

Arquivos Indexados

Vanessa Braganholo

Arquivos de Acesso Direto

- ▶ Basicamente, duas formas de acesso a um registro:
 - ▶ 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?

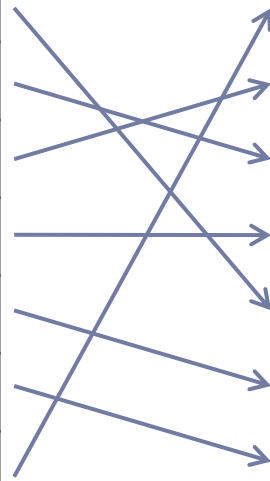
Exemplo de Índice Plano

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

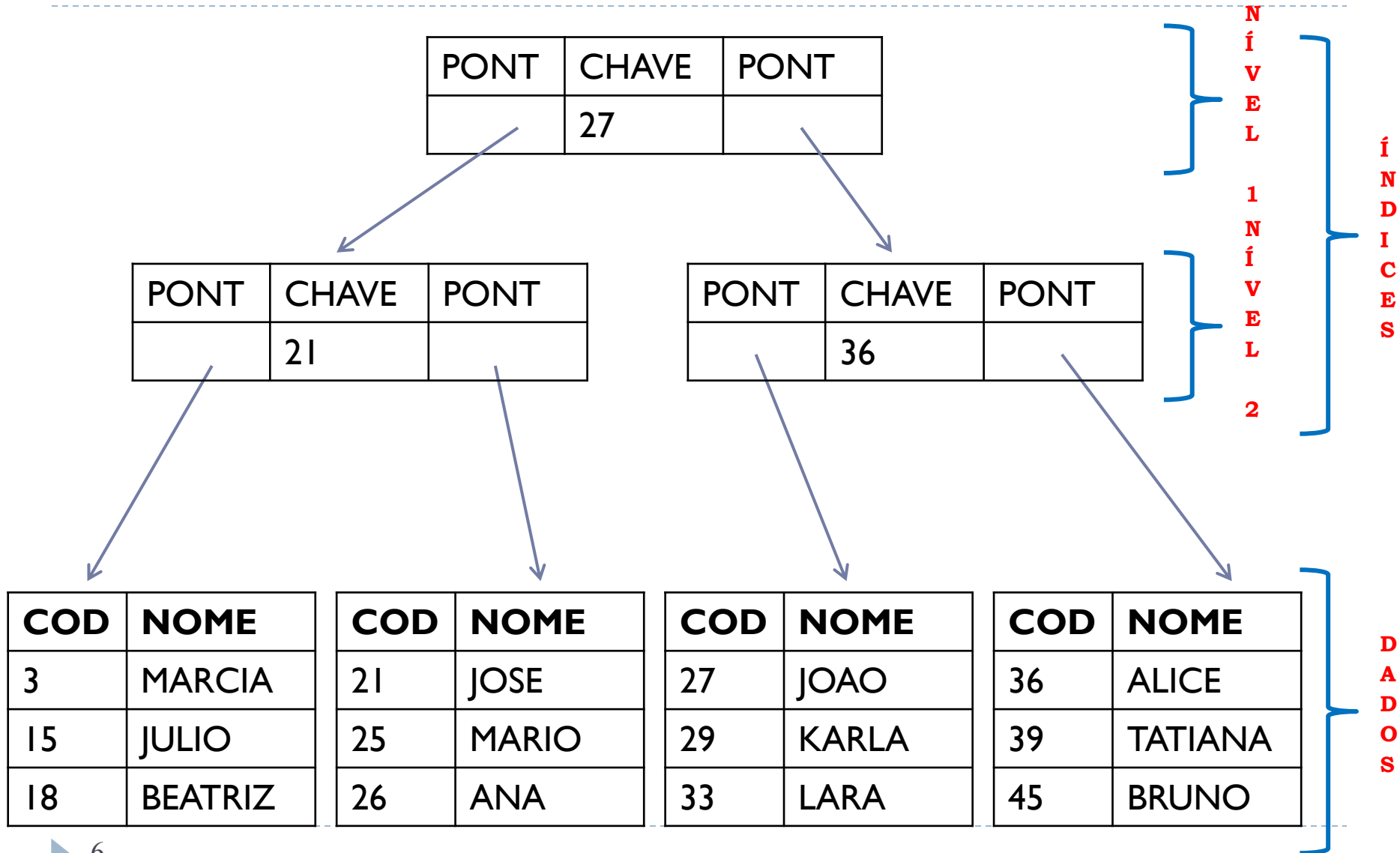
	COD	NOME
0	23	JOSE
1	10	MARIO
2	5	ANA
3	15	MARCIA
4	3	JULIO
5	16	BEATRIZ
6	21	CAMILA



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
- ▶ 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



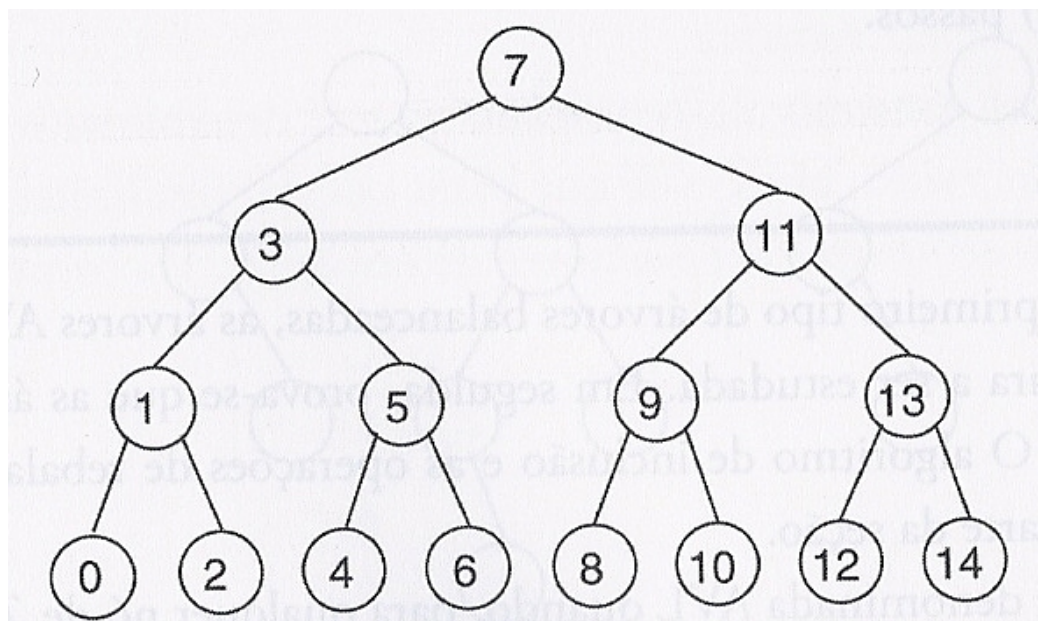
Hierarquia lembra árvore...

- ▶ A maioria das estruturas de índice é implementada por árvores de busca
 - ▶ Árvores Binárias
 - ▶ Árvores AVL
 - ▶ Árvores de Múltiplos Caminhos

Árvore de Busca Binária

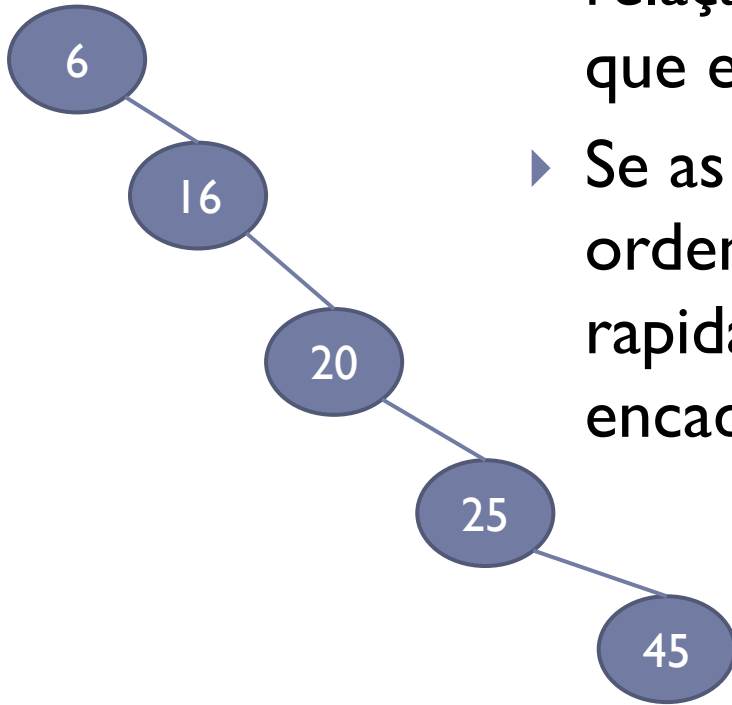
▶ Características de uma árvore de busca binária T

- ▶ todas as chaves da subárvore da esquerda de T têm valores menores que a chave do nó raiz de T
- ▶ todas as chaves da subárvore da direita de T têm valores maiores que a chave do nó raiz de T
- ▶ as subárvores esquerda e direita de T também são árvores de busca binária



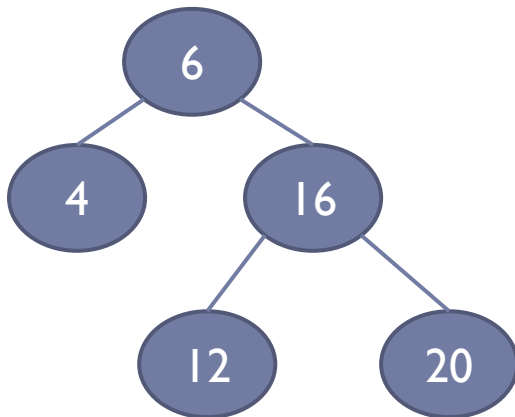
Considerações sobre Árvores Binárias

- ▶ 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

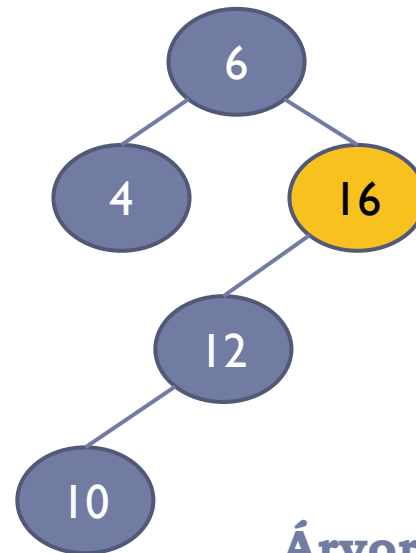


Árvores AVL

- ▶ São árvores binárias balanceadas
 - ▶ Para qualquer nó da árvore, a altura da subárvore da esquerda não pode diferir em mais de 1 unidade da altura da subárvore da direita



Árvore AVL



Árvore Não-AVL

Considerações sobre Árvores AVL

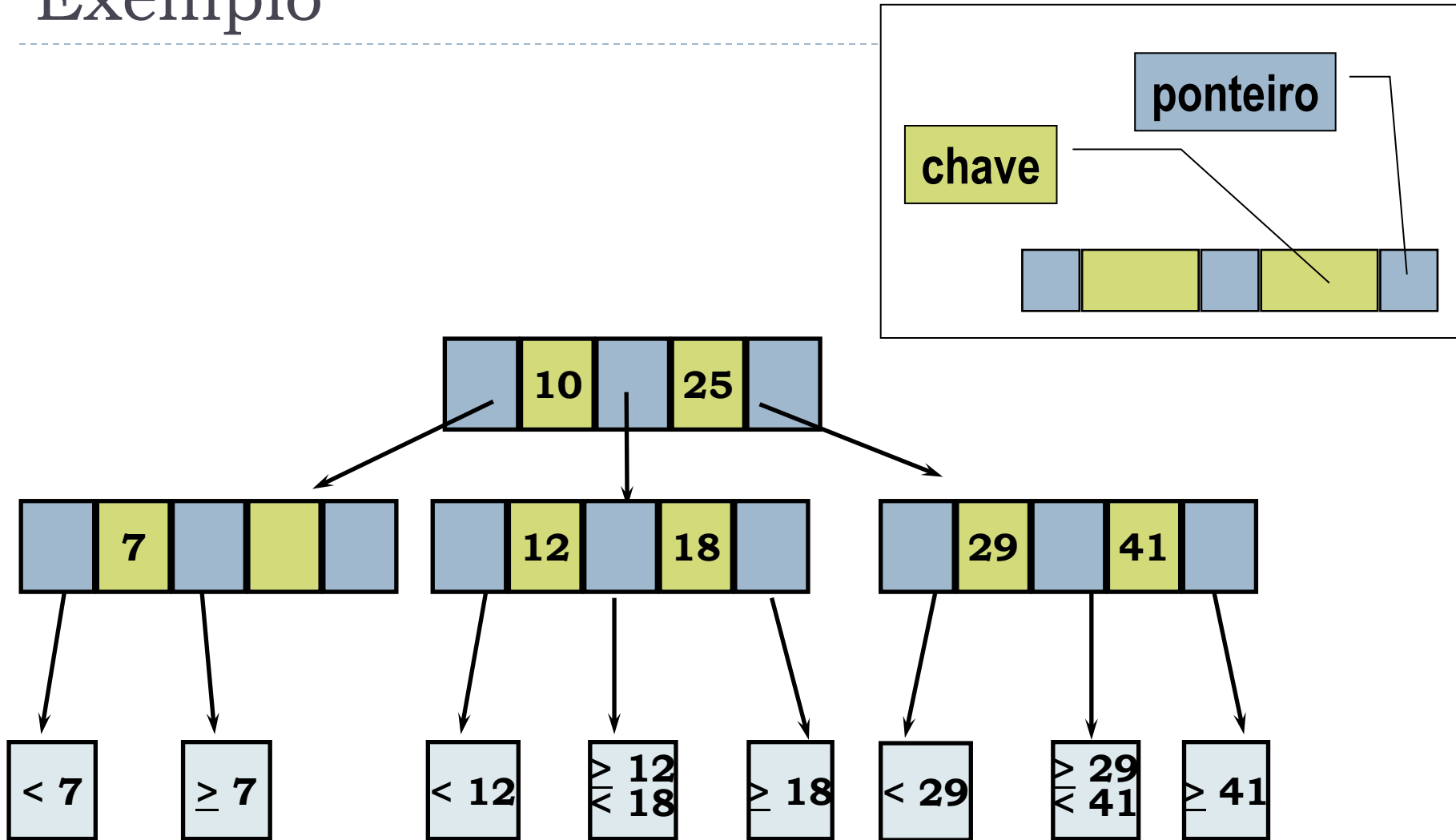
- ▶ Ainda são excessivamente altas para uso eficiente como estrutura de índice

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



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

Exemplos de Árvores Múltiplos Caminhos

- ▶ **Árvore B**
- ▶ **Árvore B***
- ▶ **Árvore B+**
- ▶ **Tries**

Árvores 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

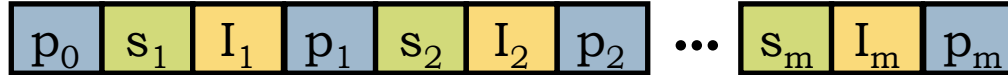
Árvore B

▶ Outras propriedades

- ▶ 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 l e $2d$ chaves
- ▶ Em cada página, as chaves estão ordenadas: s_1, \dots, s_m , onde $d \leq m \leq 2d$, exceto para a raiz onde $l \leq m \leq 2d$
- ▶ P contém $m+1$ ponteiros p_0, p_1, \dots, 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 (I_k) relativos àquela chave

Árvore B

Estrutura de uma página (nó)



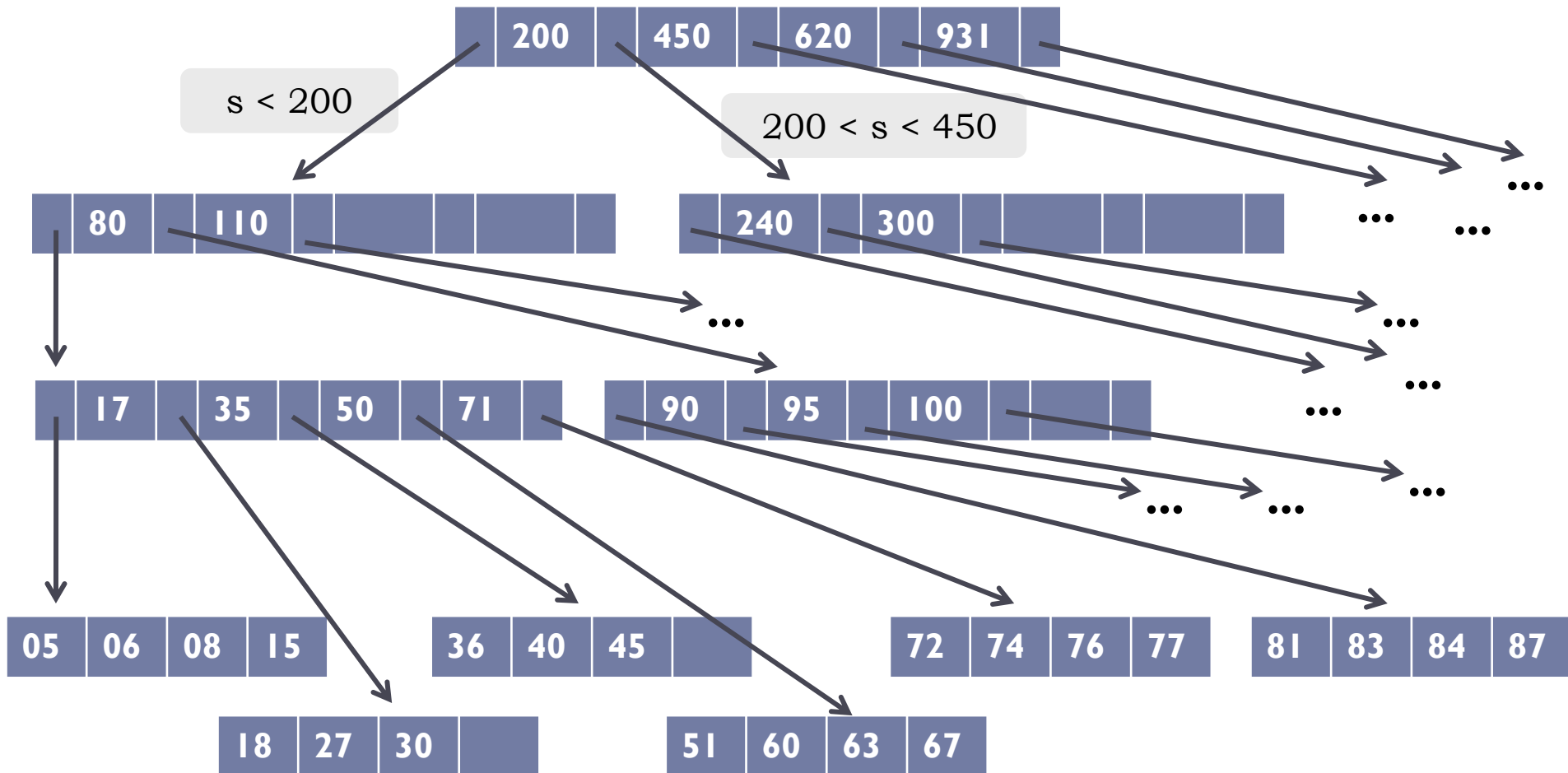
- ▶ Seja uma página **P** com **m** chaves:
 - ▶ para qualquer chave **y** pertencente à primeira página apontada por **P** (ou seja, apontada por p_0), $y < s_1$
 - ▶ para qualquer chave **y** pertencente à página apontada por p_k , $1 \leq k \leq m-1$, $s_k < y < s_{k+1}$
 - ▶ para qualquer chave **y**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

Busca de uma chave x em Árvore B

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

Árvore B: Exemplo

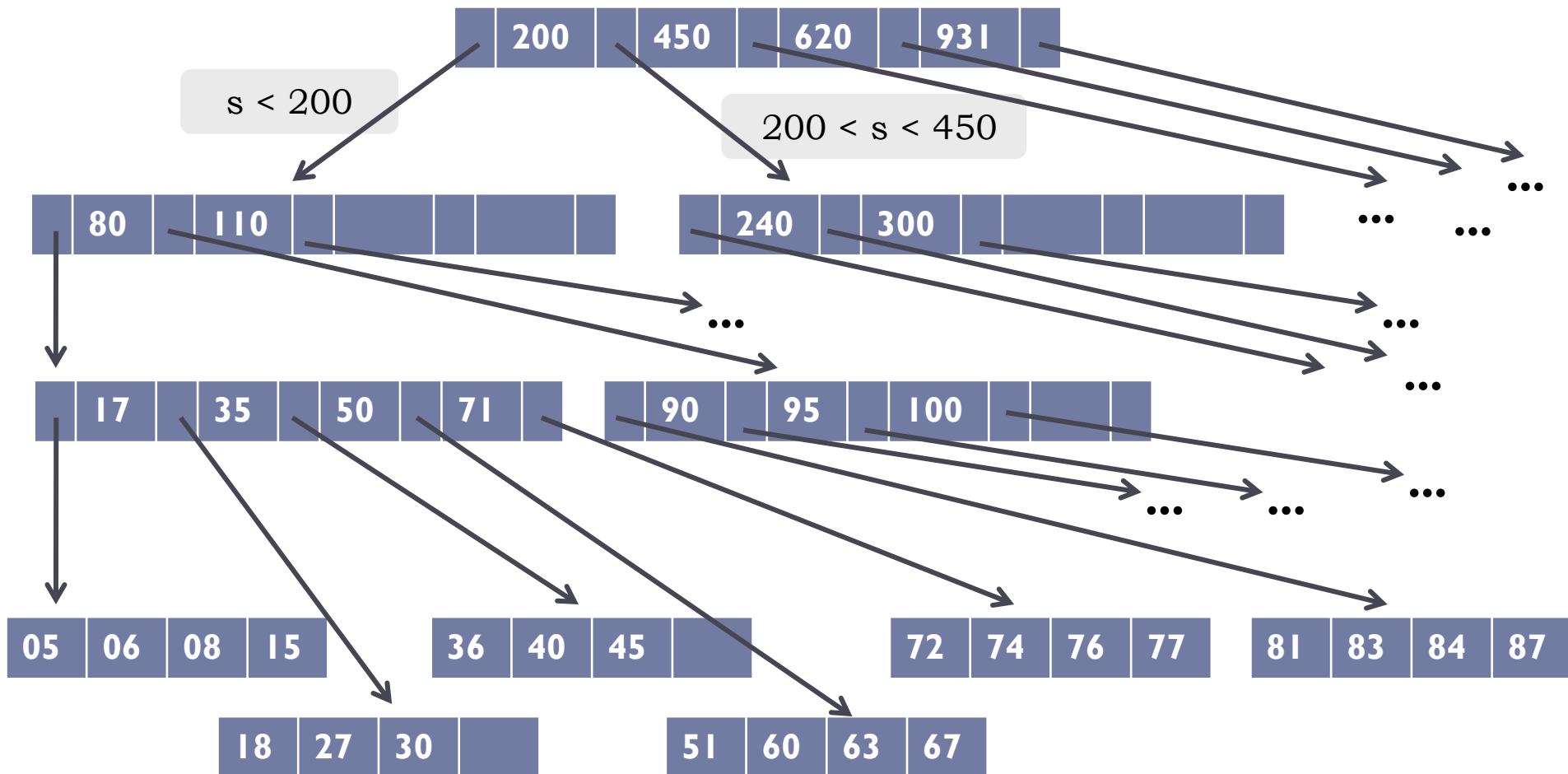
ordem $d = 2$



Buscar chaves 240, 76 e 85 na árvore

ordem $d = 2$

Árvore B: Exemplo



Algoritmo de Busca em Árvore B

▶ Funcionamento do algoritmo

▶ Caso a chave x seja encontrada:

▶ **encontrou** = 1

▶ **pt** aponta para a página que contém a chave

▶ **pos** aponta para a posição em que a chave se encontra dentro da página

▶ Caso a chave x não seja encontrada:

▶ **encontrou** = 0

▶ **pt** aponta para a última página examinada

▶ **pos** informa a posição, nessa página, onde a chave deveria estar inserida

Algoritmo de Busca em Árvore B

```
procedimento buscaB(x, pt, encontrou, pos)
p:= ptr Luiz; pt:= λ; encontrou := 0;
enquanto p ≠ λ faça
início
    i:= 1; pos:= 1; pt:= p
    enquanto i ≤ m faça % m é o número de chaves que a página p contém
início
    se x > p↑.s[i] então i:= i+1; pos:= i + 1
    senão início
        se x = p↑.s[i] então
            p:= λ; encontrou := 1 % chave encontrada
        senão p := p↑.pont[i-1] % mudança de página
        i:= m + 2
    fim
fim {enquanto}
se i = m + 1 então p:= p↑.pont[m]
fim {enquanto}
```


Alternativa Recursiva: Algoritmo de Busca em Árvore B

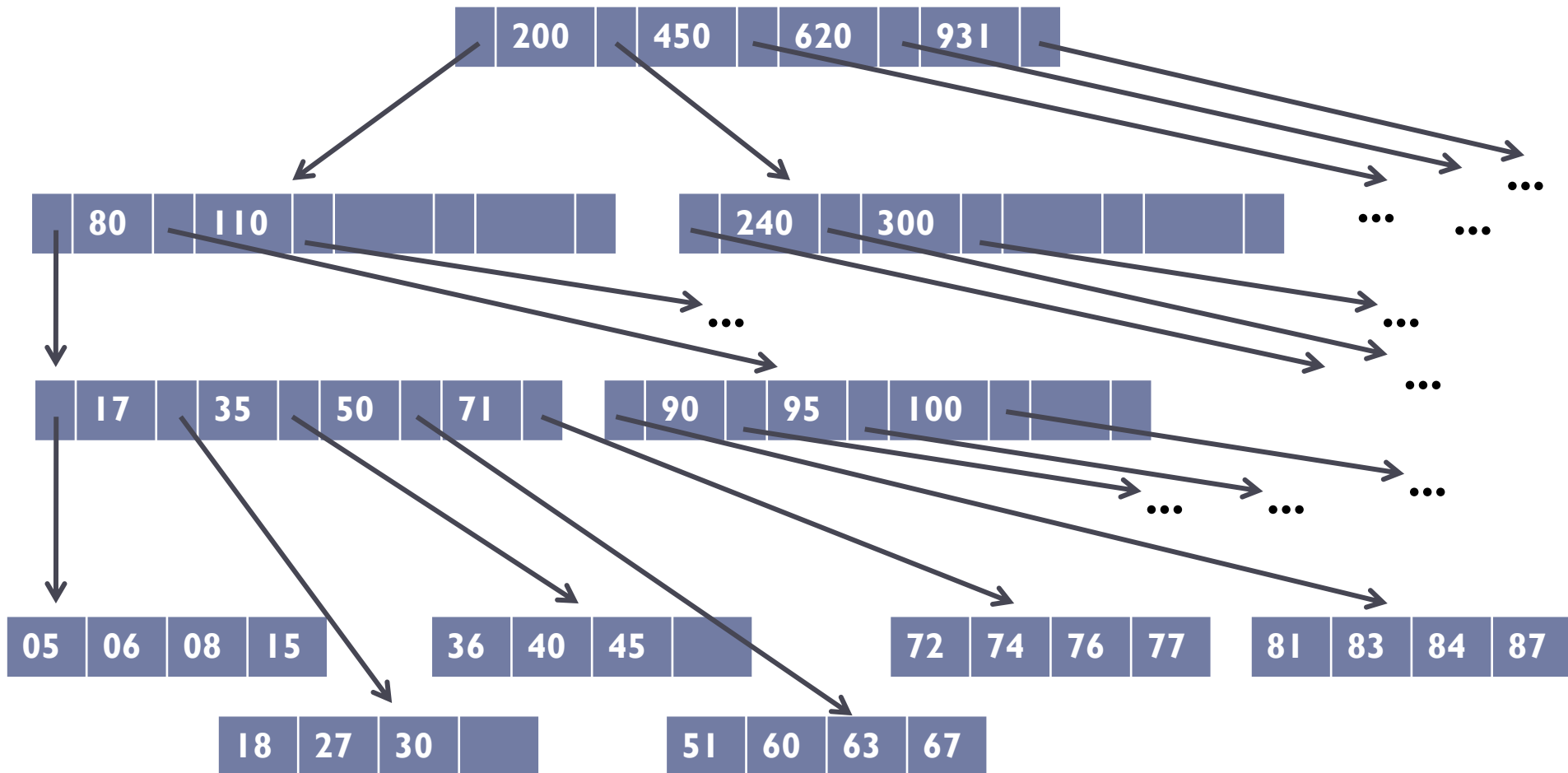
```
proc pesquisa_árvore_B (pagina, x );
  { pagina : nodo da árvore B }
  { x: valor de chave procurado }
begin
  i ← 1;
  % m é o número de chaves armazenadas na página
  while (i ≤ m[pagina] and x < si[pagina])      % pesq. sequencial no
nodo
    do i ← i + 1;
  if (i ≤ m[pagina] and x = si[pagina]) then
    return (pagina, i);                          % retorna nodo e
ordem da chave
  if (pagina é folha) then
    return nil;                                  % não encontrou
  else
    begin
      DISK-READ(pi[pagina]); % lê nodo filho do disco e prossegue
      return pesquisa_árvore_B (pi[pagina], x );
    end;
end;
```

Inserção

- ▶ Para inserir um registro de chave **x** na árvore **B**
 - ▶ Executar o algoritmo buscaB
 - ▶ Se a chave for encontrada, a inserção é inválida
 - ▶ Se a chave não for encontrada:
 - ▶ Inserir a chave na posição **pos** da folha apontada por **pt**

Árvore B: Inserção

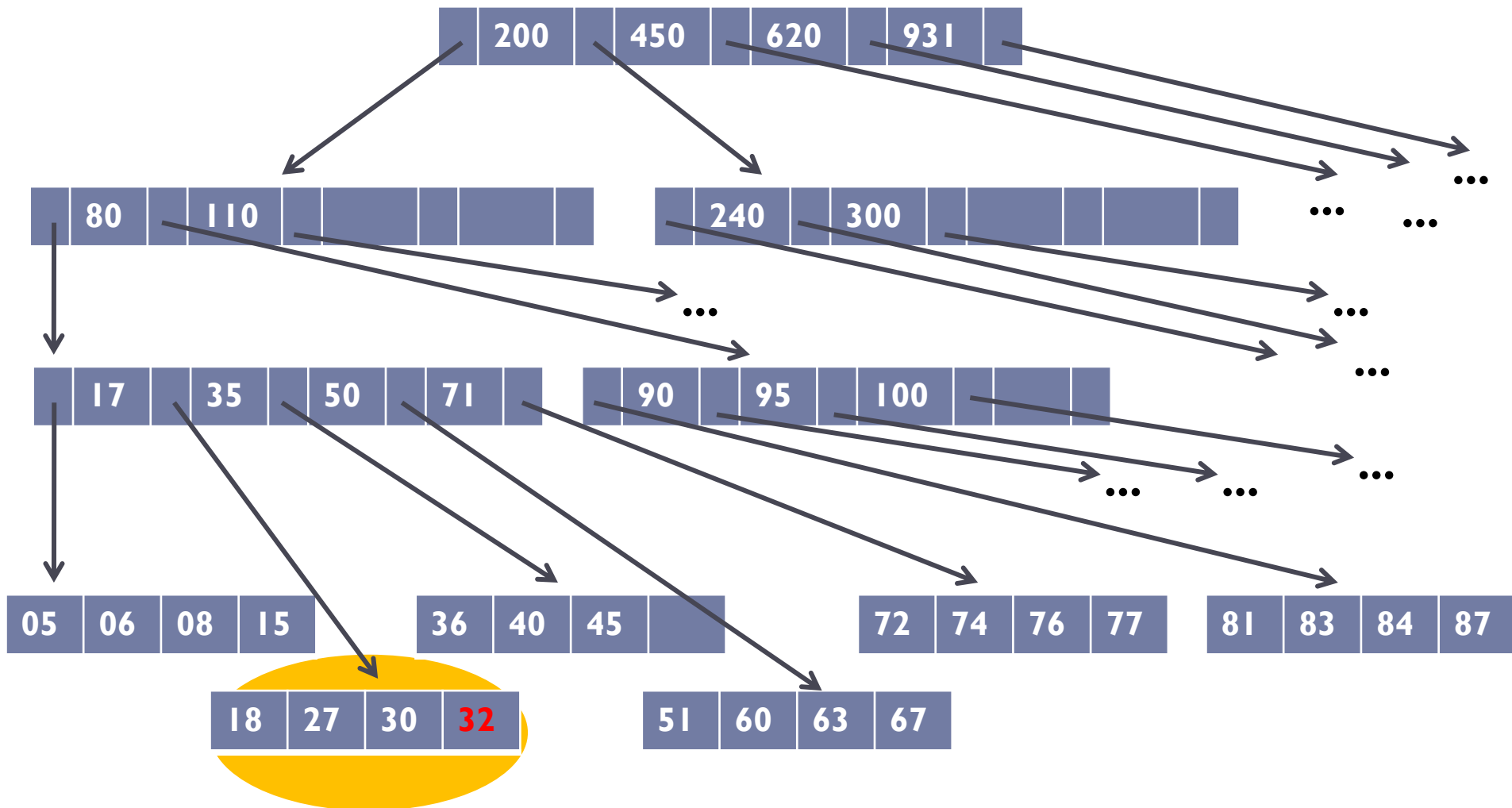
Inserir chave 32



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

Árvore B: Inserção

Inserir chave 32

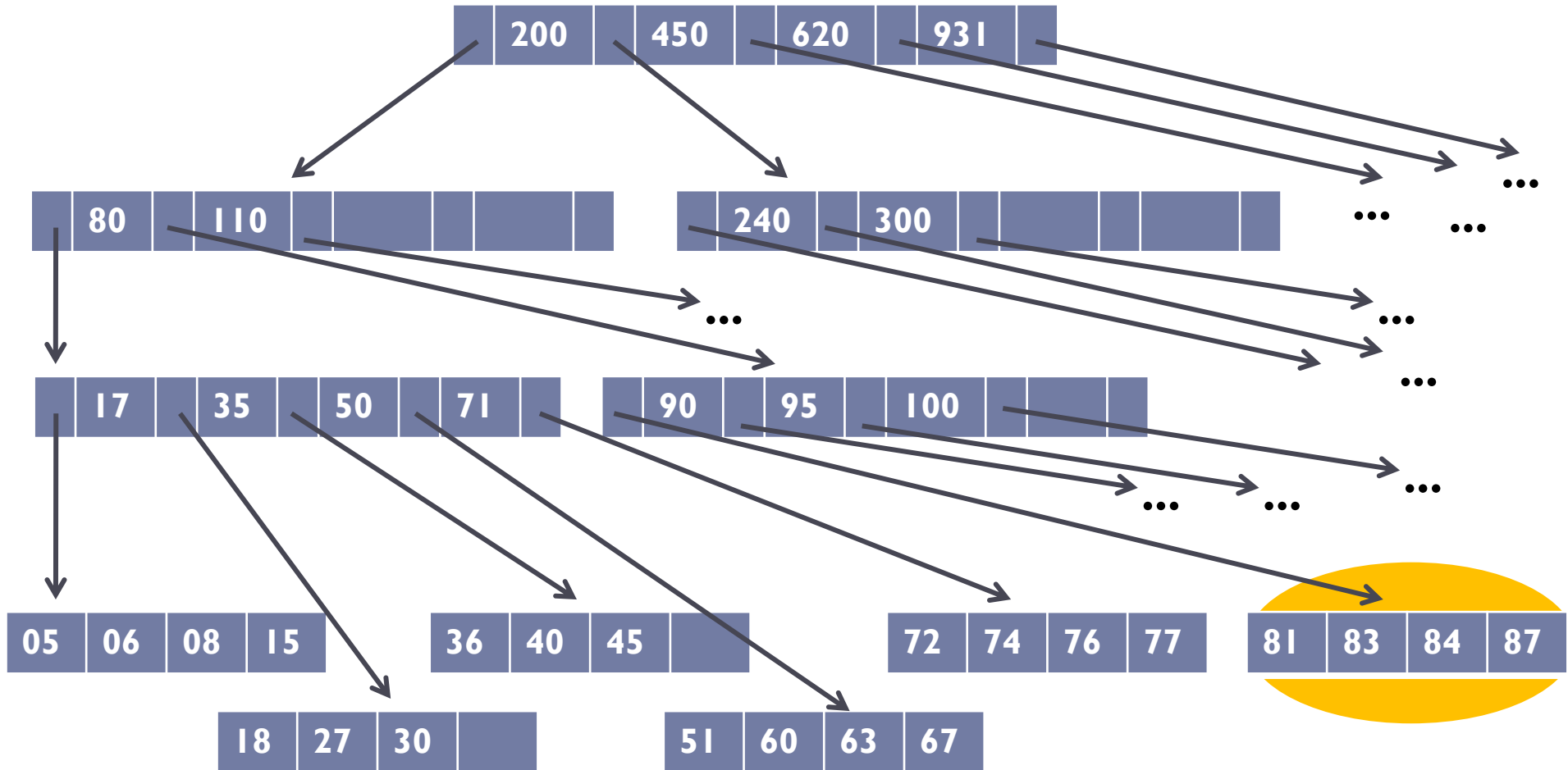


Inserir chave 85

Inserção faria página ficar com $2d+1$ chaves

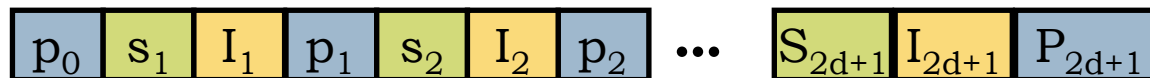
Problema: página cheia

ordem $d = 2$



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)
 - ▶ $p_0, (s_1, p_1), (s_2, p_2), \dots, (s_d, p_d), (s_{d+1}, p_{d+1}), \dots, (s_{2d+1}, p_{2d+1})$
 - ▶ graficamente:



Solução

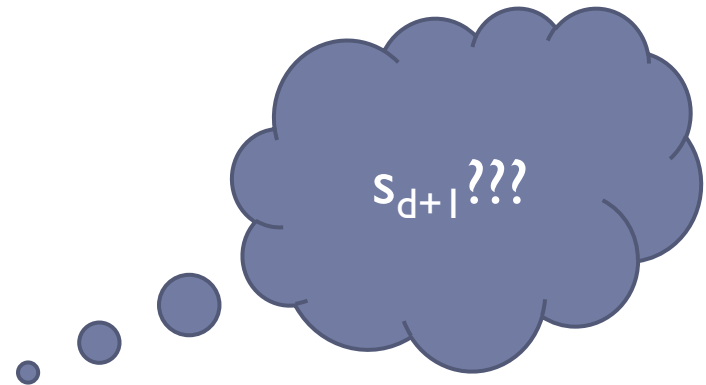
- ▶ Particionar a página em 2
 - ▶ Na página **P** permanecem **d** entradas
 - ▶ Alocar outra página, **Q**, e nela alocar as outras **d+1** entradas

- ▶ Após o particionamento
 - ▶ Estrutura da página P:
 - ▶ $p_0, (s_1, p_1), (s_2, p_2), \dots, (s_d, p_d)$

 - ▶ Estrutura da página Q:
 - ▶ $p_{d+1}, (s_{d+2}, p_{d+2}) \dots, (s_{2d+1}, s_{2d+1})$

Solução

- ▶ Particionar a página em 2
 - ▶ Na página **P** permanecem **d** entradas
 - ▶ Alocar outra página, **Q**, e nela alocar as outras **d+1** entradas
- ▶ Após o particionamento
 - ▶ Estrutura da página P:
 - ▶ $P_0, (s_1, P_1), (s_2, P_2), \dots, (s_d, P_d)$
 - ▶ Estrutura da página Q:
 - ▶ $P_{d+1}, (s_{d+2}, P_{d+2}) \dots, (s_{2d+1}, P_{2d+1})$



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 buscaB, verificando a validade da inserção
2. Se a inserção é válida, incluir a nova entrada na posição **pos** da página F apontada por **pt**
3. Verificar se a página F precisa de particionamento. Se sim, propagar o particionamento enquanto for necessário.

Discussão sobre o algoritmo

- ▶ Inserção sempre ocorre nas folhas
- ▶ Por que?

Discussão sobre o algoritmo

- ▶ Inserção sempre ocorre nas folhas
- ▶ Por que?
 - ▶ Porque o procedimento de busca só vai concluir que a chave não está na árvore quando chegar até uma folha
 - ▶ Desta forma, **pt sempre será uma folha**

Exemplo de Inserção com Particionamento

- ▶ Inserir chave 85

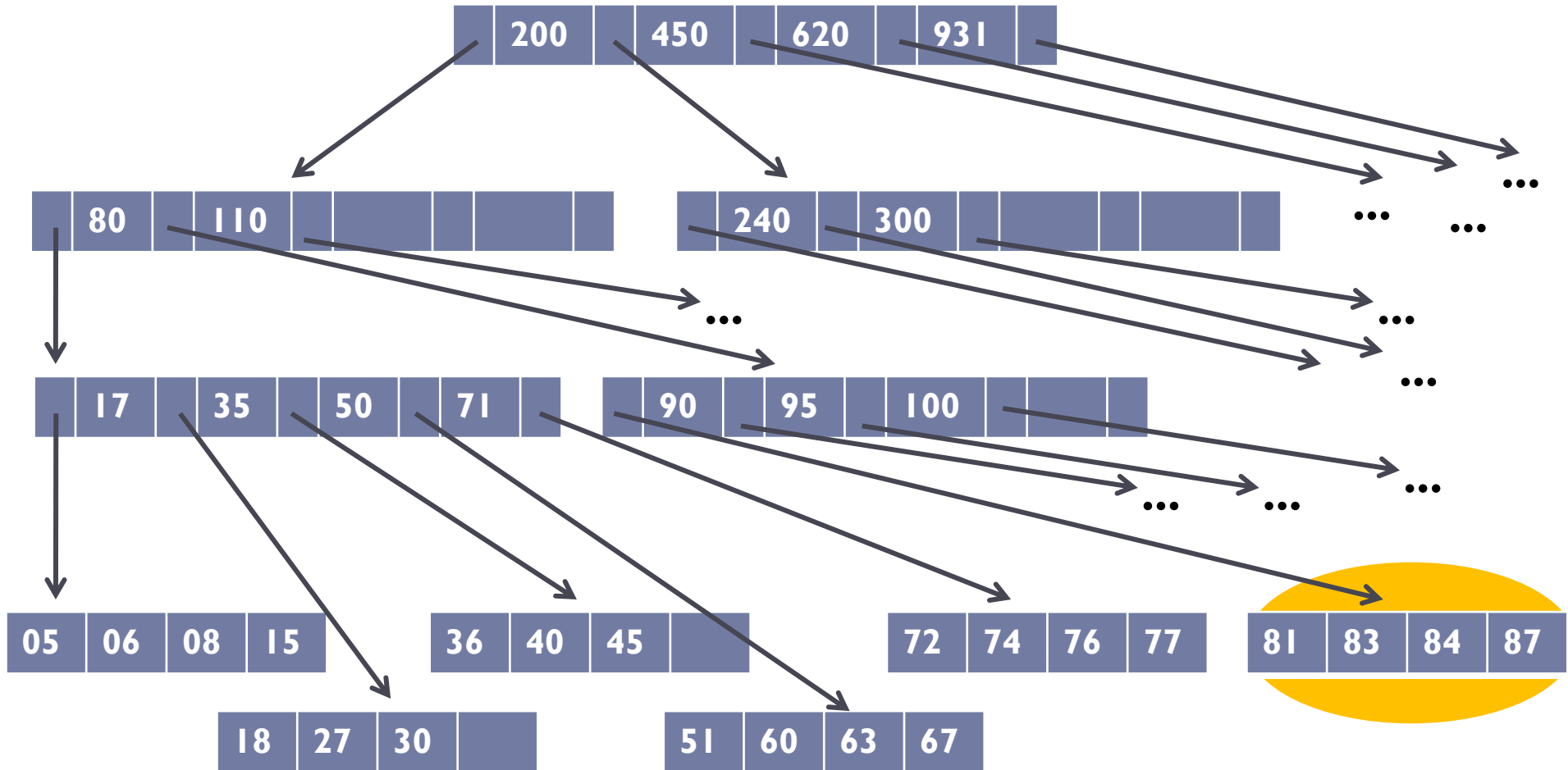
Inserção

c/ Particionamento

Inserir chave 85

Inserção faria página ficar com $2d+1$ chaves
81; 83; 84; 85; 87

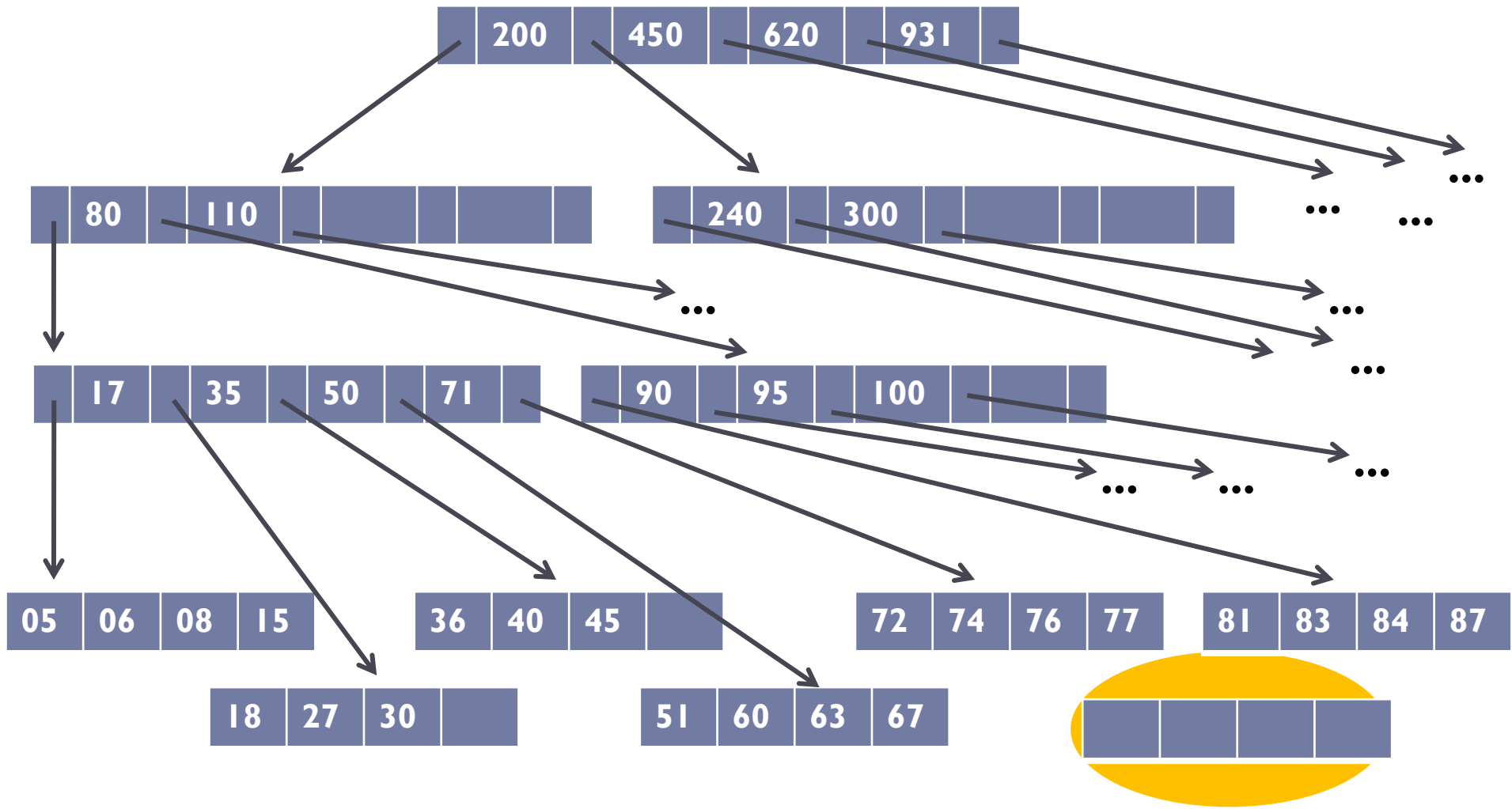
ordem $d = 2$



Inserção

c/ Particionamento

ordem d = 2

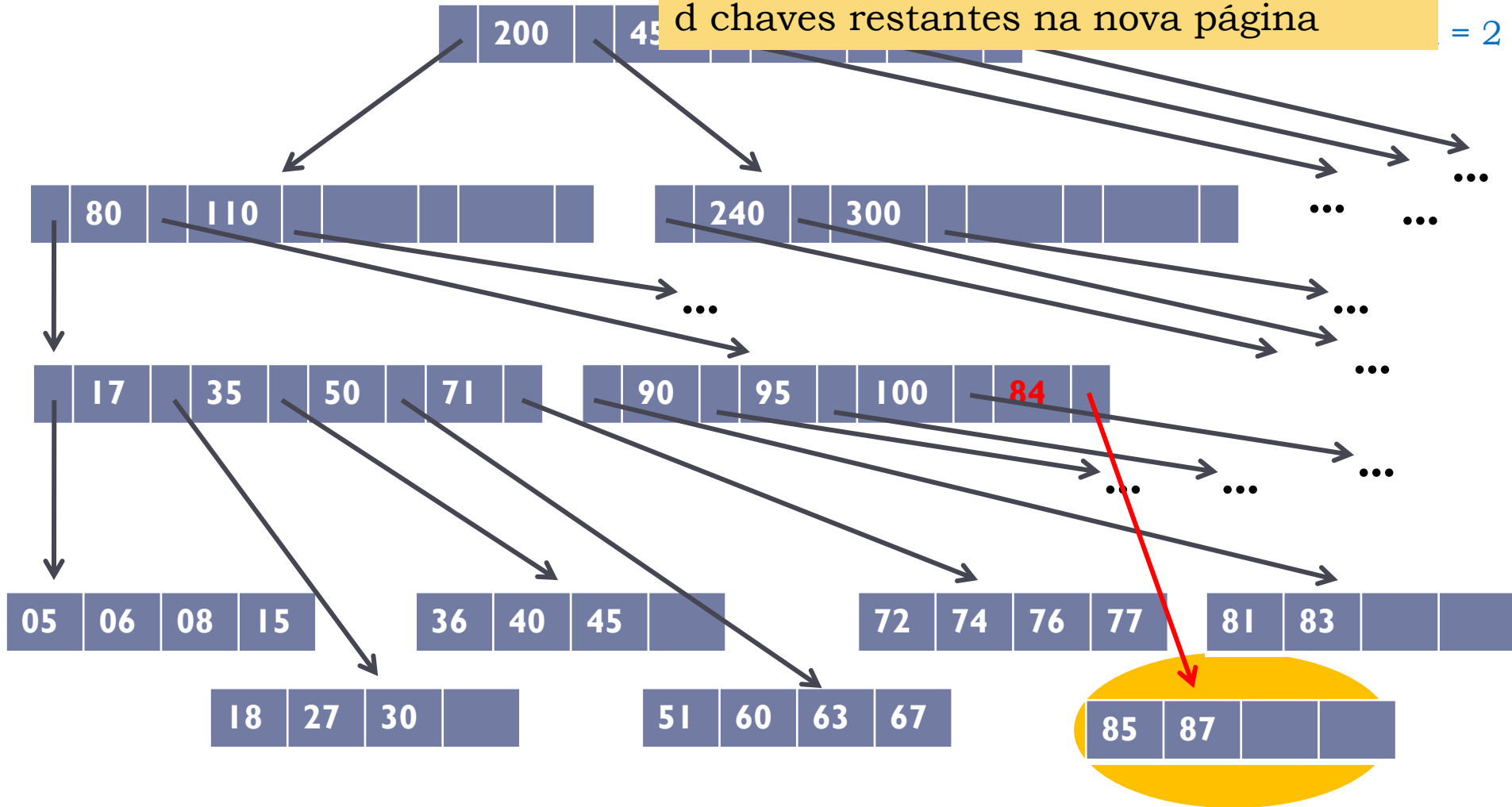


Inserção

c/ Particionamento

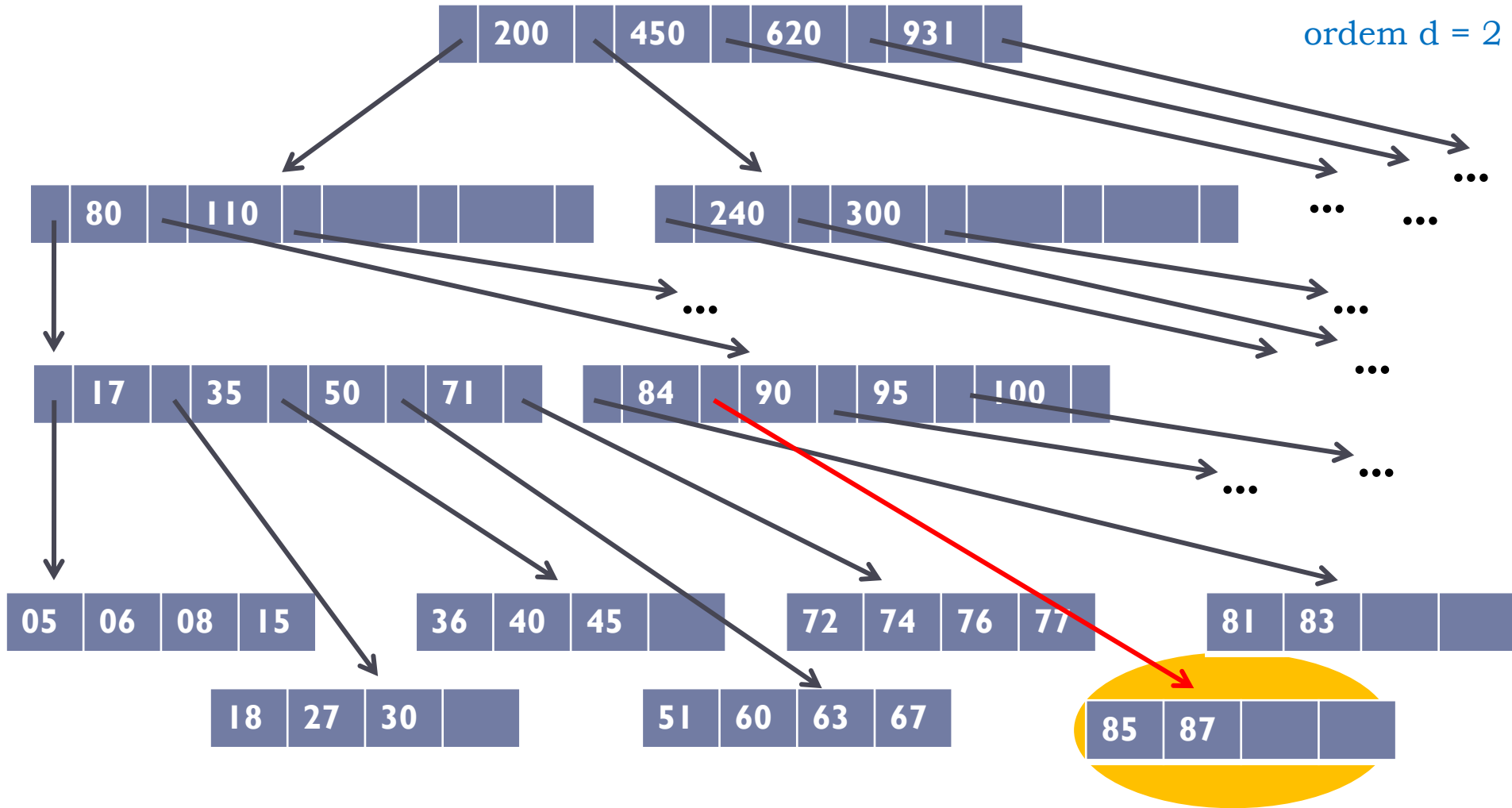
Dividir as chaves entre as duas páginas
(81; 83; 84; 85; 87)
d chaves na página original
chave d+1 sobe para nó pai
d chaves restantes na nova página

= 2



Inserção

c/ Particionamento



Exemplo de propagação

- ▶ Inserir chave 73

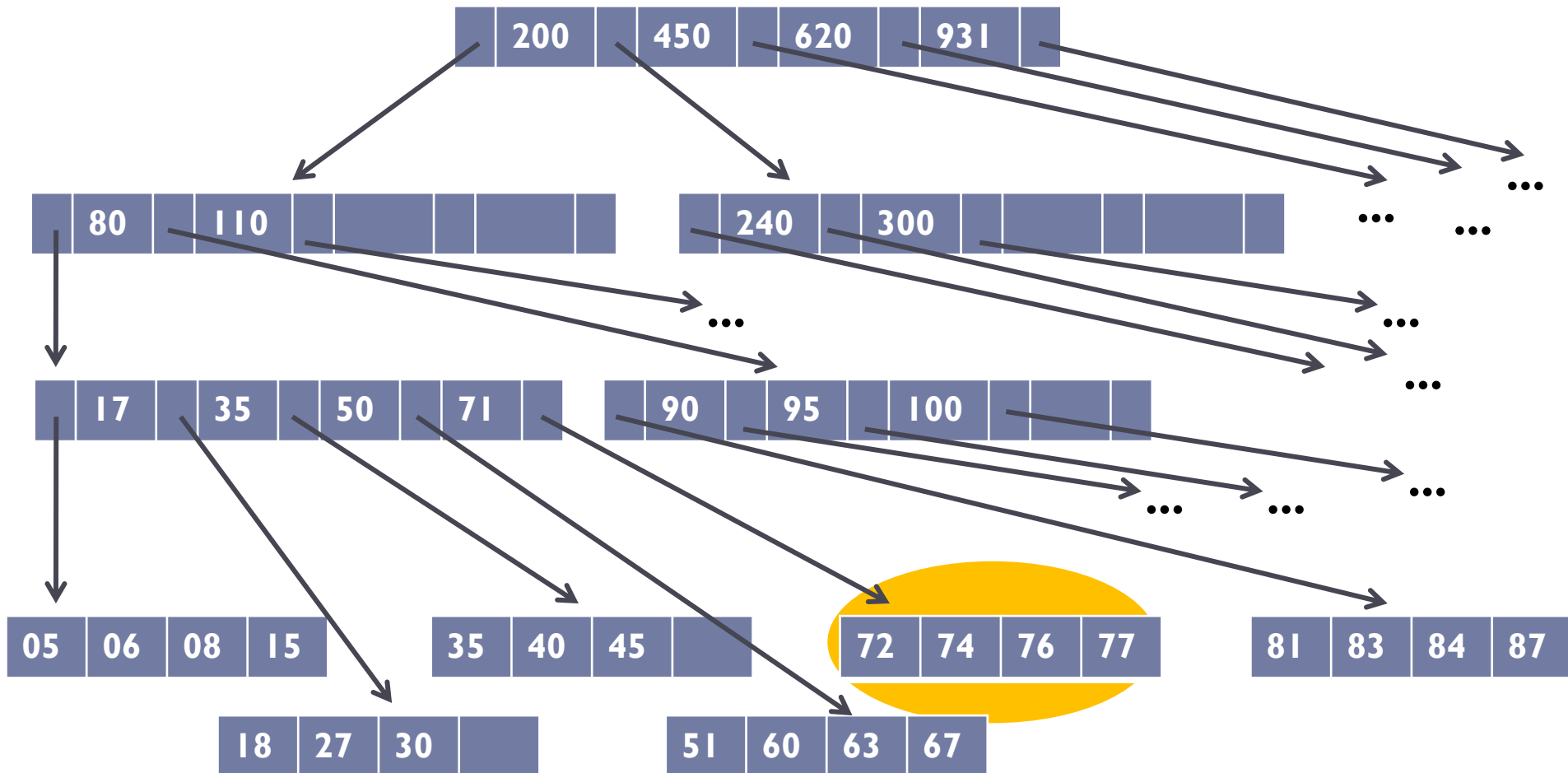
Inserção

c/ Particionamento

Inserir chave 73

Inserção faria página ficar com $2d+1$ chaves
72; 73; 74; 76; 77

ordem $d = 2$

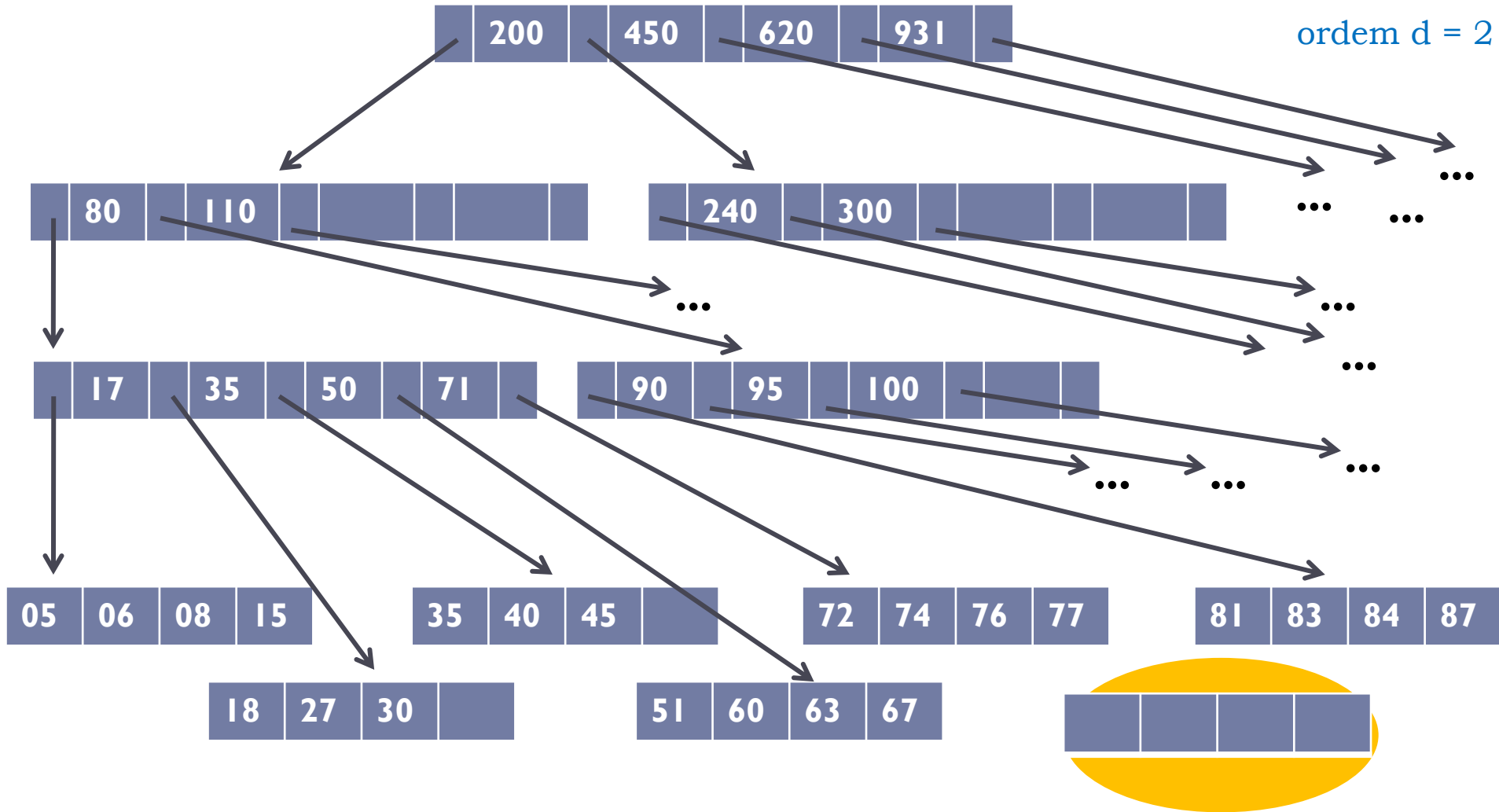


Inserção

c/ Particionamento

Criar nova página

ordem $d = 2$

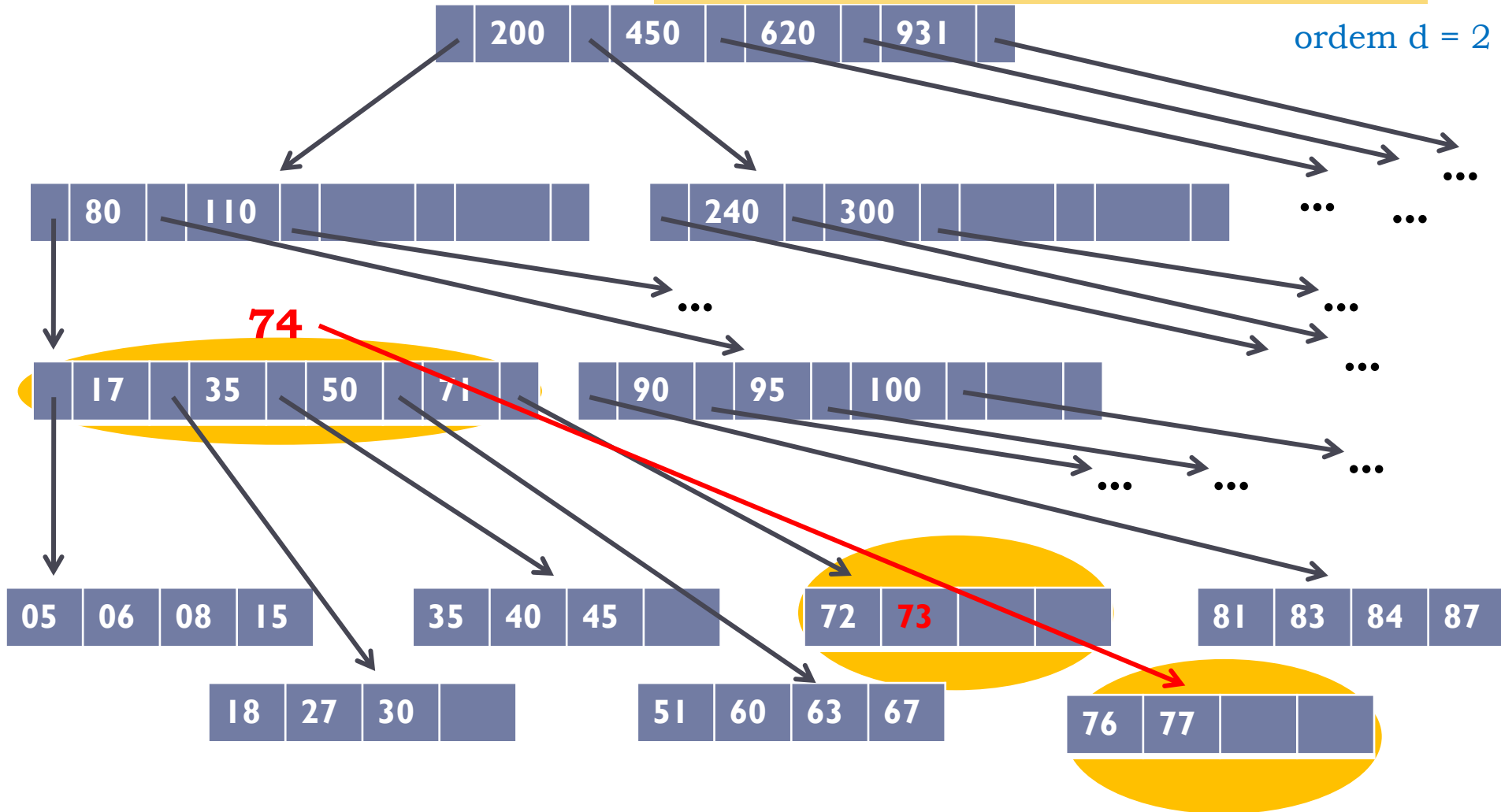


Inserção

c/ Particionamento

Dividir as chaves entre as duas páginas
d chaves na página original
chave d+1 sobe para nó pai
d chaves restantes na nova página

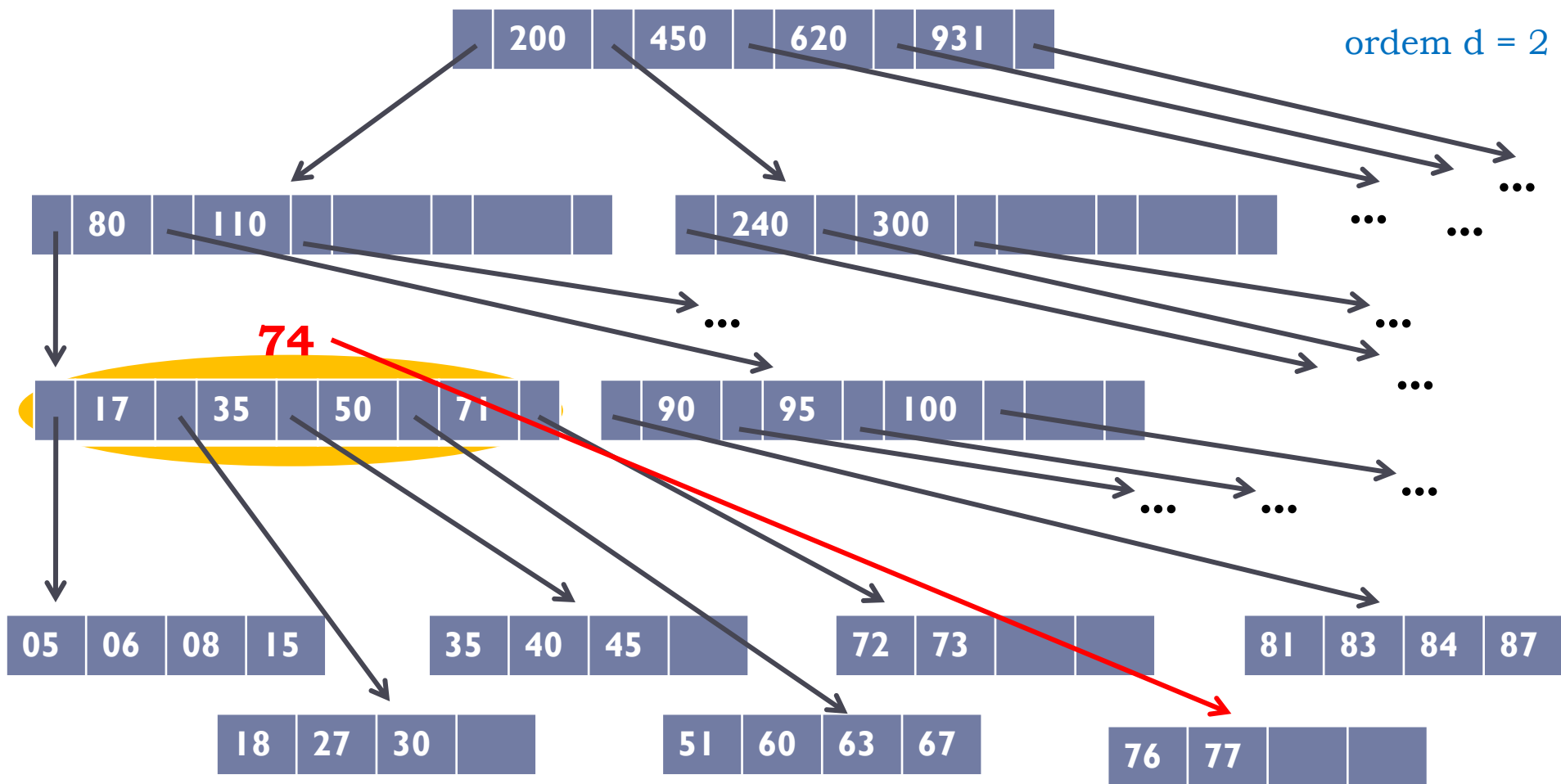
ordem $d = 2$



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

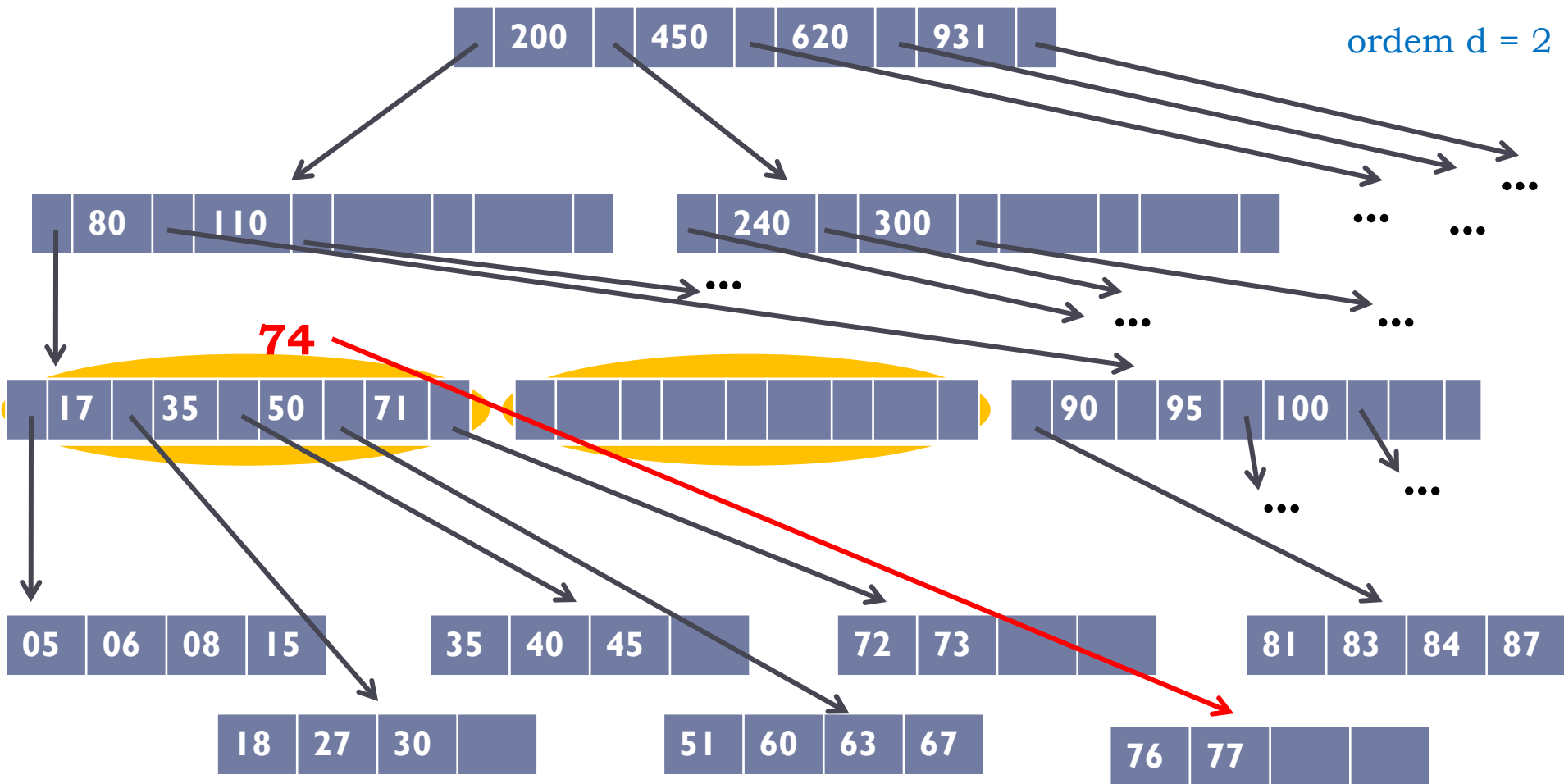
Inserção

c/ Particionamento



Inserção

c/ Particionamento

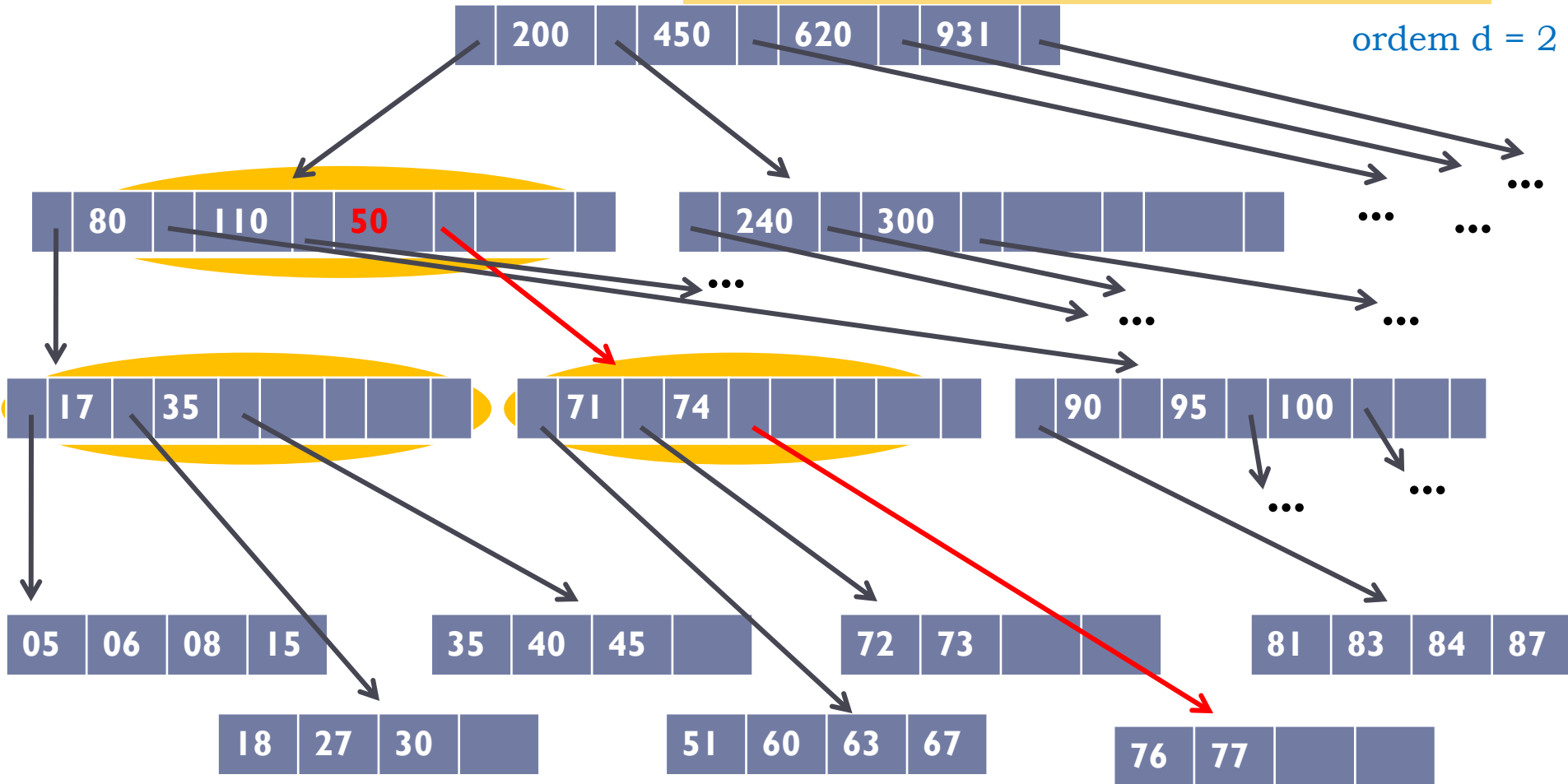


Inserção

c/ Particionamento

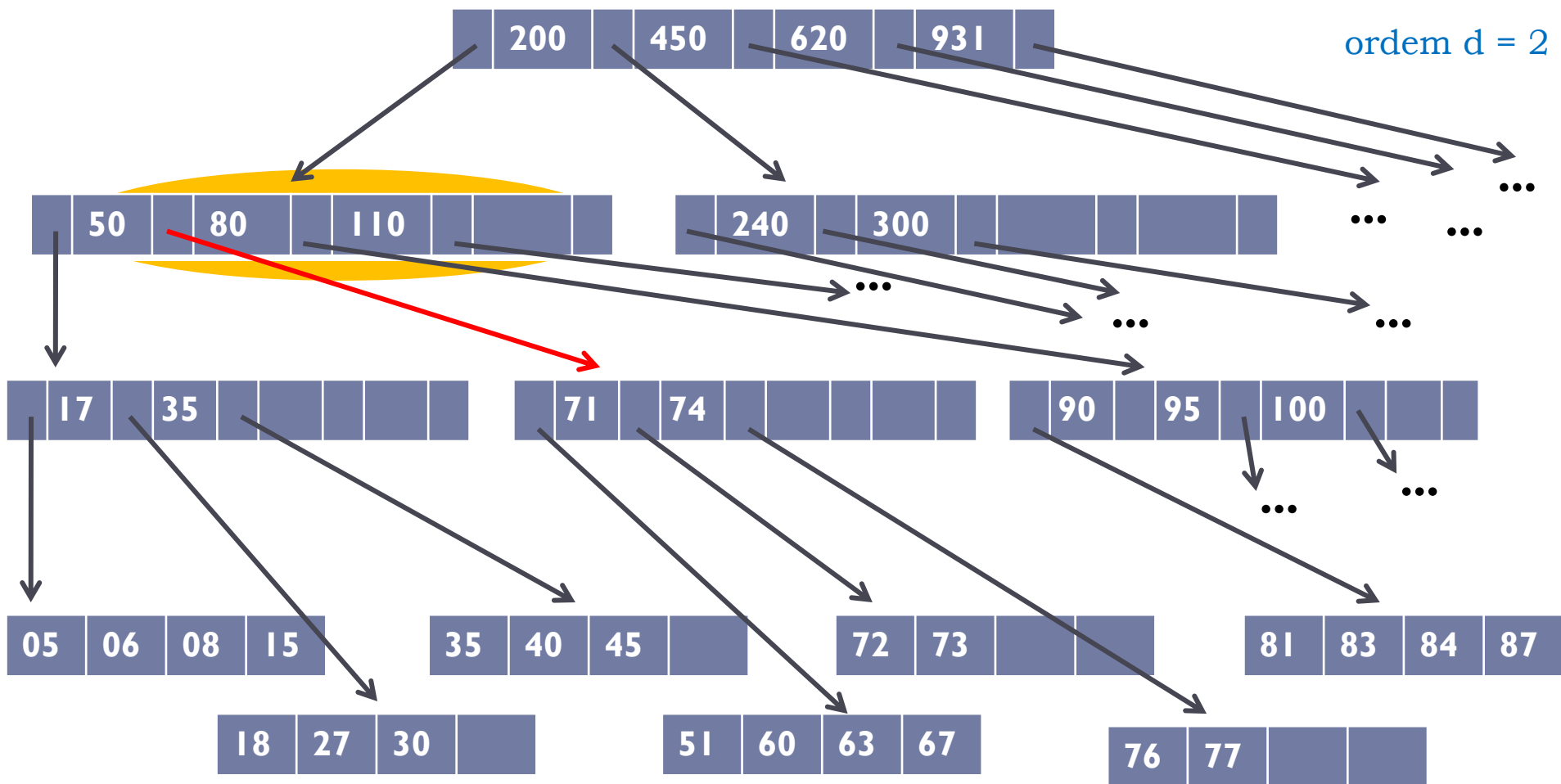
Dividir as chaves entre as duas páginas
d chaves na página original
chave d+1 sobe para nó pai
d chaves restantes na nova página

ordem d = 2



Inserção

c/ Particionamento

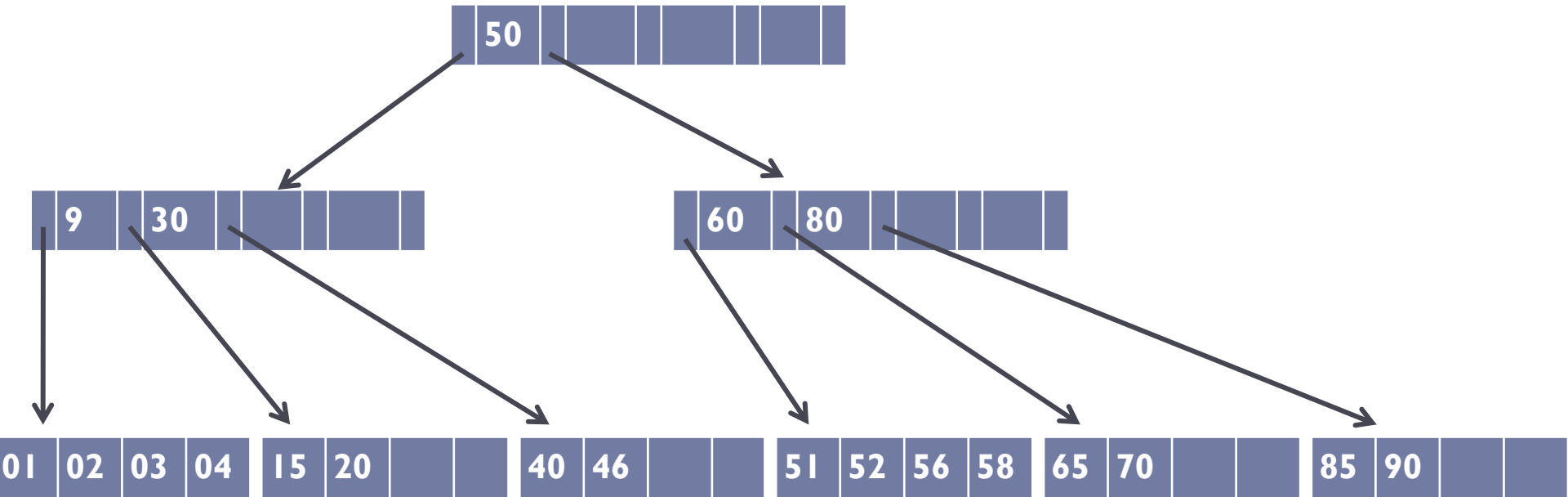


▶ 50 Atenção: os ponteiros dos nós folha foram omitidos por questões de legibilidade da figura. Na prática, todos apontam para NULL

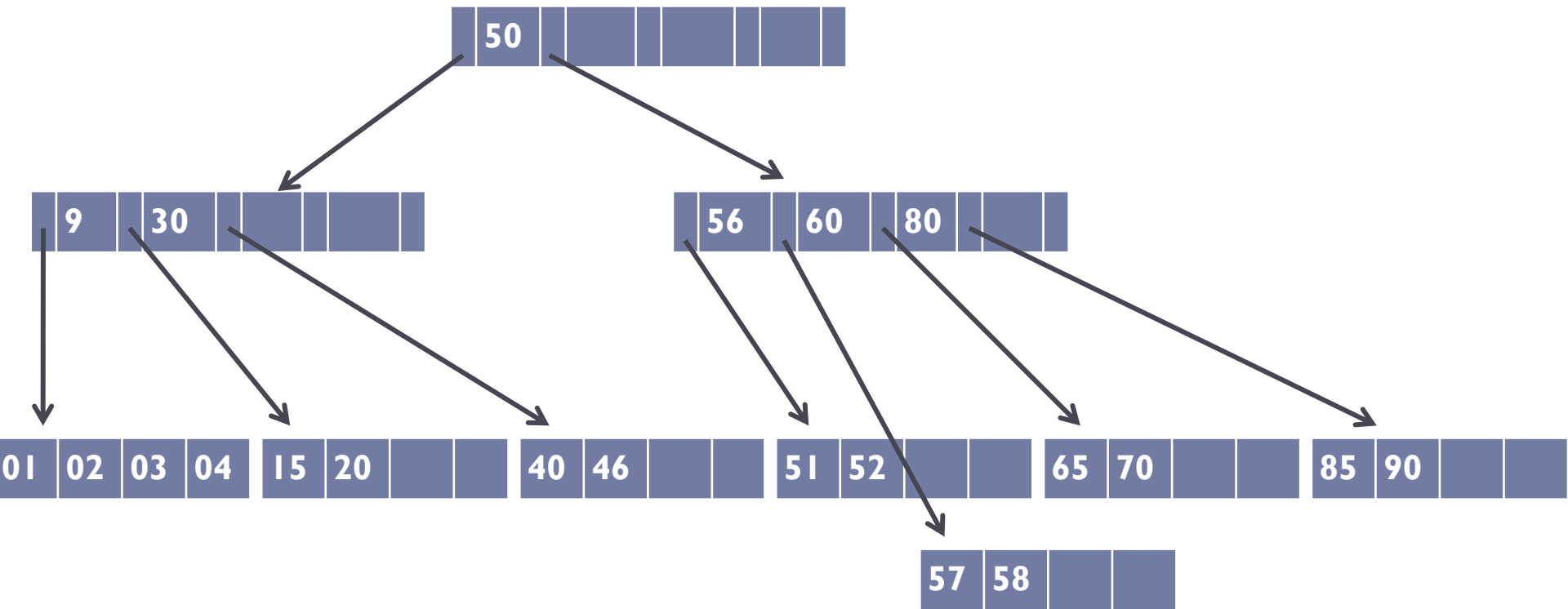
Exercício:

Inserir chaves 57, 71, 72, 73

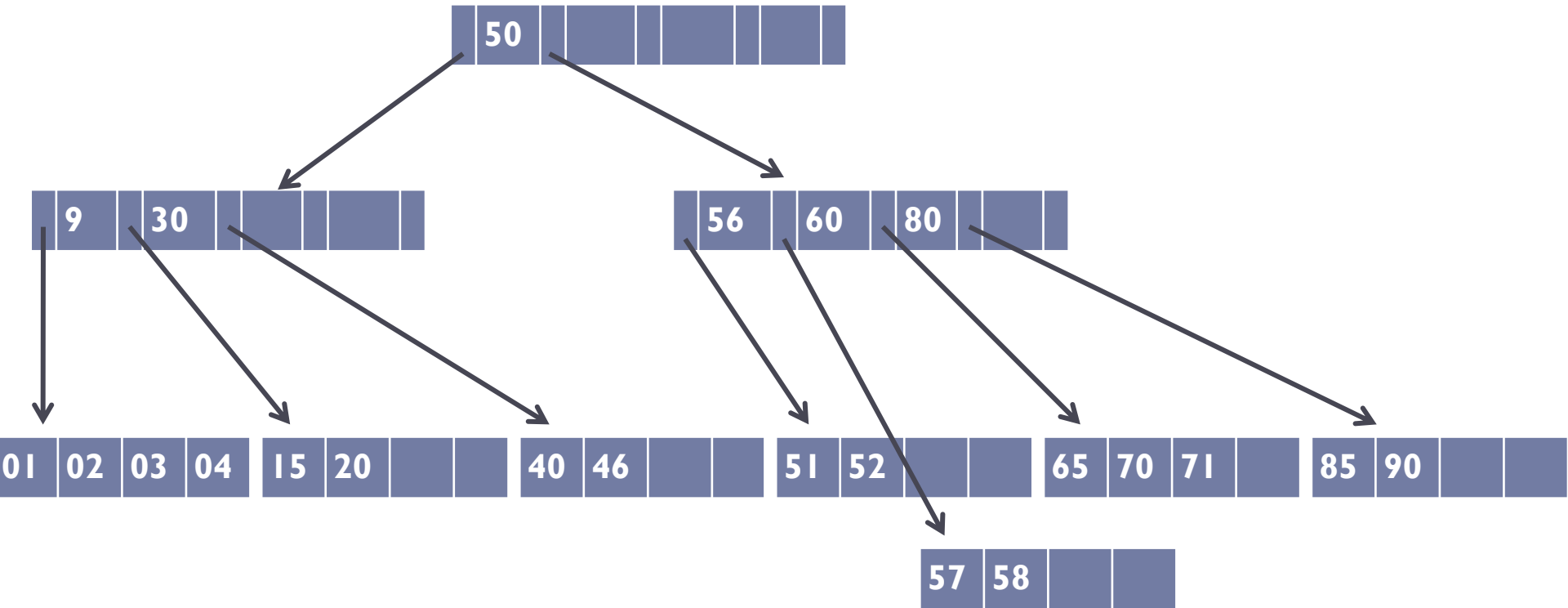
ordem $d = 2$



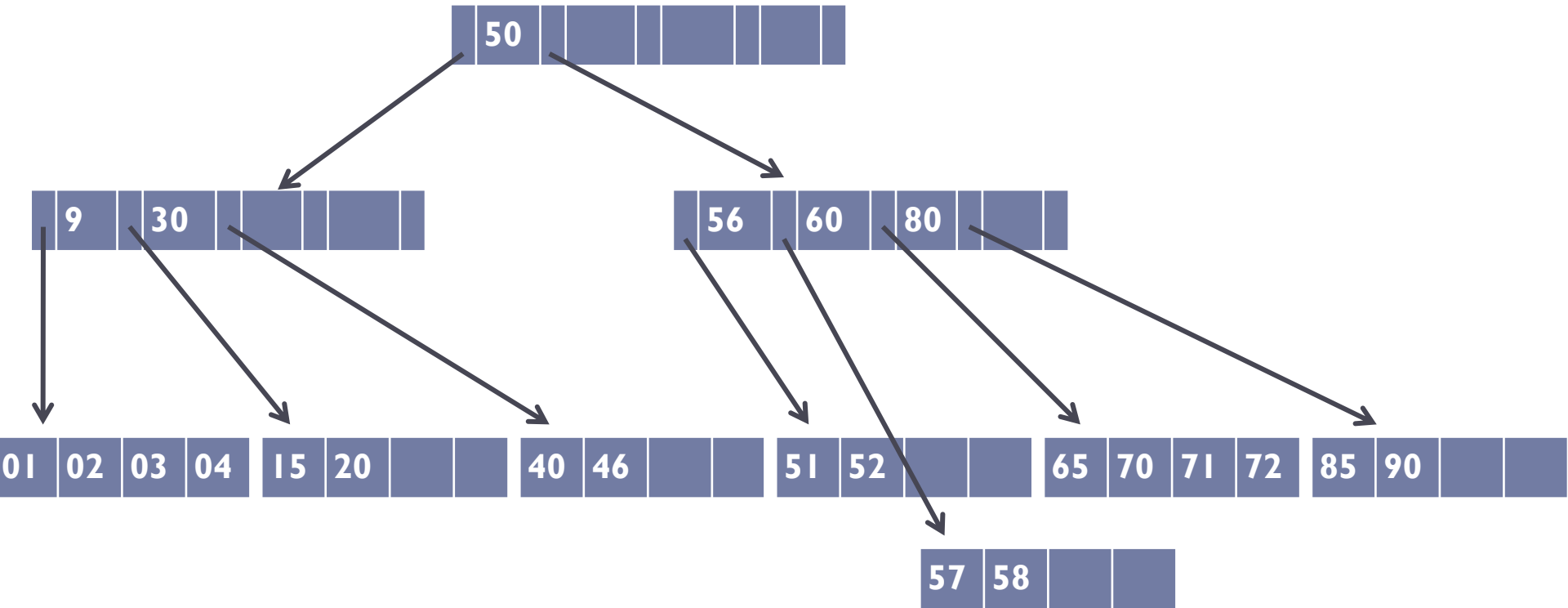
Inserção de 57



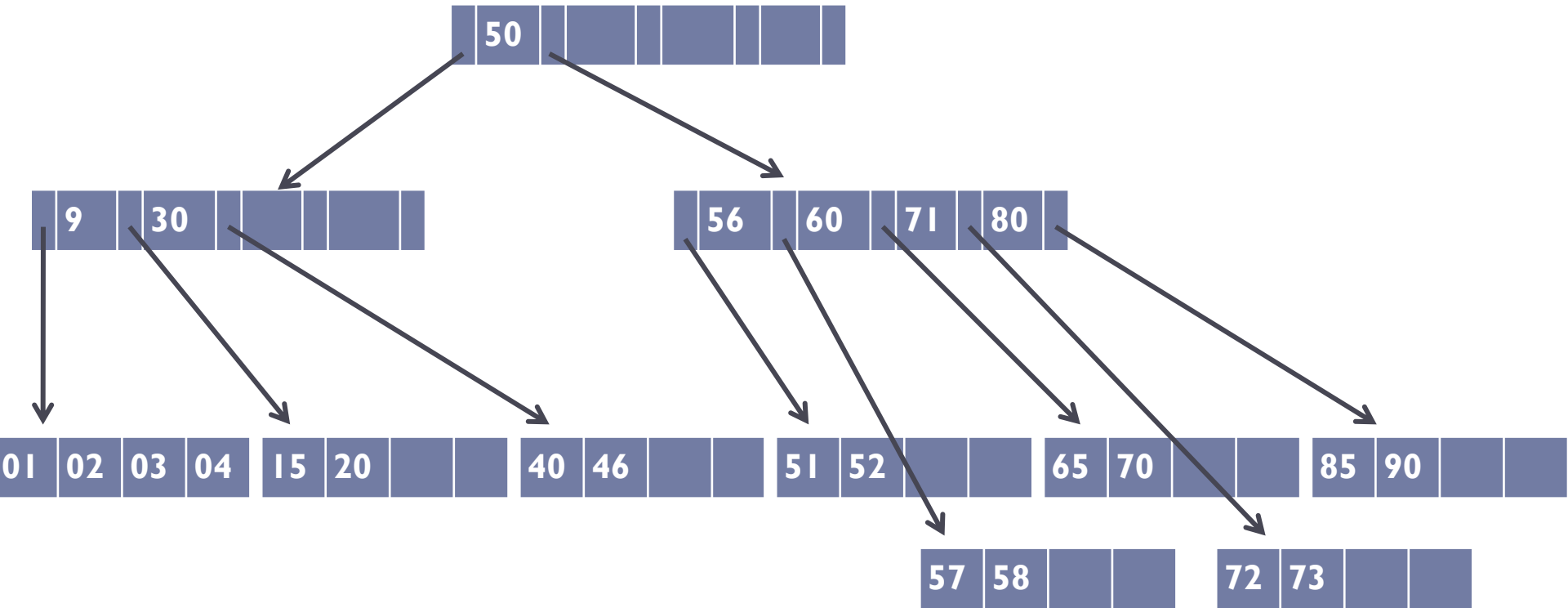
Inserção de 71



Inserção de 72



Inserção de 73



Divisão do nó raiz

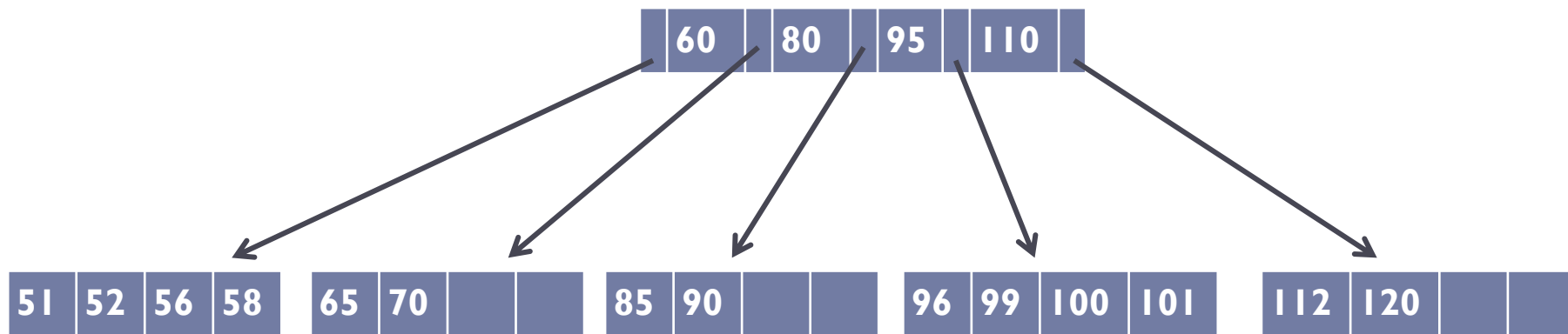
- ▶ Em alguns casos o particionamento se propaga para a raiz
- ▶ Nesse caso, o nó raiz é particionado normalmente, mas, como a raiz não tem pai, cria-se um novo nó, que passa a ser a nova raiz



Exemplo

Inserir chave 97

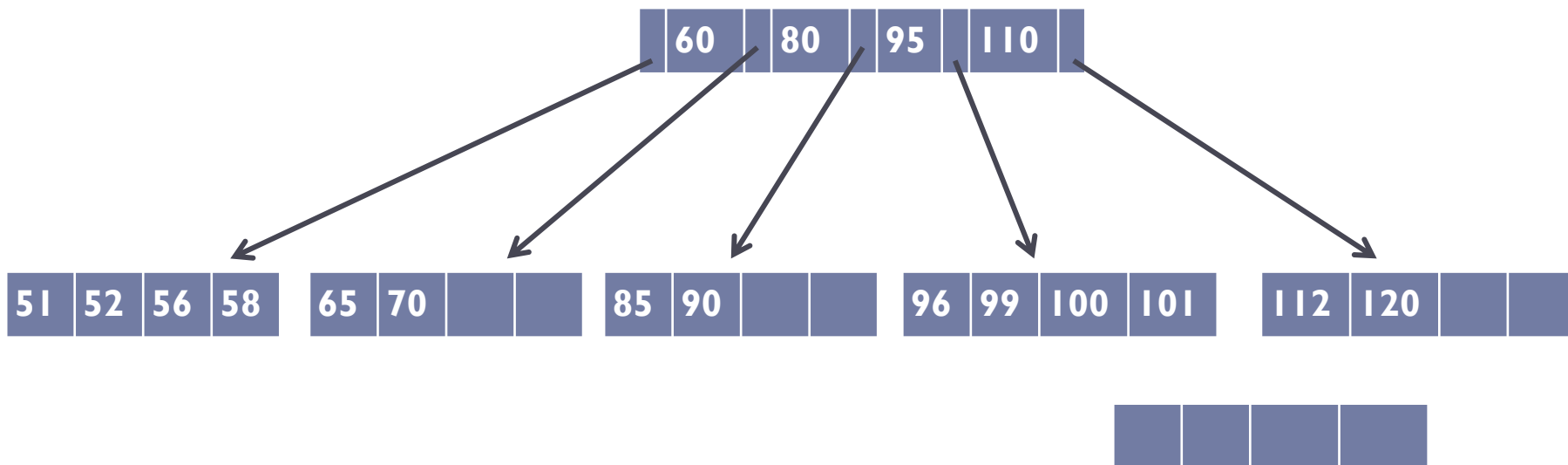
ordem $d = 2$



Exemplo

Inserir chave 97

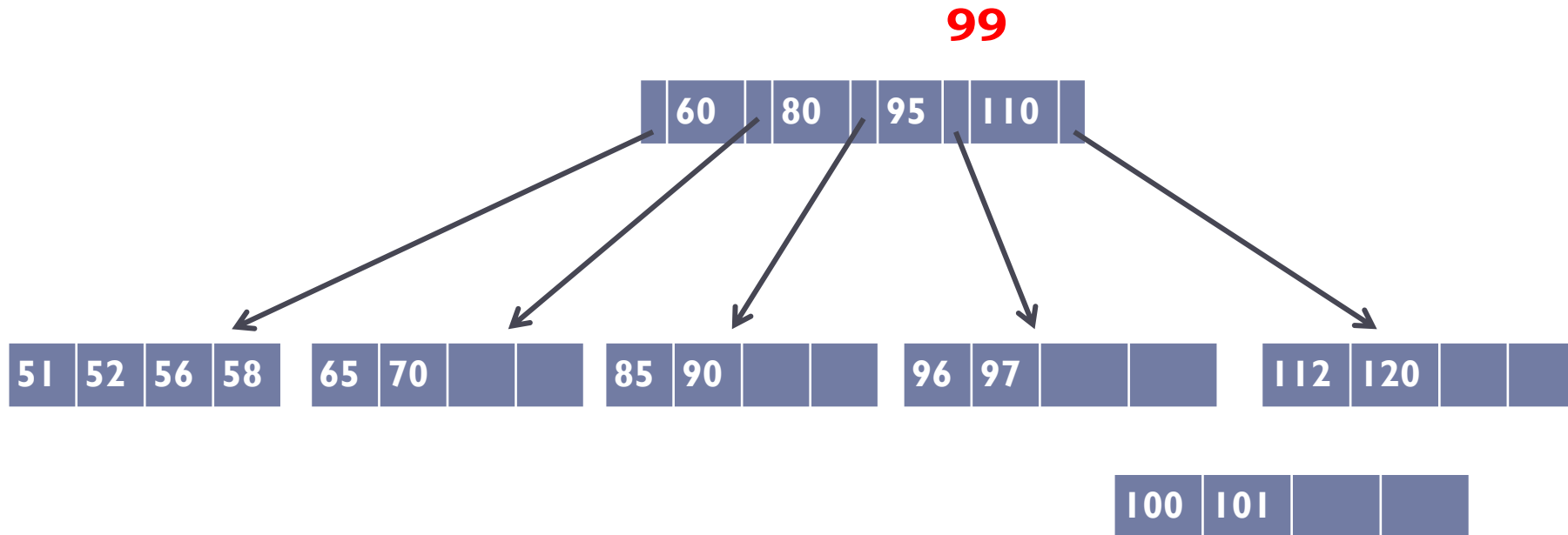
ordem $d = 2$



Exemplo

Inserir chave 97

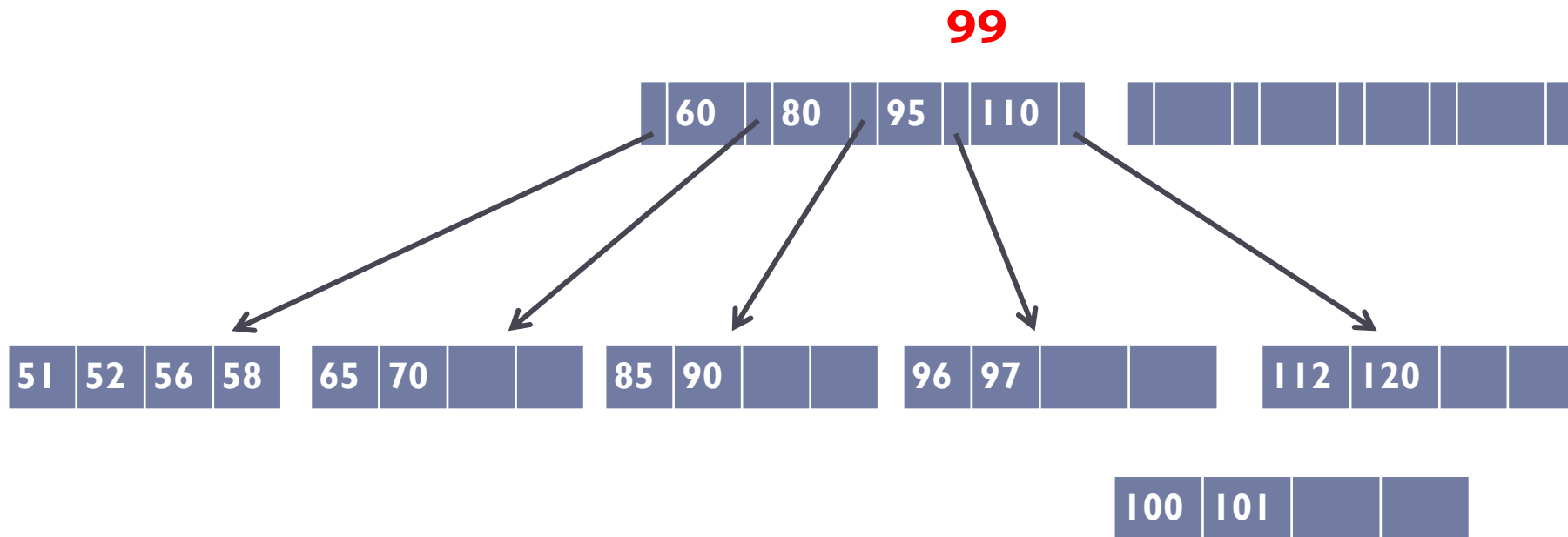
ordem $d = 2$



Exemplo

Inserir chave 97

ordem $d = 2$



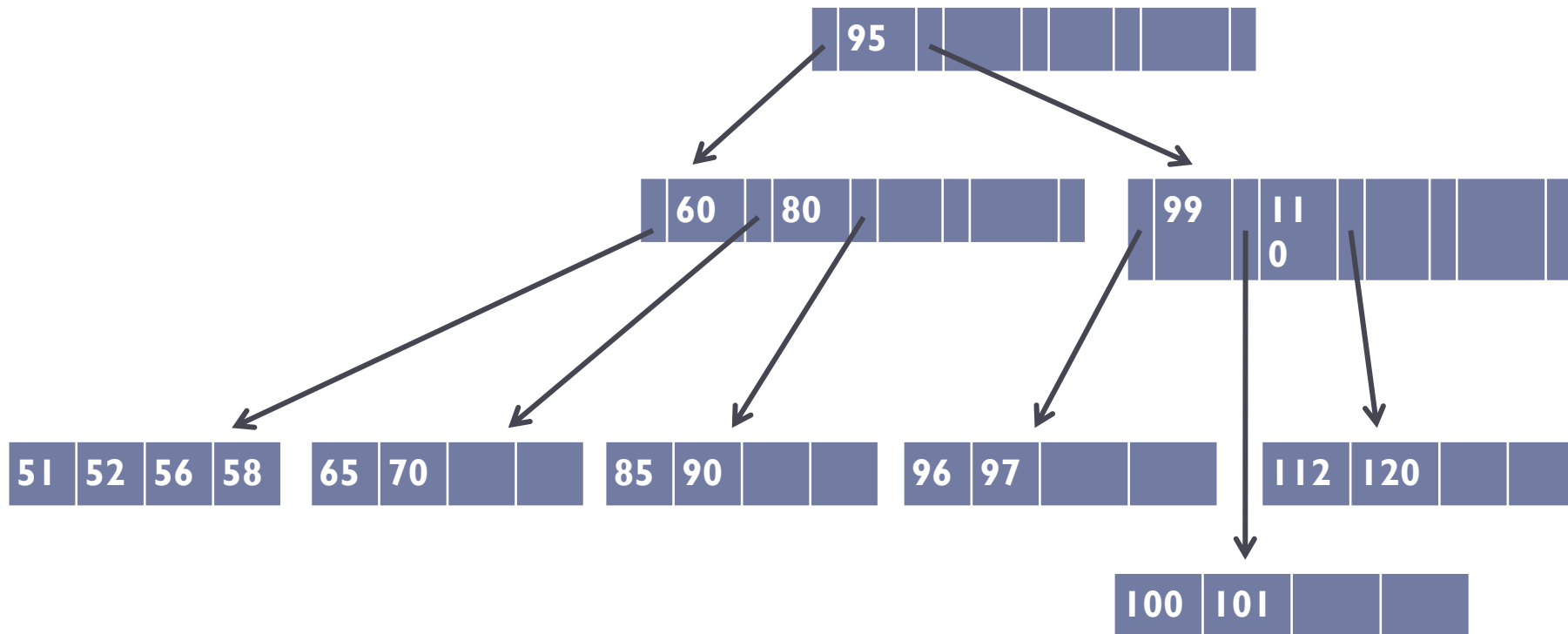
60

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

Exemplo

Inserir chave 97

ordem $d = 2$



Exclusão

- ▶ **Duas situações possíveis:**

- ▶ A entrada **x** está em um nó folha

- ▶ Neste caso, simplesmente remover a entrada **x**

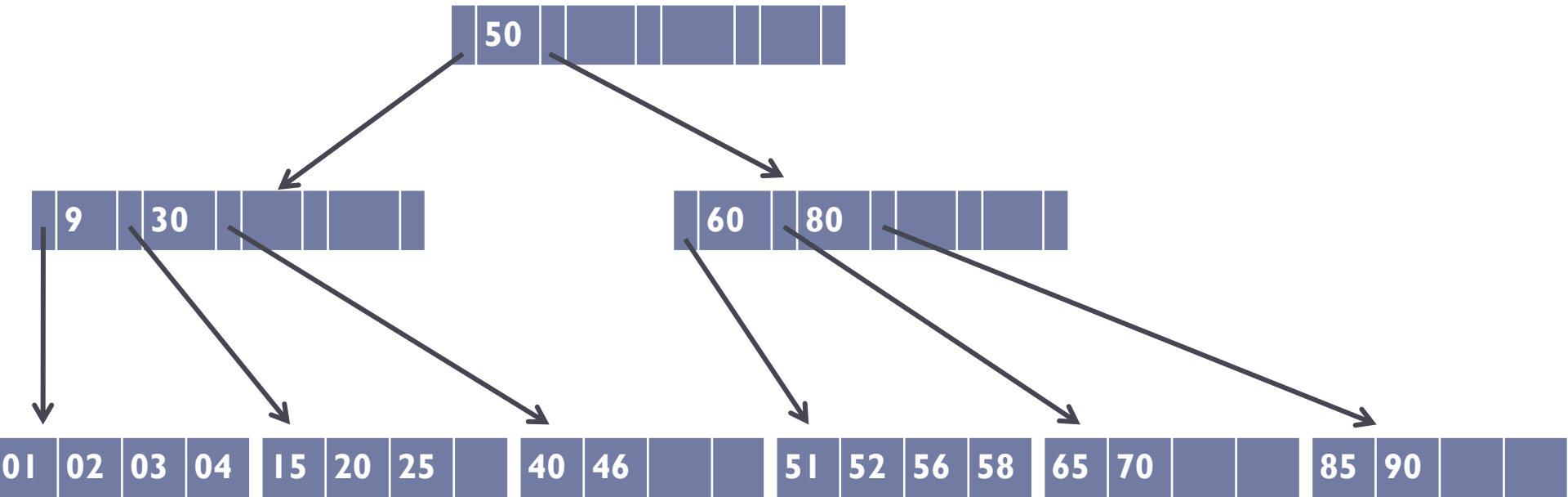
- ▶ A entrada **x** não está em um nó folha

- ▶ Substituir **x** pela chave **y** imediatamente maior

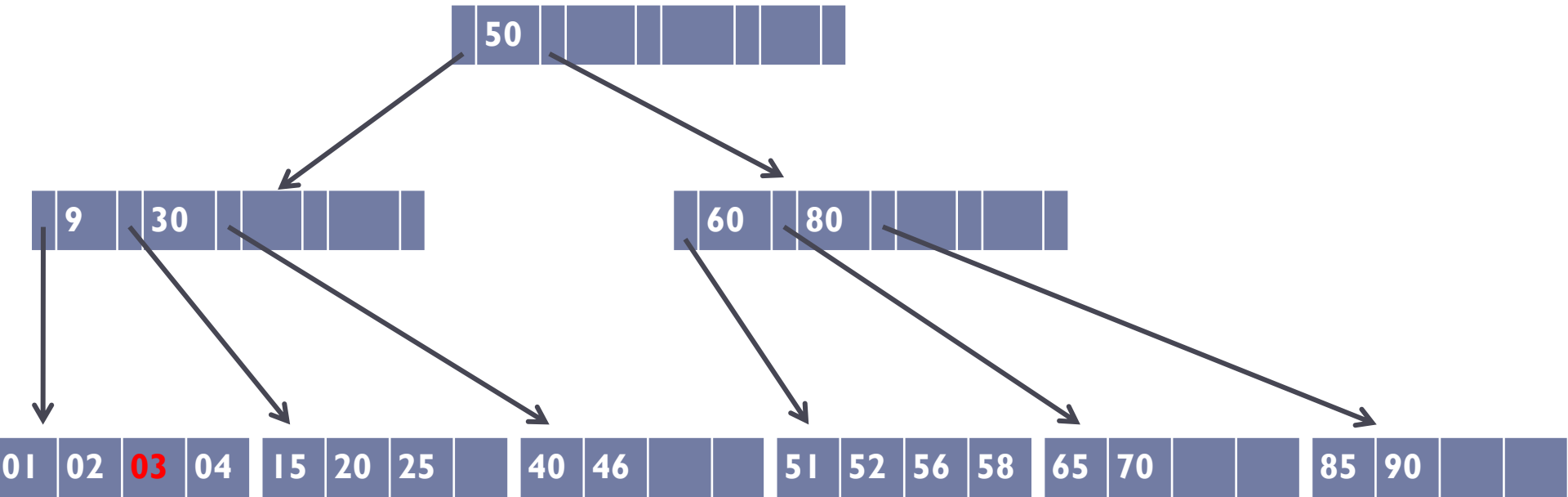
- ▶ Note que **y** necessariamente pertence a uma folha, pela forma como a árvore B é estruturada

Exemplo: Exclusão da chave 03

ordem $d = 2$

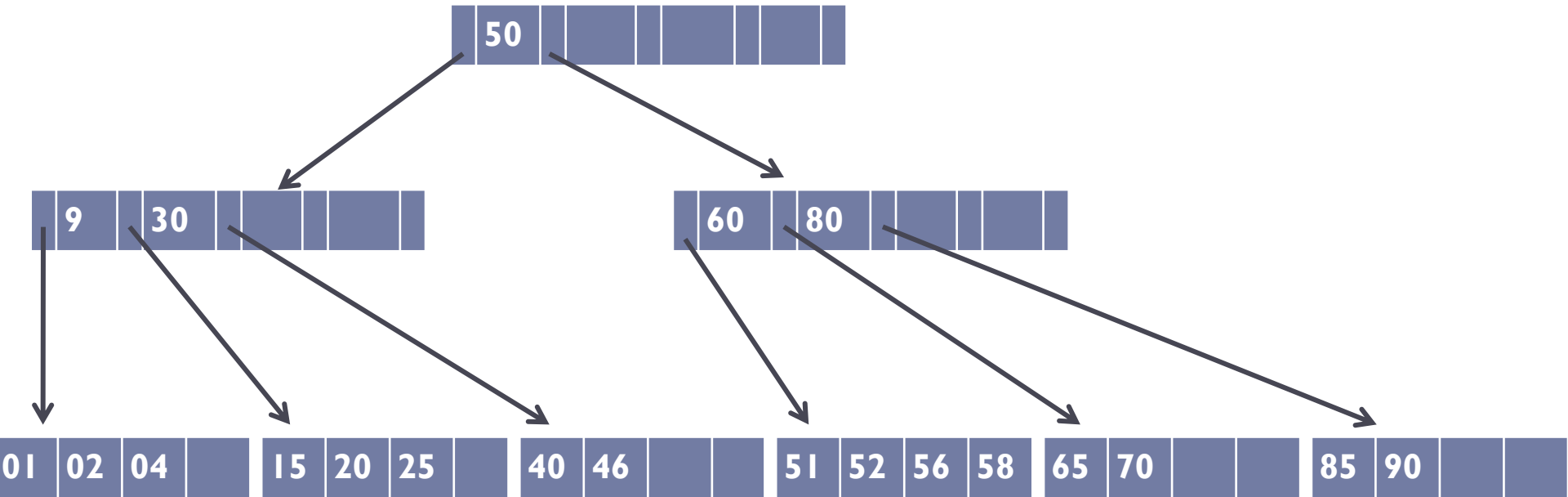


Exemplo: Exclusão da chave 03



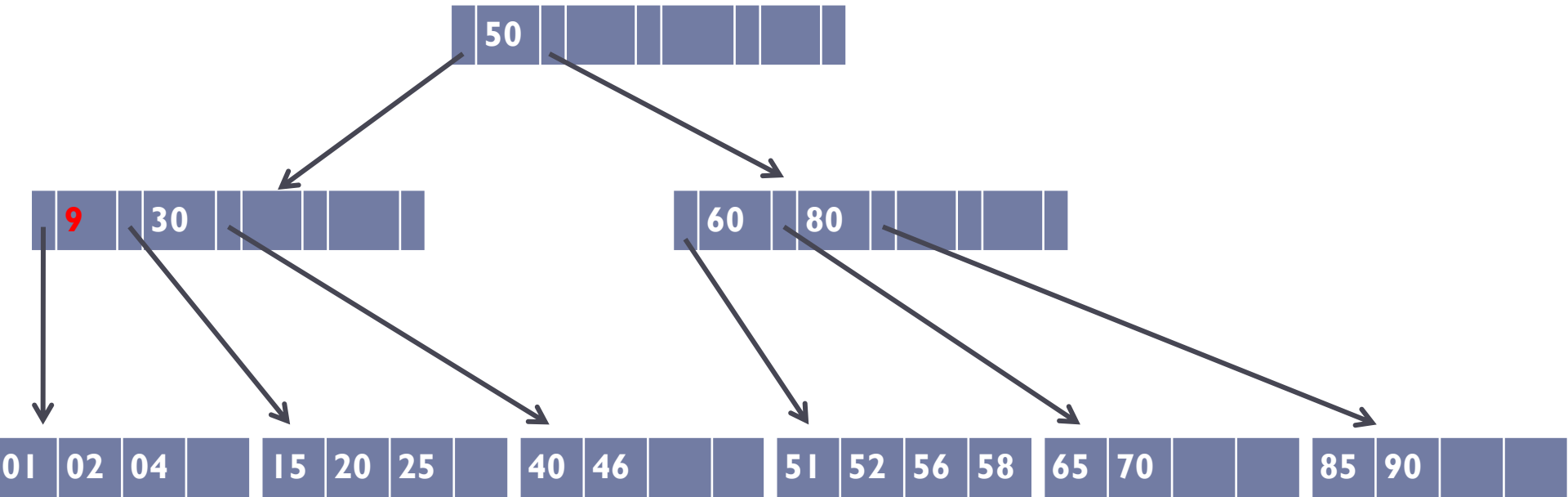
Exemplo: Exclusão da chave 03

ordem $d = 2$



Exemplo: Exclusão da chave 9

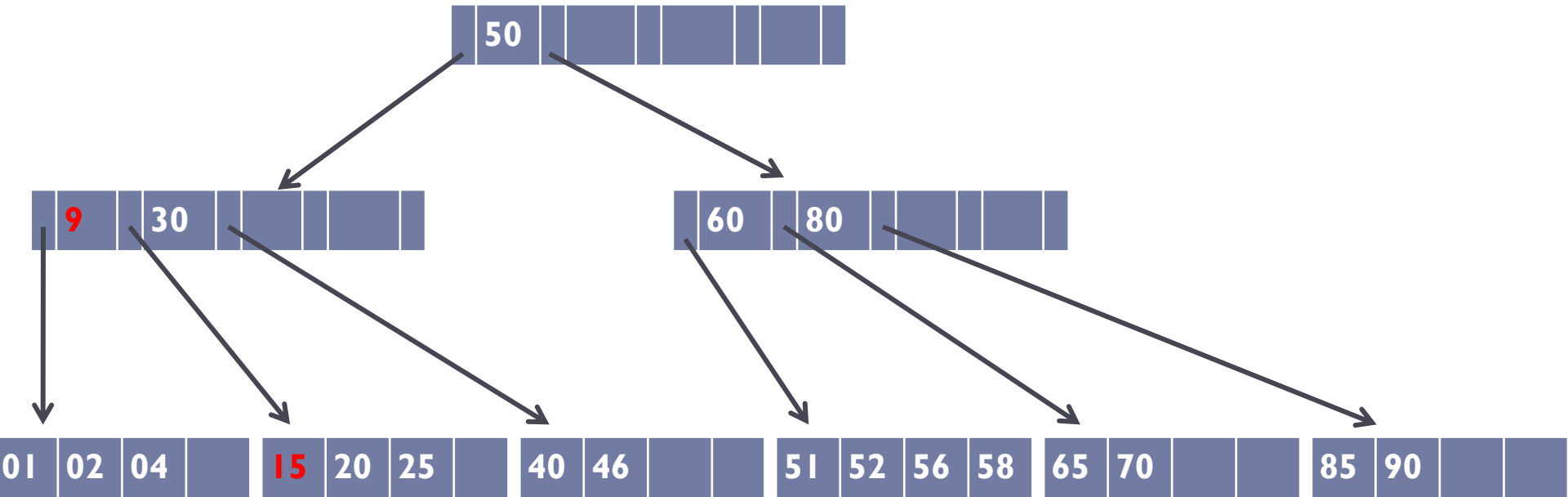
ordem $d = 2$



Substituir pela chave imediatamente maior

Exemplo: Exclusão da chave 9

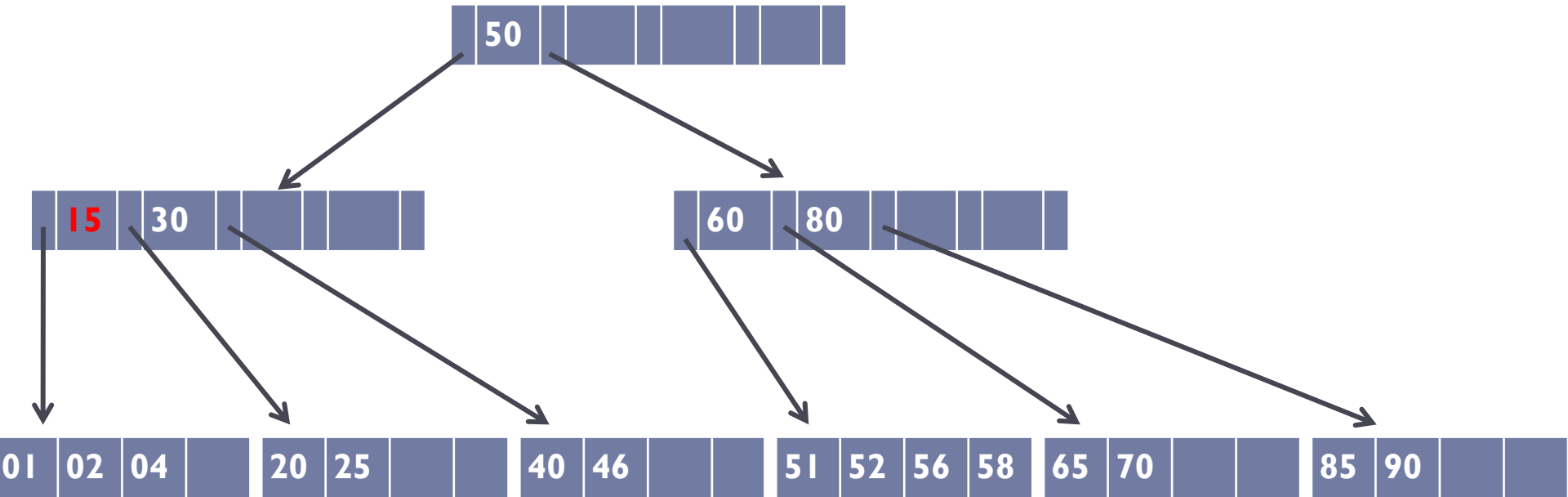
ordem $d = 2$



Substituir pela chave imediatamente maior

Exemplo: Exclusão da chave 9

ordem $d = 2$

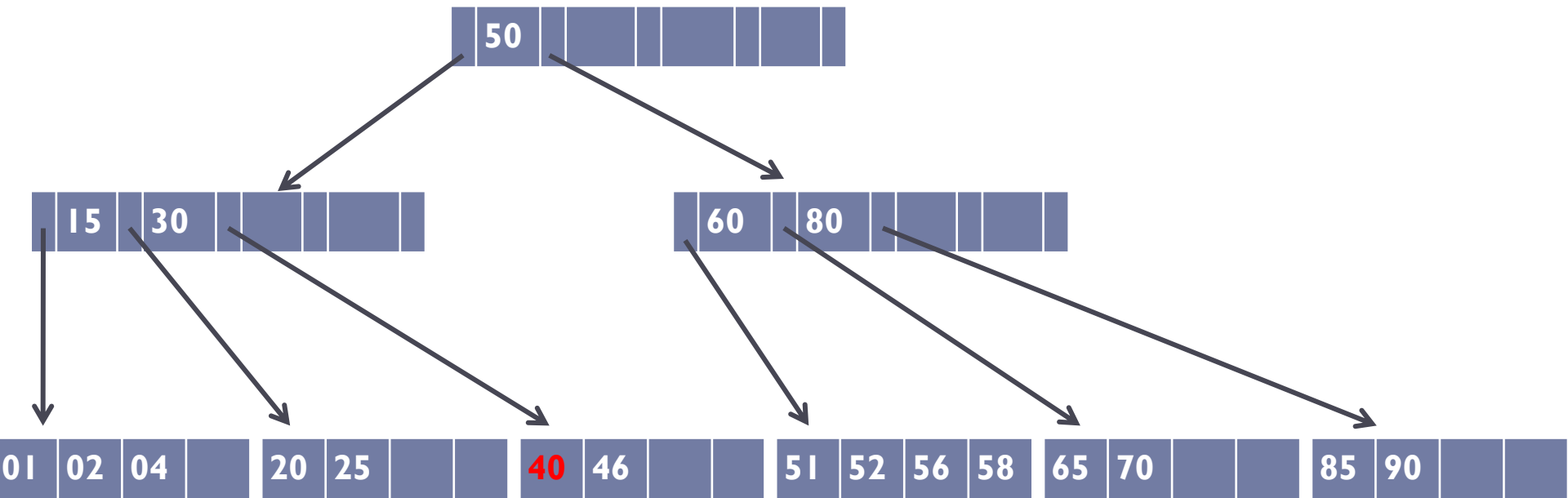


Substituir pela chave imediatamente maior



Exemplo: Exclusão da chave 40

ordem $d = 2$



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

Solução:

- ▶ Concatenação ou Redistribuição

Concatenação

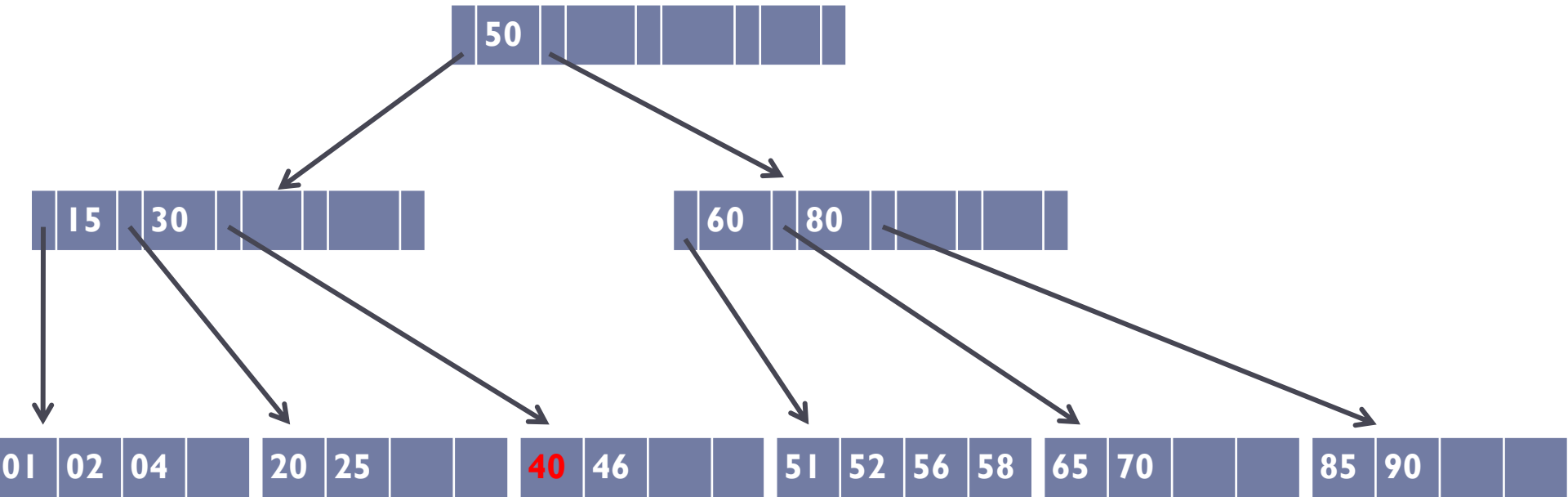
- ▶ Duas páginas **P** e **Q** são **irmãs adjacentes** se têm o mesmo pai **W** e são apontadas por dois ponteiros adjacentes em **W**
- ▶ **P** e **Q** podem ser concatenadas se:
 - ▶ são **irmãs adjacentes**; e
 - ▶ juntas possuem menos de **2d** chaves

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)

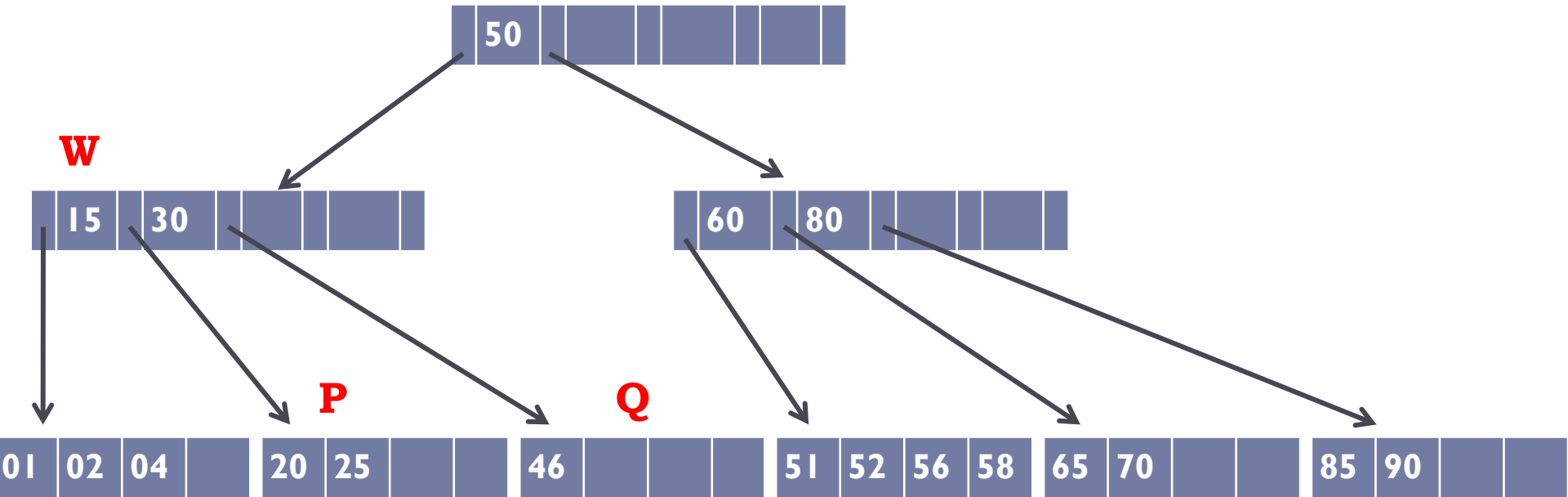
Exemplo: Exclusão da chave 40

ordem $d = 2$



Exemplo: Exclusão da chave 40

ordem $d = 2$



Página Q ficou com menos de d chaves

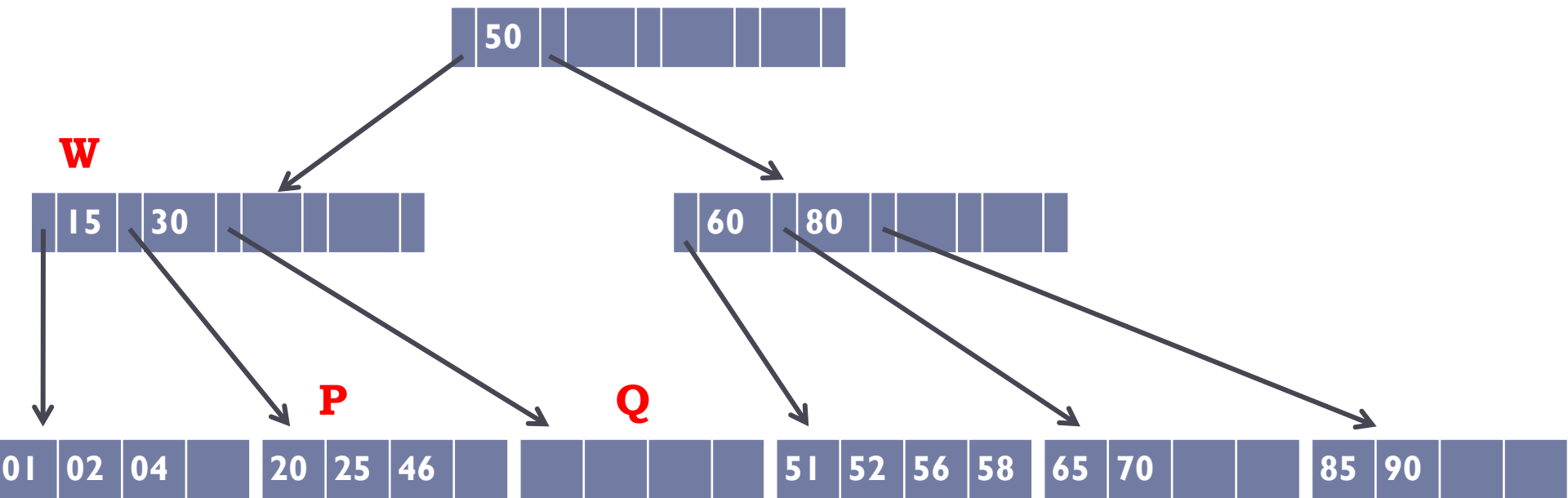
Página P e Q são irmãs adjacentes

Soma de chaves de P e Q $< 2d$

CONCATENAR P e Q

Exemplo: Exclusão da chave 40

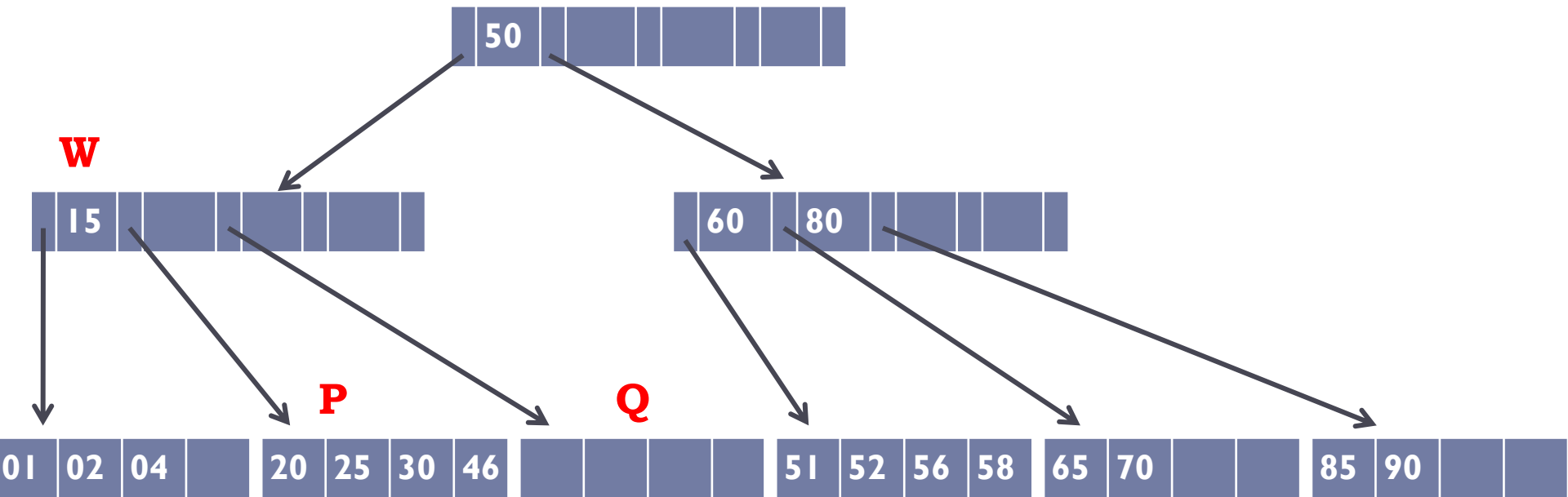
ordem $d = 2$



Transferir dados de Q para P

Exemplo: Exclusão da chave 40

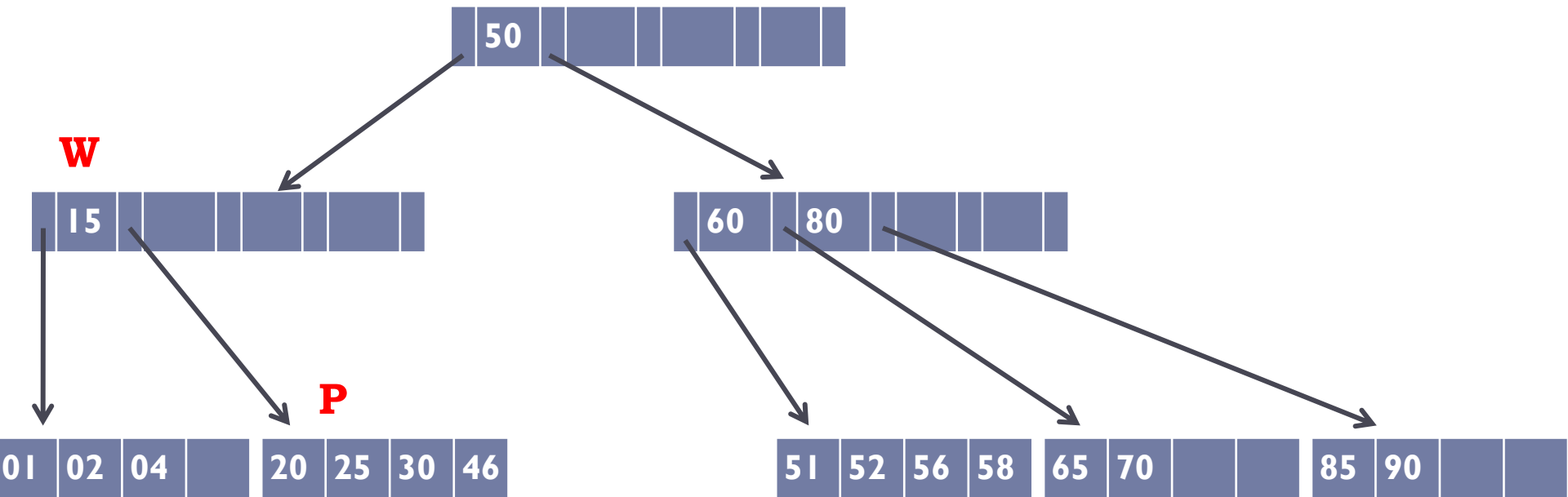
ordem $d = 2$



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

Exemplo: Exclusão da chave 40

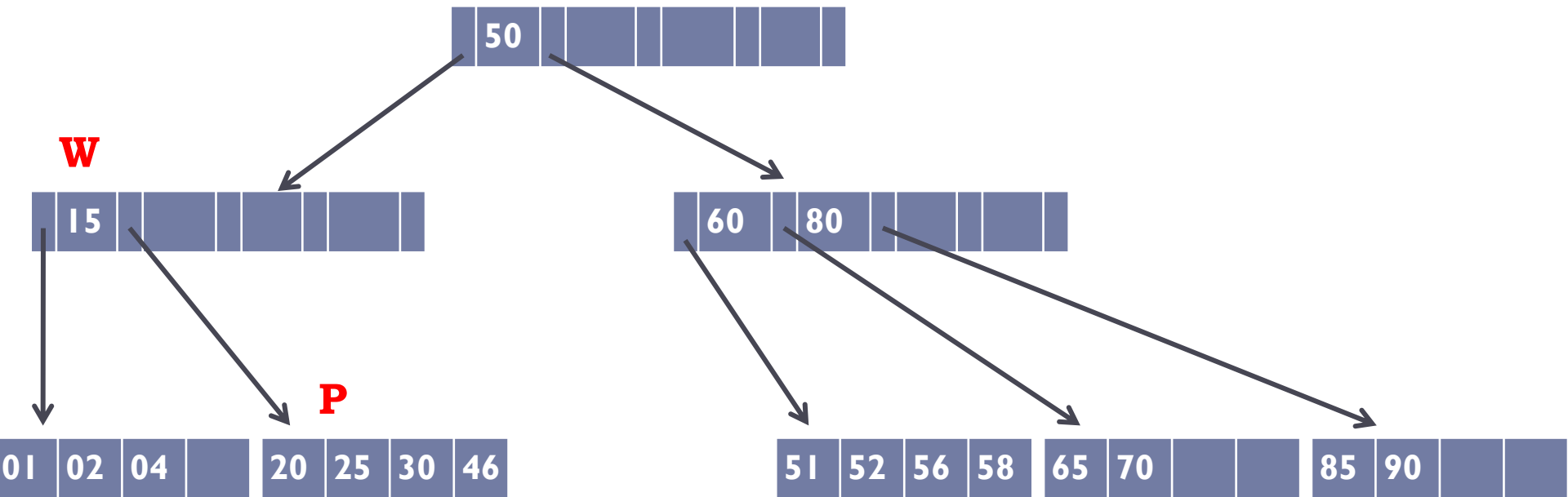
ordem $d = 2$



Eliminar página Q e ponteiro

Exemplo: Exclusão da chave 40

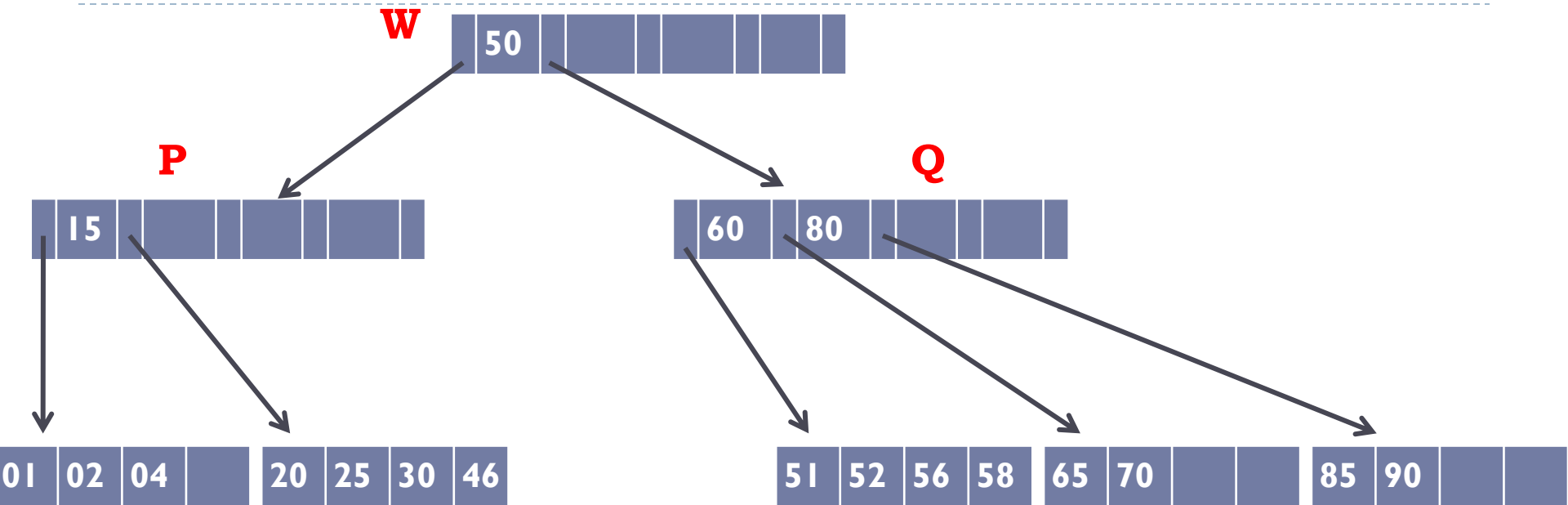
ordem $d = 2$



Página W ficou com menos de d chaves
necessário propagar operação

Exemplo: Exclusão da chave 40

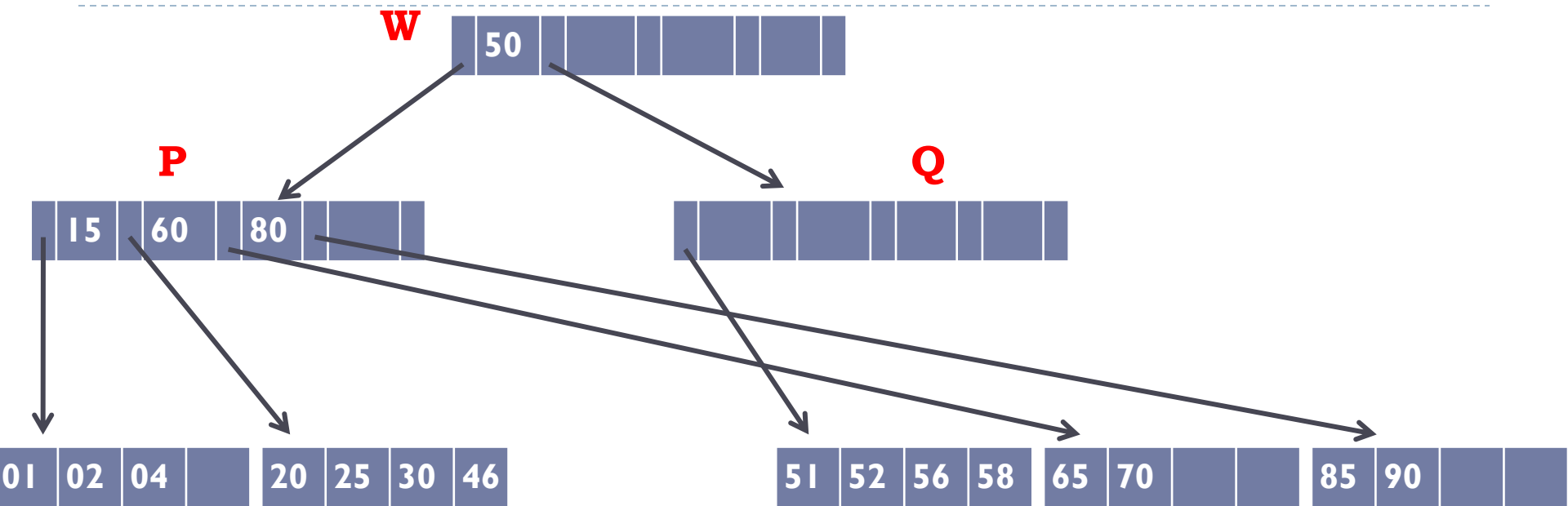
ordem $d = 2$



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

Exemplo: Exclusão da chave 40

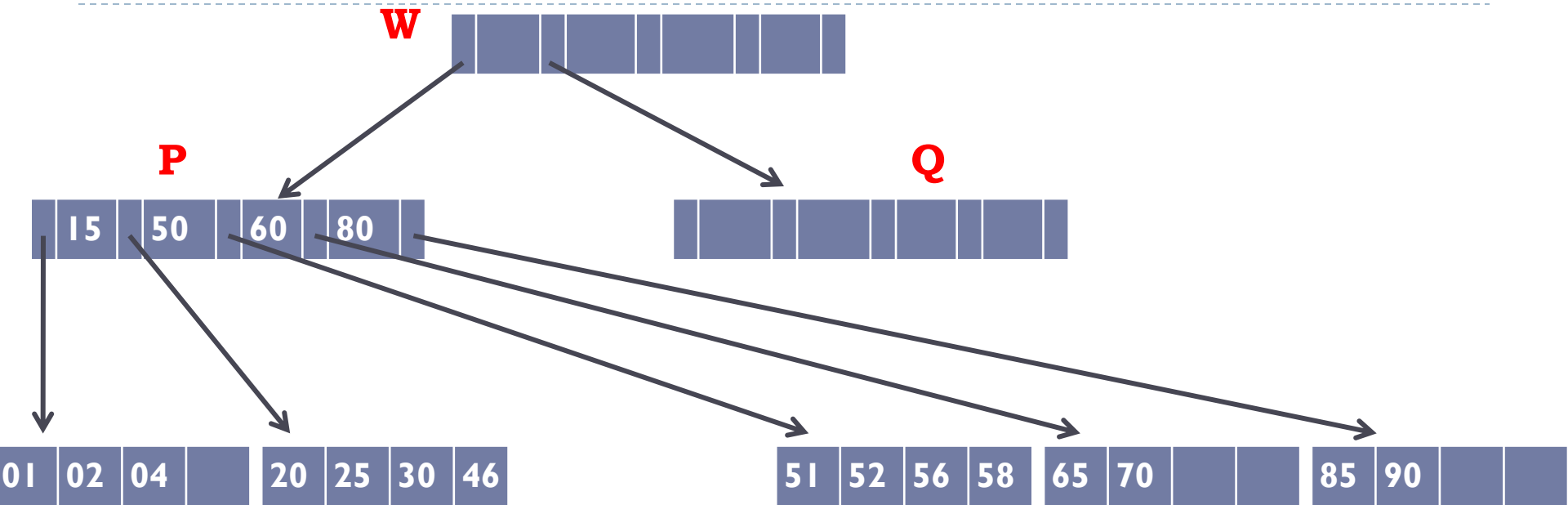
ordem $d = 2$



Transferir dados de Q para P

Exemplo: Exclusão da chave 40

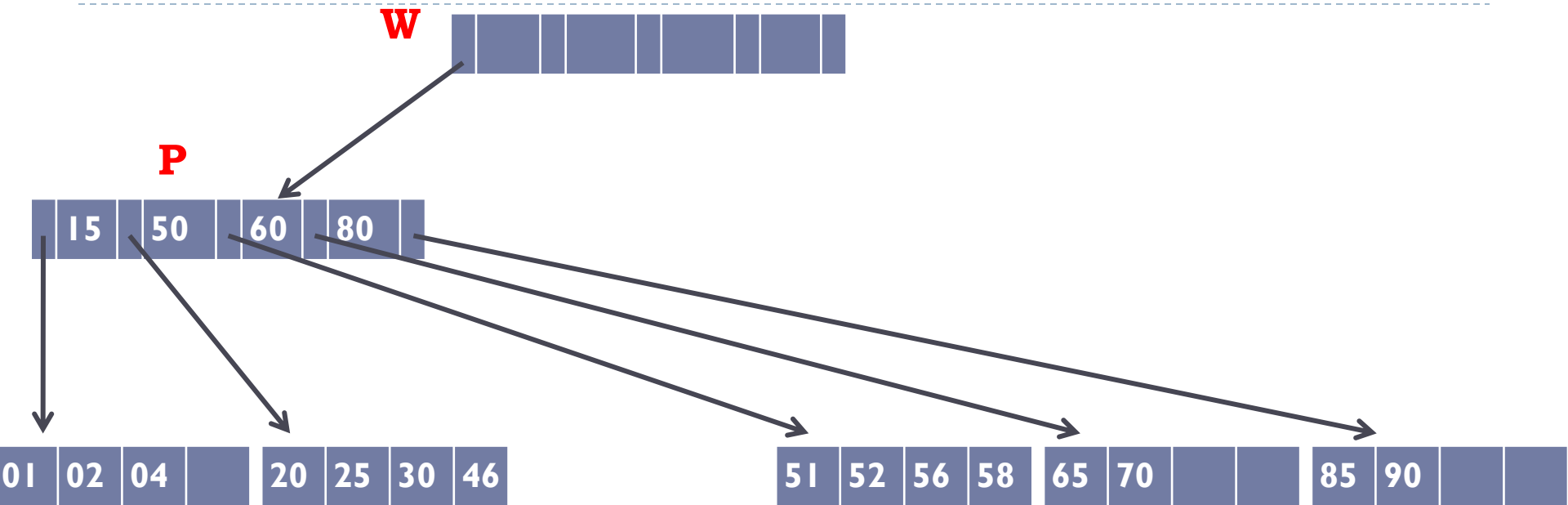
ordem $d = 2$



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

Exemplo: Exclusão da chave 40

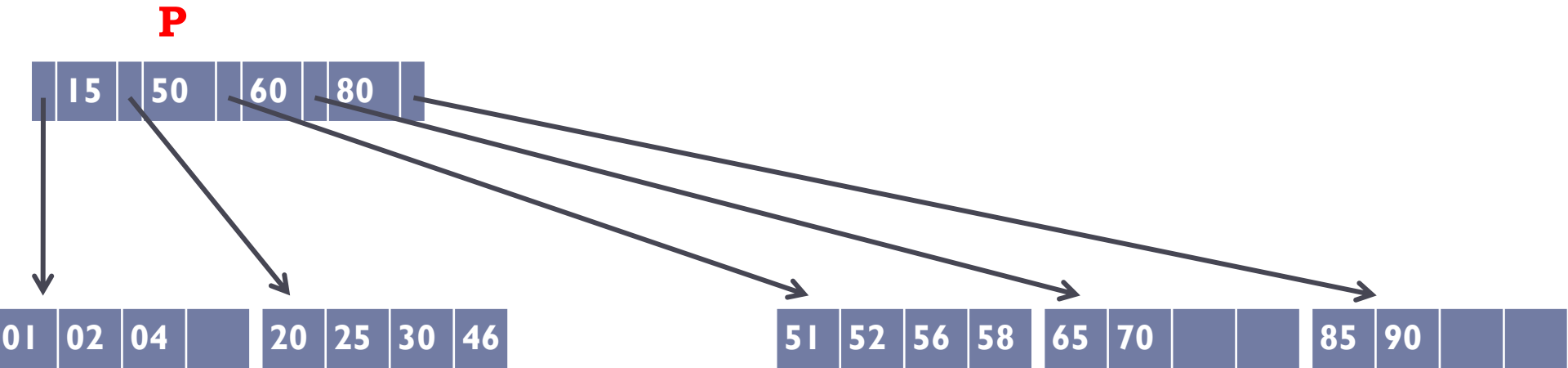
ordem $d = 2$



Eliminar página Q e ponteiro

Exemplo: Exclusão da chave 40

ordem $d = 2$



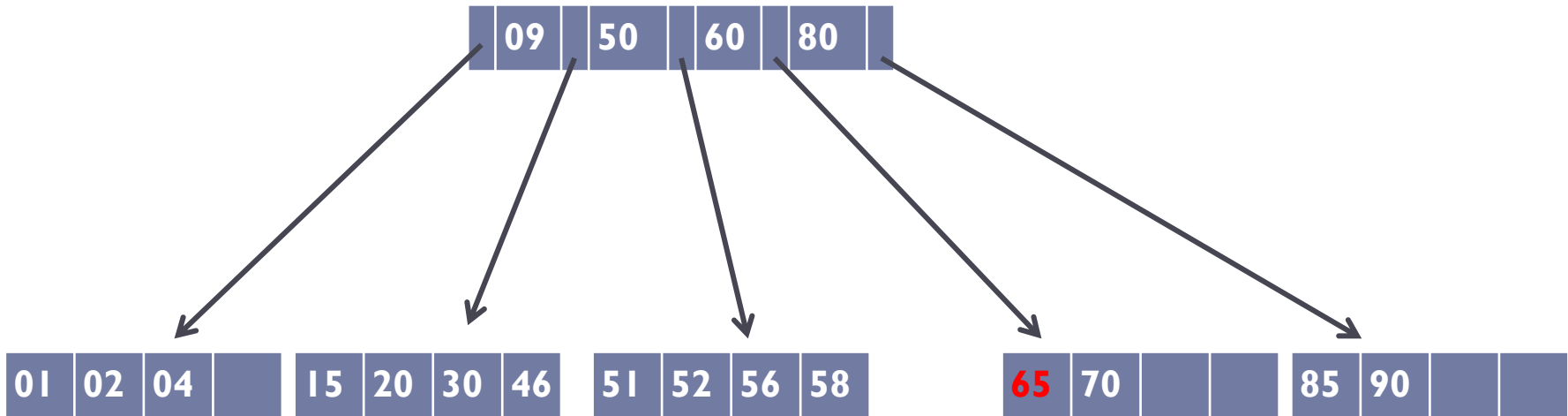
W ficou vazia e era a raiz: eliminá-la
P passa a ser a nova raiz

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

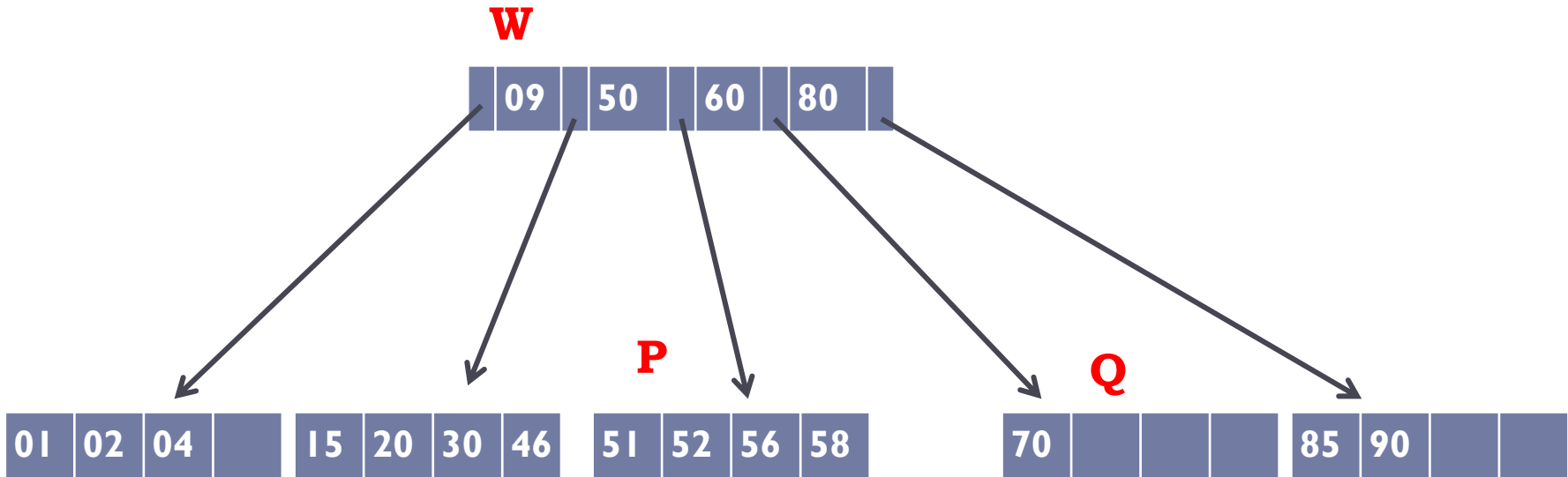
Exemplo: Exclusão da chave 65

ordem $d = 2$



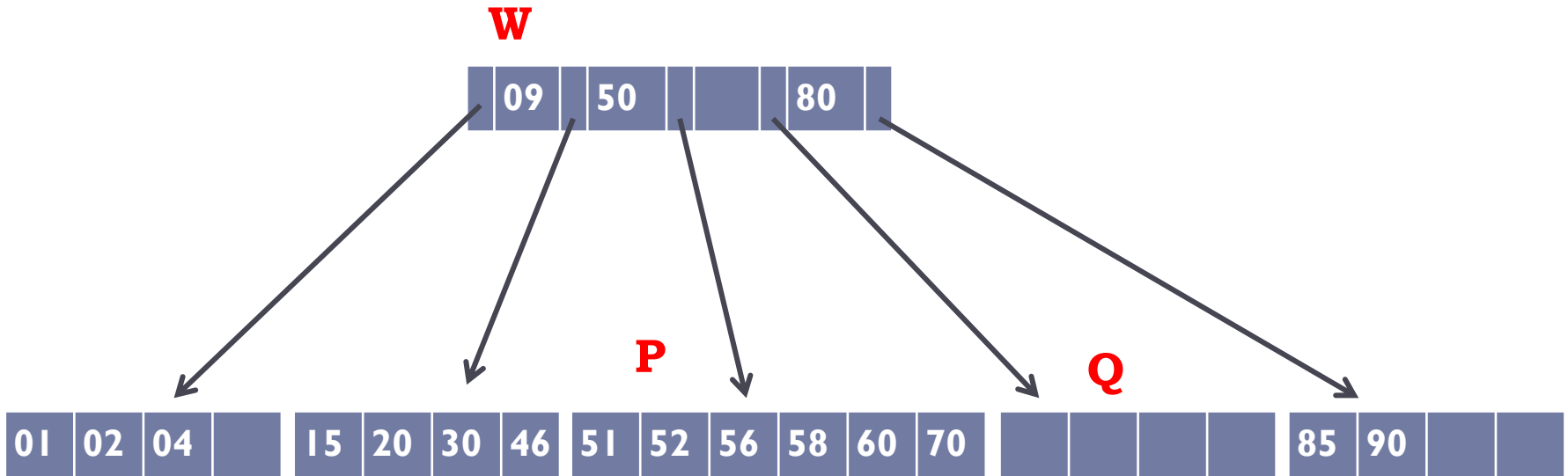
Exemplo: Exclusão da chave 65

ordem $d = 2$



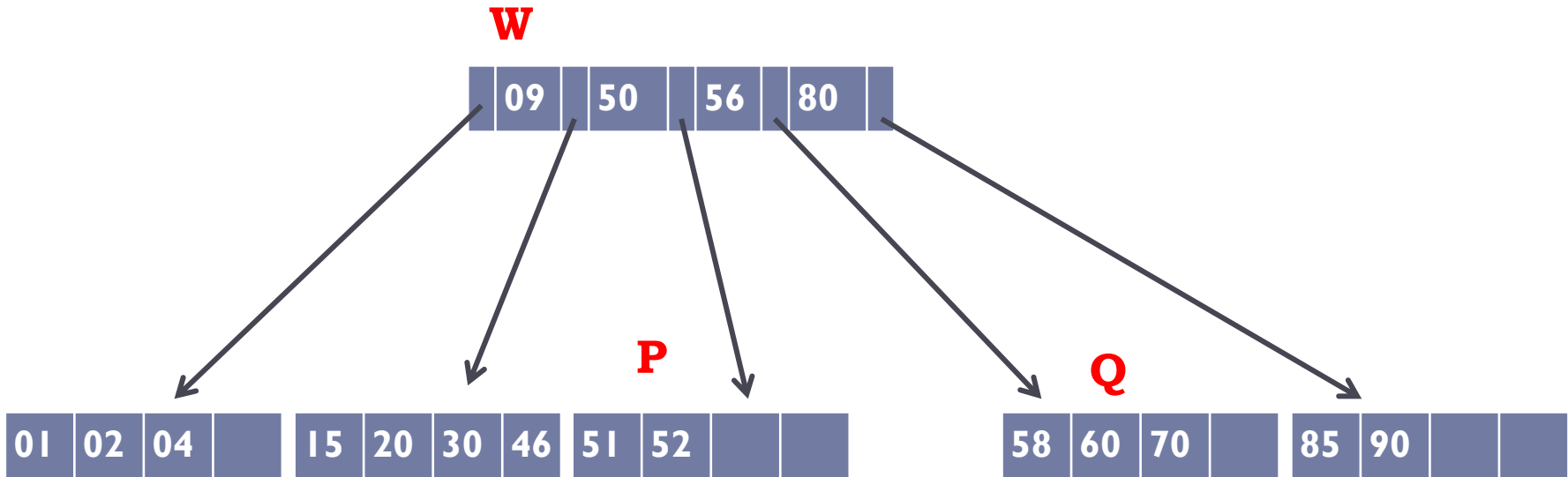
Exemplo: Exclusão da chave 65

ordem $d = 2$



Exemplo: Exclusão da chave 65

ordem $d = 2$



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 (Parte 1)

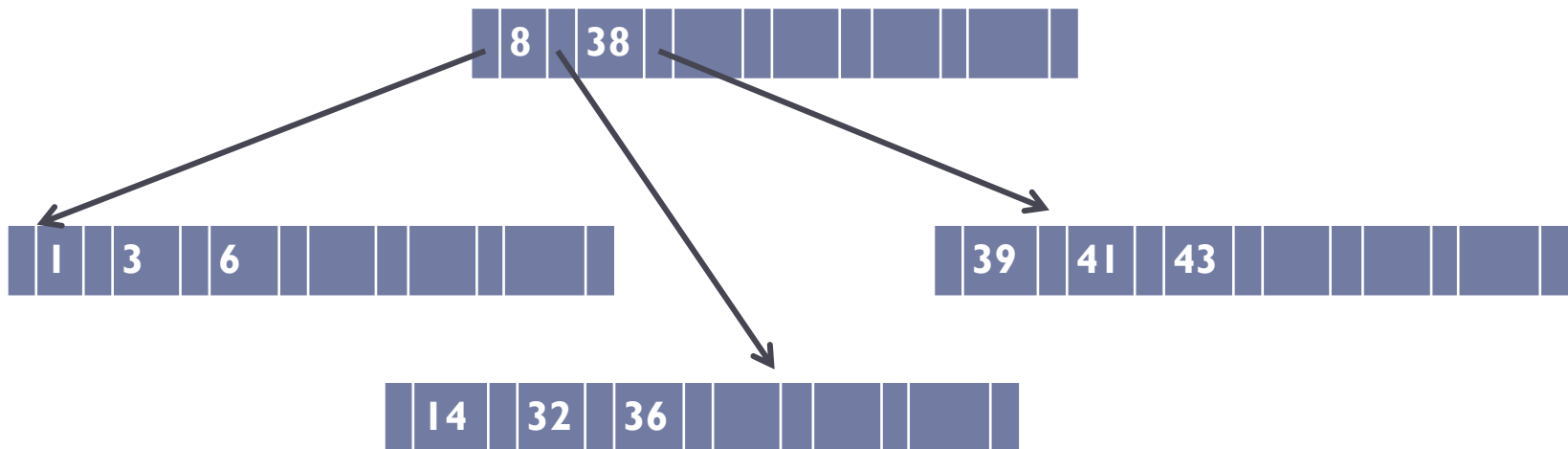
- ▶ Desenhar uma árvore B de ordem 3 que contenha as seguintes chaves: 1, 3, 6, 8, 14, 32, 36, 38, 39, 41, 43
- ▶ Dica: começar com uma árvore B vazia e ir inserindo uma chave após a outra
- ▶ Relembrando características de uma árvore B de ordem d
 - ▶ A raiz é uma folha ou tem no mínimo 2 filhos
 - ▶ 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 (Parte 2)

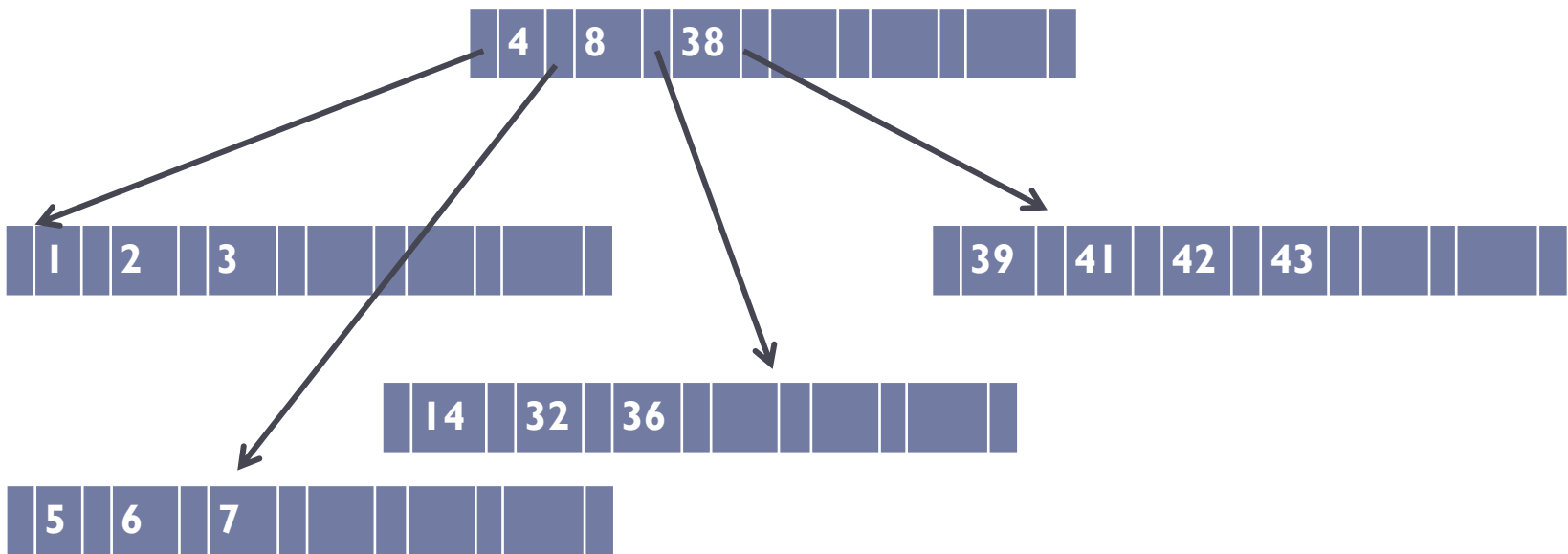
- ▶ Sobre a árvore resultante do exercício anterior, realizar as seguintes operações:
 - (a) Inserir as chaves 4, 5, 42, 2, 7
 - (b) Sobre o resultado do passo (a), excluir as chaves 14, 32

Resposta (Parte 1)

- ▶ 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

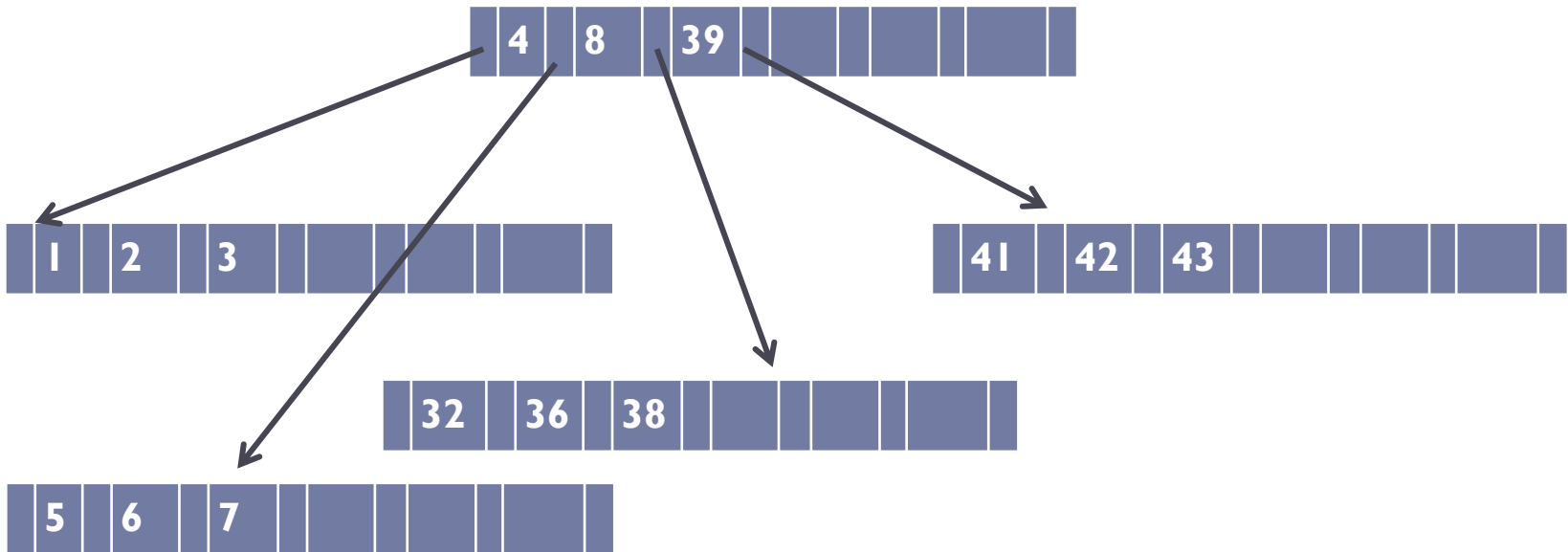


Resposta (Parte 2a) – Inserção de 4, 5, 42, 2, 7



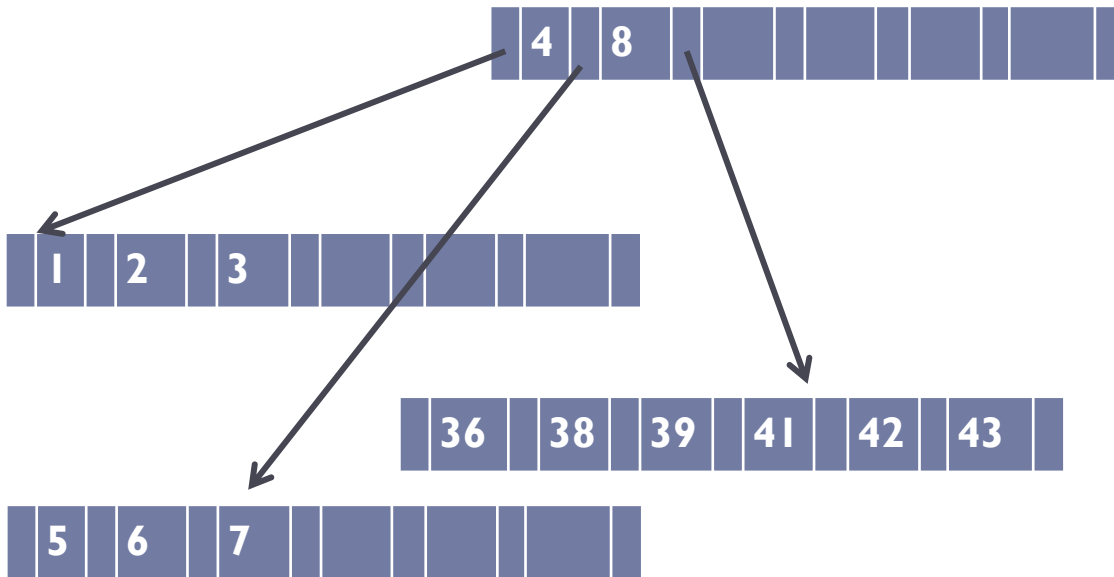
Resposta (Parte 2b) – Exclusão de 14

- ▶ É possível fazer redistribuição



Resposta (Parte 2b) – Exclusão de 32

- ▶ É necessário fazer concatenação



Implementação

- ▶ 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

- ▶ No arquivos de dados, cada nó possui
 - ▶ Inteiro representando o número de chaves (**m**) armazenadas no nó
 - ▶ Um ponteiro para o nó pai
 - ▶ Lista de $m+1$ ponteiros para os nós filho
 - ▶ Lista de m registros

Considerações sobre implementação

- ▶ A cada vez que for necessário manipular um nó, ler o nó todo para a memória, e manipulá-lo em memória
- ▶ Depois, gravar o nó todo de volta no disco

- ▶ Na nossa disciplina, vamos simplificar:
 - ▶ da mesma forma que fazíamos uma função para ler um registro e gravar um registro inteiro, agora faremos uma função que lê uma página e grava uma página inteira no disco

Árvores B*

Árvores B*

- ▶ É uma variação da árvore B
 - ▶ Todos os nós, exceto a raiz, precisam estar $2/3$ cheios (em contraste com $1/2$ exigido pela árvore B)
 - ▶ Para manter esta propriedade, os nós não são particionados logo que ficam cheios. Ao invés disso, suas chaves são compartilhadas com o nó vizinho, até que ambos fiquem cheios. Neste ponto, os dois nós são divididos em 3 nós
- ▶ Na prática, não é muito utilizada

Árvores B+

Árvores B+

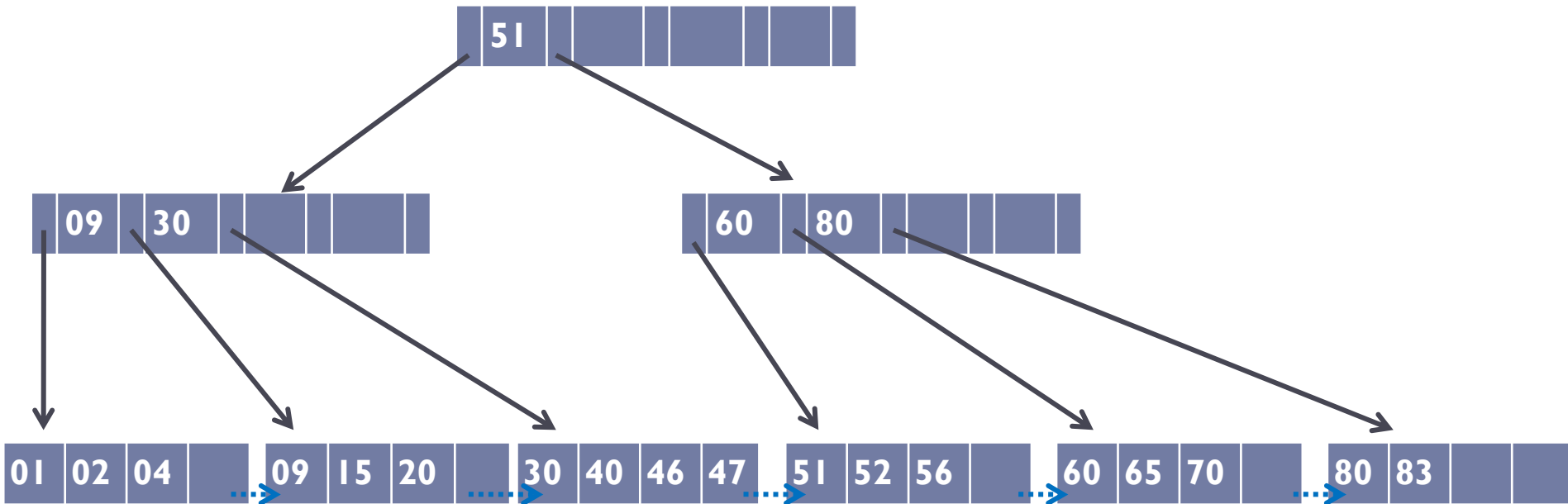
- ▶ É semelhante à árvore B, exceto por duas características muito importantes:
 - ▶ Armazena dados somente nas folhas – os nós internos servem apenas de ponteiros
 - ▶ As folhas são encadeadas

- ▶ Isso permite o armazenamento dos dados em um arquivo, e do índice em outro arquivo separado

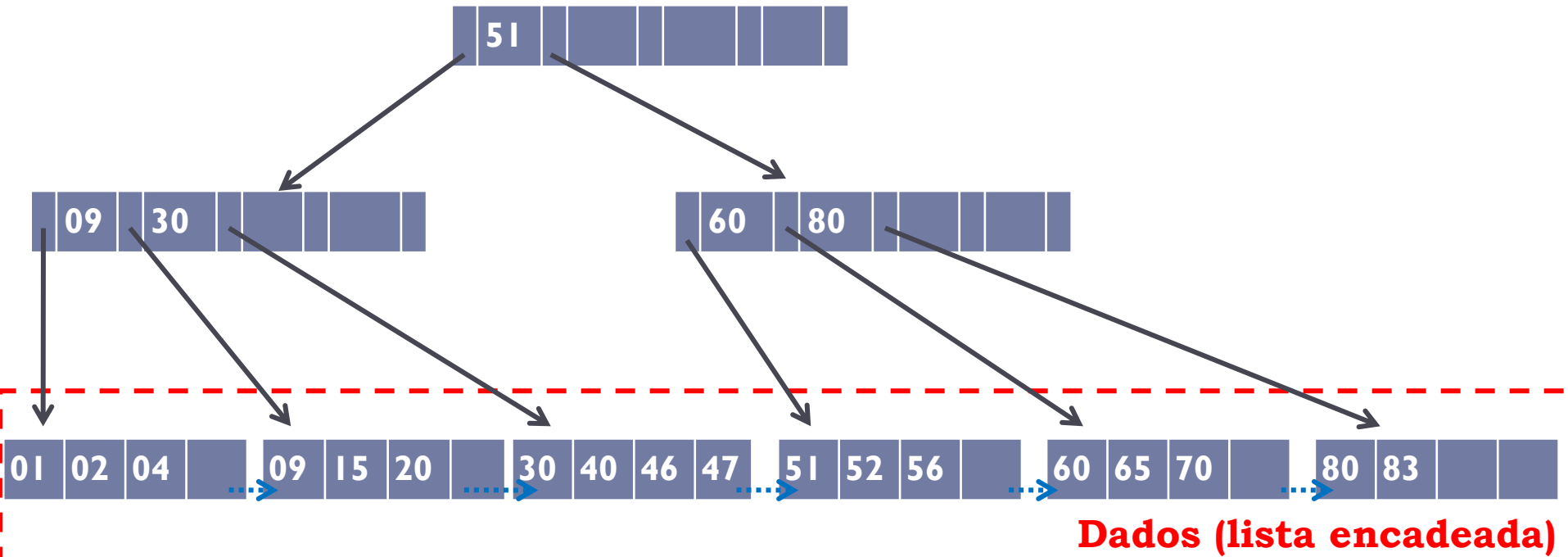
Árvore B+ na prática

- ▶ Árvores B+ são muito importantes por sua eficiência, e muito utilizadas na prática:
 - ▶ Os sistemas de arquivo **NTFS**, **ReiserFS**, **NSS**, **XFS**, e **JFS** utilizam este tipo de árvore para indexação
 - ▶ Sistemas de Gerência de Banco de Dados como **IBM DB2**, **Informix**, **Microsoft SQL Server**, **Oracle 8**, **Sybase ASE**, **PostgreSQL**, **Firebird**, **MySQL** e **SQLite** permitem o uso deste tipo de árvore para indexar tabelas
 - ▶ Outros sistemas de gerência de dados como o **CouchDB**, **Tokyo Cabinet** e **Tokyo Tyrant** permitem o uso deste tipo de árvore para acesso a dados

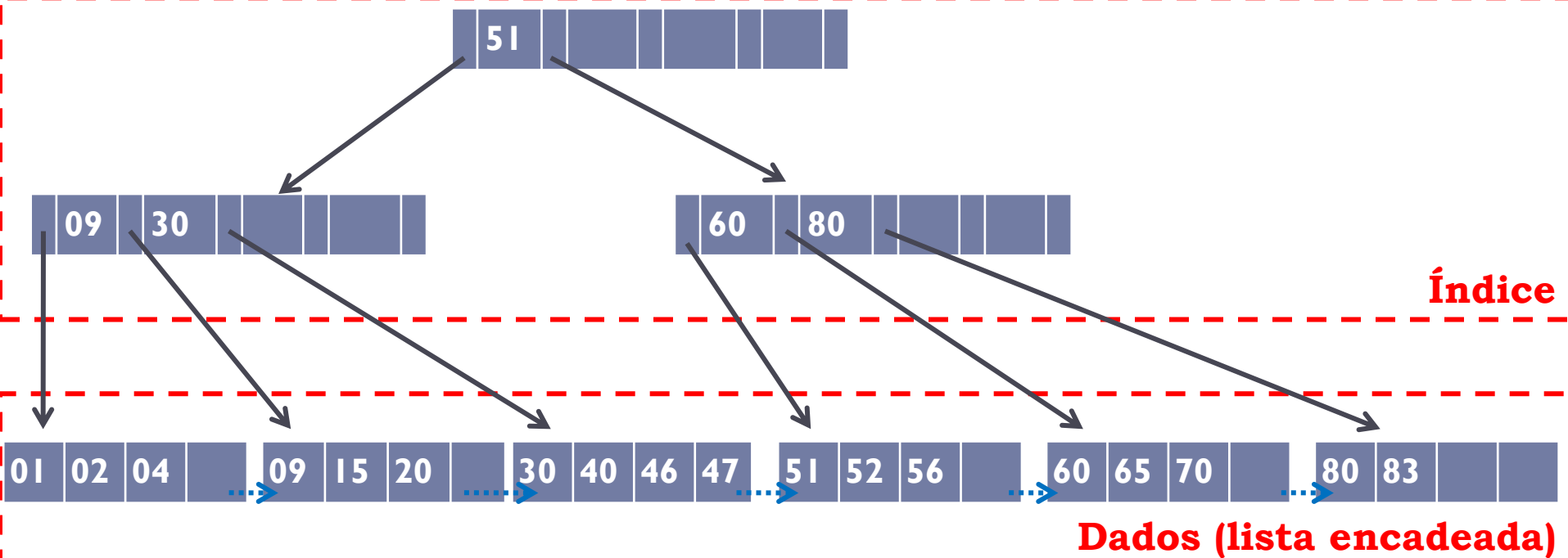
Exemplo de Árvore B+



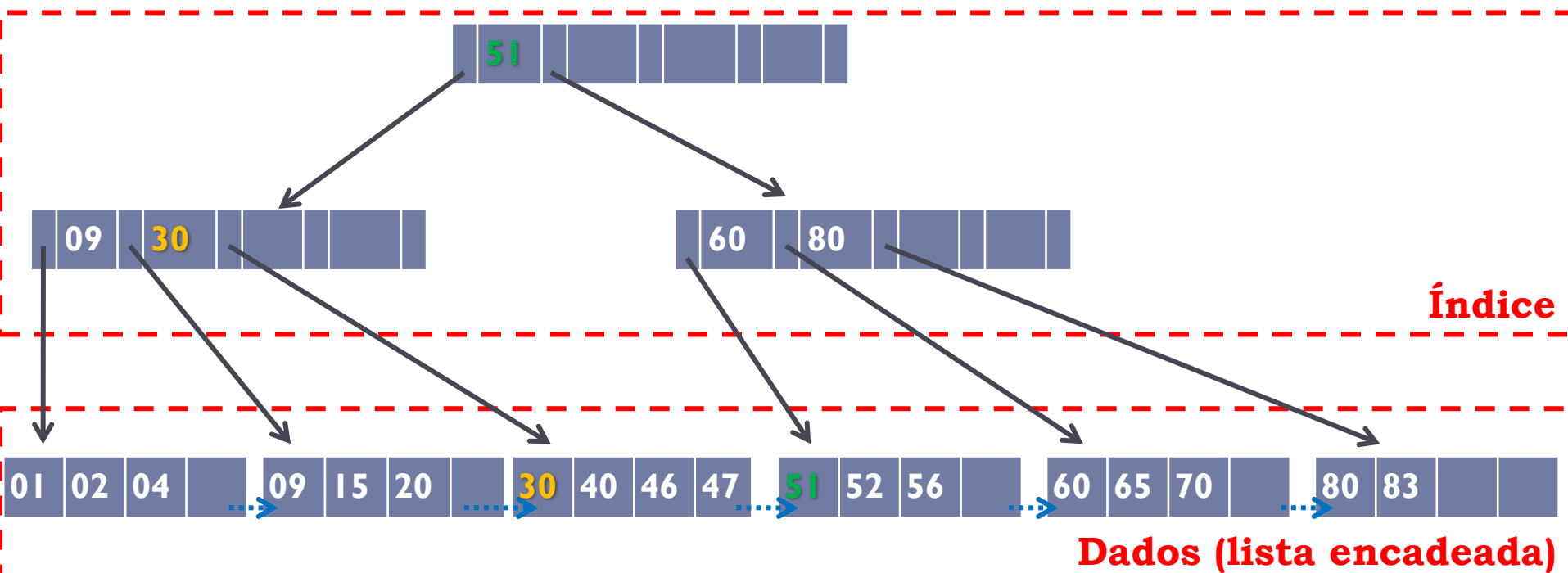
Exemplo de Árvore B+



Exemplo de Árvore B+



Exemplo de Árvore B+



▶ IMPORTANTE:

- ▶ Os valores nos índices repetem valores de chave que aparecem nas folhas (diferente do que acontece nas árvores B)

Busca

- ▶ Só se pode ter certeza de que o registro foi encontrado quando se chega em uma folha

Inserção

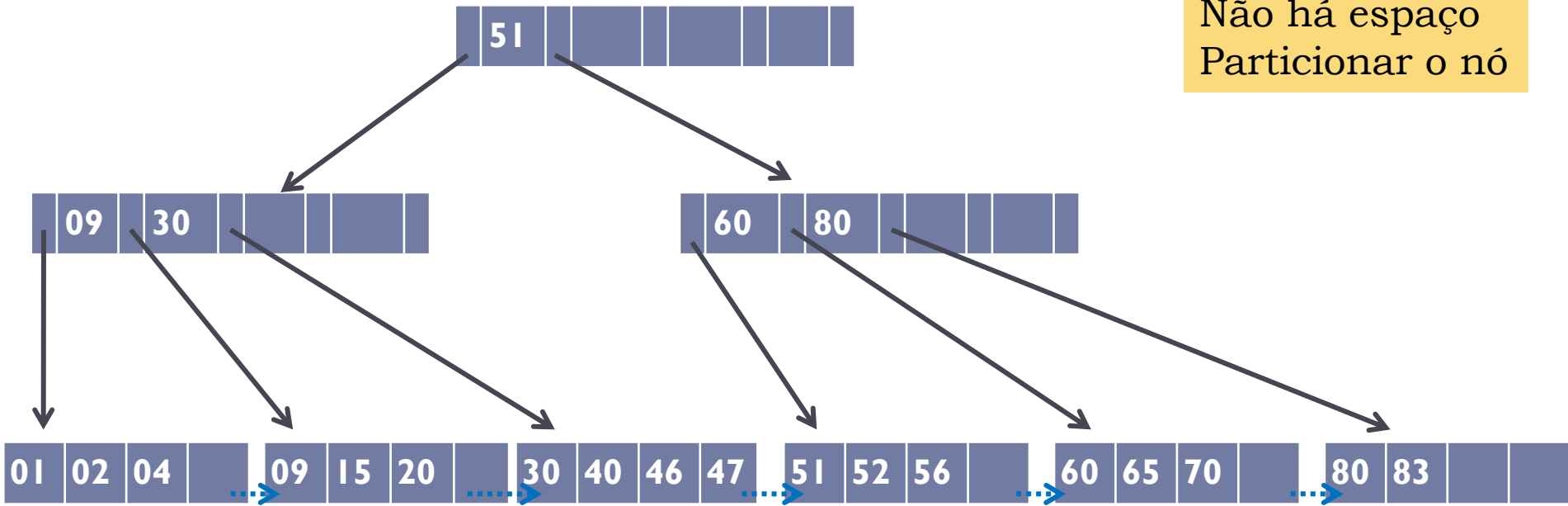
- ▶ Quando for necessário particionar um nó durante uma inserção, o mesmo raciocínio é utilizado
 - ▶ A diferença é que para a página pai sobe somente a chave. O registro fica na folha, juntamente com a sua chave
 - ▶ **ATENÇÃO:** isso vale apenas se o nó que está sendo particionado for uma folha. Se não for folha, o procedimento é o mesmo utilizado na árvore B

Exemplo de Inserção em Árvore B+

Inserir chave 32

ordem $d = 2$

Não há espaço
Particionar o nó

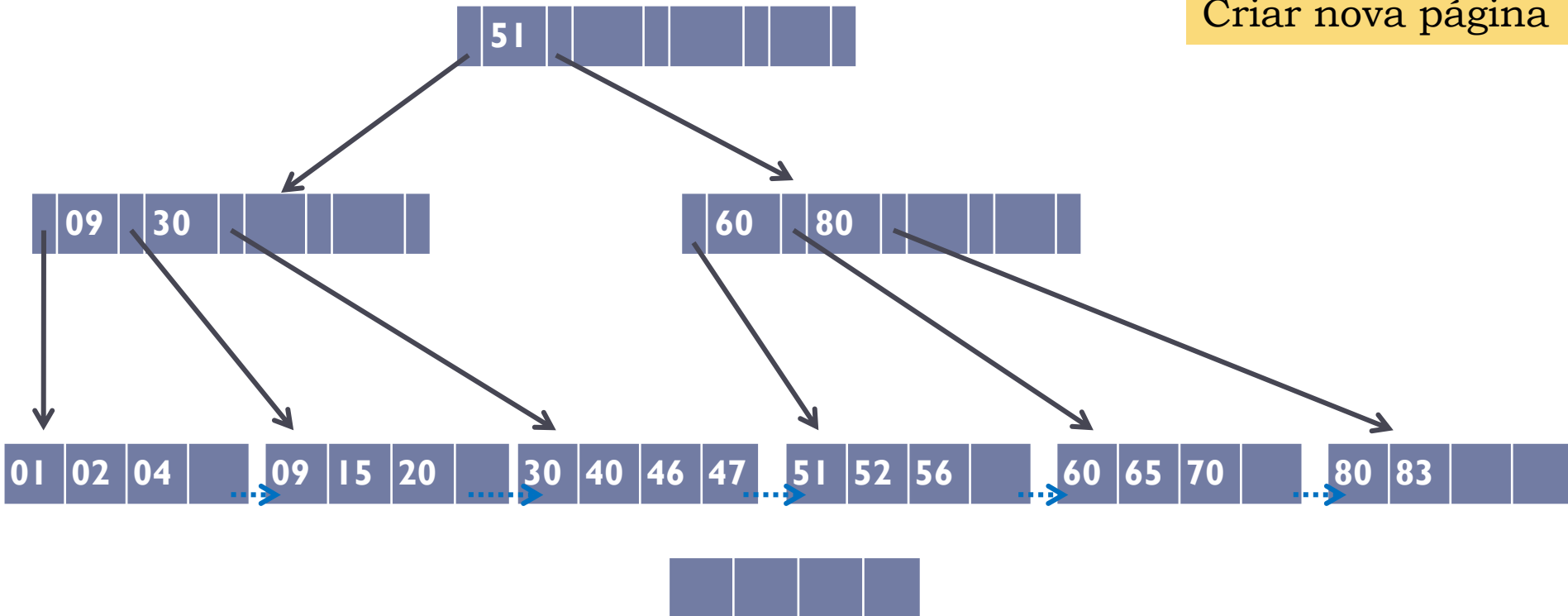


Exemplo de Inserção em Árvore B+

Inserir chave 32

ordem $d = 2$

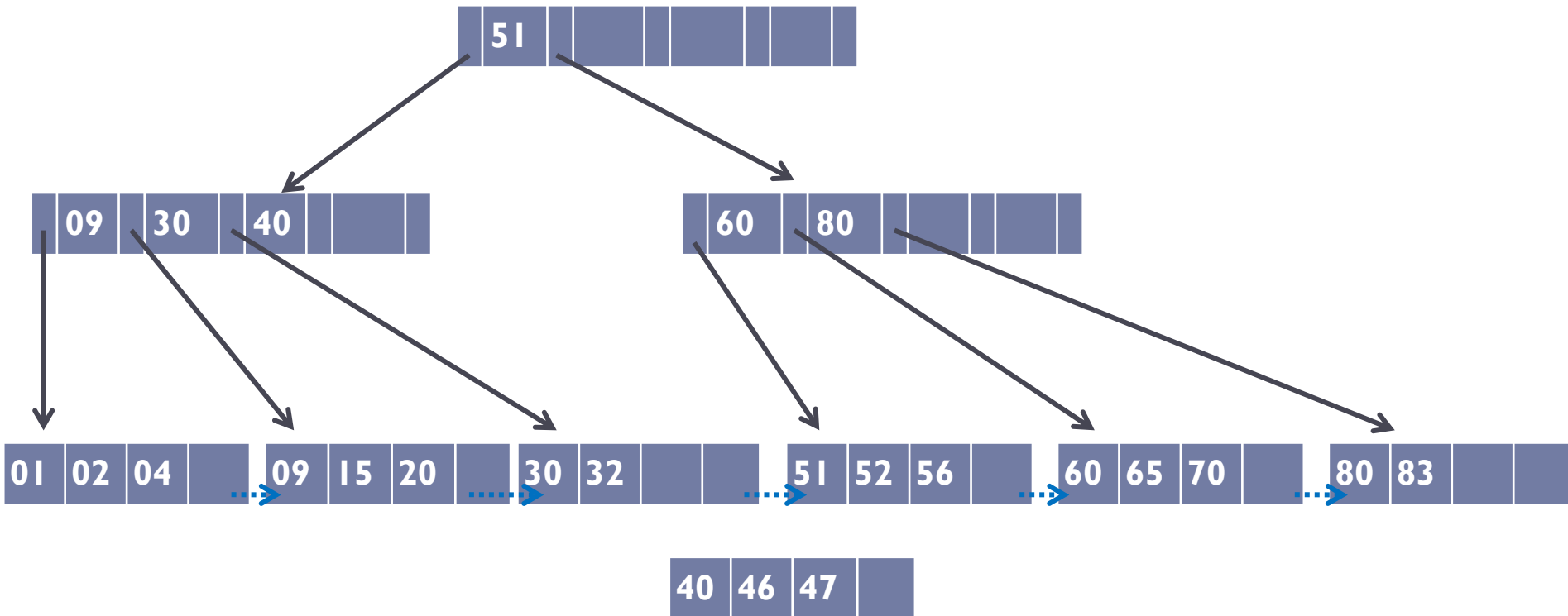
Criar nova página



Exemplo de Inserção em Árvore B+

Inserir chave 32

ordem $d = 2$

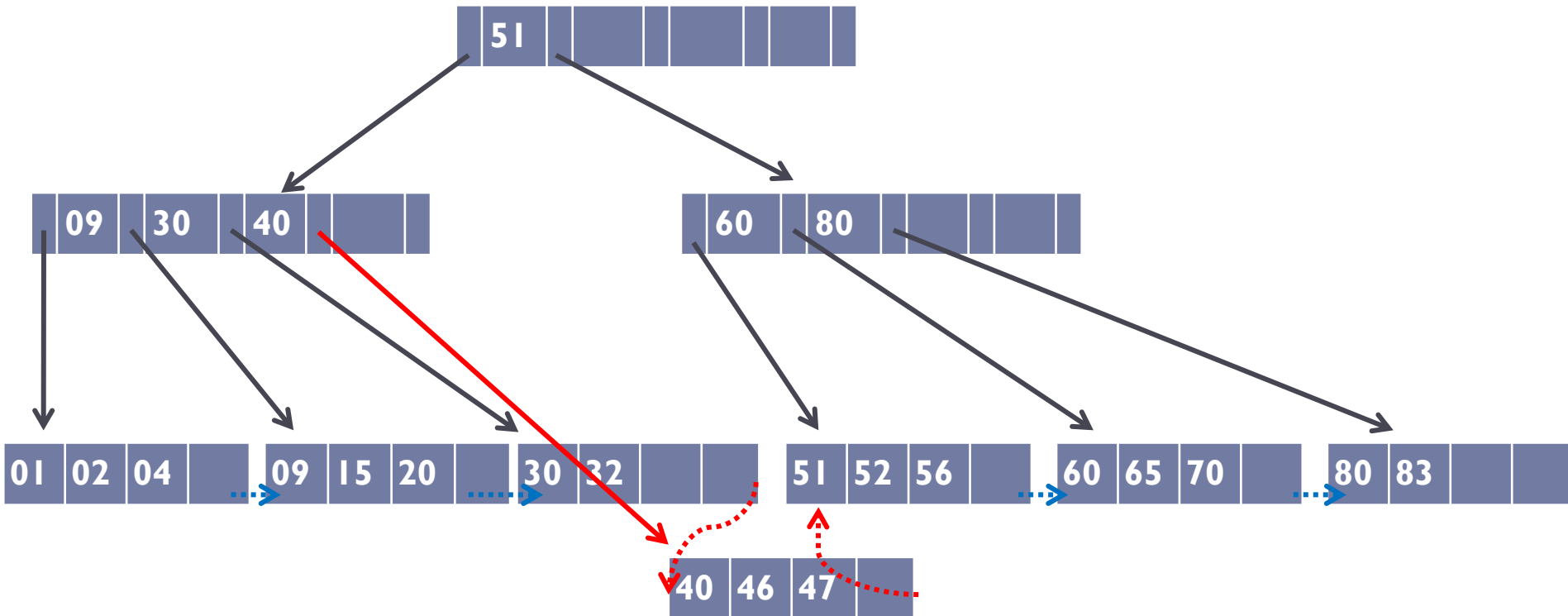


Dividir as chaves entre as duas páginas (30; 32; 40; 46; 47)
d chaves na página original
chave $d+1$ sobe para nó pai (**mas registro é mantido na nova página**)
 $d+1$ chaves restantes na nova página

Exemplo de Inserção em Árvore B+

Inserir chave 32

ordem $d = 2$



Ajustar ponteiros

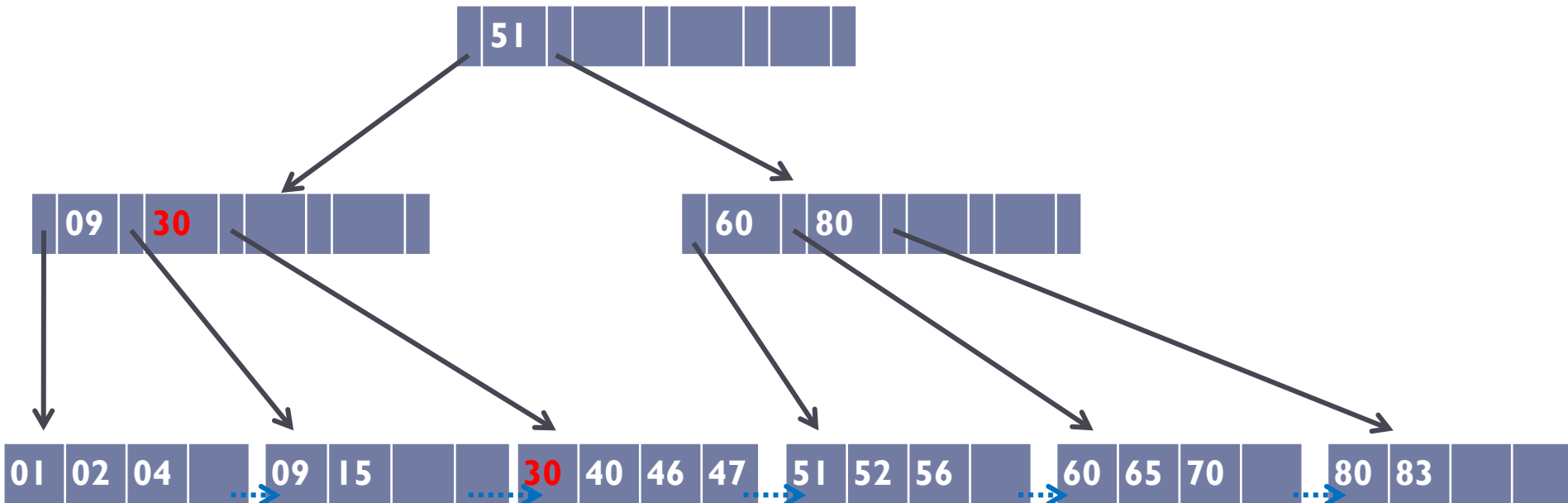
Exclusão

- ▶ Excluir apenas no nó folha
- ▶ Chaves excluídas continuam nos nós intermediários

Exemplo de Exclusão em Árvore B+

Excluir chave 30

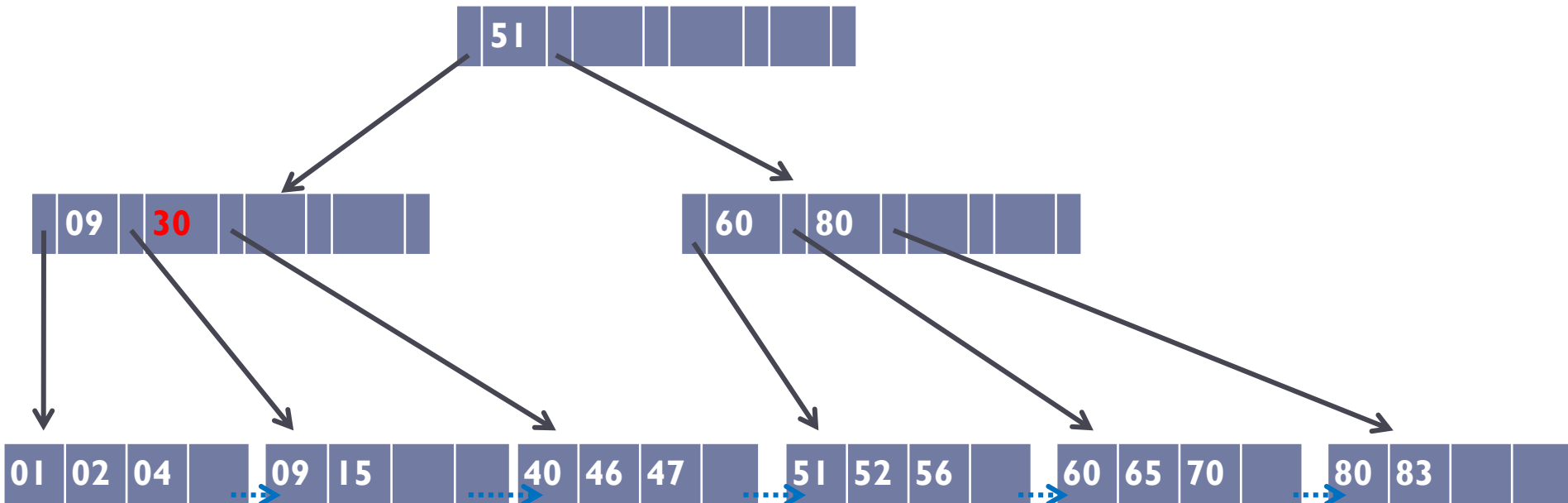
ordem $d = 2$



Exemplo de Exclusão em Árvore B+

Excluir chave 30

ordem $d = 2$



O valor 30 continua no índice!

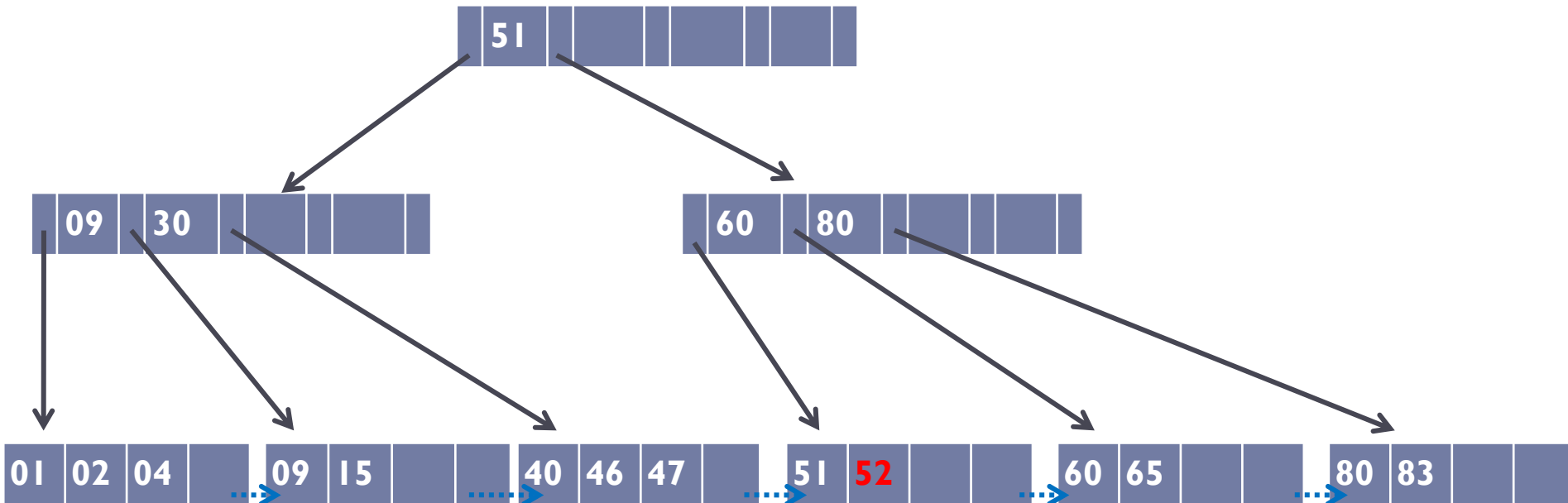
Exclusão que causa concatenação

- ▶ Exclusões que causem concatenação de folhas podem se propagar para os nós internos da árvore
- ▶ **Importante:**
 - ▶ Se a concatenação ocorrer na folha: a chave do nó pai não desce para o nó concatenado, pois ele não carrega dados com ele. Ele é simplesmente apagado.
 - ▶ Se a concatenação ocorrer em nó interno: usar a mesma lógica utilizada na árvore B

Exemplo de Exclusão em Árvore B+

Excluir chave 52

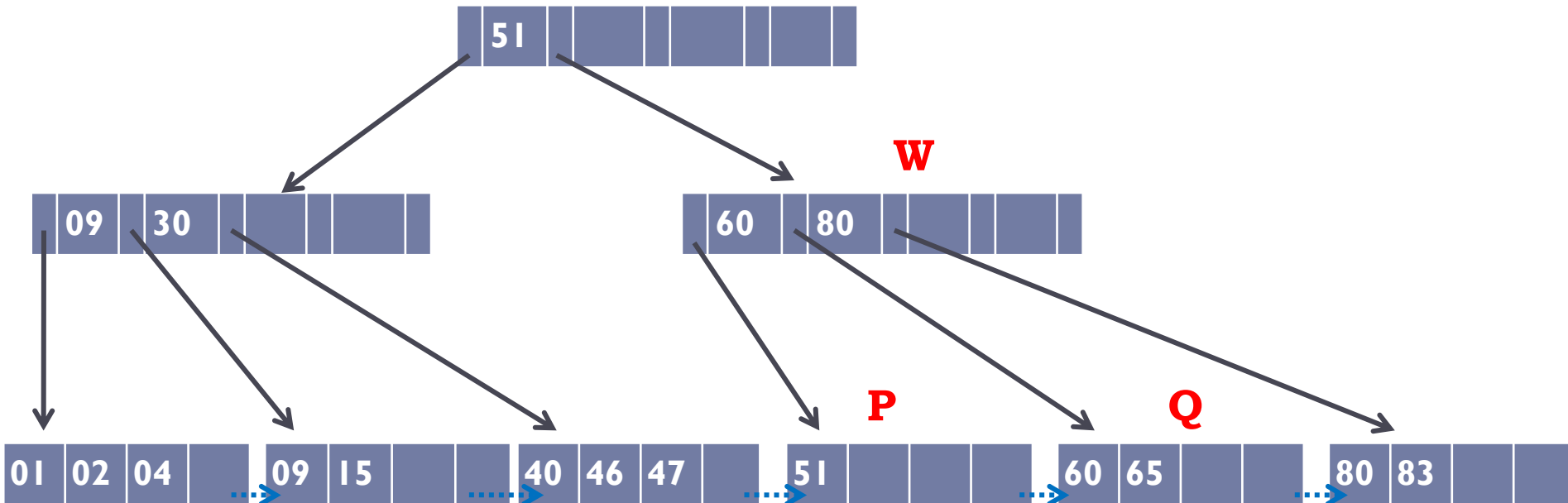
ordem $d = 2$



Exemplo de Exclusão em Árvore B+

Excluir chave 52

ordem $d = 2$

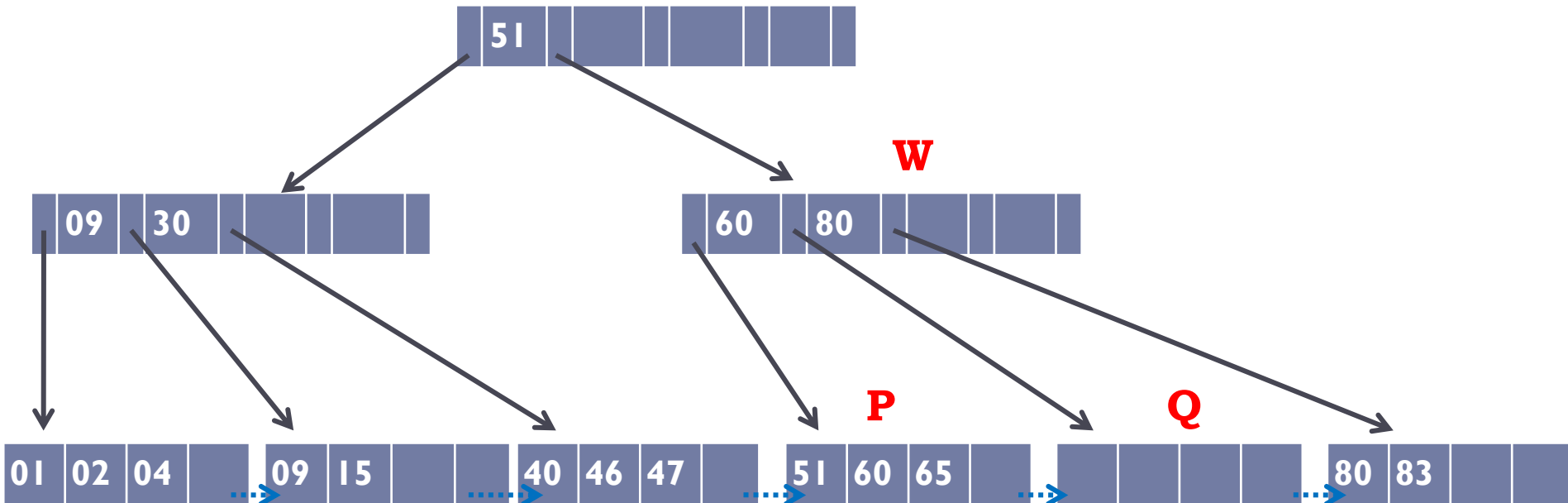


Nó ficou com menos de d entradas – necessário tratar isso
Soma dos registros de P e Q $< 2d$
Usar concatenação

Exemplo de Exclusão em Árvore B+

Excluir chave 52

ordem $d = 2$



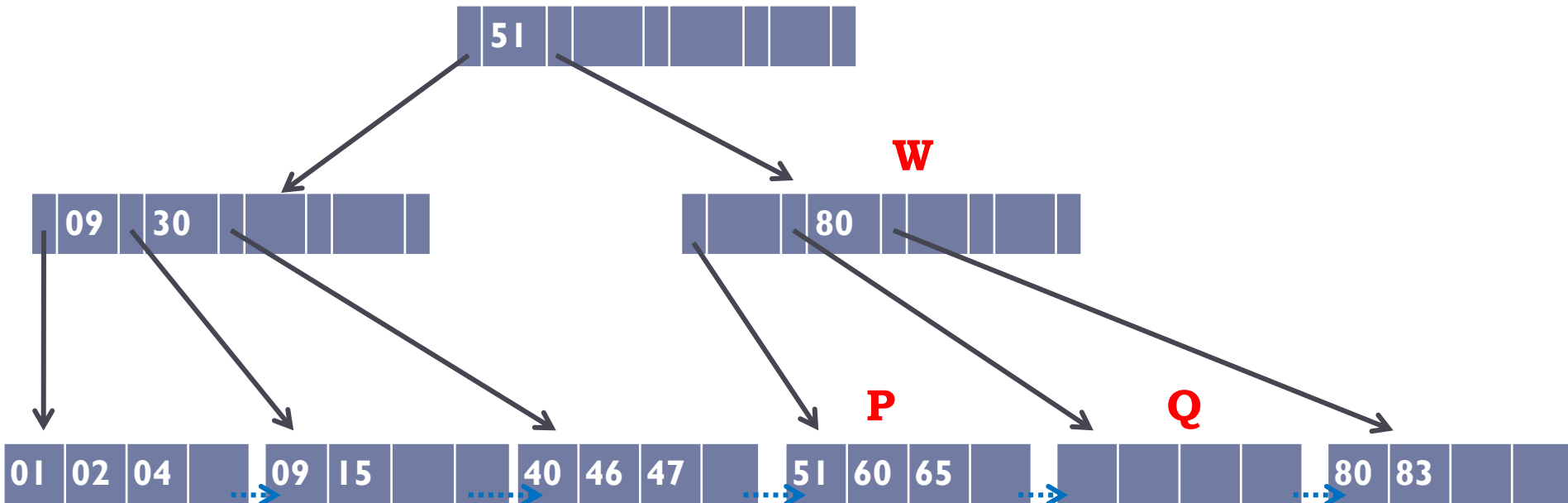
Passar os registros de Q para P

Eliminar a chave em W que divide os ponteiros para as páginas P e Q

Exemplo de Exclusão em Árvore B+

Excluir chave 52

ordem $d = 2$

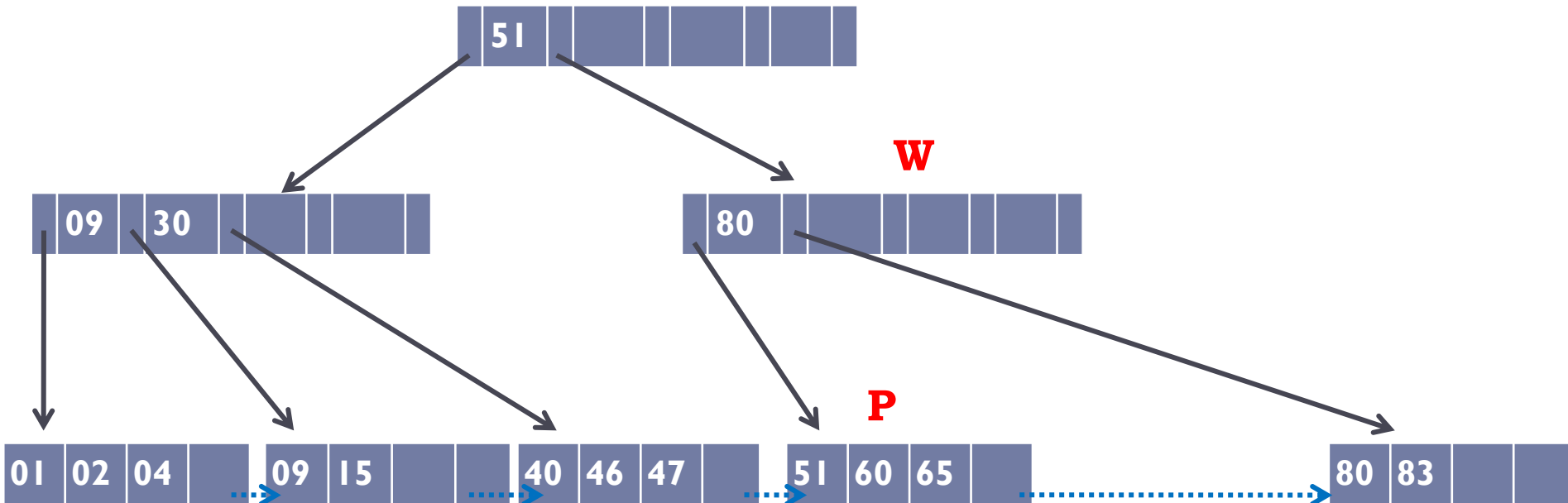


Eliminar ponteiro e nó Q

Exemplo de Exclusão em Árvore B+

Excluir chave 52

ordem $d = 2$

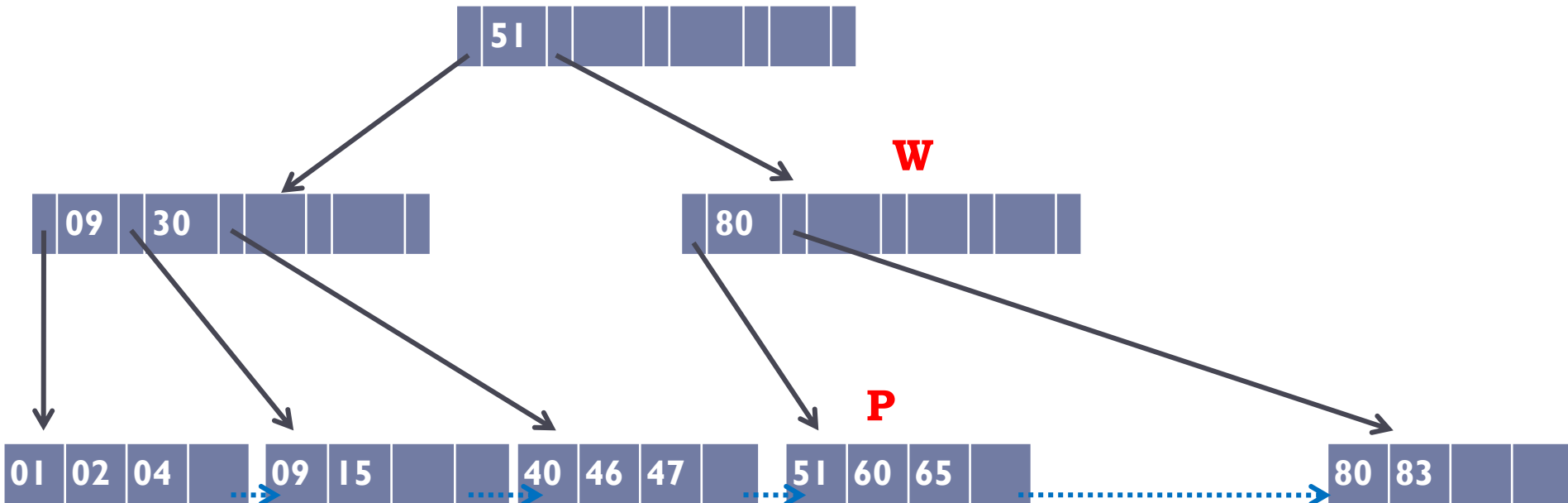


Eliminar ponteiro e nó Q, reajustar ponteiros e nó W

Exemplo de Exclusão em Árvore B+

Excluir chave 52

ordem $d = 2$

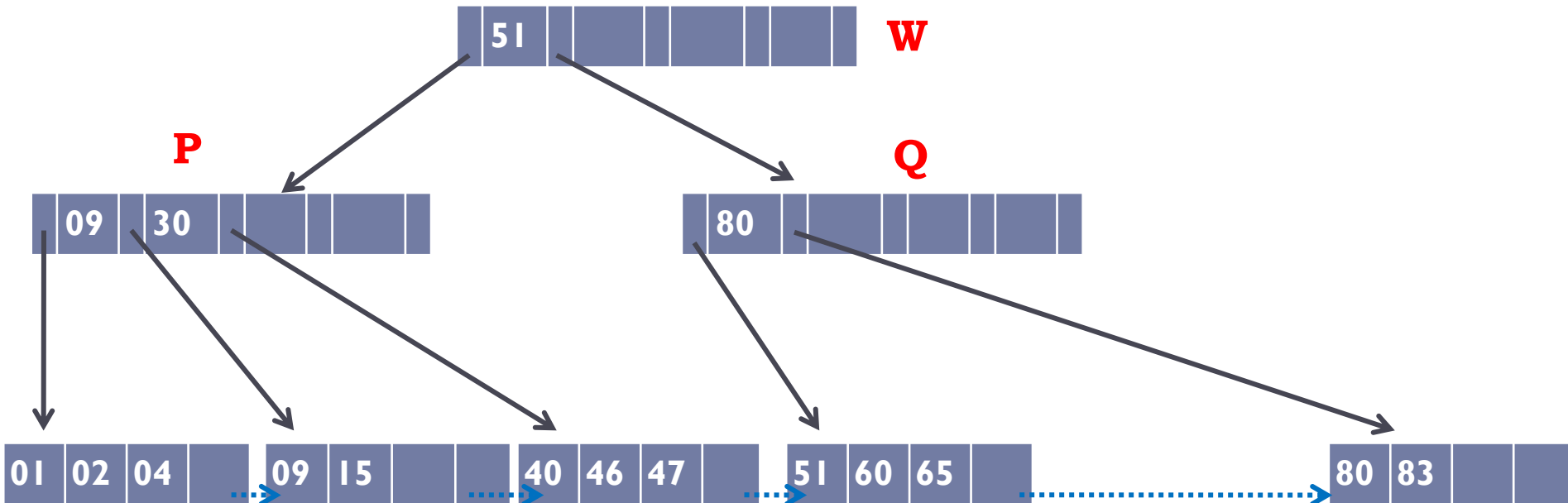


Nó W ficou com menos de d chaves

Exemplo de Exclusão em Árvore B+

Excluir chave 52

ordem $d = 2$

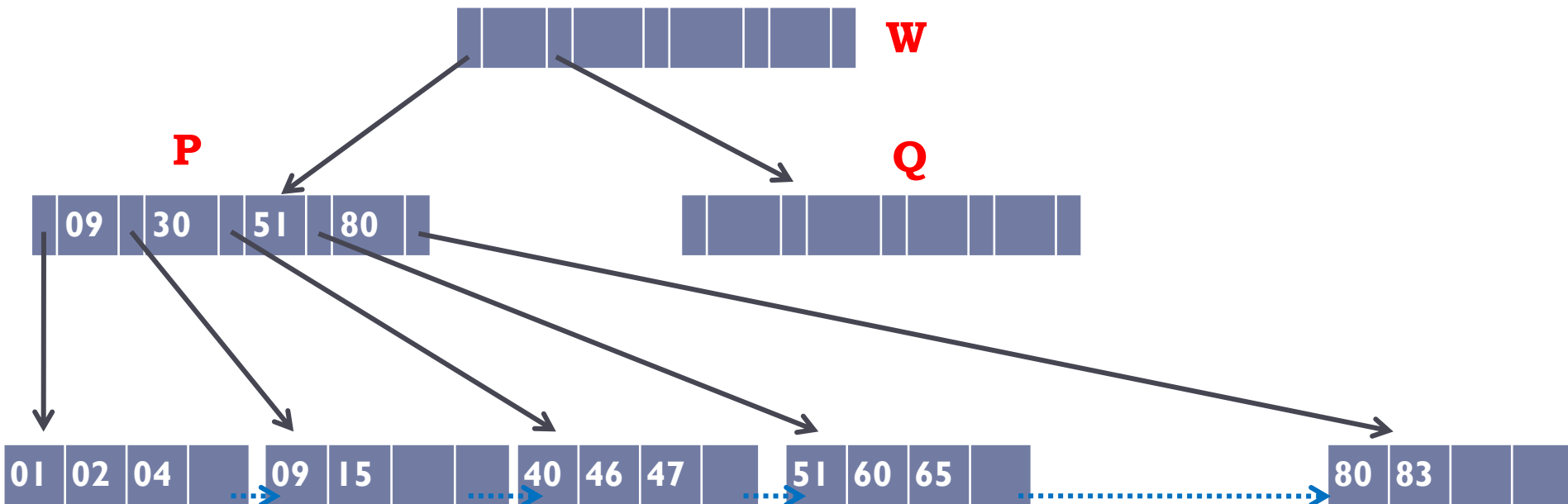


Soma de total de chaves de P e Q $< 2d$
Solução: concatenação

Exemplo de Exclusão em Árvore B+

Excluir chave 52

ordem $d = 2$



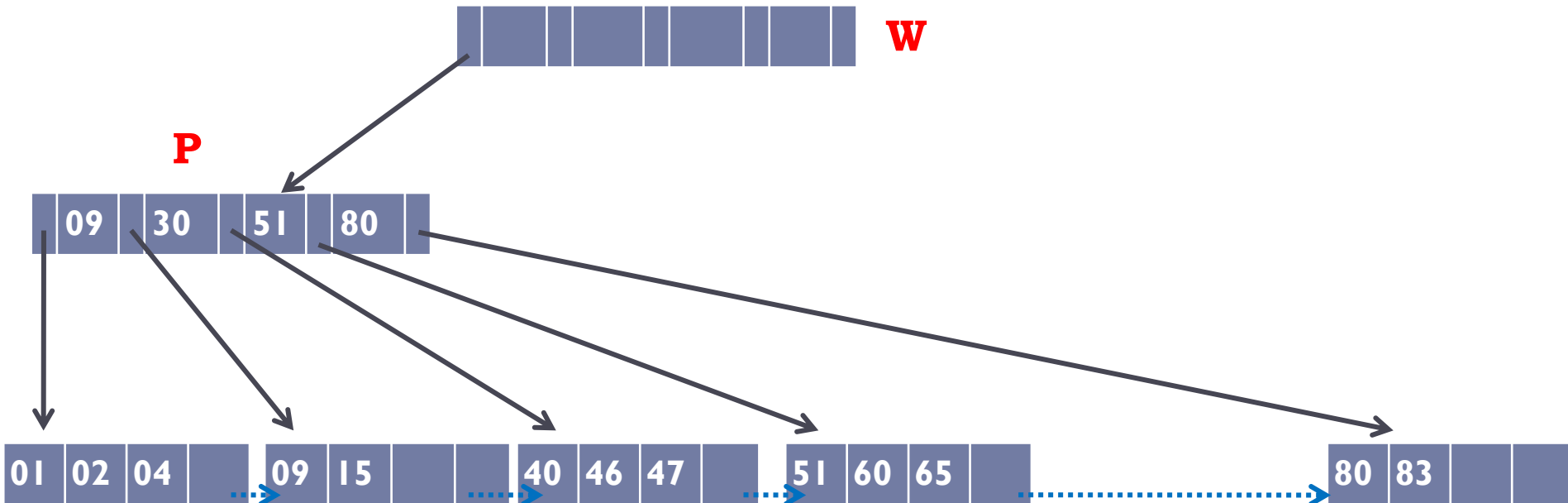
Transferir chaves para P

Atenção: com as páginas concatenadas não são folhas, chave em W também desce para P! (caso contrário, faltaria chave para separar os ponteiros para os filhos)

Exemplo de Exclusão em Árvore B+

Excluir chave 52

ordem $d = 2$

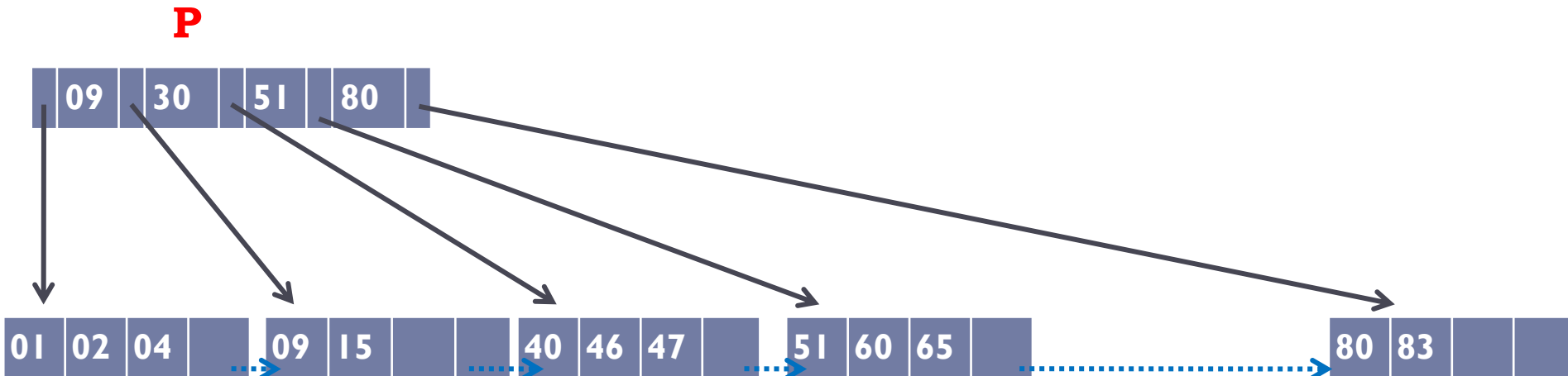


Apagar Q

Exemplo de Exclusão em Árvore B+

Excluir chave 52

ordem $d = 2$



Como a raiz ficou vazia, apagar a raiz. P é a nova raiz.

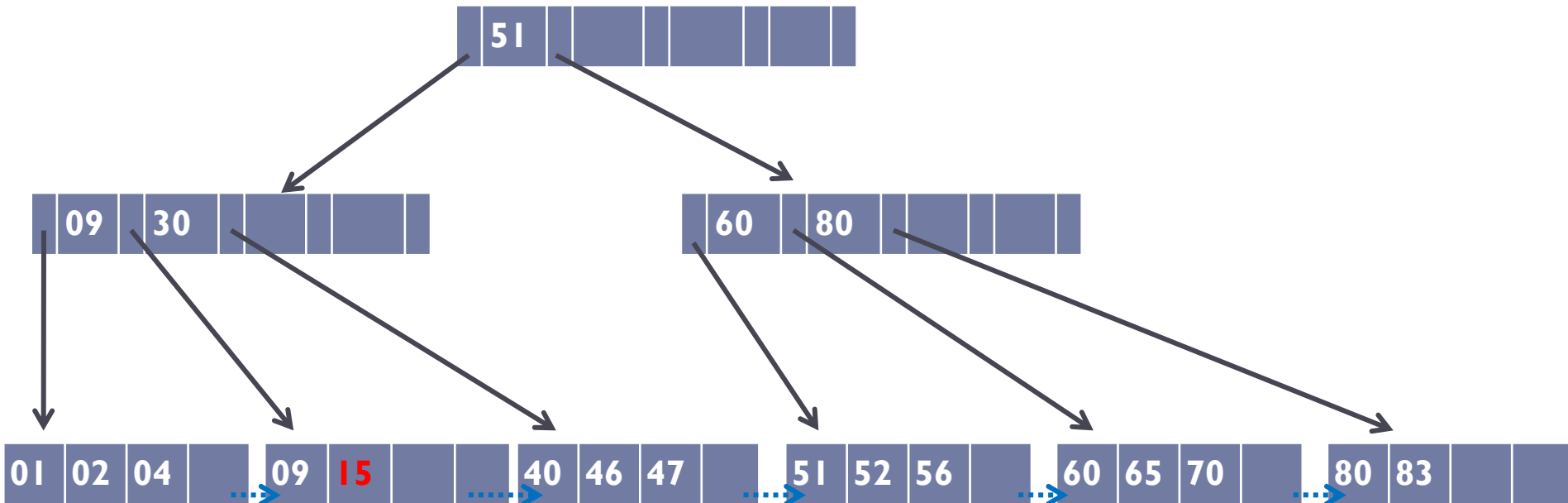
Exclusão que causa redistribuição

- ▶ Exclusões que causem redistribuição dos registros nas folhas provocam mudanças no conteúdo do índice, mas não na estrutura (não se propagam)

Exemplo de Exclusão em Árvore B+

Excluir chave 15

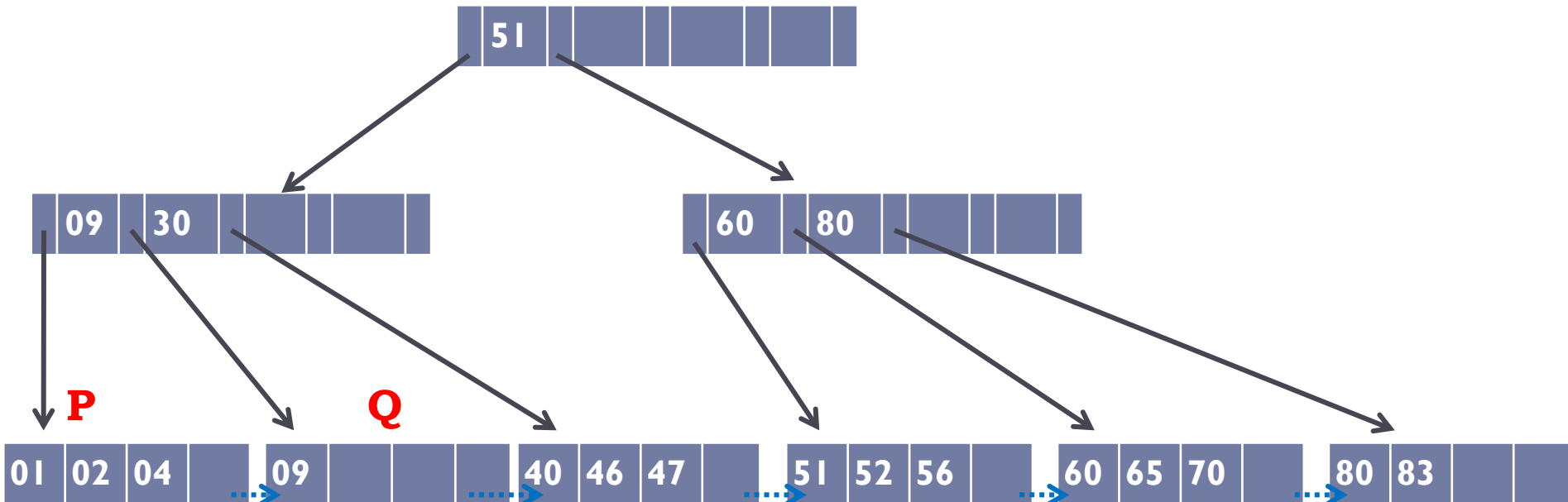
ordem $d = 2$



Exemplo de Exclusão em Árvore B+

Excluir chave 15

ordem $d = 2$

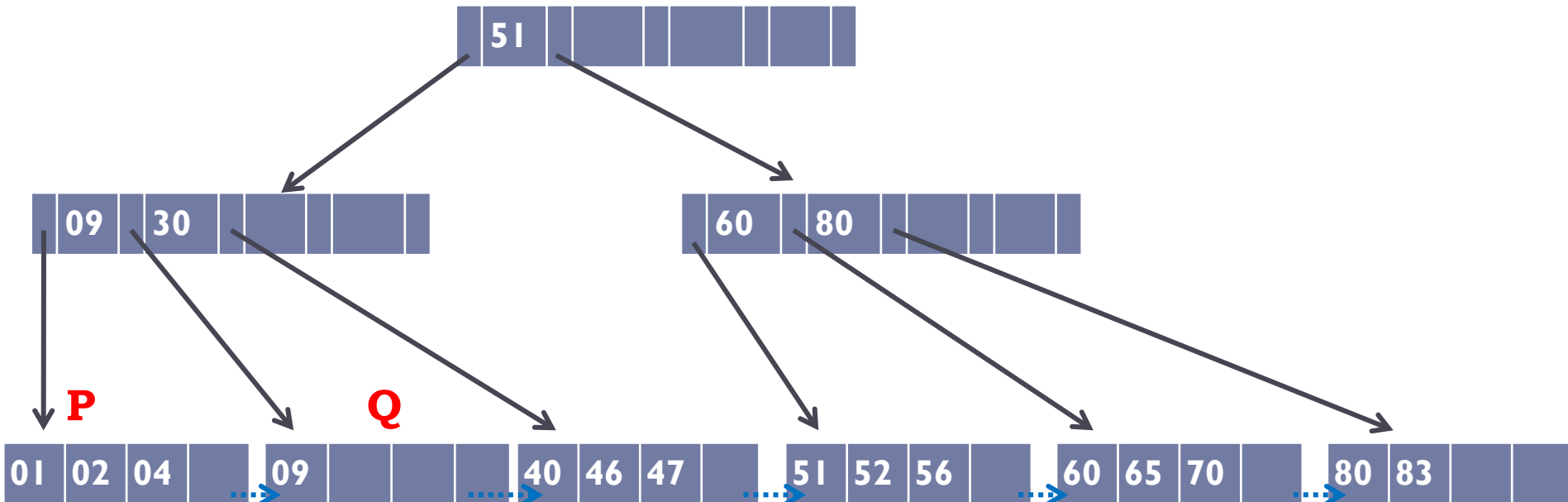


Nó ficou com menos de d entradas – necessário tratar isso
P e Q não podem ser concatenadas, pois a soma dos registros não é menor $2d$
Solução: redistribuição

Exemplo de Exclusão em Árvore B+

Excluir chave 15

ordem $d = 2$



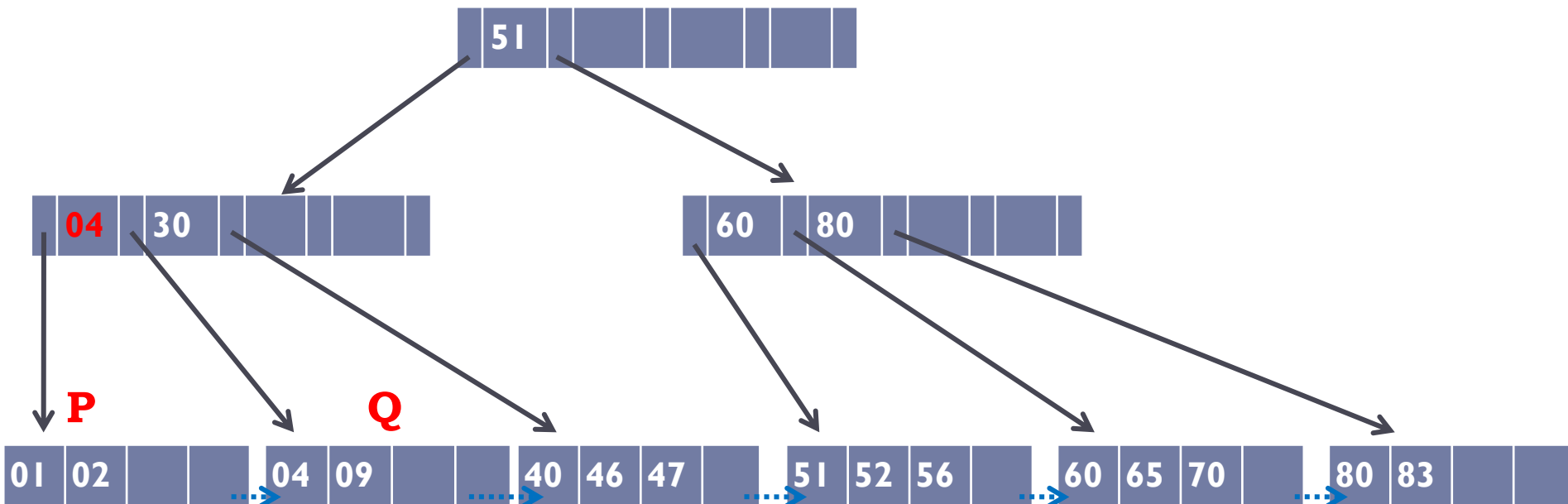
MAS... Se a chave do nó pai não precisa descer (porque não tem conteúdo, tem apenas a chave), porque não podemos concatenar P e Q?

Resposta: ao concatenar P e Q, a página concatenada ficaria cheia, e a próxima inserção neste nó causaria um particionamento. Para evitar isso, continuamos obedecendo o critério : soma da quantidade de chaves $< 2d$

Exemplo de Exclusão em Árvore B+

Excluir chave 15

ordem $d = 2$

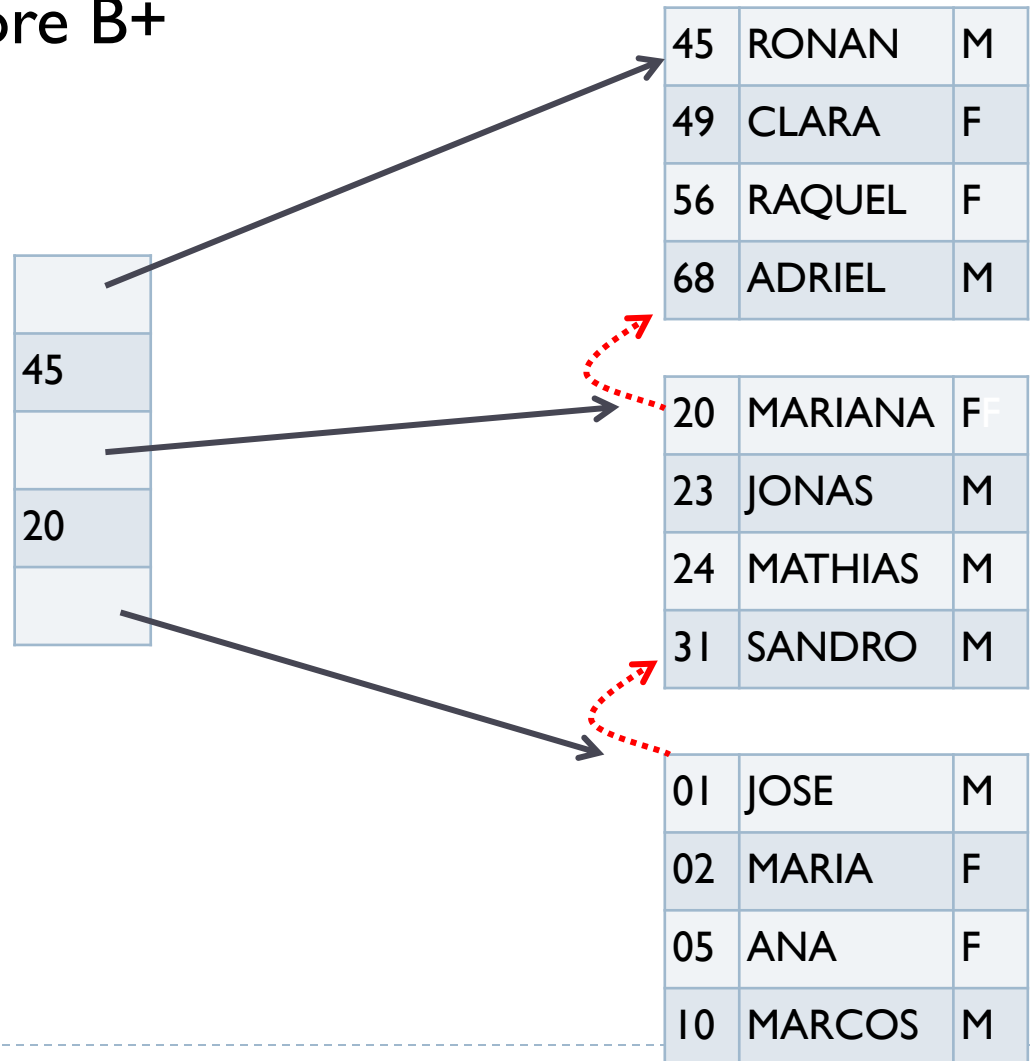


Note que a chave 4 sobe para W, mas o registro correspondente é colocado em Q

Exemplo

(Mostrando os dados nas folhas)

- ▶ Neste exemplo, a árvore B+ tem apenas o nó raiz



Considerações sobre implementação

- ▶ **Pode-se utilizar três arquivos:**
 - ▶ Um arquivo para armazenar os metadados
 - ▶ Ponteiro para a raiz da árvore
 - ▶ Flag indicando se a raiz é folha
 - ▶ Um arquivo para armazenar o índice (nós internos da árvore)
 - ▶ Um arquivo para armazenar os dados (folhas da árvore)

Estrutura do arquivo de índice

- ▶ O arquivo de índice estará estruturado em nós (blocos/páginas)
- ▶ Cada nó possui
 - ▶ Inteiro representando o número de chaves (**m**) armazenadas no nó
 - ▶ **Flag** booleano que diz se página aponta para nó folha (**TRUE** se sim, **FALSE** se não)
 - ▶ Ponteiro para o nó pai (para facilitar a implementação de concatenação)
 - ▶ $p_0, (s_1, p_1), (s_2, p_2), \dots, (s_d, p_d), (s_{d+1}, p_{d+1}), \dots, (s_{2d+1}, p_{2d+1})$, onde:
 - ▶ p_i é um ponteiro para uma página (dentro deste arquivo, se **flag** é **FALSE**, no arquivo de dados, se **flag** é **TRUE**)
 - ▶ s_i é uma chave

Estrutura do arquivo de dados

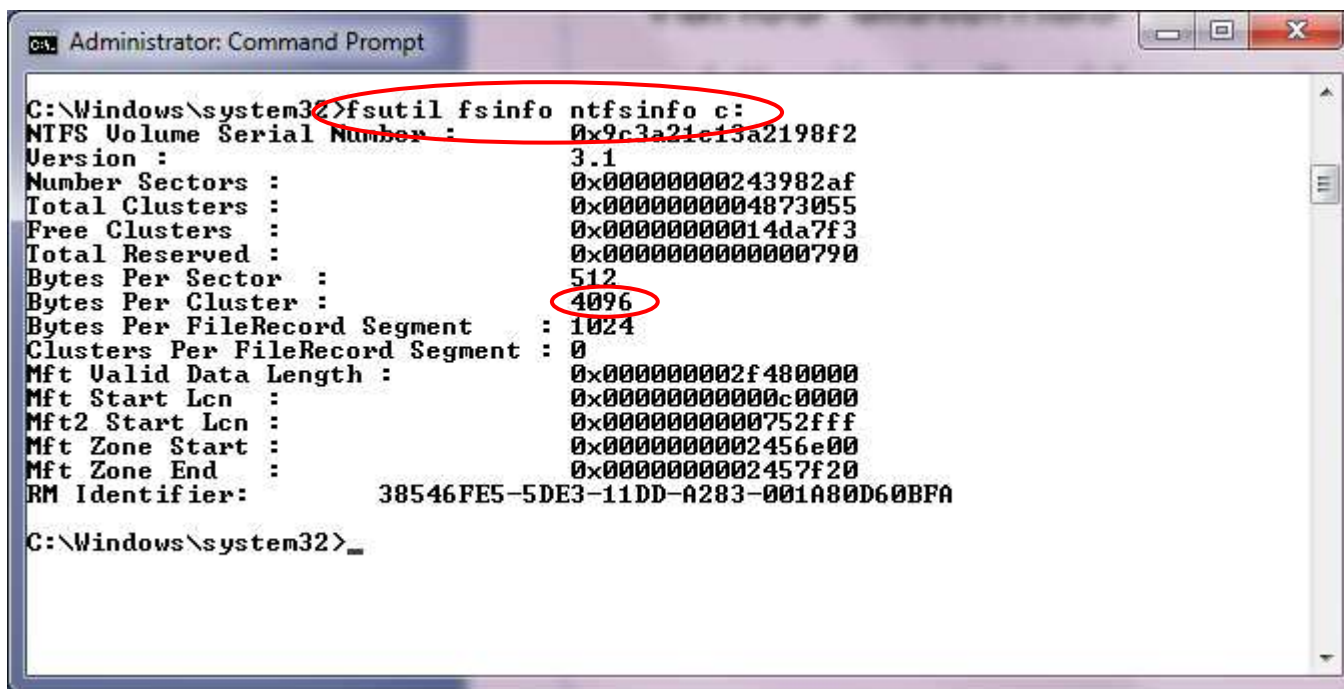
- ▶ O arquivo de dados também estará estruturado em nós (blocos/páginas)
- ▶ Cada nó possui
 - ▶ Inteiro representando o número de chaves (**m**) armazenadas no nó
 - ▶ Ponteiro para o nó pai (para facilitar a implementação de concatenação)
 - ▶ Ponteiro para a próxima página
 - ▶ **2d** registros

Considerações sobre implementação

- ▶ Se o sistema de armazenamento tem tamanho de bloco de **B** bytes, e as chaves a serem armazenadas têm tamanho **k** bytes, a árvore B+ mais eficiente é a de ordem **$d = (B / k) - 1$**
- ▶ Exemplo prático:
 - ▶ Tamanho do bloco do disco $B = 4\text{KB} = 4096$ bytes
 - ▶ Tamanho da chave $k = 4$ bytes
 - ▶ $d = (4096/4) - 1 = 1023$
 - ▶ Quantas chaves cada nó da árvore terá, nessa situação? $2d = 2046$ chaves!

Dica

- ▶ Como determinar o tamanho do bloco de disco em vários sistemas operacionais:
 - ▶ <http://arjudba.blogspot.com/2008/07/how-to-determine-os-block-size-for.html>



```
Administrator: Command Prompt
C:\Windows\system32>fsutil fsinfo ntfsinfo c:
NTFS Volume Serial Number : 0x9c3a21c13a2198f2
Version : 3.1
Number Sectors : 0x00000000243982af
Total Clusters : 0x000000004873055
Free Clusters : 0x00000000014da7f3
Total Reserved : 0x000000000000790
Bytes Per Sector : 512
Bytes Per Cluster : 4096
Bytes Per FileRecord Segment : 1024
Clusters Per FileRecord Segment : 0
Mft Valid Data Length : 0x000000002f480000
Mft Start Lcn : 0x00000000000c0000
Mft2 Start Lcn : 0x0000000000752fff
Mft Zone Start : 0x0000000002456e00
Mft Zone End : 0x0000000002457f20
RM Identifier: 38546FE5-5DE3-11DD-A283-001A80D60BFA

C:\Windows\system32>_
```

Exercício: Árvore B+

- ▶ Passo 1) Desenhar uma árvore B+ de **ordem 2** que contenha registros com as seguintes chaves: 1, 2, 3, 8, 15, 35, 36, 38, 39, 41, 43, 45, 51, 59
- ▶ Como $d = 2$:
 - ▶ Cada nó tem no máximo 4 chaves
 - ▶ Cada nó tem no máximo 5 filhos
- ▶ Passo 2) Sobre o resultado do passo 1, excluir os registros de chave: 3, 38, 1, 41
- ▶ Passo 3) Sobre o resultado do passo 2, incluir os registros de chave: 5, 14, 52, 53, 54

Exercício: Árvore B+

- ▶ Escreva um algoritmo de busca de um registro de chave x em uma árvore B+
- ▶ Escreva um algoritmo de inserção de um registro de chave x em uma árvore B+
- ▶ Escreva um algoritmo de remoção de um registro de chave x em uma árvore B+

- ▶ Em todos os exercícios acima, assuma que são conhecidos:
 - ▶ o número de chaves que um determinado nó armazena (m)
 - ▶ a ordem da árvore (d)

Implementação

- ▶ Implementar busca, inserção e exclusão em árvore B. Detalhes estão no Google Classroom.