead) / (registro)
    - (8192 – 819) / 200 ≈ 36 entradas em nós folha → ordem = **18**

Operações de I/O em árvores B+

- Considerando que a árvore B+ tem 4 níveis e implementa índice secundário, um acesso a registro é realizado lendo 5 *datablocks*
  - raiz + ramo + ramo + folha + *datablock* de dados que contém o registro
- Esta árvore é capaz de endereçar
  - 460 * 460 * 460 * 36 registros ou
  - 35.040.960 registros
- Ou seja, neste exemplo, lendo 5 *datablocks* (8KB * 5 = 40KB) é possível
  - Ler um registro entre mais de 35 milhões de registros ou
  - Ler um registro em 7.786.880KB ≈ 7GB de dados
- E para agilizar ainda mais, os blocos internos dos índices costumam ficar armazenados em *buffer*

Exercício 1

- Inserir as seguintes chaves em uma árvore B de ordem 4
  - 50, 25, 75, 10, 35, 65, 85, 90, 15 e 18

- Animação
  - https://www.cs.usfca.edu/~galles/visualization/BTree.html

Exercício 2

- Inserir as seguintes chaves em uma árvore B+ de ordem 4
  - 50, 25, 75, 10, 35, 65, 85, 90, 15 e 18
- Remover
  - 65, 50 e 85
- Animação
  - https://www.cs.usfca.edu/~galles/visualization/BTree.html

Exercícios complementares

1. Para blocos que contém dez registros ou 99 chaves e 100 ponteiros e supondo que o nó médio da árvore B+ esteja 70% ocupado (69 chaves e 70 ponteiros), obtenha:
   a) o número total de blocos para um arquivo de 1.000.000 de registros;
   b) o número de operações de E/S para se obter um registro dada a sua chave.

Dúvidas?