Árvores B

Letícia Rodrigues Bueno

UFABC

Árvores B: Introdução

- forma de armazenamento em memória secundária;

Árvores B: Introdução

- forma de armazenamento em memória secundária;
- árvores balanceadas;

Árvores B: Introdução

- forma de armazenamento em memória secundária;
- árvores balanceadas;
- assegura que folhas estão no mesmo nível;

Árvores B: Definição

- Para $d \in \mathbb{N}$, árvore B de ordem d satisfaz:

Árvores B: Definição

- Para $d \in \mathbb{N}$, árvore B de ordem d satisfaz:
  1. raiz é folha ou tem pelo menos dois filhos;

Árvores B: Definição

- Para $d \in \mathbb{N}$, árvore B de ordem d satisfaz:
  1. raiz é folha ou tem pelo menos dois filhos;
  2. cada nó diferente da raiz e folhas tem no mínimo $d + 1$ filhos;

Árvores B: Definição

- Para $d \in \mathbb{N}$, árvore B de ordem d satisfaz:
  1. raiz é folha ou tem pelo menos dois filhos;
  2. cada nó diferente da raiz e folhas tem no mínimo $d + 1$ filhos;
  3. cada nó tem no máximo $2d + 1$ filhos;

Árvores B: Definição

- Para $d \in \mathbb{N}$, árvore B de ordem d satisfaz:
  1. raiz é folha ou tem pelo menos dois filhos;
  2. cada nó diferente da raiz e folhas tem no mínimo $d + 1$ filhos;
  3. cada nó tem no máximo $2d + 1$ filhos;
  4. todas folhas estão no mesmo nível;

Árvores B: Definição

- Nó de árvore B é chamada **página** e tem propriedades:

Árvores B: Definição

- Nó de árvore B é chamada **página** e tem propriedades:
  1. Para m chaves em página P não-folha, P tem $m + 1$ filhos;

Árvores B: Definição

- Nó de árvore B é chamada **página** e tem propriedades:
  1. Para m chaves em página P não-folha, P tem $m + 1$ filhos;
  2. Cada página tem entre d e $2d$ chaves, exceto raiz que tem entre 1 e $2d$ chaves;

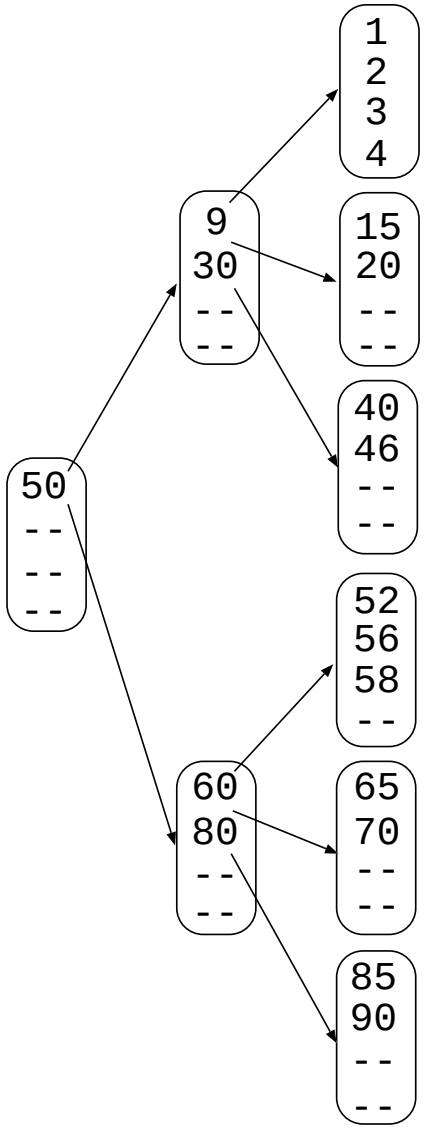
Árvores B: Definição

- Nó de árvore B é chamada **página** e tem propriedades:
  1. Para m chaves em página P não-folha, P tem $m + 1$ filhos;
  2. Cada página tem entre d e $2d$ chaves, exceto raiz que tem entre 1 e $2d$ chaves;
  3. Dentro da página, chaves estão ordenadas em ordem crescente;

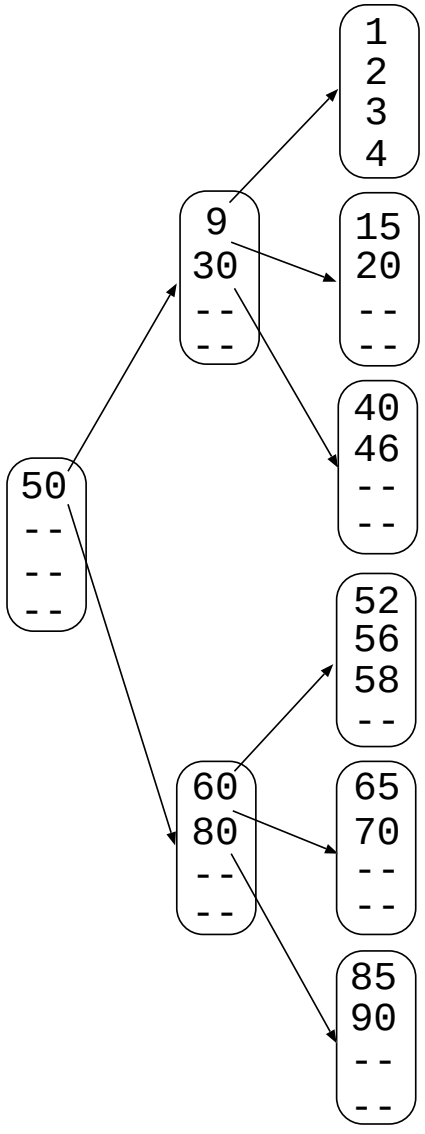
Árvores B: Definição

- Nó de árvore B é chamada **página** e tem propriedades:
  1. Para m chaves em página P não-folha, P tem $m + 1$ filhos;
  2. Cada página tem entre d e $2d$ chaves, exceto raiz que tem entre 1 e $2d$ chaves;
  3. Dentro da página, chaves estão ordenadas em ordem crescente;
  4. P contém $m + 1$ ponteiros para filhos de P;

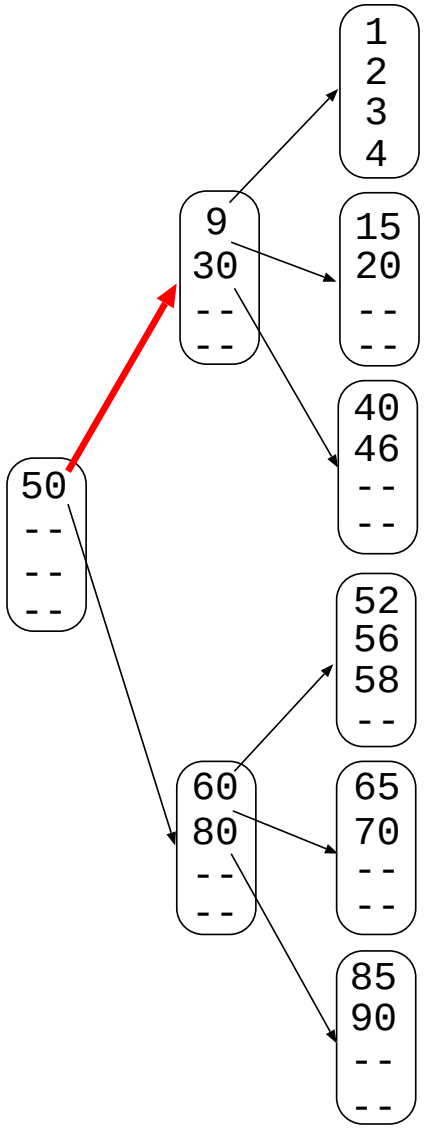
Exemplo de Árvores B com $d = 2$



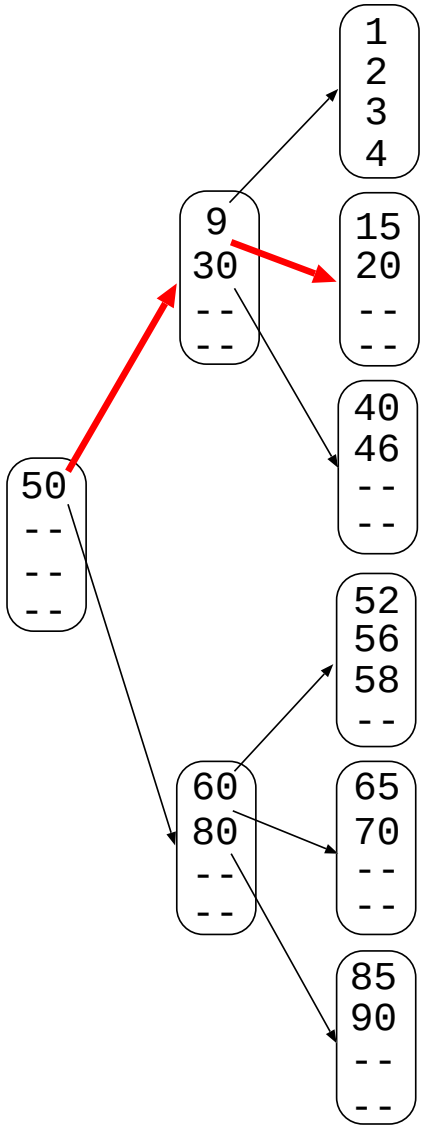
Busca da Chave 10: busca sem sucesso



Busca da Chave 10: busca sem sucesso



Busca da Chave 10: busca sem sucesso



Árvores B: algoritmo para busca

```
1  BuscaB(x, pt, f, g):
2      p = ptraiz; pt = NULL; f = 0;
3      while (p != NULL) {
4          i = 1; g = 1; pt = p;
5          while (i <= m) {
6              if (x > p->s[i]) {
7                  i = i + 1; g = i;
8              } else if (x == p->s[i]) {
9                  p = NULL; f = 1;
10             } else {
11                 p = p->pont[i - 1];
12                 i = m + 2;
13             }
14         }
15         if (i == m + 1)
16             p = p->pont[m];
17     }
```

Árvores B: algoritmo para busca

```
1  BuscaB(x, pt, f, g):
2      p = ptraiz; pt = NULL; f = 0;
3      while (p != NULL) {
4          i = 1; g = 1; pt = p;
5          while (i <= m) {
6              if (x > p->s[i]) {
7                  i = i + 1; g = i;
8              } else if (x == p->s[i]) {
9                  p = NULL; f = 1;
10             } else {
11                 p = p->pont[i - 1];
12                 i = m + 2;
13             }
14         }
15         if (i == m + 1)
16             p = p->pont[m];
17     }
```

- $f = 1$ se busca bem-sucedida (g tem posição na página e pt a página);

Árvores B: algoritmo para busca

```
1  BuscaB(x, pt, f, g):
2      p = ptraiz; pt = NULL; f = 0;
3      while (p != NULL) {
4          i = 1; g = 1; pt = p;
5          while (i <= m) {
6              if (x > p->s[i]) {
7                  i = i + 1; g = i;
8              } else if (x == p->s[i]) {
9                  p = NULL; f = 1;
10             } else {
11                 p = p->pont[i - 1];
12                 i = m + 2;
13             }
14         }
15         if (i == m + 1)
16             p = p->pont[m];
17     }
```

- $f = 1$ se busca bem-sucedida (g tem posição na página e pt a página);
- $f = 0$ se chave não encontrada (g tem posição na página onde chave deveria estar e pt tem página);

Árvores B: algoritmo para busca

```
1  BuscaB(x, pt, f, g):
2      p = ptraiz; pt = NULL; f = 0;
3      while (p != NULL) {
4          i = 1; g = 1; pt = p;
5          while (i <= m) {
6              if (x > p->s[i]) {
7                  i = i + 1; g = i;
8              } else if (x == p->s[i]) {
9                  p = NULL; f = 1;
10             } else {
11                 p = p->pont[i - 1];
12                 i = m + 2;
13             }
14         }
15         if (i == m + 1)
16             p = p->pont[m];
17     }
```

- $f = 1$ se busca bem-sucedida (g tem posição na página e pt a página);
- $f = 0$ se chave não encontrada (g tem posição na página onde chave deveria estar e pt tem página);
- **A busca dentro da página é feita com busca sequencial**

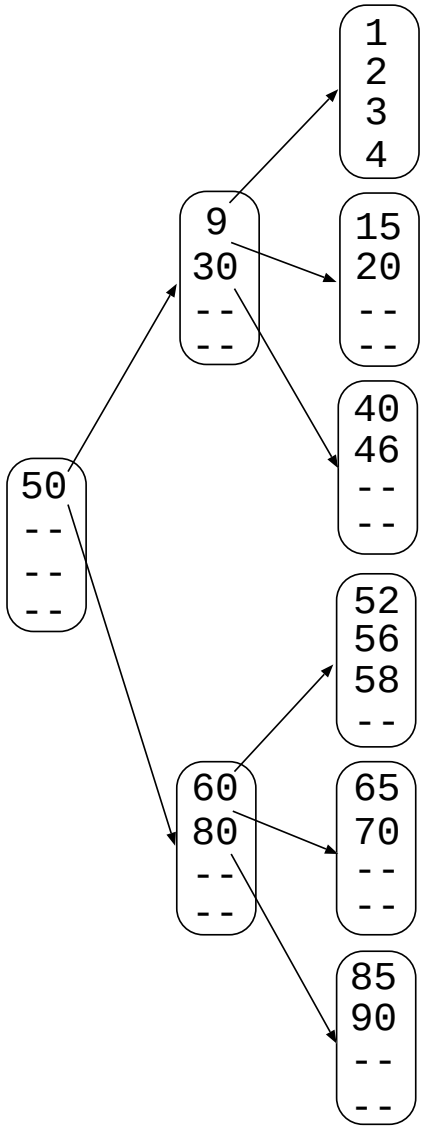
Árvores B: algoritmo para busca

```
1  BuscaB(x, pt, f, g):
2      p = ptraiz; pt = NULL; f = 0;
3      while (p != NULL) {
4          i = 1; g = 1; pt = p;
5          while (i <= m) {
6              if (x > p->s[i]) {
7                  i = i + 1; g = i;
8              } else if (x == p->s[i]) {
9                  p = NULL; f = 1;
10             } else {
11                 p = p->pont[i - 1];
12                 i = m + 2;
13             }
14         }
15         if (i == m + 1)
16             p = p->pont[m];
17     }
```

- $f = 1$ se busca bem-sucedida (g tem posição na página e pt a página);
- $f = 0$ se chave não encontrada (g tem posição na página onde chave deveria estar e pt tem página);
- **A busca dentro da página é feita com busca sequencial**
- **Pode ser melhorada???**

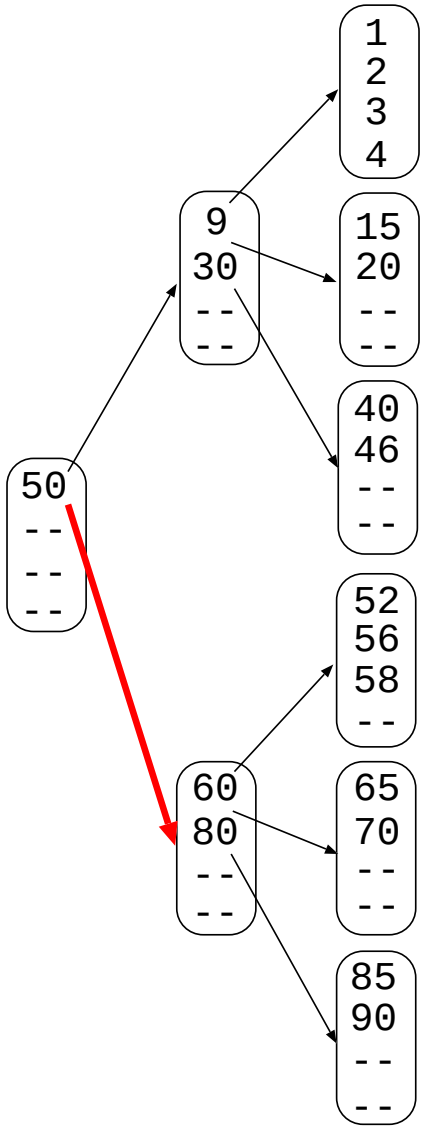
Árvores B: inserção – inserindo chave 51

Operação de balanceamento: cisão de uma página



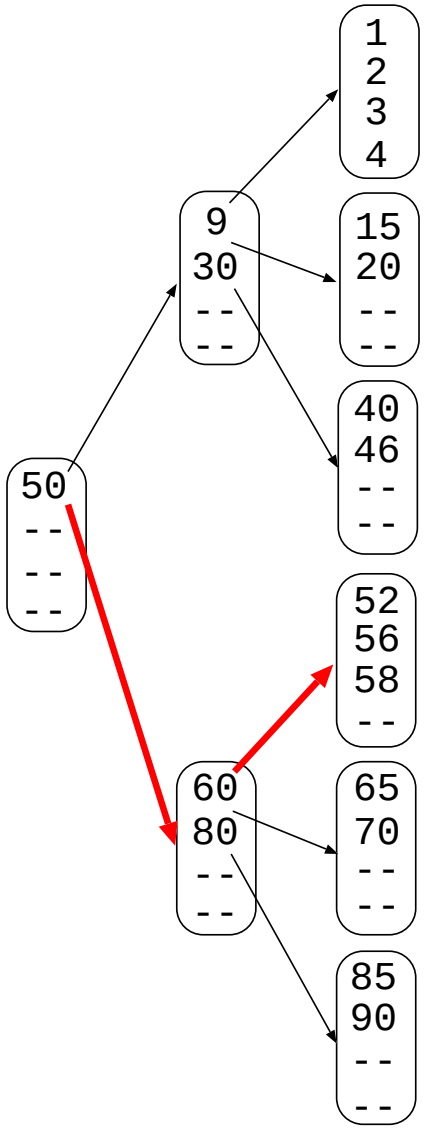
Árvores B: inserção – inserindo chave 51

Operação de balanceamento: cisão de uma página



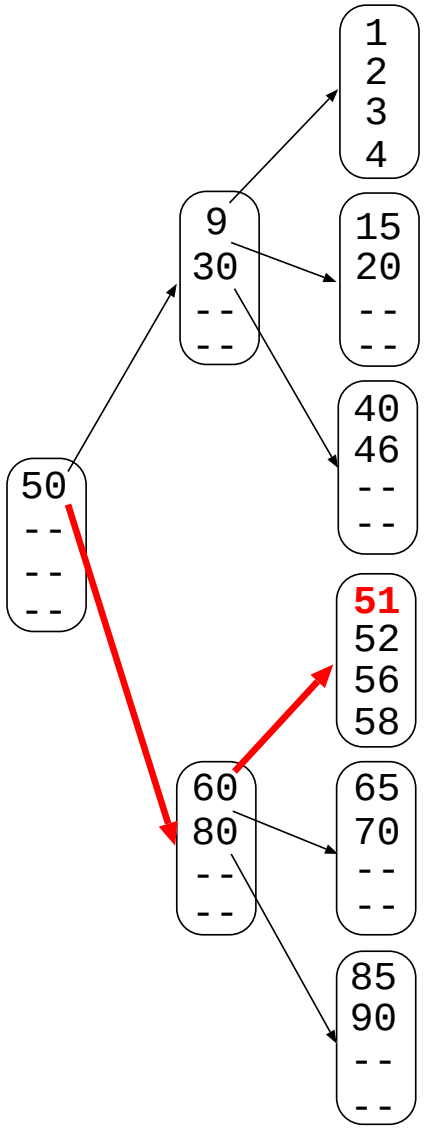
Árvores B: inserção – inserindo chave 51

Operação de balanceamento: cisão de uma página



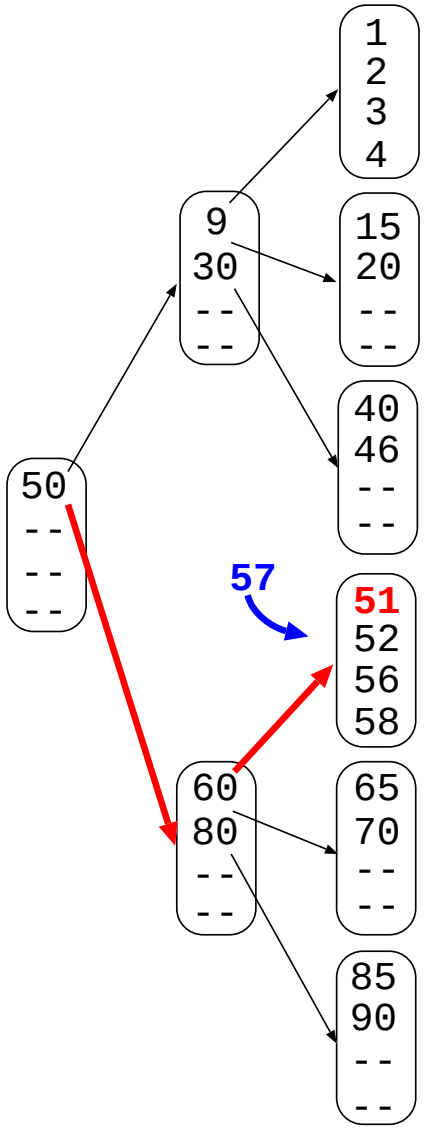
Árvores B: inserção – inserindo chave 51

Operação de balanceamento: cisão de uma página



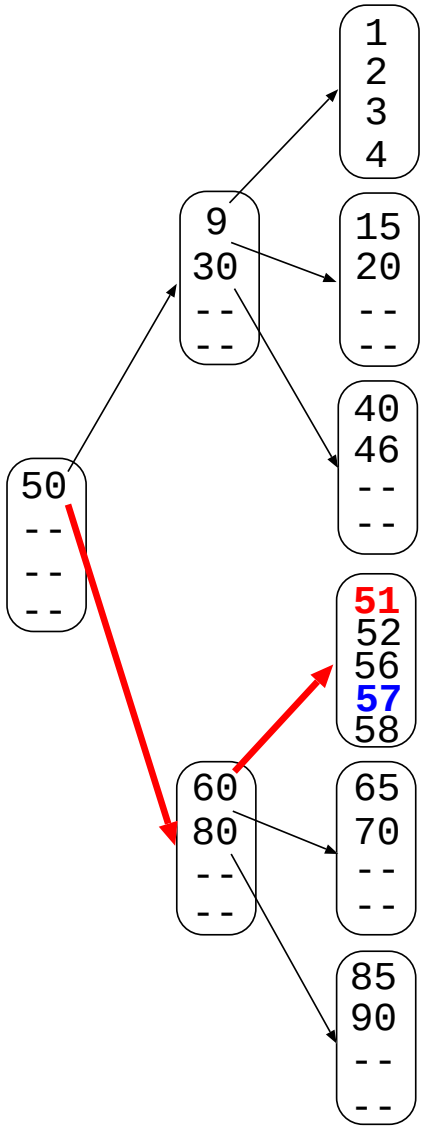
Árvores B: inserção – inserindo chave 57

Operação de balanceamento: cisão de uma página



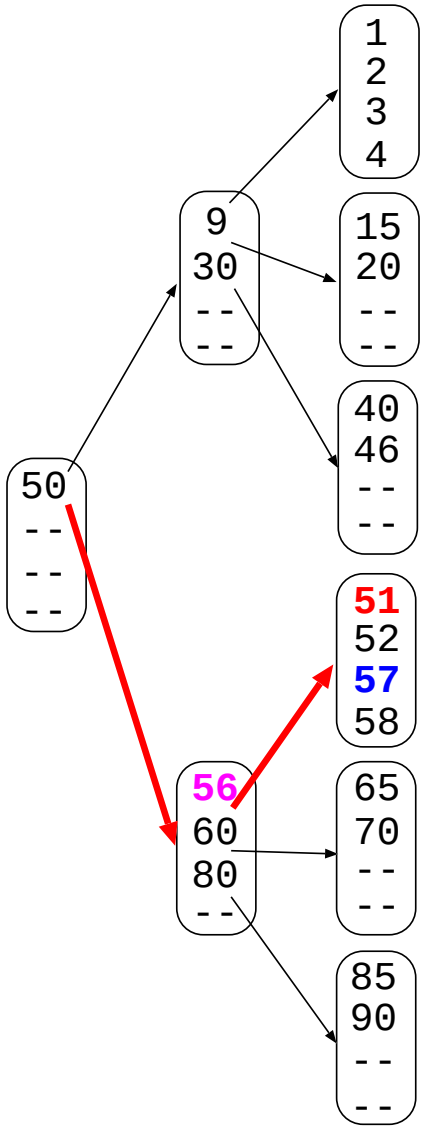
Árvores B: inserção – inserindo chave 57

Operação de balanceamento: cisão de uma página



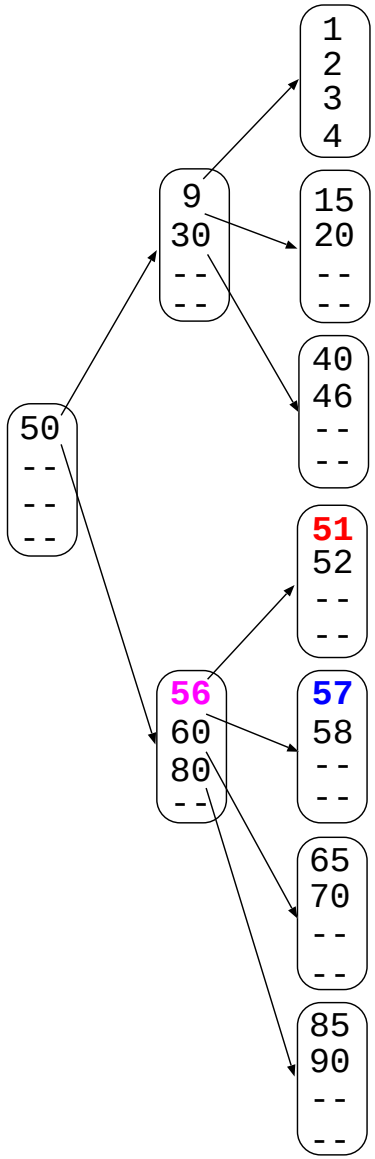
Árvores B: inserção – inserindo chave 57

Operação de balanceamento: cisão de uma página



Árvores B: inserção – inserindo chave 57

Operação de balanceamento: cisão de uma página



Árvores B: remoção

Dois casos:

Árvores B: remoção

Dois casos:

1. **chave *x* não está em folha:** *x* é substituída por *y*, a menor chave em uma folha tal que *y* é maior que *x*;

Árvores B: remoção

Dois casos:

1. **chave *x* não está em folha:** *x* é substituída por *y*, a menor chave em uma folha tal que *y* é maior que *x*;
2. chave *x* está em uma folha.

Árvores B: remoção

Dois casos:

1. **chave *x* não está em folha:** *x* é substituída por *y*, a menor chave em uma folha tal que *y* é maior que *x*;
2. chave *x* está em uma folha.

Logo, é suficiente fazer análise da remoção em uma folha!

Árvores B: remoção – removendo chave 40

Árvores B: remoção – removendo chave 40

Duas operações:

Árvores B: remoção – removendo chave 40

Duas operações:

- concatenação

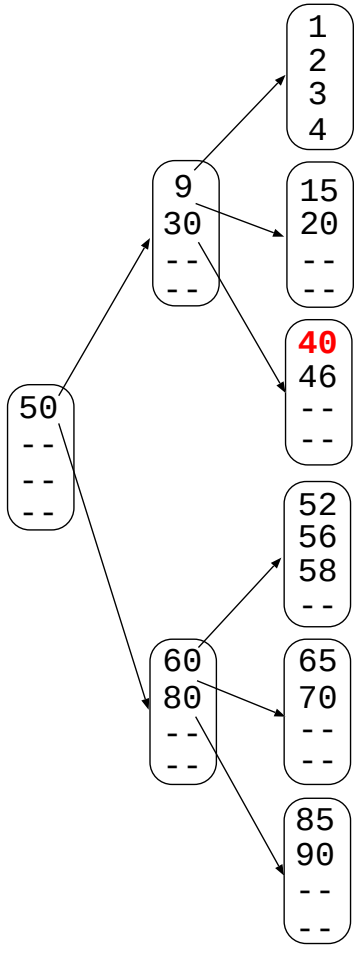
Duas operações:

- concatenação
- redistribuição

Árvores B: remoção – removendo chave 40

Duas operações:

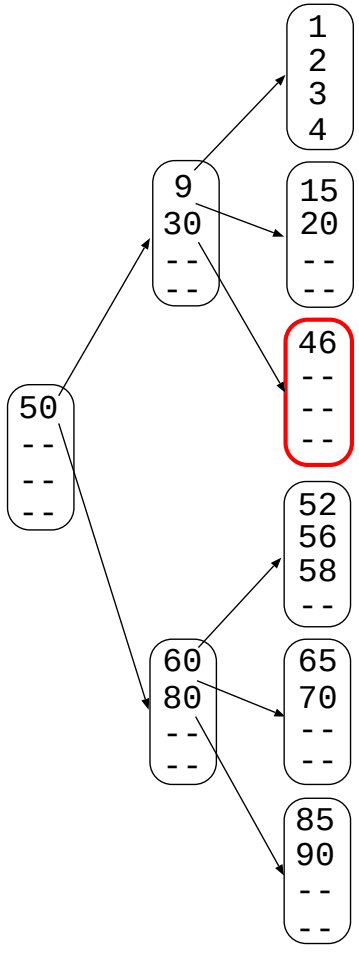
- concatenação
- redistribuição



Árvores B: remoção – removendo chave 40

Duas operações:

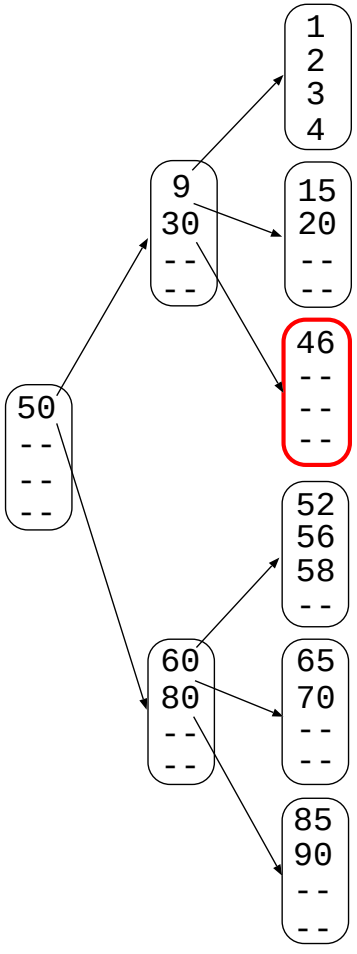
- **concatenação**
- redistribuição



Árvores B: remoção – removendo chave 40

Duas operações:

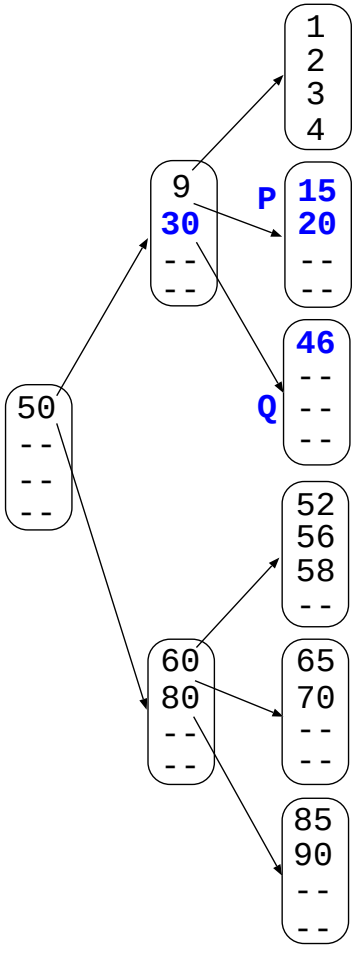
- **concatenação:**
  - irmãos adjacentes: páginas *P* e *Q* com mesmo pai e ponteiros adjacentes;
  - *P* e *Q* tem menos de 2*d* chaves;
  - Concatena *P* e *Q*;
- redistribuição



Árvores B: remoção – removendo chave 40

Duas operações:

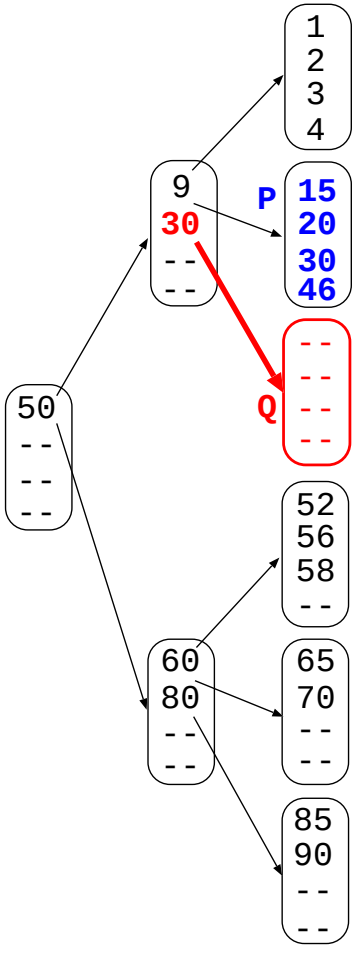
- **concatenação:**
  - irmãos adjacentes: páginas *P* e *Q* com mesmo pai e ponteiros adjacentes;
  - *P* e *Q* tem menos de 2*d* chaves;
  - Concatena *P* e *Q*;
- redistribuição



Árvores B: remoção – removendo chave 40

Duas operações:

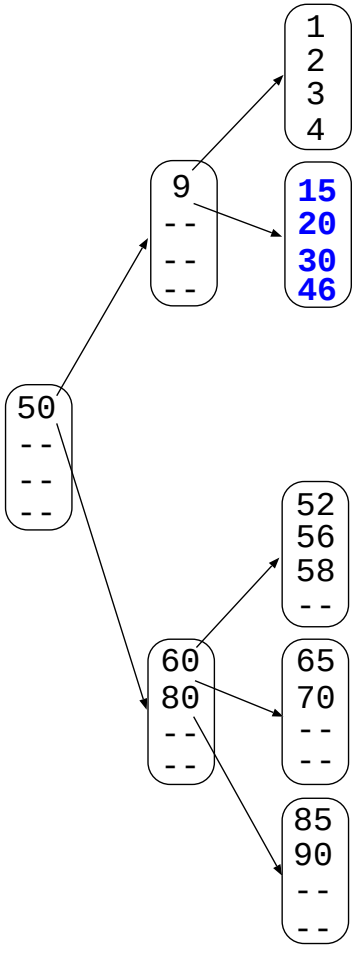
- **concatenação:**
  - irmãos adjacentes: páginas *P* e *Q* com mesmo pai e ponteiros adjacentes;
  - *P* e *Q* tem menos de 2*d* chaves;
  - Concatena *P* e *Q*;
- redistribuição



Árvores B: remoção – removendo chave 40

Duas operações:

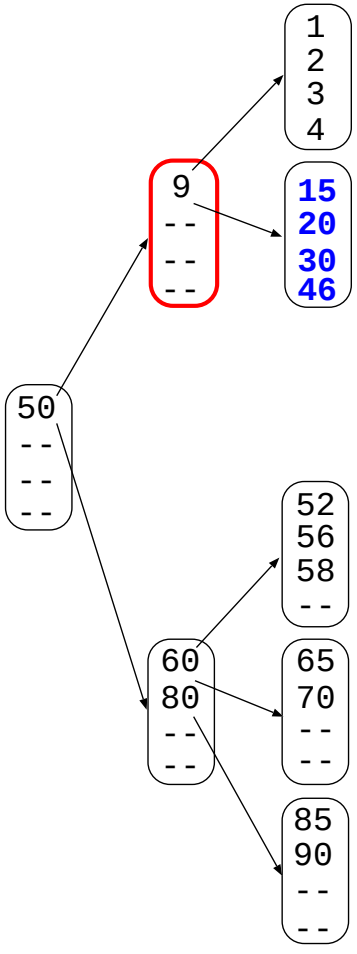
- **concatenação:**
  - irmãos adjacentes: páginas *P* e *Q* com mesmo pai e ponteiros adjacentes;
  - *P* e *Q* tem menos de 2*d* chaves;
  - Concatena *P* e *Q*;
- redistribuição



Árvores B: remoção – removendo chave 40

Duas operações:

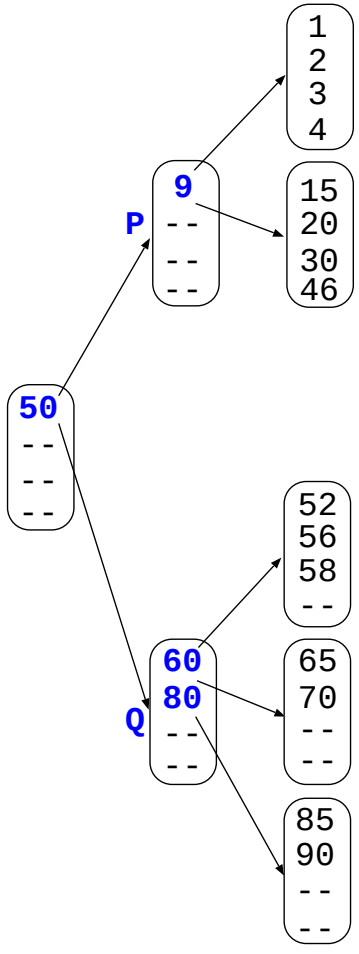
- **concatenação:**
  - irmãos adjacentes: páginas *P* e *Q* com mesmo pai e ponteiros adjacentes;
  - *P* e *Q* tem menos de 2*d* chaves;
  - Concatena *P* e *Q*;
- redistribuição



Árvores B: remoção – removendo chave 40

Duas operações:

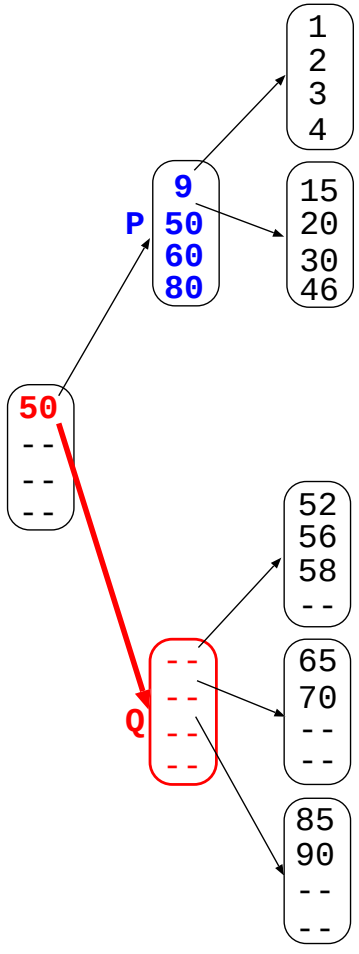
- **concatenação:**
  - irmãos adjacentes: páginas *P* e *Q* com mesmo pai e ponteiros adjacentes;
  - *P* e *Q* tem menos de 2*d* chaves;
  - Concatena *P* e *Q*;
- redistribuição



Árvores B: remoção – removendo chave 40

Duas operações:

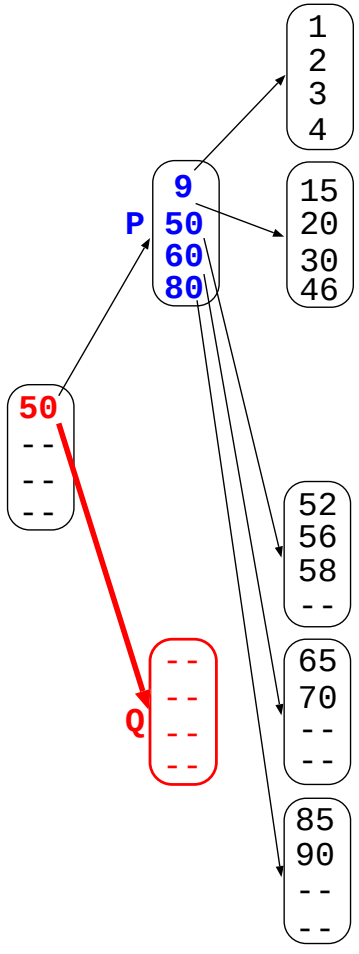
- **concatenação:**
  - irmãos adjacentes: páginas *P* e *Q* com mesmo pai e ponteiros adjacentes;
  - *P* e *Q* tem menos de 2*d* chaves;
  - Concatena *P* e *Q*;
- redistribuição



Árvores B: remoção – removendo chave 40

Duas operações:

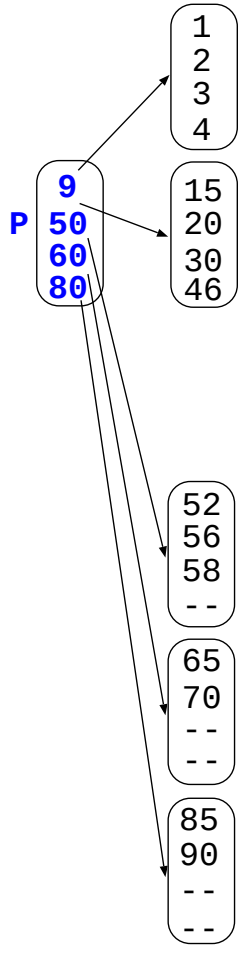
- **concatenação:**
  - irmãos adjacentes: páginas *P* e *Q* com mesmo pai e ponteiros adjacentes;
  - *P* e *Q* tem menos de 2*d* chaves;
  - Concatena *P* e *Q*;
- redistribuição



Árvores B: remoção – removendo chave 40

Duas operações:

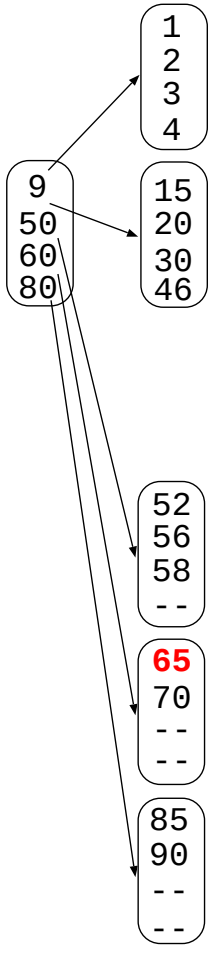
- **concatenação:**
  - irmãos adjacentes: páginas *P* e *Q* com mesmo pai e ponteiros adjacentes;
  - *P* e *Q* tem menos de 2*d* chaves;
  - Concatena *P* e *Q*;
- redistribuição



Árvores B: remoção – removendo chave 65

Duas operações:

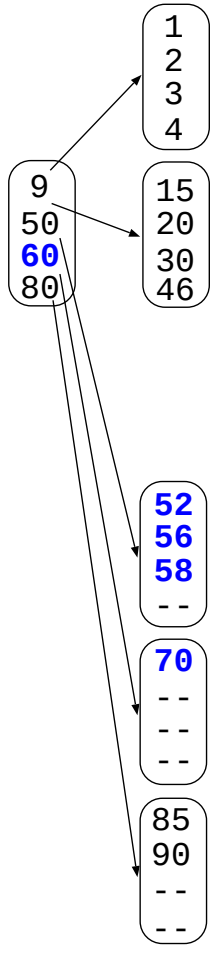
- concatenação
- **redistribuição:**
  - irmãos adjacentes *P* e *Q* com pai *W*;
  - *P* e *Q* tem 2*d* chaves ou mais;
  - redistribui chaves de *P* e *Q* com pai *W*;



Árvores B: remoção – removendo chave 65

Duas operações:

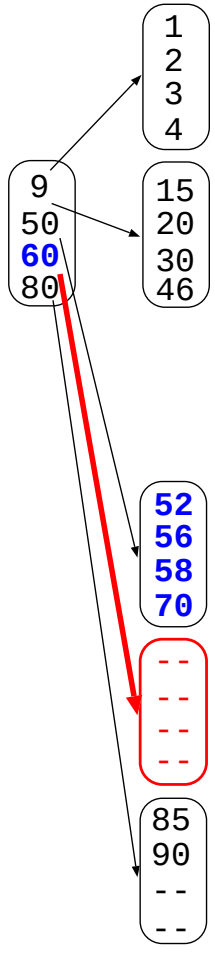
- concatenação
- **redistribuição:**
  - irmãos adjacentes *P* e *Q* com pai *W*;
  - *P* e *Q* tem 2*d* chaves ou mais;
  - redistribui chaves de *P* e *Q* com pai *W*;



Árvores B: remoção – removendo chave 65

Duas operações:

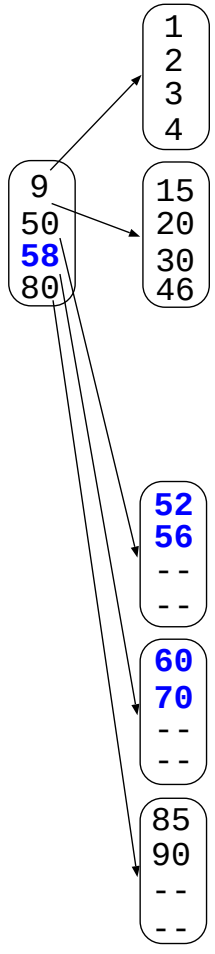
- concatenação
- **redistribuição:**
  - irmãos adjacentes *P* e *Q* com pai *W*;
  - *P* e *Q* tem 2*d* chaves ou mais;
  - redistribui chaves de *P* e *Q* com pai *W*;



Árvores B: remoção – removendo chave 65

Duas operações:

- concatenação
- **redistribuição:**
  - irmãos adjacentes *P* e *Q* com pai *W*;
  - *P* e *Q* tem 2*d* chaves ou mais;
  - redistribui chaves de *P* e *Q* com pai *W*;



Exercícios Propostos

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

Exercícios Propostos

1. Desenhe uma árvore B de ordem 3 que contenha as seguintes chaves: 1, 3, 6, 8, 14, 32, 36, 38, 39, 41, 43.
2. Desenhe uma árvore B de ordem 1 que contenha as seguintes chaves: 2, 5, 7, 10, 13, 16, 18, 21.

Exercícios Propostos

1. Desenhe uma árvore B de ordem 3 que contenha as seguintes chaves: 1, 3, 6, 8, 14, 32, 36, 38, 39, 41, 43.
2. Desenhe uma árvore B de ordem 1 que contenha as seguintes chaves: 2, 5, 7, 10, 13, 16, 18, 21.
3. Dê exemplo de uma cisão de página que se propaga até à raiz.

Exercícios Propostos

1. Desenhe uma árvore B de ordem 3 que contenha as seguintes chaves: 1, 3, 6, 8, 14, 32, 36, 38, 39, 41, 43.
2. Desenhe uma árvore B de ordem 1 que contenha as seguintes chaves: 2, 5, 7, 10, 13, 16, 18, 21.
3. Dê exemplo de uma cisão de página que se propaga até à raiz.
4. Dê exemplo de uma concatenação que se propaga até à raiz.

Exercícios Propostos

1. Desenhe uma árvore B de ordem 3 que contenha as seguintes chaves: 1, 3, 6, 8, 14, 32, 36, 38, 39, 41, 43.
2. Desenhe uma árvore B de ordem 1 que contenha as seguintes chaves: 2, 5, 7, 10, 13, 16, 18, 21.
3. Dê exemplo de uma cisão de página que se propaga até à raiz.
4. Dê exemplo de uma concatenação que se propaga até à raiz.
5. A redistribuição é propagável? Justifique.

Exercícios Propostos

1. Desenhe uma árvore B de ordem 3 que contenha as seguintes chaves: 1, 3, 6, 8, 14, 32, 36, 38, 39, 41, 43.
2. Desenhe uma árvore B de ordem 1 que contenha as seguintes chaves: 2, 5, 7, 10, 13, 16, 18, 21.
3. Dê exemplo de uma cisão de página que se propaga até à raiz.
4. Dê exemplo de uma concatenação que se propaga até à raiz.
5. A redistribuição é propagável? Justifique.
6. Escreva o algoritmo de inserção em uma árvore B.

Exercícios Propostos

1. Desenhe uma árvore B de ordem 3 que contenha as seguintes chaves: 1, 3, 6, 8, 14, 32, 36, 38, 39, 41, 43.
2. Desenhe uma árvore B de ordem 1 que contenha as seguintes chaves: 2, 5, 7, 10, 13, 16, 18, 21.
3. Dê exemplo de uma cisão de página que se propaga até à raiz.
4. Dê exemplo de uma concatenação que se propaga até à raiz.
5. A redistribuição é propagável? Justifique.
6. Escreva o algoritmo de inserção em uma árvore B.
7. Escreva o algoritmo de remoção em uma árvore B.

Bibliografia Utilizada

SZWARCFITER, J. L. e MARKENZON, L. Estruturas de Dados e seus Algoritmos, LTC, 1994.

ZIVIANI, N. Projeto de Algoritmos: com implementações em Pascal e C, 2ª edição, Cengage Learning, 2009.