Árvores B

Estrutura de Dados II

Jairo Francisco de Souza

Motivação

- Quando tabelas são muito grandes
  - Armazenamento do conjunto de chaves não pode ser efetuado na memória principal
  - Necessário uso de memória secundária
    - Dispêndio de tempo para acesso a um nó da tabela
  - Interesse
    - Criação de estrutura que minimize o tempo de acesso para a busca, inserções e remoções

Árvore B

- Árvore B
  - (Bayer e MacCreight, 1972)
  - Nós podem ter mais de uma chave
  - Operações de busca, inserção e remoção são executadas rapidamente
  - Sua construção assegura que todas as folhas estão em um mesmo nível, não importando a entrada de dados.
  - São amplamente utilizadas para armazenamento em memória secundária
    - Ex: Banco de dados

Árvore B

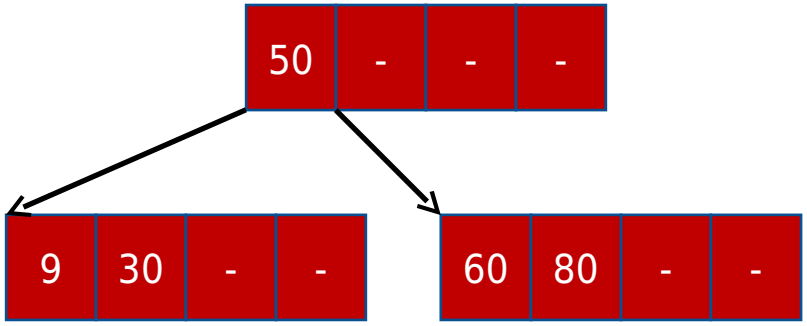
- Árvore B
  - O tamanho de cada nó pode ser tão grande quanto o tamanho de um bloco do disco.
  - O número de chaves de um nó pode variar dependendo do:
    - Tamanho das chaves
    - Tamanho de um bloco (alguns autores chamam de página)
  - Tamanho do bloco varia para cada sistema
    - 512 bytes, 4K ou mais

Árvore B

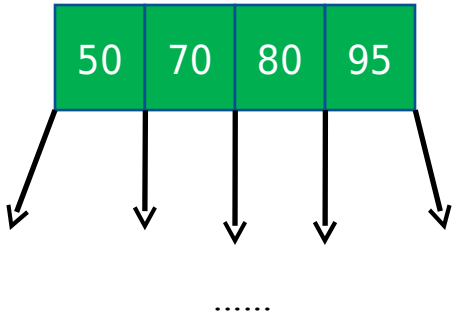
- **Propriedades**
  - A raiz é uma folha ou tem no mínimo dois filhos
  - Cada nó diferente da raiz e das folhas possui no mínimo $d + 1$ filhos
  - Cada nó tem no máximo $2d + 1$ filhos
- d é chamado de ordem da árvore
- Exemplos de árvores de ordem $d = 2$

Raiz sem filhos

Raiz com dois filhos



Nó com $2d + 1$ filhos

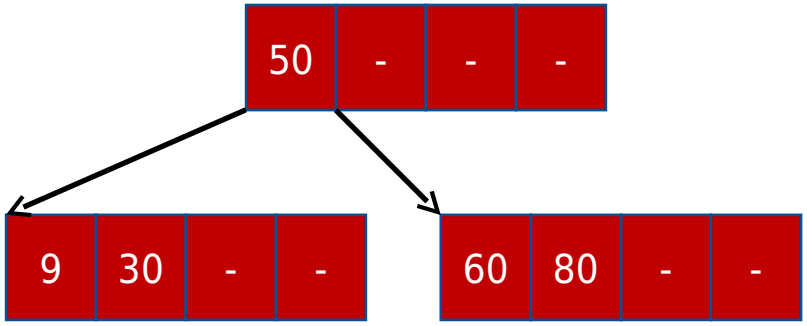


Árvore B

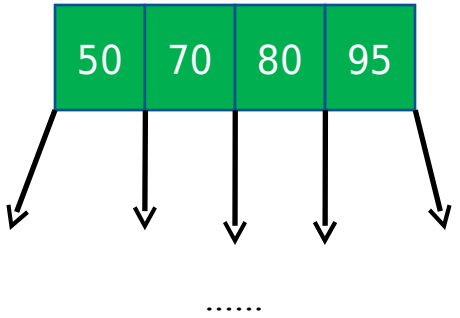
- **Propriedades**
  - Cada nó possui entre *d* e 2*d* chaves, exceto o nó raiz que possui entre 1 e 2*d* chaves.
  - Seja *m* o número de chaves em um nó *P* não folha. Então *P* tem *m* + 1 filhos
  - Em cada página P as chaves estão ordenadas

Raiz sem filhos

Raiz com dois filhos

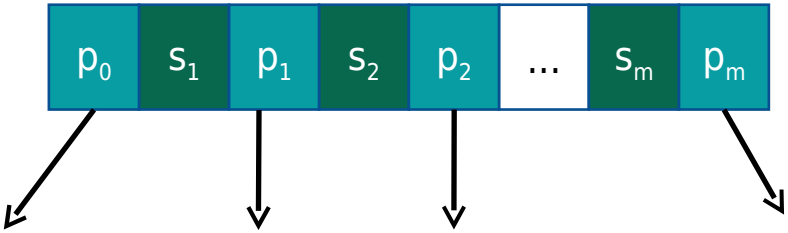


Nó com 2*d* + 1 filhos



Árvore B

- **Propriedades**
  - Seja as chaves de um nó
    - s_1, ..., s_m, $d \leq m \leq 2d$, exceto para a página raiz onde $1 \leq m \leq 2d$;
    - p_0, p_1, ..., p_m ponteiros para os filhos de P

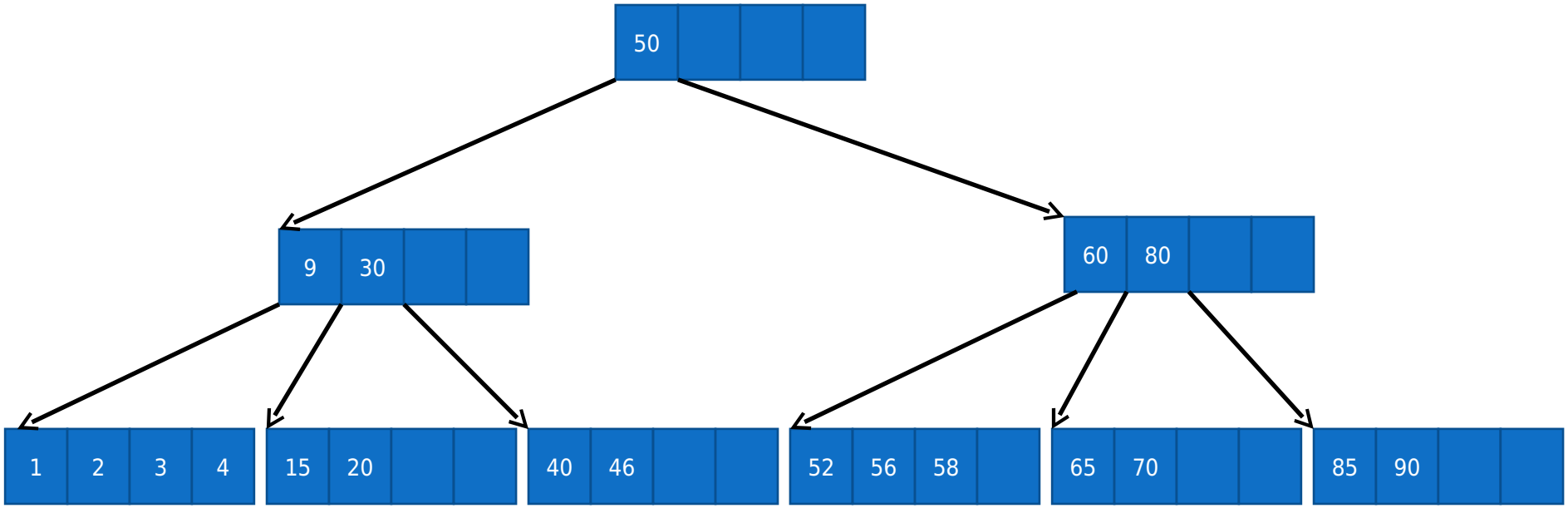


    - Para qualquer chave y, pertencente ao nó apontado por p_0, $y < s_1$;
    - Para qualquer chave y, pertencente ao nó apontado por p_k, $1 \leq k \leq m\text{-}1$, $s_k < y < s_{k+1}$;
    - Para qualquer chave y, pertencente ao nó apontado por p_m, $y > s_m$.

Árvore B - Implementação

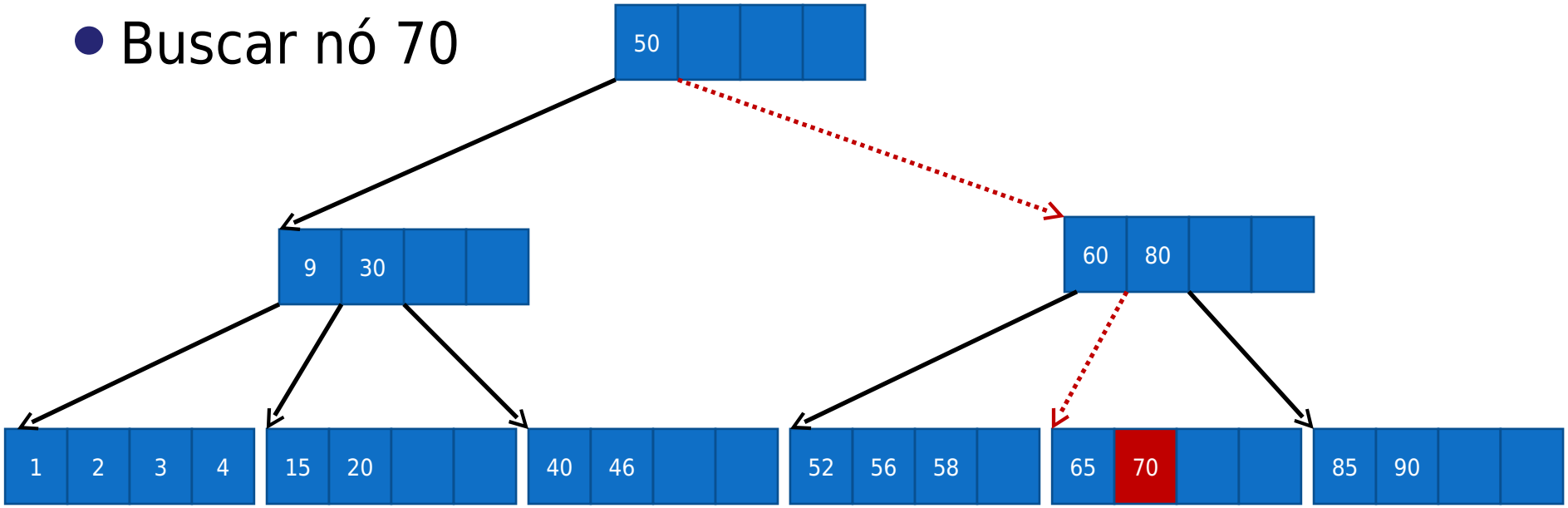
- Classe contendo
  - Array de m posições para as m chaves ($2d$ chaves)
  - Array de $m + 1$ posições para os ponteiros para os outros nós
  - Inteiro para o número de chaves do nó
  - Booleano para indicar se o nó é folha ou nó interno

Árvore B - Exemplo



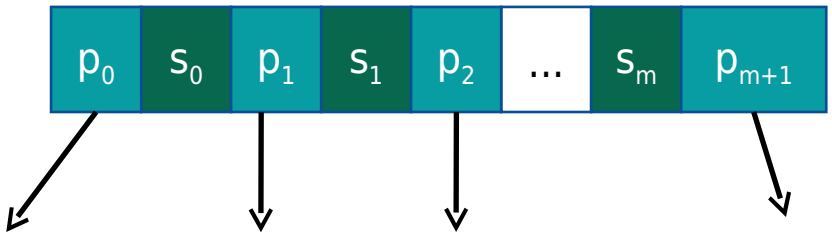
Árvore B - Busca

- Busca é semelhante a busca na árvore binária de busca
- Buscar nó 70



Árvore B - Busca

- O algoritmo da busca compara a chave x, a chave procurada, com a chave (ou as chaves) do nó raiz.
- Se a chave não se encontra no nó em questão, a busca deve prosseguir em um filho deste nó, de acordo com a propriedade:
  - Para qualquer chave *y*, pertencente ao nó apontado por p_0, $y < s_0$;
  - Para qualquer chave *y*, pertencente ao nó apontado por p_k, $1 \leq k \leq m$, $s_k < y < s_{k+1}$;
  - Para qualquer chave *y*, pertencente ao nó apontado por p_{m+1}, $y > s_m$.



Árvore B – Busca - Algoritmo

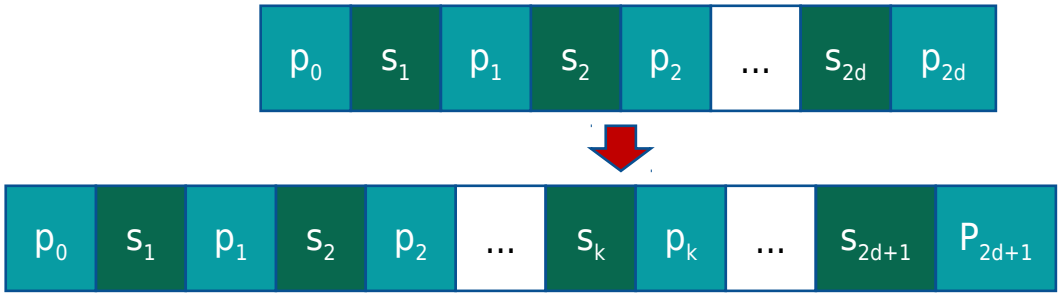
- procedimento buscaB(x, pt, f, g)
  - f (resultado da busca):
    - Se f=1 (encontrou);
      - pt aponta para o nó;
      - g aponta para a posição da chave dentro do nó
    - Se f=0 (não encontrou)
      - pt aponta aponta para o último nó pesquisado (necessariamente uma folha)
      - g aponta informa a posição onde a nova chave deve ser inserida

Árvore B – Busca - Algoritmo

```
procedimento buscaB(x, pt, f, g)
  p = ptraiz; pt=NULL; f :=0;
  enquanto p ≠ NULL faça
     i = g = 0; pt = p;
     enquanto i ≤ m faça
        se x > p->s[i] então i = g = i+1;
        senão se x = p->s[i] então
                      f = 1;
                      p = NULL;
              senão p = p->filho[i]
              i = m + 2
     se i = m + 1 então p = p->filho[m+1];
```

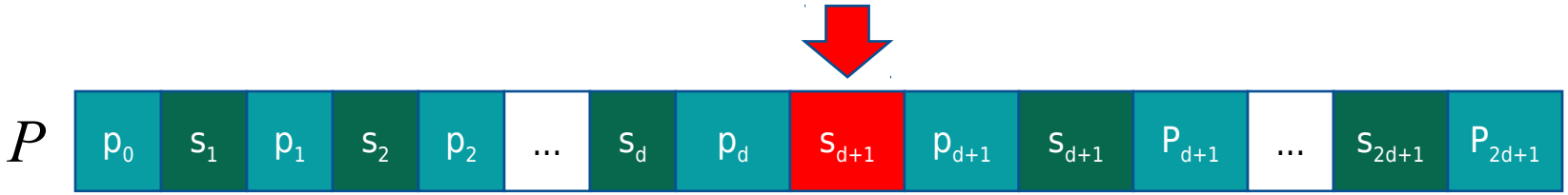
Árvore B – Inserção

- Primeiro passo: executar BuscaB
- Em seguida, analisar valores de f
  - f = 1 trata-se de inserção inválida
  - f = 0
    - x deve ser incluída na g-ésima posição da folha apontada por pt.
- Problema
  - Folha já contém 2*d* chaves
    - Inserir nova chave leva ao nó ter 2*d* + 1 chaves que não é permitido.
- Solução: reorganizar as páginas
  - Cisão de nó

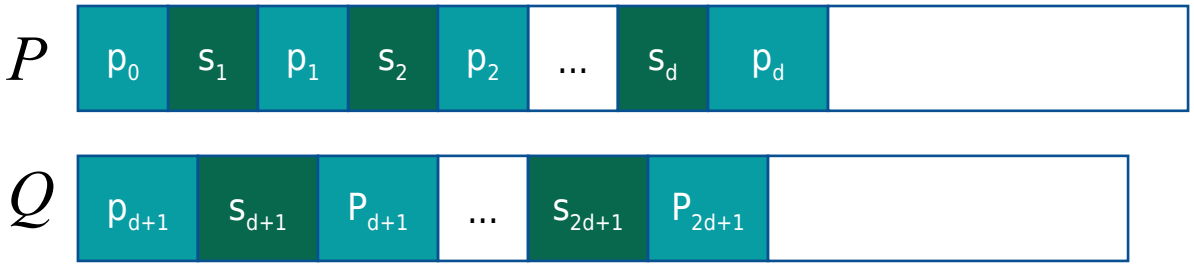


Árvore B – Inserção - Cisão

- Cisão de páginas
  - Seja um nó *P* com *overflow*
  - Transforma 1 nó com excesso em 2 nós
  - No nó *P* apontado por pt ficam *d* entradas
  - Um novo nó *Q* é criado, nele são incluídas *d* entradas
  - A chave central é incluída no nó pai (*W*) de *P* juntamente com um ponteiro para o nó Q

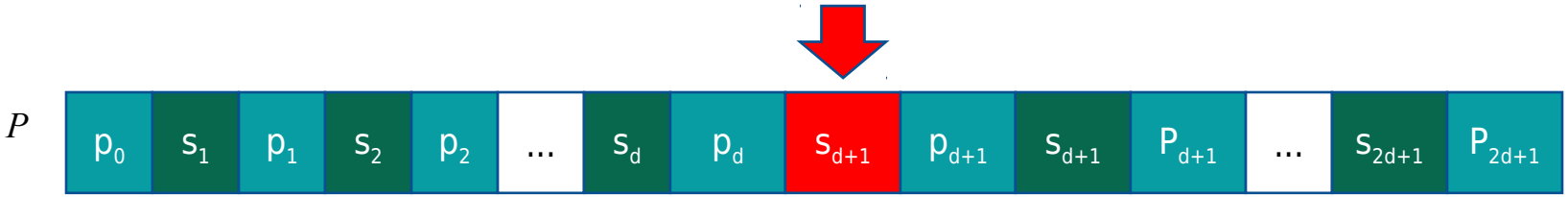


  - Cisão de *P*

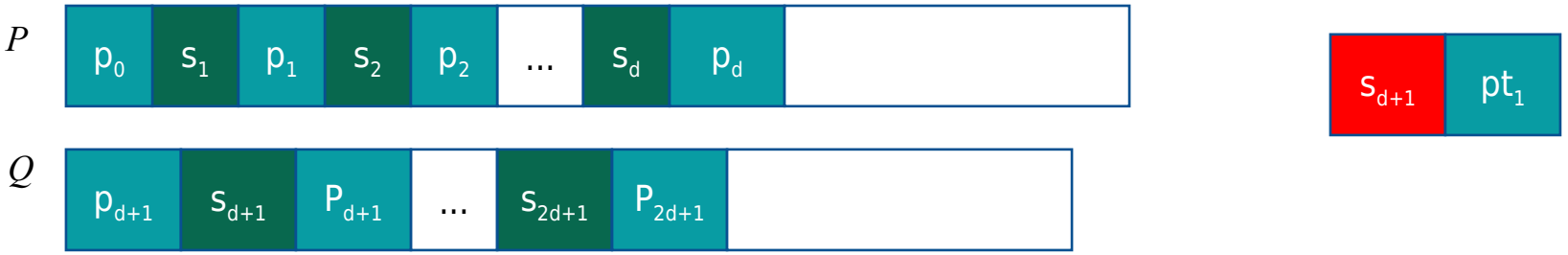


Árvore B – Inserção - Cisão

- Cisão de páginas
  - Seja um nó *P* com *overflow*
  - Transforma 1 nó com excesso em 2 nós
  - No nó *P* apontado por pt ficam *d* entradas
  - Um novo nó *Q* é criado, nele são incluídas *d* entradas
  - A chave central é incluída no nó pai (*W*) de *P* juntamente com um ponteiro para o nó Q

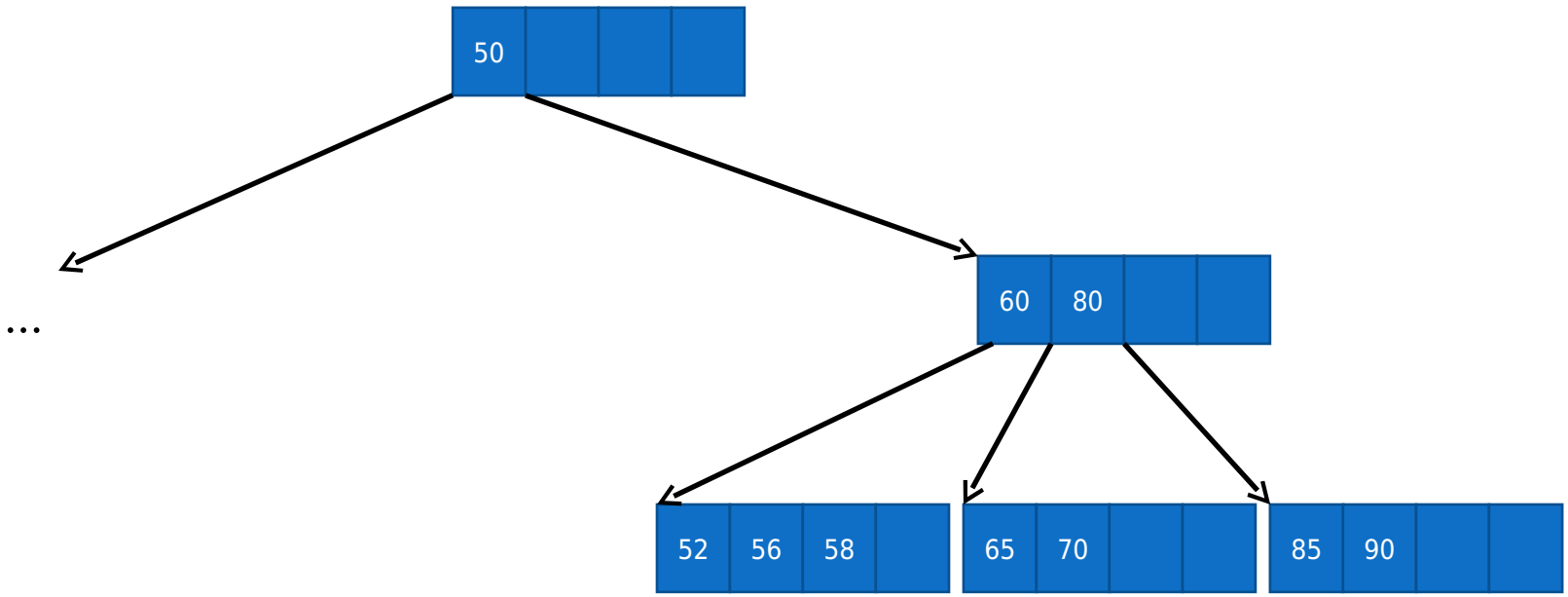


  - Cisão de *P*



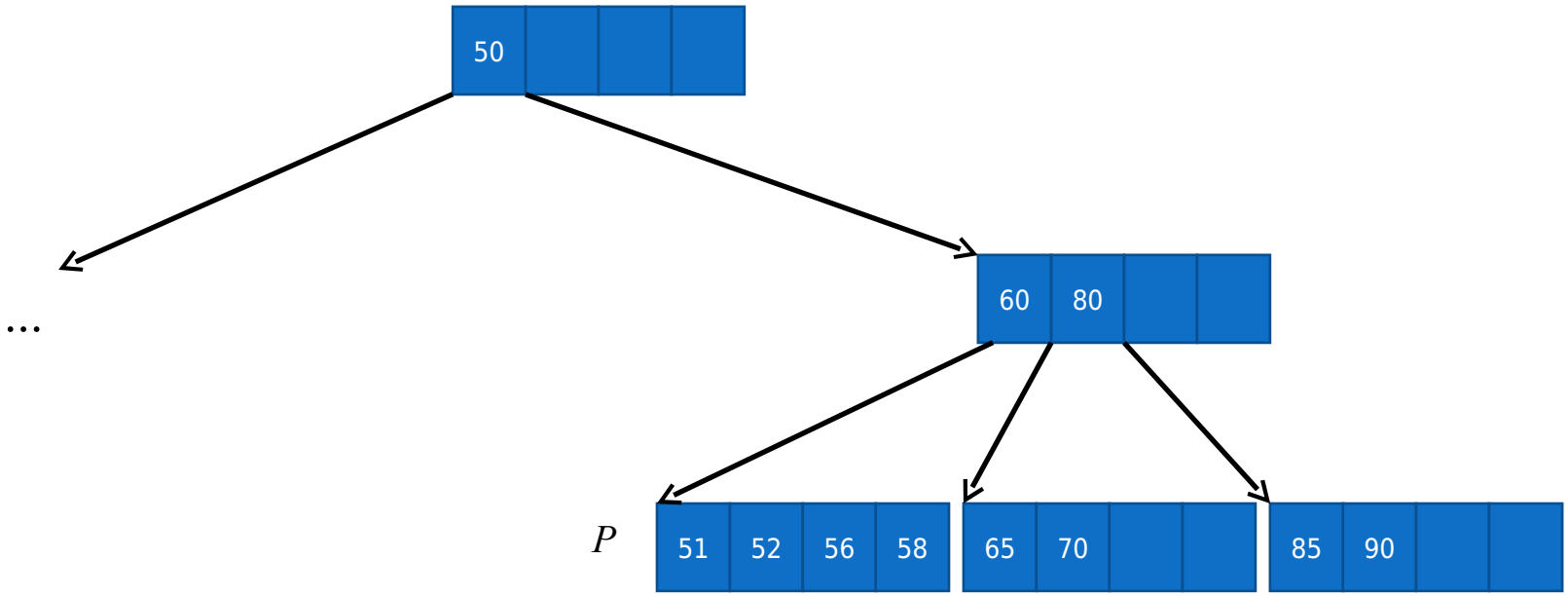
Árvore B – Inserção - Exemplo

- Inserir 51



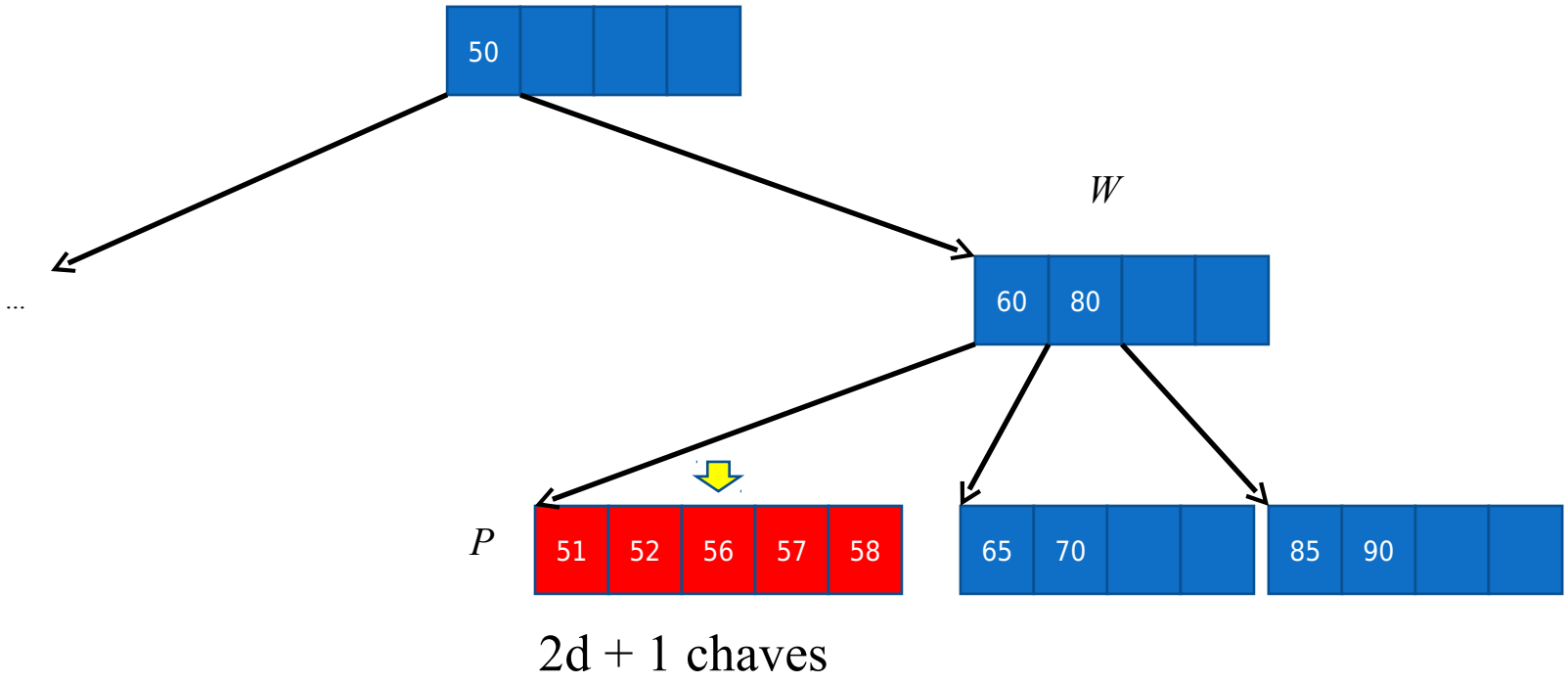
Árvore B – Inserção - Exemplo

- Inserir 57
  - Necessidade de cisão (*split*)



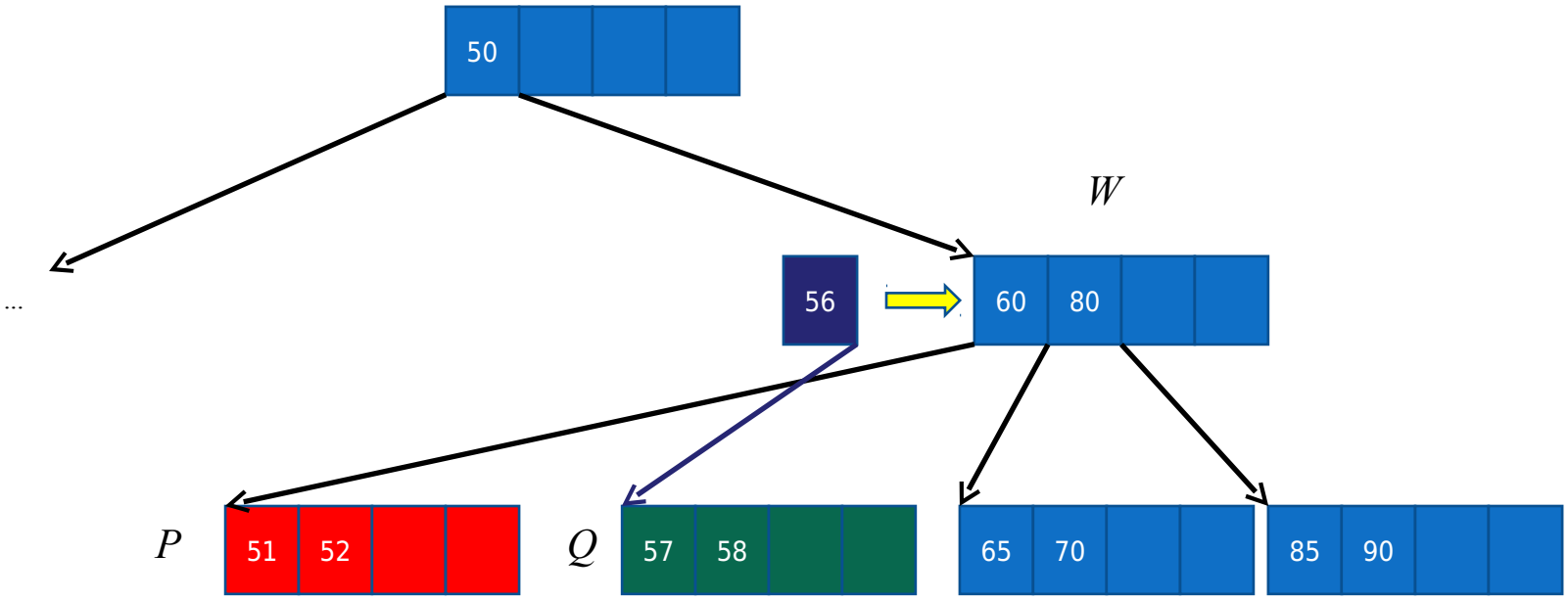
Árvore B – Inserção - Exemplo

- Inserir 57
  - Necessidade de cisão (*split*)



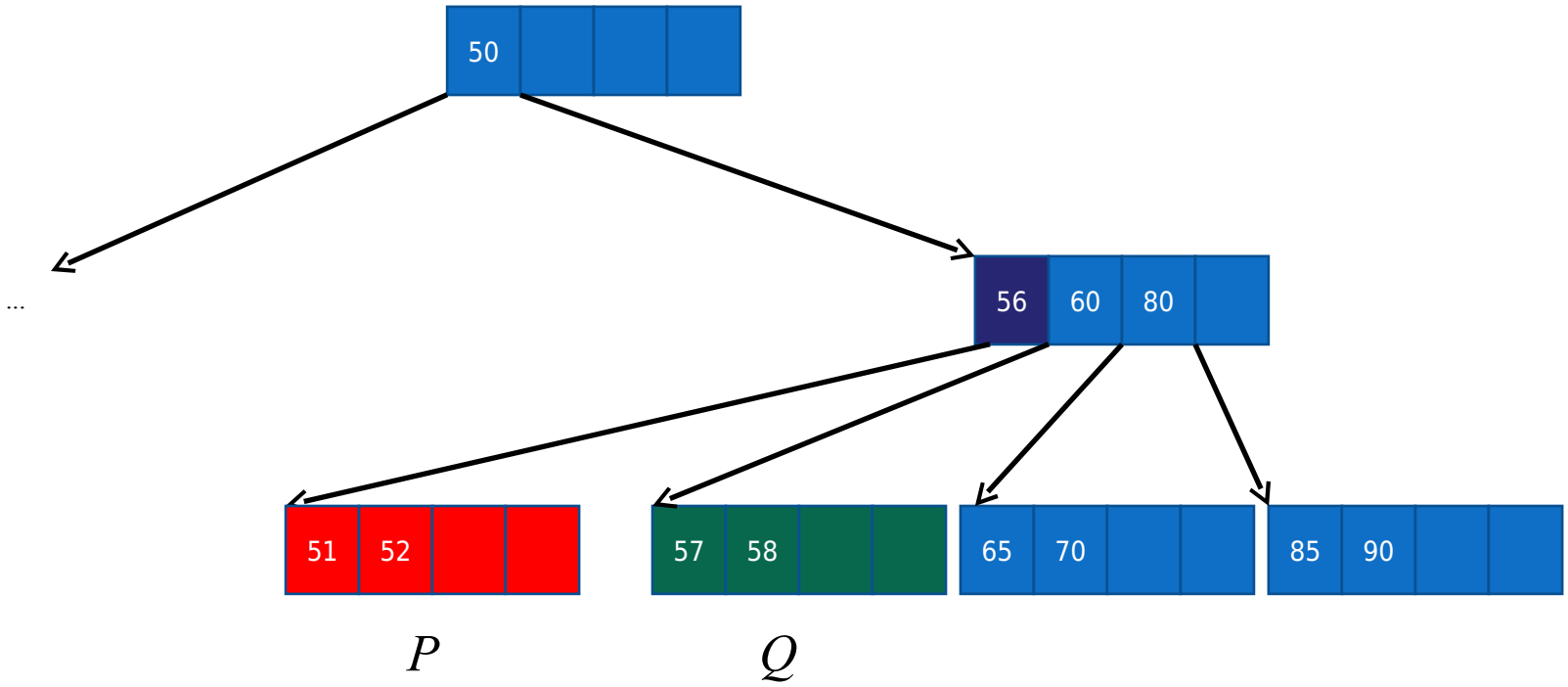
Árvore B – Inserção - Exemplo

- Inserir 57
  - Necessidade de cisão (*split*)



Árvore B – Inserção - Exemplo

- Inserir 57
  - Necessidade de cisão (*split*)



Árvore B – Inserção - Algoritmo

- Passo 1: aplicar procedimento BuscaB, verificando a validade da inserção
- Passo 2: se a inserção é válida, incluir a nova entrada na g-ésima posição da folha *F*, apontada por *pt*.
- Passo 3: verificar se a página *F* necessita de cisão. Em caso afirmativo, propagar a cisão enquanto necessário.

Árvore B - Remoção

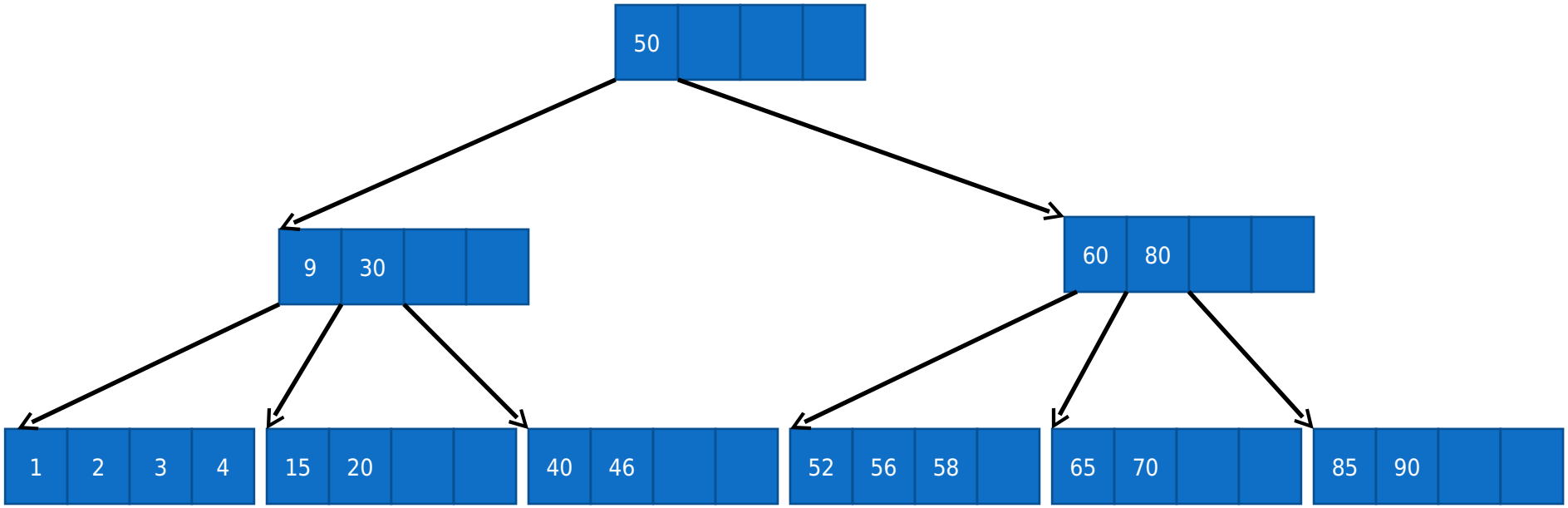
- 2 casos possíveis:
  - Chave *x* se encontra em uma folha:
    - a entrada é simplesmente removida
  - Chave *x* não se encontra em uma folha:
    - Chave *x* é substituída pela chave *y* imediatamente maior
      - y necessariamente pertence a uma folha
- A retirada de uma chave pode levar a um nó ter menos do que *d* chaves.
  - O que não é permitido

Árvore B - Remoção

- Tratamento de *underflow*
  - Opção 1) Concatenação
    - **Dois nós adjacentes (irmãos), digamos *P* e *Q* podem ser juntados se tiverem menos do que *2d* chaves**
    - Concatenação agrupa as entradas dos dois nós em um único nó
    - No nó pai *W* deixa de existir uma entrada
      - Justamente a chave que se encontra entre os ponteiros para os irmão *P* e *Q*
    - Esta chave passa a fazer parte do novo nó concatenado e seu ponteiro desaparece, uma vez que o nó *Q* foi devolvido
    - Como a soma do número de chaves de *P* e *Q* era menor do que 2*d*, o novo nó tem, no máximo, 2*d* chaves.
    - Concatenação é propagável e pode atingir a raiz.

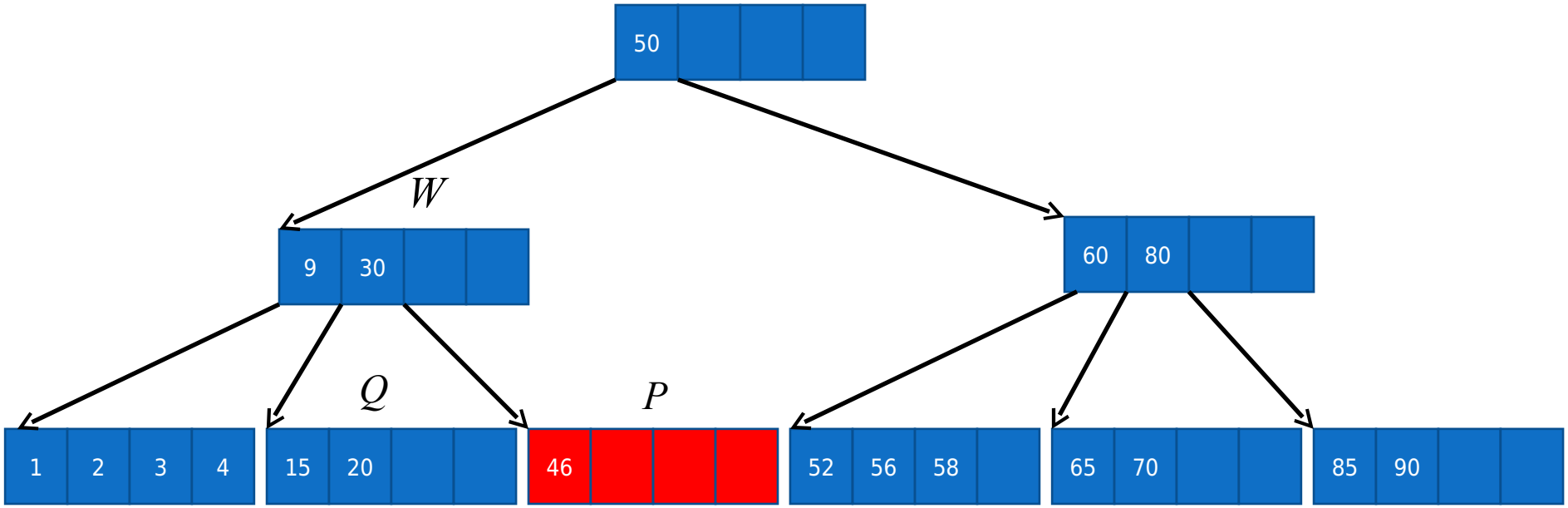
Árvore B – Remoção - Exemplo

- Remover chave 40



Árvore B – Remoção - Exemplo

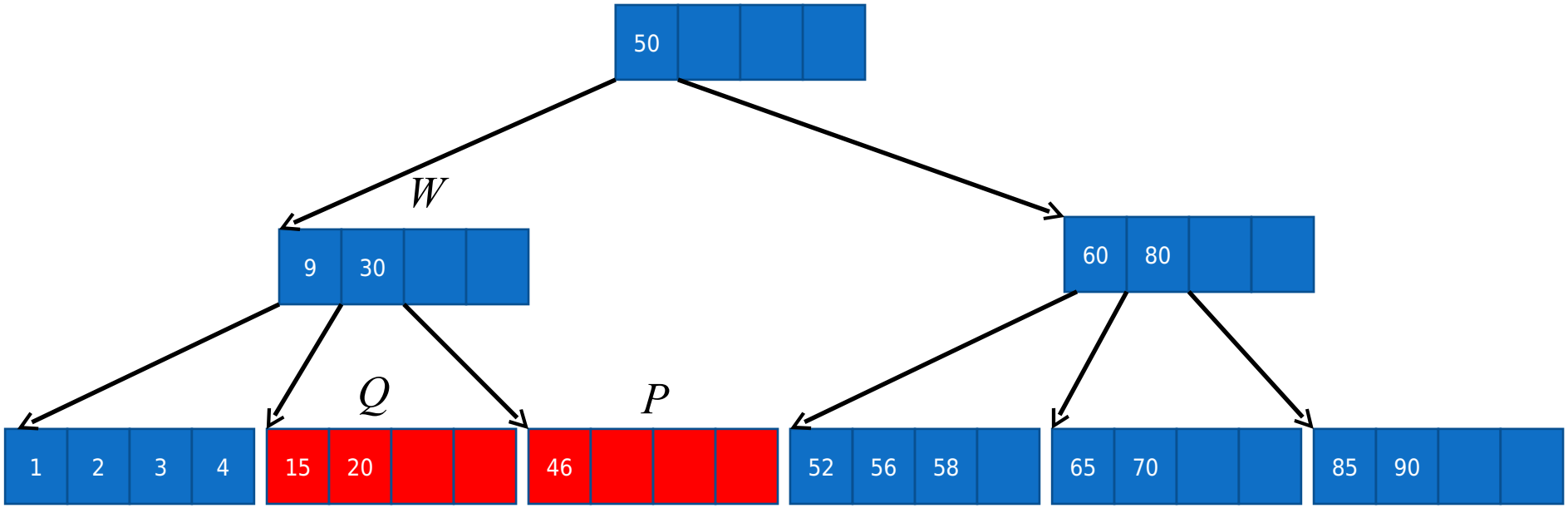
- Resultado da remoção da chave 40



Menos do que d chaves

Árvore B – Remoção - Exemplo

- Pode concatenar nós adjacentes?



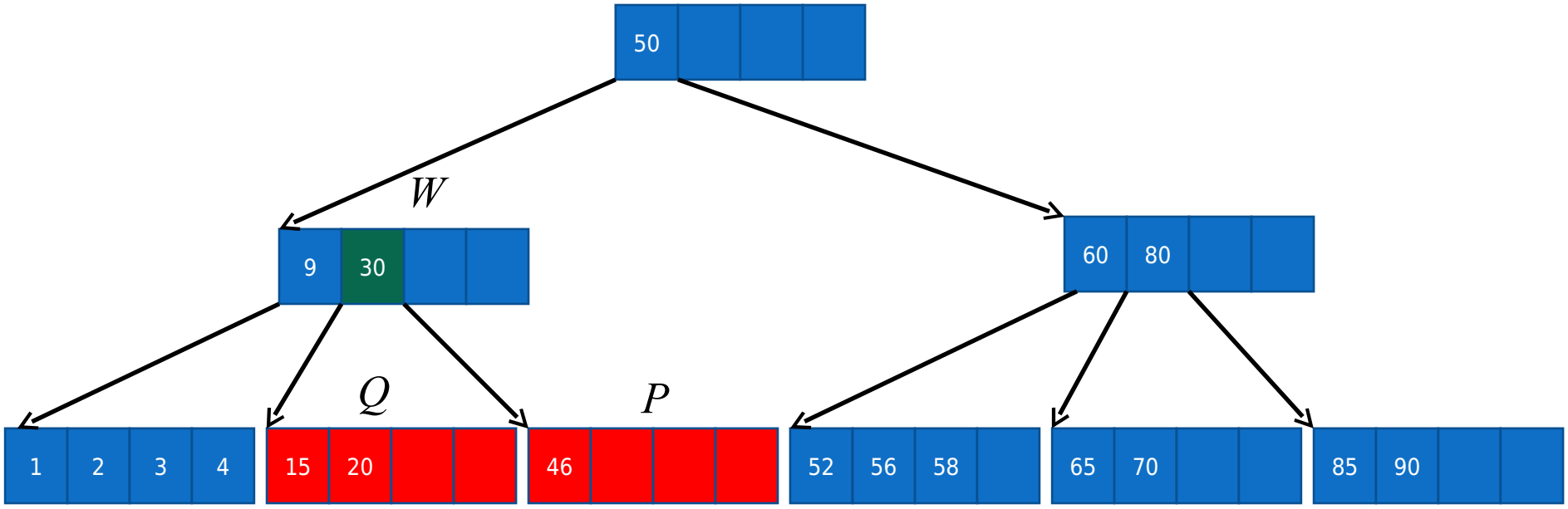
Quantidade das chaves dos nós irmão = 3

2d = 4

Pode fazer a concatenação.

Árvore B – Remoção - Exemplo

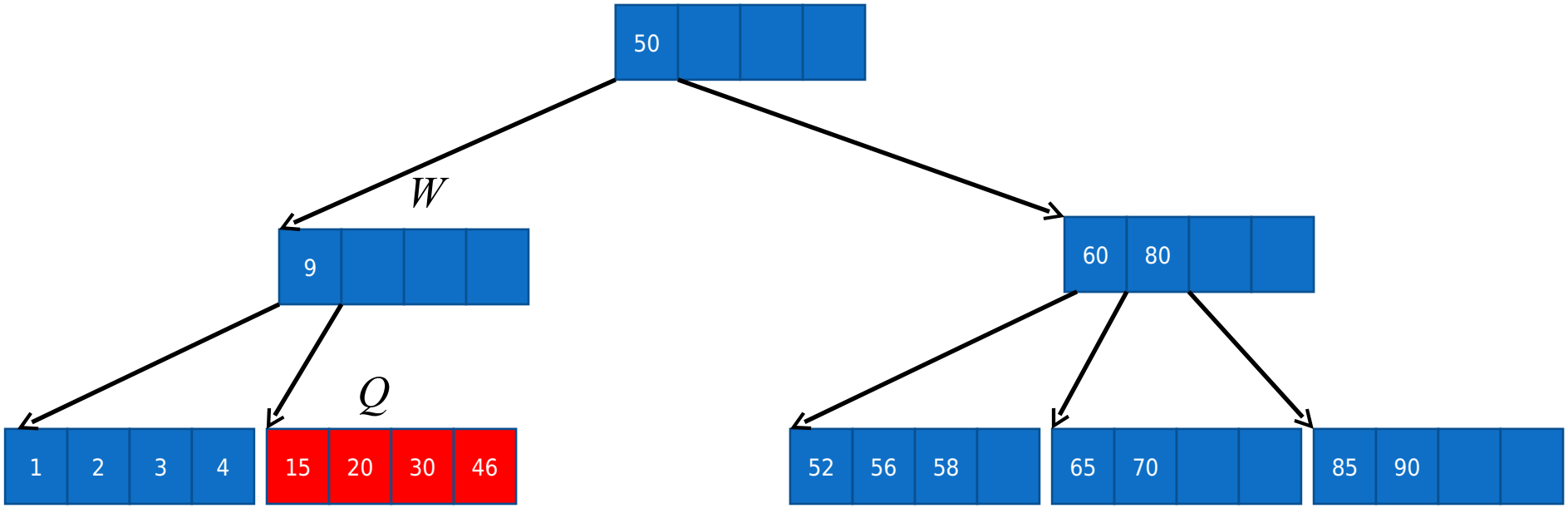
- Todas as entradas envolvidas na concatenação



30 é a chave que está entre os ponteiros que apontam para os nós adjacentes

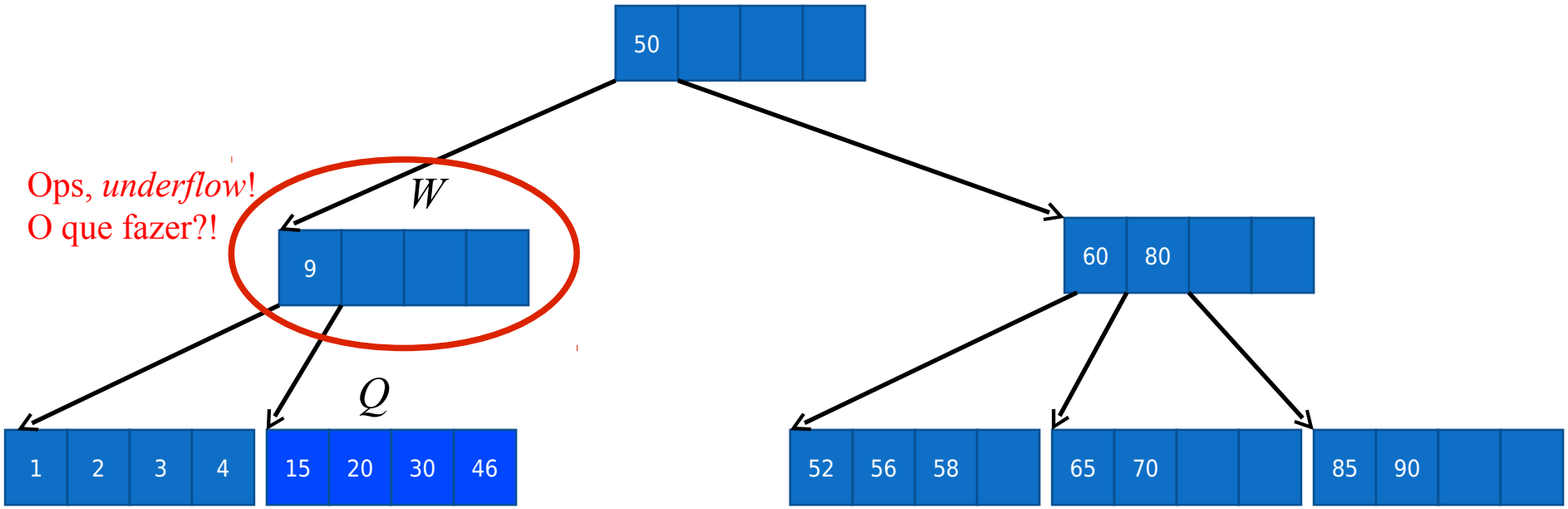
Árvore B – Remoção - Exemplo

- Resultado final



Árvore B – Remoção - Exemplo

- Resultado final

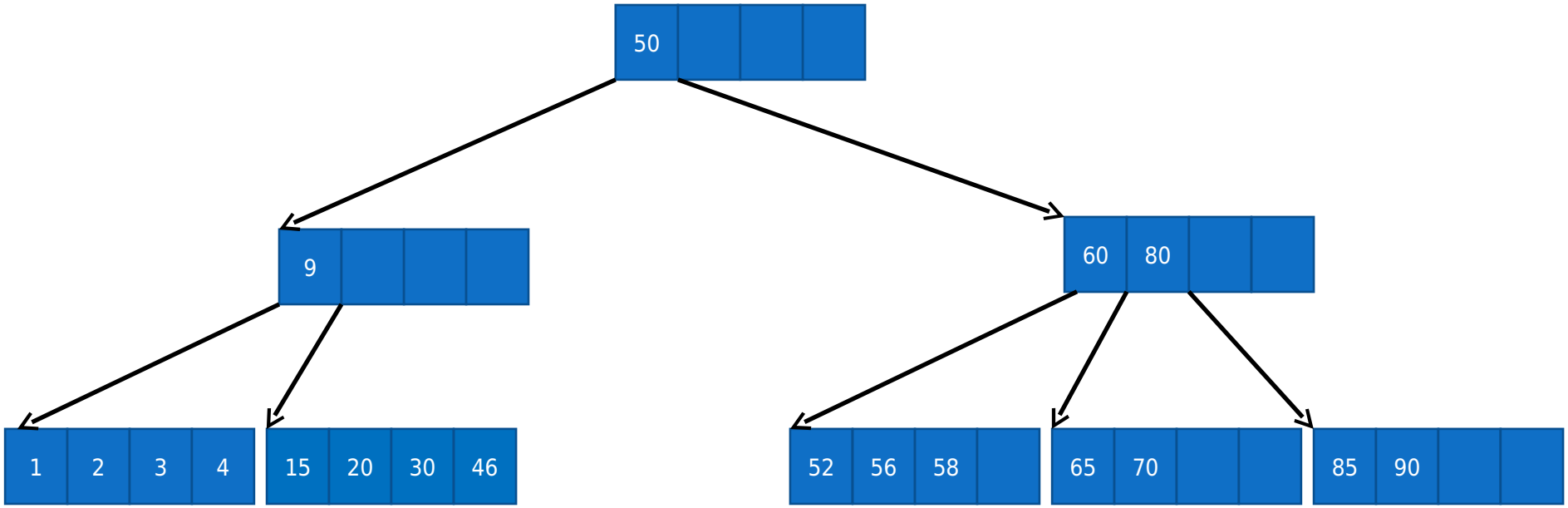


Árvore B - Remoção

- Tratamento de *underflow*
  - Opção 2) Redistribuição
    - **Dois nós adjacentes (irmãos), digamos *P* e *Q* podem ser redistribuídos se tiverem mais do que *2d* chaves**
    - Passos
      - Concatena-se *P* e *Q*
        - Resultado é um nó com mais 2*d* chaves
      - Executa-se a cisão
    - Redistribuição não é propagável
    - A página *W*, pai de *P* e *Q*, é modificada, mas seu número de chaves permanece o mesmo.

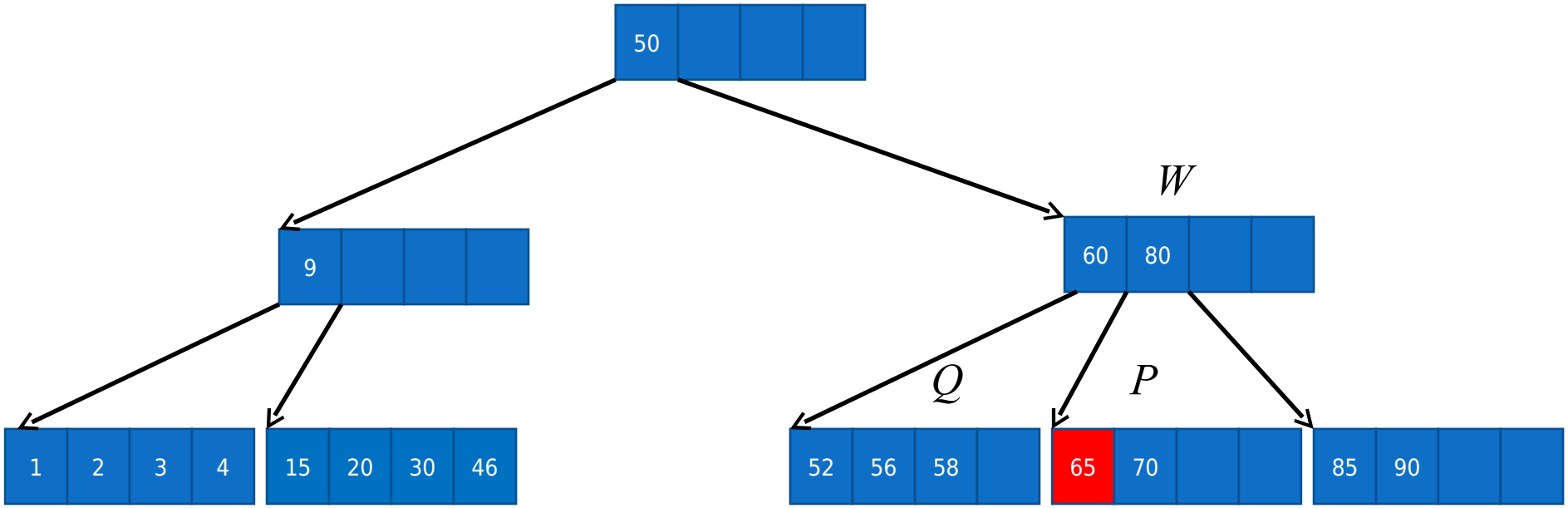
Árvore B – Remoção - Exemplo

- Remover chave 65



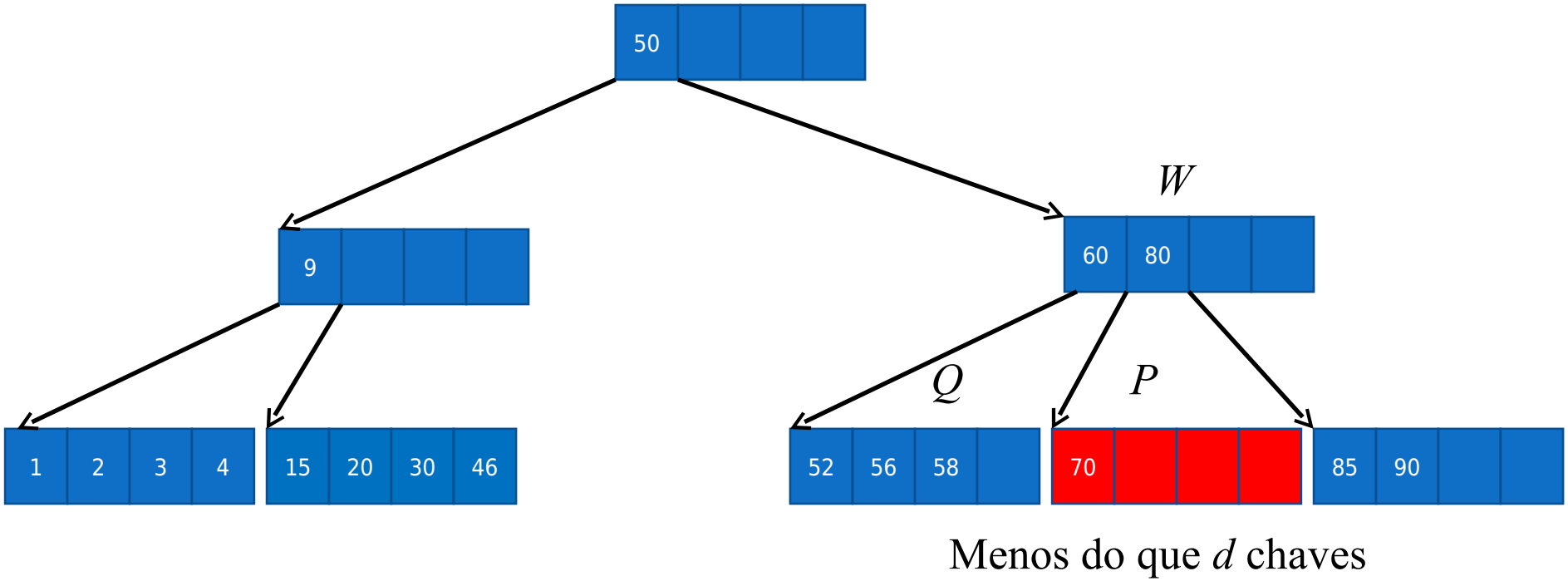
Árvore B – Remoção - Exemplo

- Remover chave 65



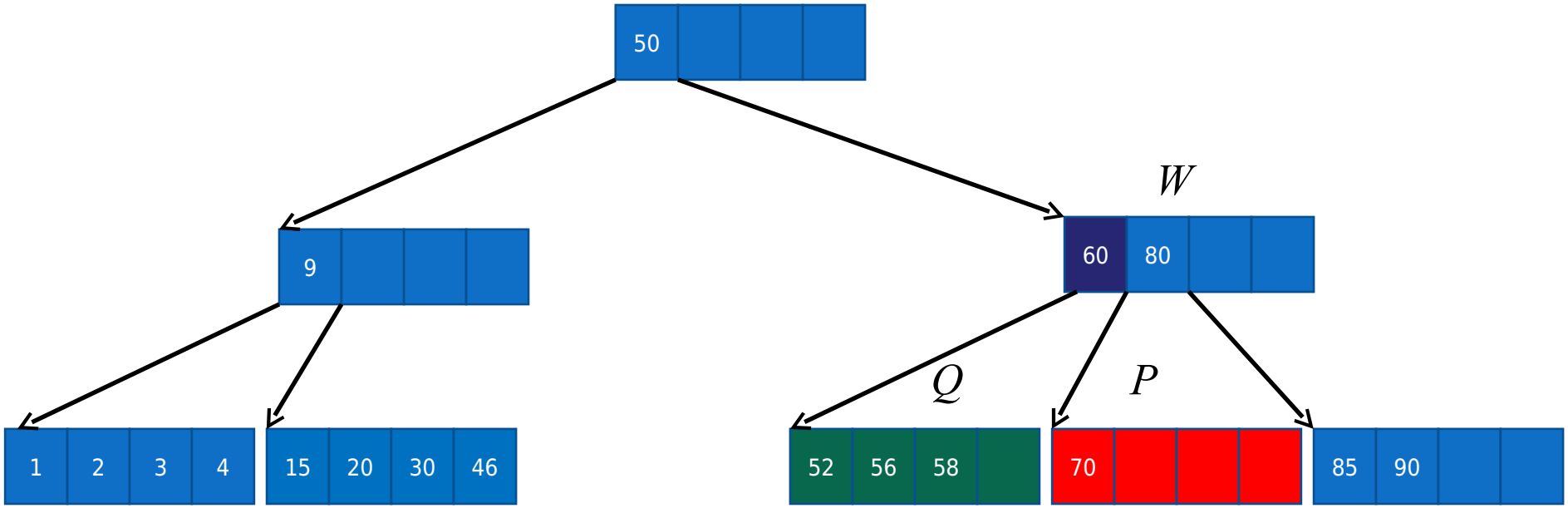
Árvore B – Remoção - Exemplo

- Remover chave 65



Árvore B – Remoção - Exemplo

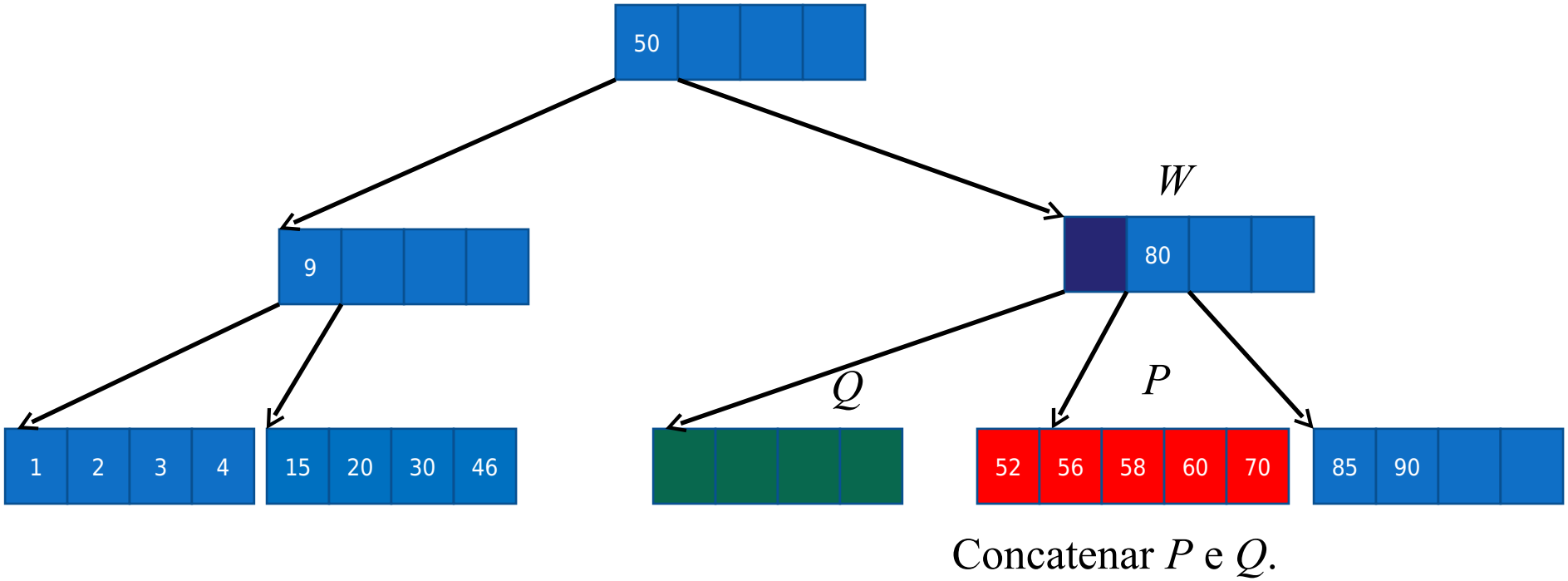
- Remover chave 65



Quantidade de chaves de P + Q = 2d
Necessário fazer redistribuição

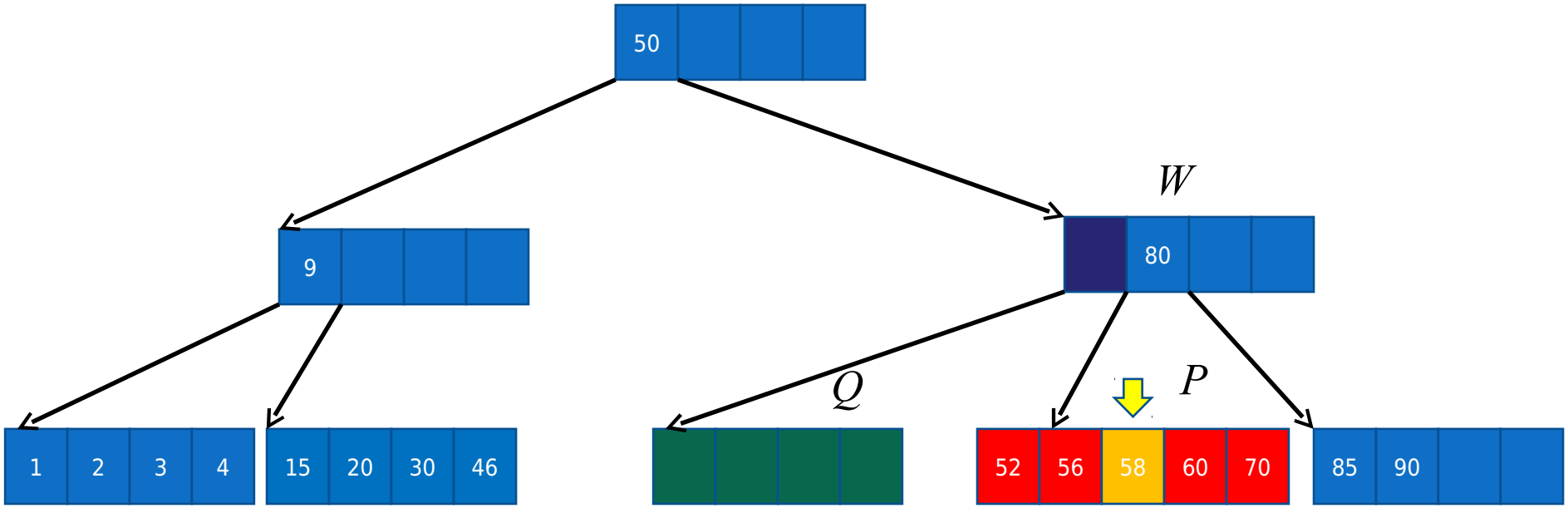
Árvore B – Remoção - Exemplo

- Remover chave 65



Árvore B – Remoção - Exemplo

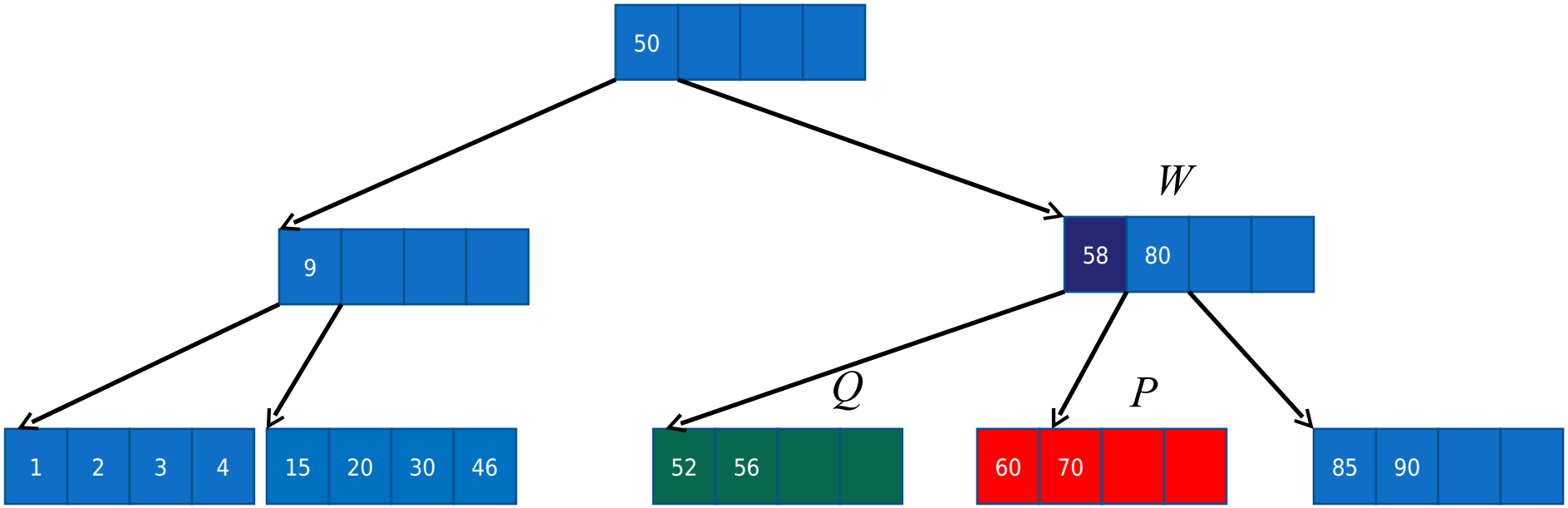
- Remover chave 65



Aplicar cisão em *P*, aproveitando *Q*

Árvore B – Remoção - Exemplo

- Remover chave 65



Aplicar cisão em P, aproveitando Q

Árvore B – Remoção - Algoritmo

- Passo 1: Aplicar o procedimento BuscaB, verificando a existência da chave *x* na árvore. Seja *P* o nó onde se encontra a chave.
- Passo 2:
  - Se *P* é uma folha, retirar a entrada correspondente à chave *x*.
  - Se não é, buscar a menor chave que se encontre em uma folha e que seja maior do que *x*. Seja *z* esta chave, e *F* o nó onde *z* se encontra.
    - Substitua a chave *x* por *z*. Fazer $P = F$.
- Passo 3:
  - Verificar se *P* contém *d* entradas. Em caso negativo, executar a operação de concatenação ou redistribuição.

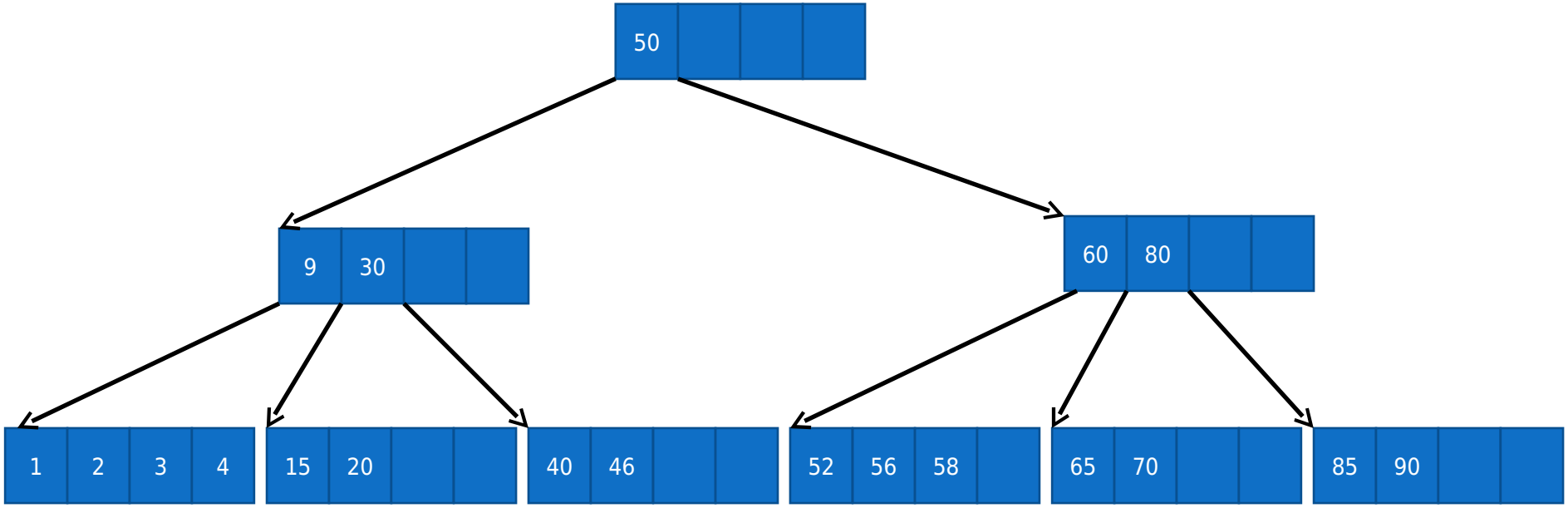
Exercício

- Insira os valores na árvore B:
  - 5, 1, 2, 3, 10, 7, 8, 9, 30, 13, 18, 19, 40, 46, 49, 80, 89

Árvore B+

Motivação

- O que acontece se solicitarmos que todas as chaves de uma árvore B sejam impressas em ordem ascendente?
- Cada leitura de um nó implica em um acesso à memória secundária!!!!

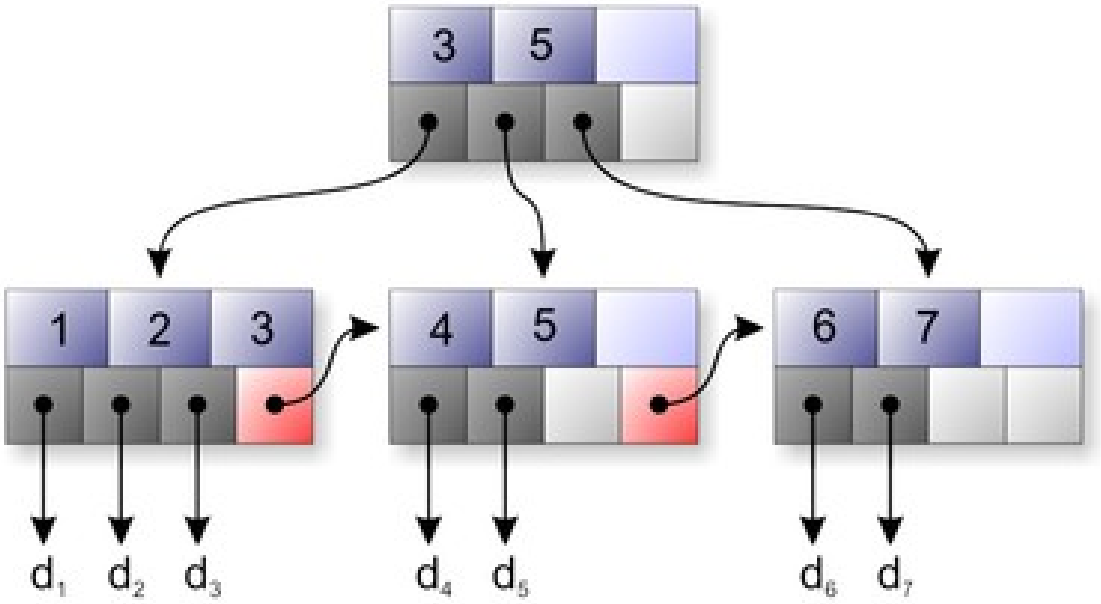


Árvore B+

- Árvore B
  - Referências aos dados são feitas a partir de quaisquer nós da árvore
- Árvore B+
  - Referências para os dados são feitas apenas a partir das folhas
  - Nós internos são indexados para acesso rápido dos dados
    - Conjunto de índices
  - Nós folha
    - São vinculados sequencialmente para formar um conjunto de seqüência
      - Varredura em ordem ascendente
  - Usado no sistema de arquivos FAT e NTFS do Windows, XFS, JFS2 e ext4 do Linux e em banco de dados relacionais como PostgreSQL e MySQL para índice de tabelas

Árvore B+ (Inserção)

- Exemplo

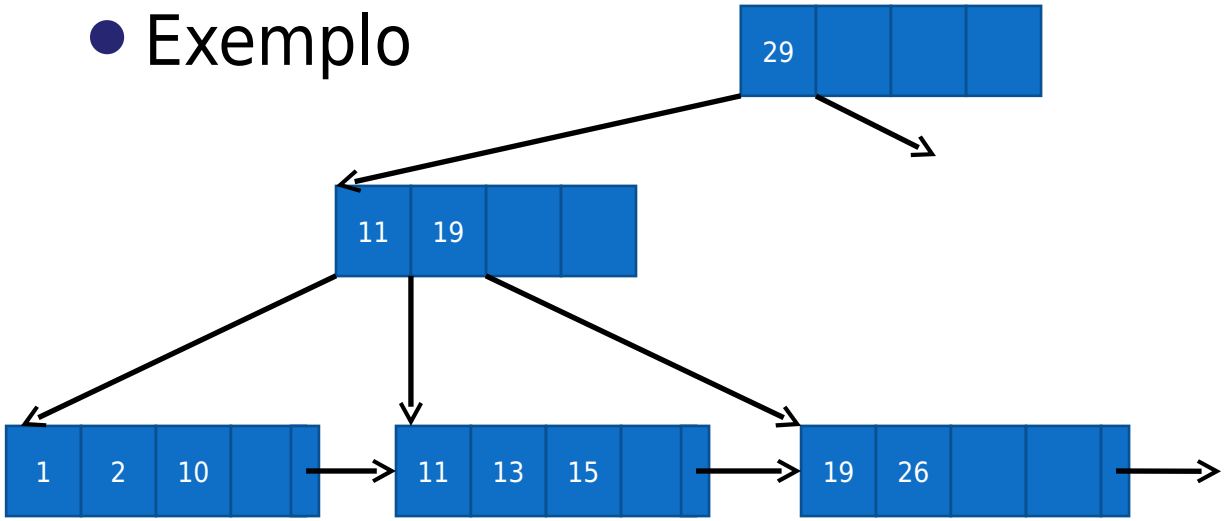


Árvore B+ (Inserção)

- Inserção (na folha)
  - Existe espaço no nó?
    - SIM
      - Inserir no nó folha e reordenar entradas
      - Nenhuma mudança é feita no conjunto do índice
    - NÃO
      - Dividir folha
      - Nova folha é incluída no conjunto de sequências
      - Distribuir chaves
      - Copiar primeira chave do nó novo para o ascendente (nó interno)
        - (importante: a chave é copiada e não movida)
      - Ascendente está cheio?
        - Não: Reorganizar chaves no nó
        - Sim: Dividir nó da mesma forma que em uma árvore B

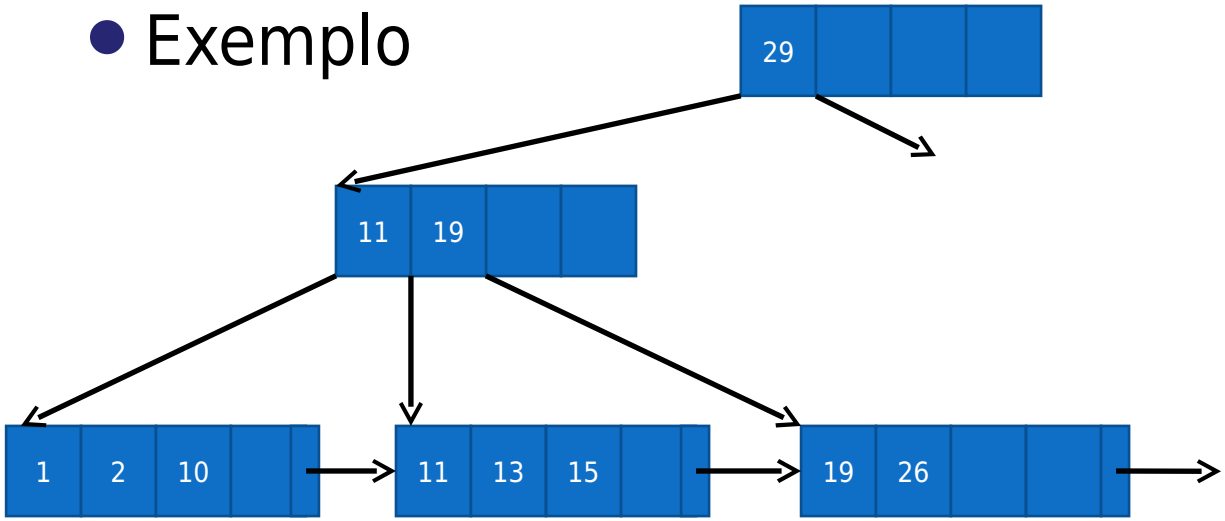
Árvore B+ (Inserção)

- Exemplo

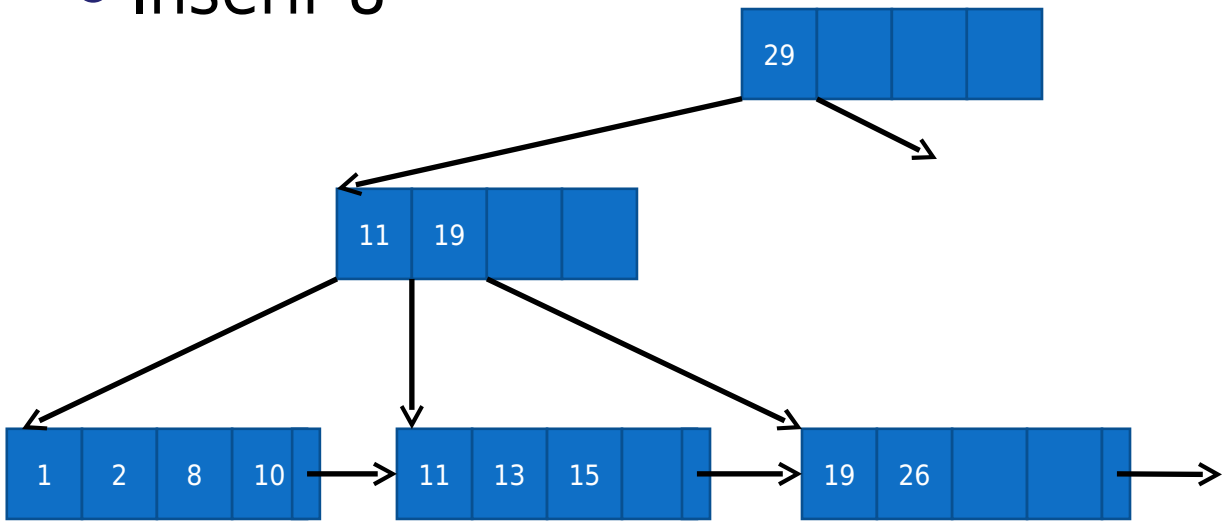


Árvore B+ (Inserção)

- Exemplo

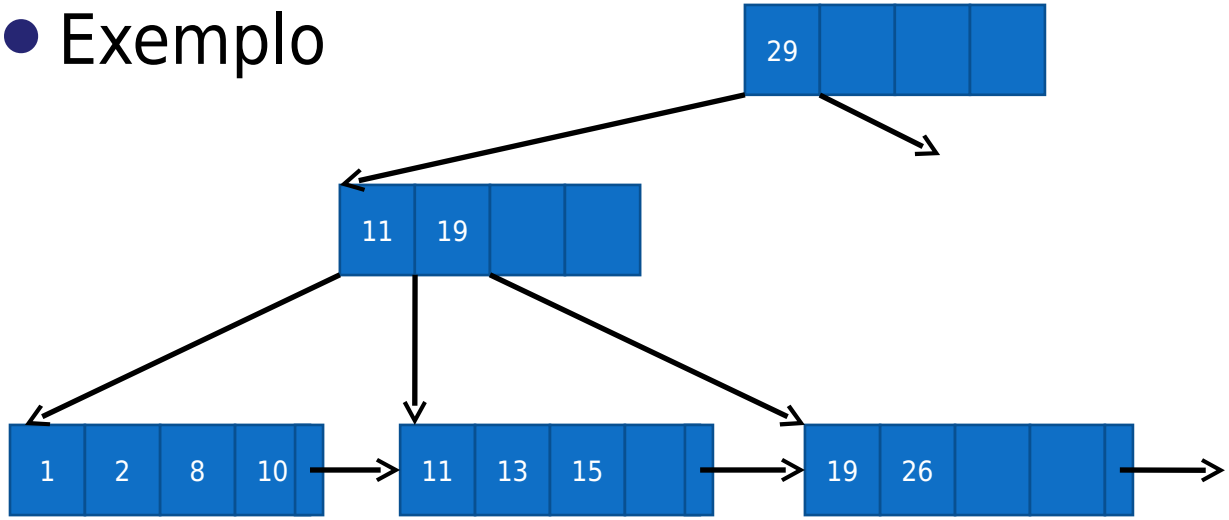


- Inserir 8

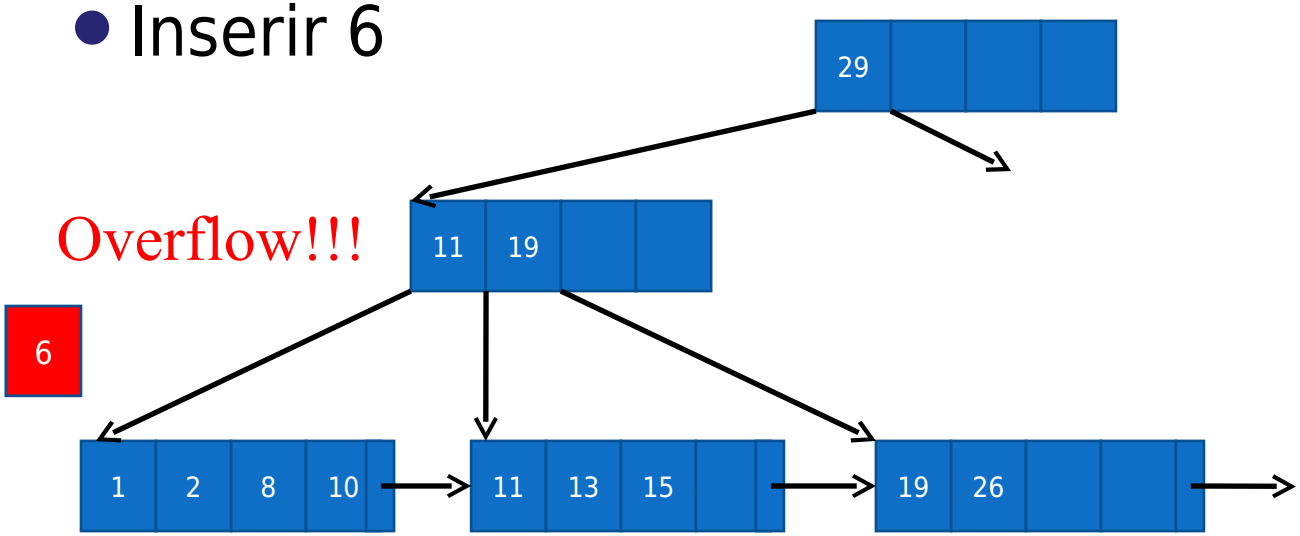


Árvore B+ (Inserção)

- Exemplo

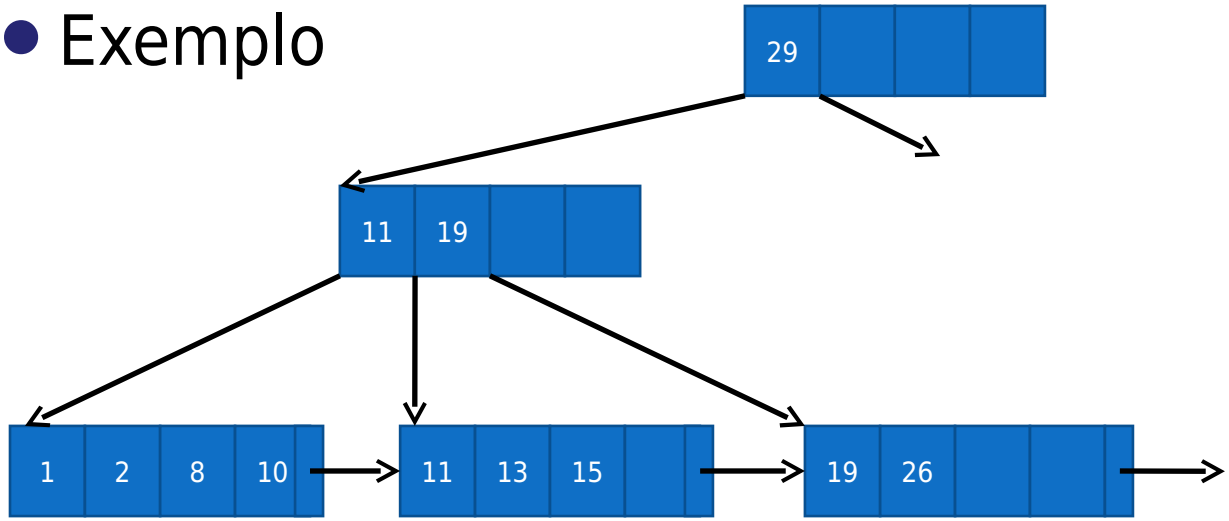


- Inserir 6

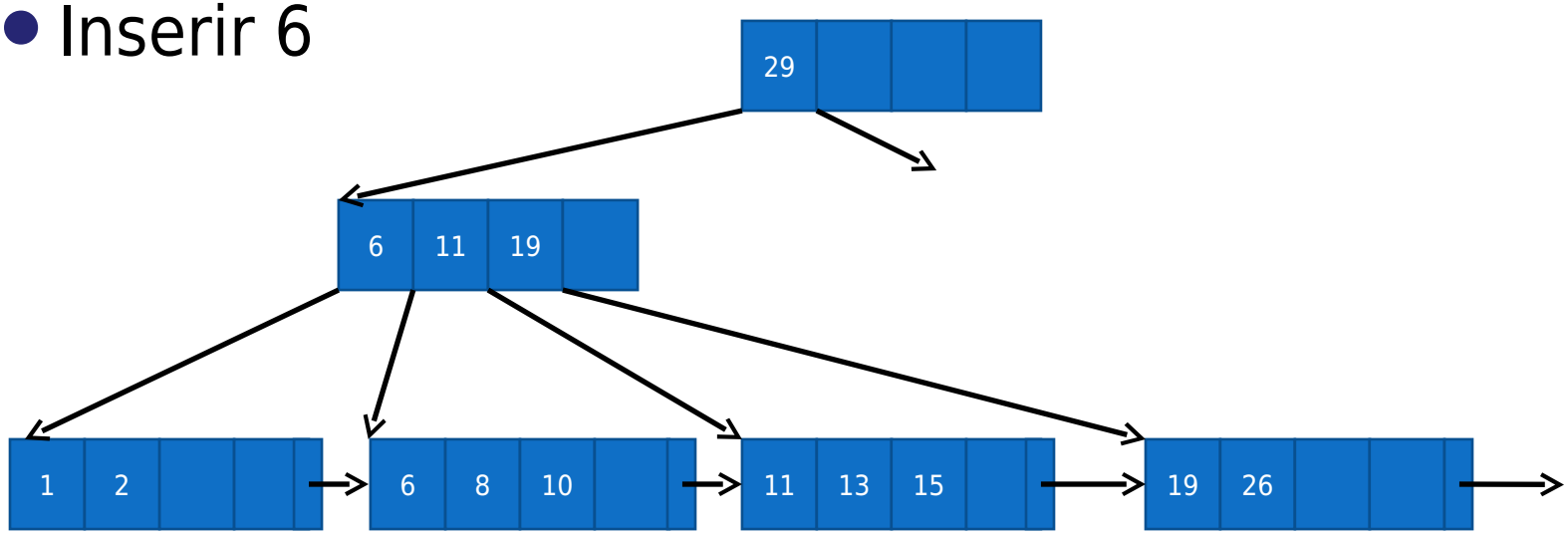


Árvore B+ (Inserção)

- Exemplo

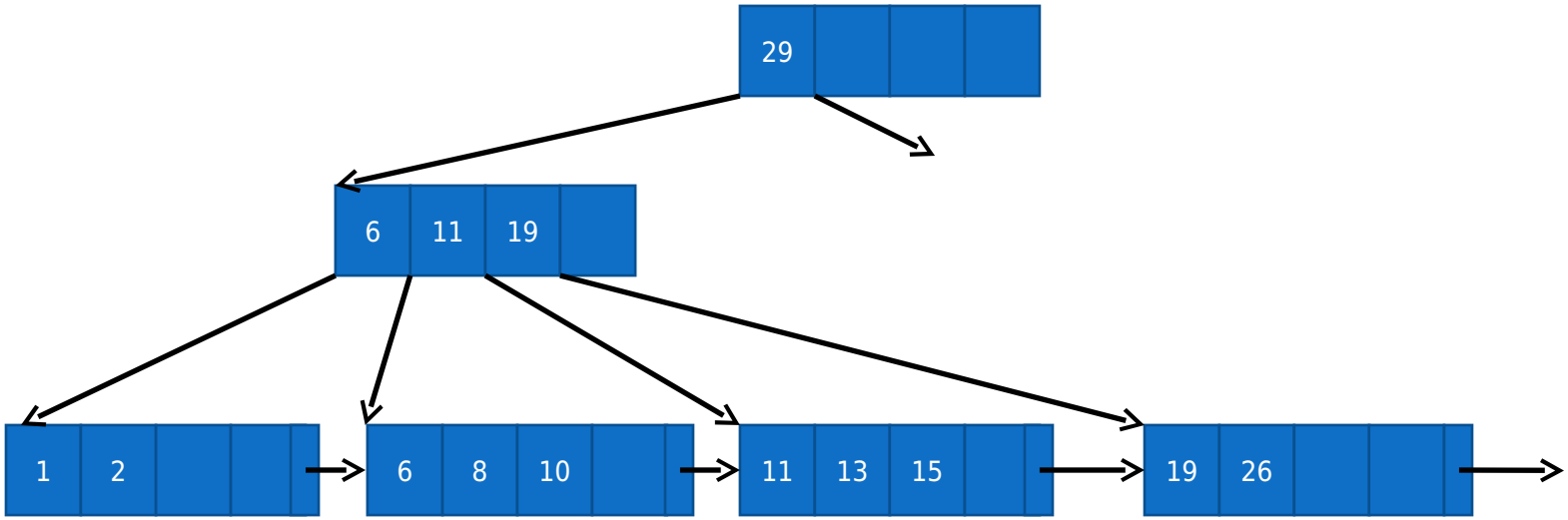


- Inserir 6



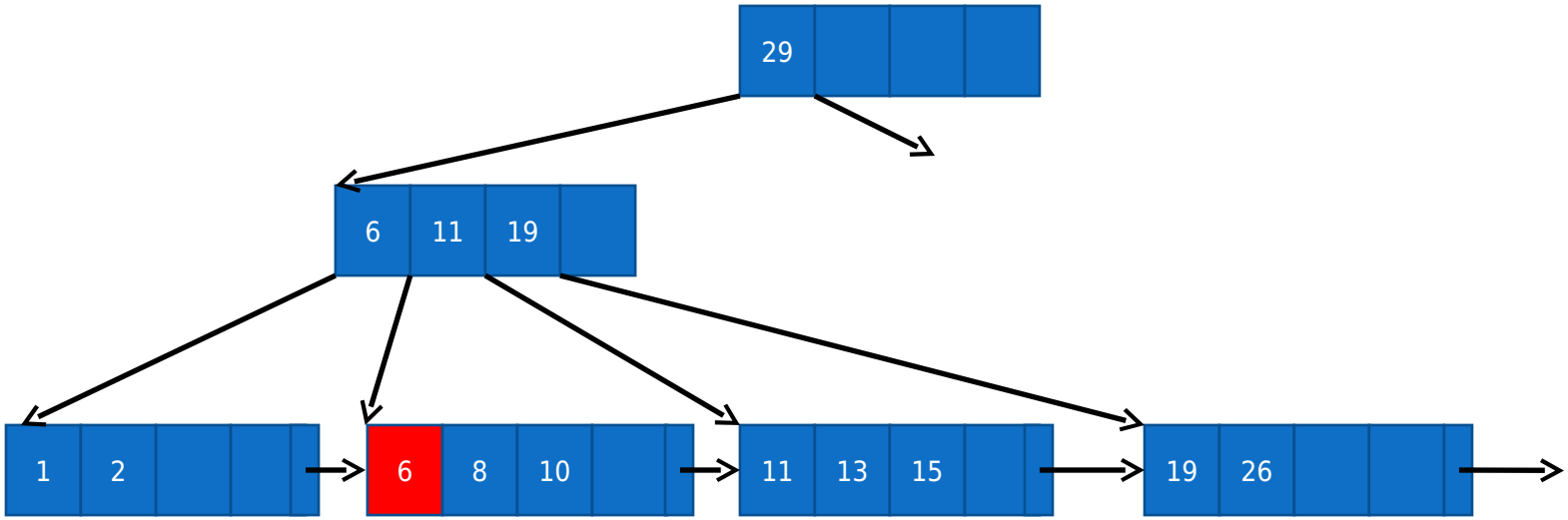
Árvore B+ (Remoção)

- Chave é removida do nó folha
  - Se **não há** *underflow*
    - Nenhuma mudança é feita no conjunto de índices
      - Mesmo se a chave também existir no conjunto de índices
        - Continua separando apropriadamente as chaves do índice
  - Exemplo: Remover 6



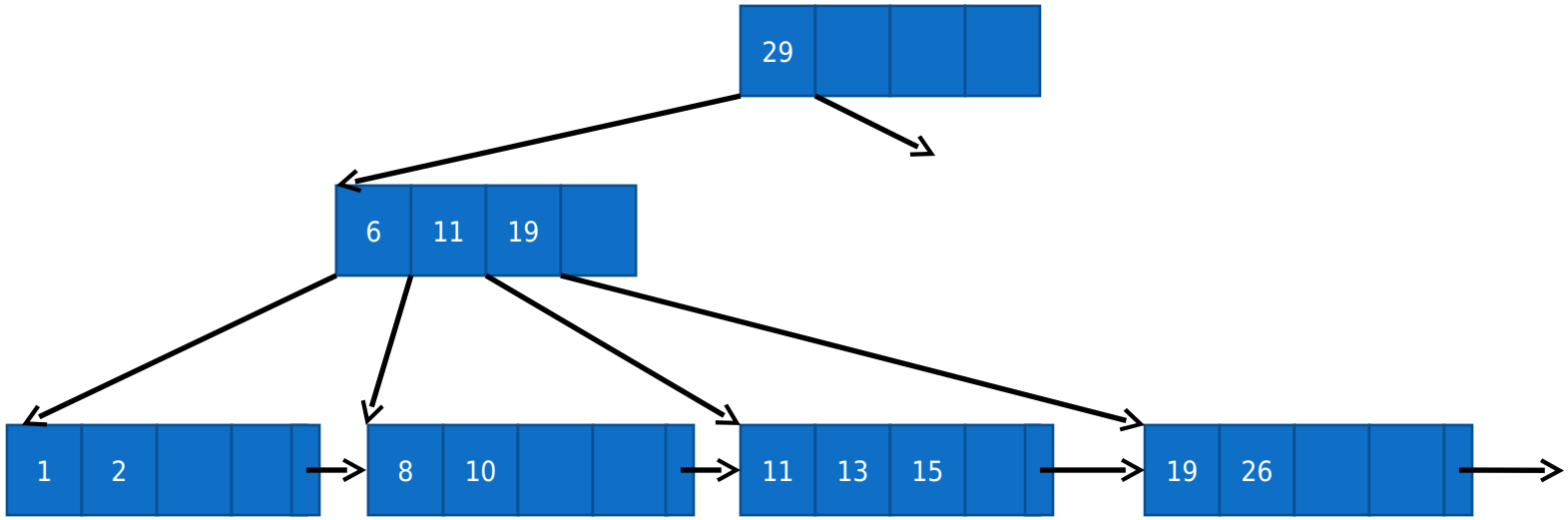
Árvore B+ (Remoção)

- Chave é removida do nó folha
  - Se **não há** *underflow*
    - Nenhuma mudança é feita no conjunto de índices
      - Mesmo se a chave também existir no conjunto de índices
        - Continua separando apropriadamente as chaves do índice
  - Exemplo: Remover 6



Árvore B+ (Remoção)

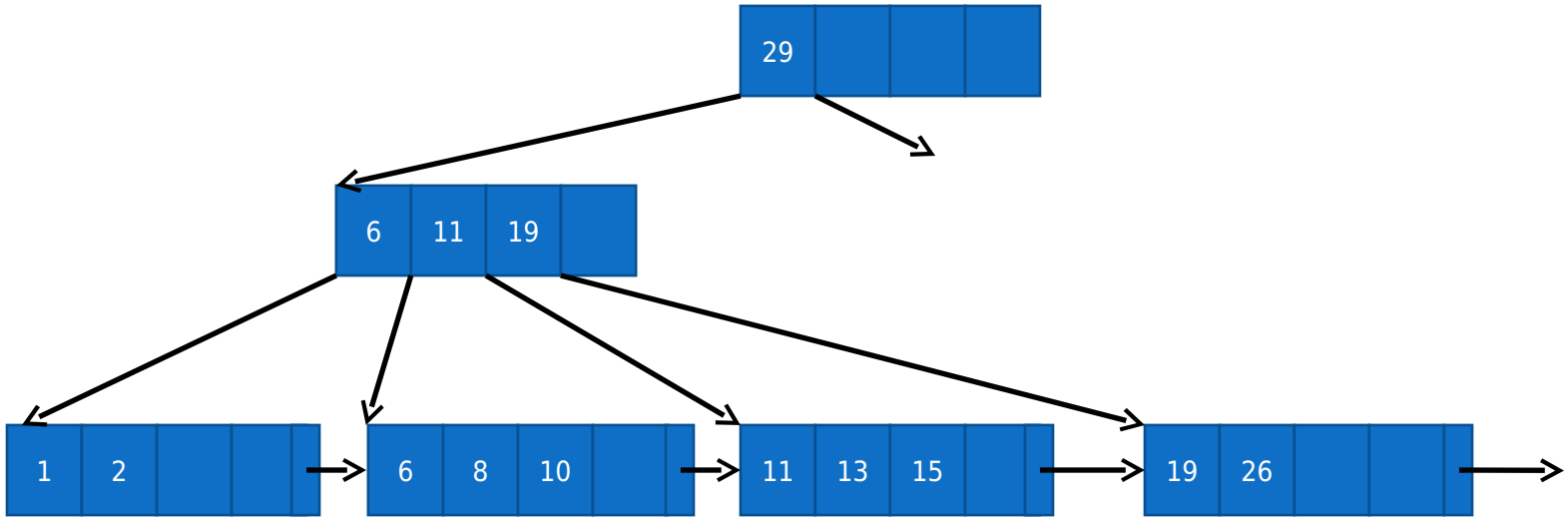
- Chave é removida do nó folha
  - Se **não há** *underflow*
    - Nenhuma mudança é feita no conjunto de índices
      - Mesmo se a chave também existir no conjunto de índices
        - Continua separando apropriadamente as chaves do índice
  - Exemplo: Remover 6



Árvore B+ (Remoção)

- Chave é removida do nó folha
  - Se **há** *underflow*
    - Chaves da folha e chaves de um irmão são distribuídas

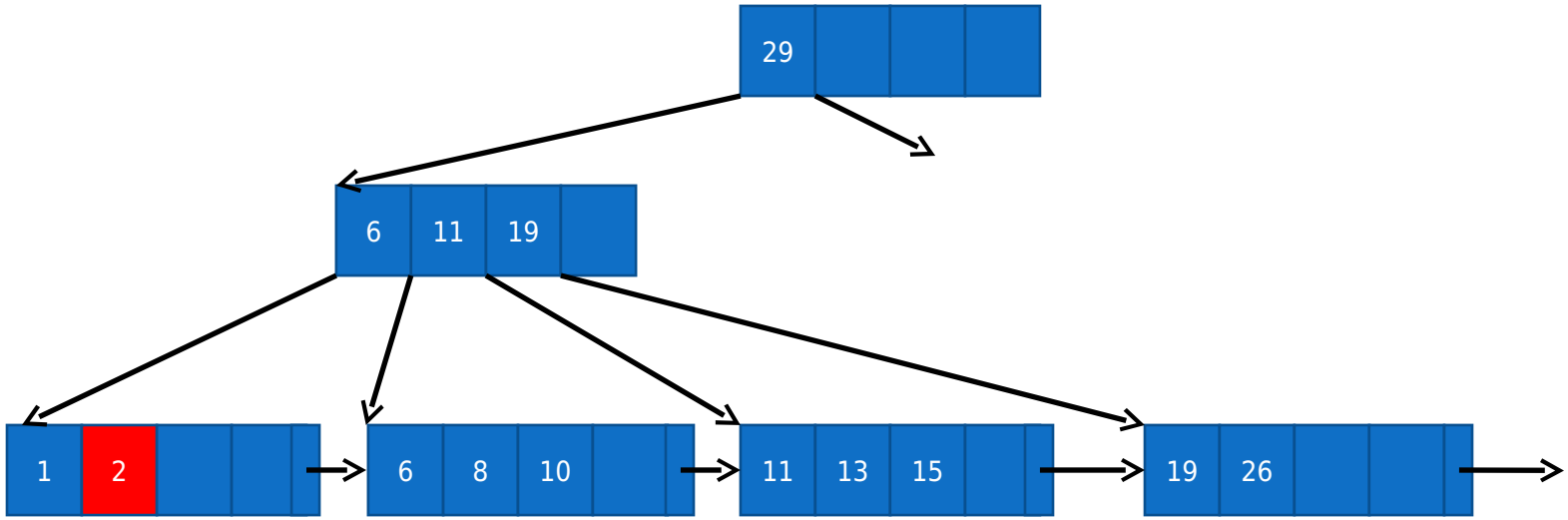
    OU
    - Folha é removida e chaves remanescentes são incluídas no irmão
  - Exemplo: Remover 2



Árvore B+ (Remoção)

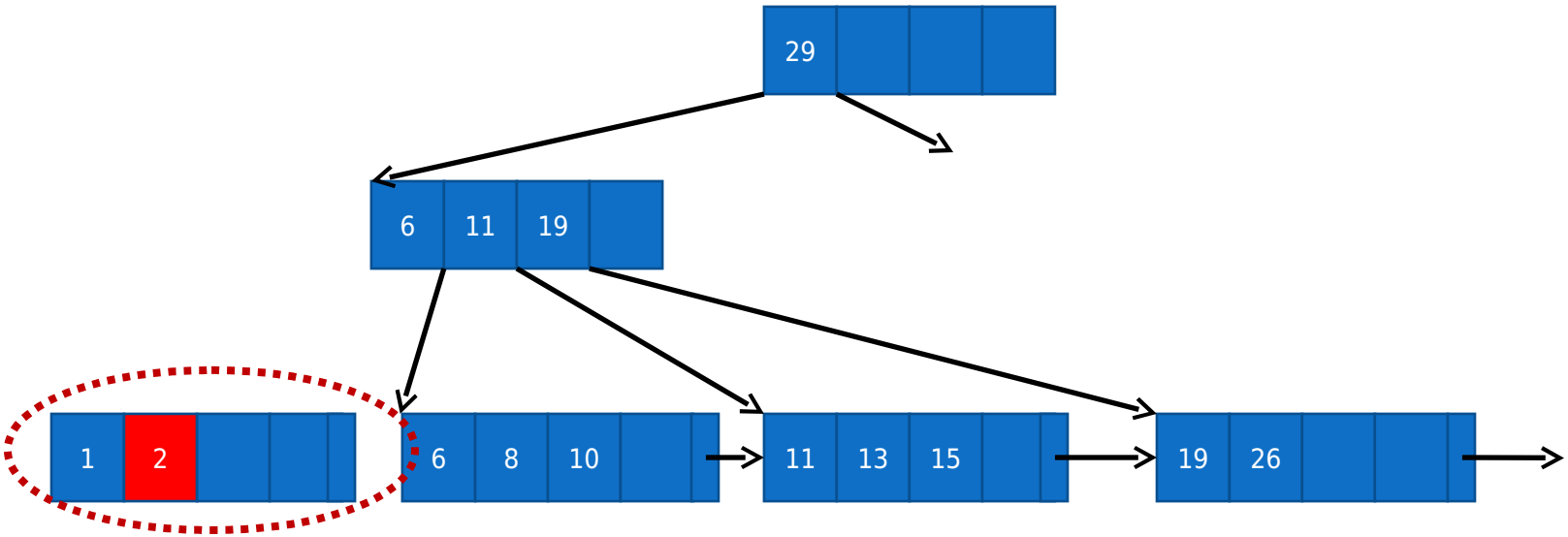
- Chave é removida do nó folha
  - Se **há** *underflow*
    - Chaves da folha e chaves de um irmão são distribuídas

    OU
    - Folha é removida e chaves remanescentes são incluídas no irmão
  - Exemplo: Remover 2



Árvore B+ (Remoção)

- Exemplo: Remover 2
  - Folha é removida e chaves remanescentes são incluídas no irmão



Árvore B+ (Remoção)

- Exemplo: Remover 2
  - Folha é removida e chaves remanescentes são incluídas no irmão
  - A primeira chave do irmão à direita do nó que permanece depois da fusão é copiada ao nó ascendente
  - Chaves no ascendente são colocadas em ordem
  - Separador no ascendente deve ser atualizado
  - *Underflow* pode ser propagado até a raiz