ÁRVORE B

Vanessa Braganholo

Estruturas de Dados e Seus Algoritmos

CONSULTA A ARQUIVOS BINÁRIOS GRANDES

Arquivos binários grandes

- Busca sequencial é muito custosa
- Se arquivo estiver ordenado pode-se fazer busca binária, mas para arquivos grandes ainda não é eficiente o suficiente

É possível acelerar a busca usando duas técnicas:

- Acesso via cálculo do endereço do registro (*hashing*)
- Acesso via estrutura de dados auxiliar (índice)

ÍNDICE

Índice é uma estrutura de dados que serve para localizar registros no arquivo de dados

Cada entrada do índice contém

- Valor da chave
- Ponteiro para o arquivo de dados

Pode-se pensar então em dois arquivos:

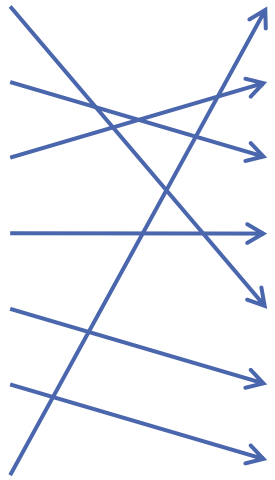
- Um de índice
- Um de dados

Isso é eficiente?

EXEMPLO DE ÍNDICE PLANO

Arquivo de Índice

	CHAVE	PONTEIRO
0	3	4
1	5	2
2	10	1
3	15	3
4	16	5
5	21	6
6	23	0



Arquivo de Dados

	COD	**NOME**
0	23	JOSE
1	10	MARIO
2	5	ANA
3	15	MARCIA
4	3	JULIO
5	16	BEATRIZ
6	21	CAMILA

ÍNDICE

Se tivermos que percorrer o arquivo de índice sequencialmente para encontrar uma determinada chave, o índice não terá muita utilidade

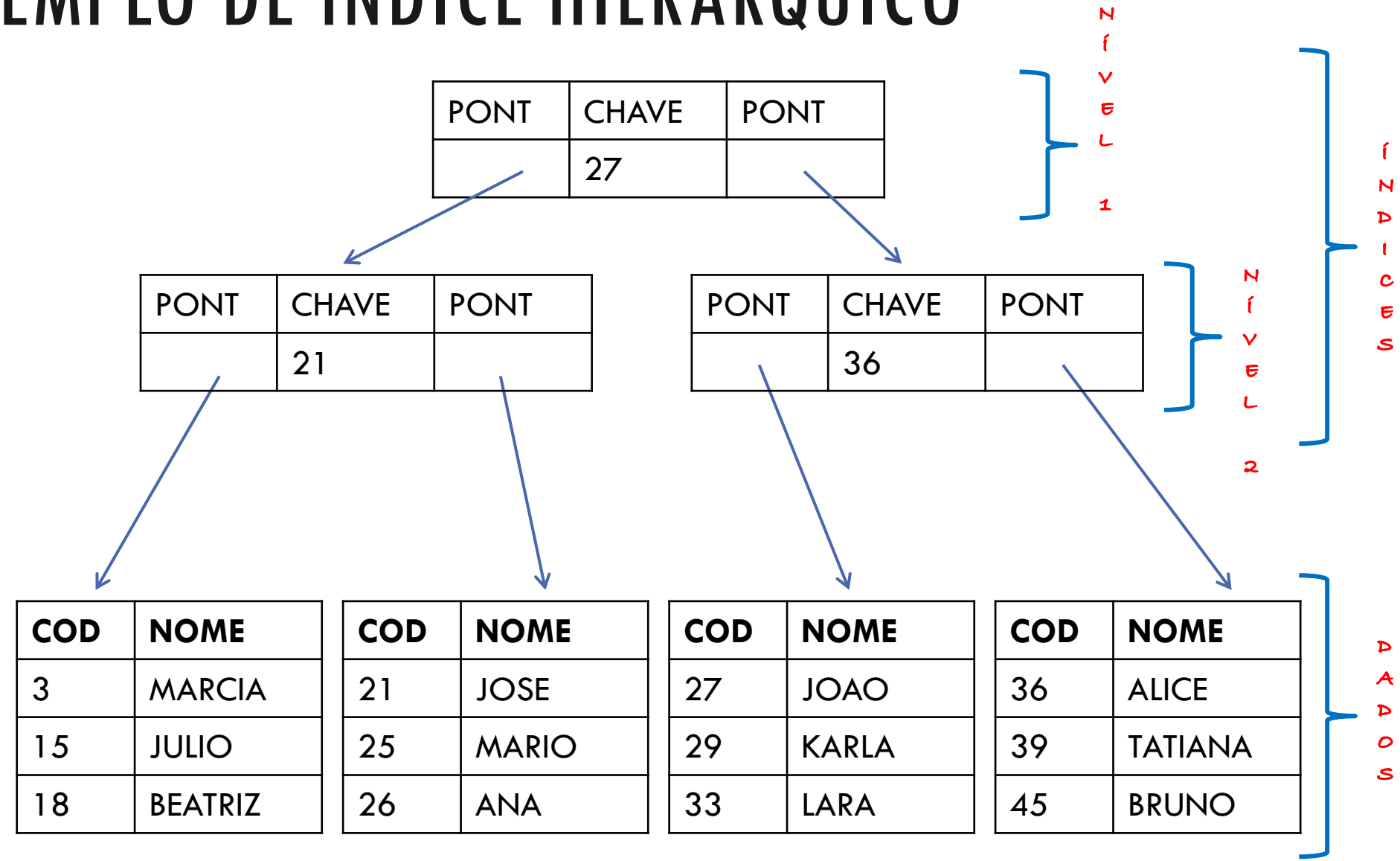
- Pode-se fazer busca um pouco mais eficiente (ex. busca binária), se o arquivo de índice estiver ordenado
- Mas mesmo assim isso não é o ideal

SOLUÇÃO: ESTRUTURAS HIERÁRQUICAS COMO ÍNDICES

Para resolver este problema:

- os índices não são estruturas sequenciais, e sim hierárquicas
- os índices não apontam para um registro específico, mas para um bloco de registros (e dentro do bloco é feita busca sequencial) – exige que os registros dentro de um bloco estejam ordenados

EXEMPLO DE ÍNDICE HIERÁRQUICO

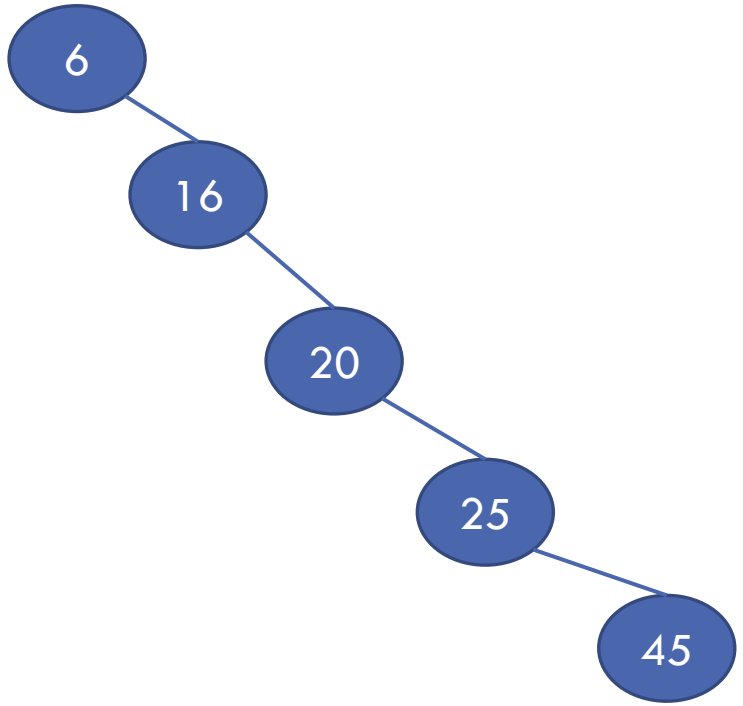


HIERARQUIA LEMBRA ÁRVORE...

A maioria das estruturas de índice é implementada por árvores de busca

- Árvores Binárias de Busca
- Árvores AVL
- Árvores de Múltiplos Caminhos

CONSIDERAÇÕES SOBRE ÁRVORES BINÁRIAS DE BUSCA

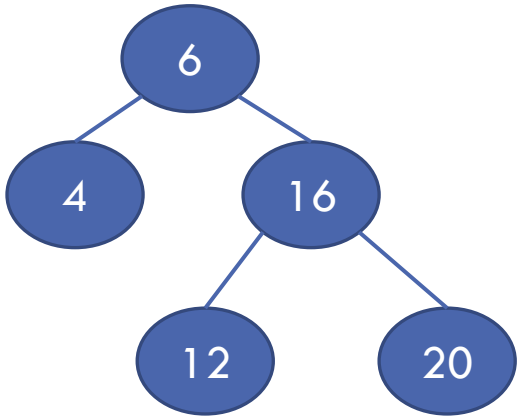


Altura tende a ser muito grande em relação ao número de nós ou registros que ela contém

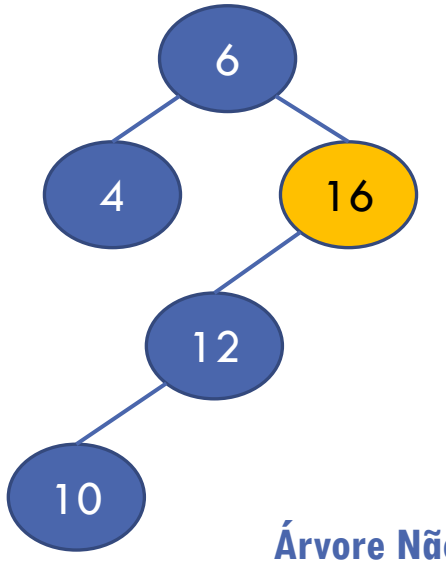
Se as chaves a serem incluídas estiverem ordenadas, a árvore degrada-se rapidamente, tornando-se uma lista encadeada

CONSIDERAÇÕES SOBRE ÁRVORES AVL

Apesar de serem árvores binárias de busca balanceadas, ainda são excessivamente altas para uso eficiente como estrutura de índice



Árvore AVL



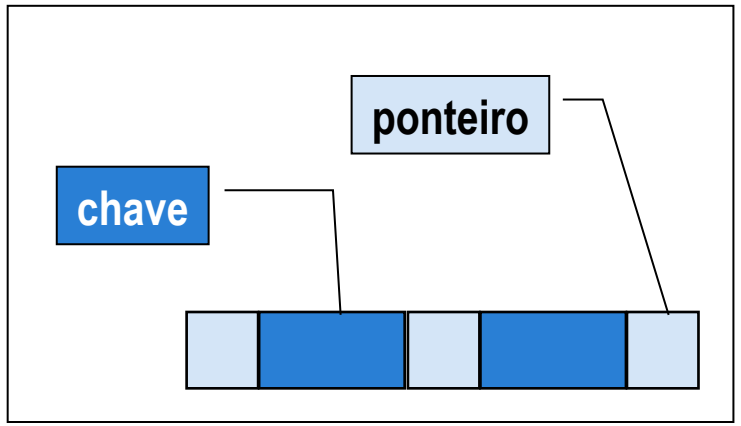
Árvore Não-AVL

SOLUÇÃO: ÁRVORES DE MÚLTIPLOS CAMINHOS

Características

- Cada nó contém ***n-1*** chaves
- Cada nó contém ***n*** filhos
- As chaves dentro do nó estão ordenadas
- As chaves dentro do nó funcionam como separadores para os ponteiros para os filhos do nó

EXEMPLO



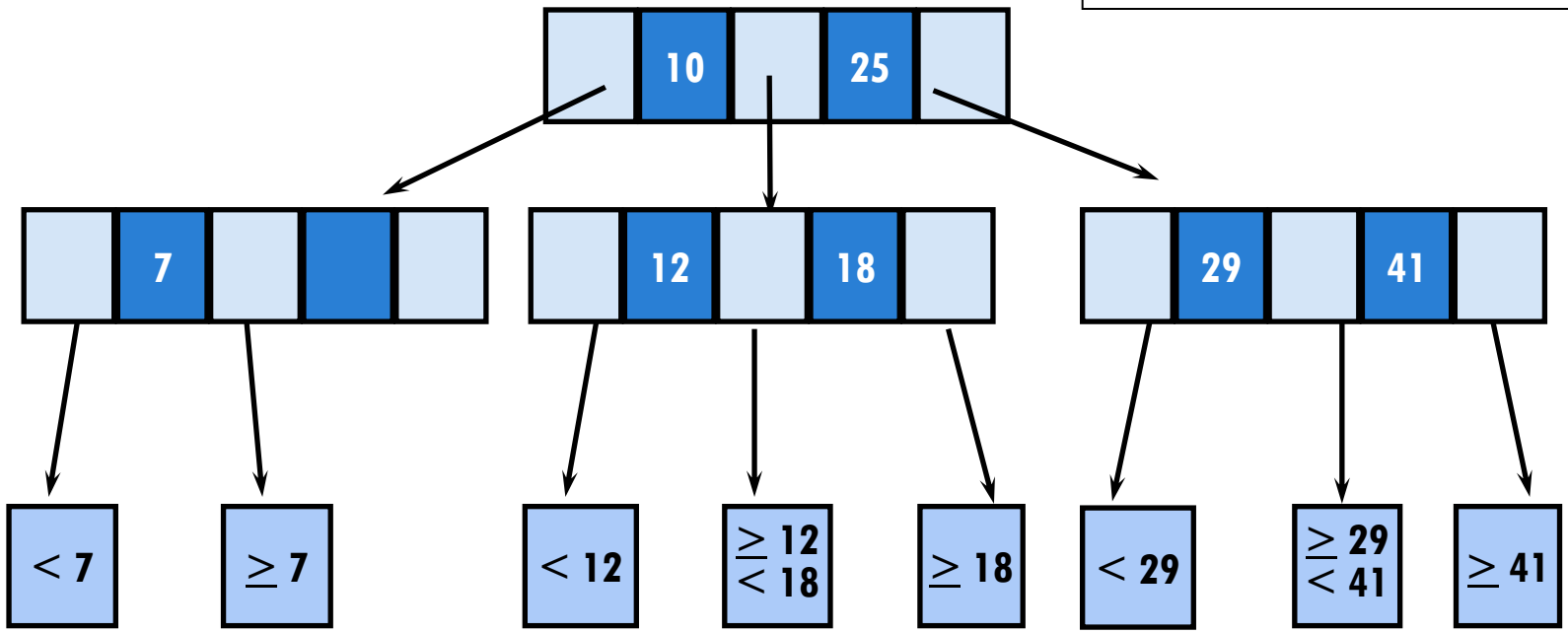


Figura: gentileza de Clesio S. Santos e Nina Edelweiss

EXEMPLOS DE ÁRVORES MÚLTIPLOS CAMINHOS

Árvore B

Árvore B+

VANTAGENS

Têm altura bem menor que as árvores binárias

Ideais para uso como **índice de arquivos em disco**

Como as árvores são baixas, são necessários poucos acessos em disco até chegar ao ponteiro para o bloco que contém o registro desejado

ÁRVORE B

Fonte de consulta: Szwarcfiter, J.; Markezon, L. Estruturas de Dados e seus Algoritmos, 3a. ed. LTC. Seção 5.5

ÁRVORE B

Consegue armazenar índice e dados na mesma estrutura (mesmo arquivo físico)

Características de uma árvore B de ordem **d**

- A raiz é uma folha ou tem no mínimo 2 filhos
- Cada nó interno (não folha e não raiz) possui no mínimo **d + 1** filhos
- Cada nó tem no máximo **2d + 1** filhos
- Todas as folhas estão no mesmo nível

Um nó de uma árvore B é também chamado de página

Uma página armazena diversos registros da tabela original

- Seu tamanho normalmente equivale ao tamanho de uma página em disco

ÁRVORE B

Seja **m** o número de chaves de uma página **P** não folha

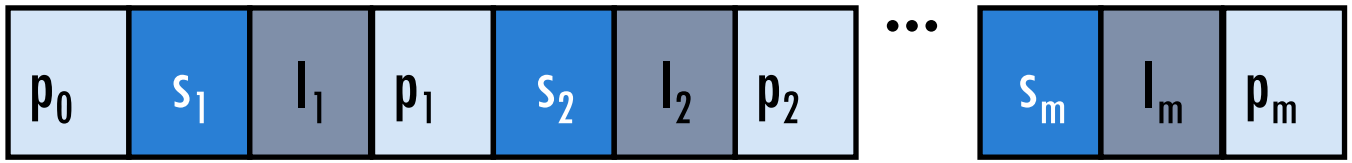
- **P** tem **m+1** filhos, **P** tem entre **d** e **2d** chaves, exceto o nó raiz, que possui entre **1** e **2d** chaves
- Em cada página, as chaves estão ordenadas: $s_1, \ldots, s_m$, onde $d \leq m \leq 2d$, exceto para a raiz onde $1 \leq m \leq 2d$
- **P** contém **m+1** ponteiros $p_0, p_1, \ldots, p_m$ para os filhos de P
- Nas páginas correspondentes às folhas, esses ponteiros apontam para NULL
- Os nós também armazenam, além da chave s_k, os dados (l_k) relativos àquela chave

ÁRVORE B

Seja uma página **P** com **m** chaves:

- para qualquer chave **y** pertencente à primeira página apontada por **P** (ou seja, apontada por p_0), $y < s_1$
- para qualquer chave **y** pertencente à página apontada por p_k, $1 \leq k \leq m-1$, $s_k < y < s_{k+1}$
- para qualquer chave **y** pertencente à página apontada por p_m, $y > s_m$

Estrutura de uma página (nó)



REPRESENTAÇÃO EM C
ÁRVORE B EM MEMÓRIA PRINCIPAL

```
typedef struct No {
   int m; //quantidade de chaves armazenadas no nó
   struct No *pont_pai; //pt para o nó pai
   int *s; //array de chaves
   struct No **p; //pt para array de pt p/ os filhos
} TNo;

//Essa estrutura é uma simplificação:
//na prática também seria necessário armazenar os outros dados
//dos registros, e não apenas as chaves
```

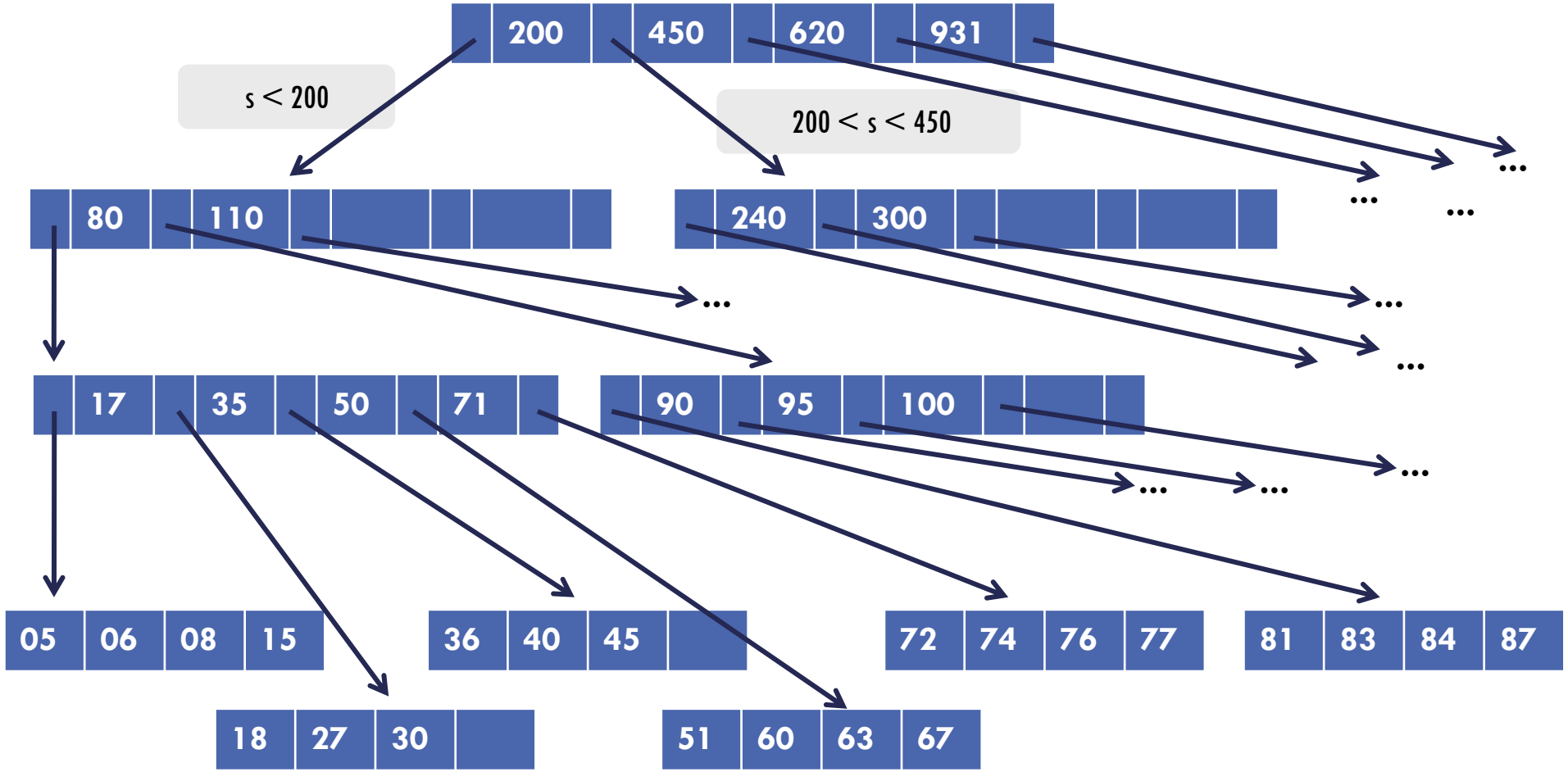
BUSCA DE UMA CHAVE X EM ÁRVORE B EM MEMÓRIA PRINCIPAL

1. pt = ponteiro p/ raiz da árvore
2. Percorra as chaves do nó apontado por pt, até encontrar X, ou até encontrar uma chave > X, ou até percorrer todas as chaves do nó
   a) Se encontrou chave X, encerre a busca retornando pt
   b) Senão, se encontrou chave maior que X, ptNovo = ponteiro da esquerda dessa chave
   c) Senão, se percorreu todas as chaves do nó atual, ptNovo = ponteiro da direita da última chave do nó
   d) Se ptNovo = NULL, encerre a busca, retornando pt (chave não está na árvore, mas, se estivesse, deveria estar no nó apontado por pt)
   e) Senão, pt = ptNovo, volte ao passo 2

Buscar chaves 240, 76 e 85 na árvore

EXEMPLO

ordem d = 2



Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

Exemplo cedido por Renata Galante

FUNÇÃO BUSCA

```
TNo *busca(TNo *no, int ch) {
    if (no != NULL) {
        int i = 0;
        while (i < no->m && ch > no->s[i]) {
            i++;
        }
        if (i < no->m && ch == no->s[i]) {
            return no; // encontrou chave
        } else if (no->p[i] != NULL) {
            return busca(no->p[i], ch);
        } else return no; //nó era folha -- não existem mais
nós a buscar, então retorna o nó onde a chave deveria estar
    } else return NULL; //nó é NULL, não há como buscar
}
```

INSERÇÃO

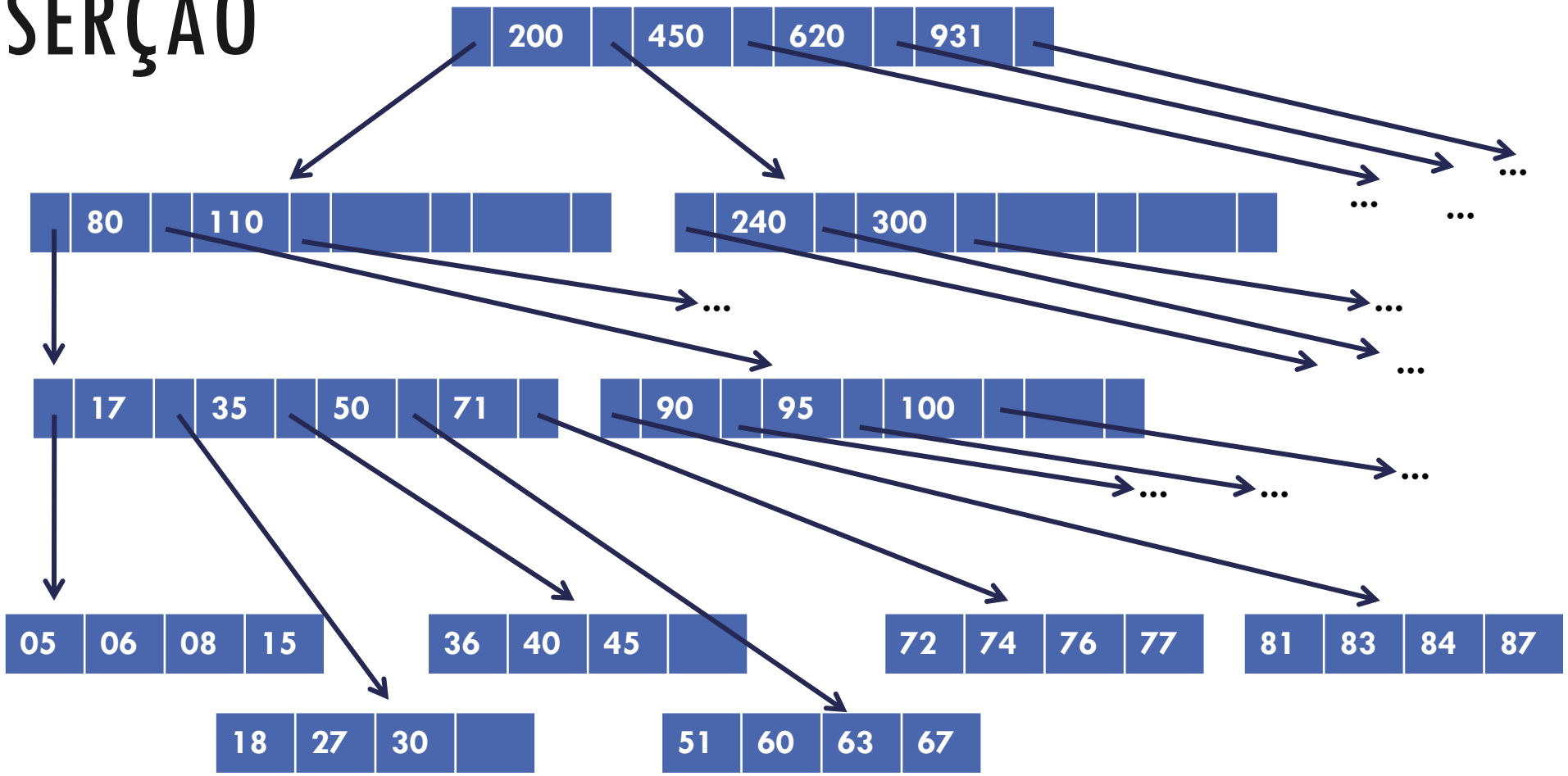
Para inserir um registro de chave **x** na árvore B

- Executar o algoritmo de busca
- Se chave está no nó retornado pela busca (é preciso checar)
  - Inserção é inválida
- Se chave não está no nó retornado pela busca:
  - Inserir a chave no nó retornado pela busca

Inserir chave 32

INSERÇÃO

ordem d = 2



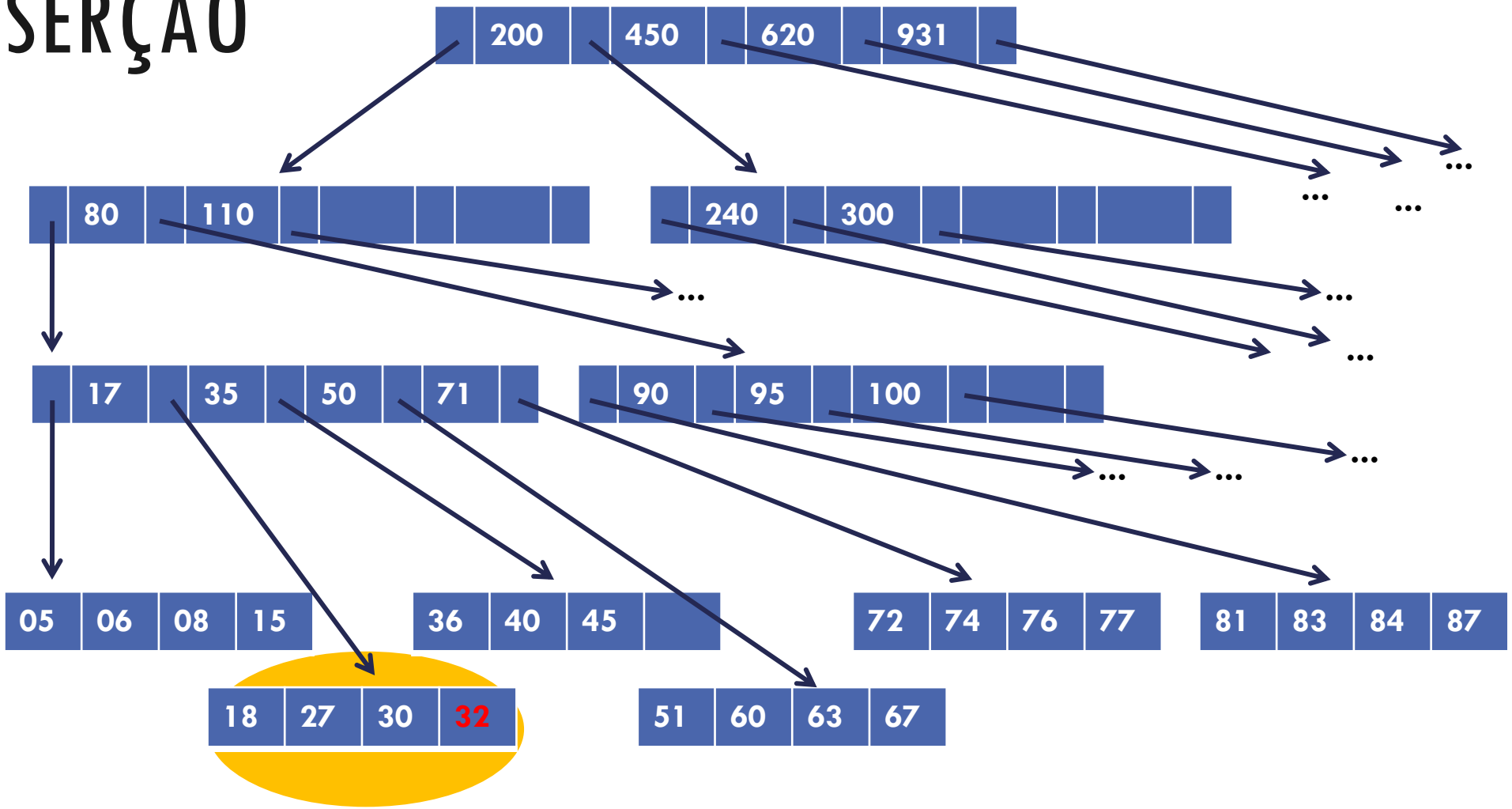
Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

Exemplo cedido por Renata Galante

Inserir chave 32

INSERÇÃO

ordem d = 2



Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

DISCUSSÃO SOBRE O ALGORITMO

Inserção sempre ocorre nas folhas

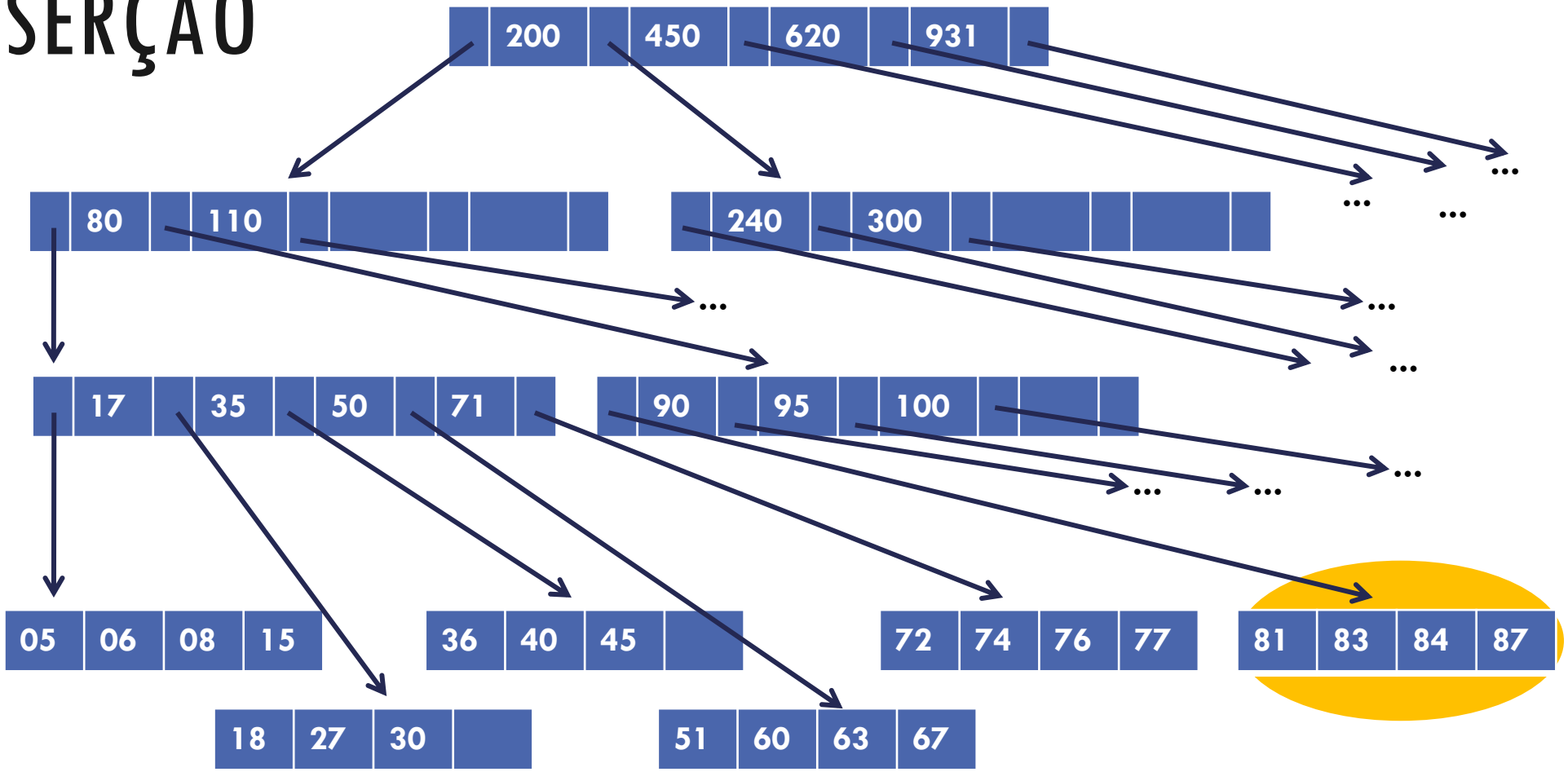
Por quê?

INSERÇÃO

Inserir chave 85

Inserção faria página ficar com 2d+1 chaves

ordem d = 2



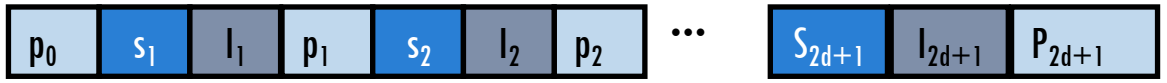
Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

PROBLEMA: PÁGINA CHEIA

É necessário reorganizar as páginas

Ao inserir uma chave em uma página cheia, sua estrutura ficaria da seguinte forma (omitindo I_k para simplificar)

- p_0, (s_1, p_1), (s_2,p_2), ..., (s_d,p_d), (s_{d+1},p_{d+1}), ..., (s_{2d+1}, p_{2d+1})



SOLUÇÃO

Particionar a página em 2

- Na página **P** permanecem **d** entradas
- Entrada **d+1** sobe para o pai
- Alocar outra página, **Q**, e nela alocar as outras **d** entradas

Após o particionamento

- Estrutura da página P:
  - $p_0, (s_1, p_1), (s_2, p_2), \ldots, (s_d, p_d)$
- Estrutura da página Q:
  - $p_{d+1}, (s_{d+2}, p_{d+2}) \ldots, (s_{2d+1}, p_{2d+1})$

ALOCAÇÃO DE S_{D+1}

O nó W, agora também pai de Q, receberá a nova entrada **(s_{d+1}, pt)**

- **pt** aponta para a nova página **Q**

Se não houver mais espaço livre em W, o processo de particionamento também é aplicado a W

PARTICIONAMENTO

Observação importante: particionamento se propaga para os pais dos nós, podendo, eventualmente, atingir a raiz da árvore

O particionamento da raiz é a **única forma de aumentar a altura da árvore**

PROCEDIMENTO DE INSERÇÃO

1. Aplicar o procedimento busca, verificando a validade da inserção
2. Se a inserção é válida, realizar inserção no nó F retornado pela busca
3. Verificar se nó F precisa de particionamento. Se sim, propagar o particionamento enquanto for necessário.

EXEMPLO DE INSERÇÃO QUE CAUSA PARTICIONAMENTO

Inserir chave 85

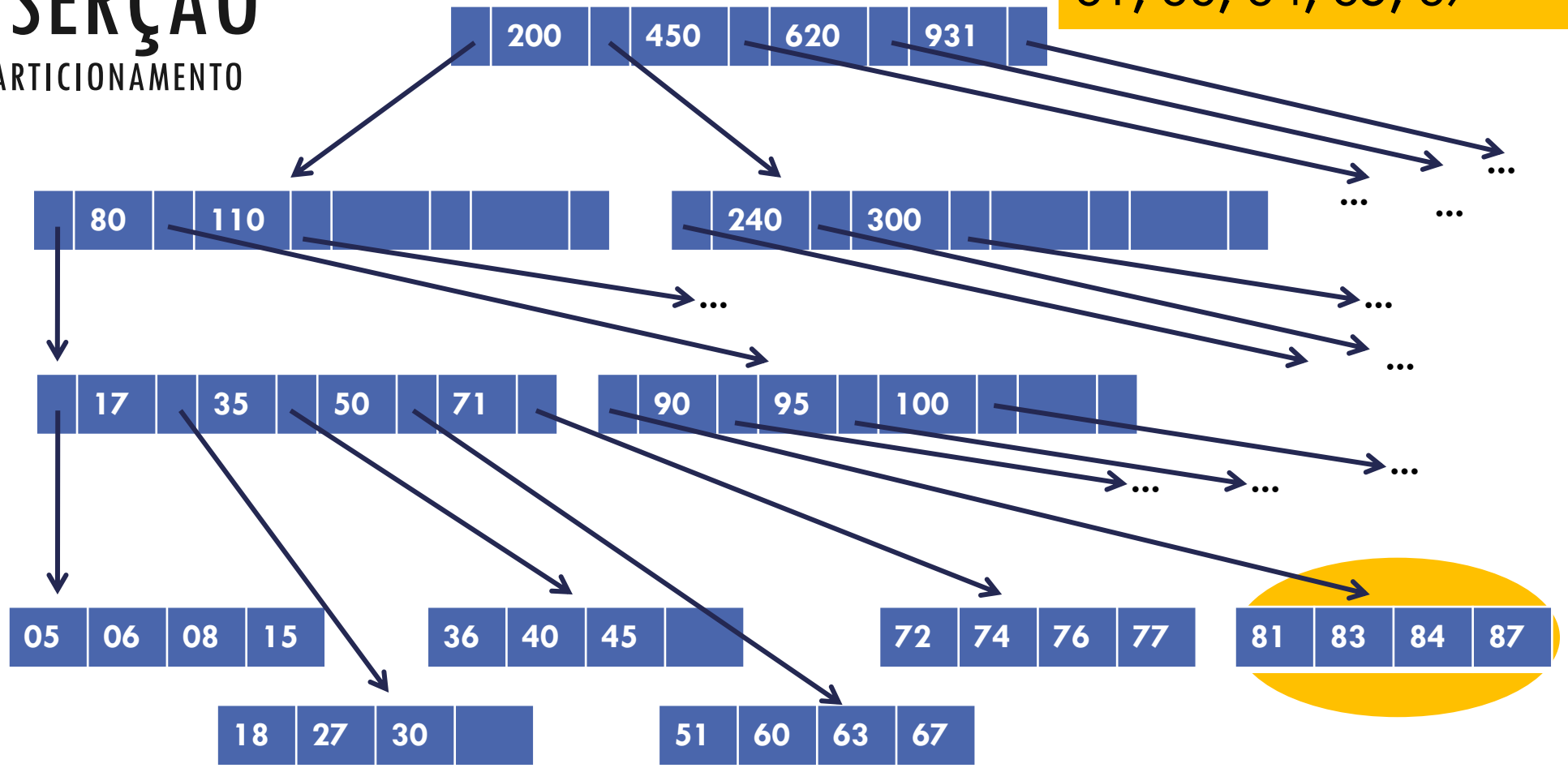
INSERÇÃO
C/ PARTICIONAMENTO

Inserir chave 85

Inserção faria página ficar com 2d+1 chaves

81; 83; 84; 85; 87

ordem d = 2

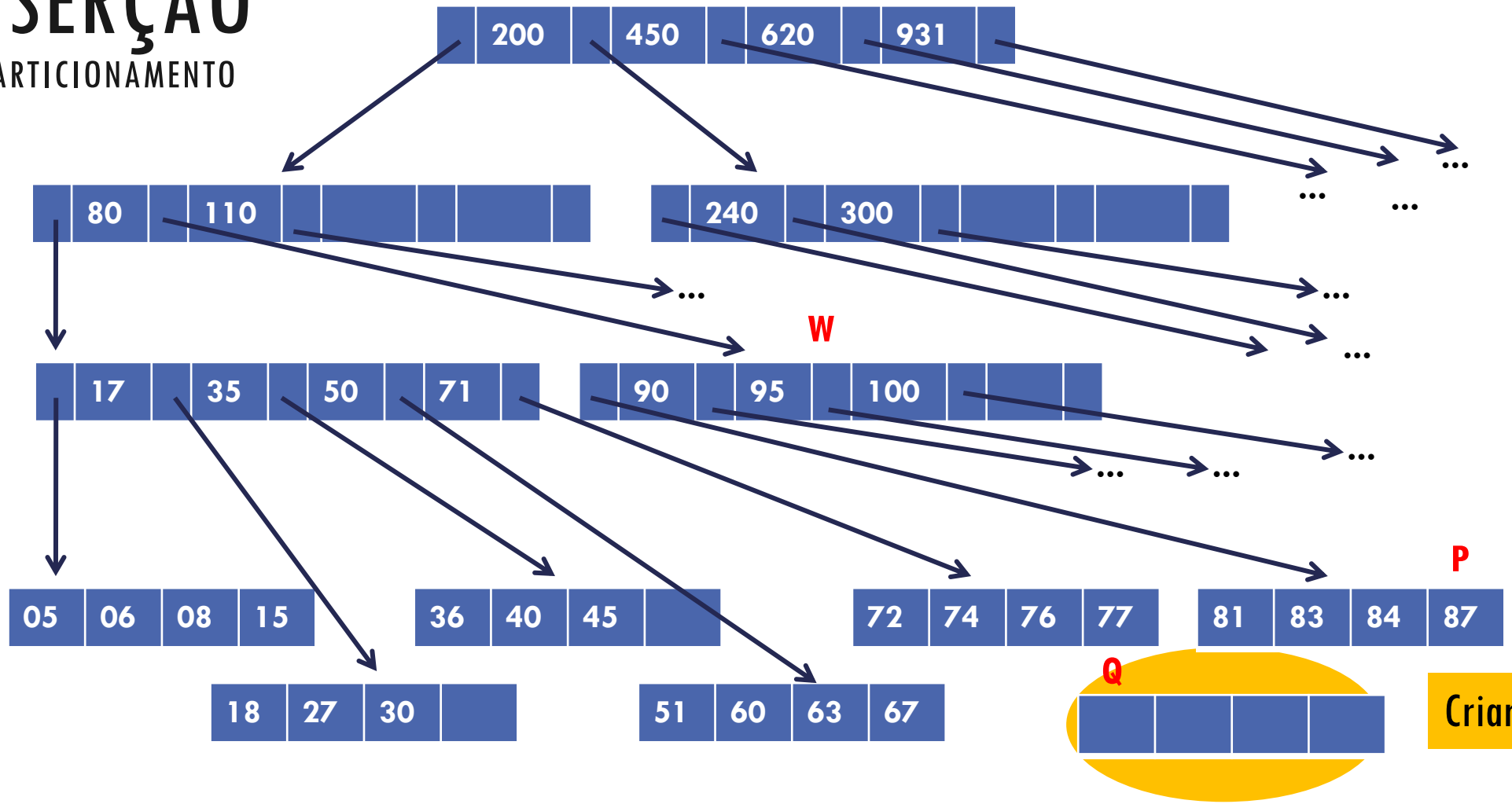


Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

Exemplo cedido por Renata Galante

INSERÇÃO
C/ PARTICIONAMENTO

ordem d = 2

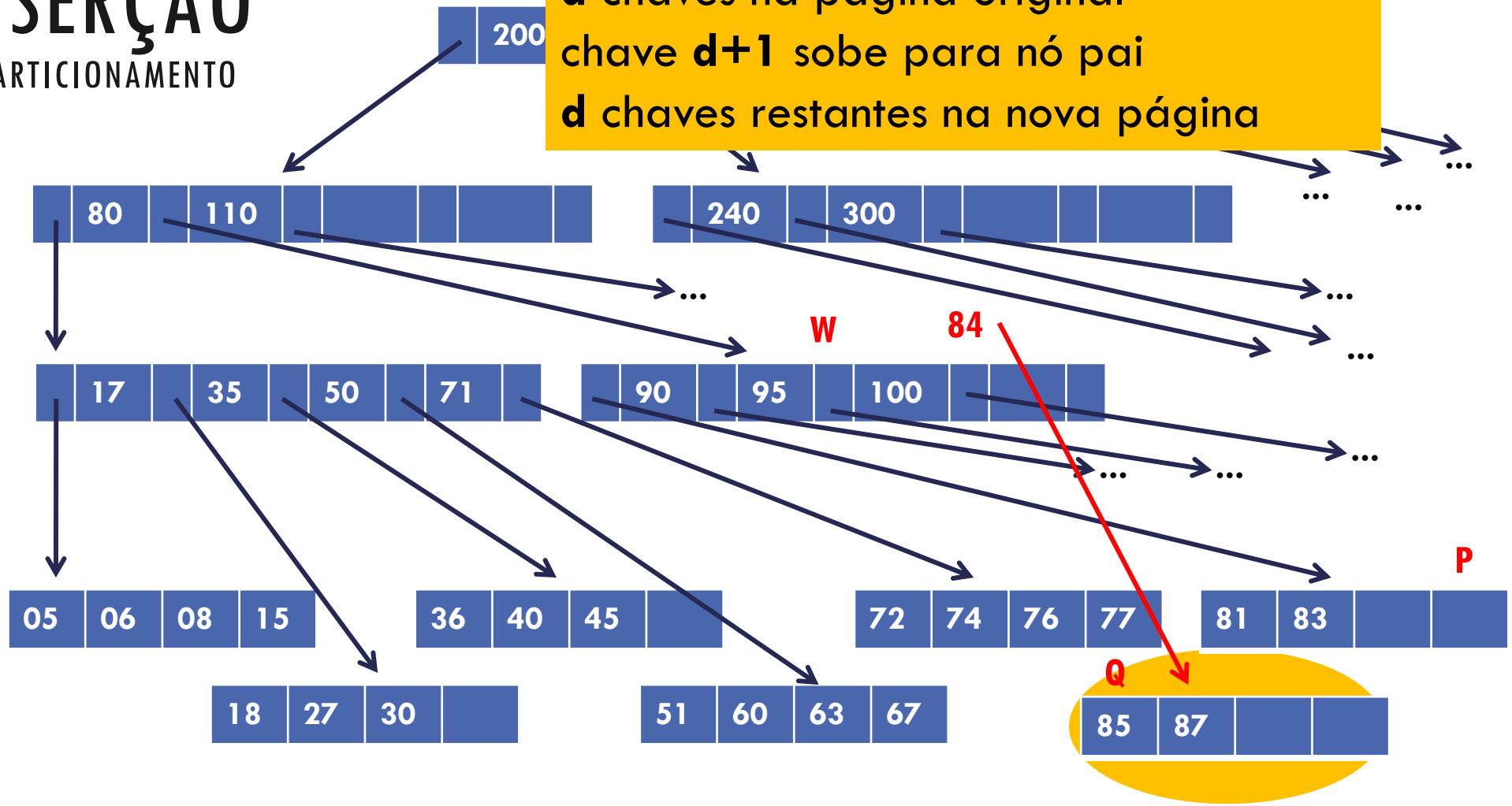


Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

INSERÇÃO
C/ PARTICIONAMENTO

Dividir as chaves entre as duas páginas
(81; 83; 84; 85; 87)
d chaves na página original
chave **d+1** sobe para nó pai
d chaves restantes na nova página

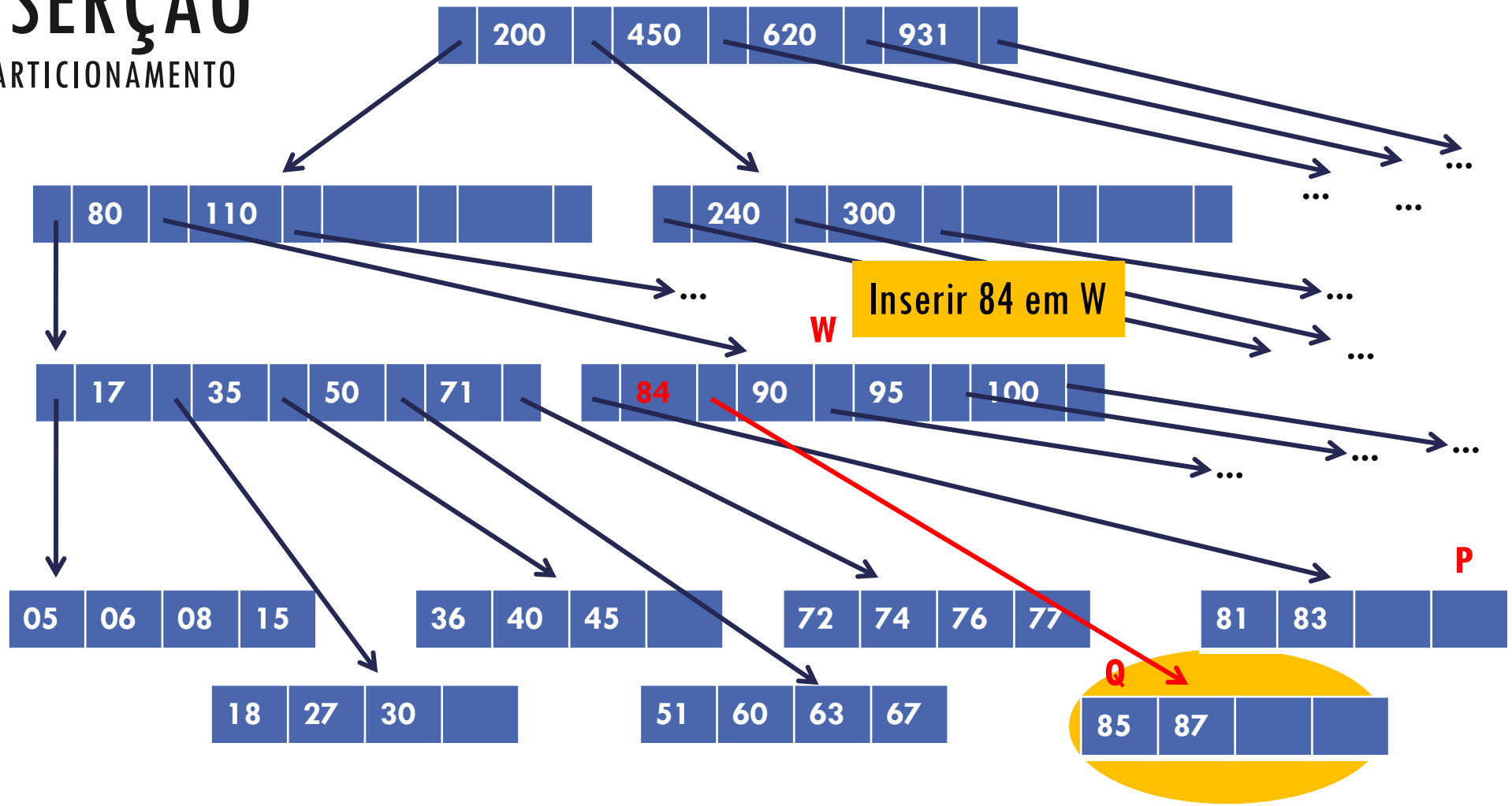
ordem d = 2



Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

INSERÇÃO
C/ PARTICIONAMENTO

ordem d = 2



Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

EXEMPLO DE PROPAGAÇÃO

Inserir chave 73

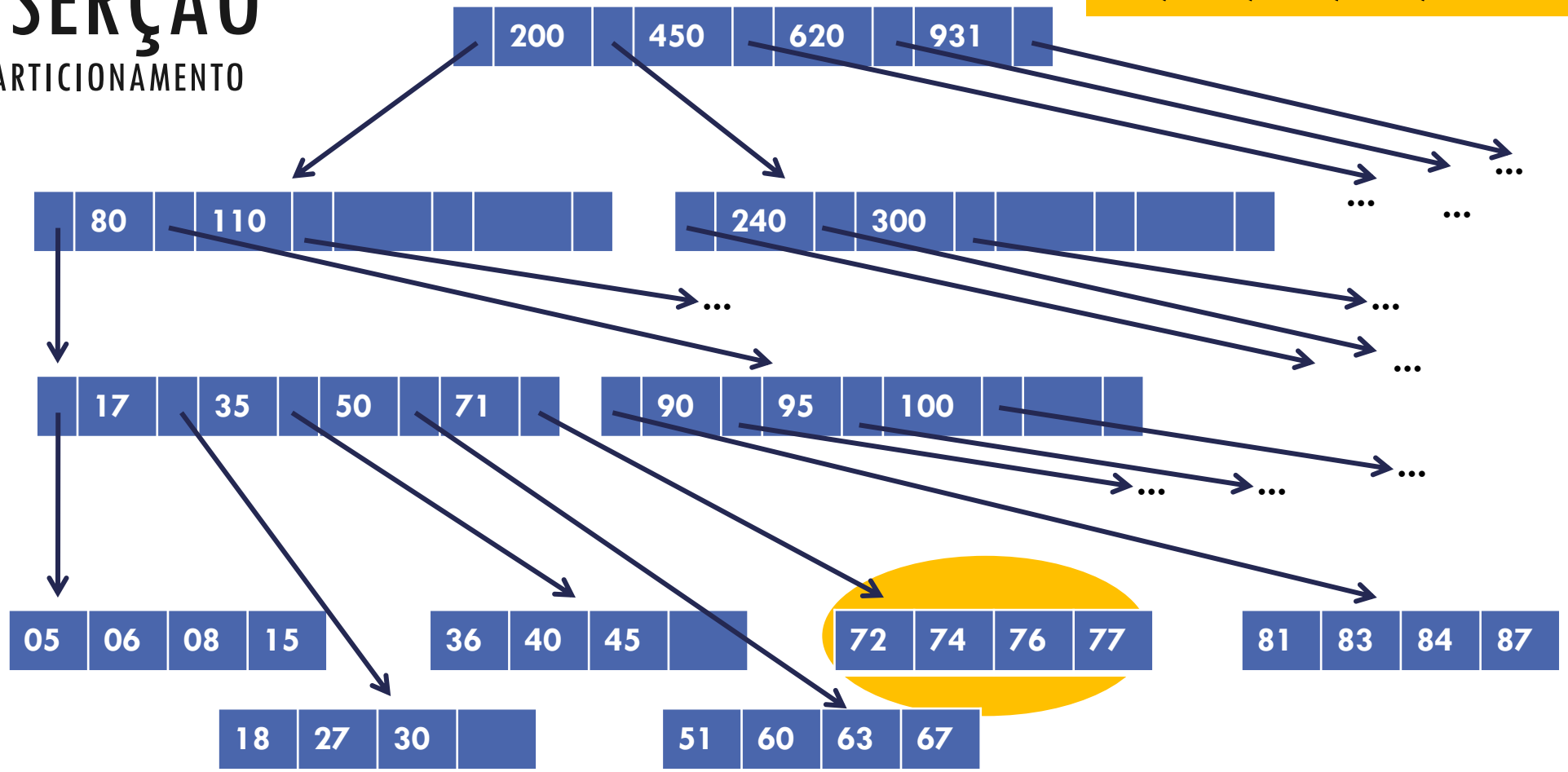
INSERÇÃO
C/ PARTICIONAMENTO

Inserir chave 73

Inserção faria página ficar com 2d+1 chaves

72; 73; 74; 76; 77

ordem d = 2

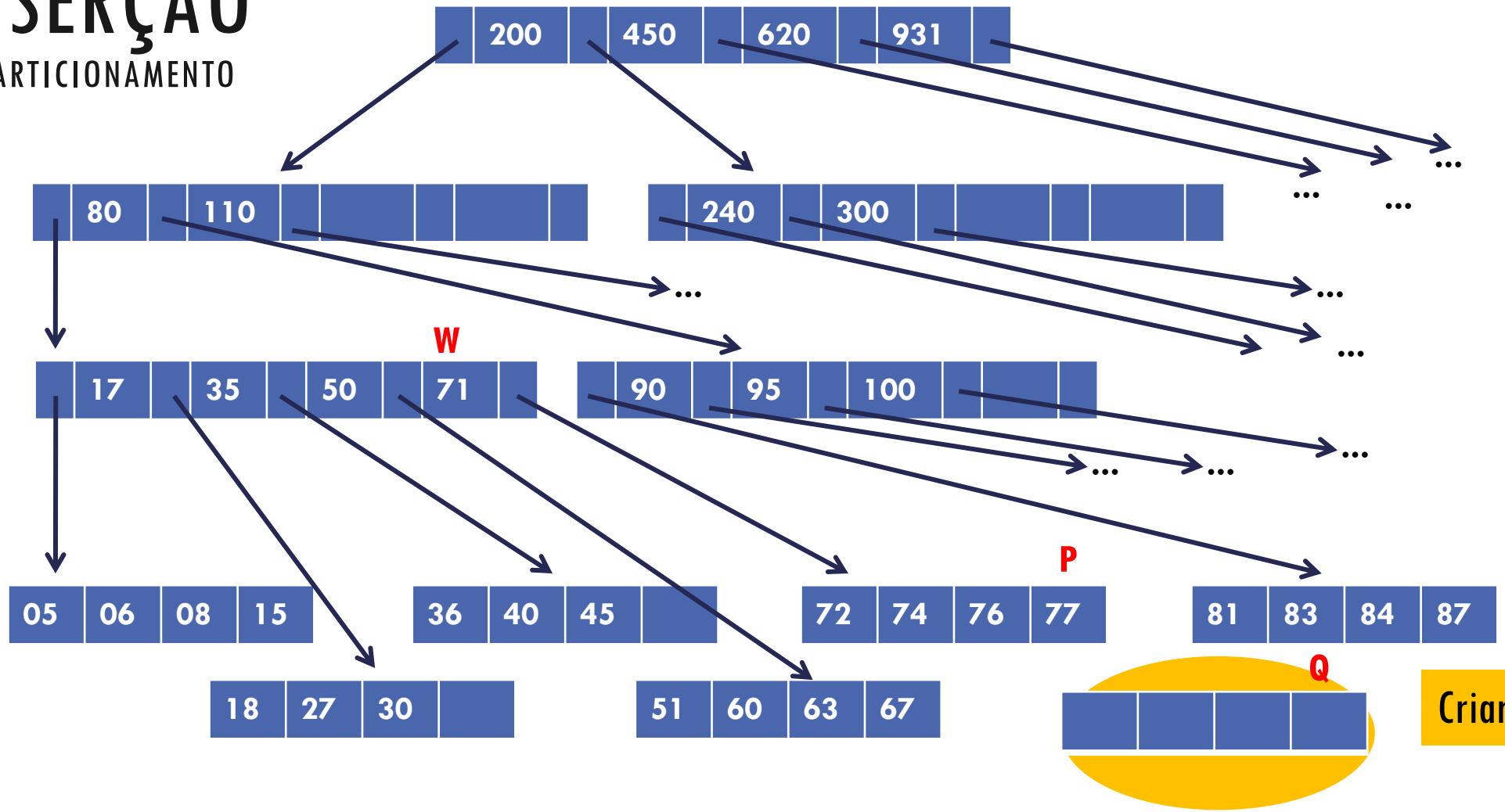


Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

Exemplo cedido por Renata Galante

INSERÇÃO
C/ PARTICIONAMENTO

ordem d = 2



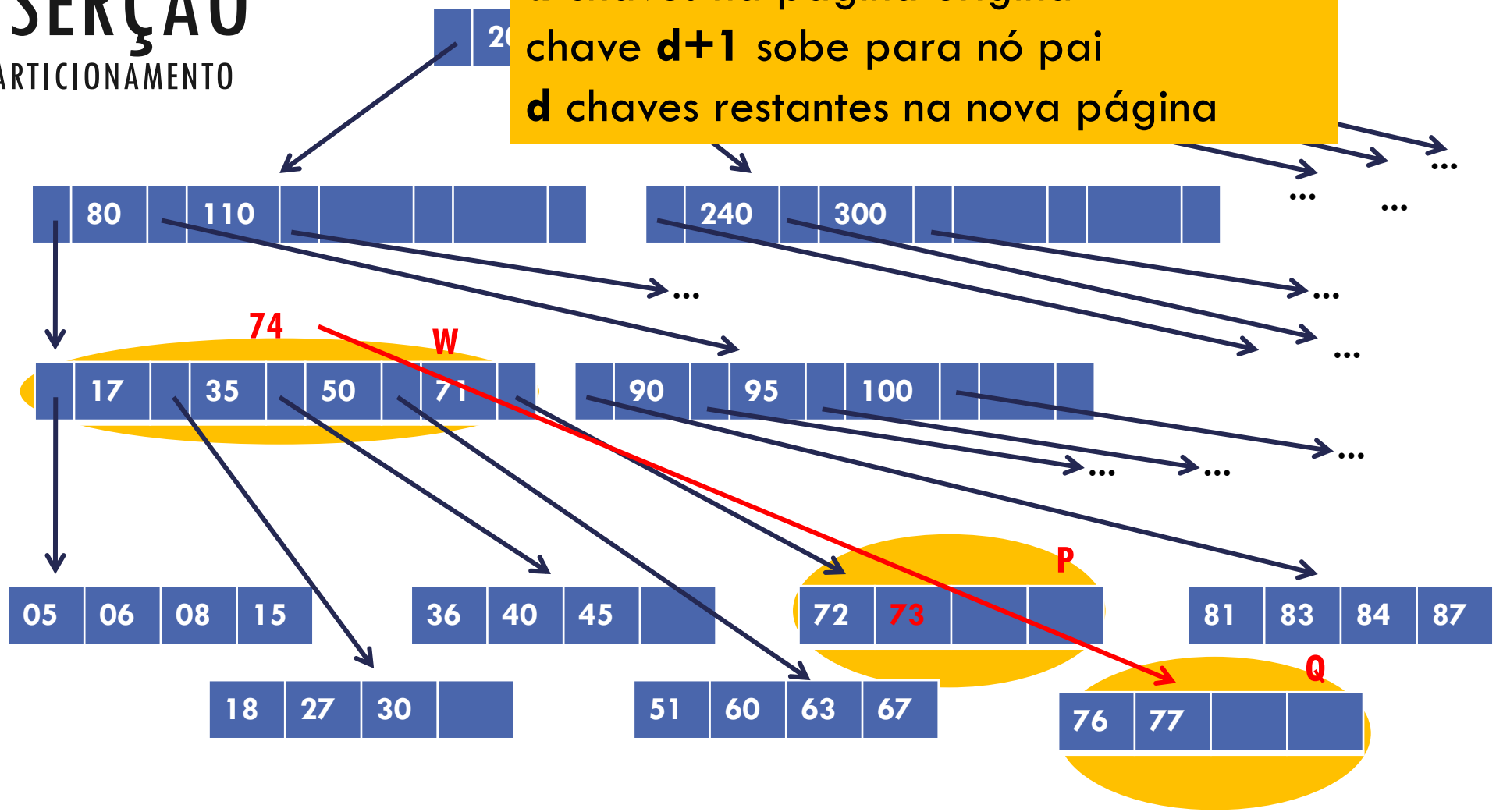
Criar nova página Q

Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

INSERÇÃO
C/ PARTICIONAMENTO

Dividir as chaves entre as duas páginas
(72; 73; 74; 76; 77)
d chaves na página original
chave **d+1** sobe para nó pai
d chaves restantes na nova página

ordem d = 2

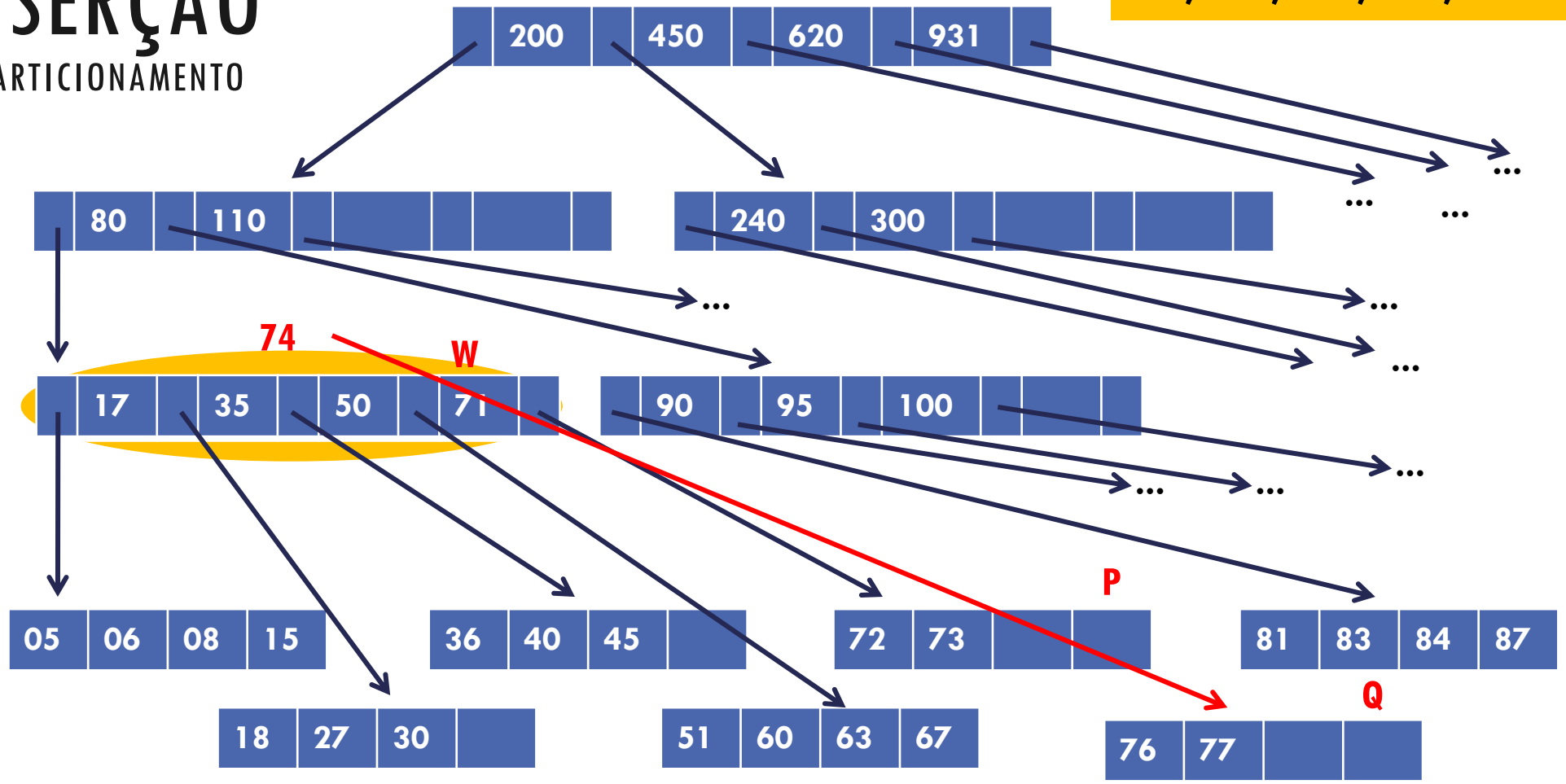


Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

INSERÇÃO
C/ PARTICIONAMENTO

Inserir 74 em W
Não há espaço: particionar nó
17; 35; 50; 71; 74

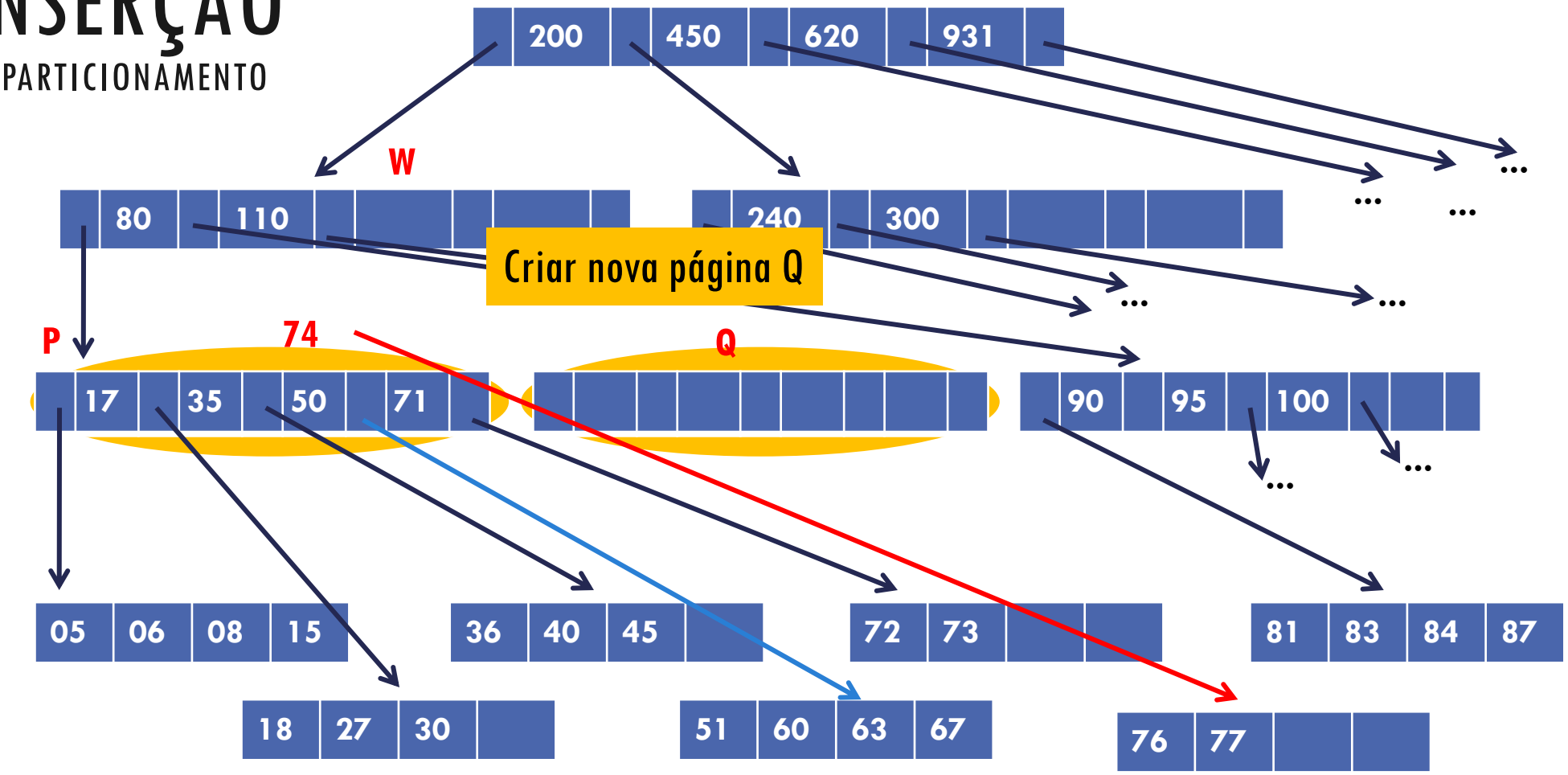
ordem d = 2



Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

INSERÇÃO
C/ PARTICIONAMENTO

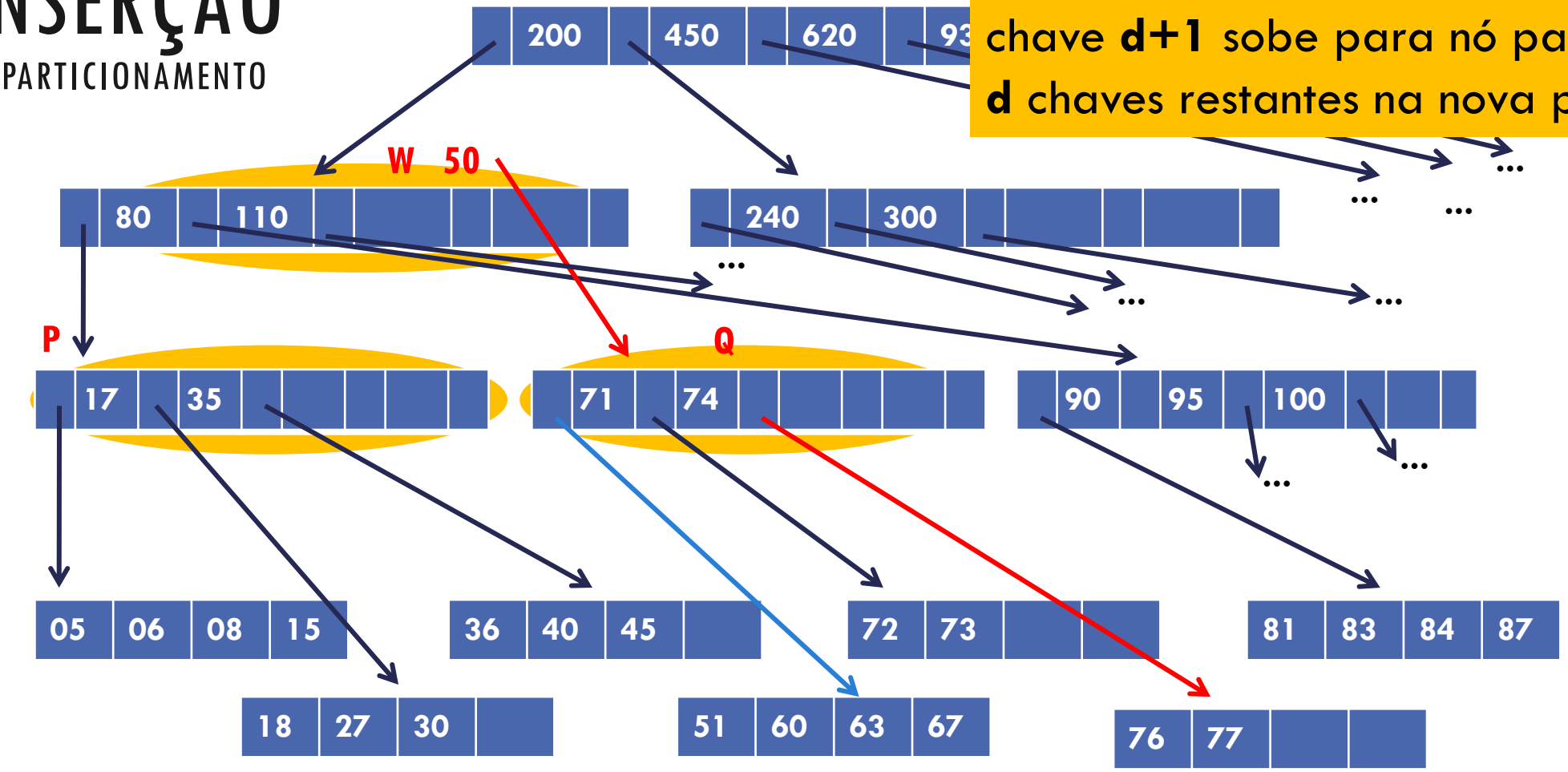
ordem d = 2



Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

INSERÇÃO
C/ PARTICIONAMENTO

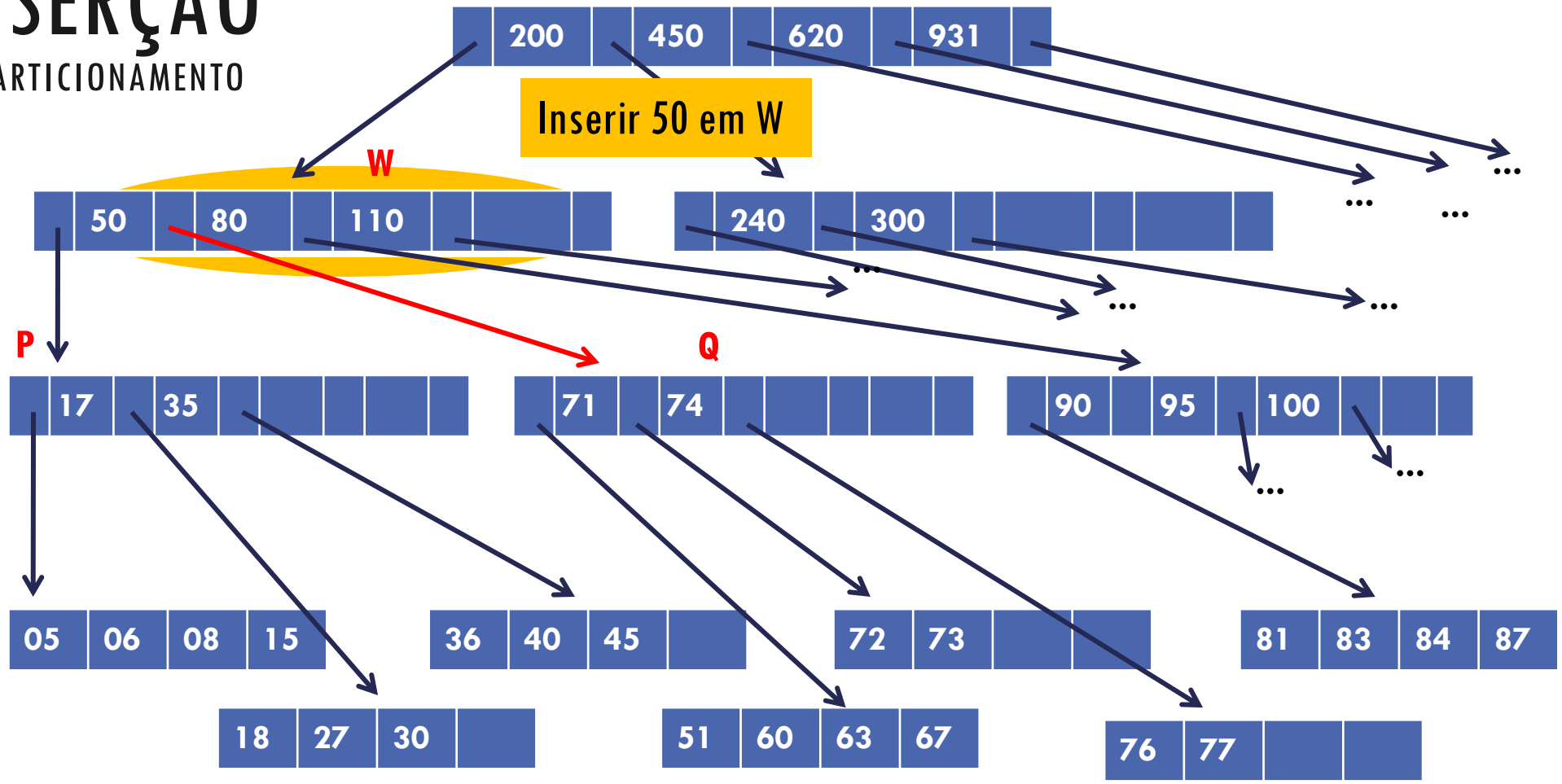
Dividir as chaves entre as duas páginas (17; 35; 50; 71; 74)
d chaves na página original
chave **d+1** sobe para nó pai
d chaves restantes na nova página



Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

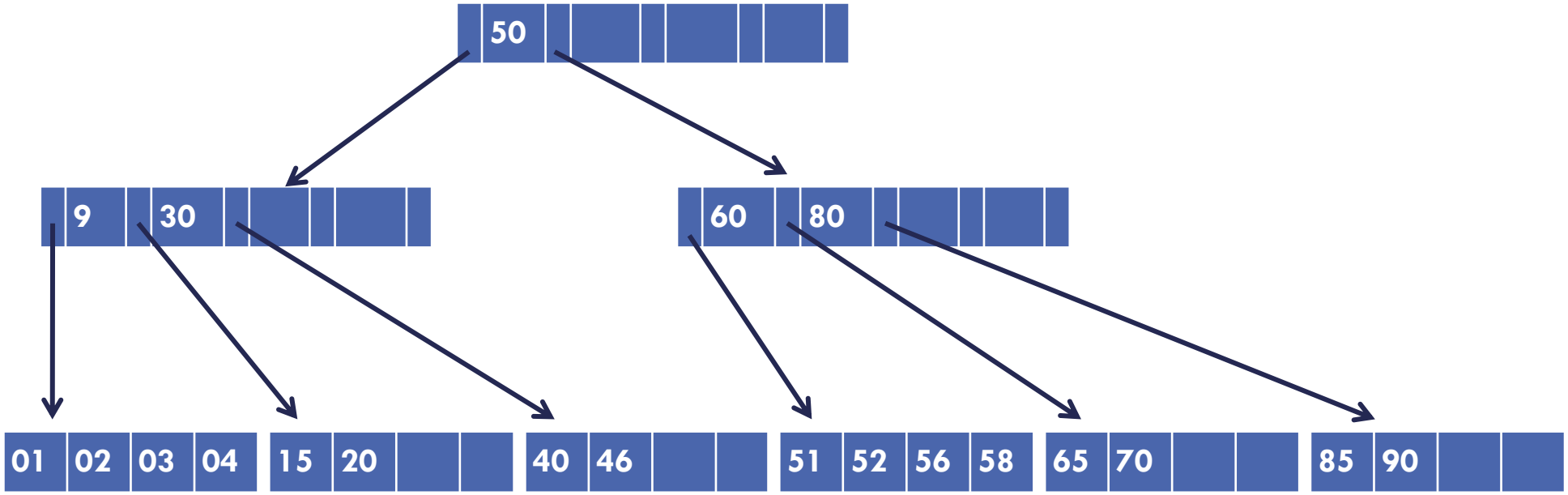
INSERÇÃO
C/ PARTICIONAMENTO

ordem d = 2



Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

EXERCÍCIO 1:
INSERIR CHAVES 57, 71, 72, 73



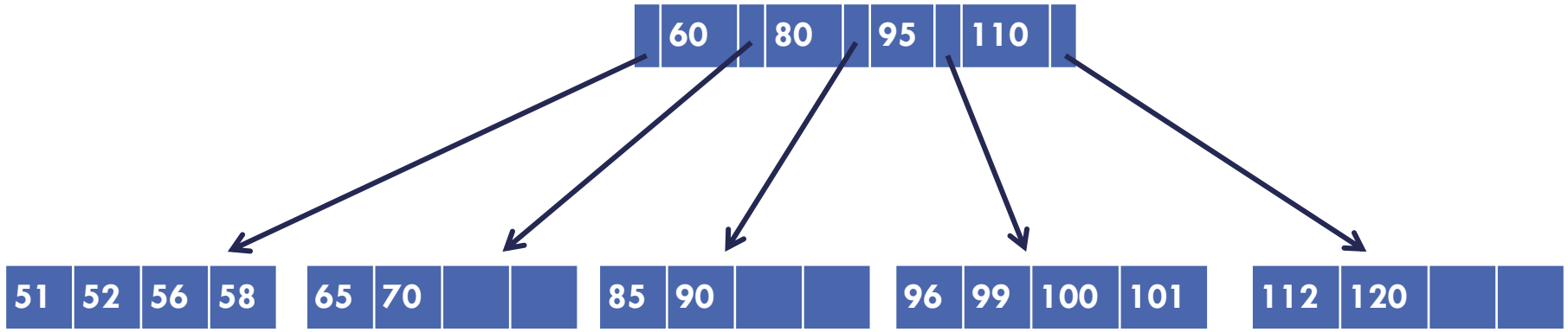
Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

DIVISÃO DO NÓ RAIZ

Em alguns casos o particionamento se propaga para a raiz

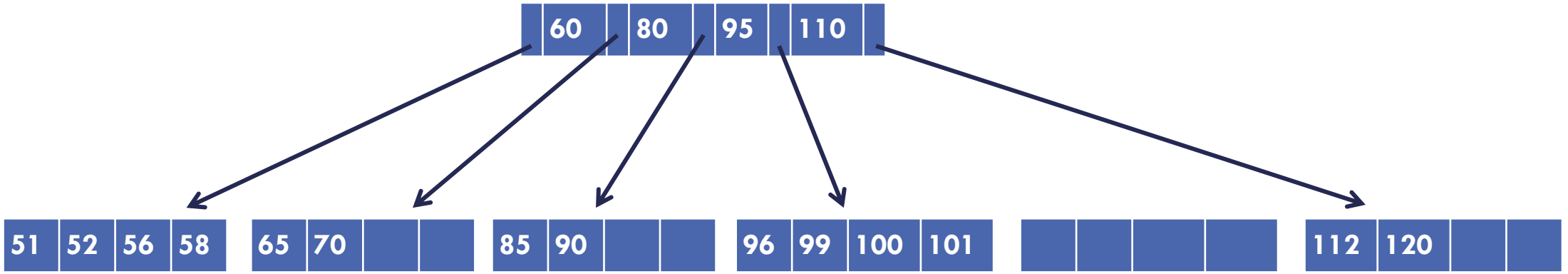
Nesse caso, o nó raiz é particionado normalmente, mas, como a raiz não tem pai, cria-se um novo nó, que passa a ser a nova raiz

EXEMPLO
INSERIR CHAVE 97



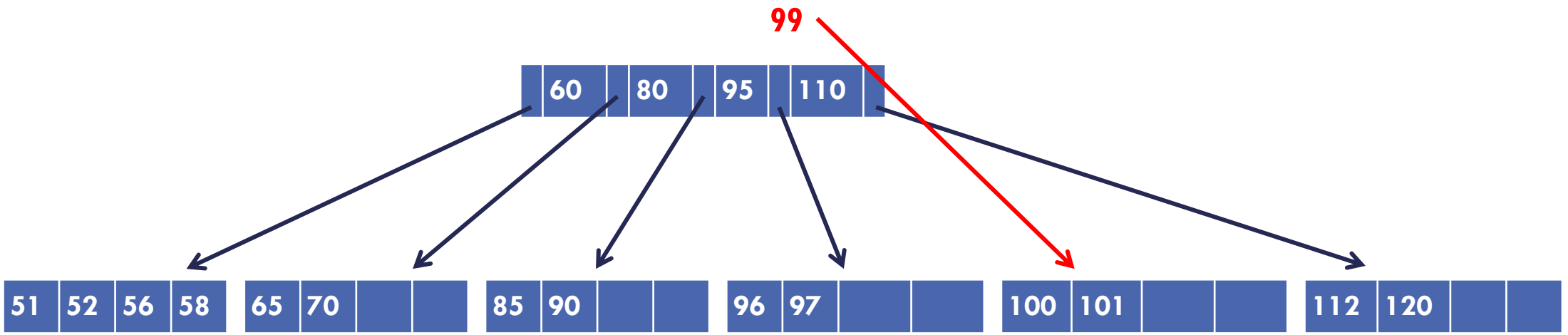
Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

EXEMPLO
INSERIR CHAVE 97



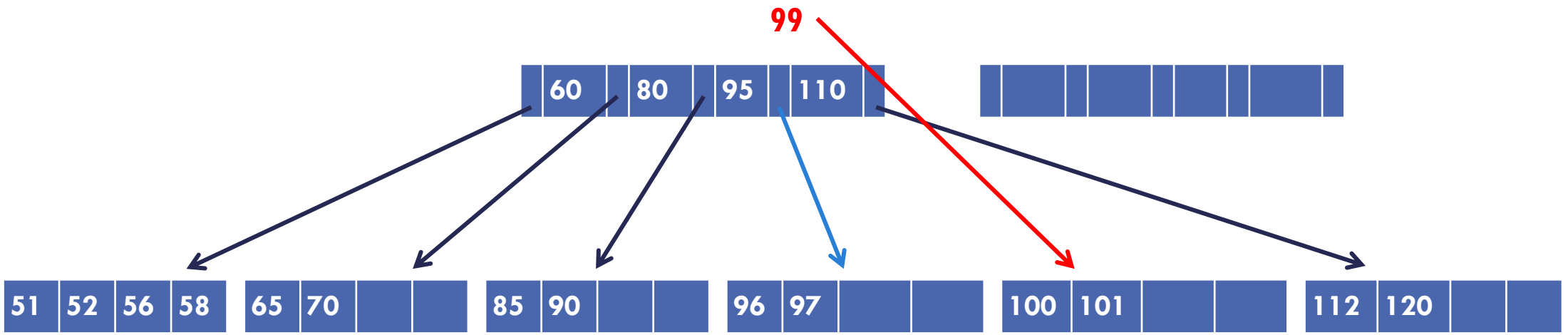
Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

EXEMPLO
INSERIR CHAVE 97



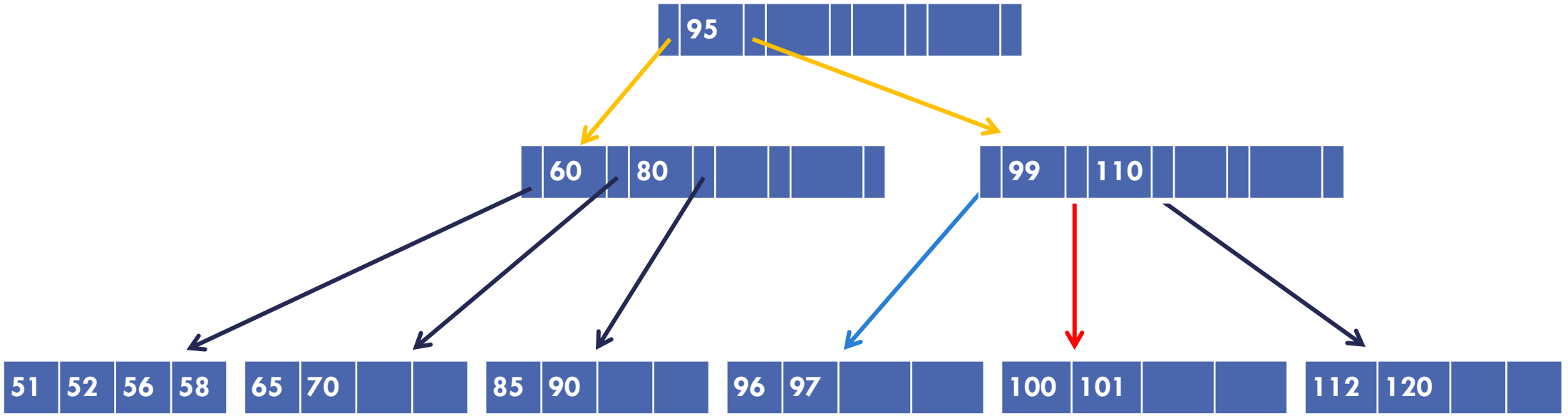
Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

EXEMPLO
INSERIR CHAVE 97



Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

EXEMPLO
INSERIR CHAVE 97



Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

IMPLEMENTAÇÃO

Ver implementação da inserção no site da disciplina

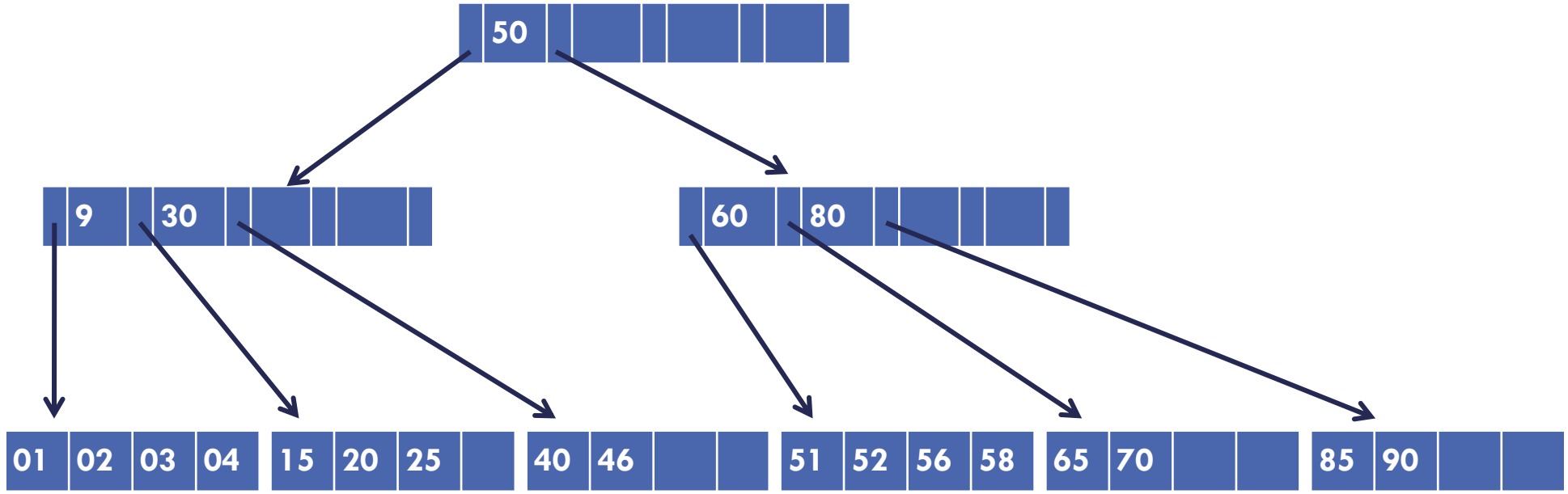
EXCLUSÃO

Duas situações possíveis:

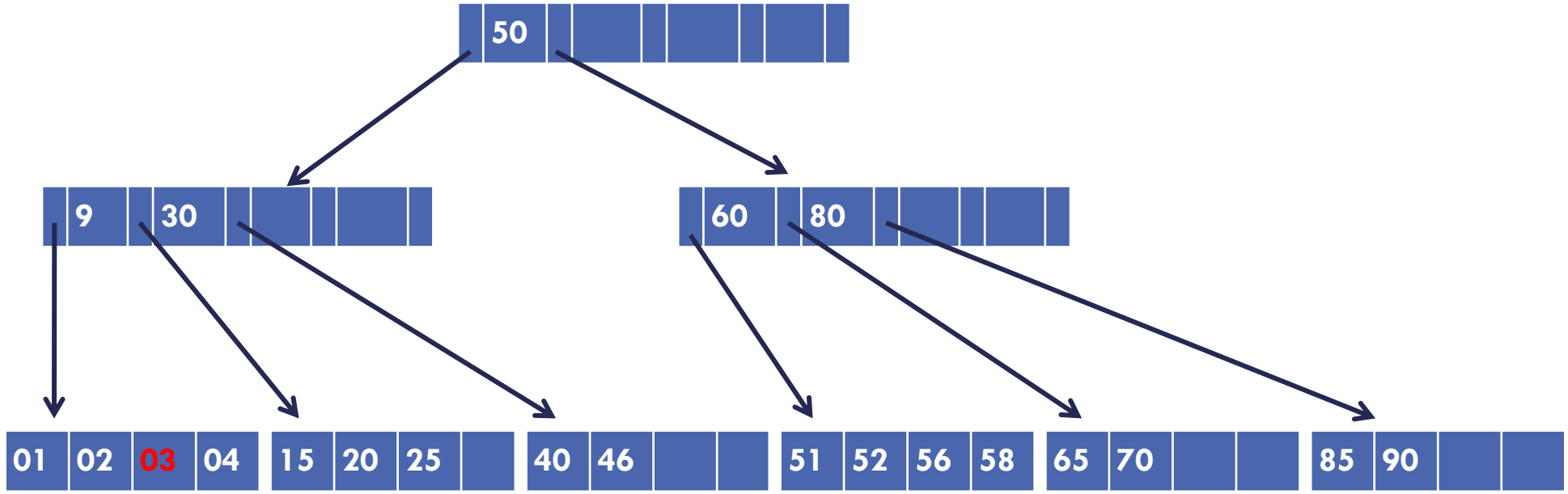
- A entrada **x** está em um nó folha
  - Neste caso, simplesmente remover a entrada **x**

- A entrada **x** não está em um nó folha
  - Substituir **x** pela chave **y** imediatamente maior
  - Note que **y** necessariamente pertence a uma folha, pela forma como a árvore B é estruturada

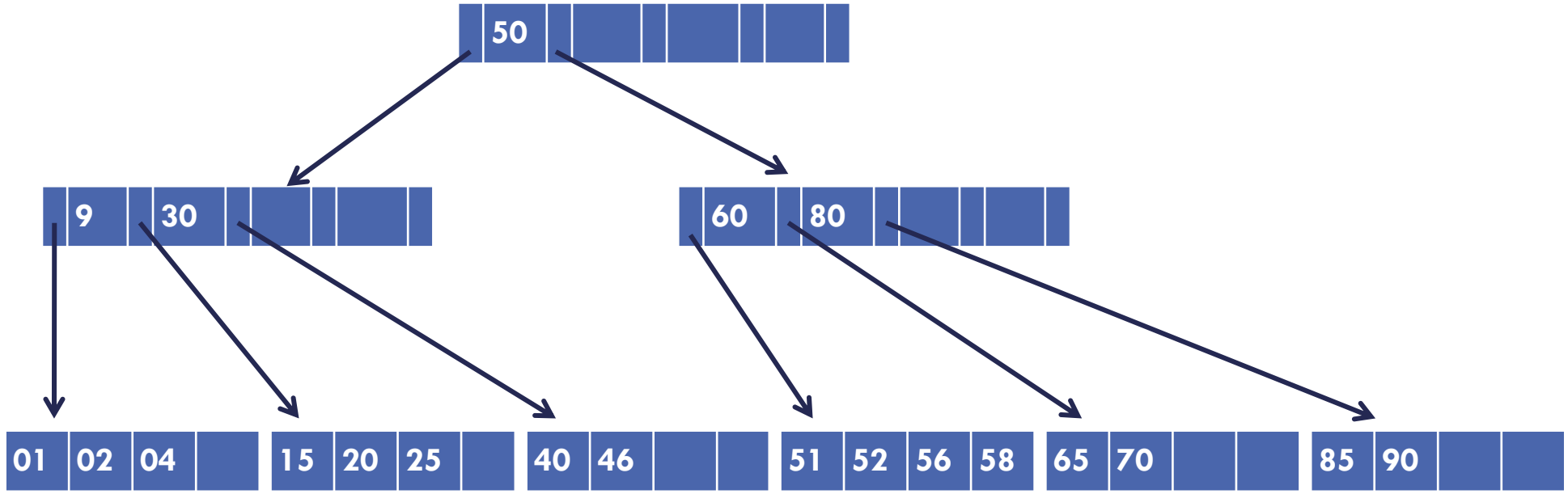
EXEMPLO: EXCLUSÃO DA CHAVE 03



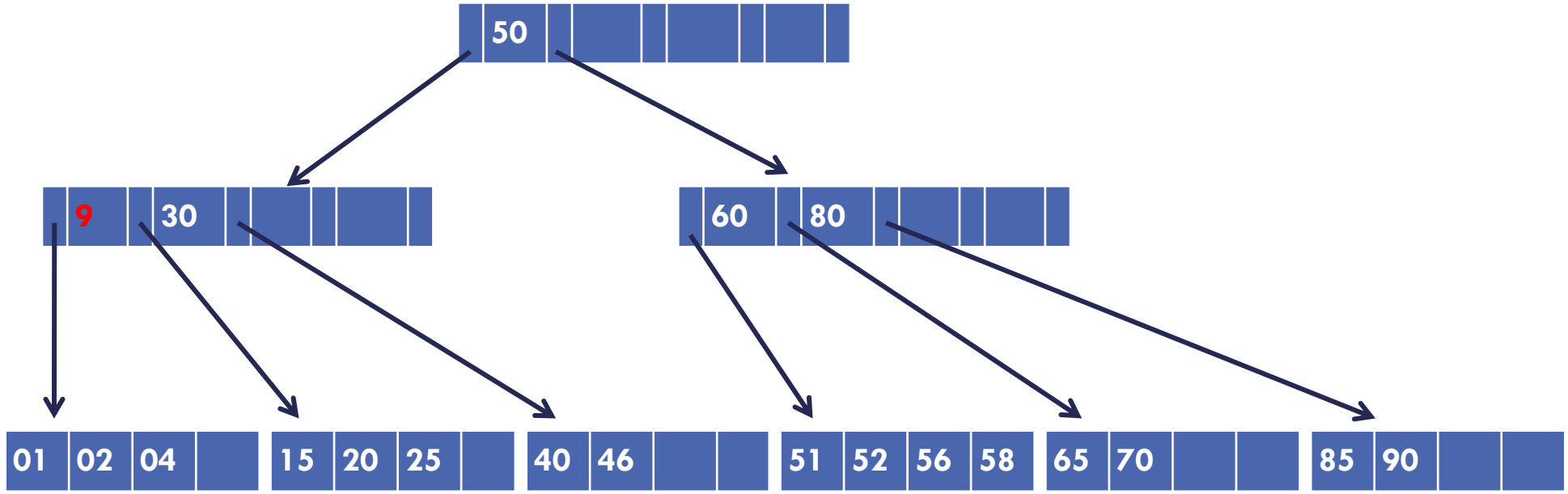
EXEMPLO: EXCLUSÃO DA CHAVE 03



EXEMPLO: EXCLUSÃO DA CHAVE 03

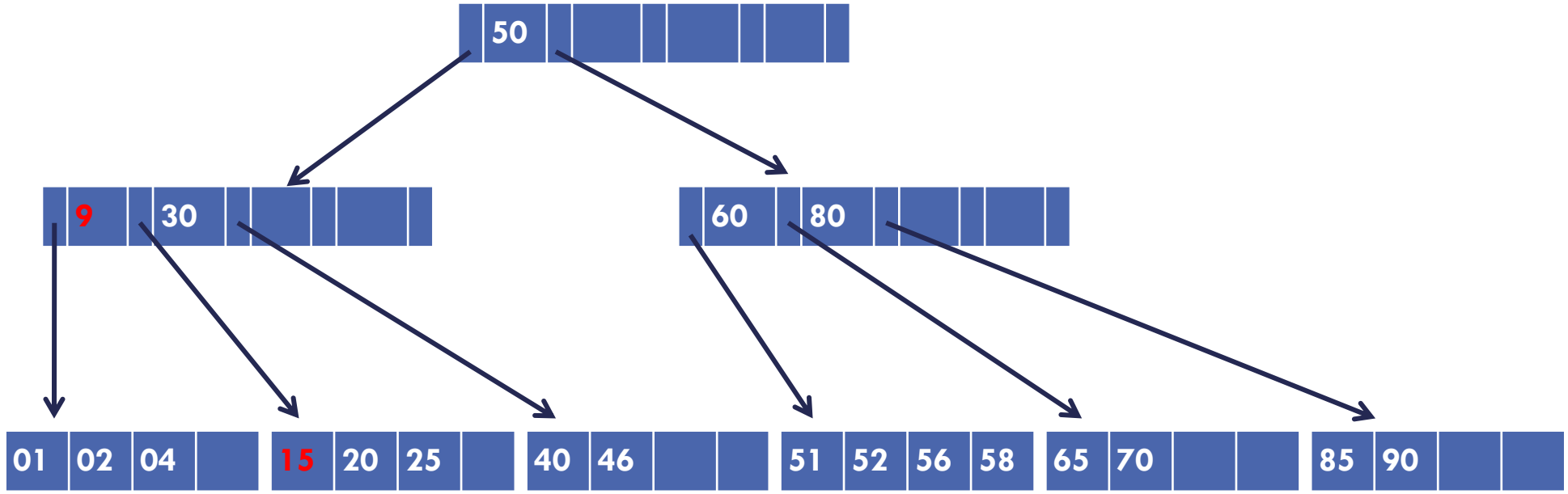


EXEMPLO: EXCLUSÃO DA CHAVE 9



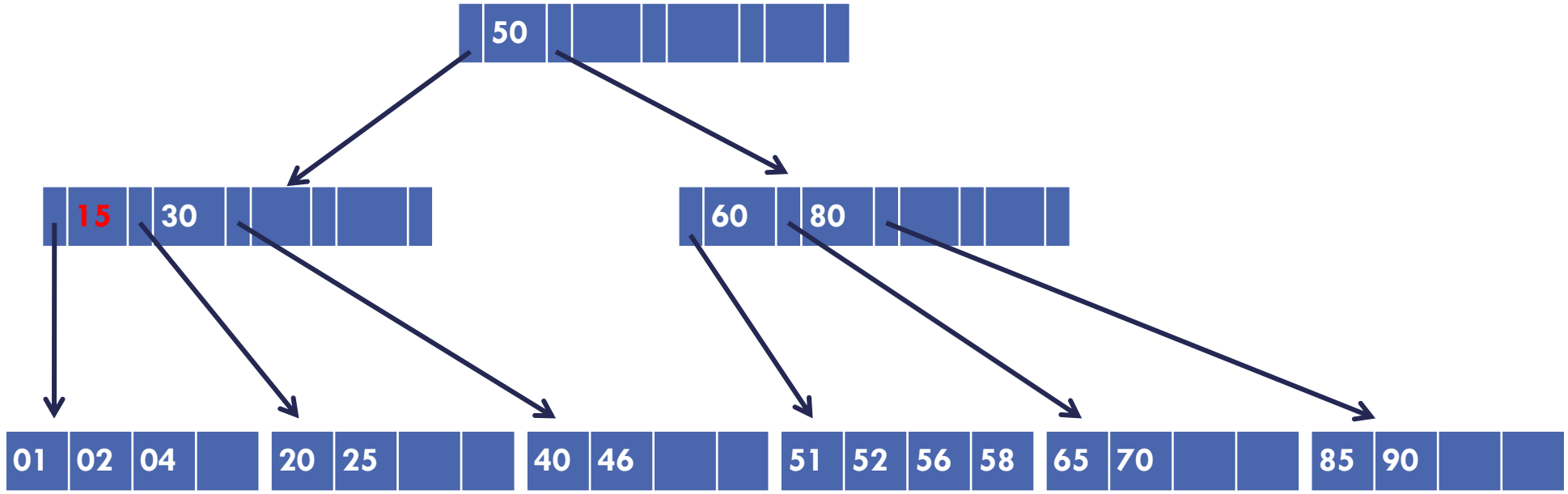
Substituir pela chave imediatamente maior

EXEMPLO: EXCLUSÃO DA CHAVE 9



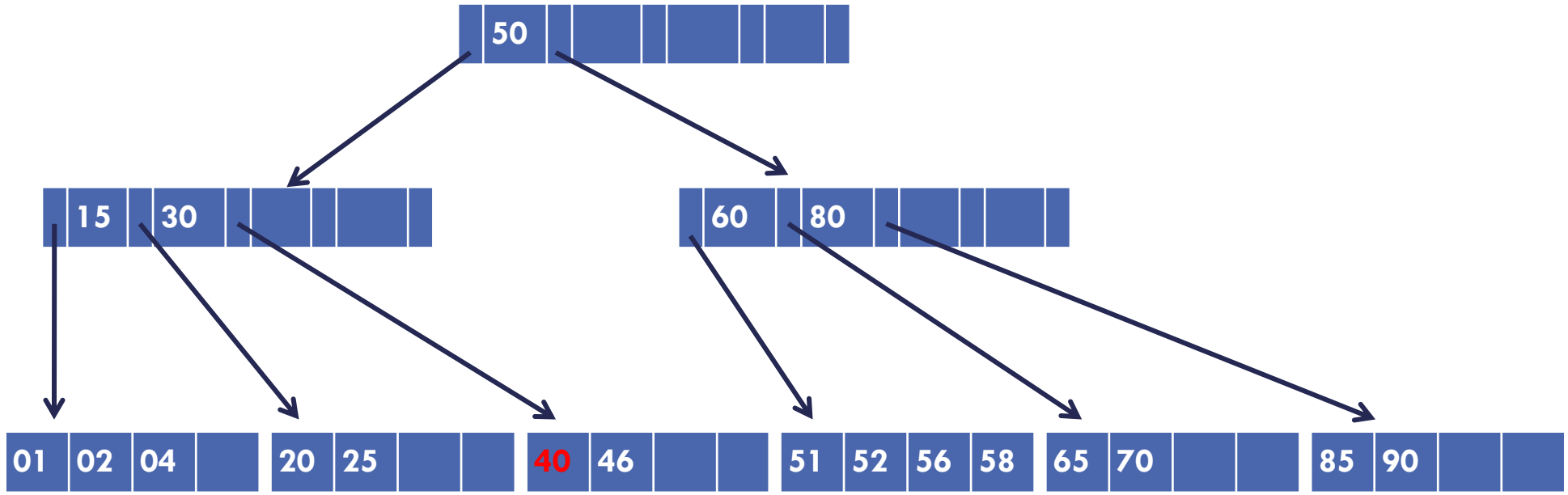
Substituir pela chave imediatamente maior

EXEMPLO: EXCLUSÃO DA CHAVE 9



Substituir pela chave imediatamente maior

EXEMPLO: EXCLUSÃO DA CHAVE 40



Problema: o nó ficaria com menos de d chaves, o que não é permitido

SOLUÇÃO:

Concatenação ou Redistribuição

CONCATENAÇÃO

Duas páginas **P** e **Q** são **irmãs adjacentes** se têm o mesmo pai **W** e são apontadas por dois ponteiros adjacentes em **W**

P e **Q** podem ser concatenadas se:

- são **irmãs adjacentes**; e
- juntas possuem menos de **2d** chaves

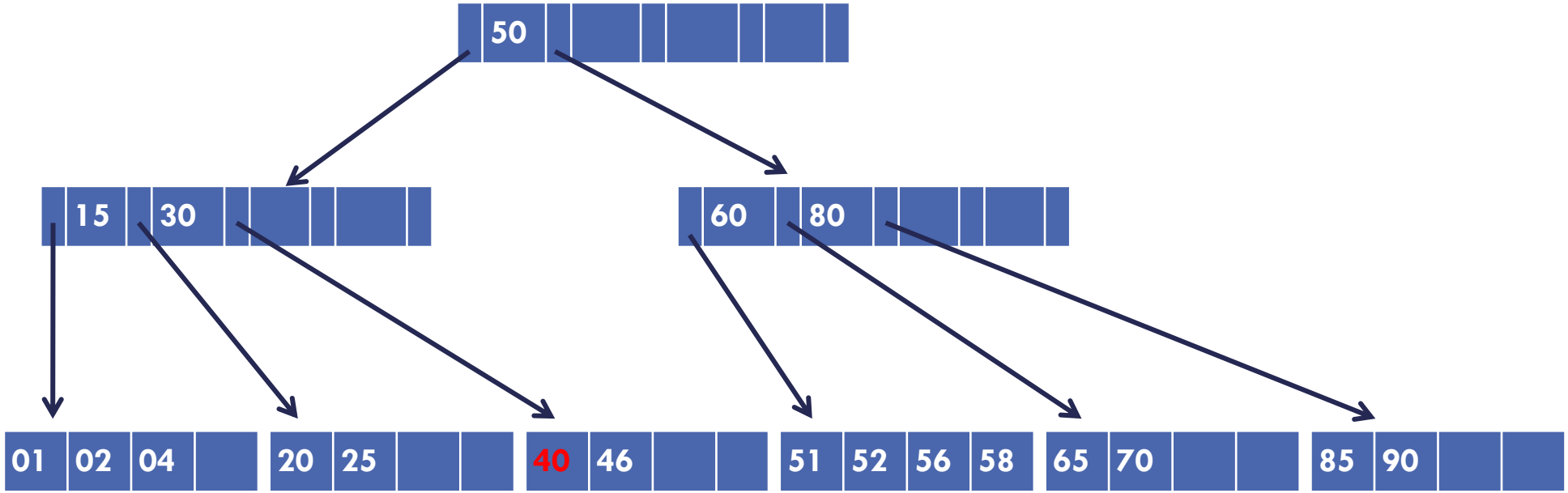
OPERAÇÃO DE CONCATENAÇÃO DE P E Q

Agrupar as entradas de **Q** em **P**

Em **W**, pegar a chave s_i que está entre os ponteiros que apontam para **P** e **Q**, e transferi-la para **P**

Em **W**, eliminar o ponteiro p_i (ponteiro que ficava junto à chave s_i que foi transferida)

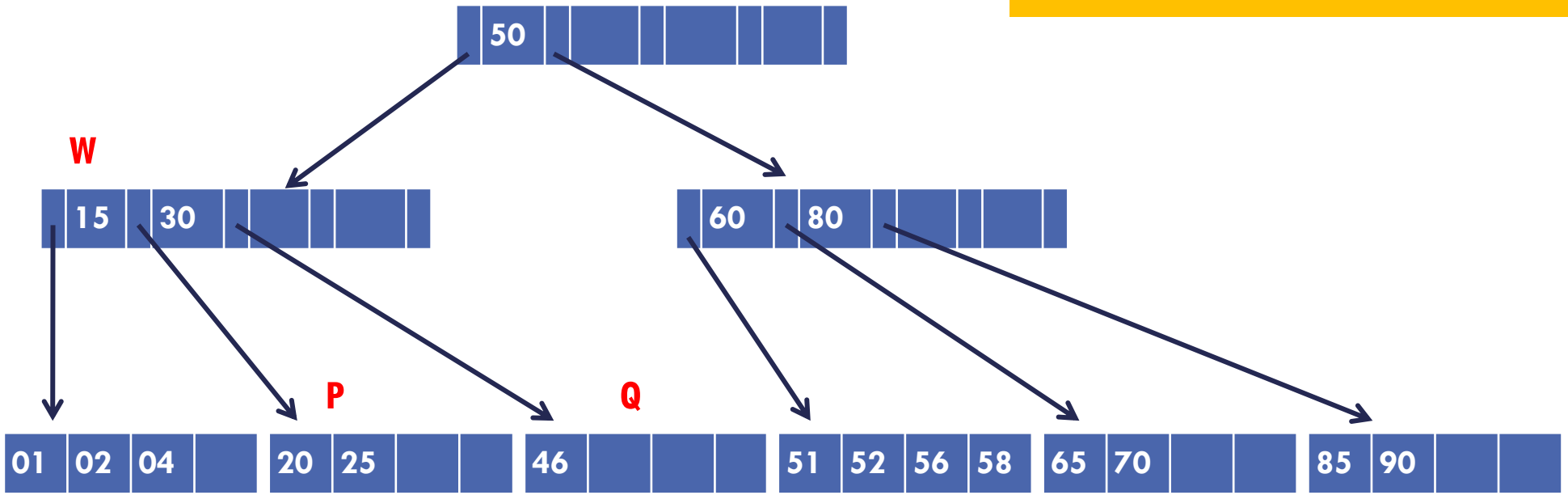
EXEMPLO: EXCLUSÃO DA CHAVE 40



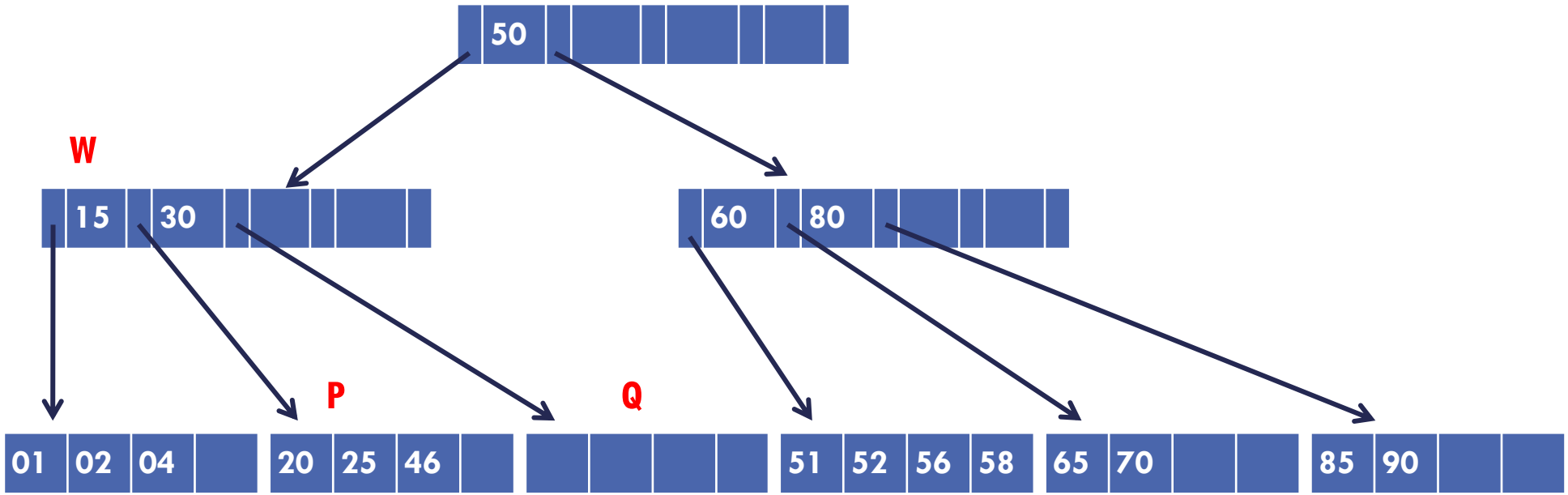
EXEMPLO: EXCLUSÃO DA CHAVE 40

Página Q ficou com menos de d chaves

Página P e Q são irmãs adjacentes
Soma de chaves de P e Q < 2d
CONCATENAR P e Q



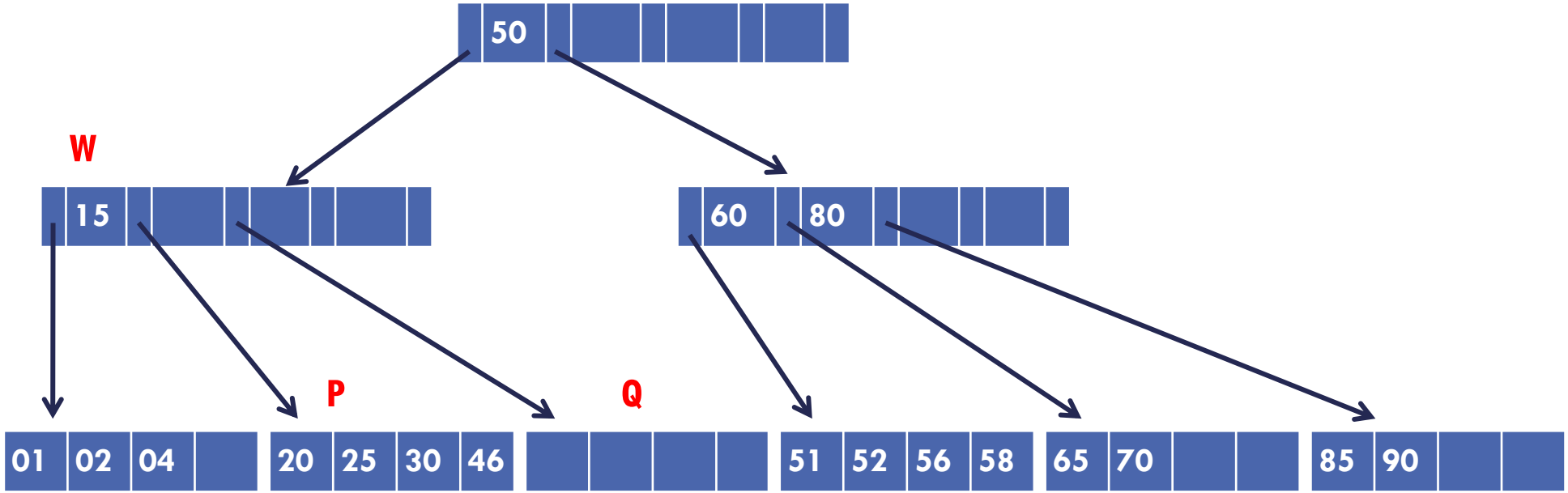
EXEMPLO: EXCLUSÃO DA CHAVE 40



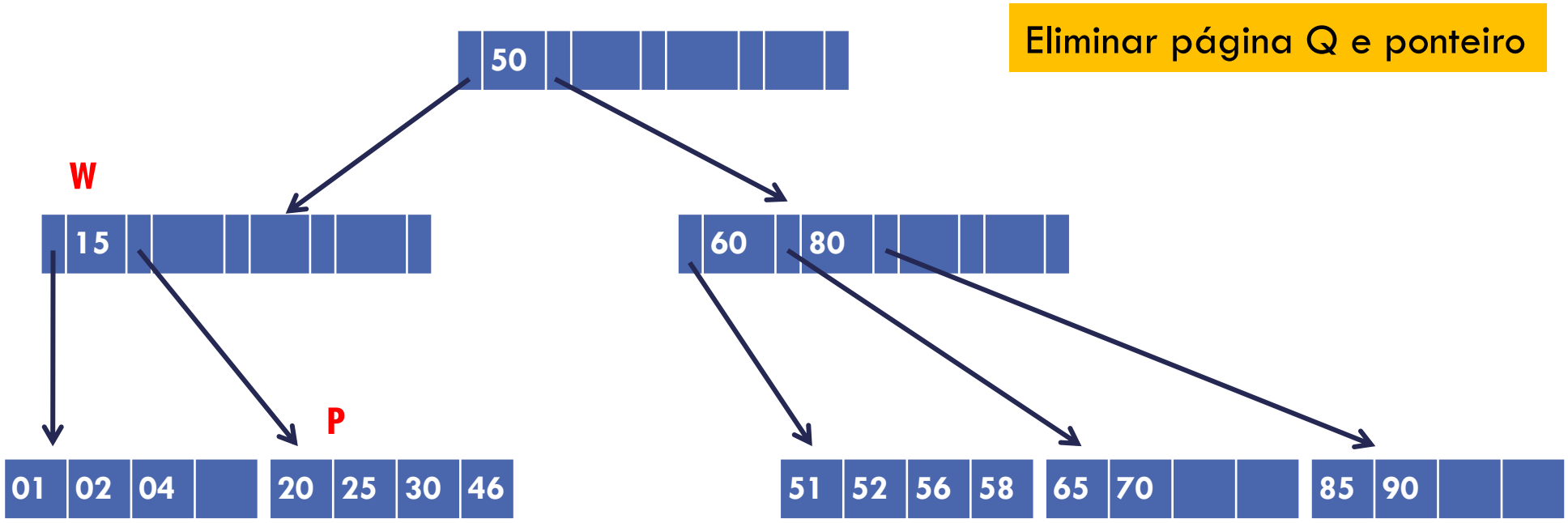
Transferir dados de Q para P

EXEMPLO: EXCLUSÃO DA CHAVE 40

Transferir chave que separa os ponteiros de P e Q em W para P

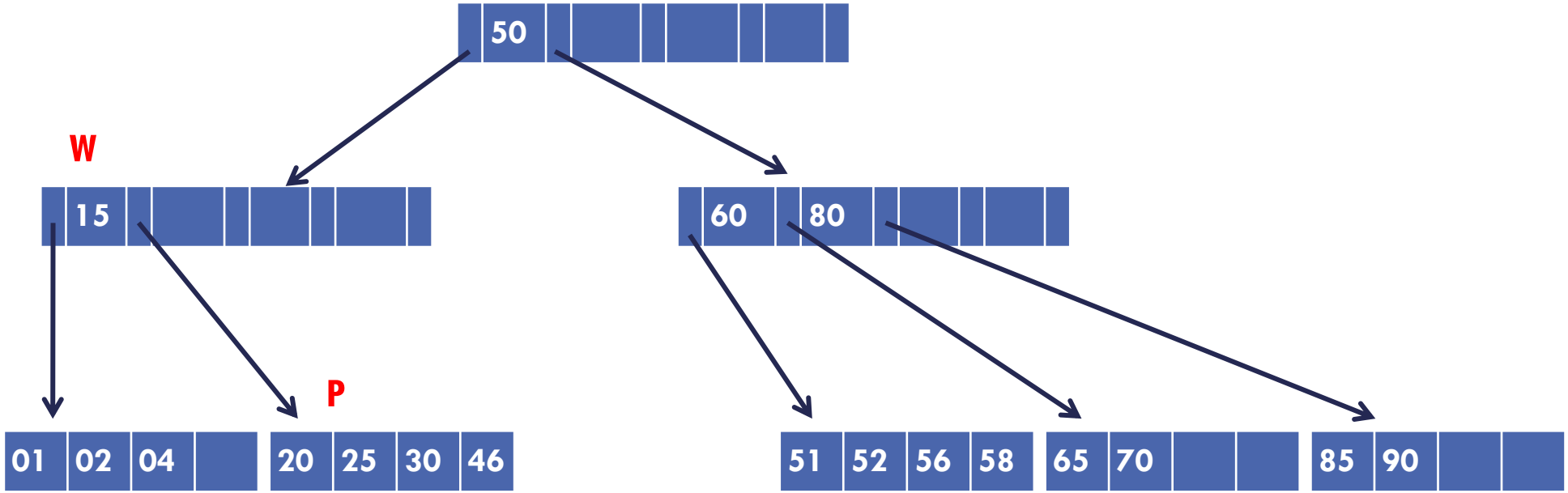


EXEMPLO: EXCLUSÃO DA CHAVE 40



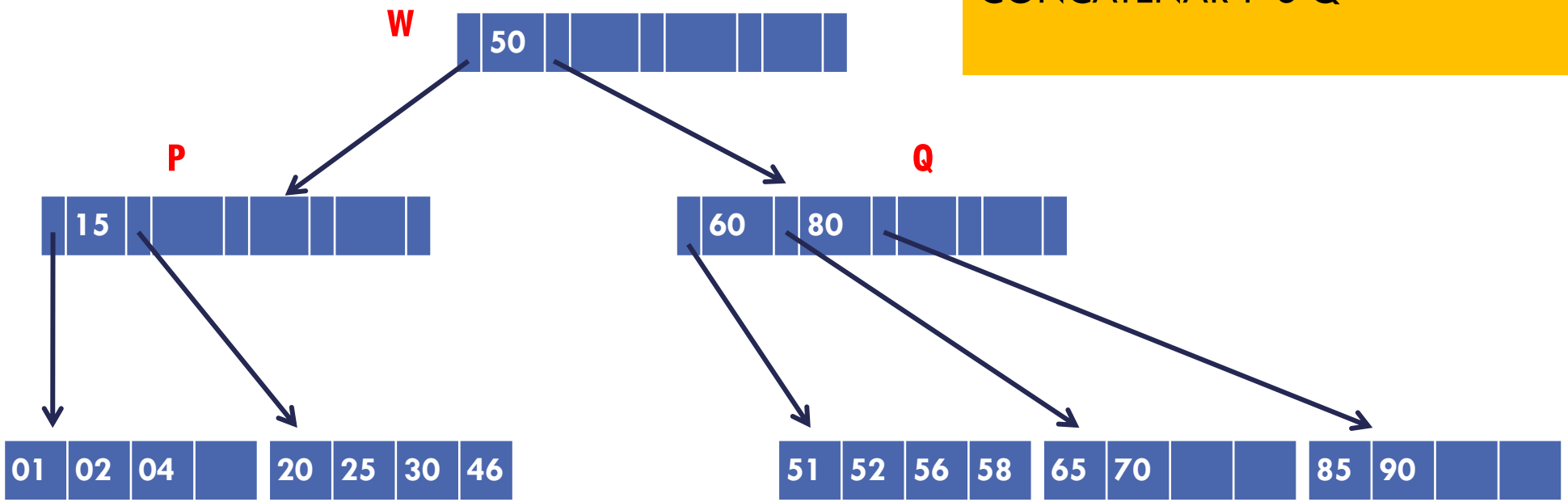
EXEMPLO: EXCLUSÃO DA CHAVE 40

Página W ficou com menos de d chaves
necessário propagar operação



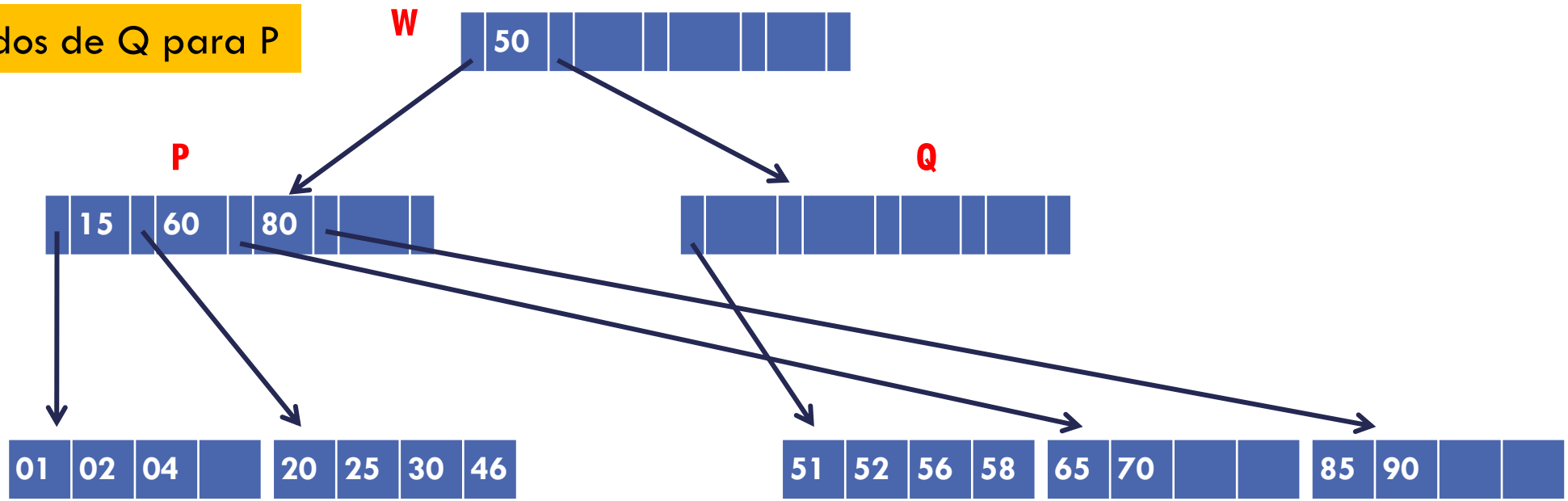
EXEMPLO: EXCLUSÃO DA CHAVE 40

Página P e Q são irmãs adjacentes
Soma de chaves de P e Q < 2d
CONCATENAR P e Q



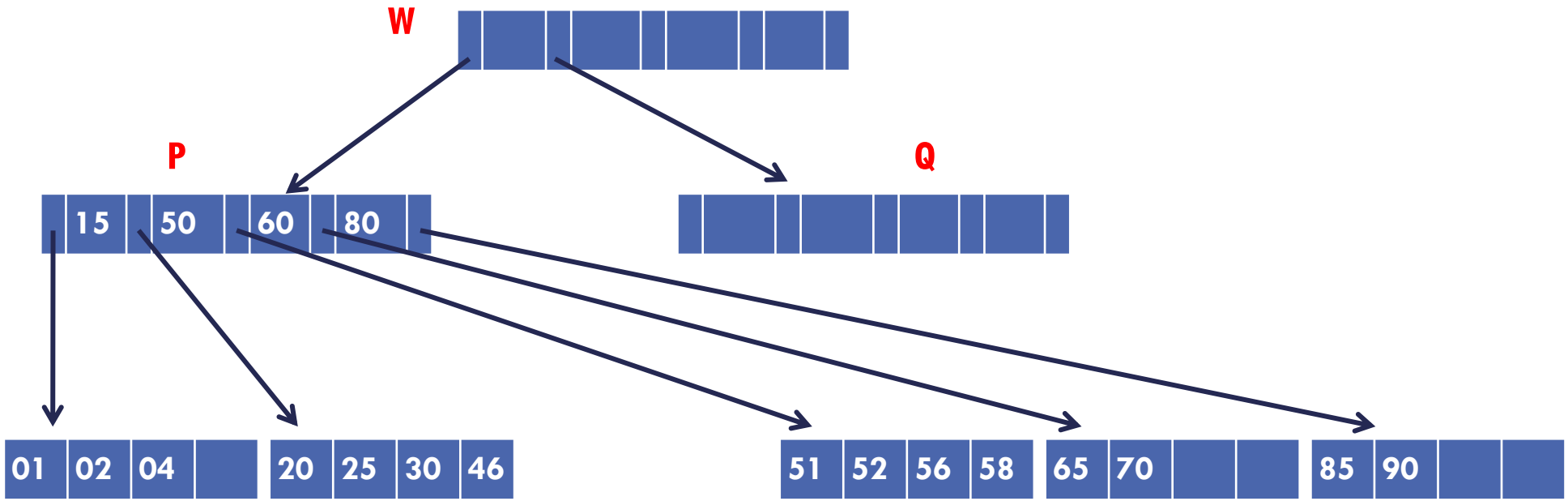
EXEMPLO: EXCLUSÃO DA CHAVE 40

Transferir dados de Q para P



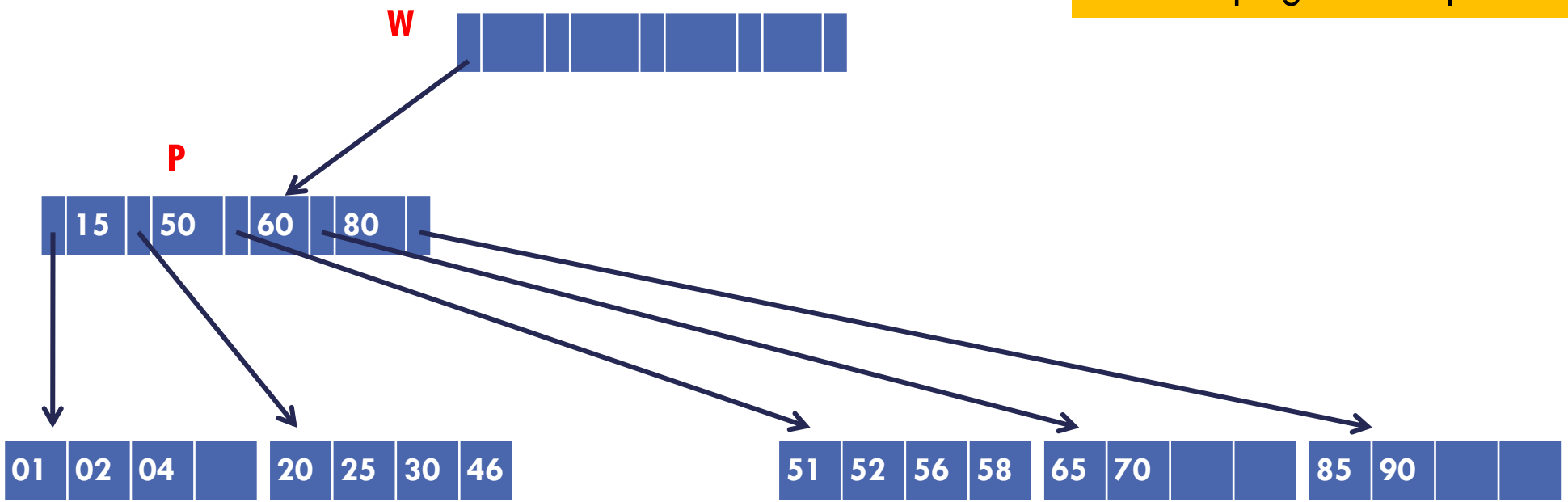
EXEMPLO: EXCLUSÃO DA CHAVE 40

Transferir chave que separa os ponteiros de P e Q em W para P



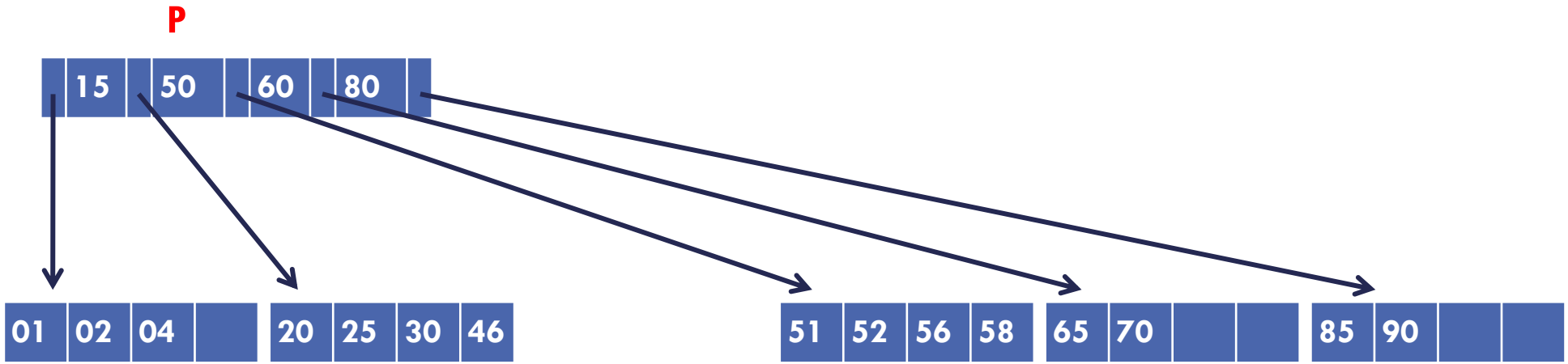
EXEMPLO: EXCLUSÃO DA CHAVE 40

Eliminar página Q e ponteiro



EXEMPLO: EXCLUSÃO DA CHAVE 40

W ficou vazia e era a raiz: eliminá-la
P passa a ser a nova raiz



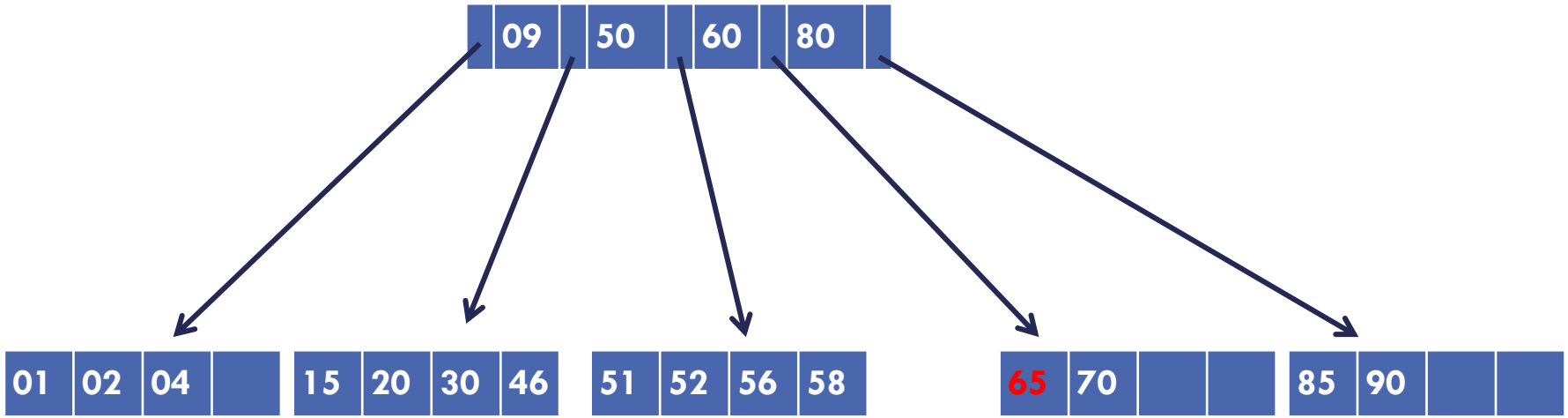
REDISTRIBUIÇÃO

Ocorre quando a soma das entradas de **P** e de seu irmão adjacente **Q** é maior ou igual a **2d**

Concatenar **P** e **Q**

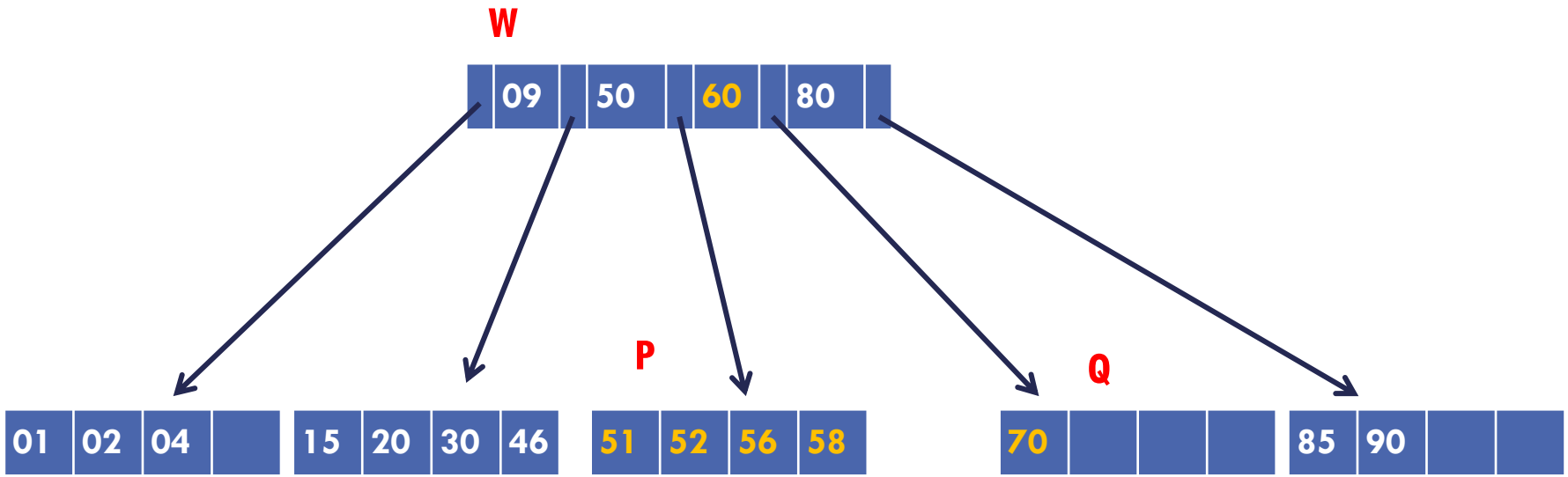
- Isso resulta em um nó **P** com mais de **2d** chaves, o que não é permitido
- Particionar o nó concatenado, usando **Q** como novo nó
- Essa operação não é propagável: o nó **W**, pai de **P** e **Q**, é alterado, mas seu número de chaves não é modificado

EXEMPLO: EXCLUSÃO DA CHAVE 65

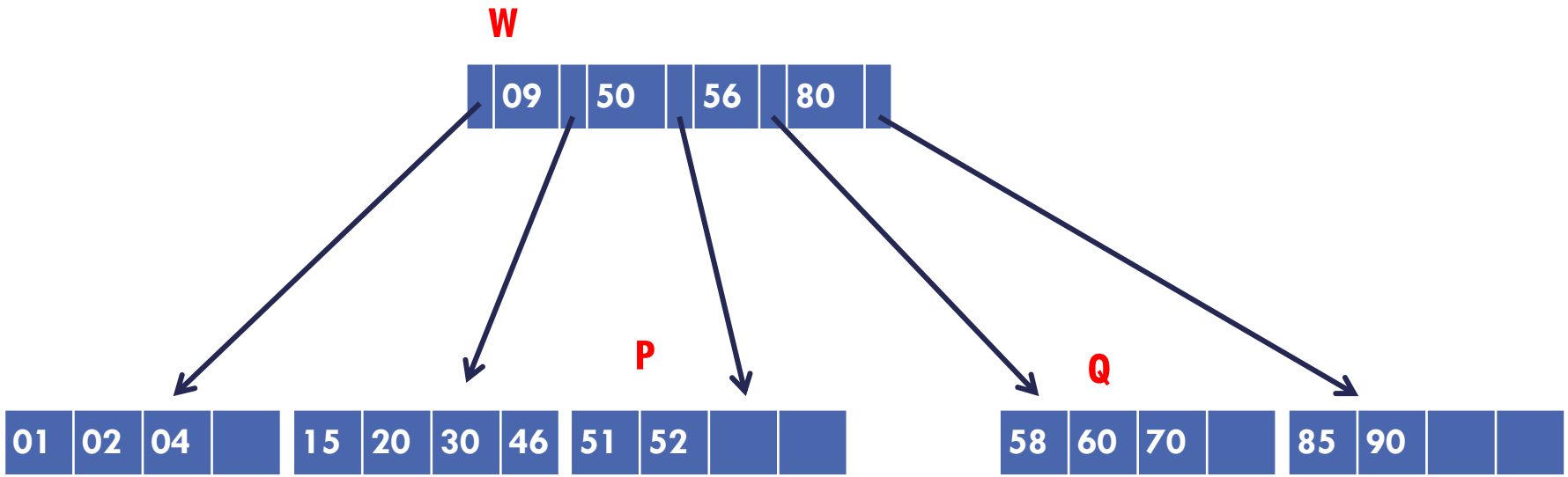


EXEMPLO: EXCLUSÃO DA CHAVE 65

Acomodar em P e Q as chaves:
51, 52, 56, 58, 60, 70
d chaves em P
chave **d+1** em W
Restante em Q



EXEMPLO: EXCLUSÃO DA CHAVE 65



E QUANDO AS DUAS ALTERNATIVAS SÃO POSSÍVEIS?

Quando for possível usar concatenação ou redistribuição (porque o nó possui 2 nós adjacentes, cada um levando a uma solução diferente), optar pela redistribuição

- Ela é menos custosa, pois não se propaga
- Ela evita que o nó fique cheio, deixando espaço para futuras inserções

EXERCÍCIO 2

Desenhar uma árvore B de ordem 3 que contenha as seguintes chaves: 8, 1, 6, 3, 14, 36, 32, 43, 39, 41, 38

Dica: começar com uma árvore B vazia e ir inserindo uma chave após a outra

Relembrando características de uma árvore B de ordem d

- A raiz é uma folha ou tem no mínimo 2 filhos
- Cada nó interno (não folha e não raiz) possui no mínimo d + 1 filhos
- Cada nó tem no máximo 2d + 1 filhos
- Todas as folhas estão no mesmo nível

EXERCÍCIO 3

Sobre a árvore resultante do exercício anterior, realizar as seguintes operações:

(a) Inserir as chaves 4, 5, 42, 2, 7

(b) Sobre o resultado do passo (a), excluir as chaves 14, 32

BUSCA DE UMA CHAVE X EM ÁRVORE B QUE INDEXA ARQUIVO EM DISCO

1. Inicie lendo a raiz da árvore a partir do disco
2. Procure **x** dentro do nó lido (pode ser usada busca binária, pois as chaves estão ordenadas dentro do nó)
   a) Se encontrou, encerra a busca;
   b) Caso contrário, continue a busca, lendo o filho correspondente, a partir do disco
3. Continue a busca até que **x** tenha sido encontrado ou que a busca tenha sido feita em uma folha da árvore (retorna o último nó pesquisado – nó onde a chave está ou deveria estar)

IMPLEMENTAÇÃO ÁRVORE B EM DISCO

Um arquivo para guardar metadados, que contém

- Um ponteiro para o nó raiz
- Um ponteiro para o próximo nó livre do arquivo

Um arquivo para guardar os dados, estruturado em nós (ou páginas/blocos)

IMPLEMENTAÇÃO ÁRVORE B EM DISCO

No arquivos de dados, cada nó possui

- Inteiro representando o número de chaves (**m**) armazenadas no nó
- Um ponteiro para o nó pai
- Array de m+1 ponteiros para os nós filho
- Array de m registros

CONSIDERAÇÕES SOBRE IMPLEMENTAÇÃO

Ponteiros são **números inteiros** que representam a posição no arquivo em disco onde o nó começa

- deve ser usado para fazer **fseek** antes de ler o nó do disco para a memória

A cada vez que for necessário manipular um nó:

- fazer fseek usando o valor do ponteiro que aponta para o nó desejado
- ler o nó todo para a memória, e manipulá-lo em memória
- depois, gravar o nó todo de volta no disco (caso ele tenha sido alterado)

EXEMPLO

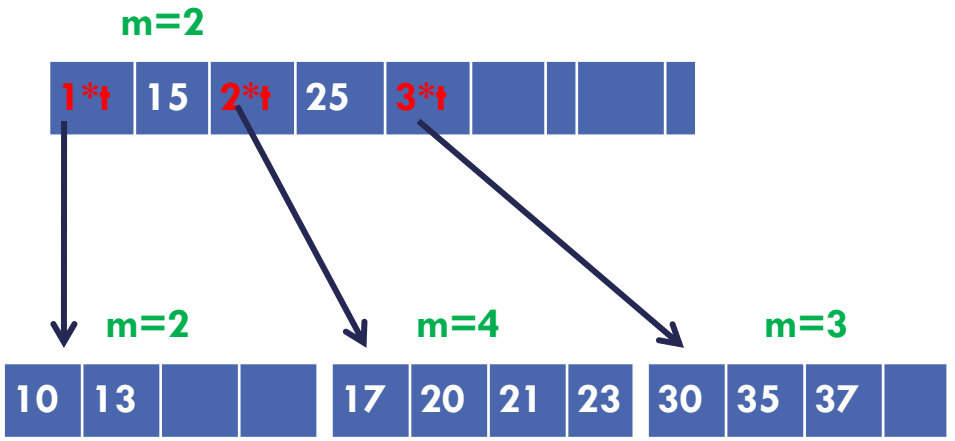
ordem d = 2

Valores em branco: chave do registro

Valores em vermelho: valor do ponteiro

t é o tamanho do nó no Arquivo de dados

Os demais dados do registro não estão representados na figura, para simplificar (apenas as chaves estão representadas)



REFERÊNCIA

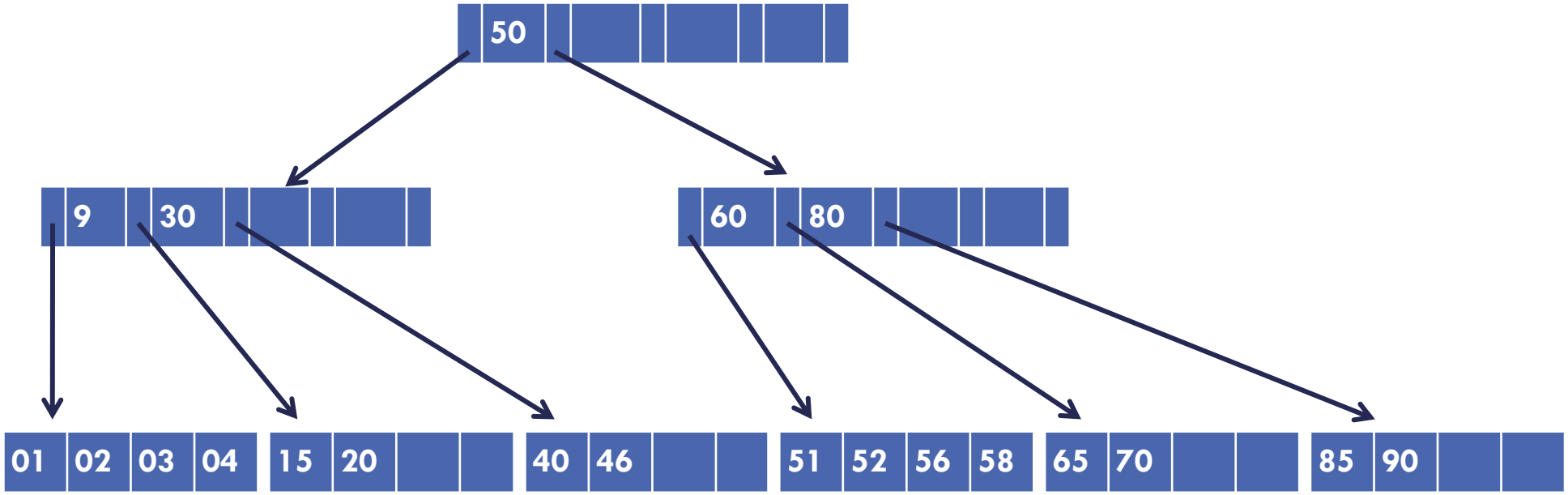
Szwarcfiter, J.; Markezon, L. Estruturas de Dados e seus Algoritmos, 3a. ed. LTC. Cap. 5

AGRADECIMENTOS

Exemplo cedido por Renata Galante

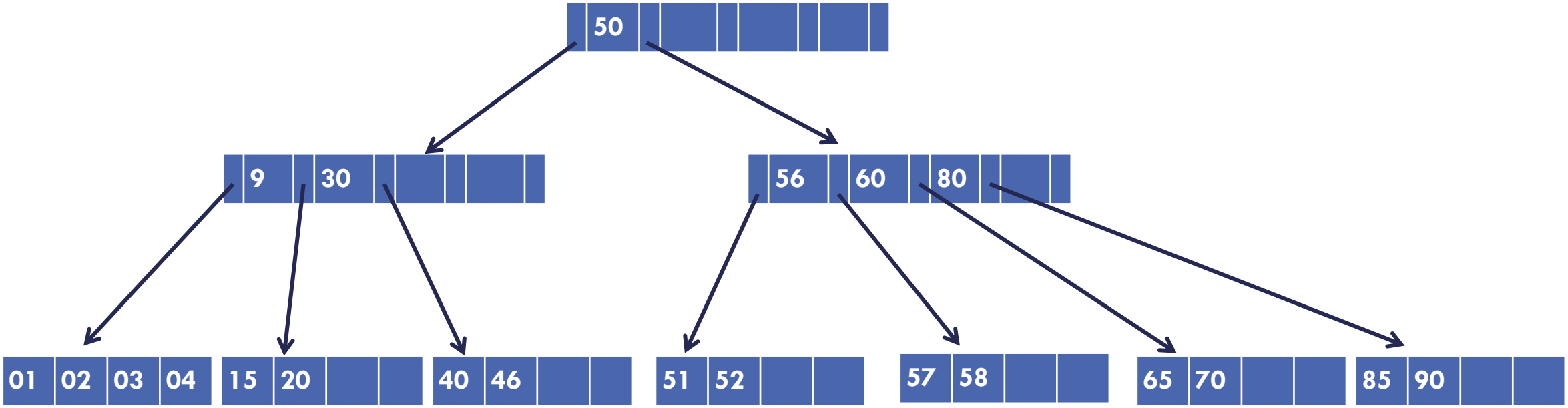
RESPOSTAS DOS EXERCÍCIOS

EXERCÍCIO 1:
INSERIR CHAVES 57, 71, 72, 73

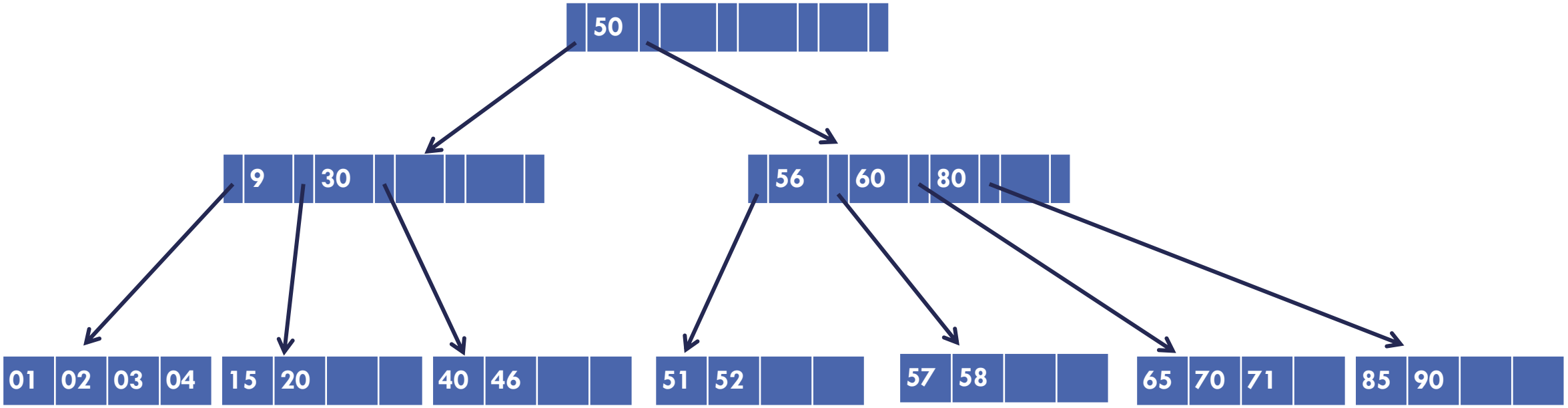


Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura.
Na prática, todos apontam para NULL

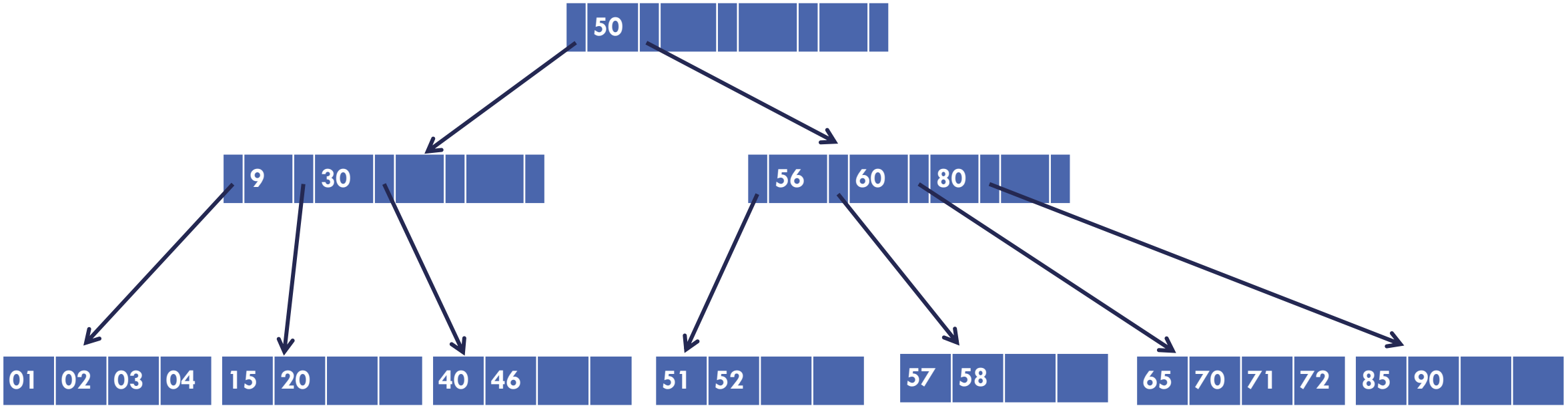
INSERÇÃO DE 57



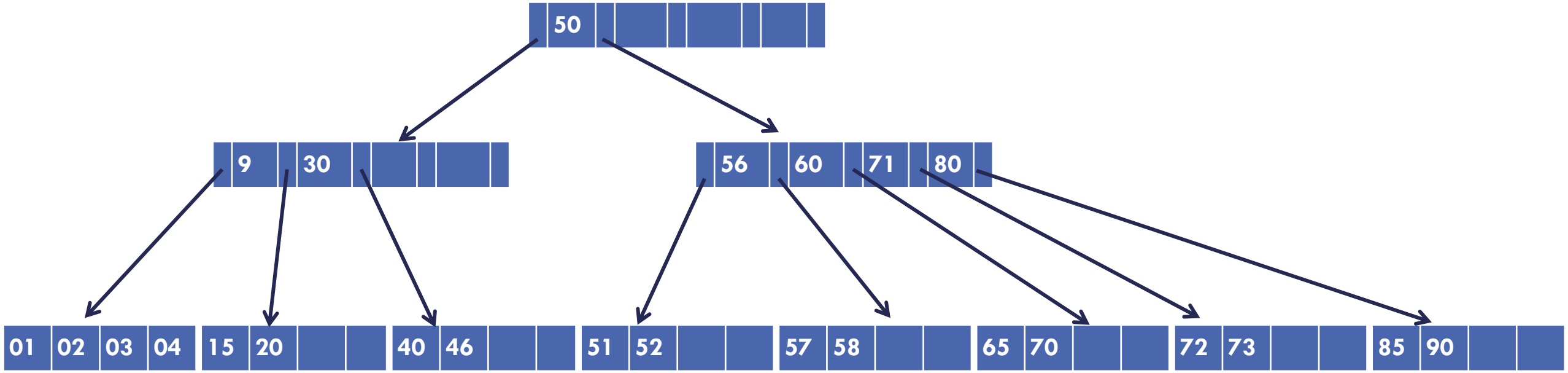
INSERÇÃO DE 71



INSERÇÃO DE 72



INSERÇÃO DE 73



EXERCÍCIO 2

Desenhar uma árvore B de ordem 3 que contenha as seguintes chaves: 8, 1, 6, 3, 14, 36, 32, 43, 39, 41, 38

Dica: começar com uma árvore B vazia e ir inserindo uma chave após a outra

Relembrando características de uma árvore B de ordem d

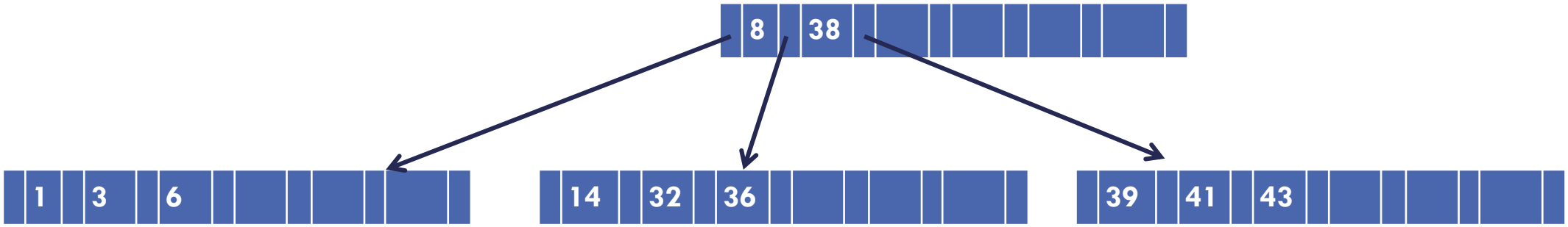
- A raiz é uma folha ou tem no mínimo 2 filhos
- Cada nó interno (não folha e não raiz) possui no mínimo d + 1 filhos
- Cada nó tem no máximo 2d + 1 filhos
- Todas as folhas estão no mesmo nível

RESPOSTA

Desenhar uma árvore B de ordem 3 que contenha as seguintes chaves: 1, 3, 6, 8, 14, 32, 36, 38, 39, 41, 43

Como d = 3:

- Cada nó tem no máximo 6 chaves
- Cada nó tem no máximo 7 filhos



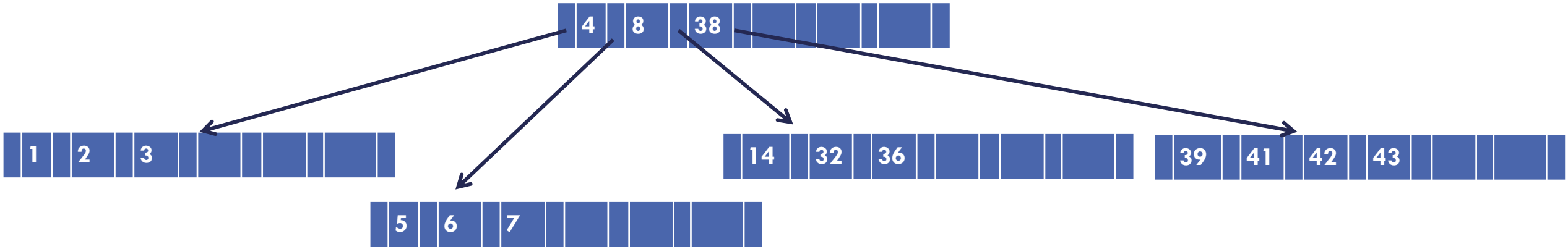
EXERCÍCIO 3

Sobre a árvore resultante do exercício anterior, realizar as seguintes operações:

(a) Inserir as chaves 4, 5, 42, 2, 7

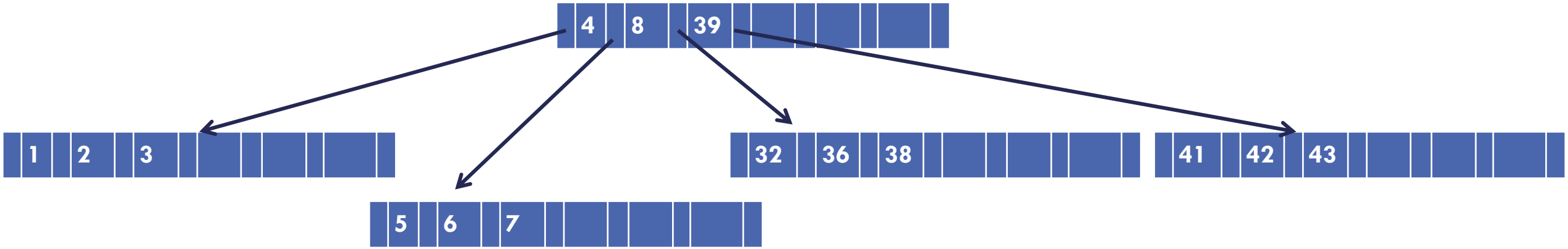
(b) Sobre o resultado do passo (a), excluir as chaves 14, 32

RESPOSTA (A) – INSERÇÃO DE 4, 5, 42, 2, 7



RESPOSTA (B) – EXCLUSÃO DE 14

É possível fazer redistribuição



RESPOSTA (B) – EXCLUSÃO DE 32

É necessário fazer concatenação