AED III
Árvores B

Ciência da Computação – IFSULDEMINAS

Primeiro Semestre de 2014

Roteiro

Árvores B

- Estrutura de dados projetada para funcionar especialmente em memória secundária (como disco magnético).
- Semelhante a árvore rubro negra, mas são melhores para minimizar operações de entrada e saída em disco.
- Muito empregada em aplicações que necessitam manipular grandes quantidades de informação tais como um banco de dados ou um sistema de arquivos.
- Diferente das árvores binárias, cada nó pode ter muitos filhos, isto é, o grau de um nó pode ser muito grande.

Árvores B

- Estrutura de dados projetada para funcionar especialmente em memória secundária (como disco magnético).
- Semelhante a árvore rubro negra, mas são melhores para minimizar operações de entrada e saída em disco.
- Muito empregada em aplicações que necessitam manipular grandes quantidades de informação tais como um banco de dados ou um sistema de arquivos.
- Diferente das árvores binárias, cada nó pode ter muitos filhos, isto é, o grau de um nó pode ser muito grande.

Árvores B

- Estrutura de dados projetada para funcionar especialmente em memória secundária (como disco magnético).
- Semelhante a árvore rubro negra, mas são melhores para minimizar operações de entrada e saída em disco.
- Muito empregada em aplicações que necessitam manipular grandes quantidades de informação tais como um banco de dados ou um sistema de arquivos.
- Diferente das árvores binárias, cada nó pode ter muitos filhos, isto é, o grau de um nó pode ser muito grande.

Árvores B

- Estrutura de dados projetada para funcionar especialmente em memória secundária (como disco magnético).
- Semelhante a árvore rubro negra, mas são melhores para minimizar operações de entrada e saída em disco.
- Muito empregada em aplicações que necessitam manipular grandes quantidades de informação tais como um banco de dados ou um sistema de arquivos.
- Diferente das árvores binárias, cada nó pode ter muitos filhos, isto é, o grau de um nó pode ser muito grande.

Árvores B

Nó

- n: número de chaves armazenadas em um nó.
- chaves: chaves do nó em ordem crescente.
- filhos ponteiros para os filhos.
- folha: valor booleano que indica se o nó é uma folha.

Árvores B

Uma árvore B de ordem mínima **t** é uma árvore de grau mínimo **t**:

Características

- Em um nó as chaves estão ordenadas.
- Todas as folhas estão no mesmo nível.
- Número de filhos:
  - De **t** a **2 * t** filhos.
  - Raiz: de **2** a **m** filhos.
- Número de chaves:
  - De **t - 1** a **2 + t - 1** chaves.
  - Raiz: de **1** a **m - 1** chaves.

Árvores B

Uma árvore B de ordem mínima **t** é uma árvore de grau mínimo **t**:

Características

- Em um nó as chaves estão ordenadas.
- Todas as folhas estão no mesmo nível.
- Número de filhos:
  - De **t** a **2 * t** filhos.
  - Raiz: de **2** a **m** filhos.
- Número de chaves:
  - De **t - 1** a **2 + t - 1** chaves.
  - Raiz: de **1** a **m - 1** chaves.

Árvores B

Uma árvore B de ordem mínima **t** é uma árvore de grau mínimo **t**:

Características

- Em um nó as chaves estão ordenadas.
- Todas as folhas estão no mesmo nível.
- Número de filhos:
  - De **t** a **2 * t** filhos.
  - Raiz: de **2** a **m** filhos.
- Número de chaves:
  - De **t - 1** a **2 + t - 1** chaves.
  - Raiz: de **1** a **m - 1** chaves.

Árvores B

Uma árvore B de ordem mínima **t** é uma árvore de grau mínimo **t**:

Características

- Em um nó as chaves estão ordenadas.
- Todas as folhas estão no mesmo nível.
- Número de filhos:
  - De **t** a **2 * t** filhos.
  - Raiz: de **2** a **m** filhos.
- Número de chaves:
  - De **t - 1** a **2 * t - 1** chaves.
  - Raiz: de **1** a **m - 1** chaves.

Árvores B

Uma árvore B de ordem mínima **t** é uma árvore de grau mínimo **t**:

Características

- Em um nó as chaves estão ordenadas.
- Todas as folhas estão no mesmo nível.
- Número de filhos:
  - De **t** a **2 * t** filhos.
  - Raiz: de **2** a **m** filhos.
- Número de chaves:
  - De **t - 1** a **2 + t - 1** chaves.
  - Raiz: de **1** a **m - 1** chaves.

Inserindo em uma árvore B

Inserção

- Tente inserir a nova chave em um folha.
- Se o número de chaves do nó exceder o número de chaves permitidas, divida a folha em duas, promovendo a chave central para o pai da folha.
- Se o número de chaves do nó pai exceder o número de chaves permitidas, divida o nó pai em dois, promovendo a chave central para o ancestral do pai.
- Esta estratégia pode ser repetida por todo caminho até o topo.
- Se necessário, a raiz é dividida em dois nós, tornando a chave central em raiz. Se isto acontecer o nível da árvore aumenta.

Inserindo em uma árvore B

Inserção

- Tente inserir a nova chave em um folha.
- Se o número de chaves do nó exceder o número de chaves permitidas, divida a folha em duas, promovendo a chave central para o pai da folha.
- Se o número de chaves do nó pai exceder o número de chaves permitidas, divida o nó pai em dois, promovendo a chave central para o ancestral do pai.
- Esta estratégia pode ser repetida por todo caminho até o topo.
- Se necessário, a raiz é dividida em dois nós, tornando a chave central em raiz. Se isto acontecer o nível da árvore aumenta.

Inserindo em uma árvore B

Inserção

- Tente inserir a nova chave em um folha.
- Se o número de chaves do nó exceder o número de chaves permitidas, divida a folha em duas, promovendo a chave central para o pai da folha.
- Se o número de chaves do nó pai exceder o número de chaves permitidas, divida o nó pai em dois, promovendo a chave central para o ancestral do pai.
- Esta estratégia pode ser repetida por todo caminho até o topo.
- Se necessário, a raiz é dividida em dois nós, tornando a chave central em raiz. Se isto acontecer o nível da árvore aumenta.

Inserindo em uma árvore B

Inserção

- Tente inserir a nova chave em um folha.
- Se o número de chaves do nó exceder o número de chaves permitidas, divida a folha em duas, promovendo a chave central para o pai da folha.
- Se o número de chaves do nó pai exceder o número de chaves permitidas, divida o nó pai em dois, promovendo a chave central para o ancestral do pai.
- Esta estratégia pode ser repetida por todo caminho até o topo.
- Se necessário, a raiz é dividida em dois nós, tornando a chave central em raiz. Se isto acontecer o nível da árvore aumenta.

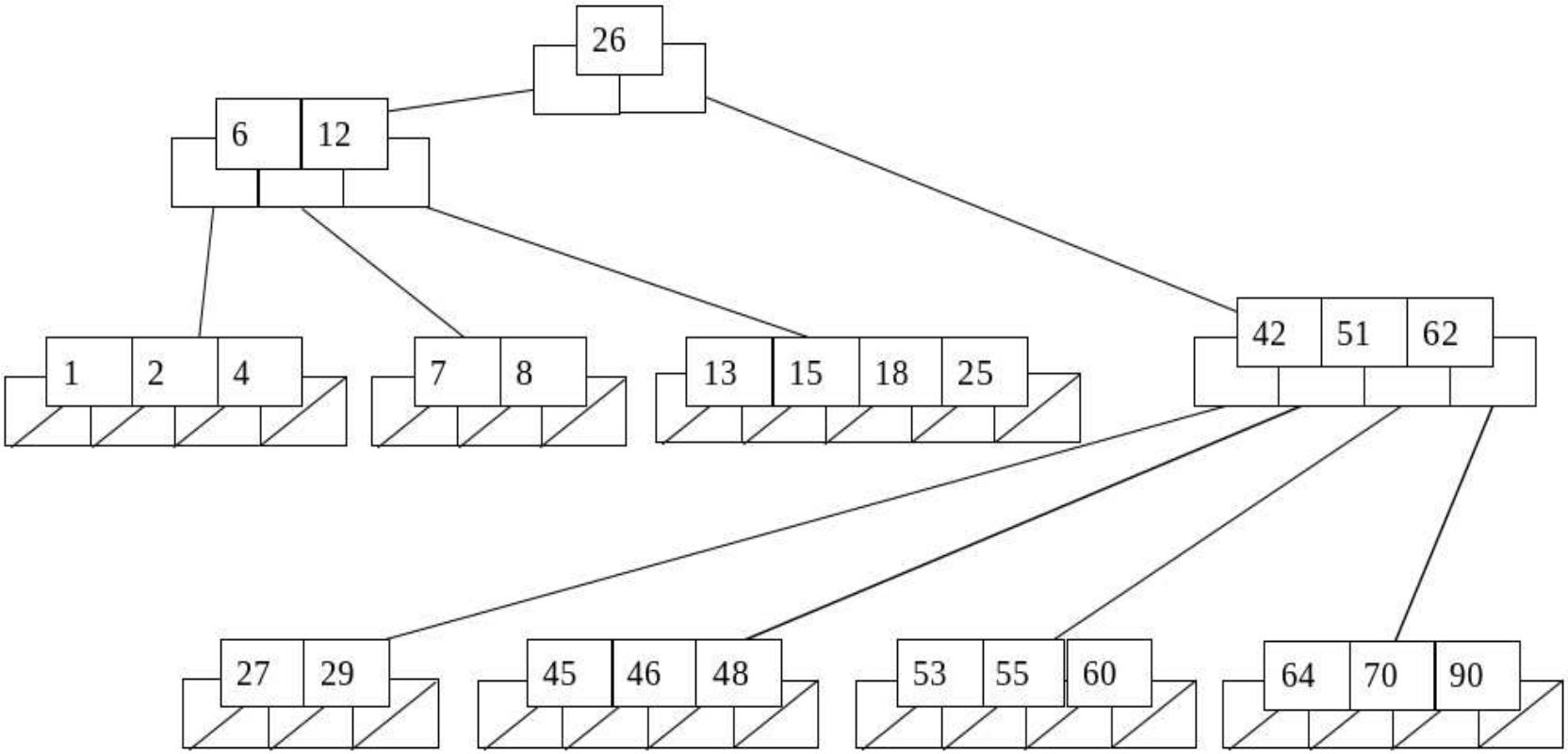
Inserindo em uma árvore B

Inserção

- Tente inserir a nova chave em um folha.
- Se o número de chaves do nó exceder o número de chaves permitidas, divida a folha em duas, promovendo a chave central para o pai da folha.
- Se o número de chaves do nó pai exceder o número de chaves permitidas, divida o nó pai em dois, promovendo a chave central para o ancestral do pai.
- Esta estratégia pode ser repetida por todo caminho até o topo.
- Se necessário, a raiz é dividida em dois nós, tornando a chave central em raiz. Se isto acontecer o nível da árvore aumenta.

Árvores B

Árvore B de grau mínimo 3:



Construindo uma árvores B

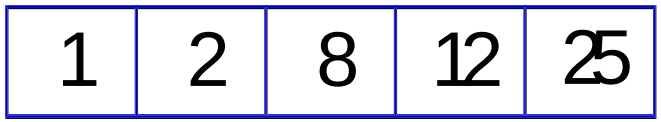
- Suponha que nós começamos com uma árvore B vazia e as chaves chegam na seguinte ordem: **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Nós queremos construir uma árvore B de garu mínimo 3.
- Os cinco primeiras chaves vão para a raiz:
- Colocar a sexta chave na raiz viola o número máximo de chaves por nó.
- Assim, quando 6 chega, pegamos a chave do meio para fazer a nova raiz.

Construindo uma árvores B

- Suponha que nós começamos com uma árvore B vazia e as chaves chegam na seguinte ordem: **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Nós queremos construir uma árvore B de garu mínimo 3.
- Os cinco primeiras chaves vão para a raiz:
- Colocar a sexta chave na raiz viola o número máximo de chaves por nó.
- Assim, quando 6 chega, pegamos a chave do meio para fazer a nova raiz.

Construindo uma árvores B

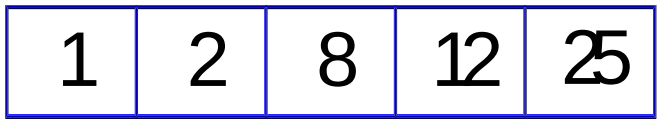
- Suponha que nós começamos com uma árvore B vazia e as chaves chegam na seguinte ordem: **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Nós queremos construir uma árvore B de garu mínimo 3.
- Os cinco primeiras chaves vão para a raiz:



- Colocar a sexta chave na raiz viola o número máximo de chaves por nó.
- Assim, quando 6 chega, pegamos a chave do meio para fazer a nova raiz.

Construindo uma árvores B

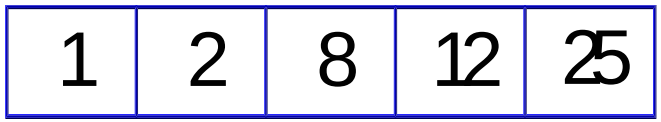
- Suponha que nós começamos com uma árvore B vazia e as chaves chegam na seguinte ordem: **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Nós queremos construir uma árvore B de garu mínimo 3.
- Os cinco primeiras chaves vão para a raiz:



- Colocar a sexta chave na raiz viola o número máximo de chaves por nó.
- Assim, quando 6 chega, pegamos a chave do meio para fazer a nova raiz.

Construindo uma árvores B

- Suponha que nós começamos com uma árvore B vazia e as chaves chegam na seguinte ordem: **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Nós queremos construir uma árvore B de garu mínimo 3.
- Os cinco primeiras chaves vão para a raiz:



- Colocar a sexta chave na raiz viola o número máximo de chaves por nó.
- Assim, quando 6 chega, pegamos a chave do meio para fazer a nova raiz.

Construindo uma árvores B

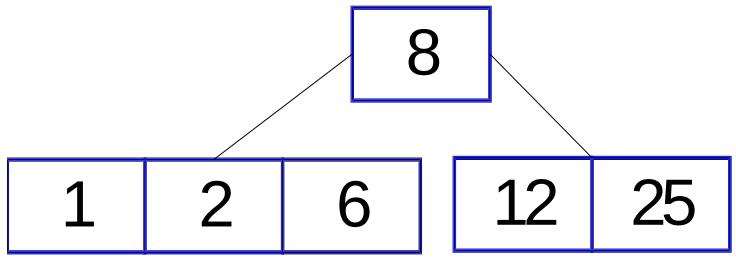
- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56.**
- Adicionando 6 à árvore.
- A insersão de 6 excede a ordem da árvore, promovemos a chave mediana.

Construindo uma árvores B

- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando 6 à árvore.
- A inserção de 6 excede a ordem da árvore, promovemos a chave mediana.

Construindo uma árvores B

- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando 6 à árvore.
- A insersão de 6 excede a ordem da árvore, promovemos a chave mediana.



Construindo uma árvores B

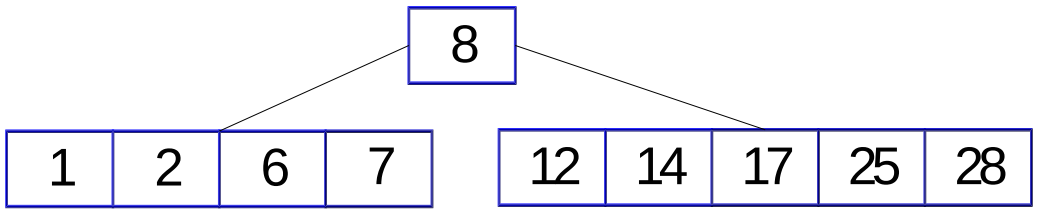
- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56.**
- Adicionando 14, 28, 17 e 7 à árvore.
- Adicionando a chave 52 à folha direita, teríamos excesso de chaves. Promovemos o nó mediano.

Construindo uma árvores B

- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando 14, 28, 17 e 7 à árvore.
- Adicionando a chave 52 à folha direita, teríamos excesso de chaves. Promovemos o nó mediano.

Construindo uma árvores B

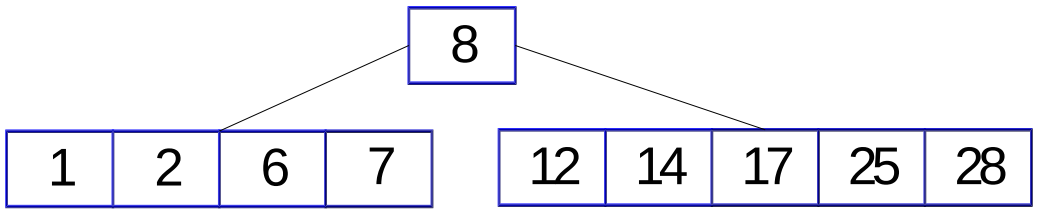
- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando 14, 28, 17 e 7 à árvore.



- Adicionando a chave 52 à folha direita, teríamos excesso de chaves. Promovemos o nó mediano.

Construindo uma árvores B

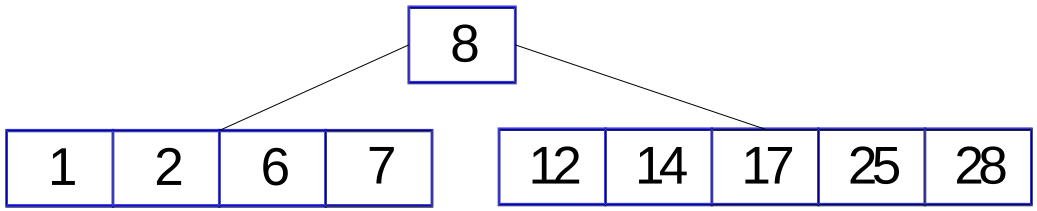
- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando 14, 28, 17 e 7 à árvore.



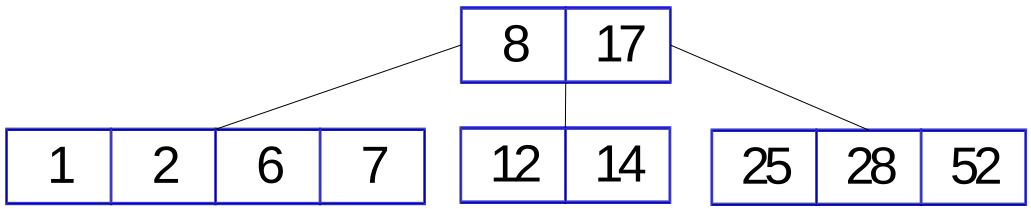
- Adicionando a chave 52 à folha direita, teríamos excesso de chaves. Promovemos o nó mediano.

Construindo uma árvores B

- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando 14, 28, 17 e 7 à árvore.



- Adicionando a chave 52 à folha direita, teríamos excesso de chaves. Promovemos o nó mediano.



Construindo uma árvores B

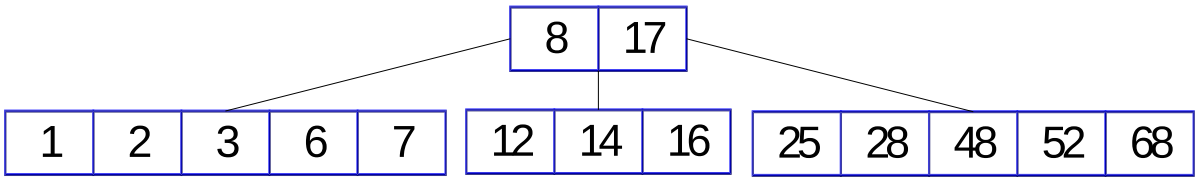
- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56.**
- Adicionando 16, 48, 68 e 3 à árvore.
- Adicionando a chave 26 à folha esquerda, teríamos excesso de chaves. Promovemos o nó mediano.

Construindo uma árvores B

- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando 16, 48, 68 e 3 à árvore.
- Adicionando a chave 26 à folha esquerda, teríamos excesso de chaves. Promovemos o nó mediano.

Construindo uma árvores B

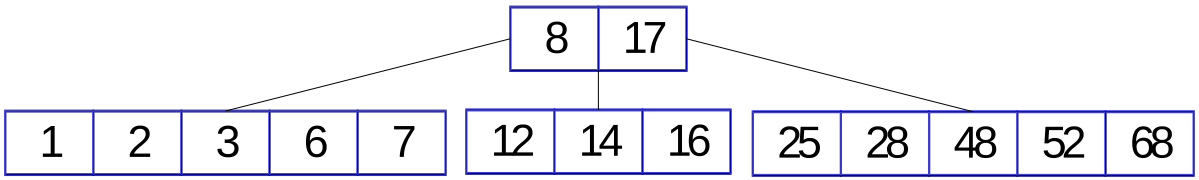
- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando 16, 48, 68 e 3 à árvore.



- Adicionando a chave 26 à folha esquerda, teríamos excesso de chaves. Promovemos o nó mediano.

Construindo uma árvores B

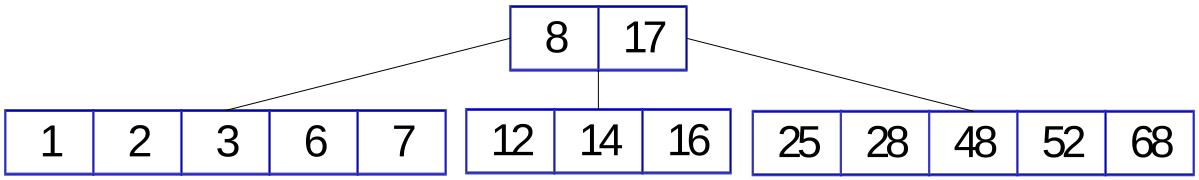
- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando 16, 48, 68 e 3 à árvore.



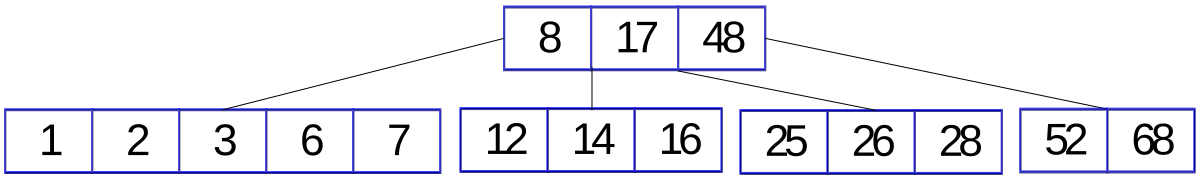
- Adicionando a chave 26 à folha esquerda, teríamos excesso de chaves. Promovemos o nó mediano.

Construindo uma árvores B

- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando 16, 48, 68 e 3 à árvore.



- Adicionando a chave 26 à folha esquerda, teríamos excesso de chaves. Promovemos o nó mediano.



Construindo uma árvores B

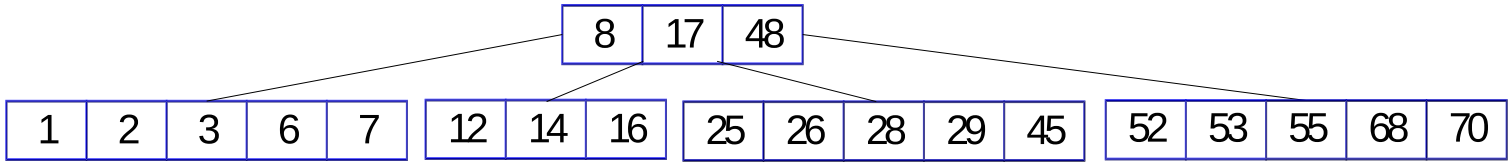
- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando 26, 29, 53, 55, 45 e 70 à árvore.
- Adicionando a chave 24 à folha, teríamos excesso de chaves. Promovemos o nó mediano.

Construindo uma árvores B

- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando 26, 29, 53, 55, 45 e 70 à árvore.
- Adicionando a chave 24 à folha, teríamos excesso de chaves. Promovemos o nó mediano.

Construindo uma árvores B

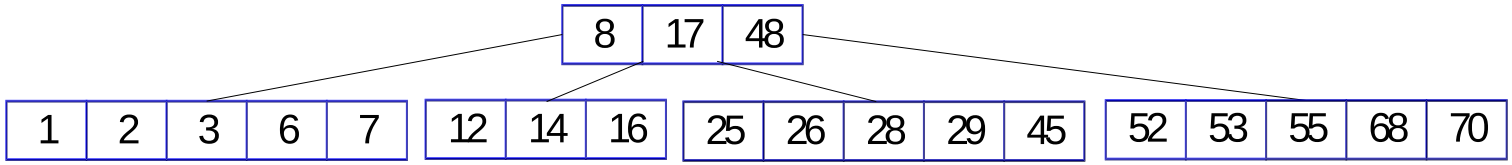
- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando 26, 29, 53, 55, 45 e 70 à árvore.



- Adicionando a chave 24 à folha, teríamos excesso de chaves. Promovemos o nó mediano.

Construindo uma árvores B

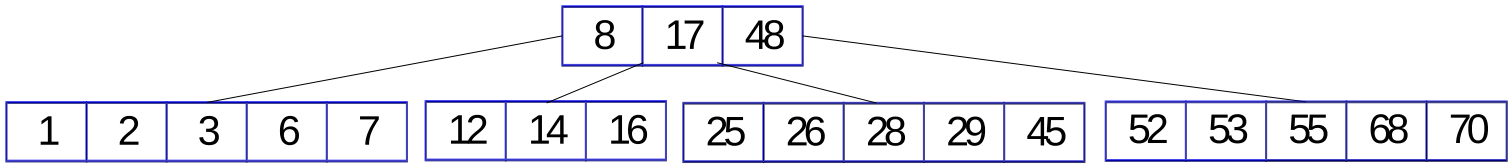
- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando 26, 29, 53, 55, 45 e 70 à árvore.



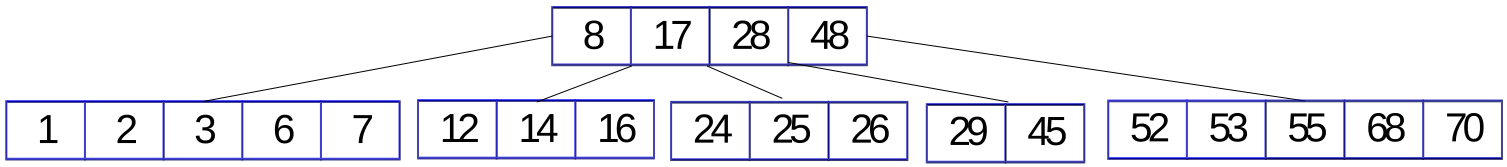
- Adicionando a chave 24 à folha, teríamos excesso de chaves. Promovemos o nó mediano.

Construindo uma árvores B

- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando 26, 29, 53, 55, 45 e 70 à árvore.



- Adicionando a chave 24 à folha, teríamos excesso de chaves. Promovemos o nó mediano.



Construindo uma árvores B

- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando a chave 5 à folha, teríamos excesso de chaves. Promovemos o nó mediano.
- Adicionando a chave 56 à folha, teríamos excesso de chaves. Promovemos o nó mediano.

A inserção fez com que a altura da árvore aumentasse.

Construindo uma árvores B

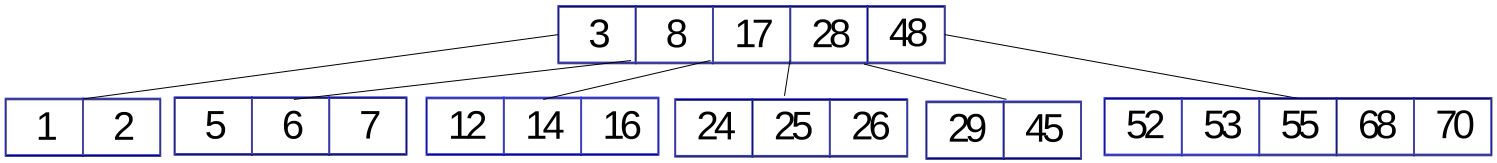
- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando a chave 5 à folha, teríamos excesso de chaves. Promovemos o nó mediano.

- Adicionando a chave 56 à folha, teríamos excesso de chaves. Promovemos o nó mediano.

A inserção fez com que a altura da árvore aumentasse.

Construindo uma árvores B

- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando a chave 5 à folha, teríamos excesso de chaves. Promovemos o nó mediano.

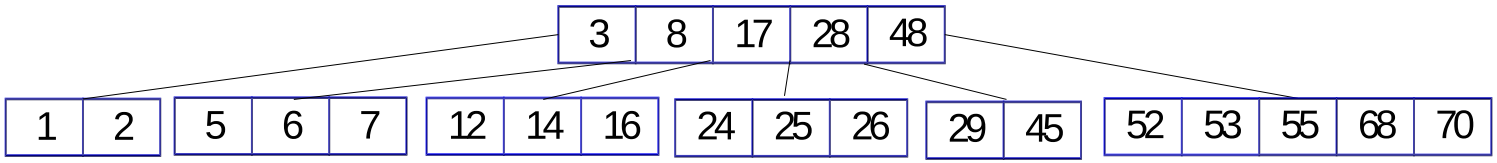


- Adicionando a chave 56 à folha, teríamos excesso de chaves. Promovemos o nó mediano.

A inserção fez com que a altura da árvore aumentasse.

Construindo uma árvores B

- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando a chave 5 à folha, teríamos excesso de chaves. Promovemos o nó mediano.

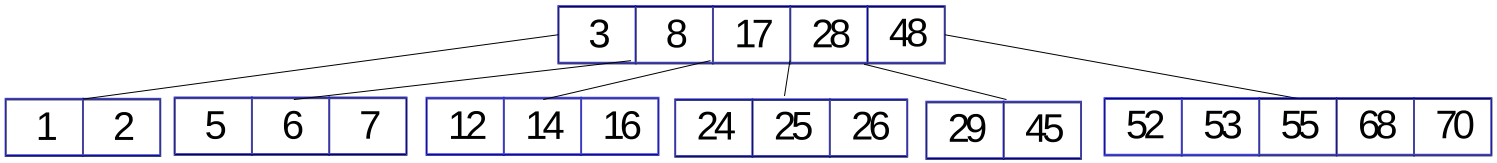


- Adicionando a chave 56 à folha, teríamos excesso de chaves. Promovemos o nó mediano.

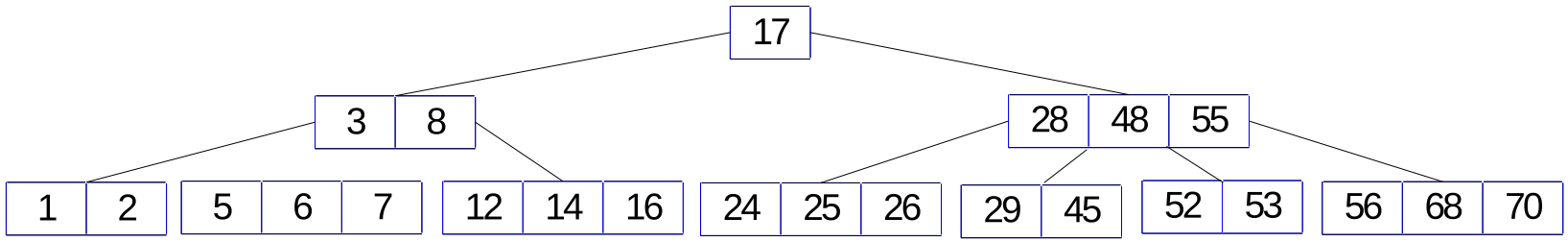
A inserção fez com que a altura da árvore aumentasse.

Construindo uma árvores B

- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando a chave 5 à folha, teríamos excesso de chaves. Promovemos o nó mediano.



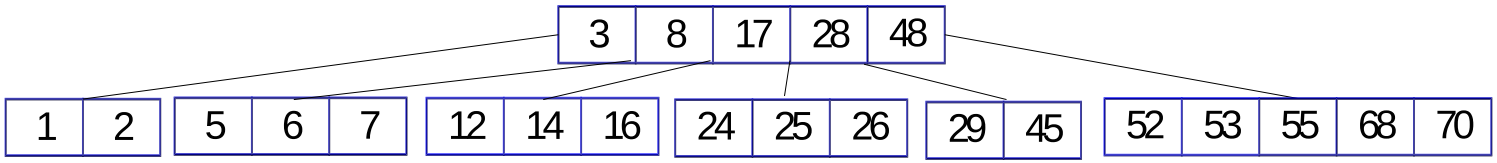
- Adicionando a chave 56 à folha, teríamos excesso de chaves. Promovemos o nó mediano.



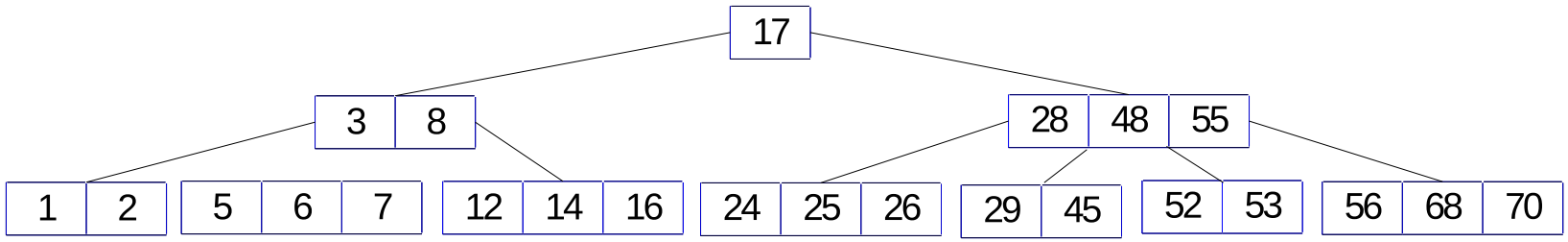
A inserção fez com que a altura da árvore aumentasse.

Construindo uma árvores B

- **1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45, 70, 24, 5 e 56**.
- Adicionando a chave 5 à folha, teríamos excesso de chaves. Promovemos o nó mediano.



- Adicionando a chave 56 à folha, teríamos excesso de chaves. Promovemos o nó mediano.



A inserção fez com que a altura da árvore aumentasse.

Aplicação

1. Insira as seguintes chaves em uma árvore B de grau mínimo 3: 3, 7, 9, 23, 45, 1, 5, 14, 25, 24, 13, 11, 8, 19, 4, 31, 35, 56, 15, 60, 16, 20 e 22.

Split

- Função responsável por dividir um nó cheio.
- A função recebe como parâmetros *x* e *i*:
  - x: nó interno não cheio.
  - i: índice de um filho de x, tal que x.filhos[i] é um nó cheio.
- A função reparte o filho de x em dois e ajusta x de forma que ele tenha uma chave adicional.

Split

- Função responsável por dividir um nó cheio.
- A função recebe como parâmetros *x* e *i*:
  - x: nó interno não cheio.
  - i: índice de um filho de x, tal que x.filhos[i] é um nó cheio.
- A função reparte o filho de x em dois e ajusta x de forma que ele tenha uma chave adicional.

Split

- Função responsável por dividir um nó cheio.
- A função recebe como parâmetros *x* e *i*:
  - x: nó interno não cheio.
  - i: índice de um filho de x, tal que x.filhos[i] é um nó cheio.
- A função reparte o filho de x em dois e ajusta x de forma que ele tenha uma chave adicional.

Split

- Função responsável por dividir um nó cheio.
- A função recebe como parâmetros *x* e *i*:
  - x: nó interno não cheio.
  - i: índice de um filho de x, tal que x.filhos[i] é um nó cheio.
- A função reparte o filho de x em dois e ajusta x de forma que ele tenha uma chave adicional.

Split

```
B-TREE-SPLIT(x, i) {
   z = ALLOCATE-NODE();

   y = x.filhos[i];                          // filho de x

   z.folha = y.folha;                        // z folha, se y folha

   z.n = t - 1;                              // z tem o mínimo de chaves

   for j = 1 to t - 1                        // copia chaves de y para z
      z.chaves[j] = y.chaves[j + t];

   if not y.folha
      for j = 1 to t                         // copia filhos de y para z
         z.filhos[j] = y.filhos[j + t];

   y.n = t - 1;                              // y tem o mínimo de chaves
```

Split

```
   for j = x.n + 1 downto i + 1        // desloca filhos de x
      x.filhos[j + 1] = x.filhos[j];

   x.filhos[i + 1] = z;                // faz x apontar para z

   for j = x.n downto i                // desloca chaves de x
      x.chaves[j + 1] = x.chaves[j];

   x.chaves[i] = y.chaves[t];          // inclui nova chave em x

   x.n = x.n + 1;

   DISK-WRITE(y);
   DISK-WRITE(z);
   DISK-WRITE(x);
}
```

Inserção

- A função de inserção recebe a raiz da árvore, T, e a chave a ser inserida, k.

```
B-TREE-INSERT(T, k) {
   r = T;

   if r.n == 2t -1                    // raiz cheia
      s = ALLOCATE-NODE();
      T = s;
      s.folha = false;
      s.n = 0;
      s.filhos[1] = r;
      B-TREE-SPLIT(s, 1);
      B-TREE-INSERT-NONFULL(s, k);
   else
      B-TREE-INSERT-NONFULL(r, k);
}
```

Inserção

- A função auxiliar de inserção recebe um nó x e a chave k a ser inserida.
- A inserção é feita nas folhas. Assim, a função percorre a árvore (de forma recursiva) até encontrar uma folha para a inserção.

```
B-TREE-INSERT-NONFULL(x, k) {
  i = x.n;

  if x.folha

    // desloca chaves do nó para abrir espaço
    while i >= 1 e k < x.chaves[i]
      x.chaves[i + 1] = x.chaves[i];
      i = i - 1;

    // insere a nova chave
    x.chaves[i + 1] = k;
    x.n = x.n + 1;
    DISK-WRITE(x);
```

Inserção

- A função auxiliar de inserção recebe um nó x e a chave k a ser inserida.
- A inserção é feita nas folhas. Assim, a função percorre a árvore (de forma recursiva) até encontrar uma folha para a inserção.

```
B-TREE-INSERT-NONFULL(x, k) {
  i = x.n;

  if x.folha

    // desloca chaves do nó para abrir espaço
    while i >= 1 e k < x.chaves[i]
      x.chaves[i + 1] = x.chaves[i];
      i = i - 1;

    // insere a nova chave
    x.chaves[i + 1] = k;
    x.n = x.n + 1;
    DISK-WRITE(x);
```

Inserção

- A função auxiliar de inserção recebe um nó x e a chave k a ser inserida.
- A inserção é feita nas folhas. Assim, a função percorre a árvore (de forma recursiva) até encontrar uma folha para a inserção.

```
B-TREE-INSERT-NONFULL(x, k) {
  i = x.n;

  if x.folha

    // desloca chaves do nó para abrir espaço
    while i >= 1 e k < x.chaves[i]
      x.chaves[i + 1] = x.chaves[i];
      i = i - 1;

    // insere a nova chave
    x.chaves[i + 1] = k;
    x.n = x.n + 1;
    DISK-WRITE(x);
```

Inserção

- A função auxiliar de inserção recebe um nó x e a chave k a ser inserida.
- A inserção é feita nas folhas. Assim, a função percorre a árvore (de forma recursiva) até encontrar uma folha para a inserção.

```
B-TREE-INSERT-NONFULL(x, k) {
   i = x.n;

   if x.folha

      // desloca chaves do nó para abrir espaço
      while i >= 1 e k < x.chaves[i]
         x.chaves[i + 1] = x.chaves[i];
         i = i - 1;

      // insere a nova chave
      x.chaves[i + 1] = k;
      x.n = x.n + 1;
      DISK-WRITE(x);
```

Inserção

```
   else
      // encontra filho para inserção
      while i >= 1 e k < x.chaves[i]
         i = i - 1;

      i = i + 1;
      DISK-READ(x.filhos[i]);

      // se nó do filho estiver cheio
      if x.filhos[i].n == 2t - 1
         B-TREE-SPLIT(x, i);

         // define o nó da inserção
         if k > x.chaves[i]
            i = i + 1;

      B-TREE-INSERT-NONFULL(x.filhos[i], k);
}
```

Remoção

- Durante a inserção, a nova chave é inserida sempre em um nó folha. Para a exclusão temos quatro casos possíveis:
  - **Caso 1:** Se a chave a ser removida está em um nó folha e removê-la não faz com que o nó fique com menos que o número mínimo de chaves permitidas, podemos remover a chave.
  - **Caso 2:** Se a chave a ser removida não está em uma folha então é garantido que seu antecessor ou sucessor está em uma folha. Neste caso, podemos excluir a chave e promover a chave antecessora ou sucessora para a posição da chave removida.

Remoção

- Durante a inserção, a nova chave é inserida sempre em um nó folha. Para a exclusão temos quatro casos possíveis:
  - **Caso 1:** Se a chave a ser removida está em um nó folha e removê-la não faz com que o nó fique com menos que o número mínimo de chaves permitidas, podemos remover a chave.
  - **Caso 2:** Se a chave a ser removida não está em uma folha então é garantido que seu antecessor ou sucessor está em uma folha. Neste caso, podemos excluir a chave e promover a chave antecessora ou sucessora para a posição da chave removida.

Remoção

- Durante a inserção, a nova chave é inserida sempre em um nó folha. Para a exclusão temos quatro casos possíveis:
  - **Caso 1:** Se a chave a ser removida está em um nó folha e removê-la não faz com que o nó fique com menos que o número mínimo de chaves permitidas, podemos remover a chave.
  - **Caso 2:** Se a chave a ser removida não está em uma folha então é garantido que seu antecessor ou sucessor está em uma folha. Neste caso, podemos excluir a chave e promover a chave antecessora ou sucessora para a posição da chave removida.

Remoção

- Se os casos **1** ou **2** levam a um nó folha que contém menos que o número mínimo de chaves, então temos que ver os irmãos imediatamente adjacentes à folha em questão:
  - **Caso 3:** Se um deles tem mais que o número mínimo de chaves, então nós podemos promover uma de suas chaves para o pai e tomar a chave pai em folha.
  - **Caso 4:** Se nenhum deles tem mais que o número de mínimo de chaves, as folhas podem ser combinadas com seu pai (o oposto da promoção de uma chave) e a nova folha terá o número correto de chaves. Se este passo deixar o progenitor com muito poucas chaves, repita o processo até a raiz, se necessário.

Remoção

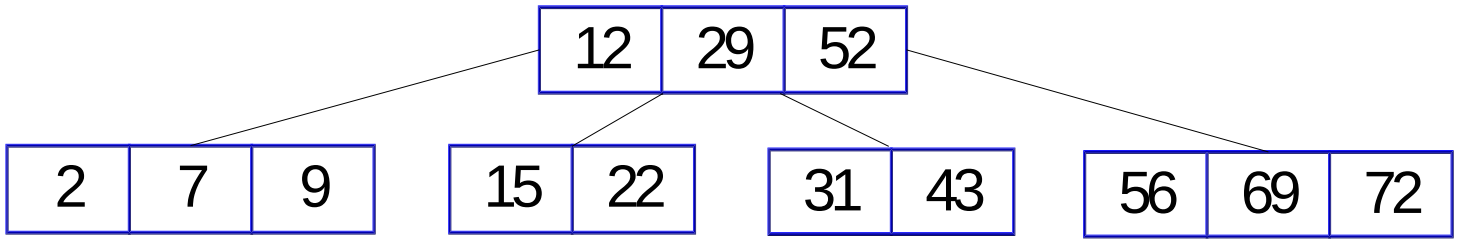
- Se os casos **1** ou **2** levam a um nó folha que contém menos que o número mínimo de chaves, então temos que ver os irmãos imediatamente adjacentes à folha em questão:
  - **Caso 3:** Se um deles tem mais que o número mínimo de chaves, então nós podemos promover uma de suas chaves para o pai e tomar a chave pai em folha.
  - **Caso 4:** Se nenhum deles tem mais que o número de mínimo de chaves, as folhas podem ser combinadas com seu pai (o oposto da promoção de uma chave) e a nova folha terá o número correto de chaves. Se este passo deixar o progenitor com muito poucas chaves, repita o processo até a raiz, se necessário.

Remoção

- Se os casos **1** ou **2** levam a um nó folha que contém menos que o número mínimo de chaves, então temos que ver os irmãos imediatamente adjacentes à folha em questão:
  - **Caso 3:** Se um deles tem mais que o número mínimo de chaves, então nós podemos promover uma de suas chaves para o pai e tomar a chave pai em folha.
  - **Caso 4:** Se nenhum deles tem mais que o número de mínimo de chaves, as folhas podem ser combinadas com seu pai (o oposto da promoção de uma chave) e a nova folha terá o número correto de chaves. Se este passo deixar o progenitor com muito poucas chaves, repita o processo até a raiz, se necessário.

Caso 1: Remoção Simples

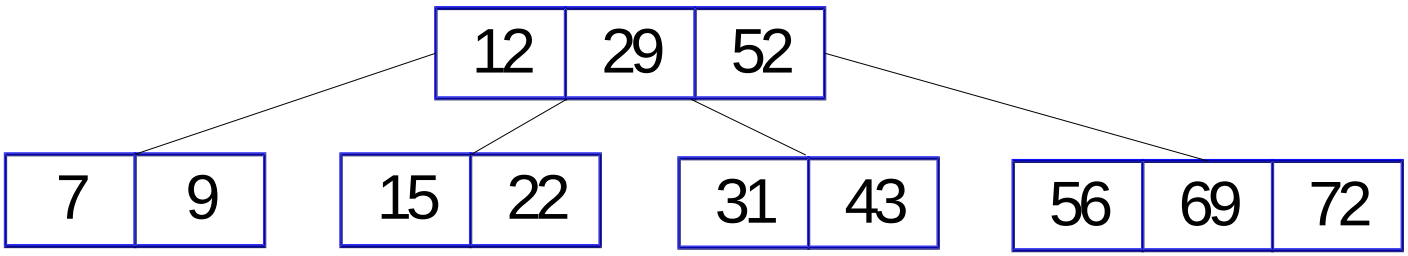
Assumindo uma árvore B de grau mínimo 3:



Removendo a chave 2.

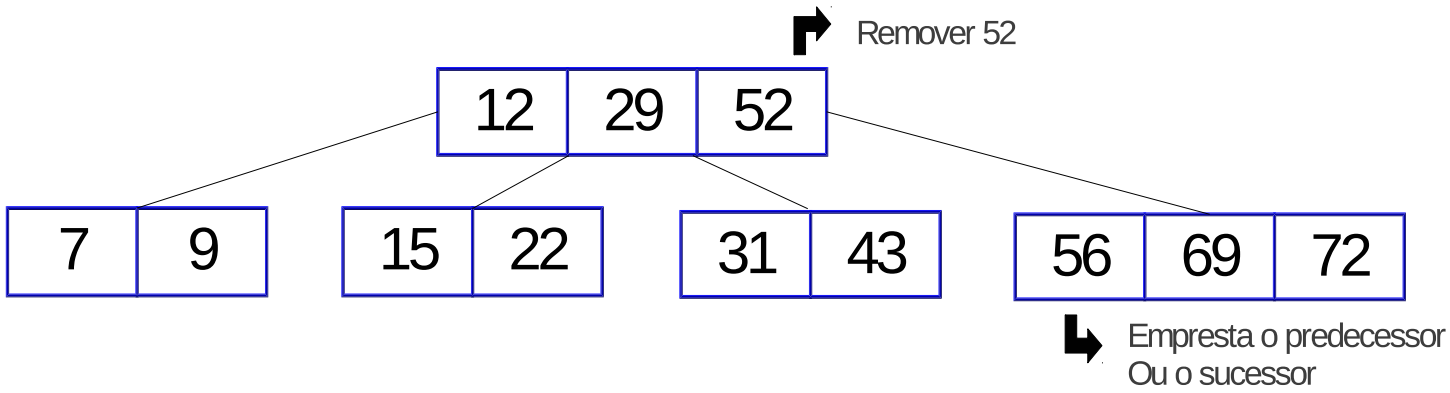
Caso 1: Remoção Simples

Assumindo uma árvore B de grau mínimo 3:



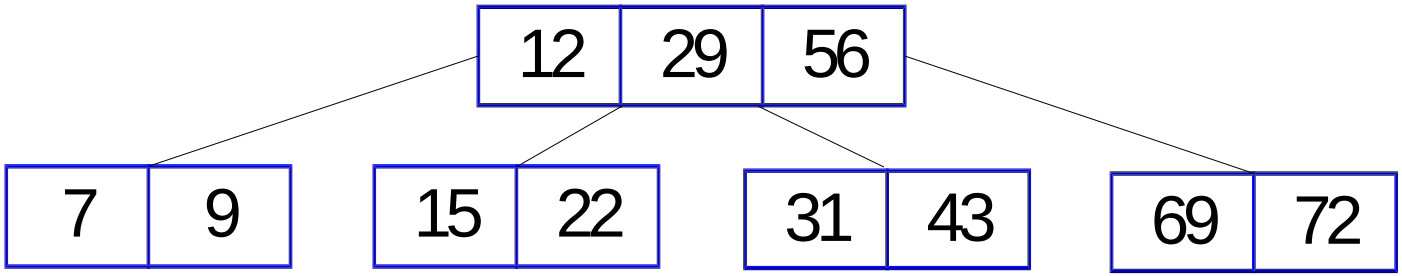
Remove 2: as chaves são suficiente, basta remover.

Caso 2: Remoção de um nó interno



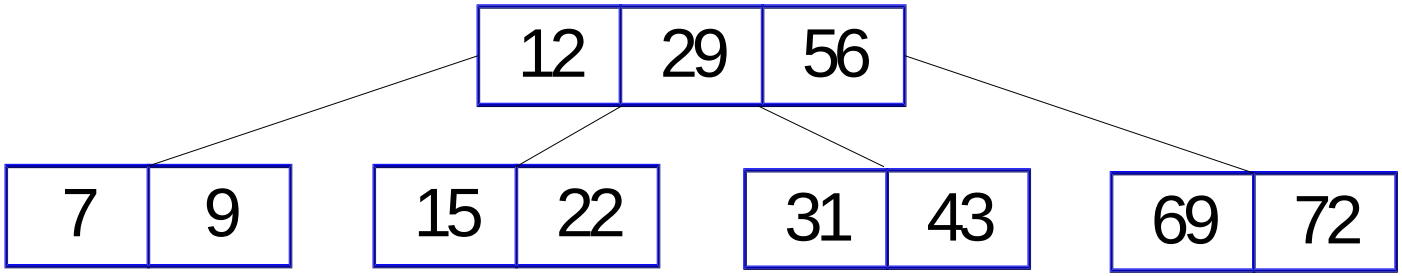
Removendo a chave 52.

Caso 2: Remoção de um nó interno



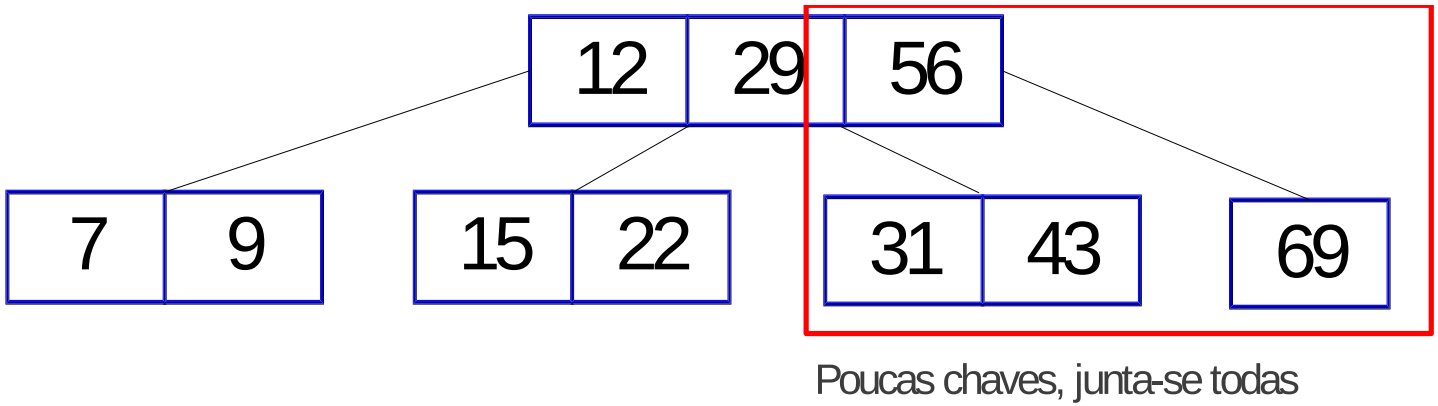
Remove 52: 52 é um nó interno, tomamos o sucessor emprestado.

Caso 4: Poucas chaves nos nós irmãos



Removendo a chave 72.

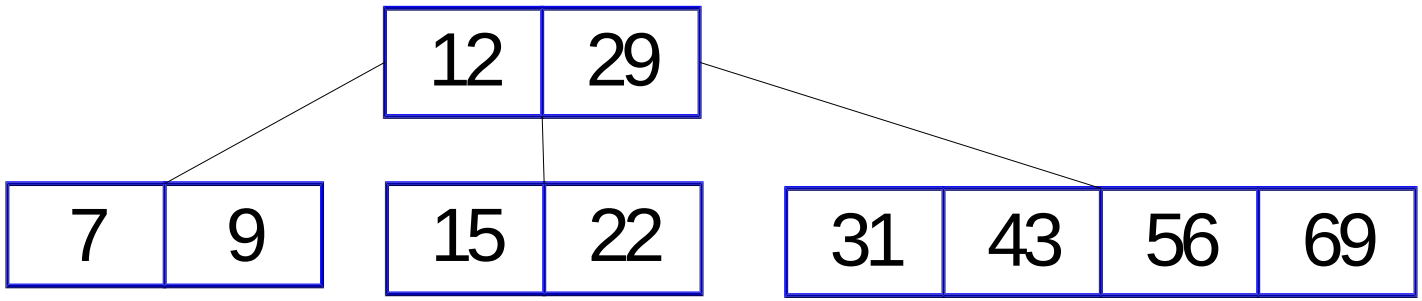
Caso 4: Poucas chaves nos nós irmãos



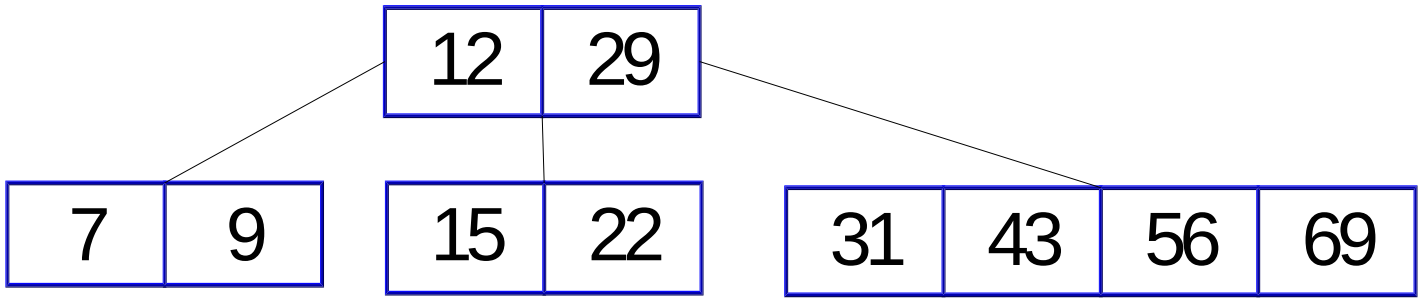
Remove 72: como há poucas chaves juntamos as chaves formando um novo nó.

Caso 4: Poucas chaves nos nós irmãos

Árvore resultante:

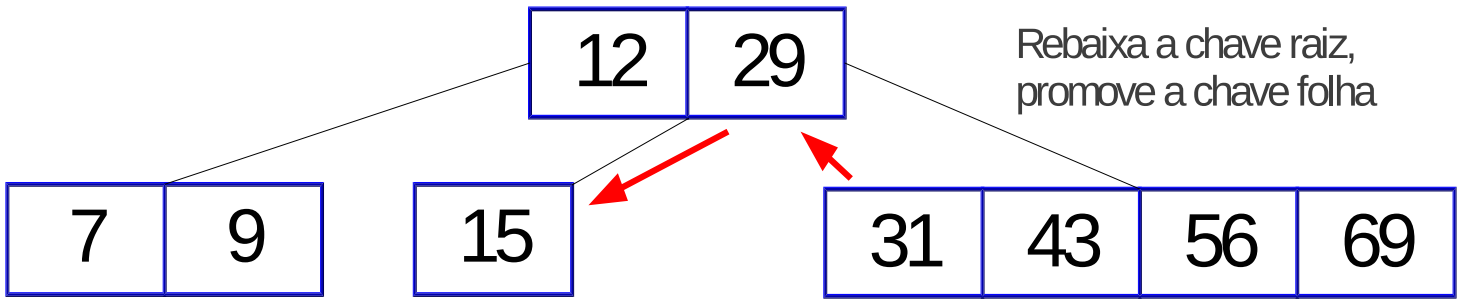


Caso 3: Irmãos suficientes



Removendo a chave 22.

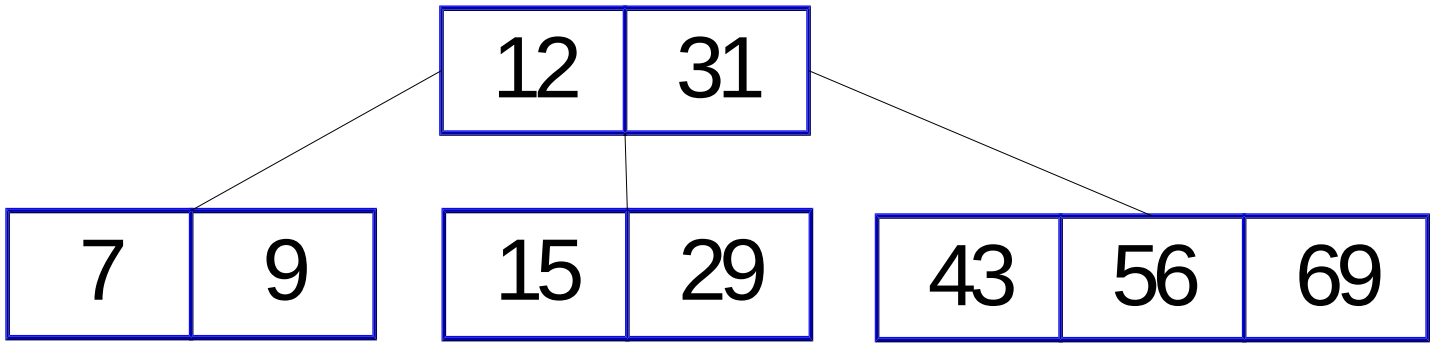
Caso 3: Irmãos suficientes



Remove 22: como os irmãos são suficientes, rebaixa a chave raiz e promove a chave folha.

Caso 3: Irmãos suficientes

Árvore resultante:



Aplicação

1. Dada a árvore B de grau mínimo 3 do último exercício: 3, 7, 9, 23, 45, 1, 5, 14, 25, 24, 13, 11, 8, 19, 4, 31, 35, 56, 15, 60, 16, 20 e 22. Remova as chaves 4, 5, 7, 3 e 14.