

# ÁRVORE B+

Vanessa Braganholo  
Estruturas de Dados e Seus  
Algoritmos

# 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

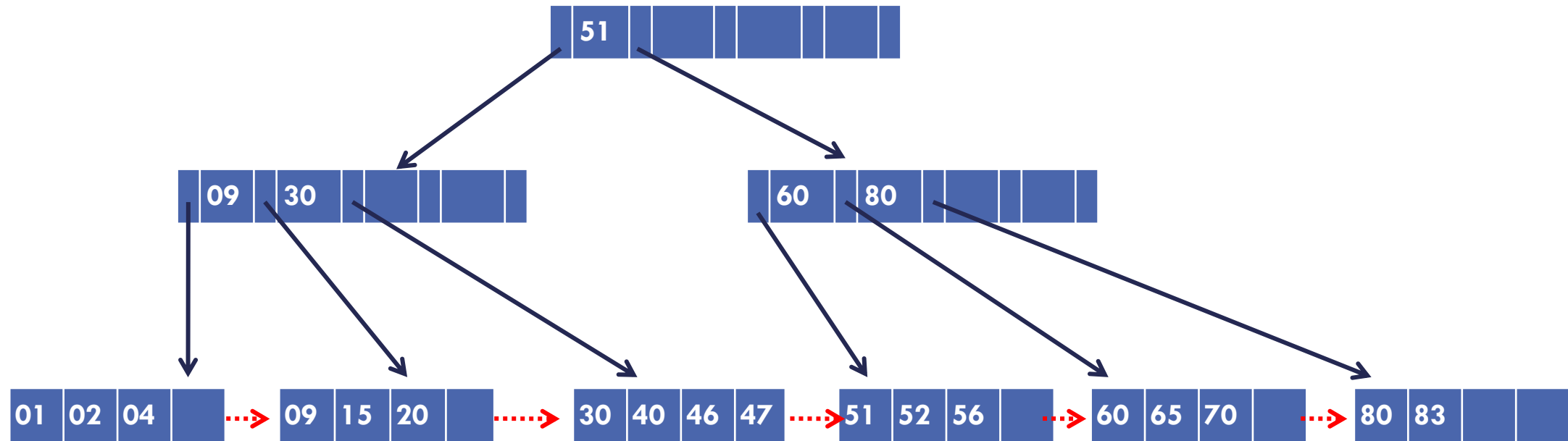
Isso permite o armazenamento dos **dados em um arquivo**, e do **índice em outro arquivo** separado

# ÁRVORE B+ NA PRÁTICA

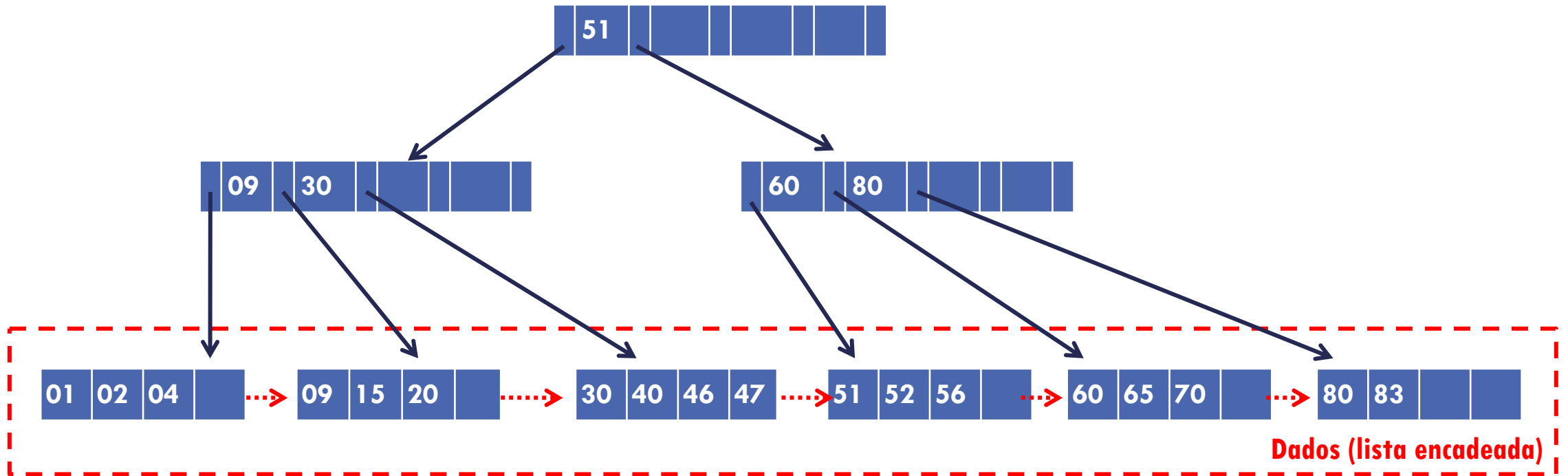
Árvores B+ são muito importantes por sua eficiência, e muito utilizadas na prática:

- 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

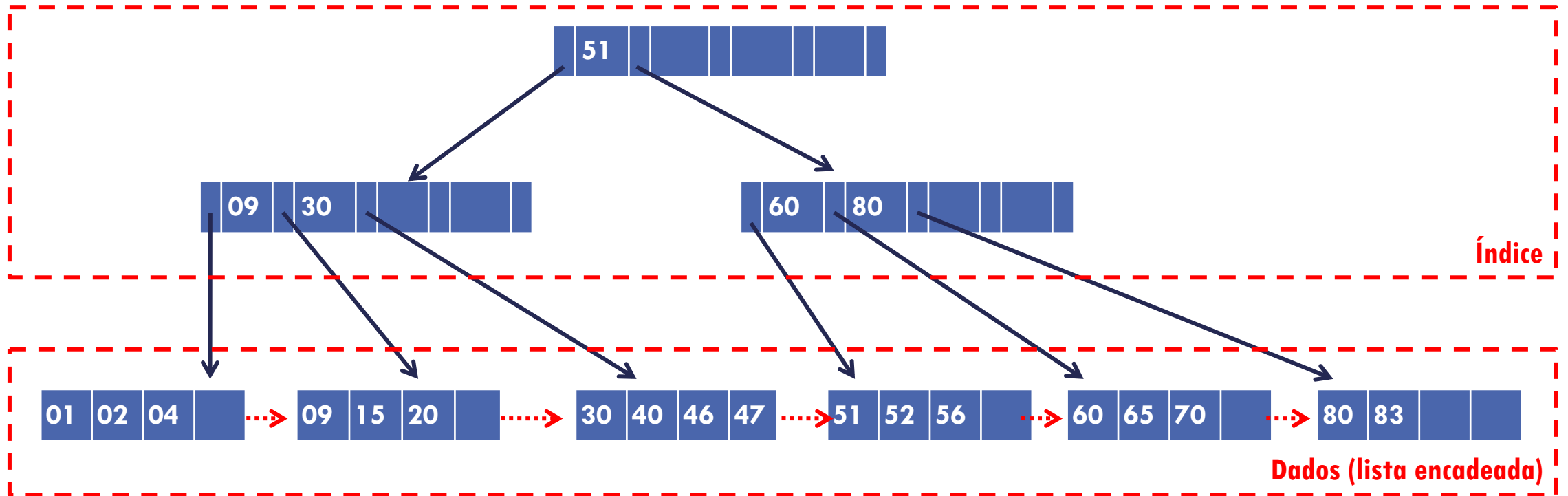
# EXEMPLO DE ÁRVORE B+ DE ORDEM $D = 2$



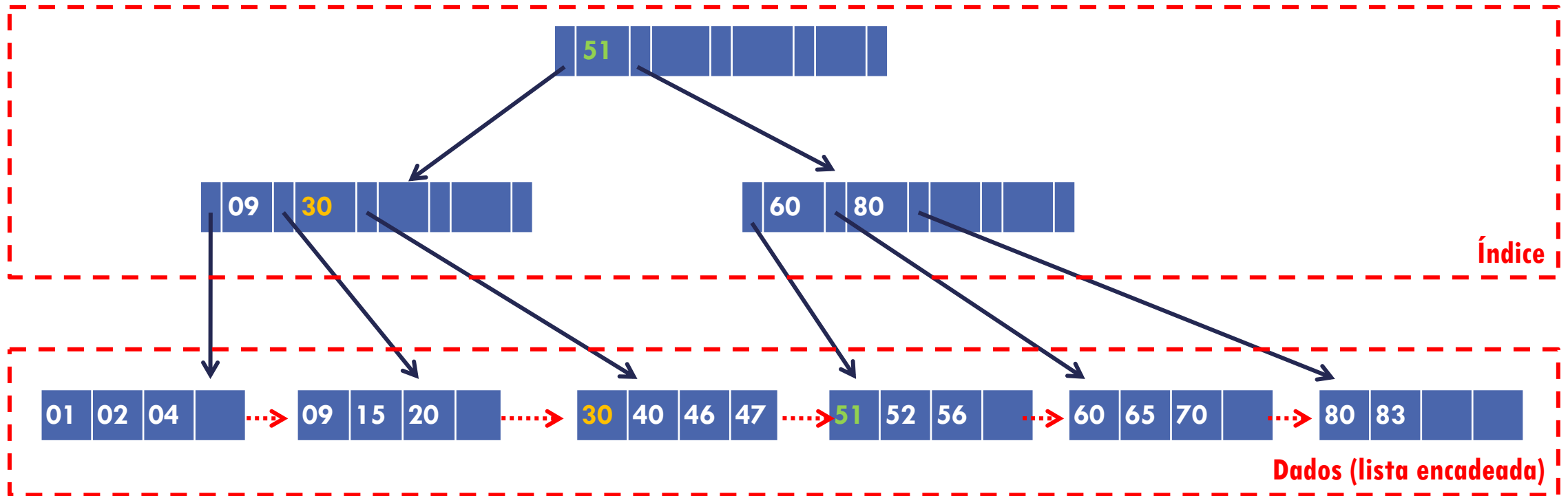
# EXEMPLO DE ÁRVORE B+ DE ORDEM $D = 2$



# EXEMPLO DE ÁRVORE B+ DE ORDEM $D = 2$



# EXEMPLO DE ÁRVORE B+ DE ORDEM D = 2



## IMPORTANTE:

- Índices **repetem valores** de chave que aparecem nas folhas (diferente do que acontece nas árvores B)

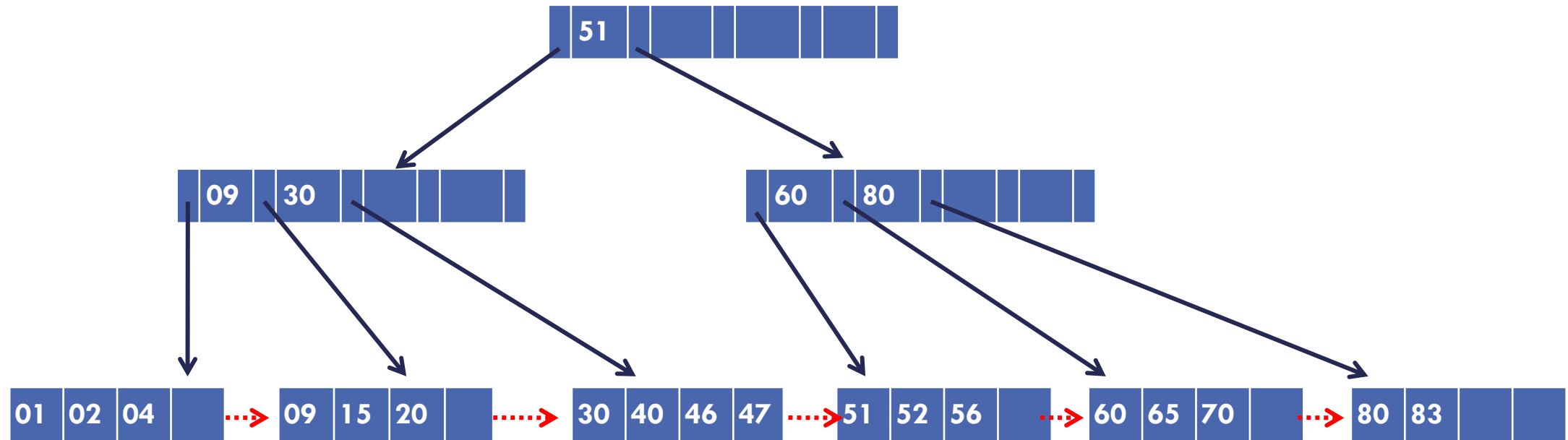
# BUSCA

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

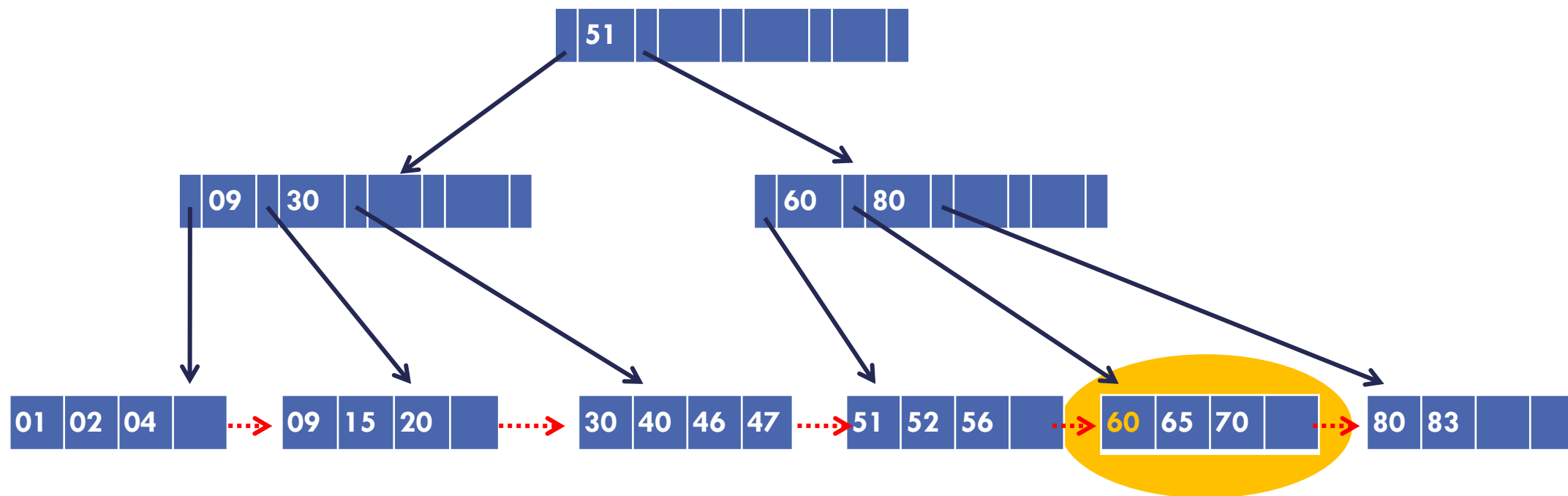
Notar que comparações agora não são apenas  $>$ , mas  $\geq$



# EXEMPLO: BUSCA DE 60



# EXEMPLO: BUSCA DE 60



# INSERÇÃO

Quando for necessário particionar um nó durante uma inserção, o mesmo raciocínio do particionamento em Árvore B 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+

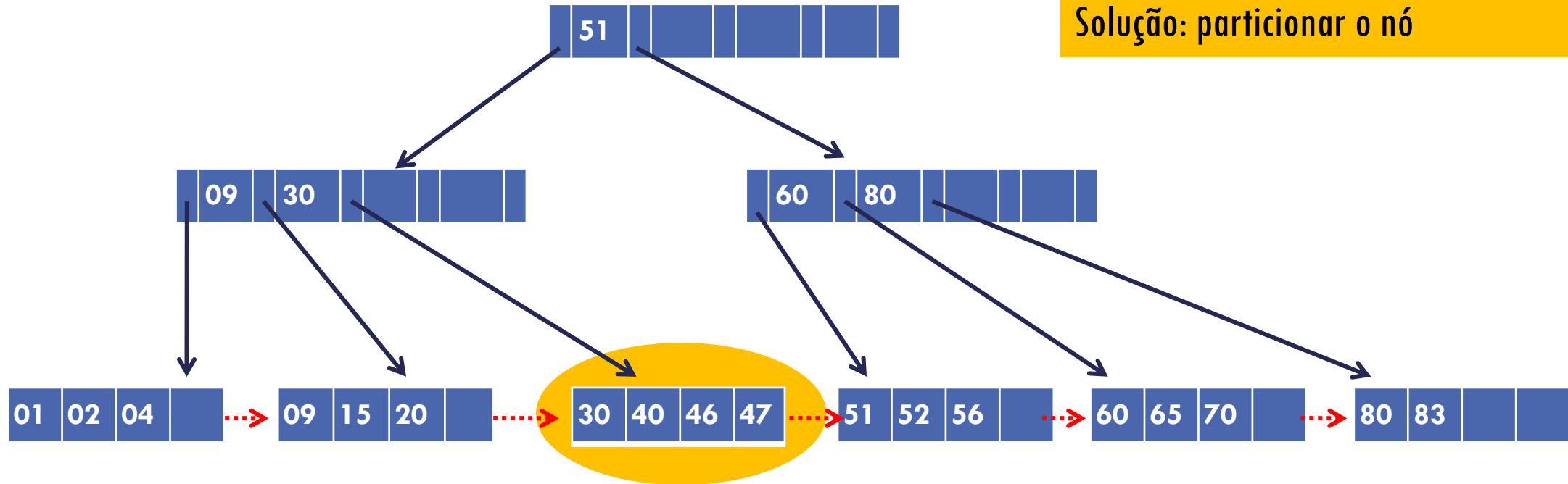
## INSERIR CHAVE 32

ordem  $d = 2$

Inserir chave 32

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

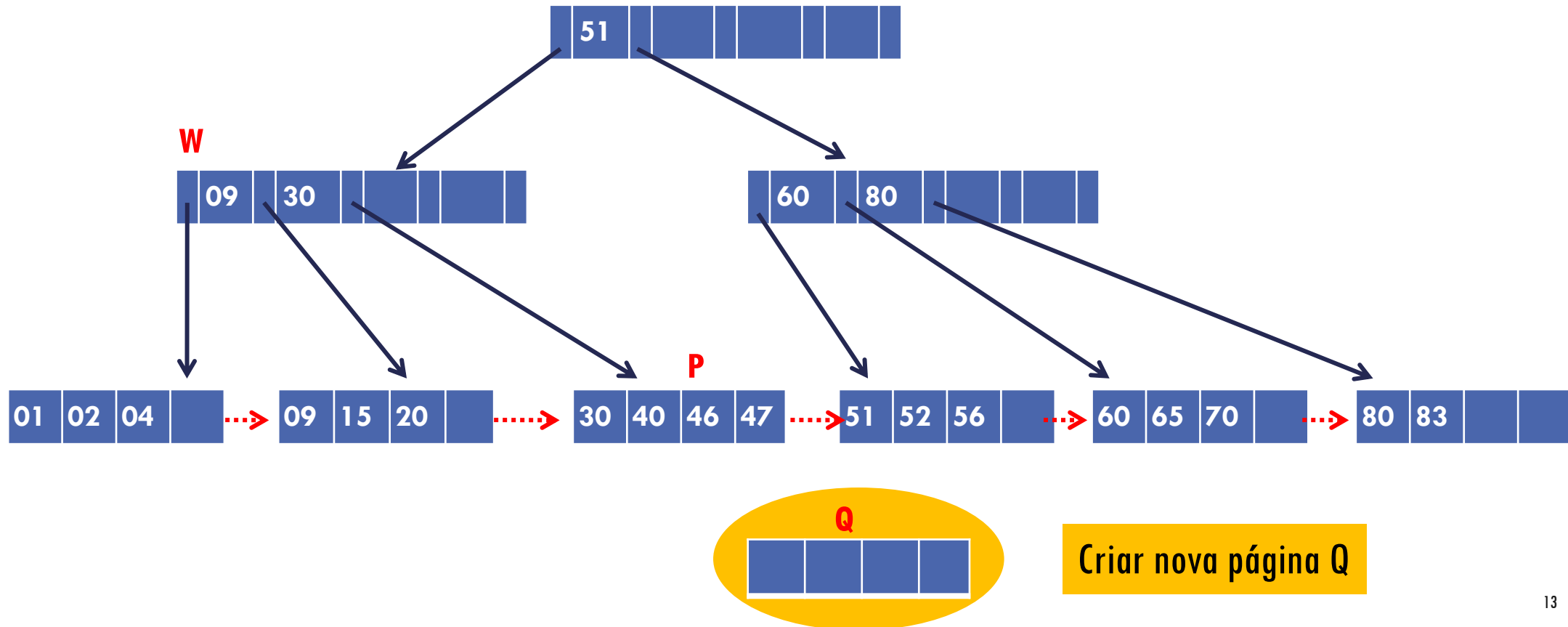
Solução: particionar o nó



# EXEMPLO DE INSERÇÃO EM ÁRVORE B+

## INSERIR CHAVE 32

ordem  $d = 2$



# EXEMPLO DE INSERÇÃO EM ÁRVORE B+

## INSERIR CHAVE 32

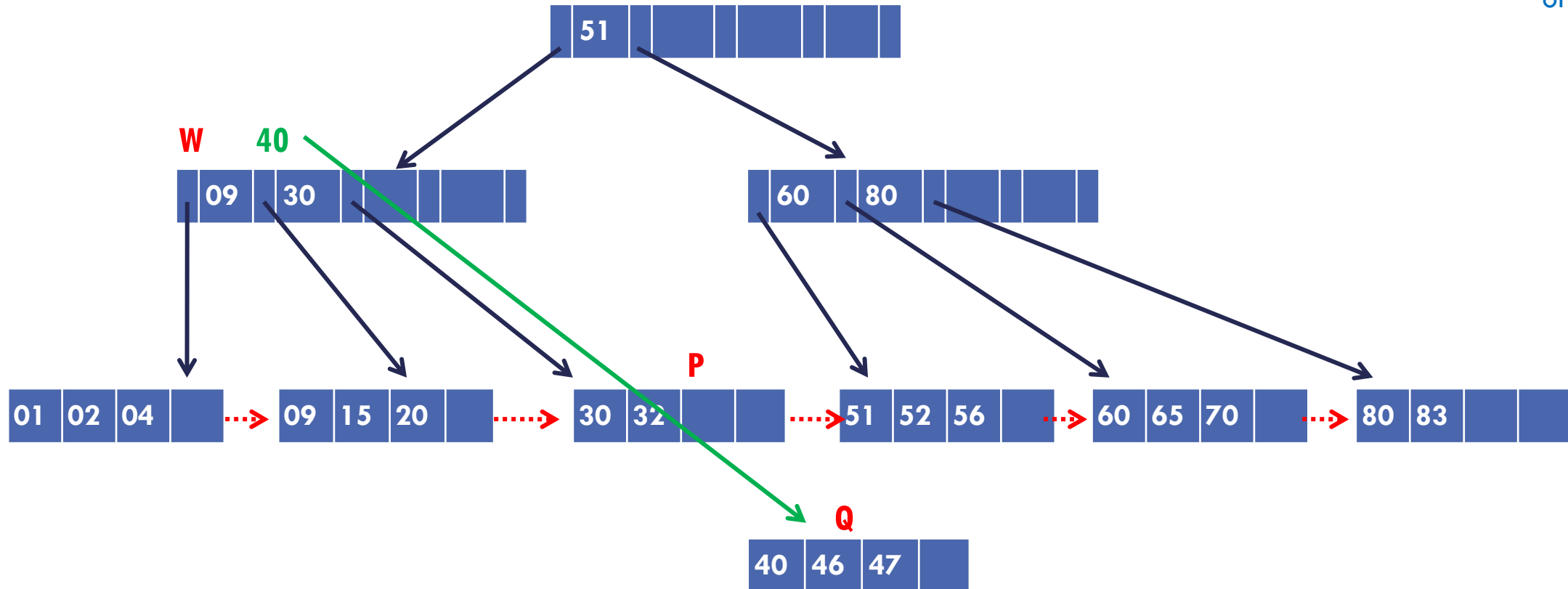
Dividir as chaves entre as duas páginas (30; 32; 40; 46; 47)

$d$  chaves na página original  $P$

chave  $d+1$  sobe para nó pai  $W$  (**mas registro é mantido na nova página**)

$d+1$  chaves restantes na nova página  $Q$

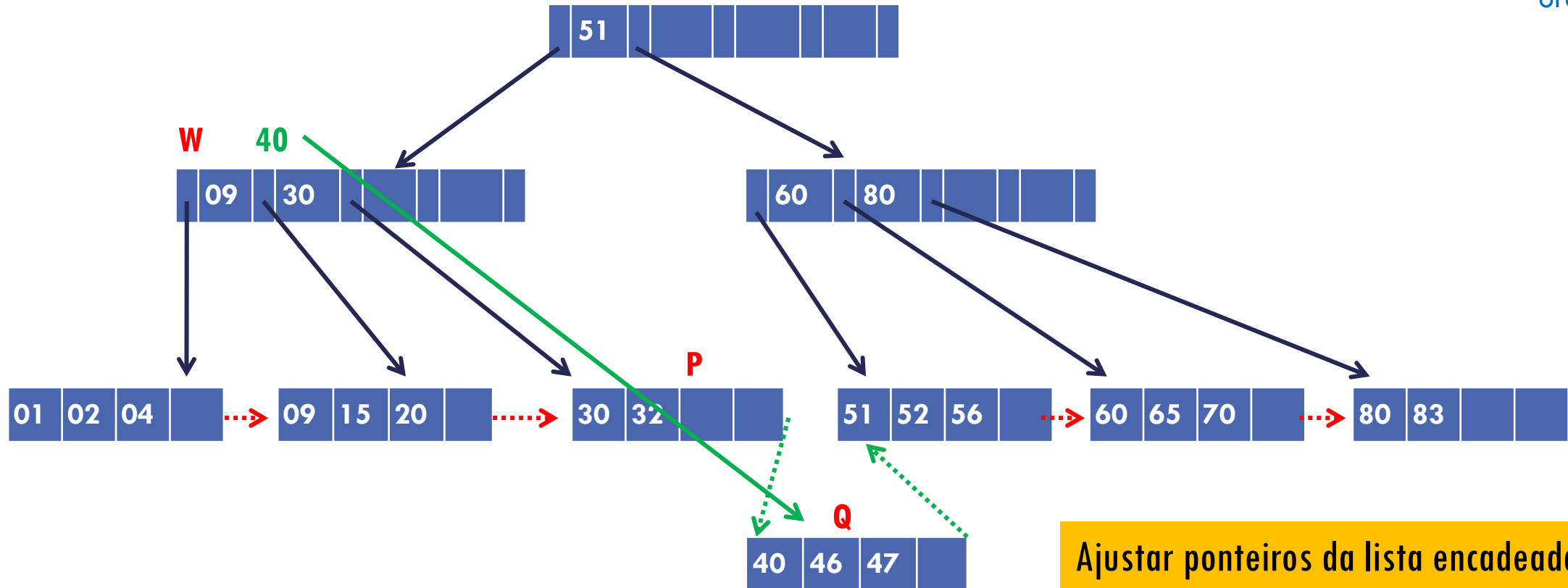
ordem  $d = 2$



# EXEMPLO DE INSERÇÃO EM ÁRVORE B+

## INSERIR CHAVE 32

ordem  $d = 2$

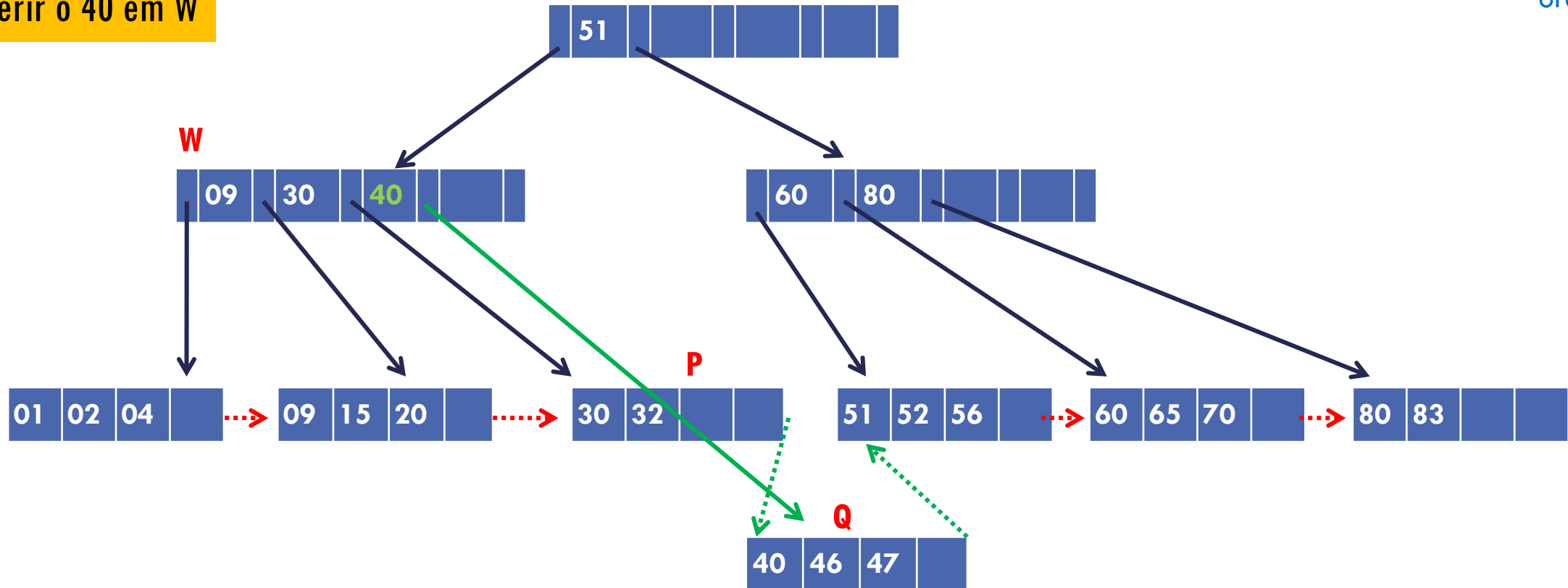


# EXEMPLO DE INSERÇÃO EM ÁRVORE B+

## INSERIR CHAVE 32

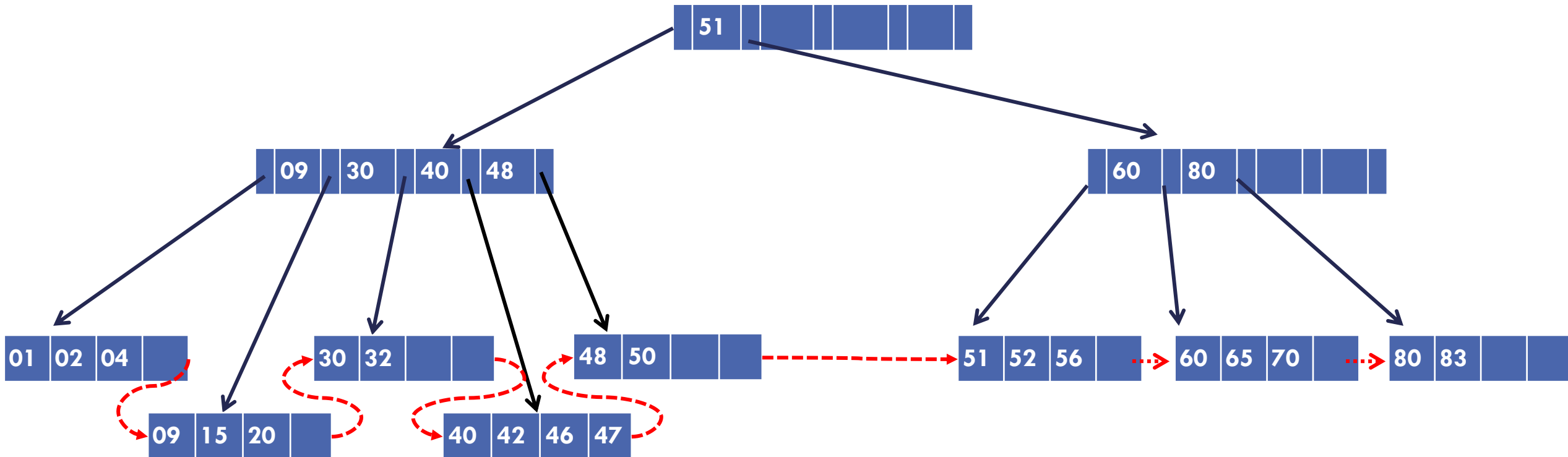
Inserir o 40 em W

ordem  $d = 2$





# EXEMPLO DE INSERÇÃO COM PARTICIONAMENTO DE NÓ INTERNO: INSERIR CHAVE 44

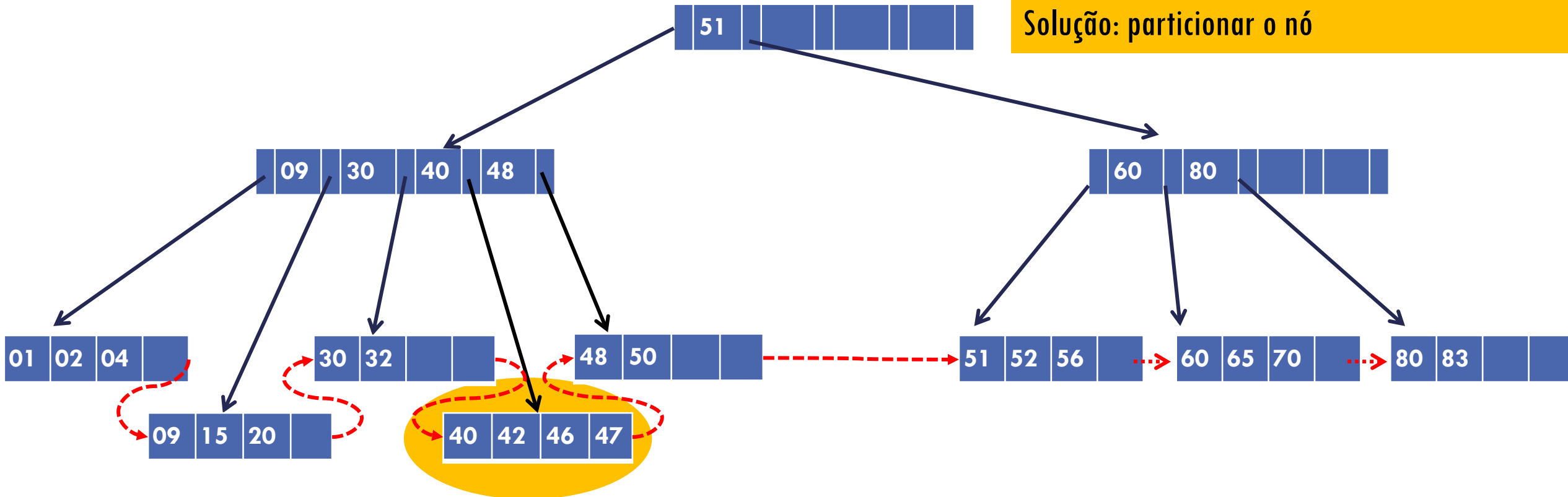


# EXEMPLO DE INSERÇÃO COM PARTICIONAMENTO DE NÓ INTERNO: INSERIR CHAVE 44

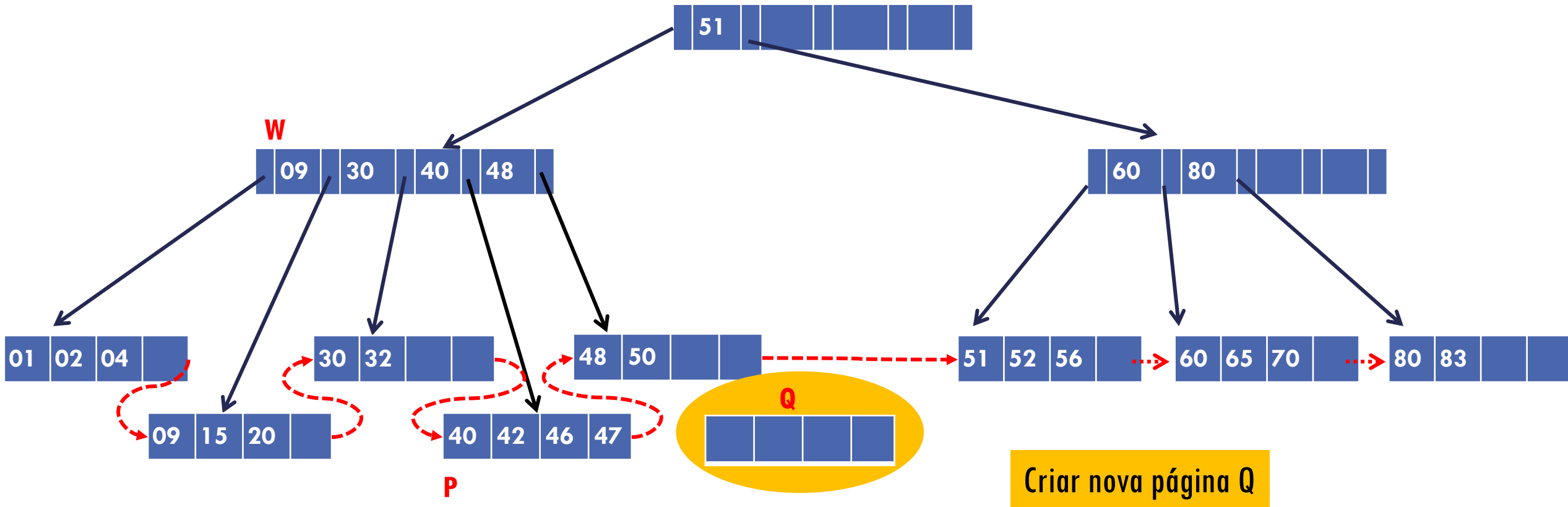
Inserir chave 44

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

Solução: particionar o nó

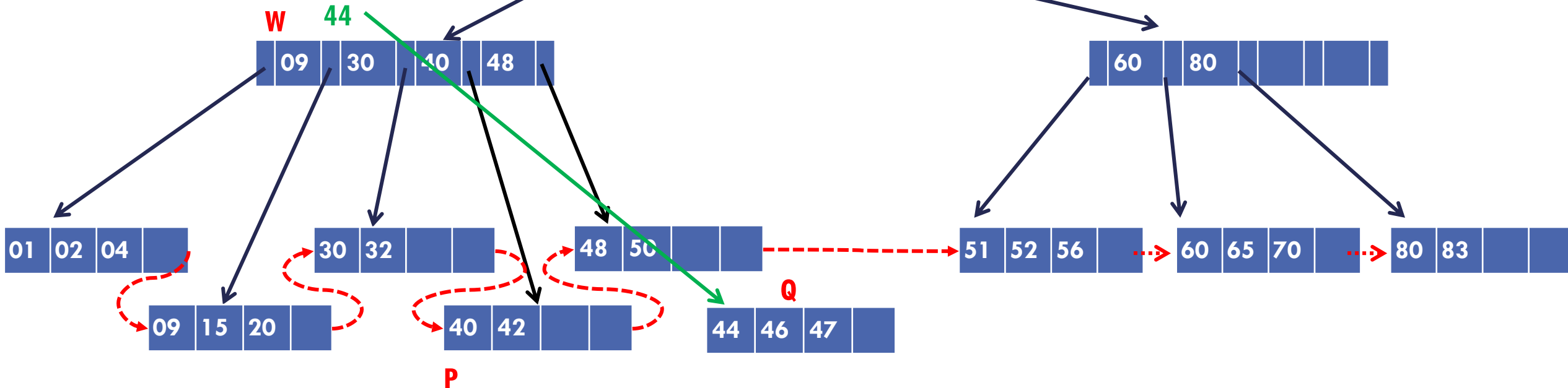


# EXEMPLO DE INSERÇÃO COM PARTICIONAMENTO DE NÓ INTERNO: INSERIR CHAVE 44

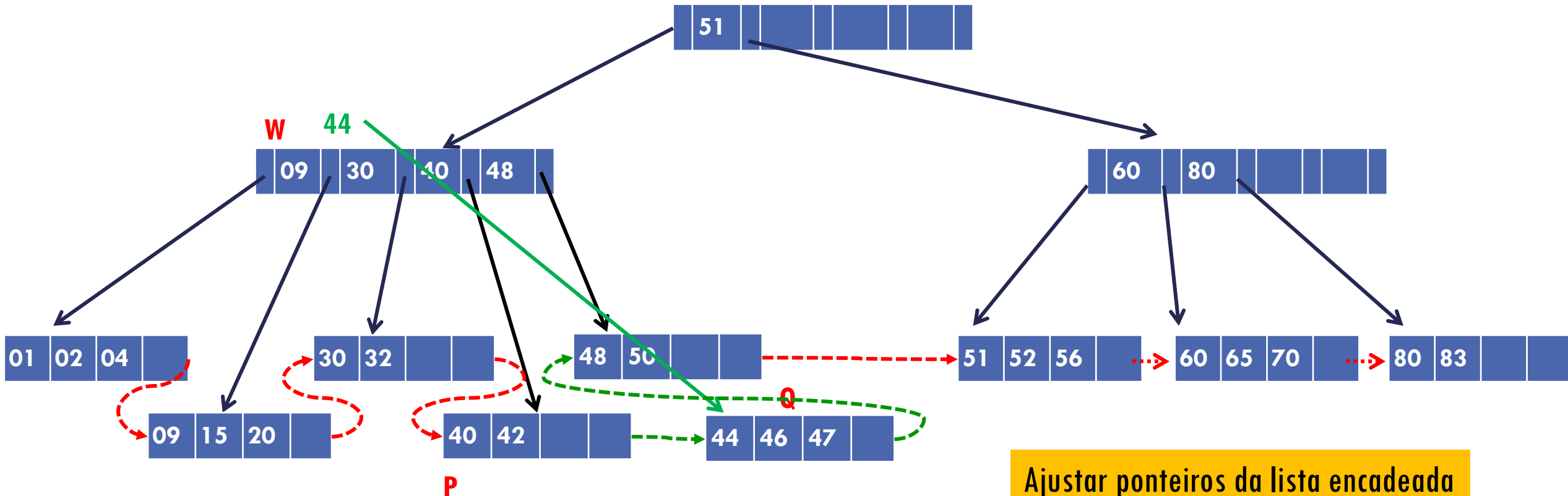


# EXEMPLO DE INSERÇÃO COM PARTICIONAMENTO DE NÓ INTERNO: INSERIR CHAVE 44

Dividir as chaves entre as duas páginas (40; 42; 44; 46; 47)  
 $d$  chaves na página original P  
 chave  $d+1$  sobe para nó pai W (mas registro é mantido na nova página)  
 $d+1$  chaves restantes na nova página Q



# EXEMPLO DE INSERÇÃO COM PARTICIONAMENTO DE NÓ INTERNO: INSERIR CHAVE 44

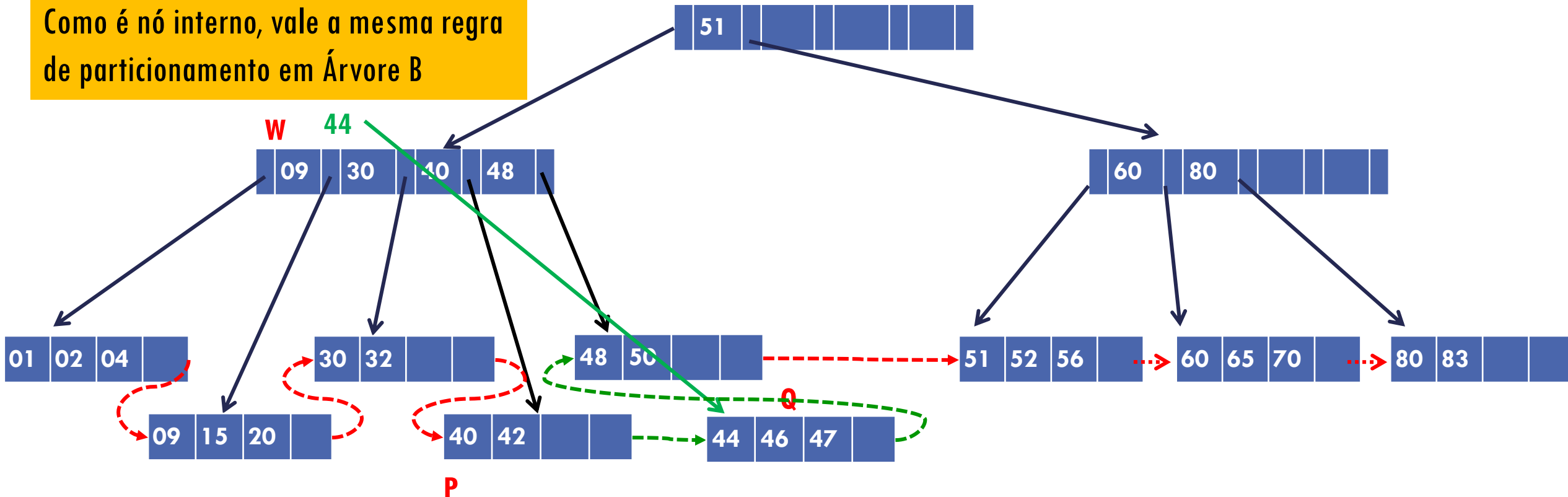


# EXEMPLO DE INSERÇÃO COM PARTICIONAMENTO DE NÓ INTERNO: INSERIR CHAVE 44

Inserir 44 em W

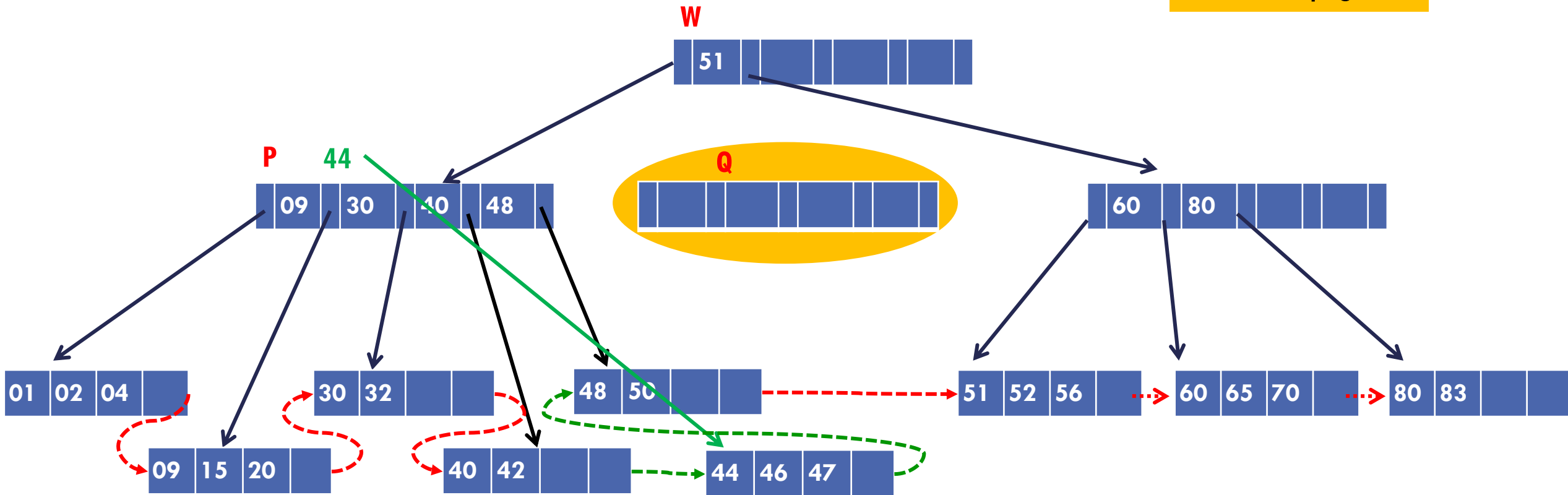
Não há espaço: particionar

Como é nó interno, vale a mesma regra de particionamento em Árvore B



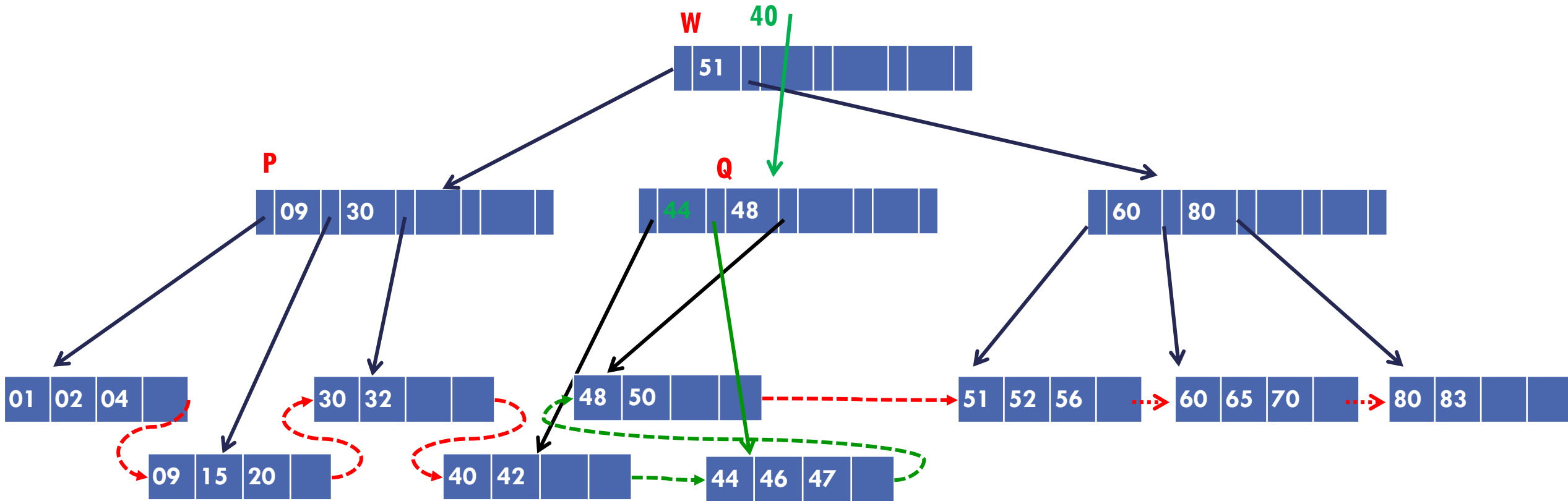
# EXEMPLO DE INSERÇÃO COM PARTICIONAMENTO DE NÓ INTERNO: INSERIR CHAVE 44

Criar nova página Q



Dividir as chaves entre as duas páginas (09; 30; 40; 44; 48)  
 $d$  chaves na página original P  
 chave  $d+1$  sobe para nó pai W  
 chaves  $d+2$  em diante na nova página Q

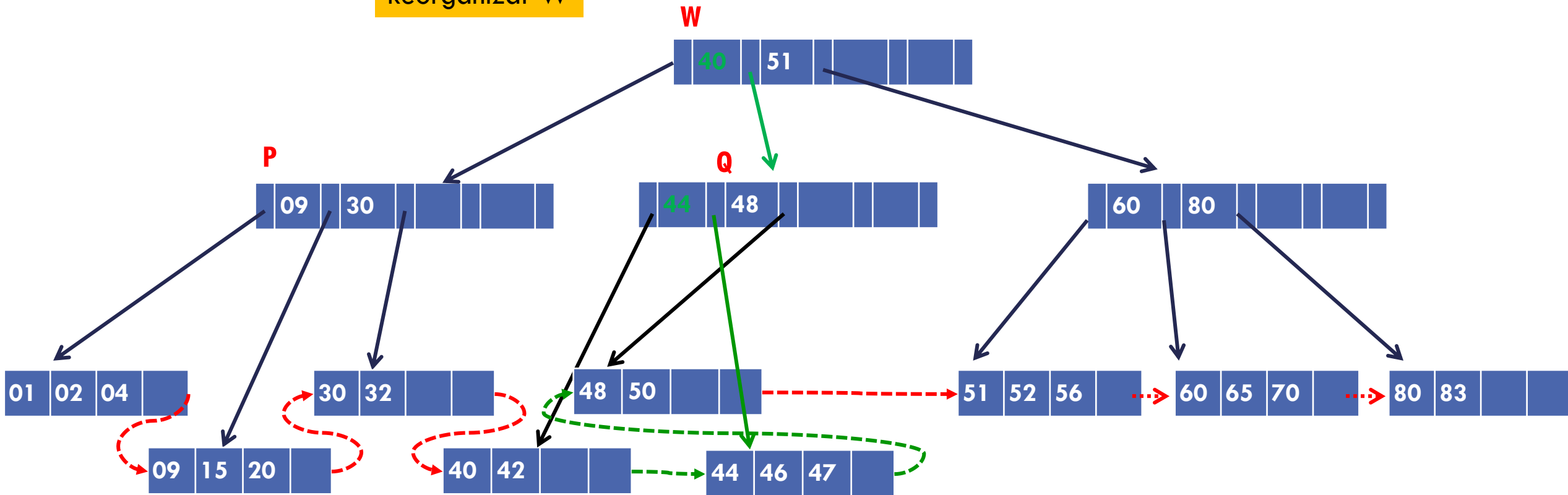
# B+ TREE PARTICIONAMENTO DE NO INTERNO: INSERIR CHAVE 44





# EXEMPLO DE INSERÇÃO COM PARTICIONAMENTO DE NÓ INTERNO: INSERIR CHAVE 44

Reorganizar W



# EXCLUSÃO

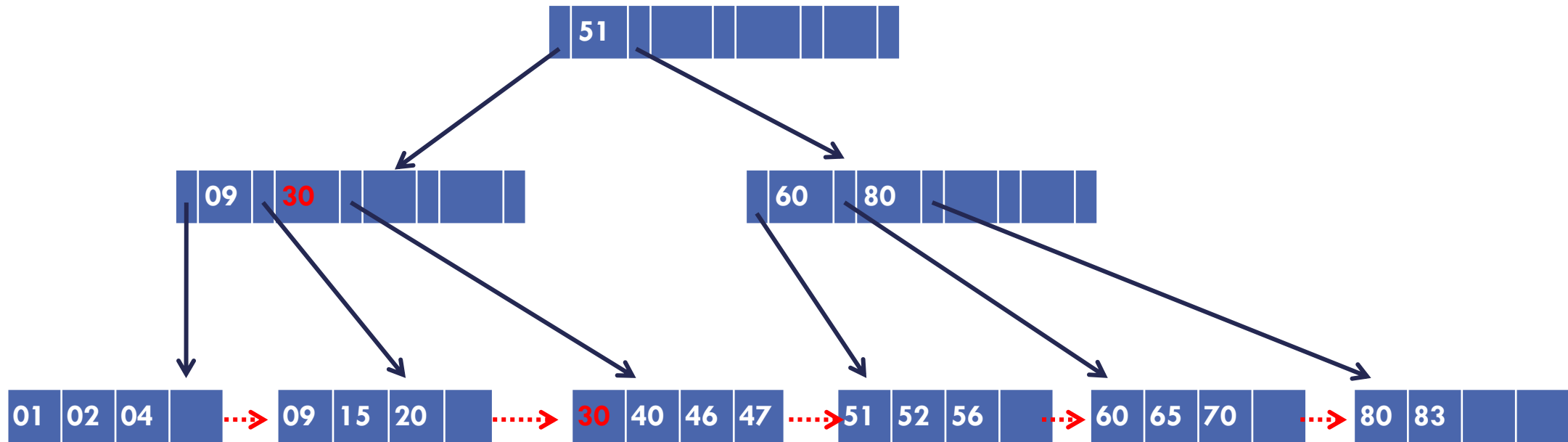
Excluir apenas no nó folha

Chaves excluídas continuam nos nós intermediários

# EXEMPLO DE EXCLUSÃO EM ÁRVORE B+

## EXCLUIR CHAVE 30

ordem  $d = 2$

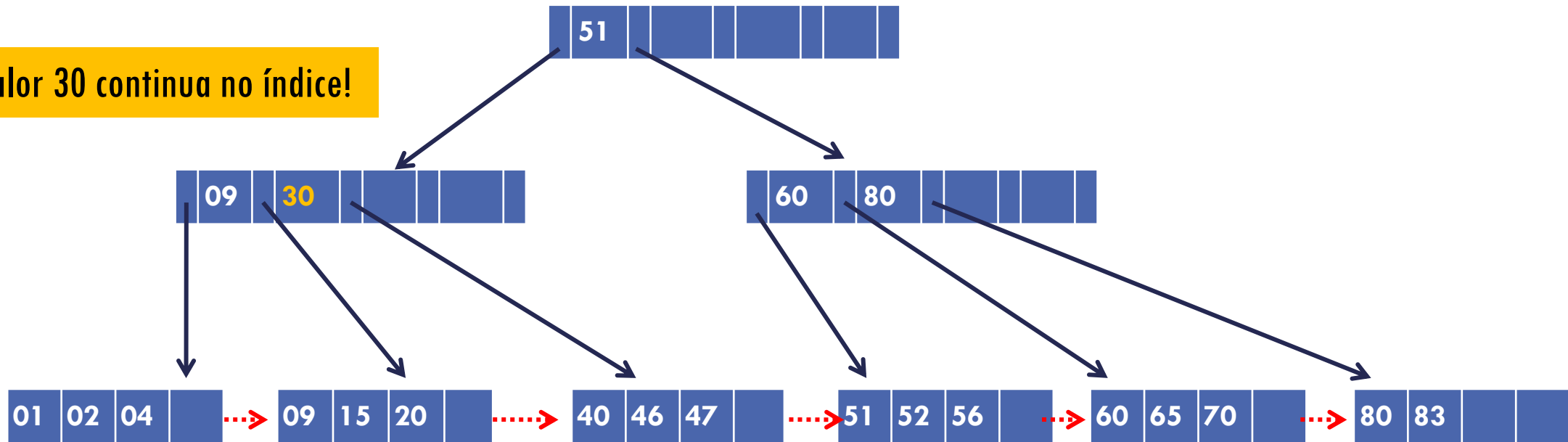


# EXEMPLO DE EXCLUSÃO EM ÁRVORE B+

## 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

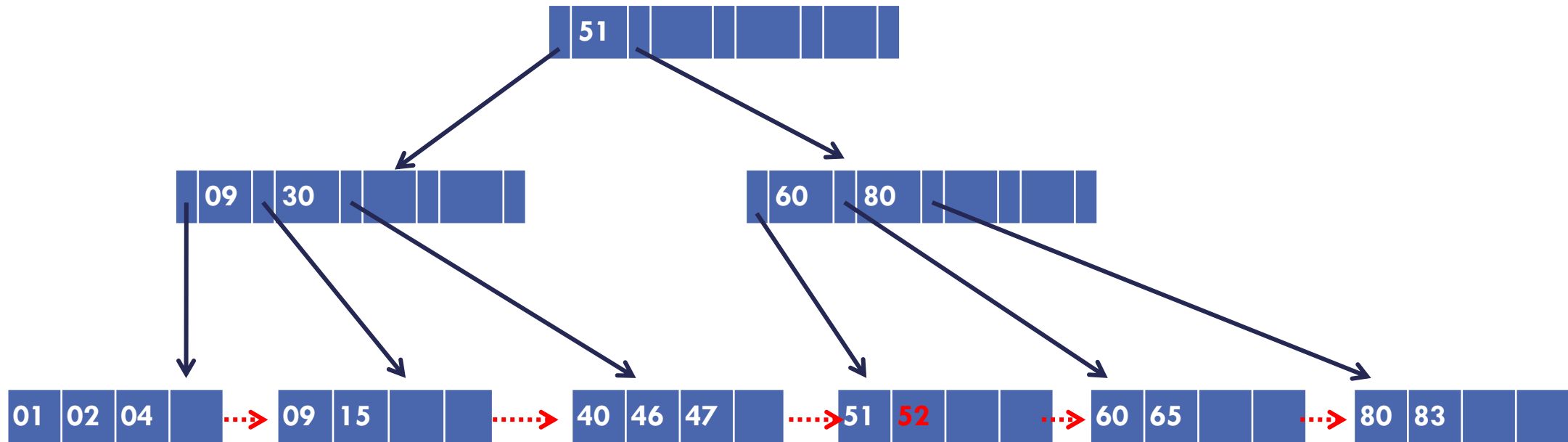
## Importante:

- 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**: usa-se a mesma lógica utilizada na **árvore B**

# EXEMPLO DE EXCLUSÃO EM ÁRVORE B+

## EXCLUIR CHAVE 52

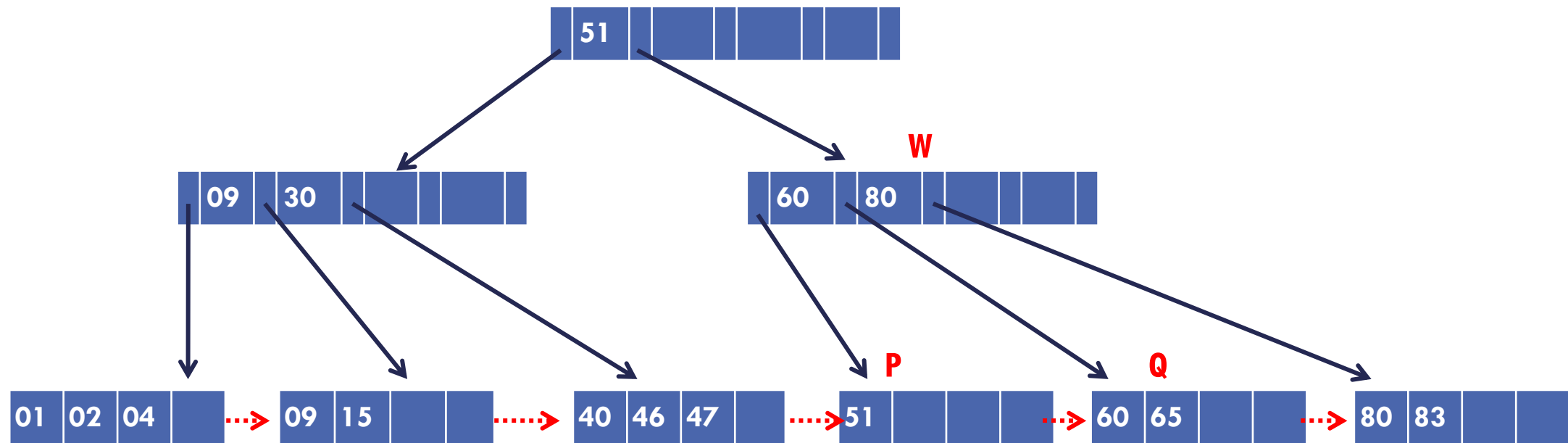
ordem  $d = 2$



# EXEMPLO DE EXCLUSÃO EM ÁRVORE B+

## EXCLUIR CHAVE 52

ordem  $d = 2$

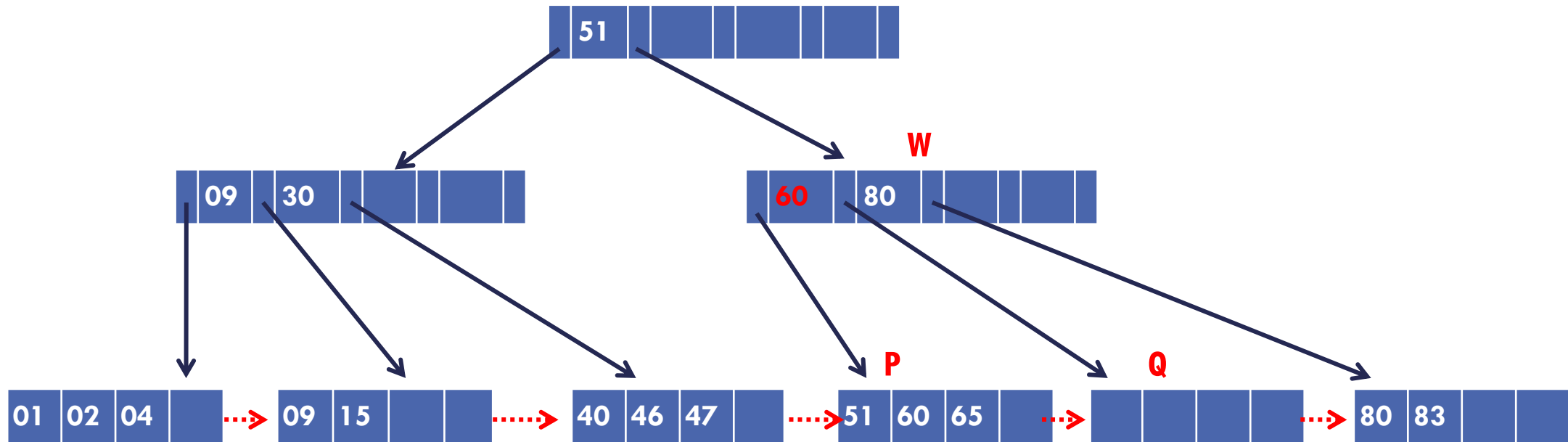


Nó ficou com menos de  $d$  entradas – necessário tratar isso  
Soma dos registros de P e Q  $< 2d$   
Usar concatenação

# EXEMPLO DE EXCLUSÃO EM ÁRVORE B+

## EXCLUIR CHAVE 52

ordem  $d = 2$



Concatenação:

Passar os registros de Q para P

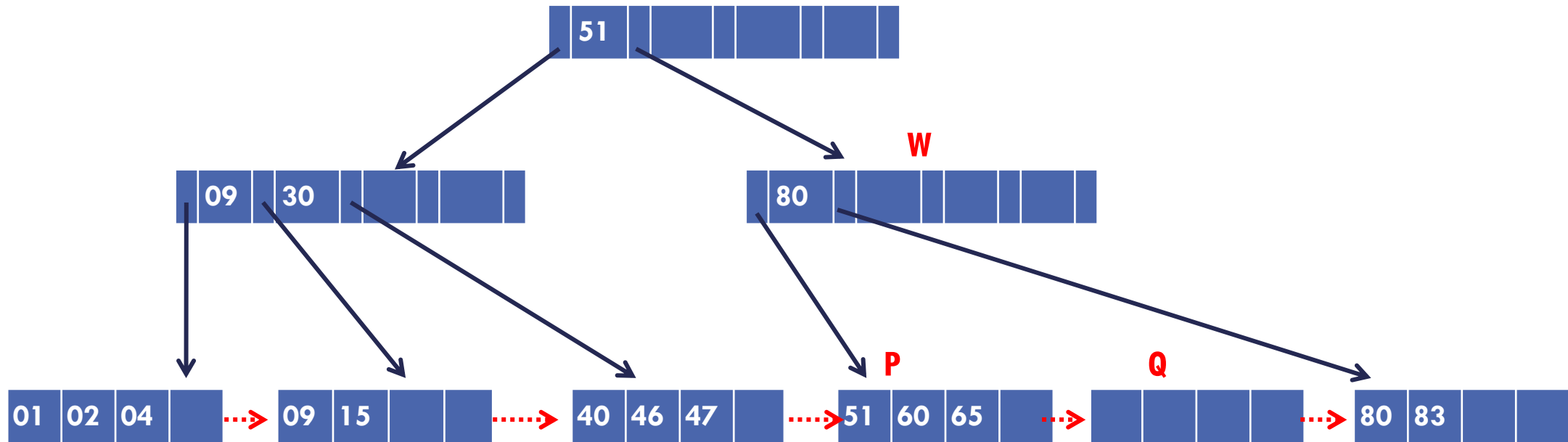
Eliminar a chave em W que divide os ponteiros para as páginas P e Q



# EXEMPLO DE EXCLUSÃO EM ÁRVORE B+

## EXCLUIR CHAVE 52

ordem  $d = 2$



Concatenação:

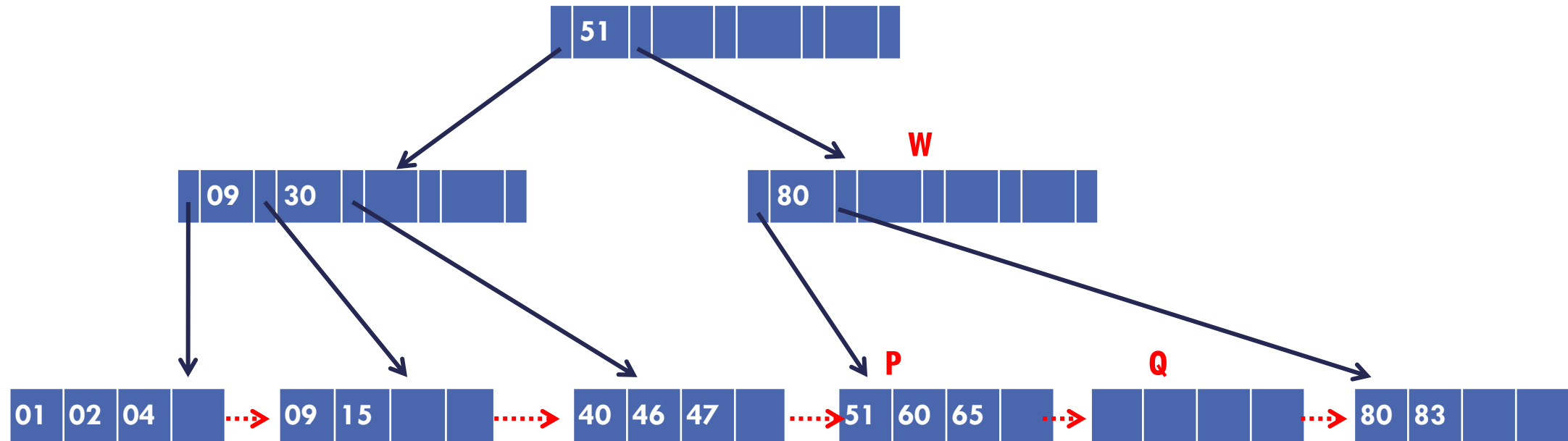
Passar os registros de Q para P

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  $d = 2$

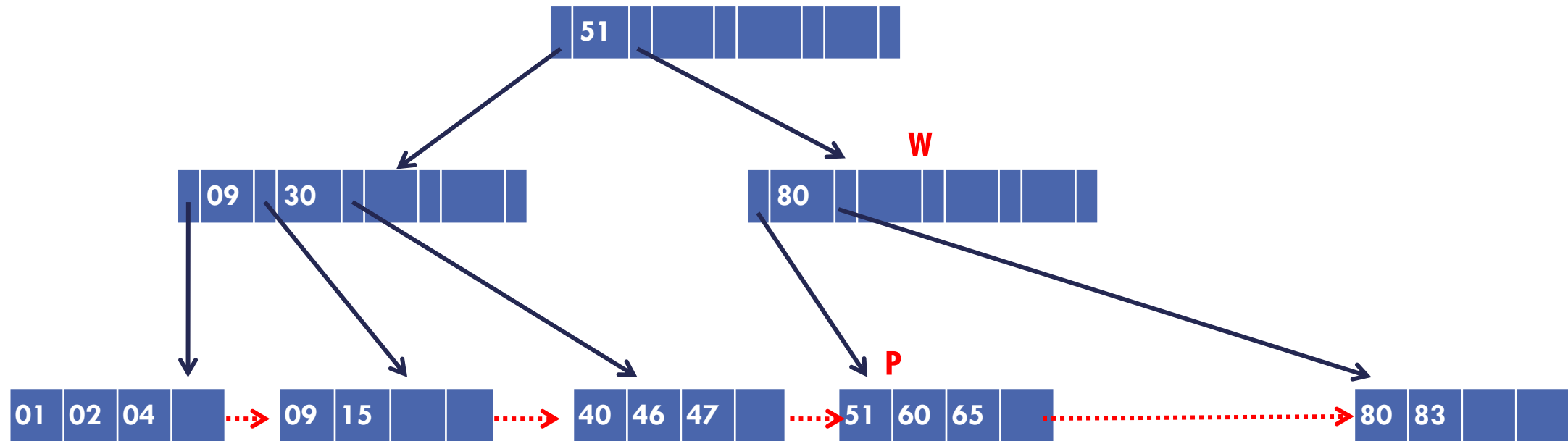


Eliminar nó Q

# EXEMPLO DE EXCLUSÃO EM ÁRVORE B+

## EXCLUIR CHAVE 52

ordem  $d = 2$

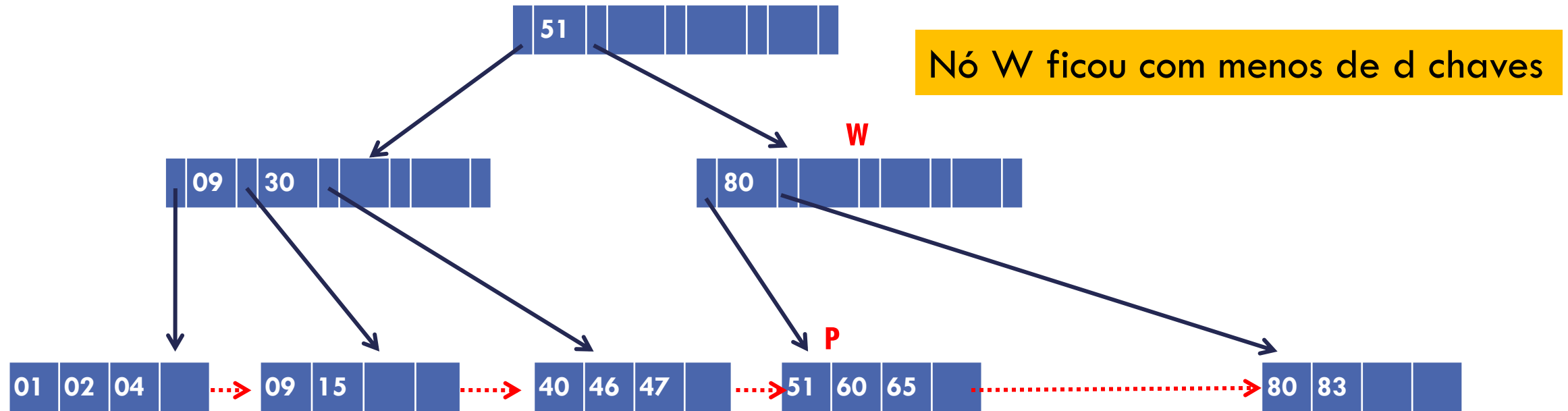


Eliminar nó Q

# EXEMPLO DE EXCLUSÃO EM ÁRVORE B+

## EXCLUIR CHAVE 52

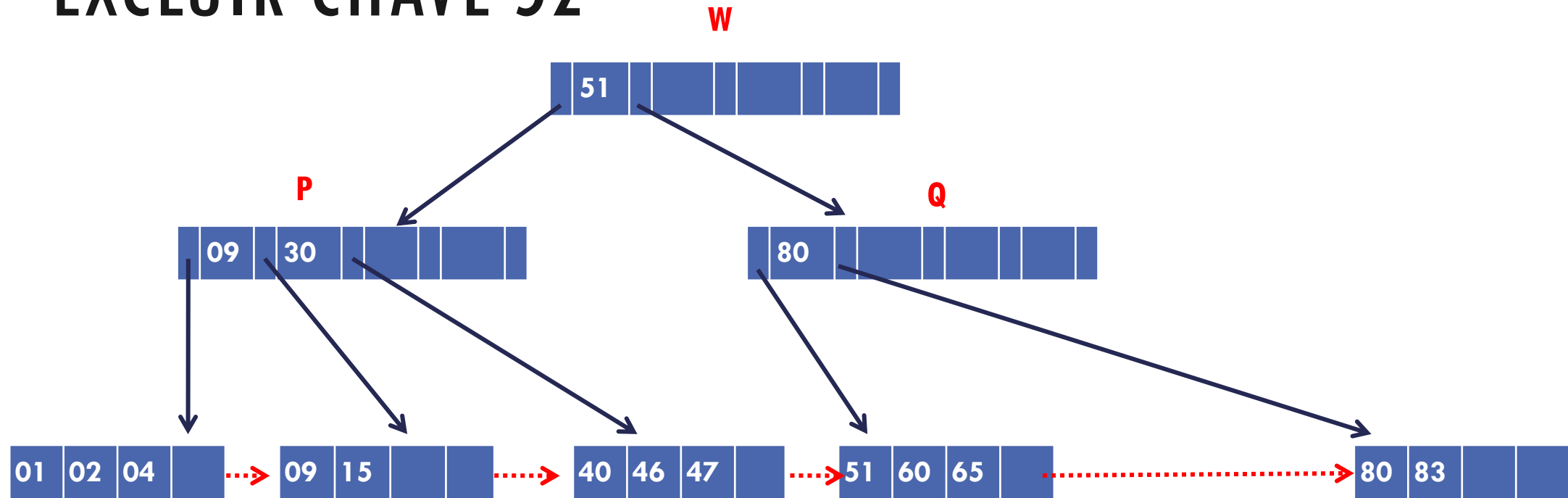
ordem  $d = 2$



# EXEMPLO DE EXCLUSÃO EM ÁRVORE B+

## EXCLUIR CHAVE 52

ordem  $d = 2$

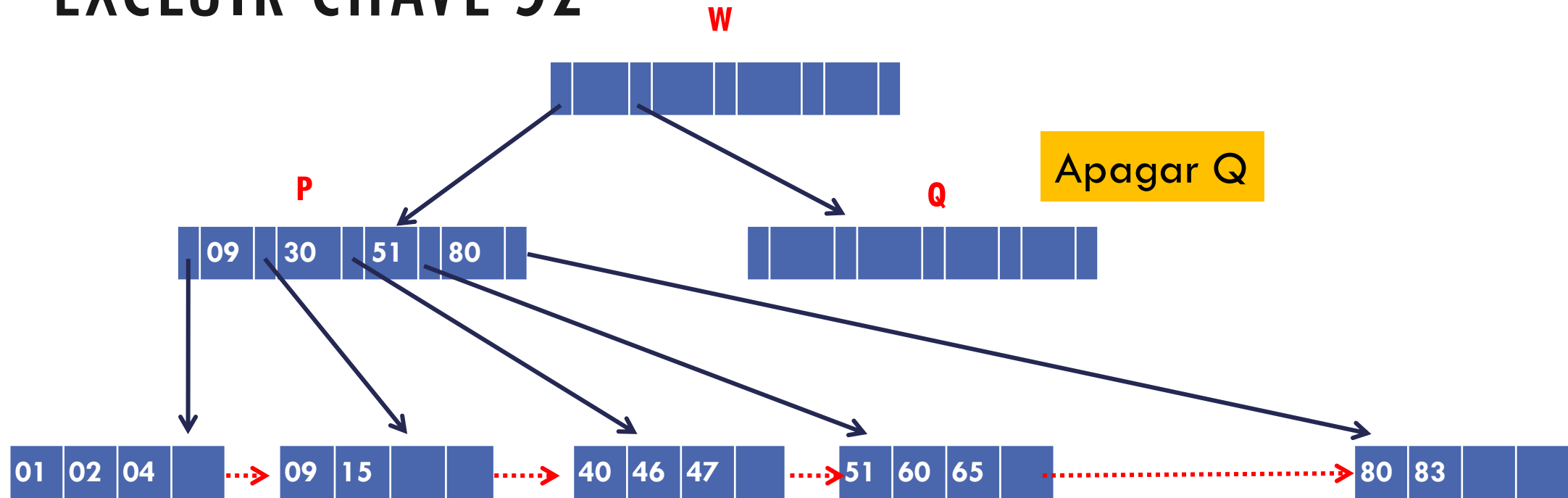


Soma de total de chaves de P e Q  $< 2d$   
Solução: concatenação

# EXEMPLO DE EXCLUSÃO EM ÁRVORE B+

## EXCLUIR CHAVE 52

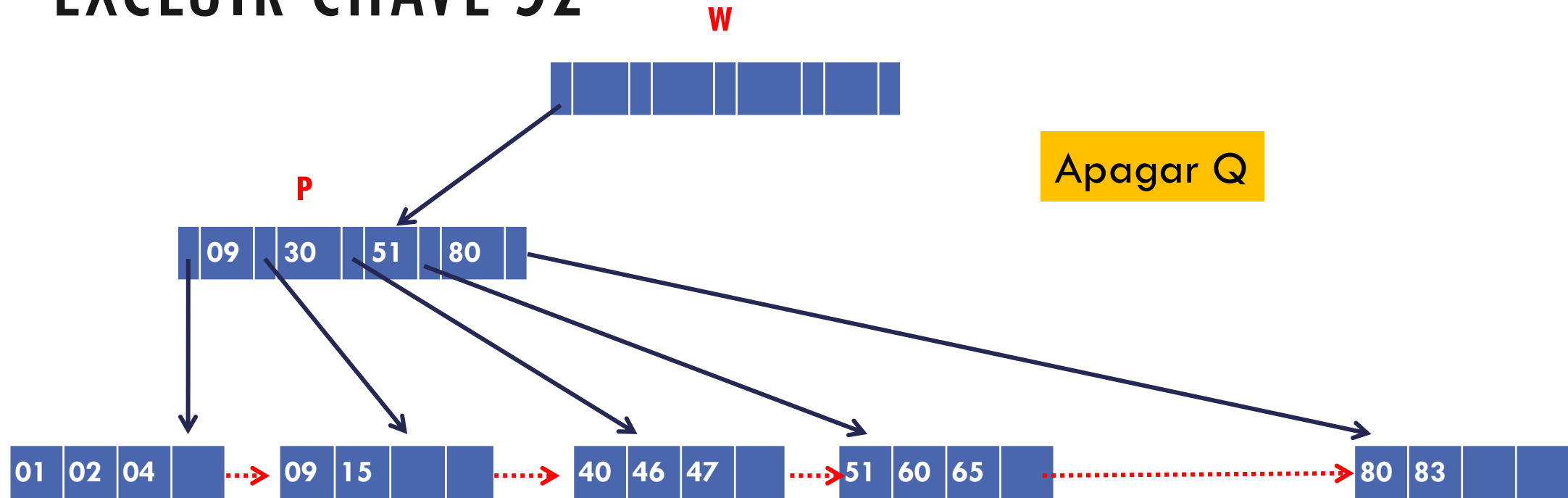
ordem  $d = 2$



# EXEMPLO DE EXCLUSÃO EM ÁRVORE B+

## EXCLUIR CHAVE 52

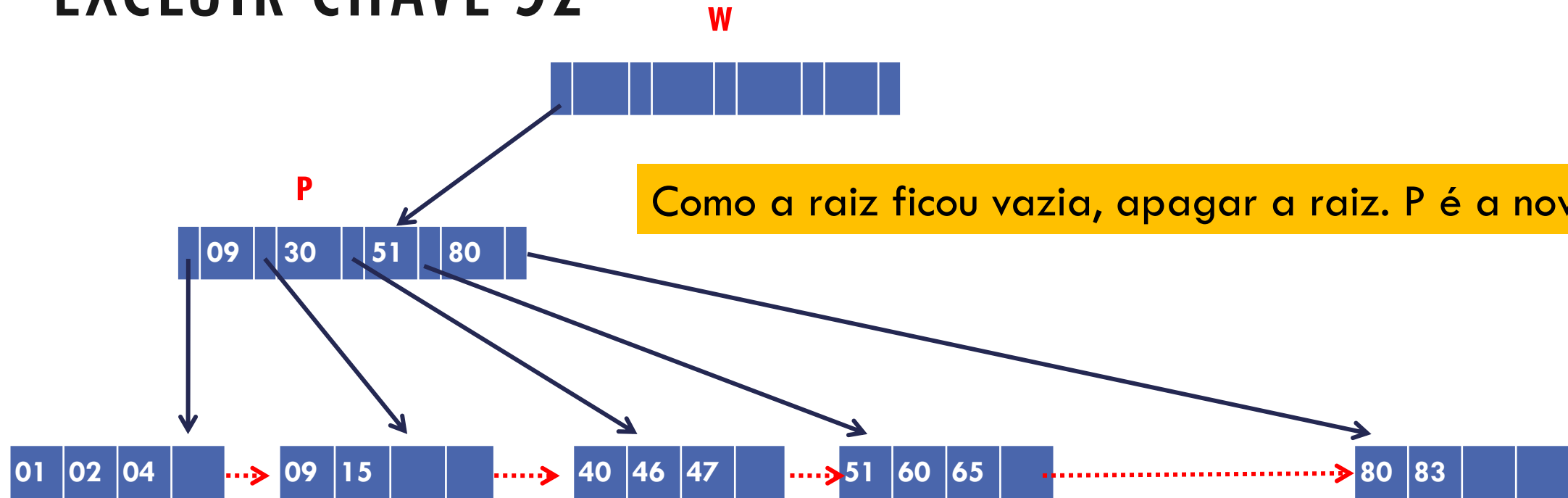
ordem  $d = 2$



# EXEMPLO DE EXCLUSÃO EM ÁRVORE B+

## EXCLUIR CHAVE 52

ordem  $d = 2$

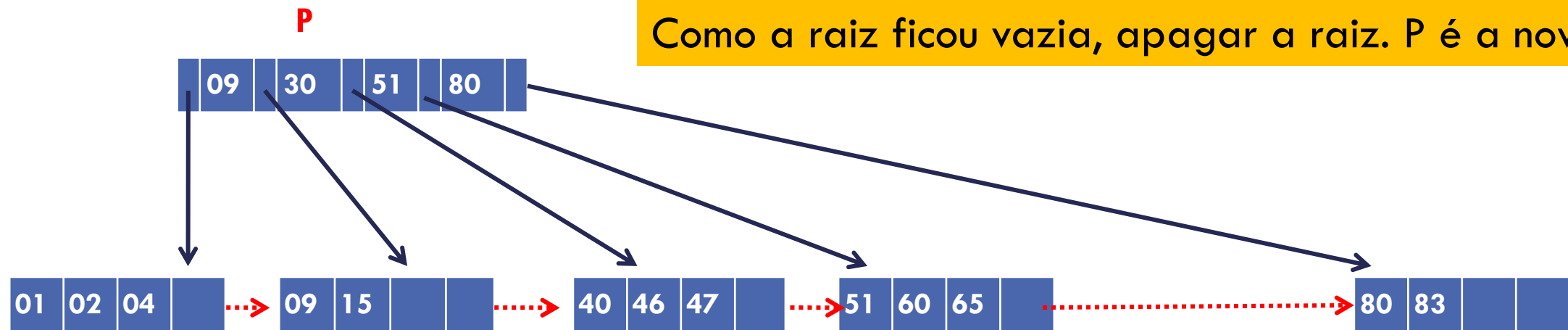




# EXEMPLO DE EXCLUSÃO EM ÁRVORE B+

## EXCLUIR CHAVE 52

ordem  $d = 2$



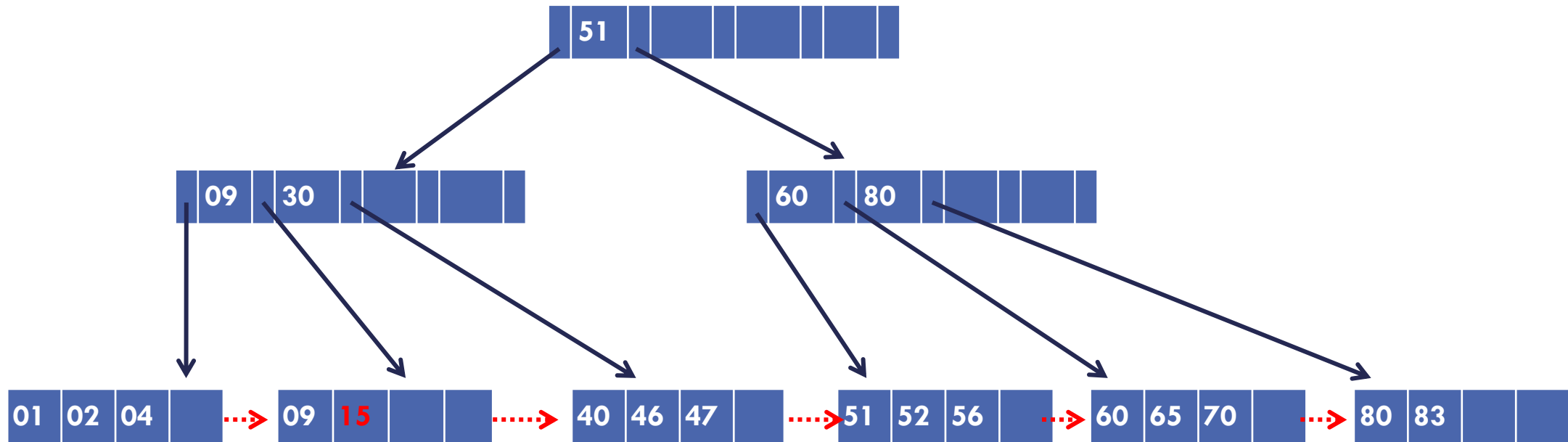
# EXCLUSÃO QUE CAUSA REDISTRIBUIÇÃO

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

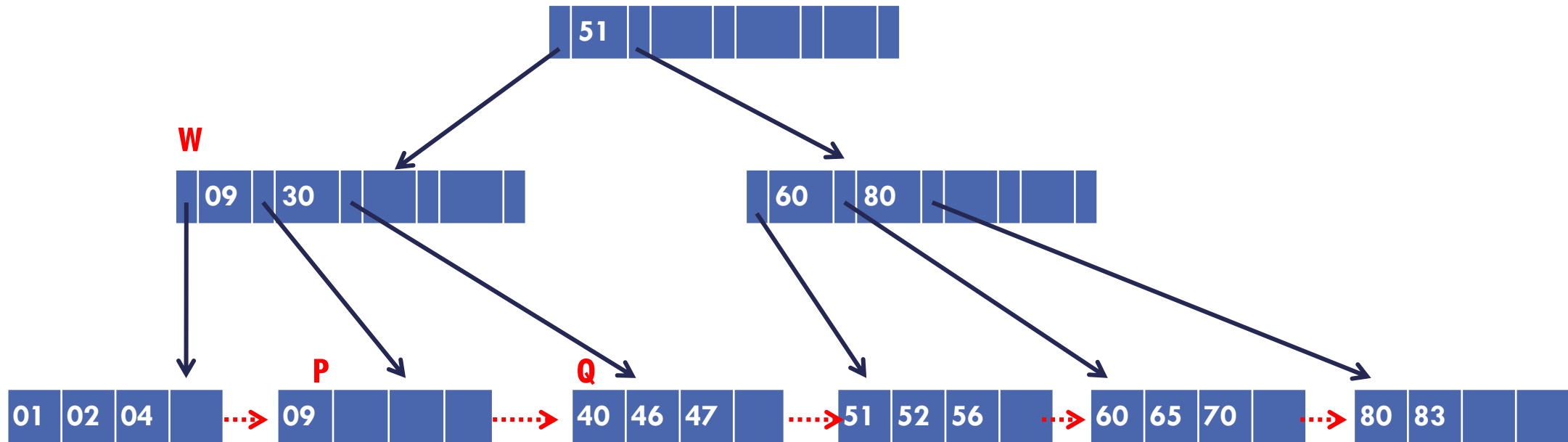
ordem  $d = 2$



# EXEMPLO DE EXCLUSÃO EM ÁRVORE B+

## EXCLUIR CHAVE 15

ordem  $d = 2$

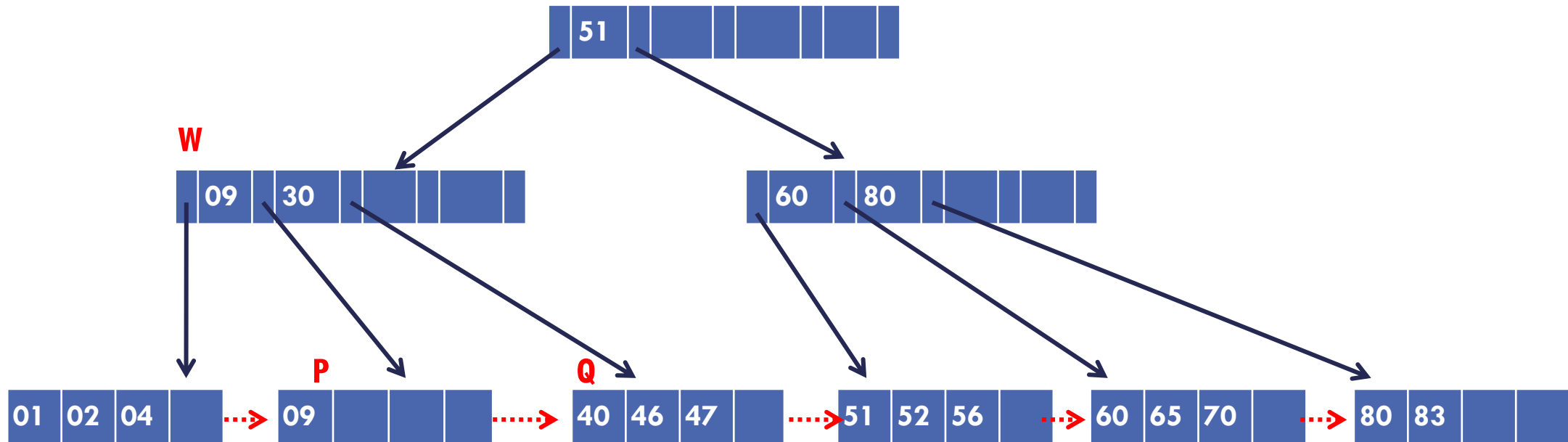


Nó ficou com menos de  $d$  entradas – necessário tratar isso  
A soma dos registros dos irmãos adjacentes é  $\leq 2d$   
Solução: **redistribuição**  
Como existem 2 opções, vamos optar pelo **nó da direita**

# EXEMPLO DE EXCLUSÃO EM ÁRVORE B+

## EXCLUIR CHAVE 15

ordem  $d = 2$

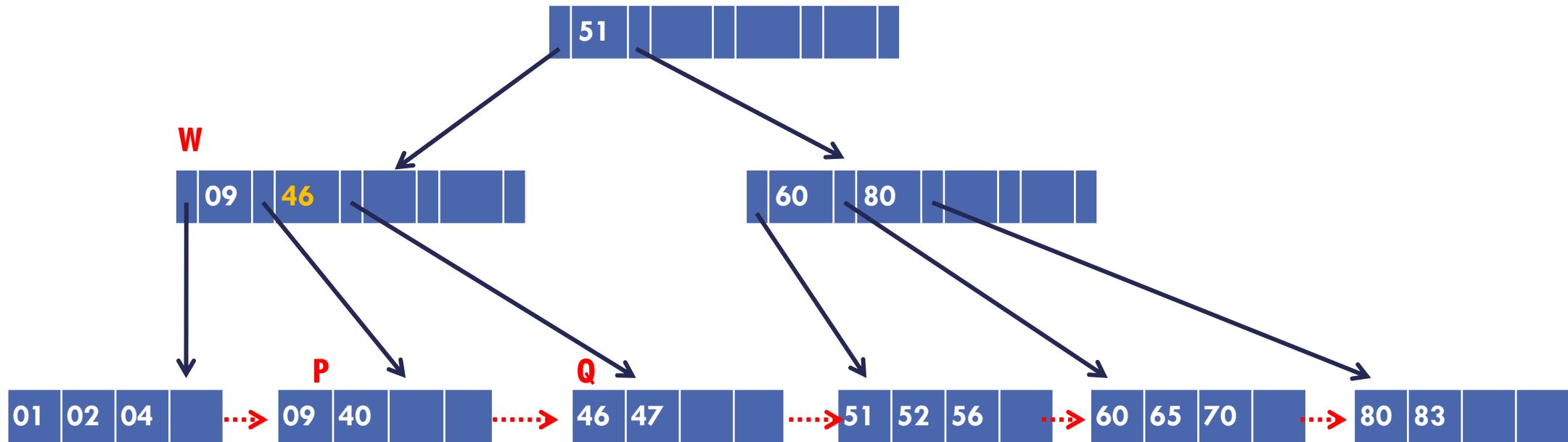


**MAS...** Se a chave do nó pai não precisa descer (porque não tem conteúdo, tem apenas a chave), porque não podemos concatenar P e Q, já que nesse exemplo a soma é  $= 2d$ ?  
**Resposta:** ao concatenar P e Q, a página concatenada ficaria cheia, e a próxima inserção neste nó causaria um particionamento. Para evitar isso, continuamos obedecendo o critério: fazer concatenação apenas quando a soma da quantidade de chaves  $< 2d$ , e, sempre que tivermos as duas opções, optaremos pela redistribuição, que não se propaga.

# EXEMPLO DE EXCLUSÃO EM ÁRVORE B+

## EXCLUIR CHAVE 15

ordem  $d = 2$



(09; 40; 46; 47)

d primeiras chaves ficam em P

Chave  $d+1$  sobe para substituir a chave que já existia lá

Registros  $d+1$  em diante ficam em Q

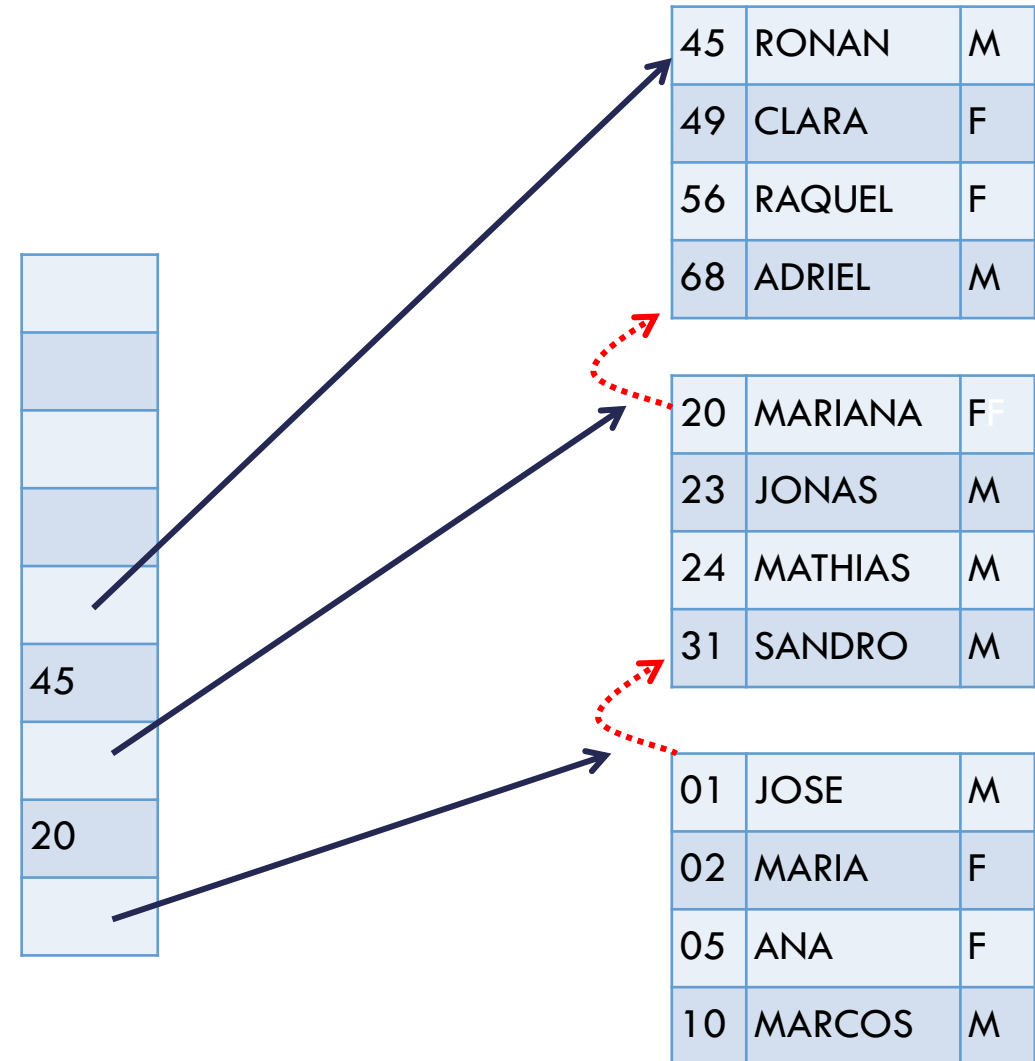
Note que a chave 46 sobe para W para substituir a chave 30, mas o registro correspondente é colocado em Q.

# EXEMPLO

(MOSTRANDO OS DADOS NAS FOLHAS)

Neste exemplo, a árvore B+ tem o nó raiz e 3 folhas

Ordem da árvore é  $d = 2$



# CONSIDERAÇÕES SOBRE IMPLEMENTAÇÃO EM DISCO

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  **$d = (B / k) - 1$**

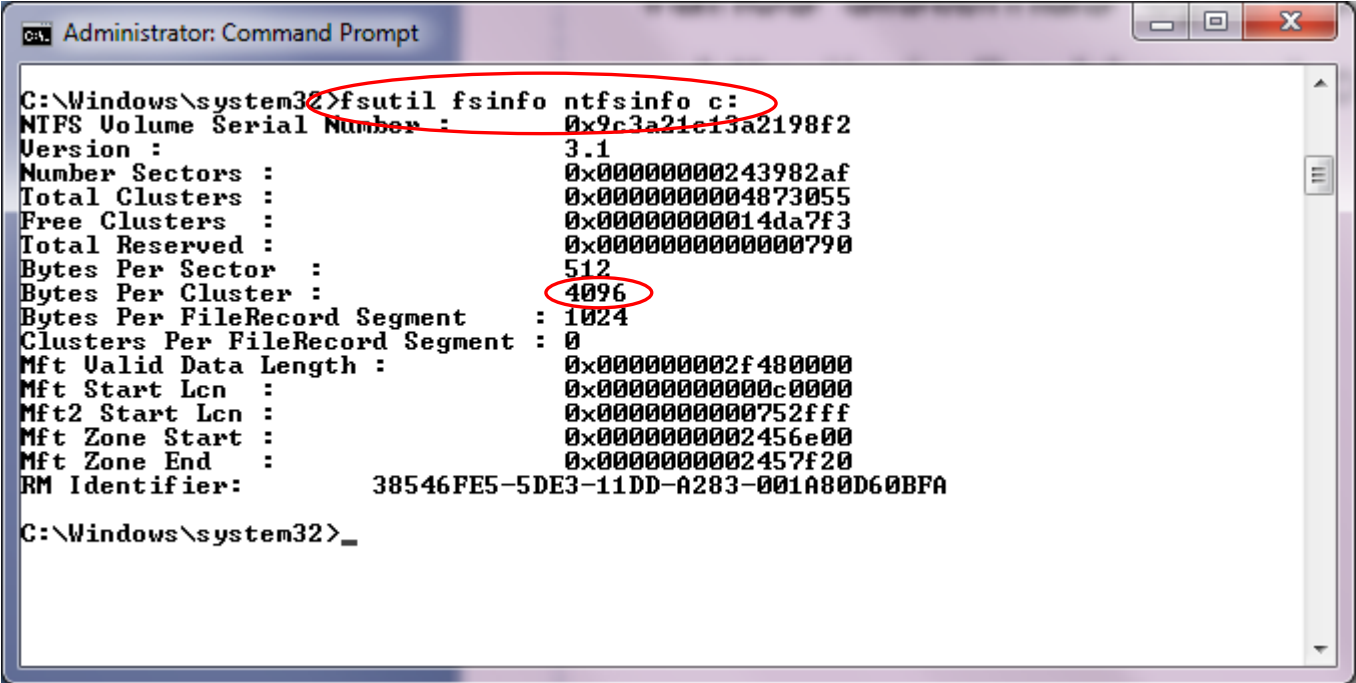
Exemplo prático:

- Tamanho do bloco do disco  $B = 4KB = 4096$  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 : 0x000000000243982af
Total Clusters : 0x0000000004873055
Free Clusters : 0x000000000014da7f3
Total Reserved : 0x0000000000000790
Bytes Per Sector : 512
Bytes Per Cluster : 4096
Bytes Per FileRecord Segment : 1024
Clusters Per FileRecord Segment : 0
Mft Valid Data Length : 0x0000000002f480000
Mft Start Lcn : 0x000000000000c0000
Mft2 Start Lcn : 0x00000000000752fff
Mft Zone Start : 0x00000000002456e00
Mft Zone End : 0x00000000002457f20
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

# IMPLEMENTAÇÃO

Implementar busca, inserção e exclusão em árvore B+ armazenada em disco.  
Detalhes estão no Google Classroom.

# REFERÊNCIA

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

# AGRADECIMENTOS

Exemplo cedido por Renata Galante