Estrutura de Dados

Árvores B

B-tree

Profa.: Márcia Sampaio Lima

EST - UEA

Árvores B

- São árvores balanceadas projetadas para trabalhar com dispositivos de armazenamento secundário como discos magnéticos.

- Elas visam otimizar as operações de entrada e saída nos dispositivos.

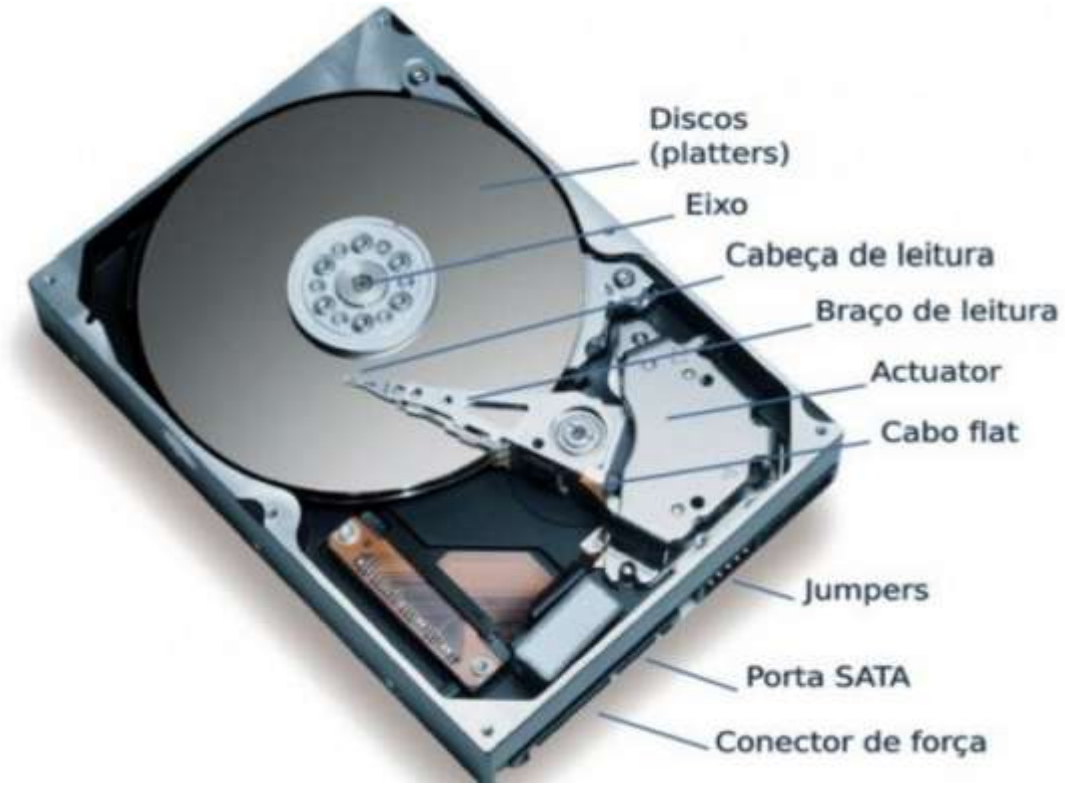
Árvores B

- Permite a inserção, remoção e busca de chaves numa complexidade de tempo logarítmica.
- Por isso, é muito empregada em aplicações que necessitam manipular grandes quantidades de informação tais como um banco de dados.

Árvores B

- O tempo de acesso às informações em um disco é prejudicado principalmente pelo tempo de posicionamento do braço de leitura.
- Uma vez que o braço esteja posicionado no local correto, a leitura pode ser feita de forma bastante rápida.
- Desta forma, devemos minimizar o número de acessos ao disco.

Árvore B

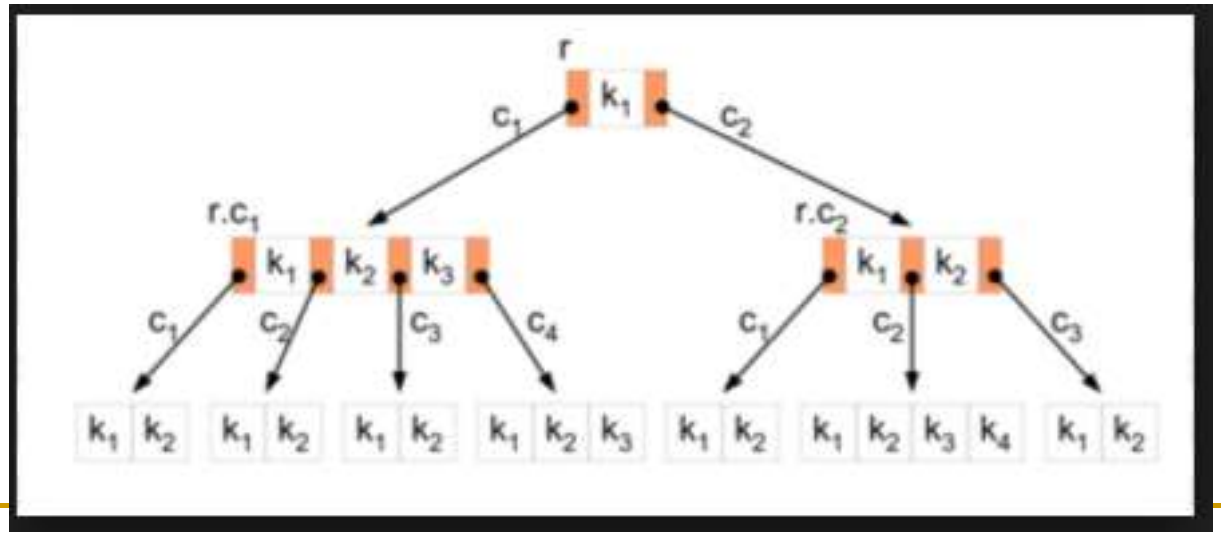

Discos
(platters)
Eixo
Cabeça de leitura
Braço de leitura
Actuator
Cabo flat
Jumpers
Porta SATA
Conector de força

Árvores B

- Idéia das **árvores B:**
  - Trabalhar com dispositivos de memória secundária.
  - Quanto menos acessos a disco a ED proporcionar, melhor será o desempenho do sistema na operação de busca sobre os dados manipulados.

Árvores B

- Diferente das árvores binárias, cada nó em uma árvore B pode ter muitos filhos, isto é, o grau de um nó pode ser muito grande.



Árvore B

- As árvores B permitem manter **mais de uma chave em cada nó** da estrutura.
- Proporciona uma organização de ponteiros de forma que as operações são executadas rapidamente.
- Sua construção assegura **que todas as folhas se encontram no mesmo nível**, não importando a ordem de entrada dos dados

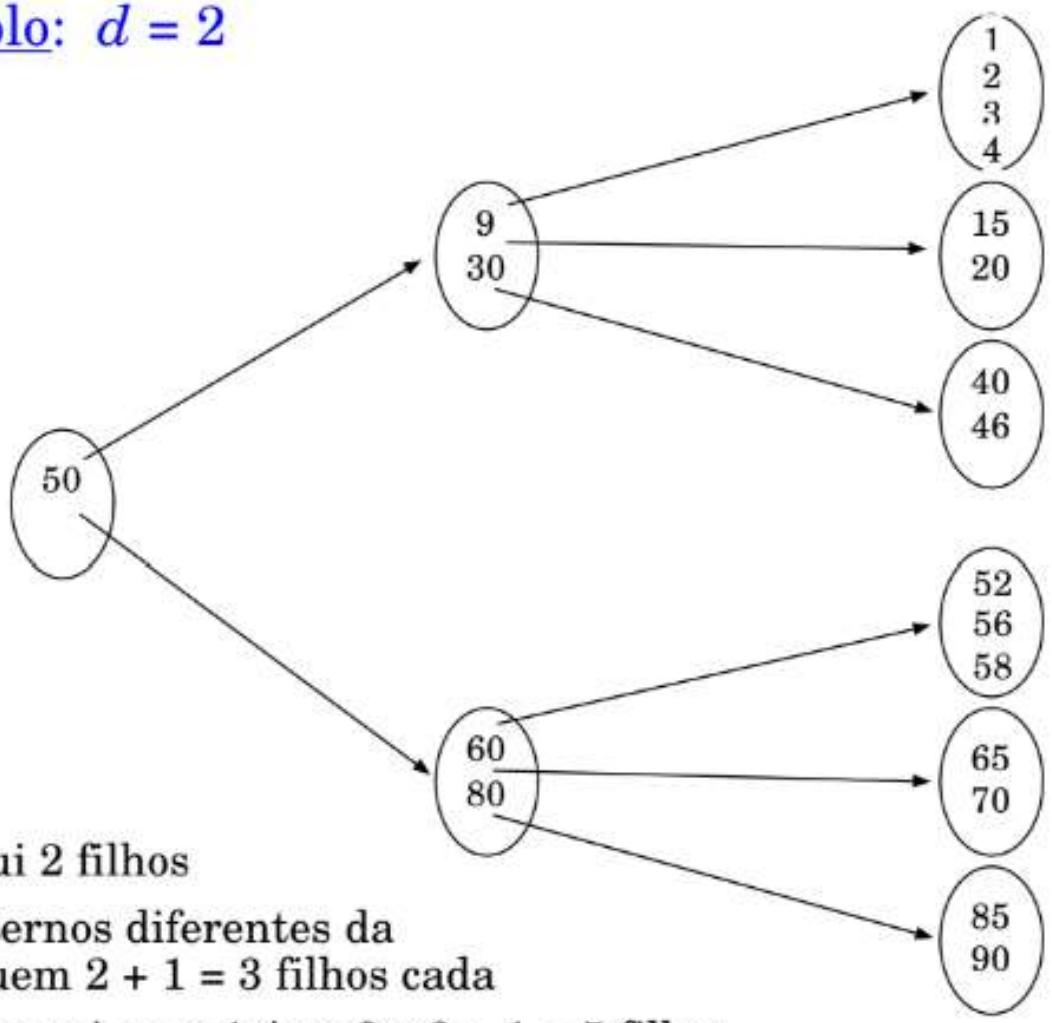
Árvores B

- Definição 1:
  - Seja *d* um número natural, temos:

Uma árvore *B de ordem d* é uma árvore ordenada com as seguintes propriedades:

1. Se a raiz não é folha, possui no mínimo **2 filhos**.
2. Cada nó interno, ≠ raiz, possui no mínimo ***d+1* filhos.**
3. Cada nó possui no máximo ***2d+1*** filhos.
4. Todas as folhas estão no mesmo nível.

Exemplo: $d = 2$

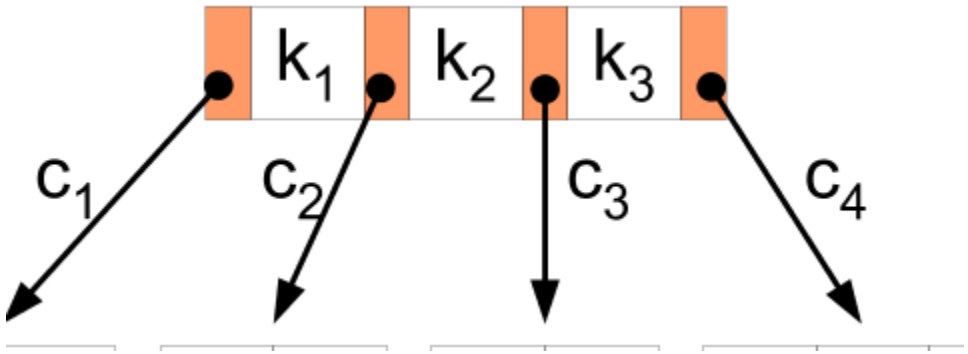


Observe:

1) raiz possui 2 filhos
2) os nós internos diferentes da raiz possuem 2 + 1 = 3 filhos cada
3) cada nó possui no máximo $2 \times 2 + 1 = 5$ filhos
4) todas as folhas estão no nível 3

Árvores B

- Nomenclatura:
  - Nós de uma árvore B são ***páginas***
  - Cada página armazena várias ***chaves***



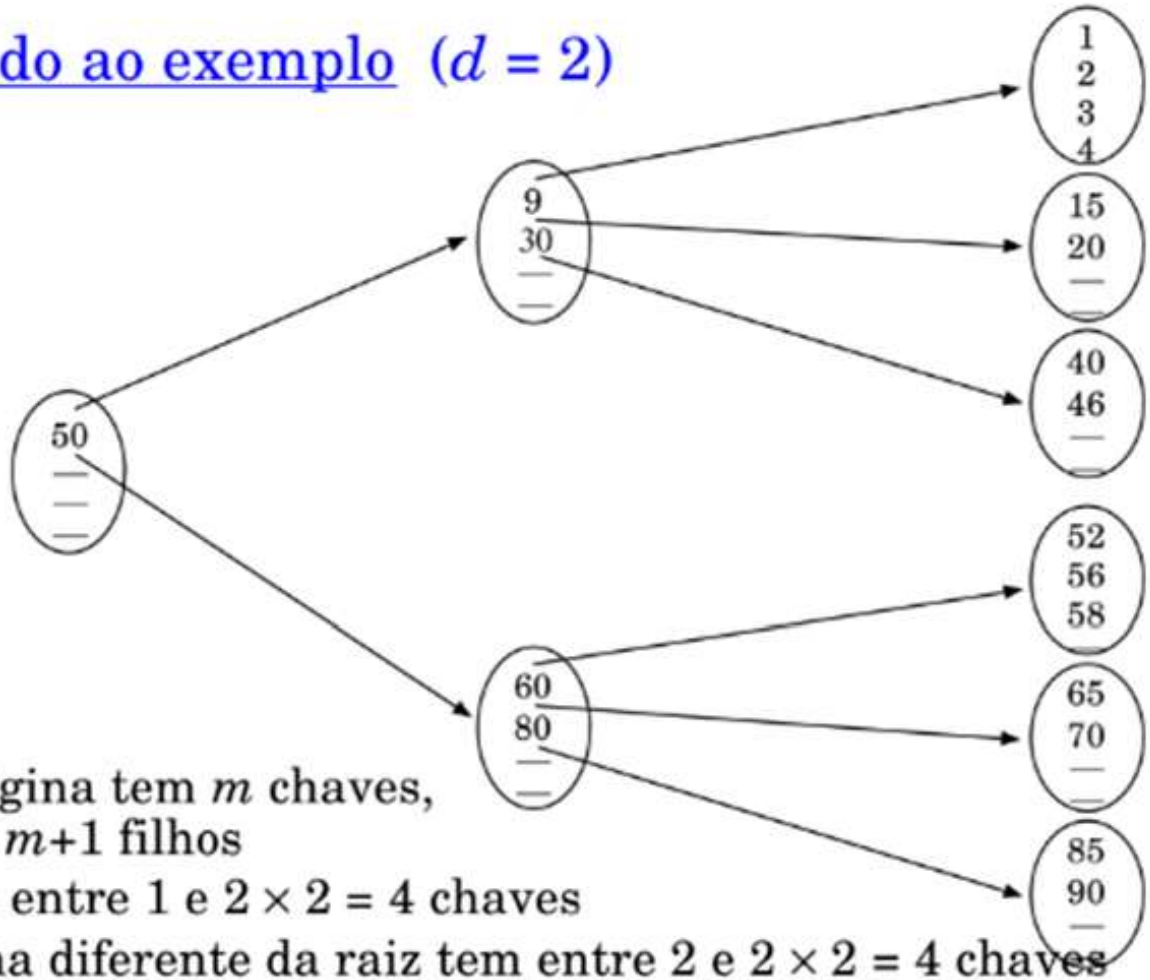
Árvore B

- Propriedades:
  - Se uma página **P** não folha possui **m** chaves, então **P** possui ***m + 1*** filhos.
  - A raiz possui entre **1** e **2d** chaves
  - Cada página ≠ da raiz possui entre **d** e **2d** chaves
  - Em cada página **P** com **m** chaves, as chaves estão ordenadas:

$$s_1 < s_2 < .... < s_m$$

  - **P** contém **m + 1** ponteiros (***p_0, p_1, ..., p_m***) apontando para seus filhos.

Voltando ao exemplo ($d = 2$)



Observe:

a) se uma página tem m chaves, então tem $m+1$ filhos

b) a raiz tem entre 1 e $2 \times 2 = 4$ chaves

c) cada página diferente da raiz tem entre 2 e $2 \times 2 = 4$ chaves

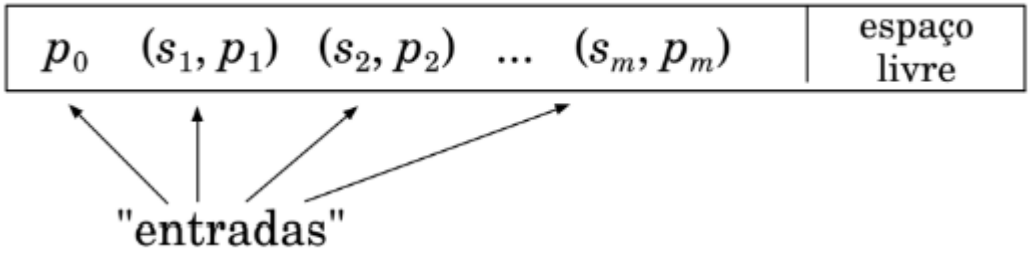
d) em cada página as chaves estão ordenadas

e) de cada página com m chaves partem $m+1$ ponteiros (as folhas têm ponteiros nulos)

Observe a estrutura das entradas, de acordo com a tela anterior.

Árvore B

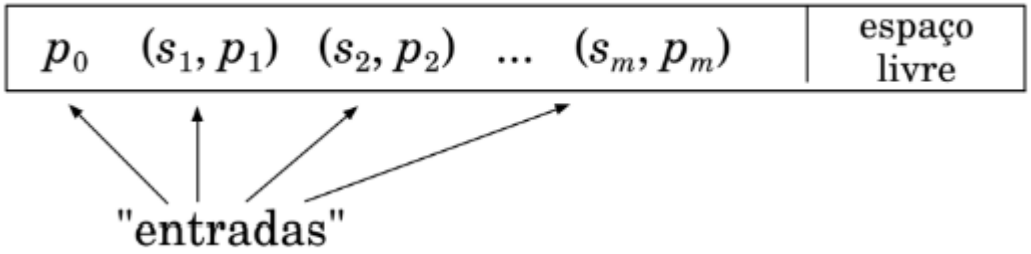
- Estrutura de um página:



  - Toda chave ***s*** pertencente a página filha apontada por p_0 satisfaz $s < s_1$
  - Toda chave ***s*** pertencente a página filha apontada por p_m satisfaz $s < s_m$

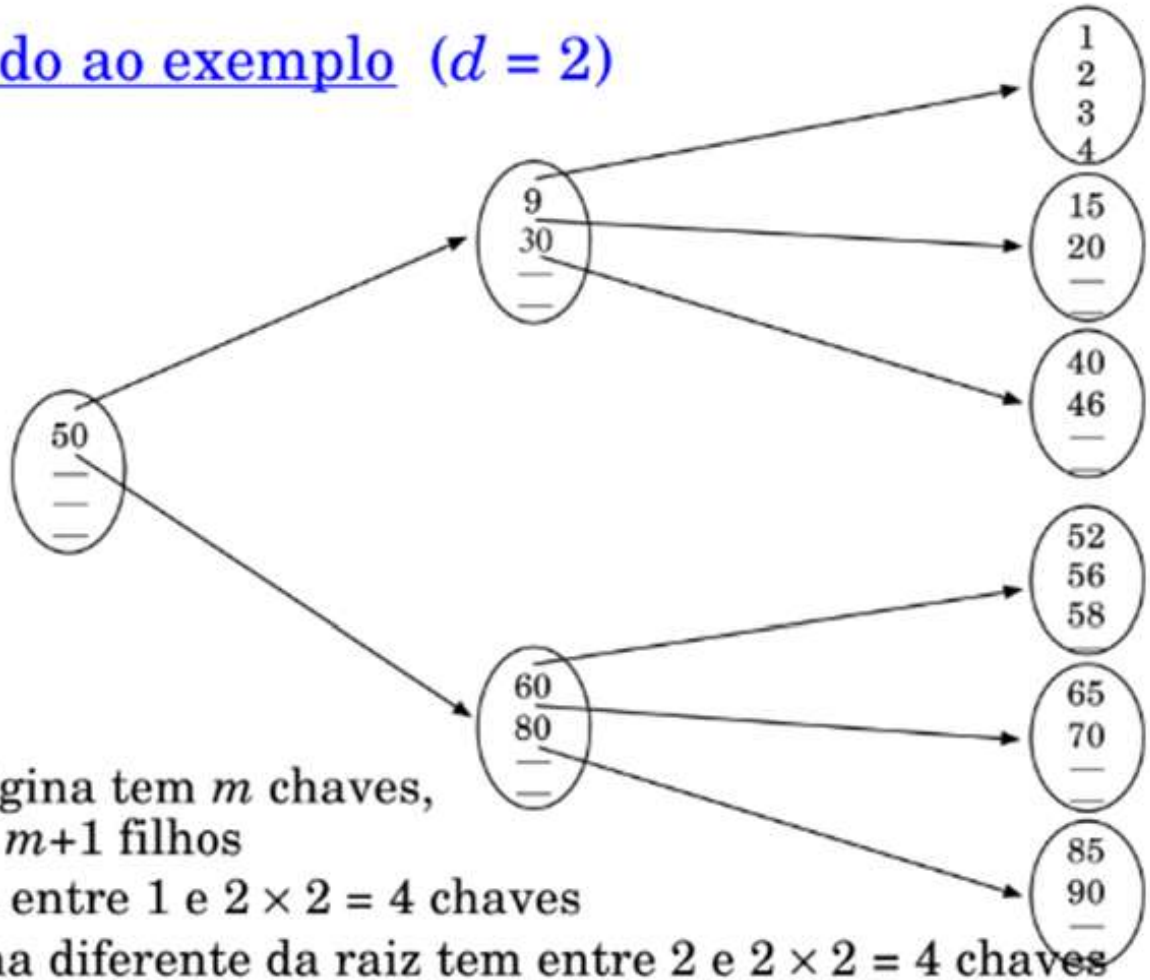
Árvore B

- Estrutura de um página:



  - Toda chave ***s*** pertencente a página filha apontada por p_k (1 ≤ k ≤ m-1) satisfaz $s_k < s < s_{k+1}$

Voltando ao exemplo ($d = 2$)



Observe:

a) se uma página tem m chaves, então tem $m+1$ filhos

b) a raiz tem entre 1 e $2 \times 2 = 4$ chaves

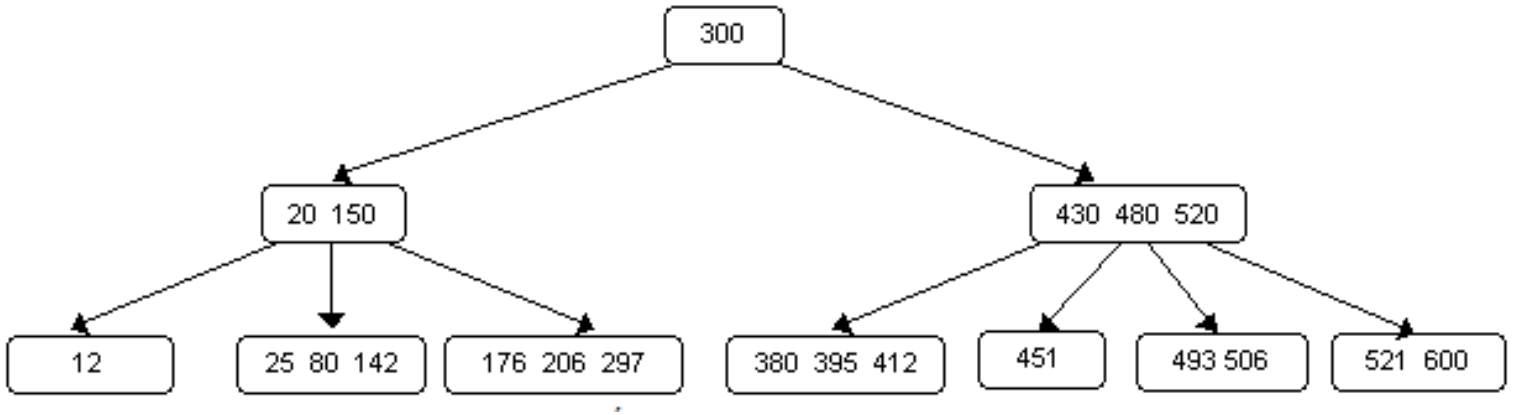
c) cada página diferente da raiz tem entre 2 e $2 \times 2 = 4$ chaves

d) em cada página as chaves estão ordenadas

e) de cada página com m chaves partem $m+1$ ponteiros (as folhas têm ponteiros nulos)

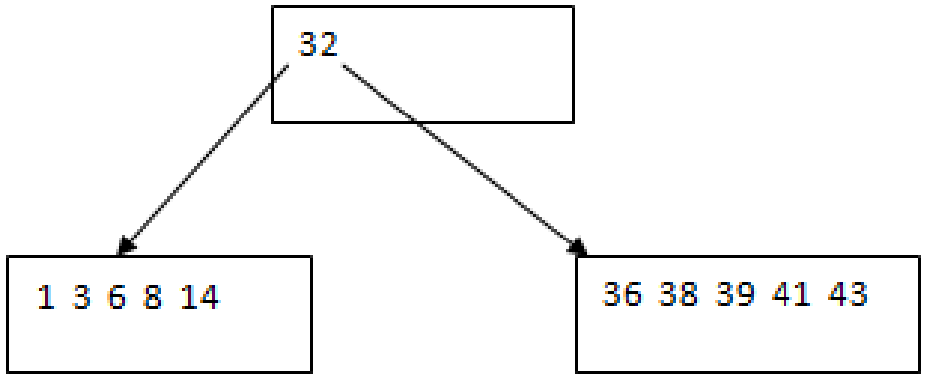
Observe a estrutura das entradas, de acordo com a tela anterior.

Árvore B


300
20 150
430 480 520
12
25 80 142
176 206 297
380 395 412
451
493 506
521 600

Exemplo

- Árvore B de ordem 3:
  - Chaves: 1, 3, 6, 8, 14, 32, 36, 39, 41, 43
  - N. mínimo chaves por páginas:
    - Por página: 3 <–> 6 (entre d e 2d chaves)
    - Na raiz: 1 <-> 6 (entre 1 e 2d chaves)
  - N. máximo chaves por páginas*: 2d = 6*



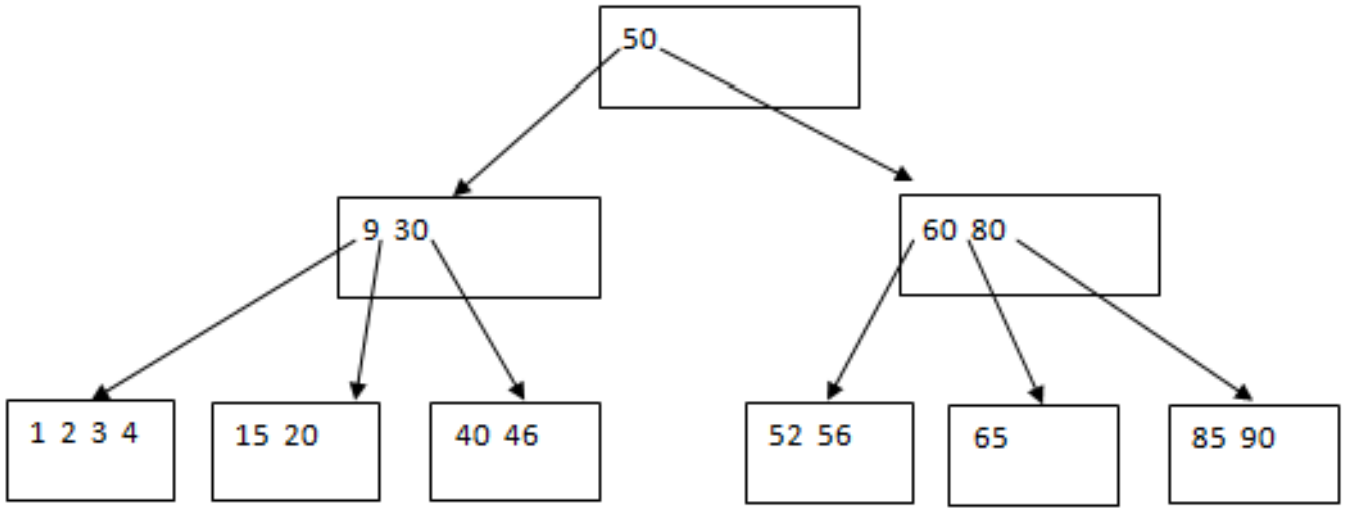
Árvore B -Busca

- Seja **s** a chave procurada:

1. Procura na lista ordenada de chaves da página raiz.
2. Se s for encontrada, interrompa.
3. Senão: busca na página filha seguindo o ponteiro adequado.
   - Se $s < s_1$, usar p_0
   - Se $s_i < s < s_{i+1}$, usar p_i $(1 \leq i \leq m\text{-}1)$
   - Se $s > s_m$, usar p_m
4. Repete a busca até que:
   - ou s seja encontrada
   - ou atinja-se um ponteiro nulo

Árvore B -Busca

- Buscar as chaves s = 56 e s =25



Busca em árvore B

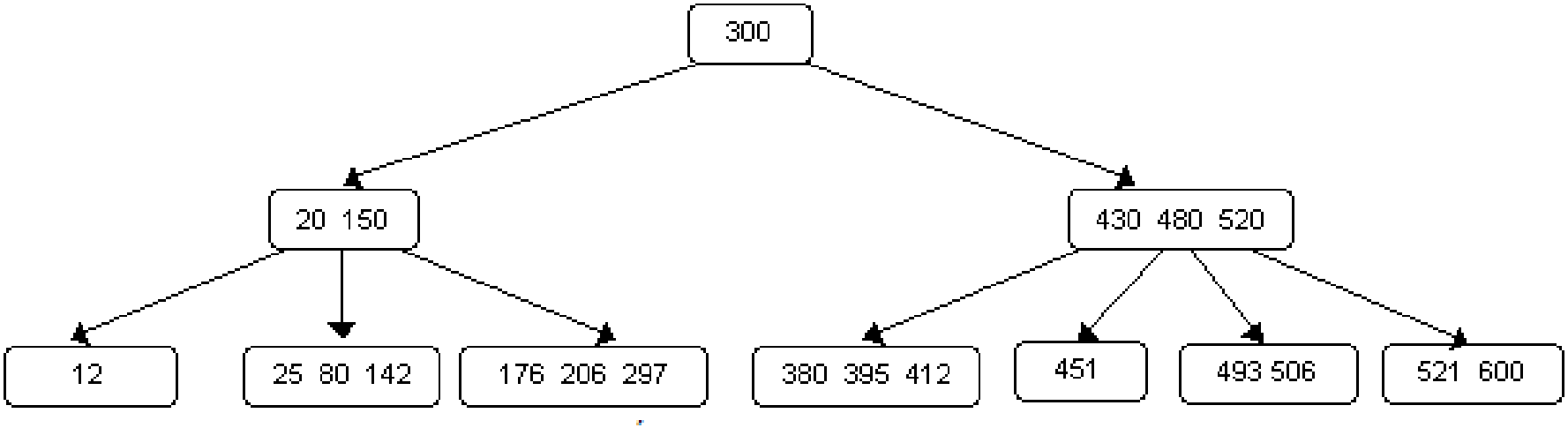
- Parecida com a busca em ABB, exceto o fato de que se deve decidir entre vários caminhos.
- Como as chaves estão ordenadas, basta realizar uma busca binária nos elementos de cada nó.

Busca em árvore B

- Se a chave não for encontrada no nó em questão, continua-se a busca nos filhos deste nó, realizando-se novamente a busca binária.
- Caso o nó não esteja contido na árvore a busca terminará ao encontrar um ponteiro igual a ***NULL.***

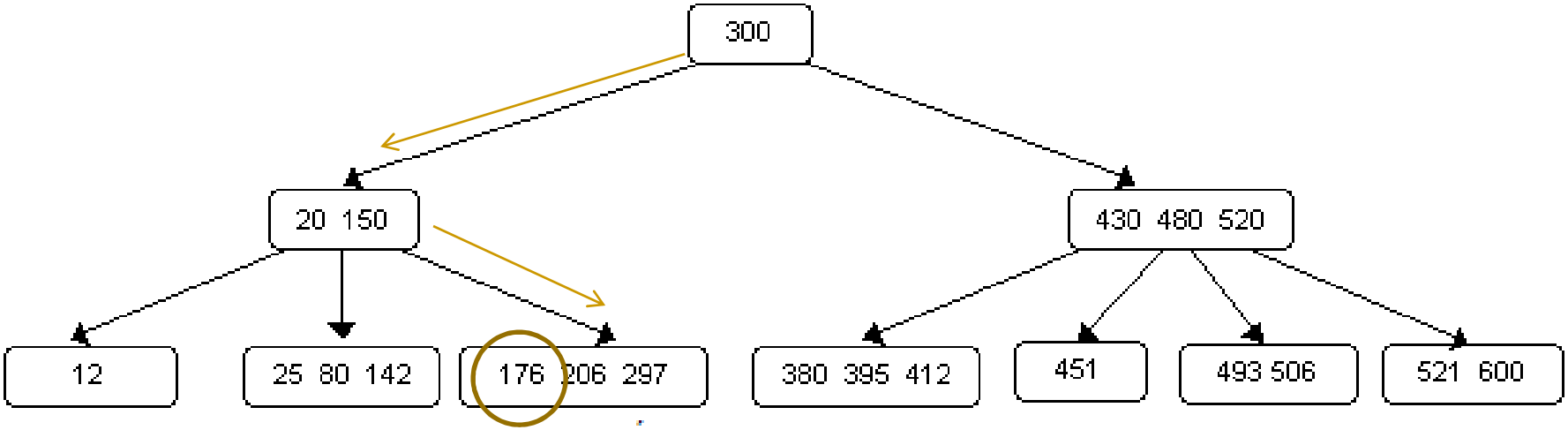
Árvore B

- Buscar a chave 176 :



Árvore B

- Buscar a chave 176 :



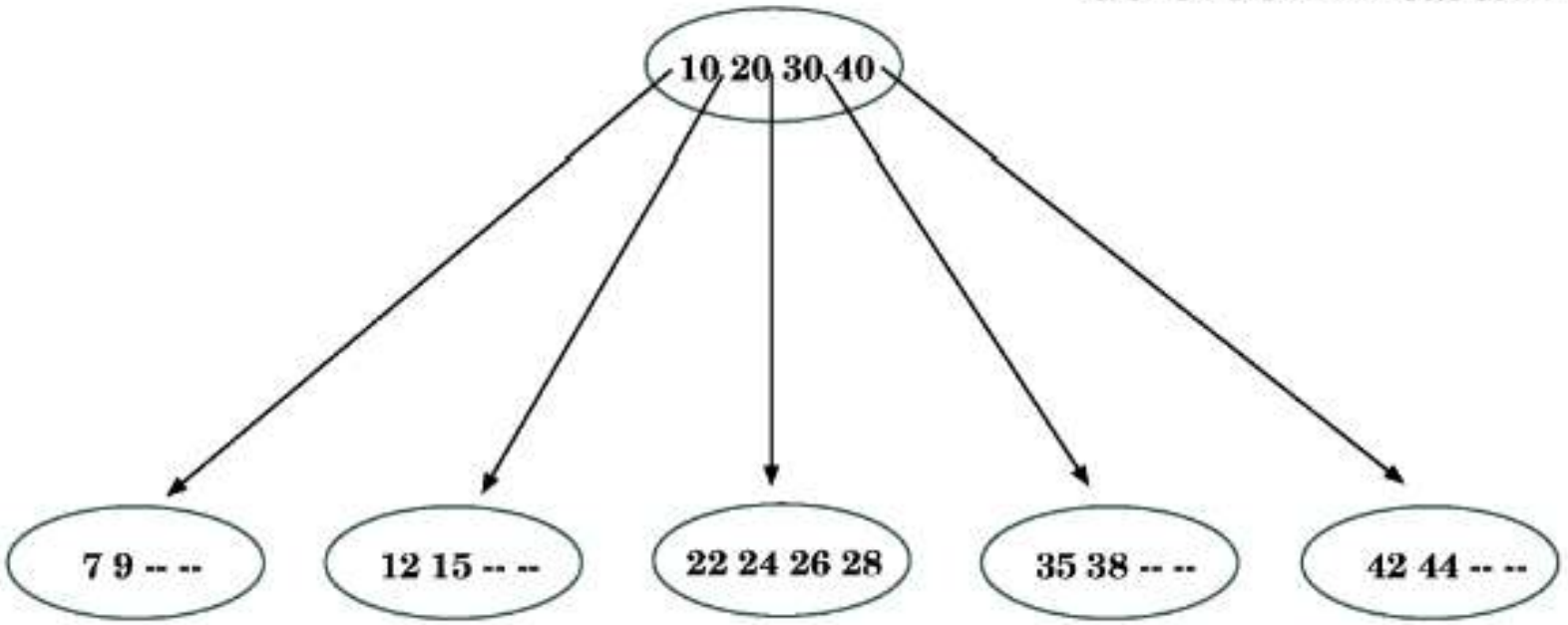
Árvore B - Inserção

- Seja **s** a chave a ser inserida:

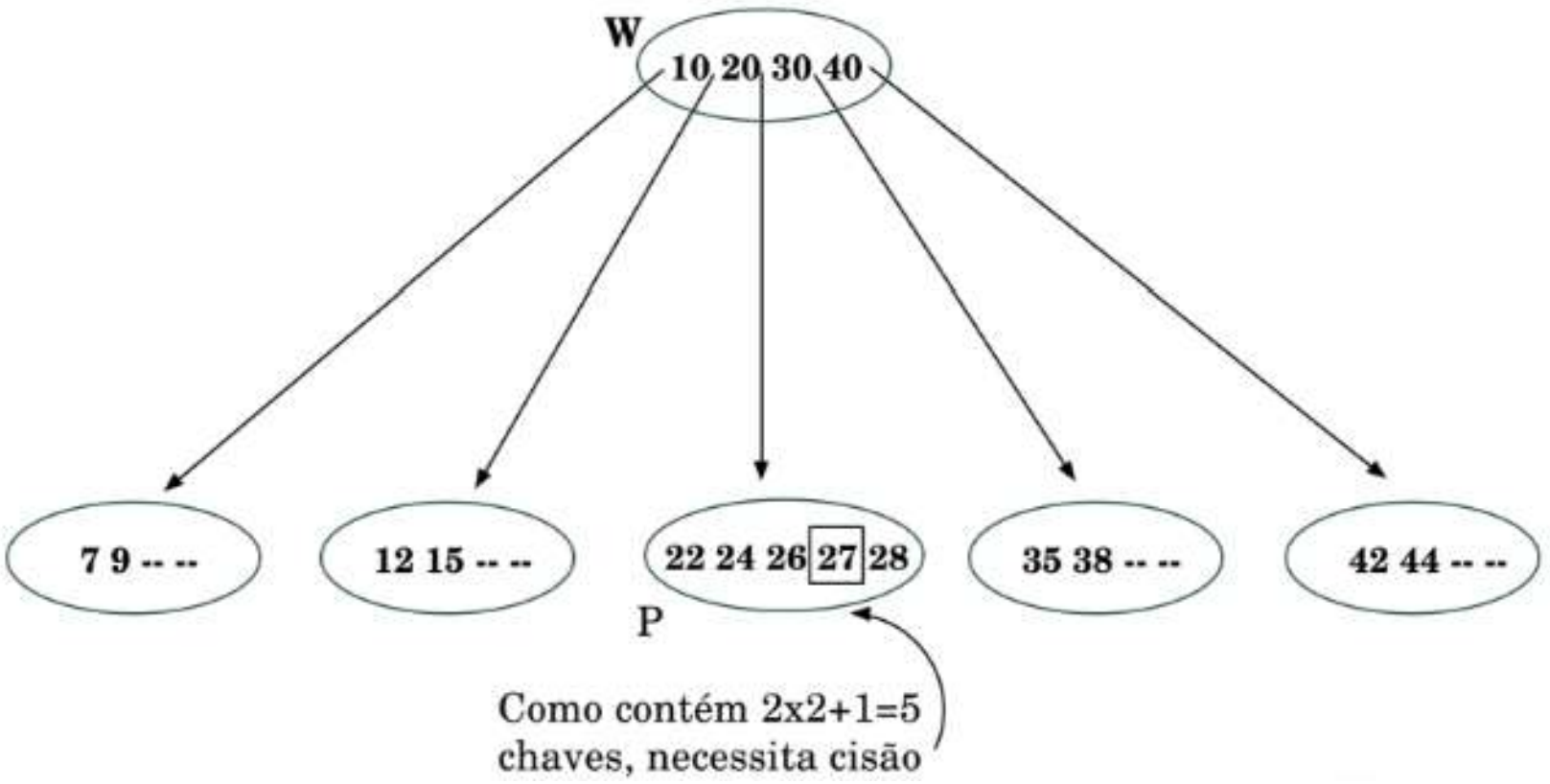
1. Procura **s** na árvore B:
    1. Se achou, então interrompa (inválido)
    2. Senão, insere **s** na página folha onde a busca se encerrou (mantendo a ordenação da lista de chaves na página).
    3. Se a página folha contém + de **2d** chaves:
        1. É efetuada a cisão, pois o limite máximo de chaves por folha é **2d.**

Árvore B - Inserção

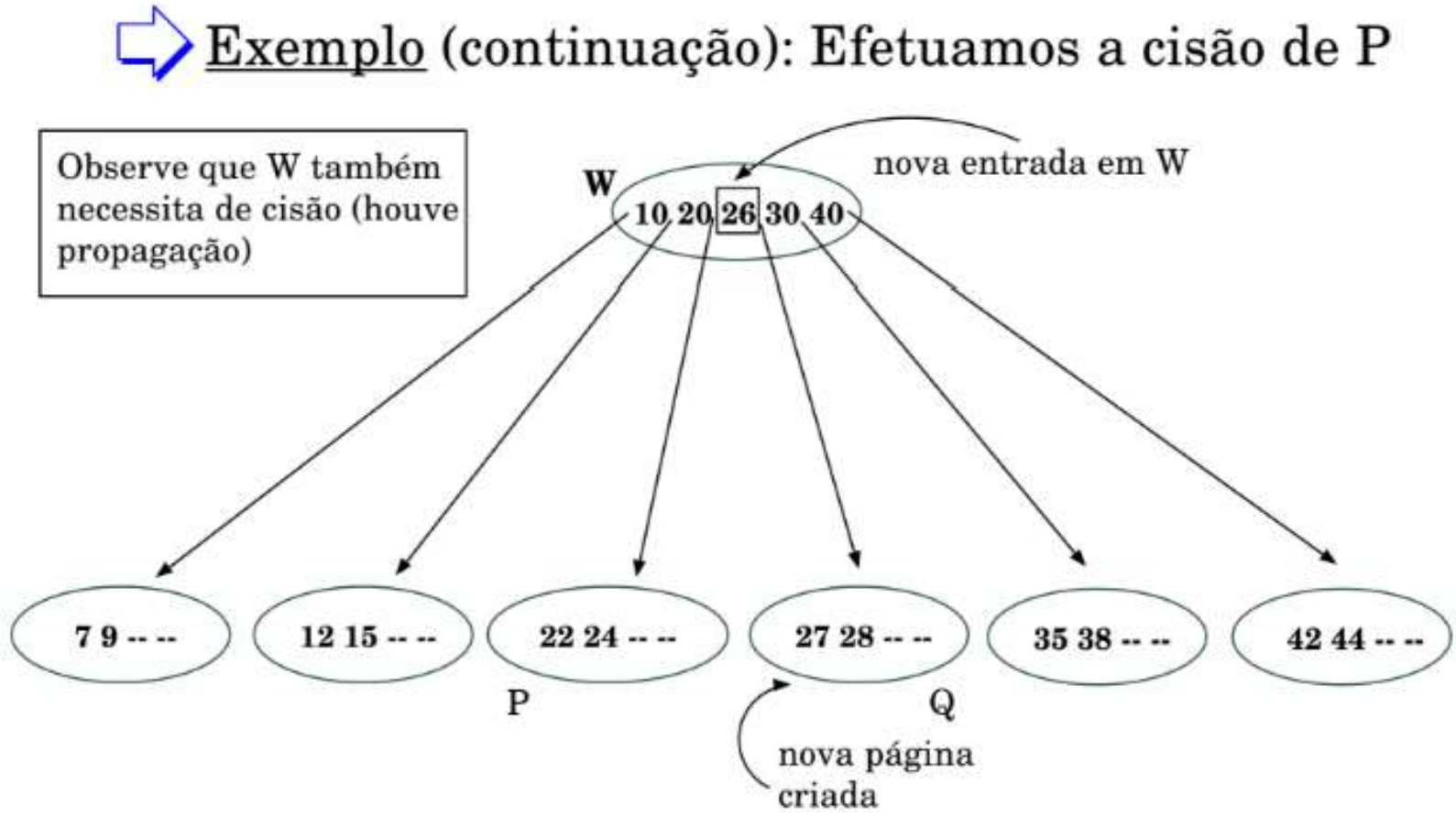
Exemplo: inserir a chave $s = 27$ na árvore B de ordem 2 abaixo



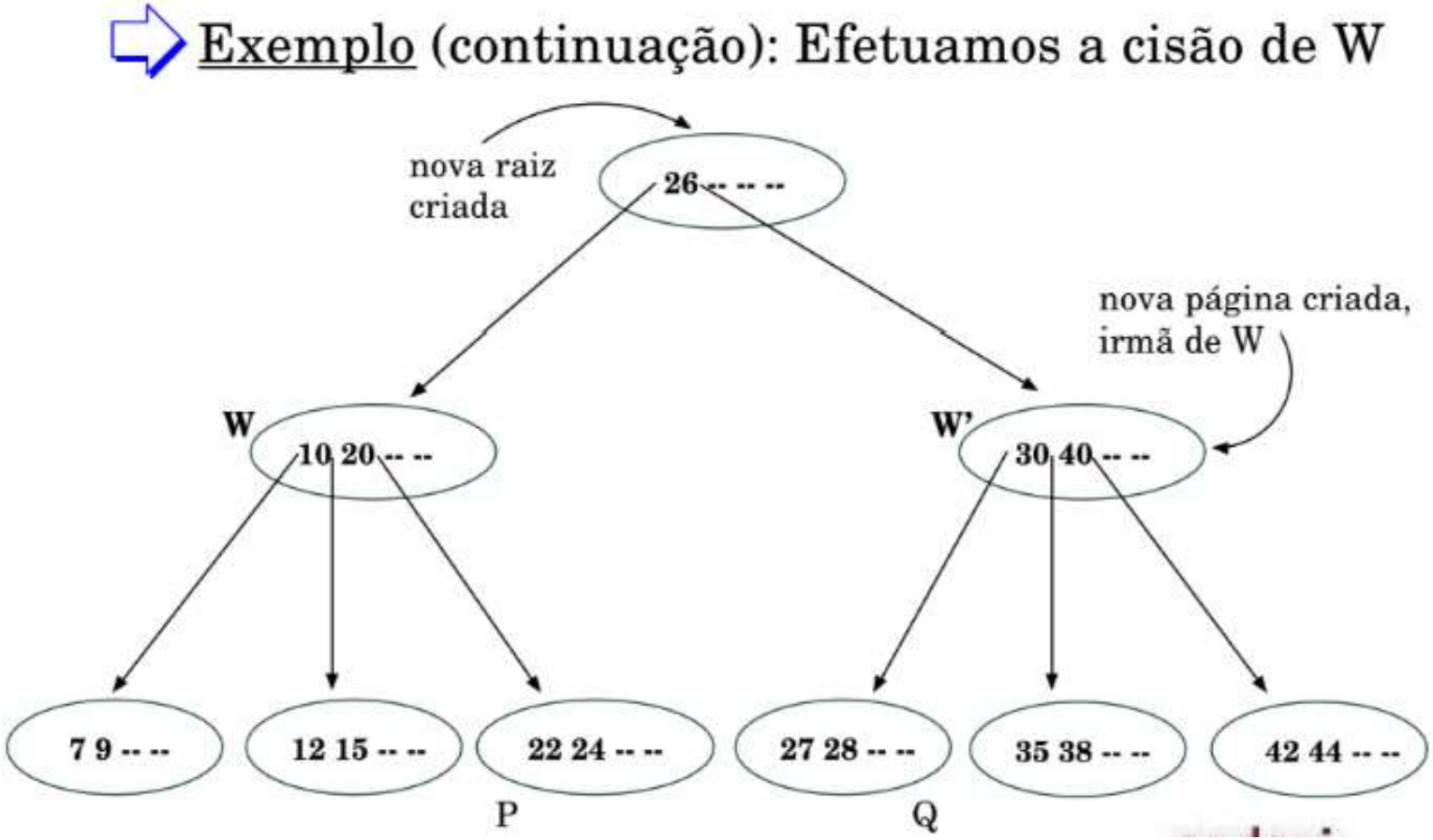
Árvore B - Inserção



Árvore B - Inserção



Árvore B - Inserção



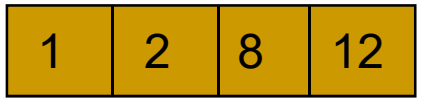
Árvores B

- **Definição:** Uma *árvore B* possui as seguintes propriedades:
  - Todo o nó ***X*** possui os seguintes campos:
    - ***n***, o número de chaves armazenadas em ***X***;
    - as ***n*** chaves $k_1, k_2 ... k_n$ são armazenadas em ordem crescente;
    - ***folha***, que indica se ***X*** é uma folha ou um nó interno.

```
struct no_arvoreB {
    int num_chaves;
    char chaves[2*t-1];
    arvoreB *filhos[2*t];
    bool folha;
};
```

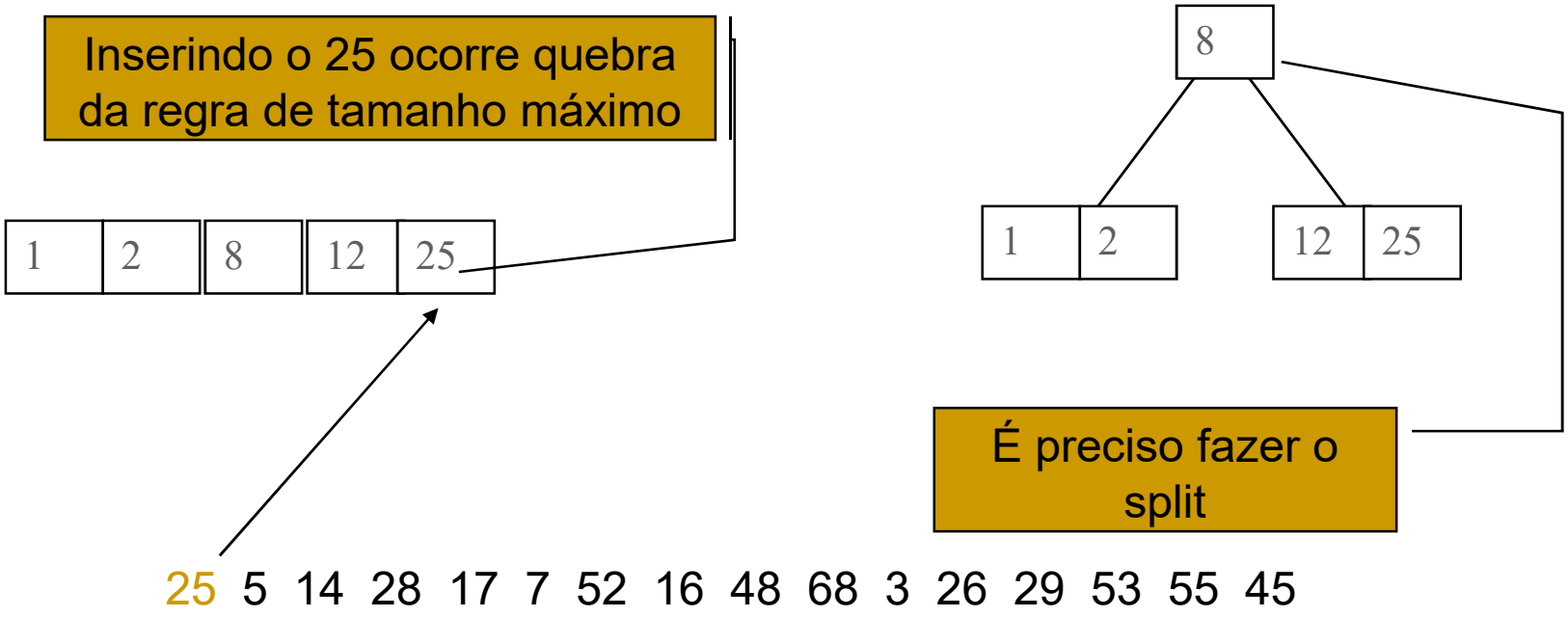
Árvore B

- Suponha que iniciemos com uma árvore B vazia e as chaves devem ser inseridas na seguinte ordem:1 12 8 2 25 6 14 28 17 7 52 16 48 68 3 26 29 53 55 45
- Queremos construir uma árvore B de ordem 2
- Os 4 primeiros elementos vão para a raíz:



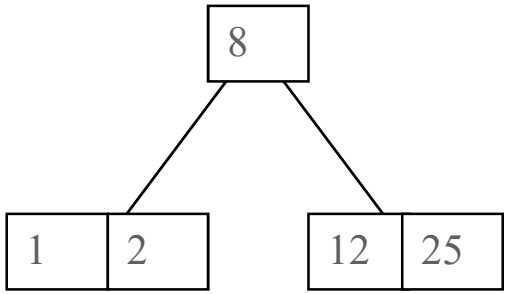
- O quinto elemento extrapola o tamanho do nó
- Assim, quando inserimos o 25 devemos dividir o nó em duas partes e colocar o elemento do meio como nova raiz

Árvore B

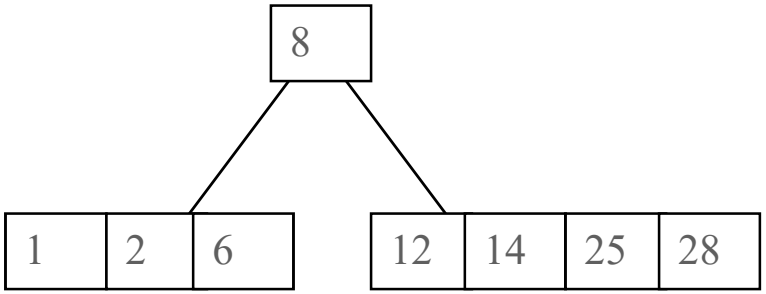


25 5 14 28 17 7 52 16 48 68 3 26 29 53 55 45

Árvore B



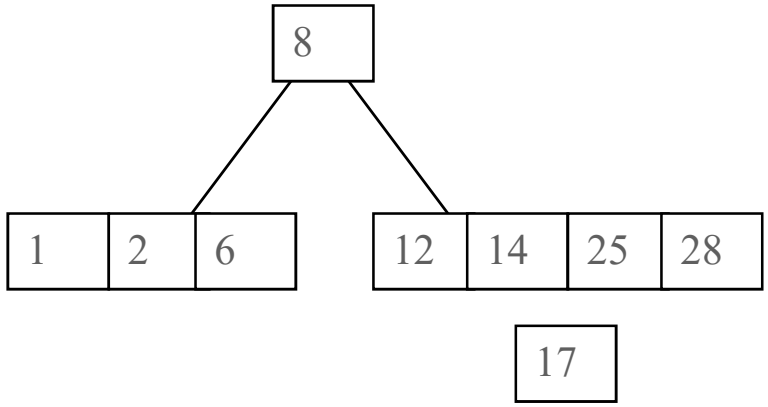
Em seguida colocamos 6, 14 e 28 :



6 14 28 17 7 52 16 48 68 3 26 29 53 55 45

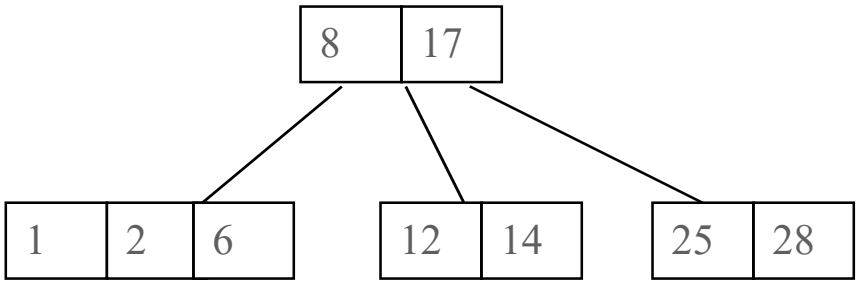
Árvore B

Adicionando 17 à árvore teremos outro split…



17 7 52 16 48 68 3 26 29 53 55 45

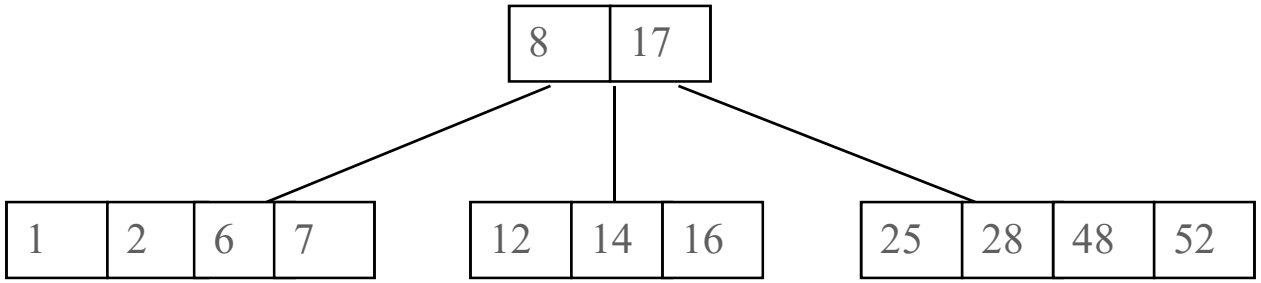
Árvore B



17 7 52 16 48 68 3 26 29 53 55 45

Árvore B

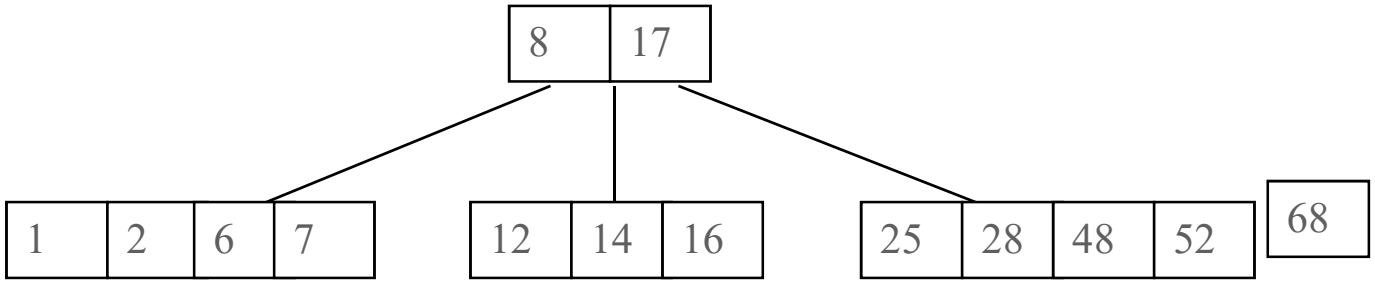
Continuando com 7, 52, 16 e 48



7 52 16 48 68 3 26 29 53 55 45

Árvore B

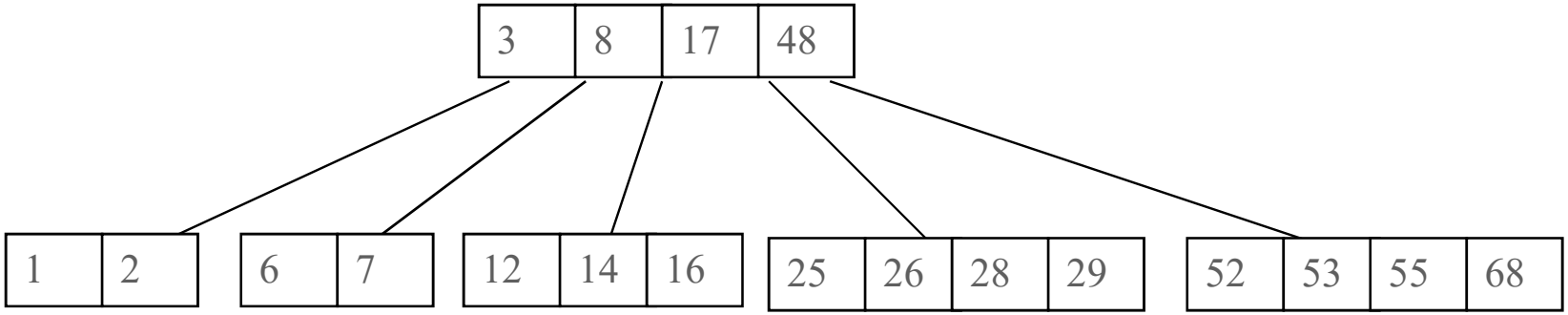
E agora, inserindo o 68...



68 3 26 29 53 55 45

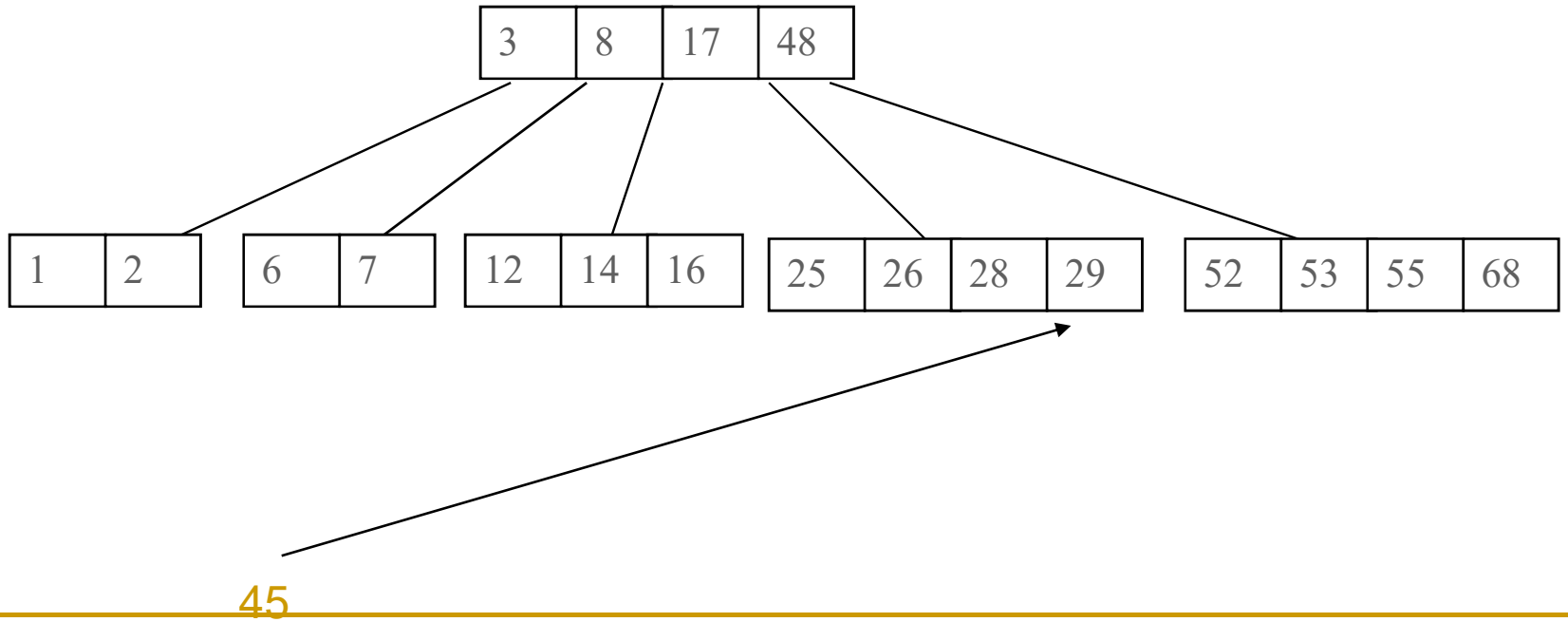
Árvore B

Adicionando 68 à árvore causa um "split" na folha mais à direita, fazendo com que o 48 suba à raiz. Quando inserimos o 3 o "split" é na folha mais à esquerda (o 3 sobe); 26, 29, 53, 55 vão para as folhas:



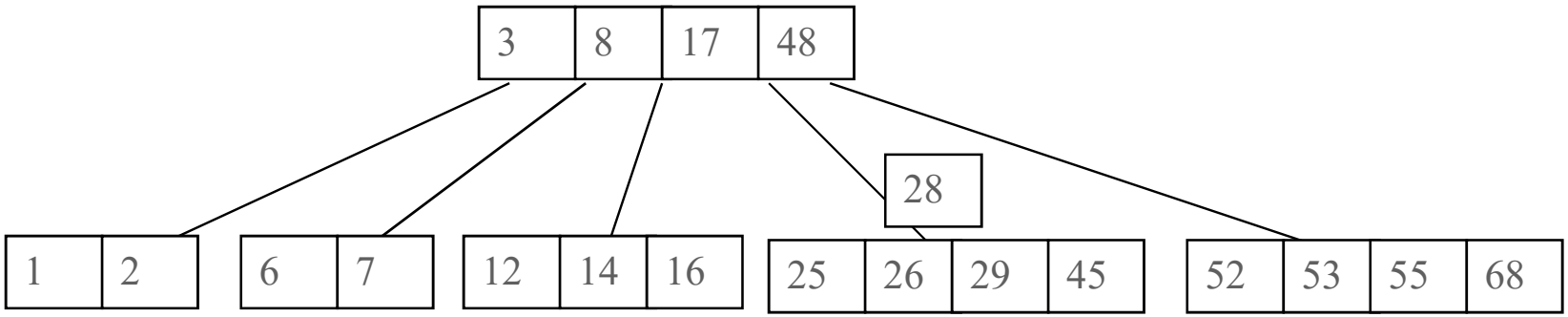
Árvore B

Por fim o 45:



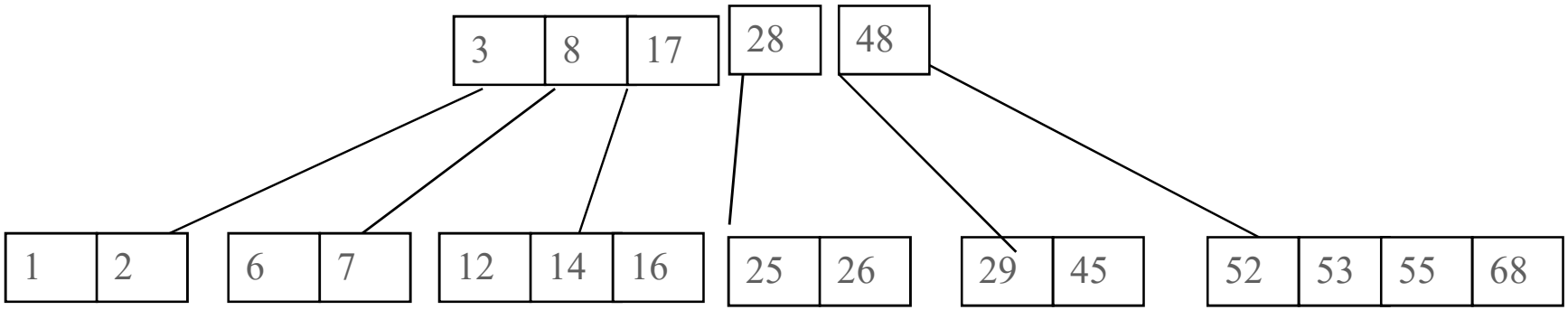
Árvore B

Por fim, quando inserimos o 45, isso forçará com que o 28 suba para a raiz... Mas a raiz também está cheia !



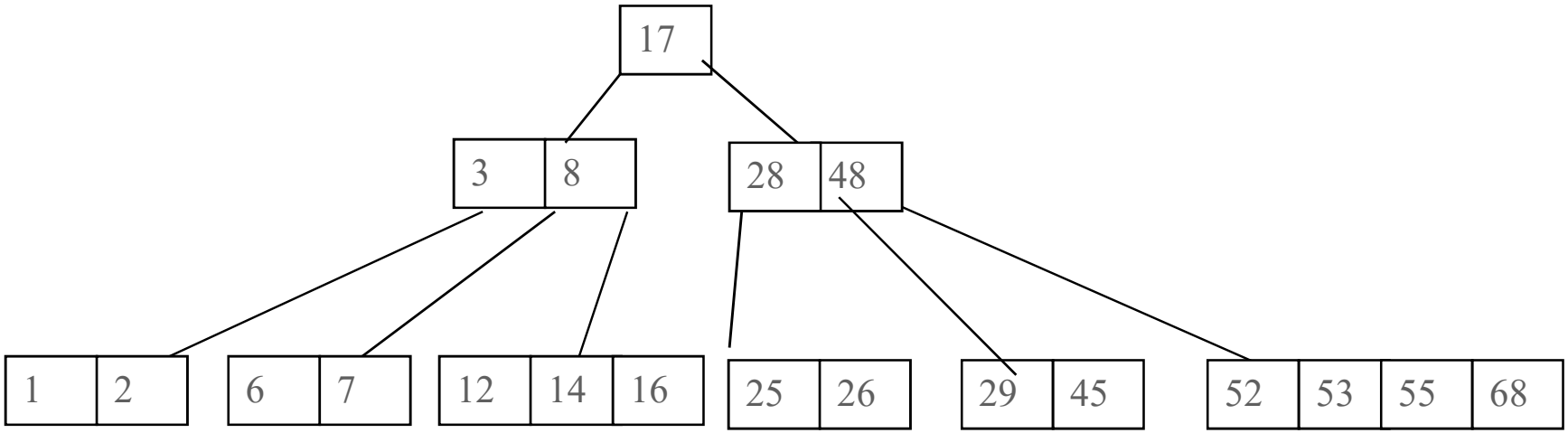
Árvore B

Por fim, quando inserimos o 45, isso forçará com que o 28 suba para a raiz... Mas a raiz também está cheia !

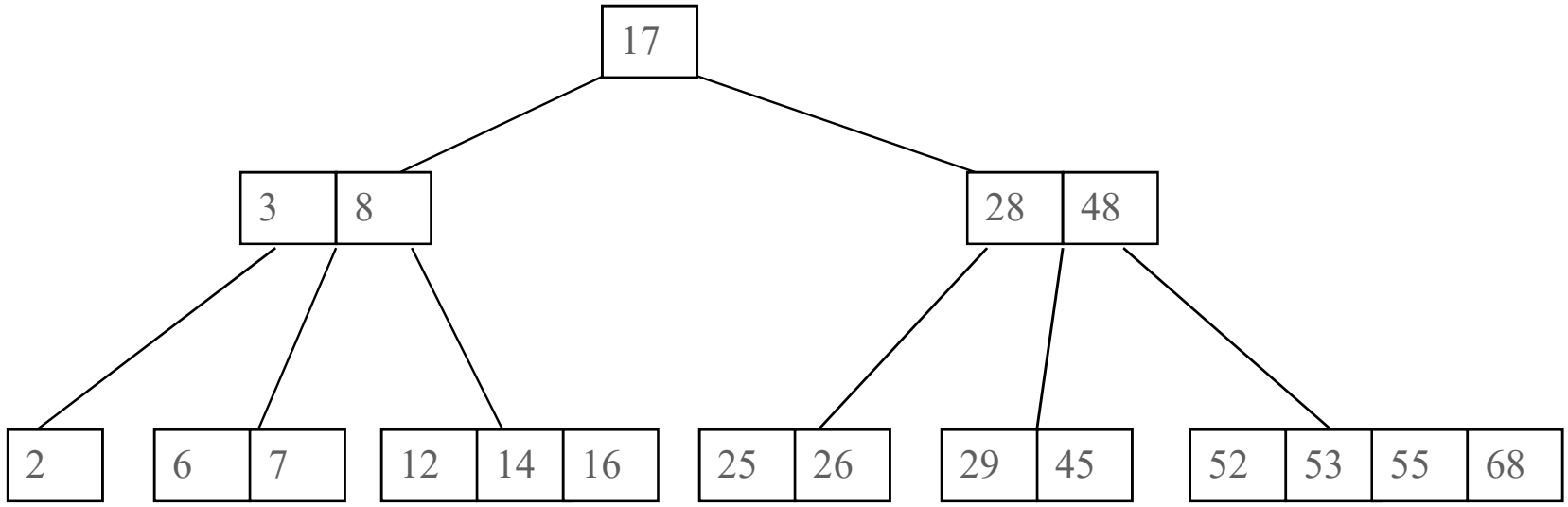


Árvore B

O 17 tem que subir para se tornar a nova raiz... lembrem-se que a raiz pode ter um único elemento.



Árvore B



Árvore B - remoção

- Seja ***s*** a chave a remover e ***B*** a árvore B.
- Efetua-se a busca (**s**, ***B***)
- Se não encontrar busca (**s**, ***B***):
  - A chave **s** não pertence a estrutura ***B***.

- Caso contrário:
  - Se s estiver numa página-folha → remova
  - Se s estiver numa página não-folha → coloque no seu lugar a chave **x** imediatamente maior que **s** (x sempre estará numa folha)

Árvore B - remoção

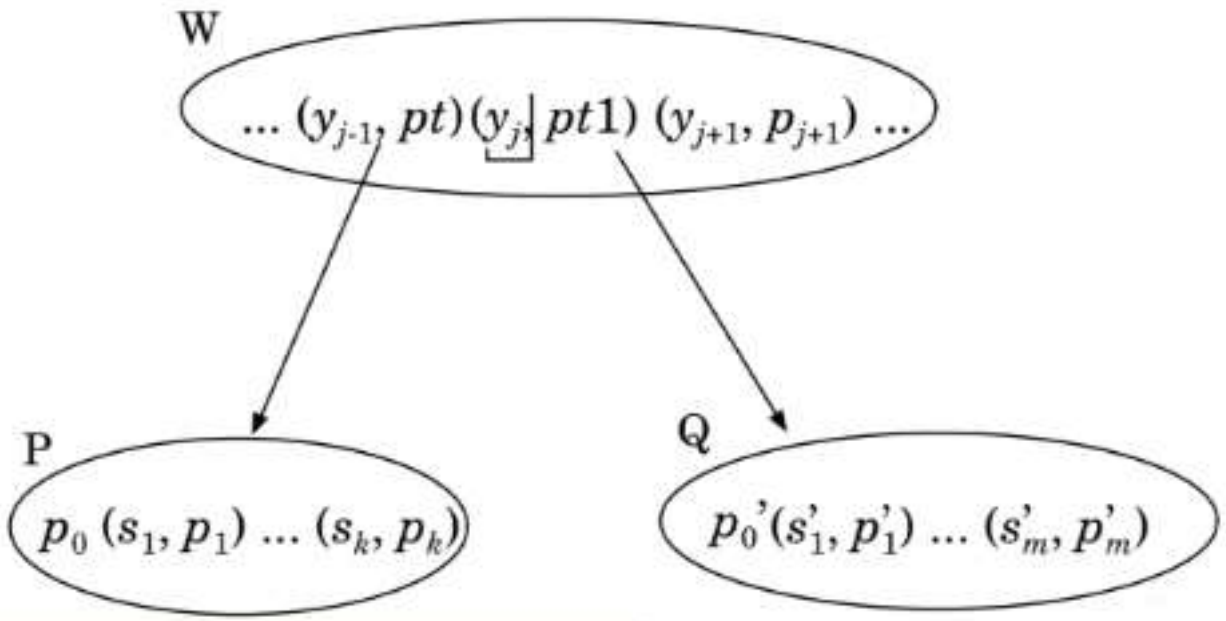
- Se a página-folha onde **foi efetuada a remoção** ficou com menos de d chaves, é preciso fazer:

- **Concatenação** de chaves
- **Redistribuição** de chaves

Árvore B – remoção - concatenação

- Concatenação:
  - Aplica-se quando a página P (com k < d chaves) possui uma irmão Q (com m chaves) tal que:

$$k + m < 2d$$

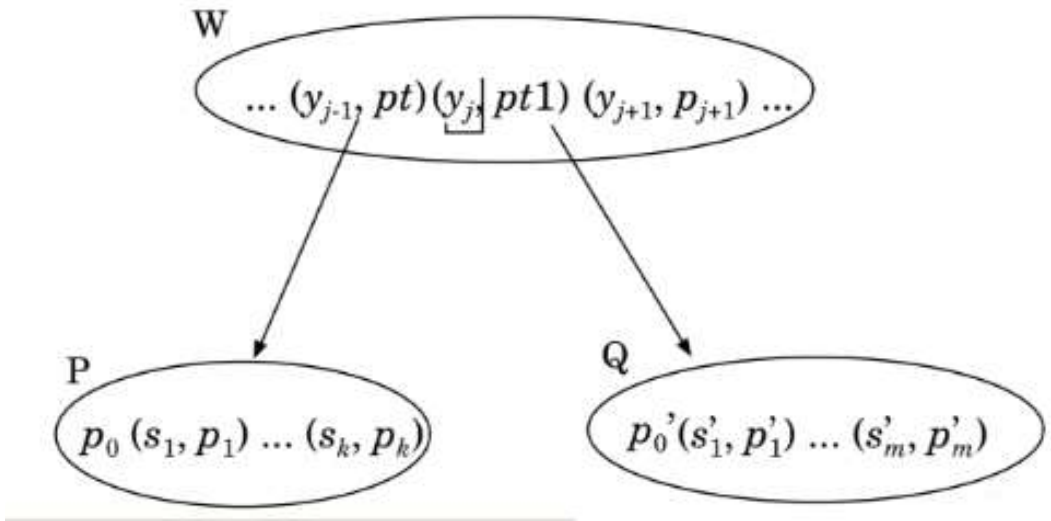


Árvore B – remoção - concatenação

- Página onde foi efetuada a remoção: P
- P está com menos que **d** chaves (k < d)
  - Limite inferior violado
- Irmã de P é Q. Q possui m chaves.
- A soma k + m < 2d:
  - Podemos concatenar as páginas P e Q em uma única página.
  - A qtde máxima de chaves em uma página não será violada!!!

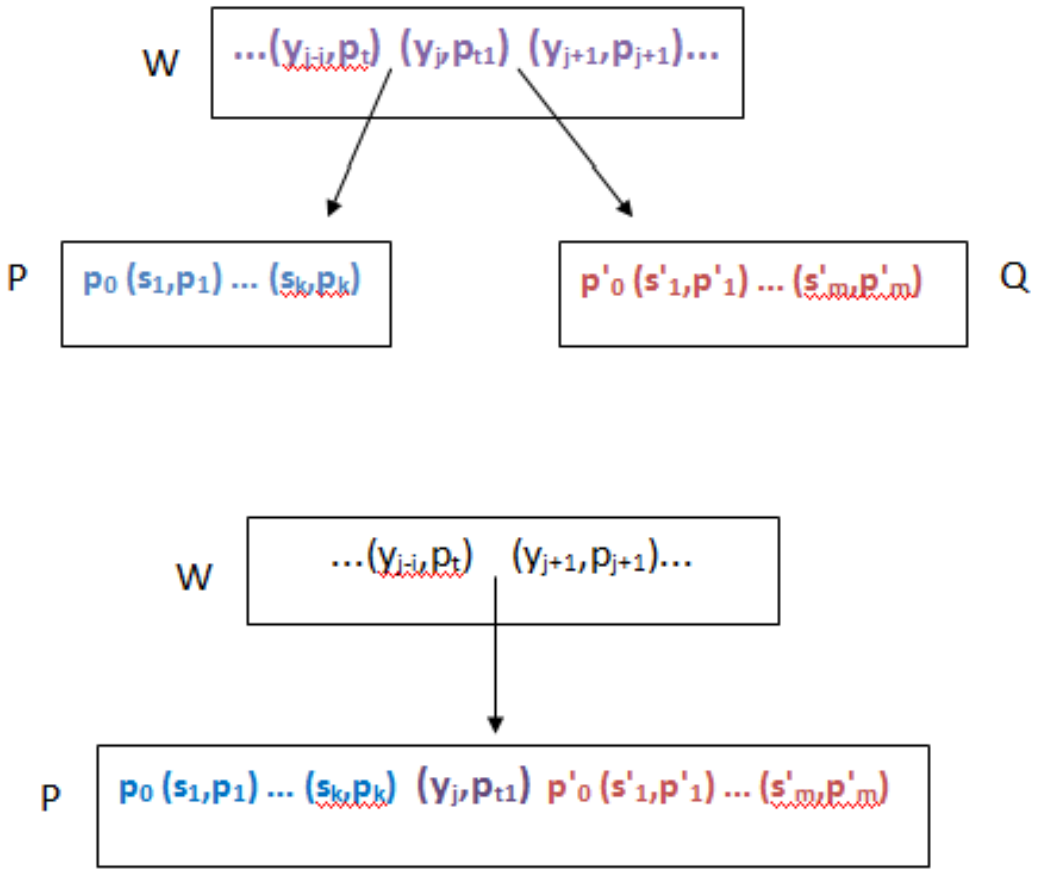
Árvore B – remoção - concatenação

- Concatenação de P com Q.
- Concatenação exige acerto na página pai de ambos as páginas (W).



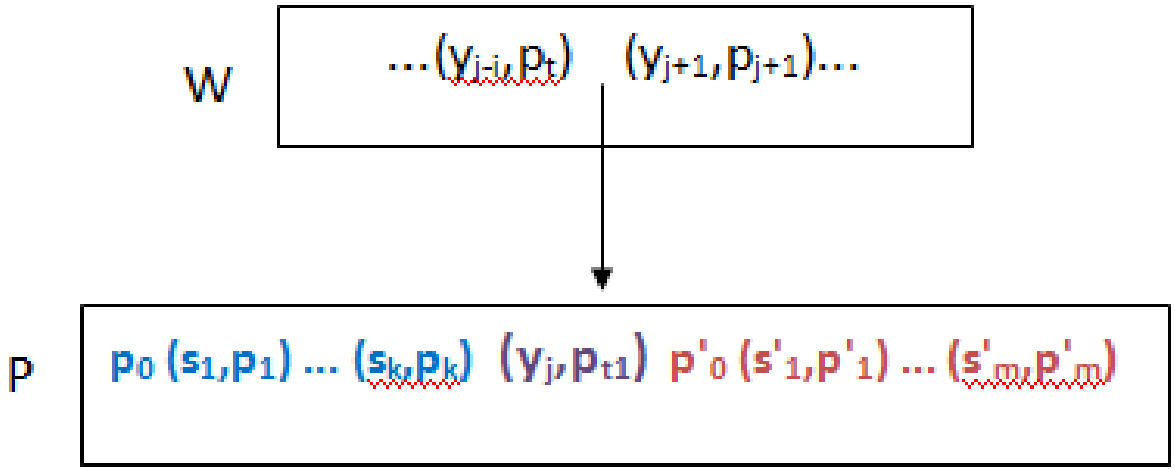
Árvore B – remoção - concatenação

- Uma entrade de W tem que migrar para página P → faz-se acerto de ponteiros na estrutura.
- W: 1 -
- P U Q: 1 entrada +
- Q deixa de existir
- Ponteiro p/ Q tb



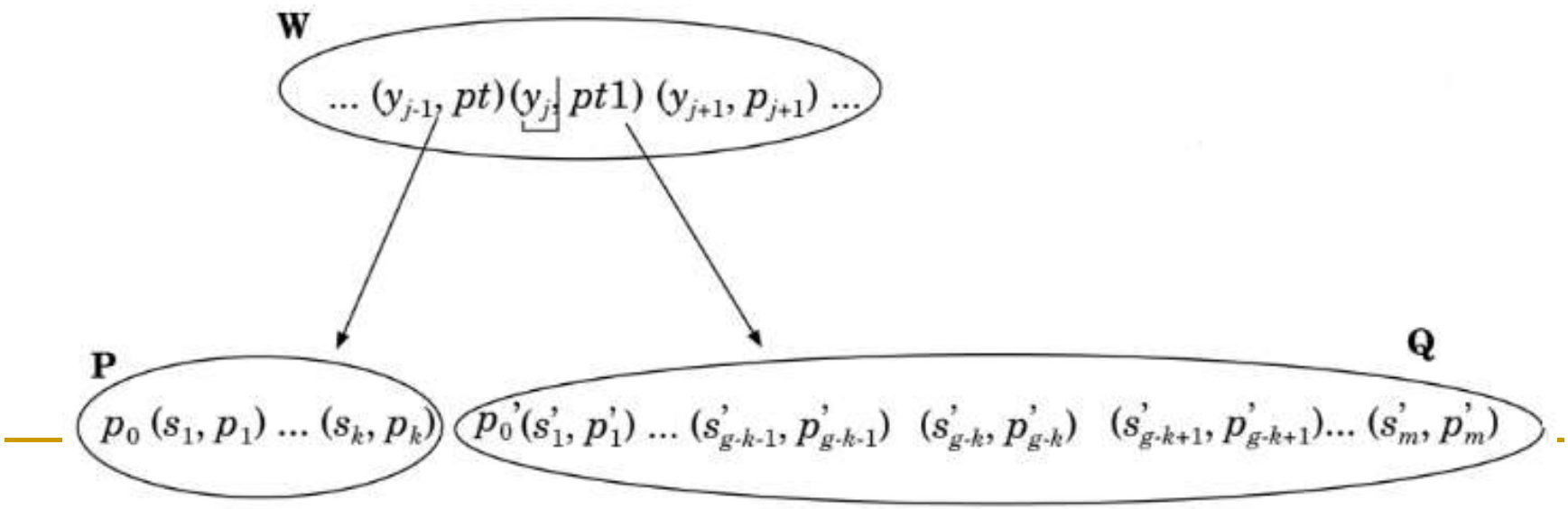
Árvore B – remoção - concatenação

- Concatenação é propagável:
  - Pode ser que W, por ter perdido uma entrada, tenha ficado com menos que 2d chaves.



Árvore B – remoção - redistribuição

- Redistribuição:
  - Aplicamos a redistribuição quando a página P (com ***k < d*** chaves) possui uma irmã Q (com ***m*** chaves) tal que ***k + m ≥ 2d***



Árvore B – remoção - redistribuição

- Página onde foi efetuada a remoção: P
- P está com menos que **d** chaves (k < d)
  - Limite inferior violado
- Irmã de P é Q. Q possui m chaves.
- A soma k + m ≥ 2d:
  - Não pode concatenar P e Q → estoura limite
  - Redistribuição → ***algumas*** chaves de Q vão migrar para P!!!
    - P e Q continuam a existir!!!

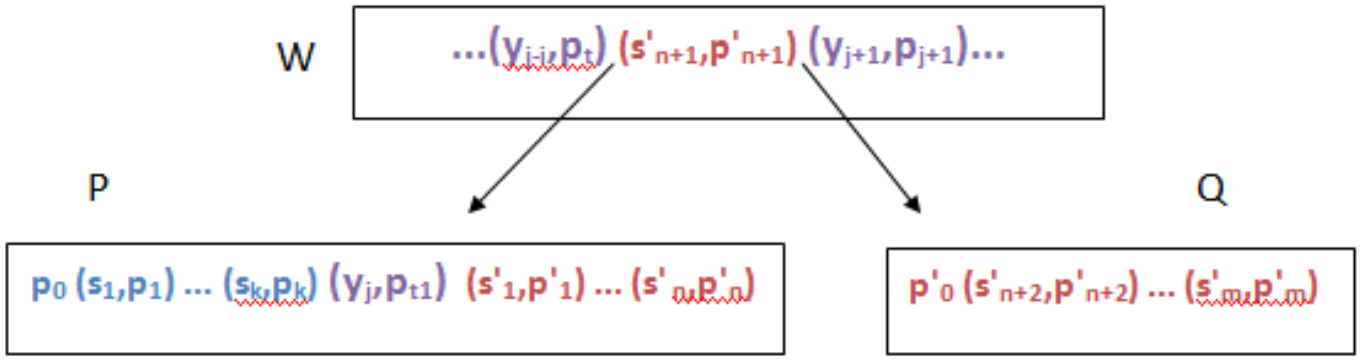
Árvore B – remoção - redistribuição

- Algumas chaves de Q vão migrar para P → precisa de acerto na página pai!!!
  - Escolhe um valor ***g*** que representa o número de chaves de cada página.
  - ***g*** é o número de chaves em cada página.

$$g = \left\lfloor \frac{k+m}{2} \right\rfloor$$

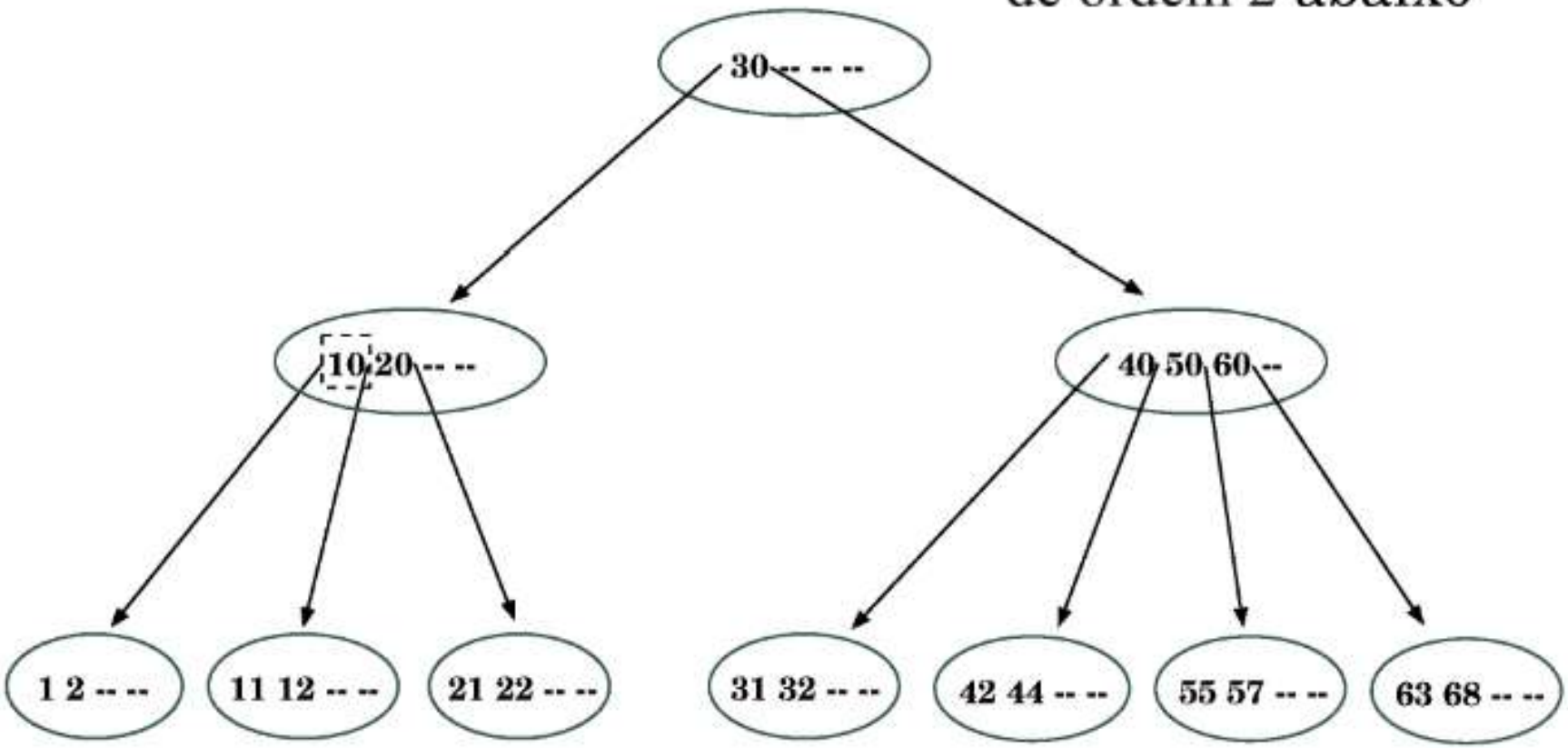
Árvore B – remoção - redistribuição

- Redistribuição – não propagável



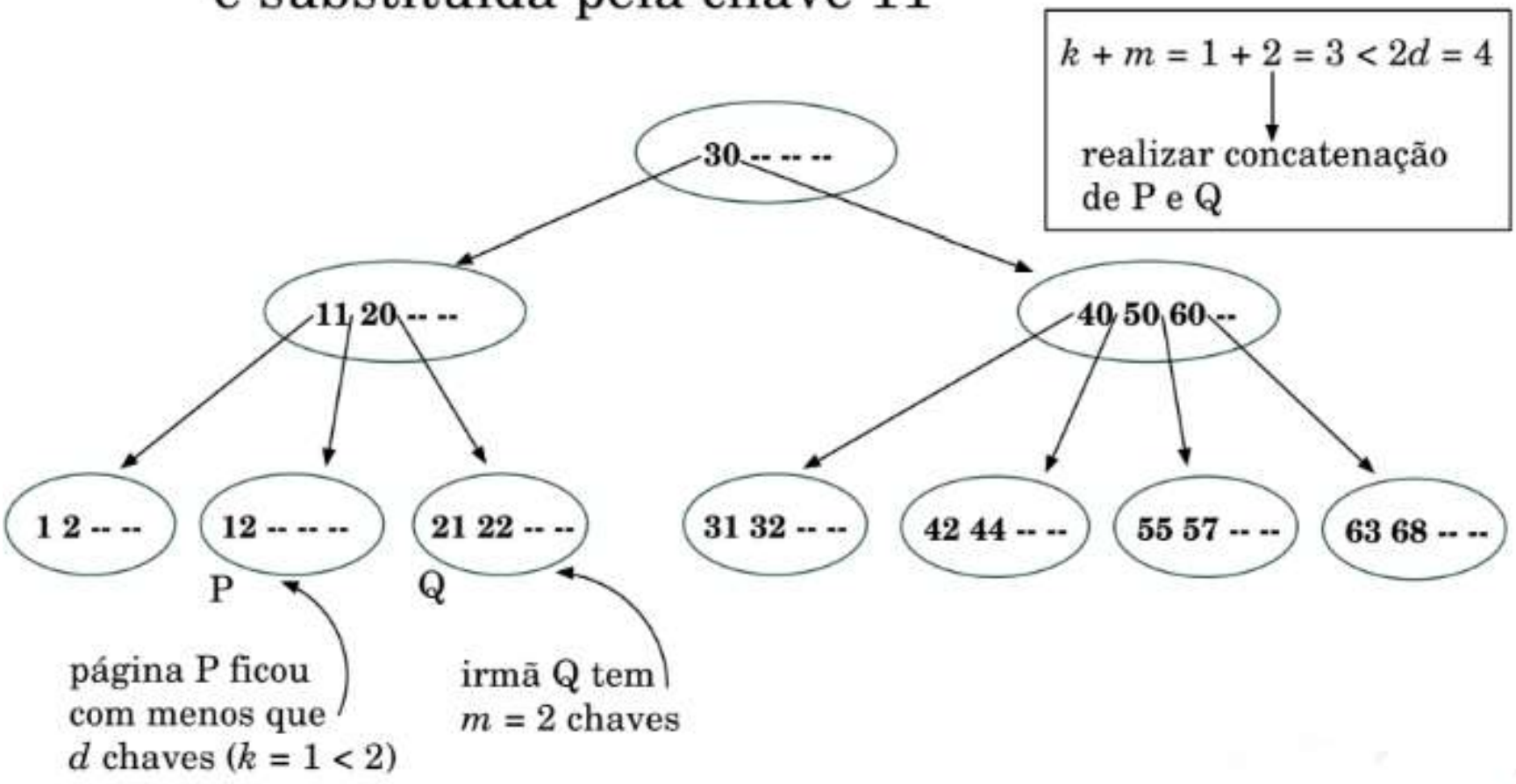
Árvore B – remoção - Exemplo

Exemplo: Remover a chave $s = 10$ da árvore B de ordem 2 abaixo

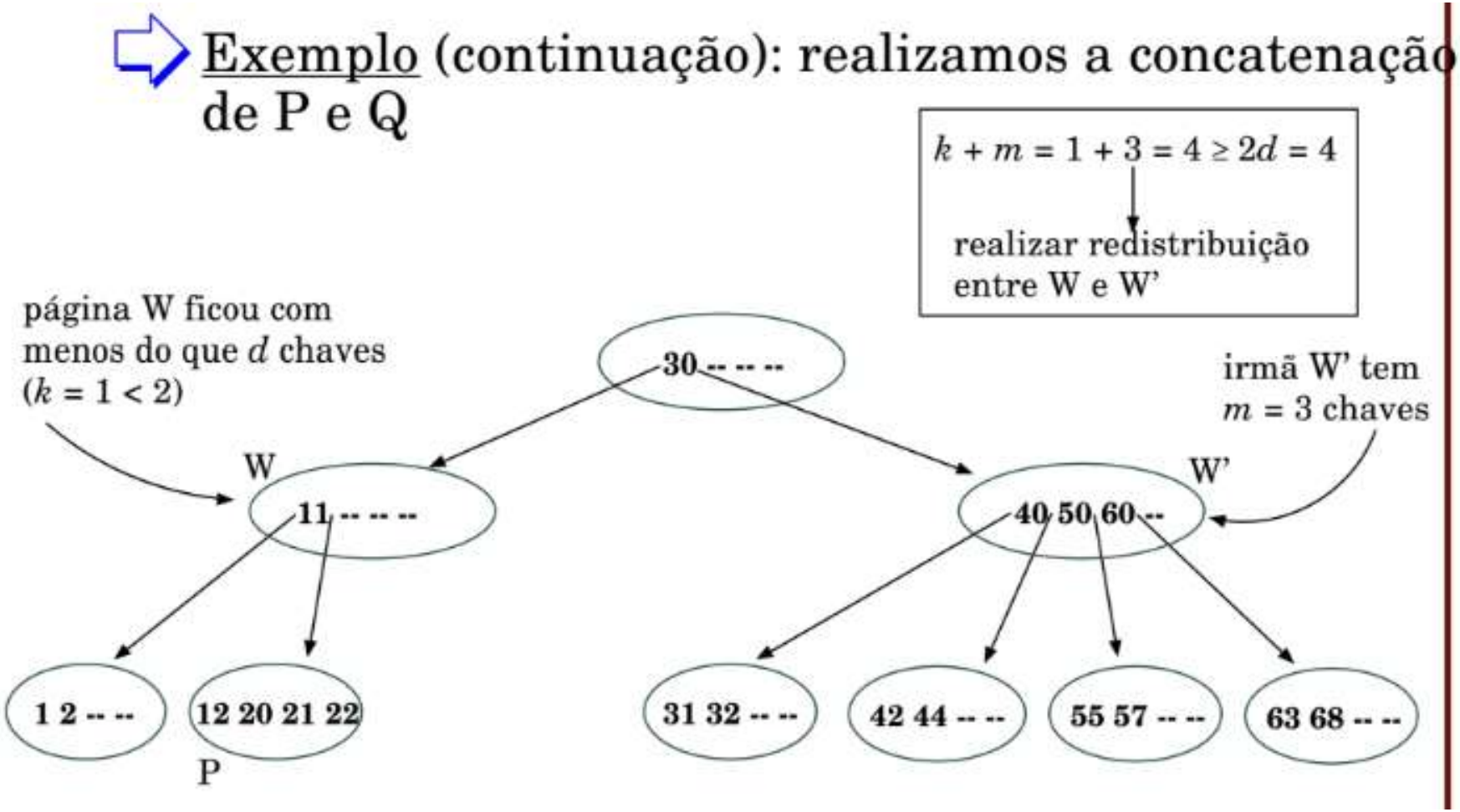


Árvore B – remoção - exemplo

Exemplo (continuação): a chave é 10 removida e substituída pela chave 11

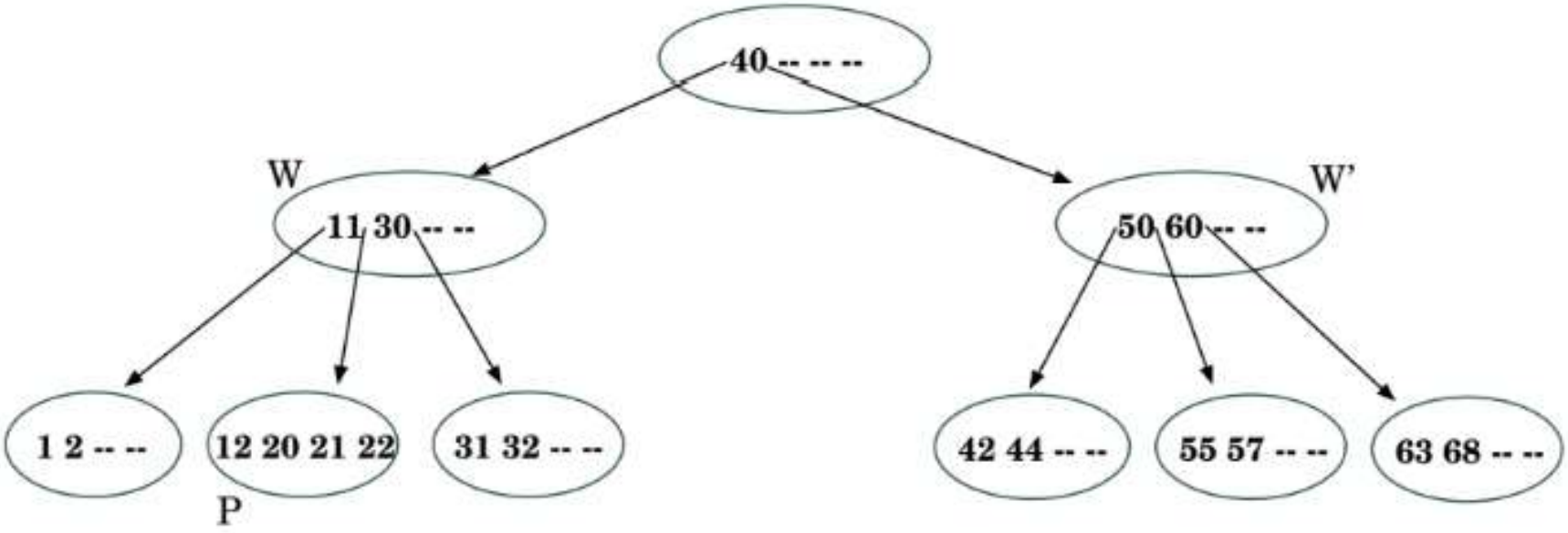


Árvore B – remoção - exemplo



Árvore B – remoção - exemplo

Exemplo (continuação): realizamos a redistribuição entre W e W'



Árvore B – remoção - exemplo

Árvore B – remoção - exemplo