Árvores B*

Letícia Rodrigues Bueno

UFABC

Árvores B*

Árvores B*

Aplicações que utilizam algum tipo de implementação de Árvores B ou B*:

Árvores B*

Aplicações que utilizam algum tipo de implementação de Árvores B ou B*:

- **Sistemas de arquivos:** NTFS, ReiserFS, NSS, XFS, JFS, ReFS, HFS+ e HFS (Apple), AIX, sistemas linux como brtfs e Ext4, etc;

Árvores B*

Aplicações que utilizam algum tipo de implementação de Árvores B ou B*:

- **Sistemas de arquivos:** NTFS, ReiserFS, NSS, XFS, JFS, ReFS, HFS+ e HFS (Apple), AIX, sistemas linux como brtfs e Ext4, etc;
- **Bancos de Dados Relacionais:** IBM DB2, Informix, Microsoft SQL Server, Oracle 8, Sybase ASE, SQLite, etc;

Árvores B*: Introdução

Árvores B*: Introdução

- todos registros são armazenados no último nível (folhas);

Árvores B*: Introdução

- todos registros são armazenados no último nível (folhas);
- **níveis acima do último nível:** é o índice organizado como árvore B;

Árvores B*: Introdução

- todos registros são armazenados no último nível (folhas);
- **níveis acima do último nível:** é o índice organizado como árvore B;
- no índice só aparecem chaves, sem a informação associada;

Árvores B*: Introdução

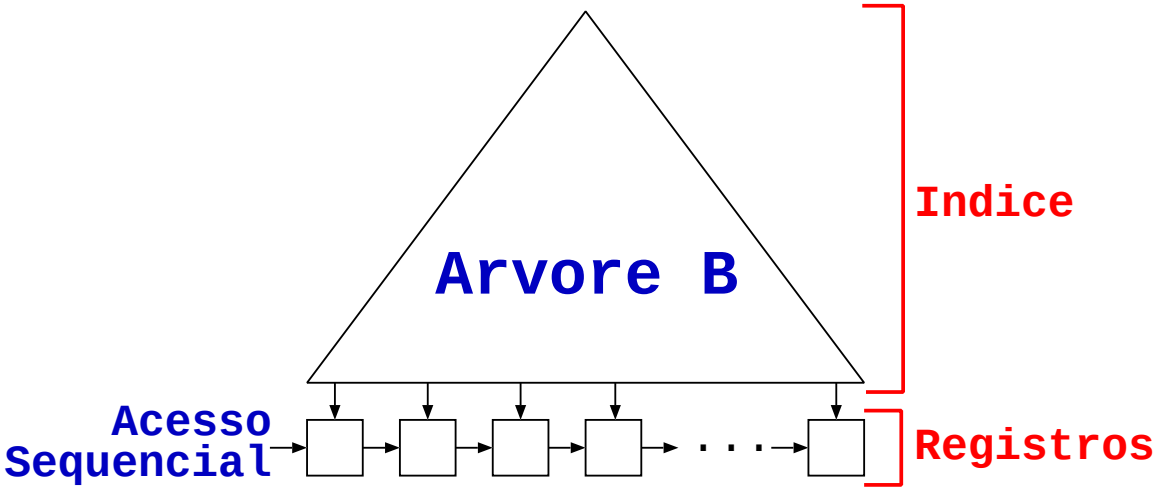
- todos registros são armazenados no último nível (folhas);
- **níveis acima do último nível:** é o índice organizado como árvore B;
- no índice só aparecem chaves, sem a informação associada;
- páginas folha contém todos registros do arquivo;

Árvores B*: Introdução

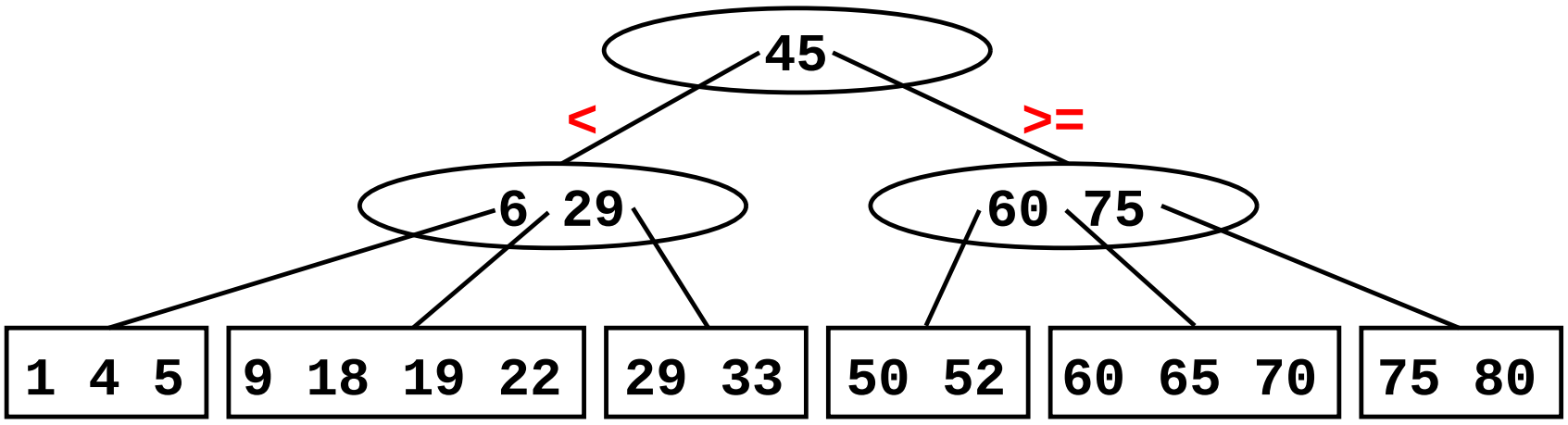
- todos registros são armazenados no último nível (folhas);
- **níveis acima do último nível:** é o índice organizado como árvore B;
- no índice só aparecem chaves, sem a informação associada;
- páginas folha contém todos registros do arquivo;
- **páginas folha conectadas da esquerda para direita (lista encadeada):** acesso sequencial mais eficiente que acesso via índice;

Árvores B*: Introdução

- todos registros são armazenados no último nível (folhas);
- **níveis acima do último nível:** é o índice organizado como árvore B;
- no índice só aparecem chaves, sem a informação associada;
- páginas folha contém todos registros do arquivo;
- **páginas folha conectadas da esquerda para direita (lista encadeada):** acesso sequencial mais eficiente que acesso via índice;



Exemplo de Árvore B* com $d = 2$



Árvores B*: estrutura em C

```
typedef struct Registro {
    int chave; /* e outros componentes*/
}
typedef enum {
    Interna, Externa
} PaginaTipo;
typedef struct Pagina *Apontador;
typedef struct Pagina {
    PaginaTipo Pt;
    union {
        struct {
            int ni;
            int ri[m];
            Apontador pi[m+1];
        } U0;
        struct {
            int ne;
            Registro re[m2];
        } U1;
    } U;
} Pagina;
```

Árvores B*: estrutura em C

```
typedef struct Registro {
    int chave; /* e outros componentes*/
}
typedef enum {
    Interna, Externa
} PaginaTipo;
typedef struct Pagina *Apontador;
typedef struct Pagina {
    PaginaTipo Pt;
    union {
        struct {
            int ni;
            int ri[m];
            Apontador pi[m+1];
        } U0;
        struct {
            int ne;
            Registro re[m2];
        } U1;
    } U;
} Pagina;
```

- **ni**: número chaves na página;

Árvores B*: estrutura em C

```
typedef struct Registro {
    int chave; /* e outros componentes*/
}
typedef enum {
    Interna, Externa
} PaginaTipo;
typedef struct Pagina *Apontador;
typedef struct Pagina {
    PaginaTipo Pt;
    union {
        struct {
            int ni;
            int ri[m];
            Apontador pi[m+1];
        } U0;
        struct {
            int ne;
            Registro re[m2];
        } U1;
    } U;
} Pagina;
```

- **ni**: número chaves na página;
- **ri**: vetor de chaves;

Árvores B*: estrutura em C

```
typedef struct Registro {
    int chave; /* e outros componentes*/
}
typedef enum {
    Interna, Externa
} PaginaTipo;
typedef struct Pagina *Apontador;
typedef struct Pagina {
    PaginaTipo Pt;
    union {
        struct {
            int ni;
            int ri[m];
            Apontador pi[m+1];
        } U0;
        struct {
            int ne;
            Registro re[m2];
        } U1;
    } U;
} Pagina;
```

- **ni**: número chaves na página;
- **ri**: vetor de chaves;
- **pi**: vetor de ponteiros;

Árvores B*: estrutura em C

```
typedef struct Registro {
    int chave; /* e outros componentes*/
}
typedef enum {
    Interna, Externa
} PaginaTipo;
typedef struct Pagina *Apontador;
typedef struct Pagina {
    PaginaTipo Pt;
    union {
        struct {
            int ni;
            int ri[m];
            Apontador pi[m+1];
        } U0;
        struct {
            int ne;
            Registro re[m2];
        } U1;
    } U;
} Pagina;
```

- **ni**: número chaves na página;
- **ri**: vetor de chaves;
- **pi**: vetor de ponteiros;
- **ne**: número registros na página;

Árvores B*: estrutura em C

```
typedef struct Registro {
    int chave; /* e outros componentes*/
}
typedef enum {
    Interna, Externa
} PaginaTipo;
typedef struct Pagina *Apontador;
typedef struct Pagina {
    PaginaTipo Pt;
    union {
        struct {
            int ni;
            int ri[m];
            Apontador pi[m+1];
        } U0;
        struct {
            int ne;
            Registro re[m2];
        } U1;
    } U;
} Pagina;
```

- **ni**: número chaves na página;
- **ri**: vetor de chaves;
- **pi**: vetor de ponteiros;
- **ne**: número registros na página;
- **re**: vetor de registros;

Árvores B*: estrutura em C

```
typedef struct Registro {
    int chave; /* e outros componentes*/
}
typedef enum {
    Interna, Externa
} PaginaTipo;
typedef struct Pagina *Apontador;
typedef struct Pagina {
    PaginaTipo Pt;
    union {
        struct {
            int ni;
            int ri[m];
            Apontador pi[m+1];
        } U0;
        struct {
            int ne;
            Registro re[m2];
        } U1;
    } U;
} Pagina;
```

- **ni**: número chaves na página;
- **ri**: vetor de chaves;
- **pi**: vetor de ponteiros;
- **ne**: número registros na página;
- **re**: vetor de registros;
- **union:** conserva espaço ao permitir tipos diferentes serem armazenados no mesmo espaço;

Árvores B*: Vantagens sobre Árvores B

Árvores B*: Vantagens sobre Árvores B

- acesso sequencial mais eficiente;

Árvores B*: Vantagens sobre Árvores B

- acesso sequencial mais eficiente;
- facilitam acesso concorrente ao arquivo;

Árvores B*: Vantagens sobre Árvores B

- acesso sequencial mais eficiente;
- facilitam acesso concorrente ao arquivo;
- páginas folha sem ponteiros: espaço para mais registros;

Árvores B*: Vantagens sobre Árvores B

- acesso sequencial mais eficiente;
- facilitam acesso concorrente ao arquivo;
- páginas folha sem ponteiros: espaço para mais registros;
- algumas aplicações requerem 2 pontos de vista de arquivo:

Árvores B*: Vantagens sobre Árvores B

- acesso sequencial mais eficiente;
- facilitam acesso concorrente ao arquivo;
- páginas folha sem ponteiros: espaço para mais registros;
- algumas aplicações requerem 2 pontos de vista de arquivo:
  1. acesso sequencial;

Árvores B*: Vantagens sobre Árvores B

- acesso sequencial mais eficiente;
- facilitam acesso concorrente ao arquivo;
- páginas folha sem ponteiros: espaço para mais registros;
- algumas aplicações requerem 2 pontos de vista de arquivo:
  1. acesso sequencial;
  2. acesso por índice;

Árvores B*: Vantagens sobre Árvores B

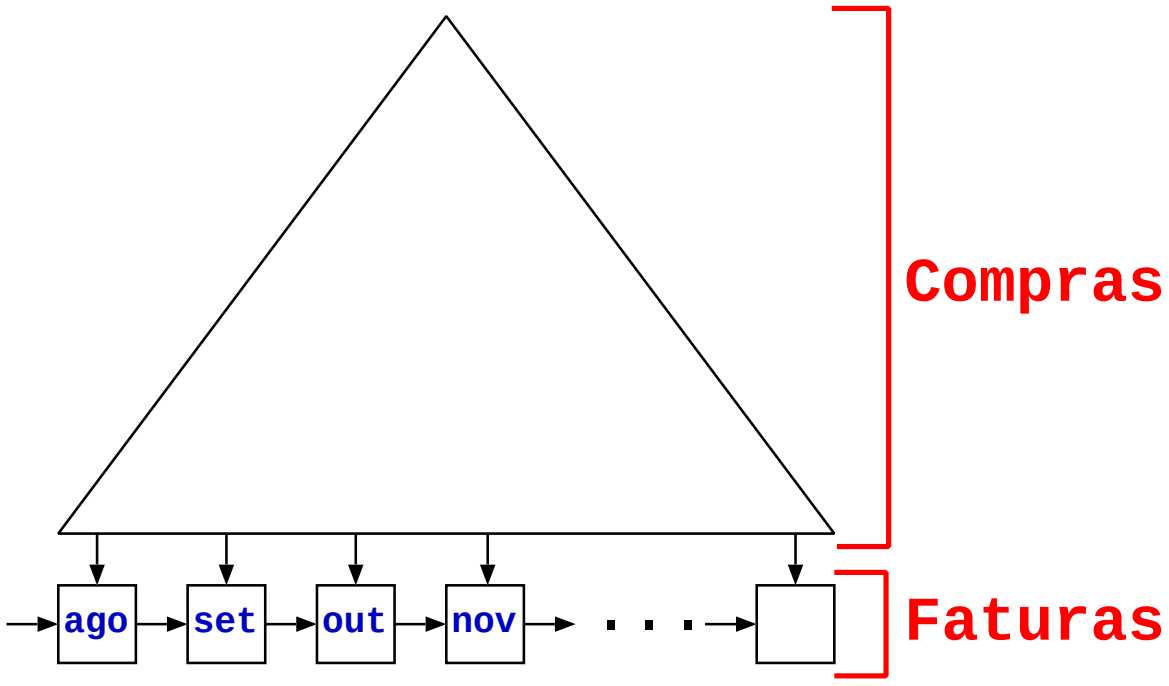
- acesso sequencial mais eficiente;
- facilitam acesso concorrente ao arquivo;
- páginas folha sem ponteiros: espaço para mais registros;
- algumas aplicações requerem 2 pontos de vista de arquivo:
  1. acesso sequencial;
  2. acesso por índice;

Exemplo (sistema de cartão de crédito):

Árvores B*: Vantagens sobre Árvores B

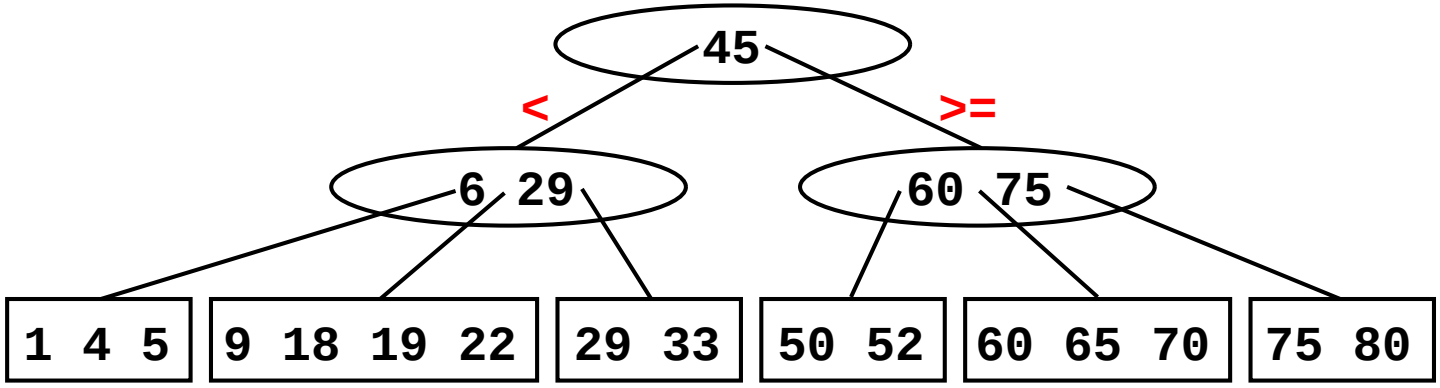
- acesso sequencial mais eficiente;
- facilitam acesso concorrente ao arquivo;
- páginas folha sem ponteiros: espaço para mais registros;
- algumas aplicações requerem 2 pontos de vista de arquivo:
  1. acesso sequencial;
  2. acesso por índice;

Exemplo (sistema de cartão de crédito):

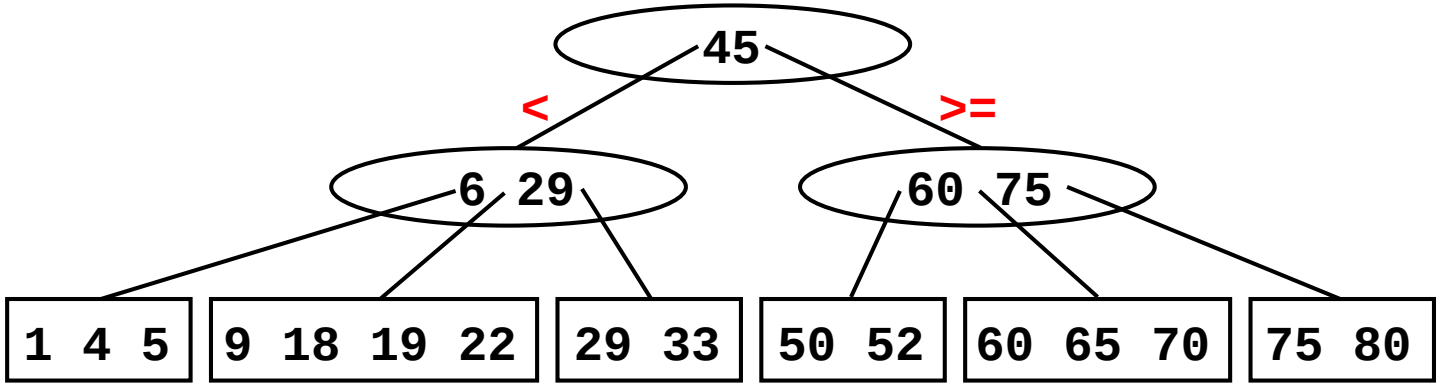


Busca em Árvores B*

Busca em Árvores B*

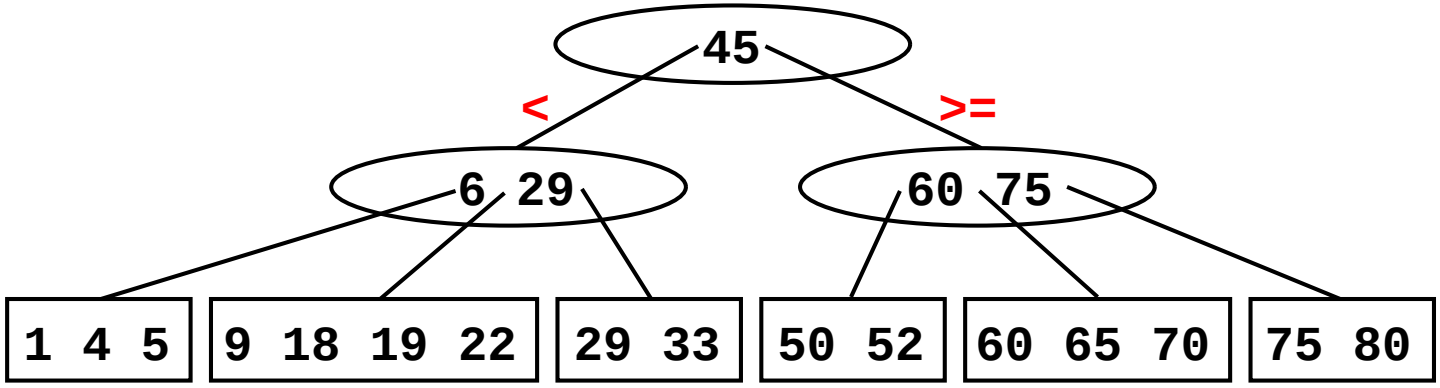


Busca em Árvores B*



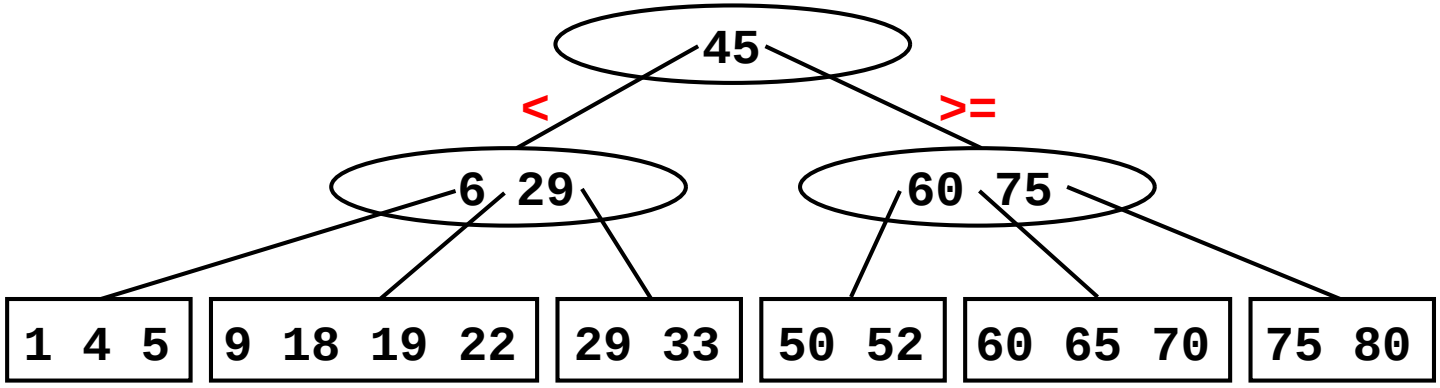
- pesquisa não pára se chave procurada é encontrada em página do índice;

Busca em Árvores B*



- pesquisa não pára se chave procurada é encontrada em página do índice;
- nesse caso, segue ponteiro à direita;

Busca em Árvores B*



- pesquisa não pára se chave procurada é encontrada em página do índice;
- nesse caso, segue ponteiro à direita;
- valores no índice são irrelevantes desde que conduzam à página folha correta;

Inserção em Árvores B*

Inserção em Árvores B*

- essencialmente igual à árvores B, porém...

Inserção em Árvores B*

- essencialmente igual à árvores B, porém...
- **na cisão:** algoritmo faz cópia da chave na página pai, ao invés de mover todo registro;

Inserção em Árvores B*

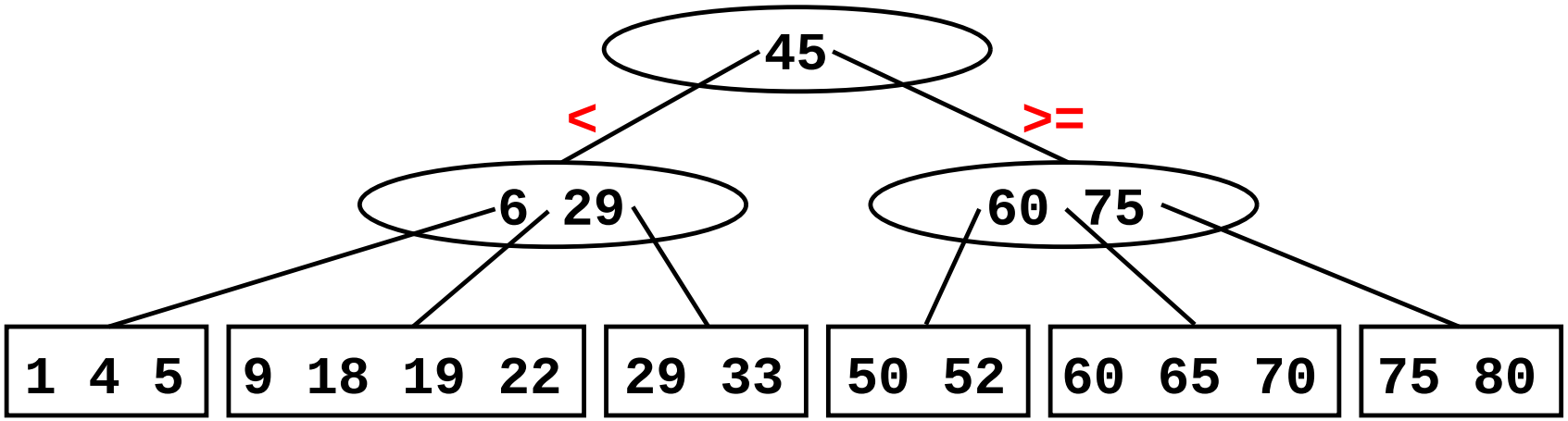
- essencialmente igual à árvores B, porém...
- **na cisão:** algoritmo faz cópia da chave na página pai, ao invés de mover todo registro;
- registro é retido na página folha da direita;

Inserção em Árvores B*

- essencialmente igual à árvores B, porém...
- **na cisão:** algoritmo faz cópia da chave na página pai, ao invés de mover todo registro;
- registro é retido na página folha da direita;
- **Inserindo chave** 17**:**

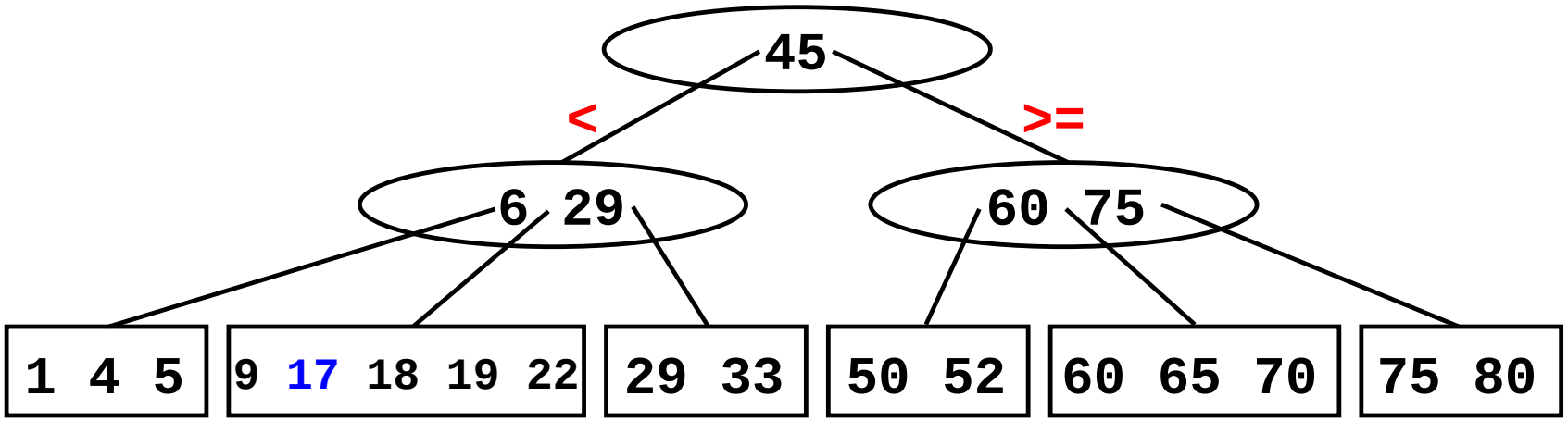
Inserção em Árvores B*

- essencialmente igual à árvores B, porém...
- **na cisão:** algoritmo faz cópia da chave na página pai, ao invés de mover todo registro;
- registro é retido na página folha da direita;
- **Inserindo chave** 17**:**



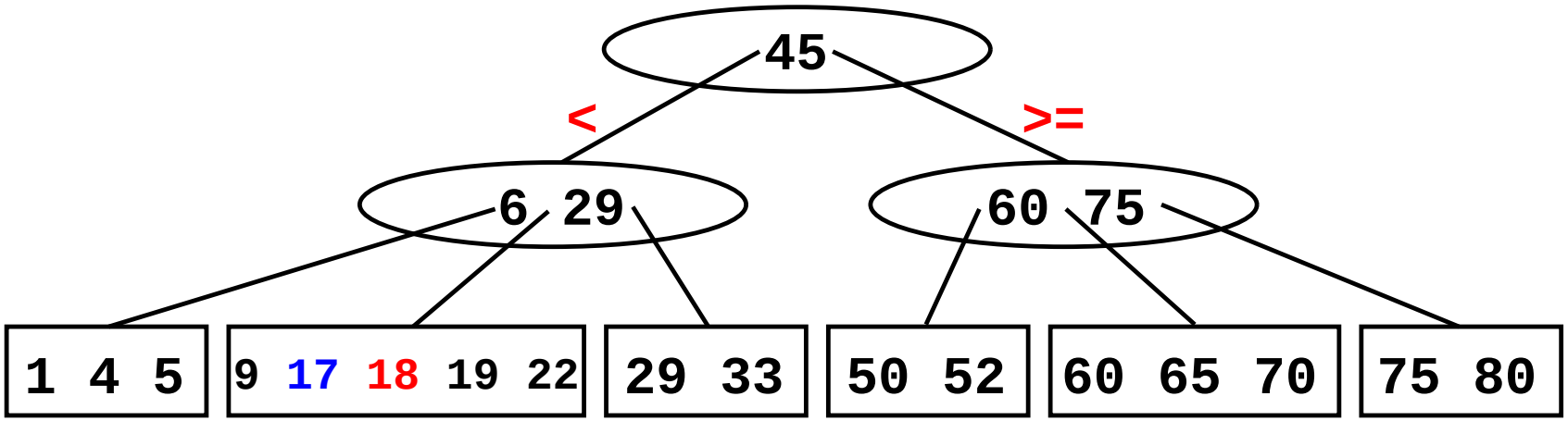
Inserção em Árvores B*

- essencialmente igual à árvores B, porém...
- **na cisão:** algoritmo faz cópia da chave na página pai, ao invés de mover todo registro;
- registro é retido na página folha da direita;
- **Inserindo chave** 17**:**



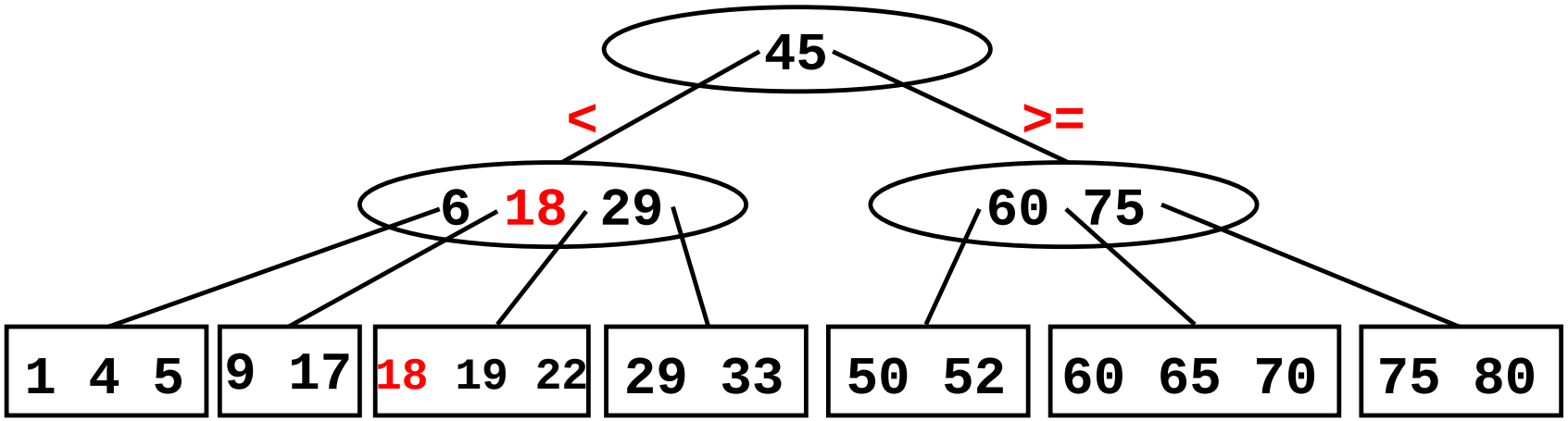
Inserção em Árvores B*

- essencialmente igual à árvores B, porém...
- **na cisão:** algoritmo faz cópia da chave na página pai, ao invés de mover todo registro;
- registro é retido na página folha da direita;
- **Inserindo chave** 17**:**



Inserção em Árvores B*

- essencialmente igual à árvores B, porém...
- **na cisão:** algoritmo faz cópia da chave na página pai, ao invés de mover todo registro;
- registro é retido na página folha da direita;
- **Inserindo chave** 17**:**



Remoção em Árvores B*

Remoção em Árvores B*

- mais simples que em árvores B;

Remoção em Árvores B*

- mais simples que em árvores B;
- registro a ser removido **já está** em folha;

Remoção em Árvores B*

- mais simples que em árvores B;
- registro a ser removido **já está** em folha;
- página folha tem pelo menos *d* registros, logo...

Remoção em Árvores B*

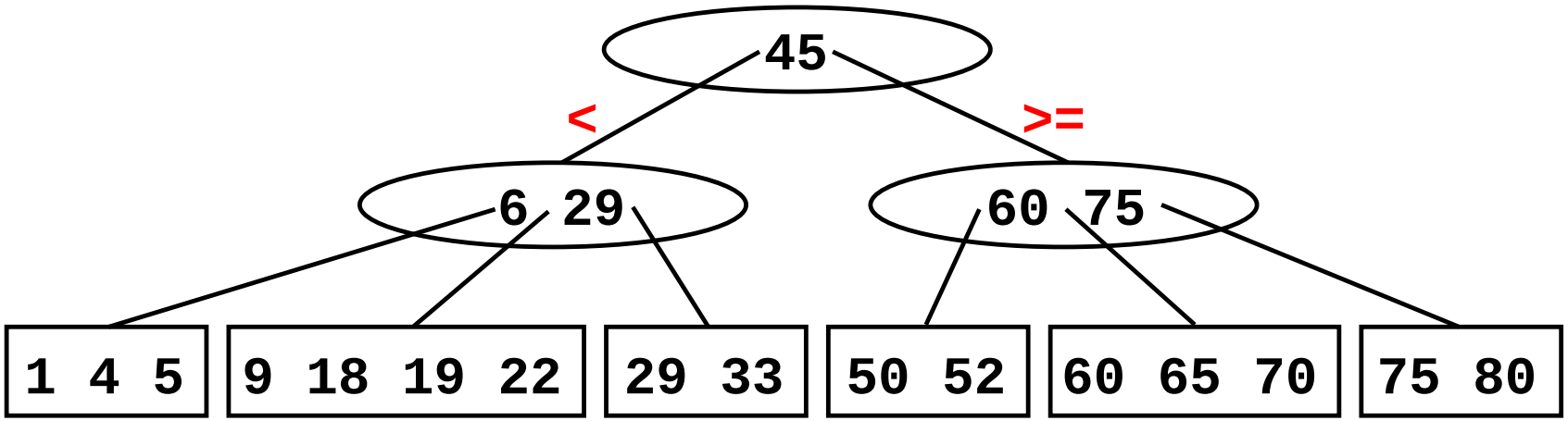
- mais simples que em árvores B;
- registro a ser removido **já está** em folha;
- página folha tem pelo menos *d* registros, logo...
- páginas índice não precisam ser modificadas, mesmo se tem cópia da chave do registro removido;

Remoção em Árvores B*

- mais simples que em árvores B;
- registro a ser removido **já está** em folha;
- página folha tem pelo menos *d* registros, logo...
- páginas índice não precisam ser modificadas, mesmo se tem cópia da chave do registro removido;
- **Removendo** 5**,** 19**,** 22**,** 60**:**

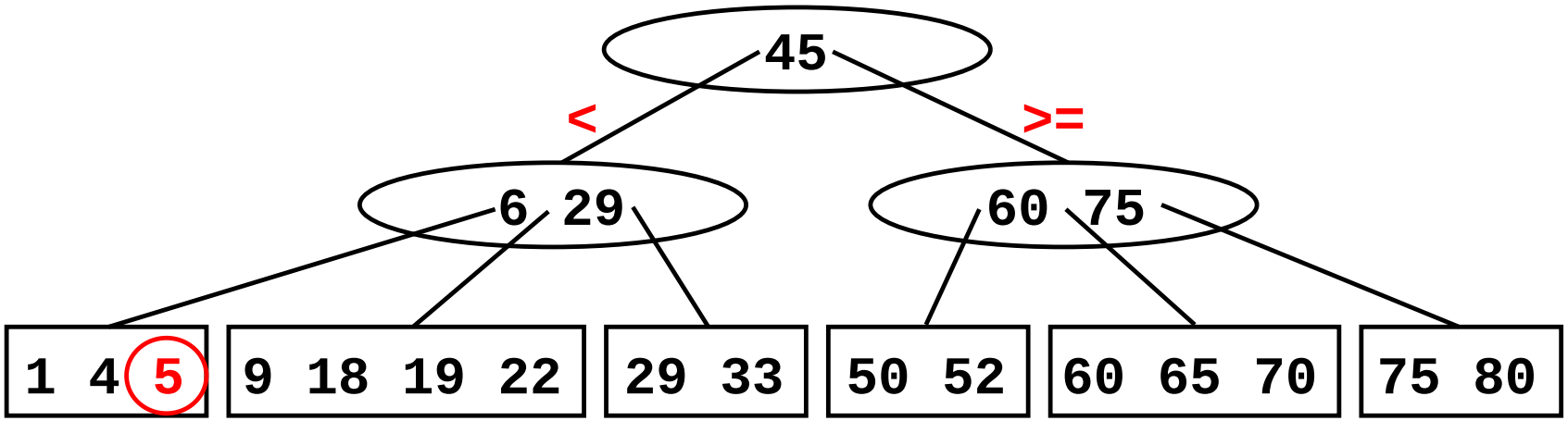
Remoção em Árvores B*

- mais simples que em árvores B;
- registro a ser removido **já está** em folha;
- página folha tem pelo menos *d* registros, logo...
- páginas índice não precisam ser modificadas, mesmo se tem cópia da chave do registro removido;
- **Removendo** 5**,** 19**,** 22**,** 60**:**



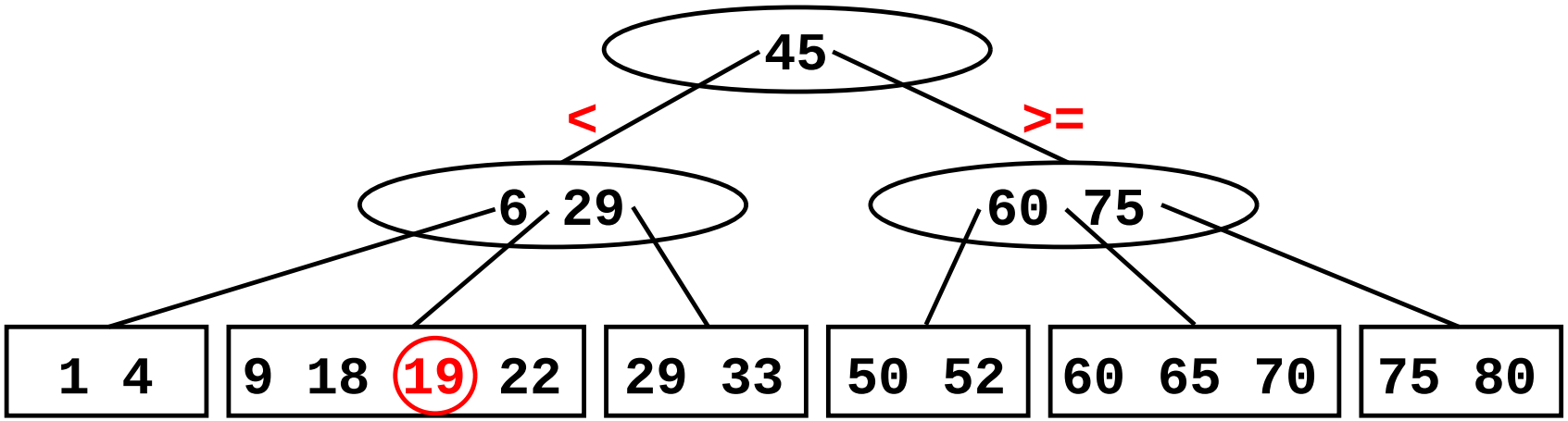
Remoção em Árvores B*

- mais simples que em árvores B;
- registro a ser removido **já está** em folha;
- página folha tem pelo menos *d* registros, logo...
- páginas índice não precisam ser modificadas, mesmo se tem cópia da chave do registro removido;
- **Removendo** 5**,** 19**,** 22**,** 60**:**



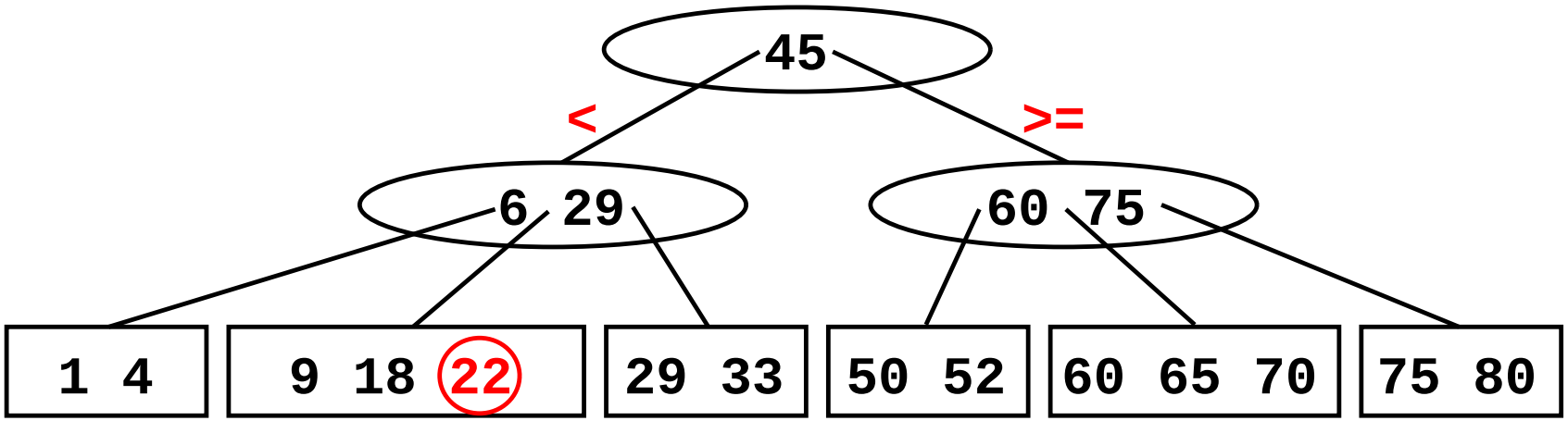
Remoção em Árvores B*

- mais simples que em árvores B;
- registro a ser removido **já está** em folha;
- página folha tem pelo menos *d* registros, logo...
- páginas índice não precisam ser modificadas, mesmo se tem cópia da chave do registro removido;
- **Removendo** 5, 19, 22, 60**:**



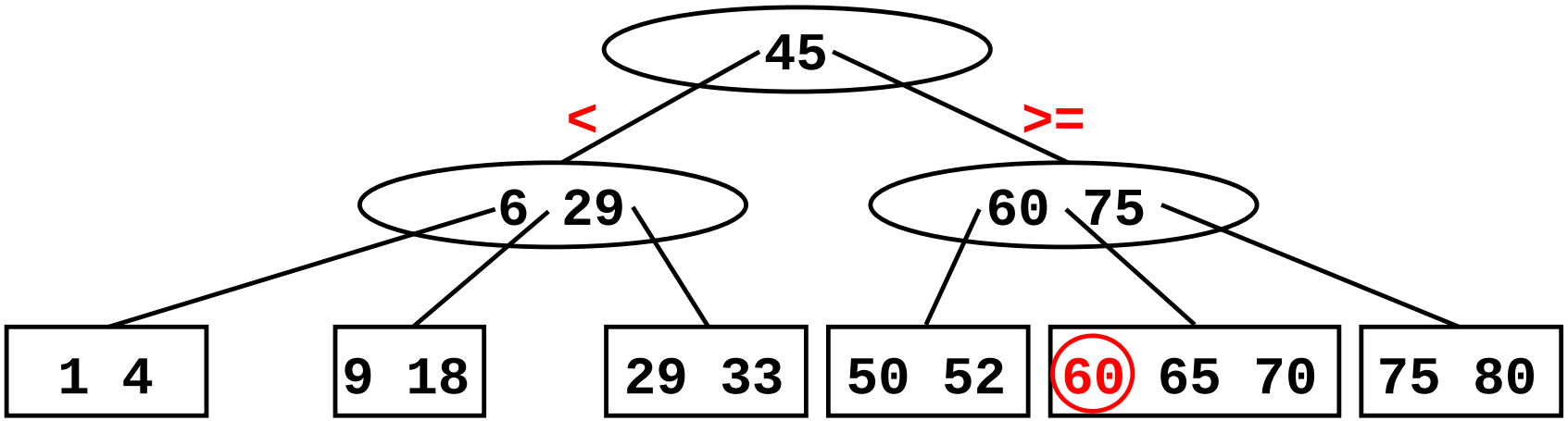
Remoção em Árvores B*

- mais simples que em árvores B;
- registro a ser removido **já está** em folha;
- página folha tem pelo menos *d* registros, logo...
- páginas índice não precisam ser modificadas, mesmo se tem cópia da chave do registro removido;
- **Removendo** 5**,** 19**,** 22**,** 60**:**



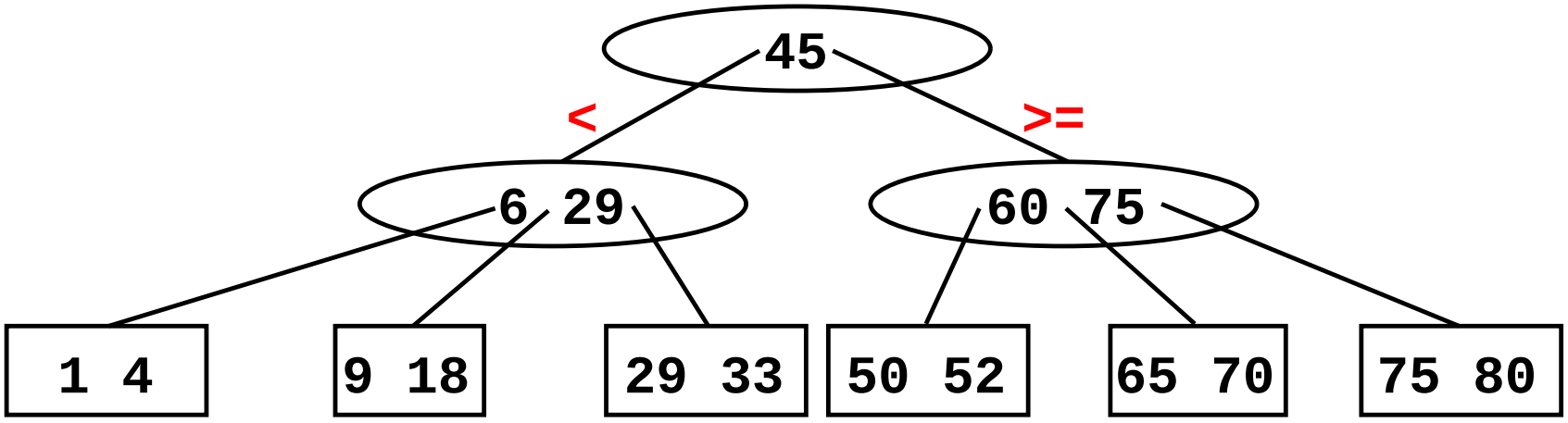
Remoção em Árvores B*

- mais simples que em árvores B;
- registro a ser removido **já está** em folha;
- página folha tem pelo menos *d* registros, logo...
- páginas índice não precisam ser modificadas, mesmo se tem cópia da chave do registro removido;
- **Removendo** 5**,** 19**,** 22**,** 60**:**



Remoção em Árvores B*

- mais simples que em árvores B;
- registro a ser removido **já está** em folha;
- página folha tem pelo menos *d* registros, logo...
- páginas índice não precisam ser modificadas, mesmo se tem cópia da chave do registro removido;
- **Removendo** 5**,** 19**,** 22**,** 60**:**



Remoção em Árvores B*

Remoção em Árvores B*

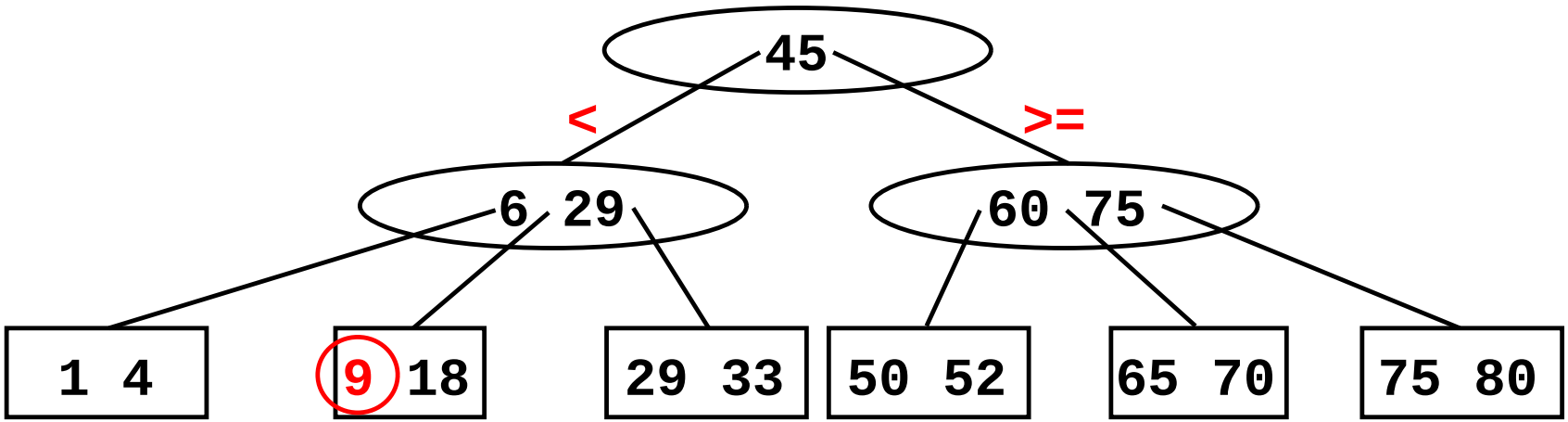
- Atenção à concatenação e redistribuição de chaves em páginas índice e páginas folha!

Remoção em Árvores B*

- Atenção à concatenação e redistribuição de chaves em páginas índice e páginas folha!
- **Removendo 9:**

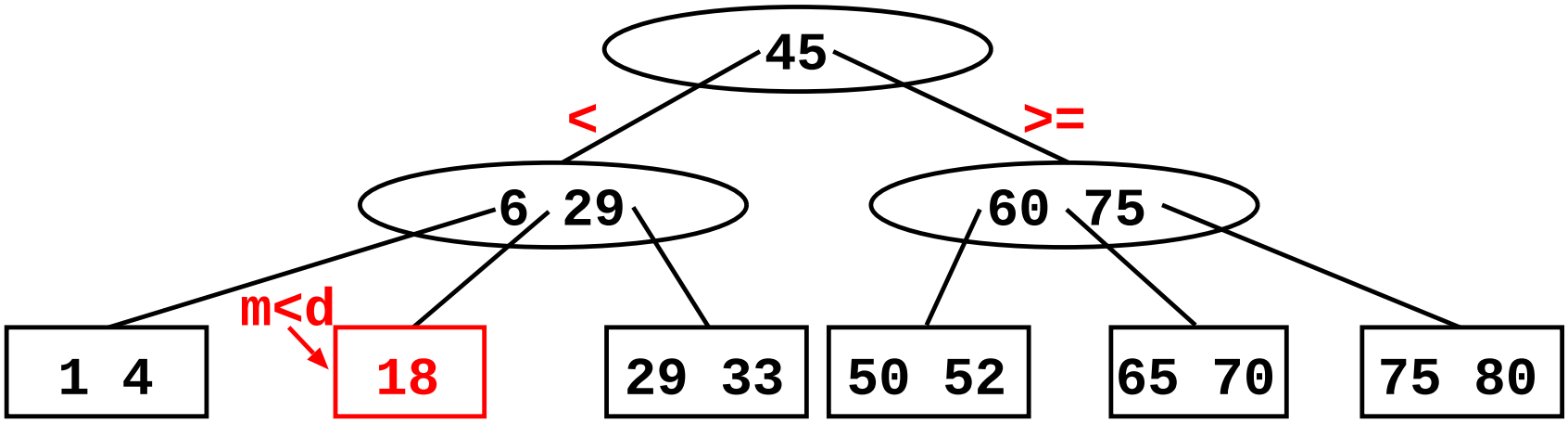
Remoção em Árvores B*

- Atenção à concatenação e redistribuição de chaves em páginas índice e páginas folha!
- **Removendo 9:**



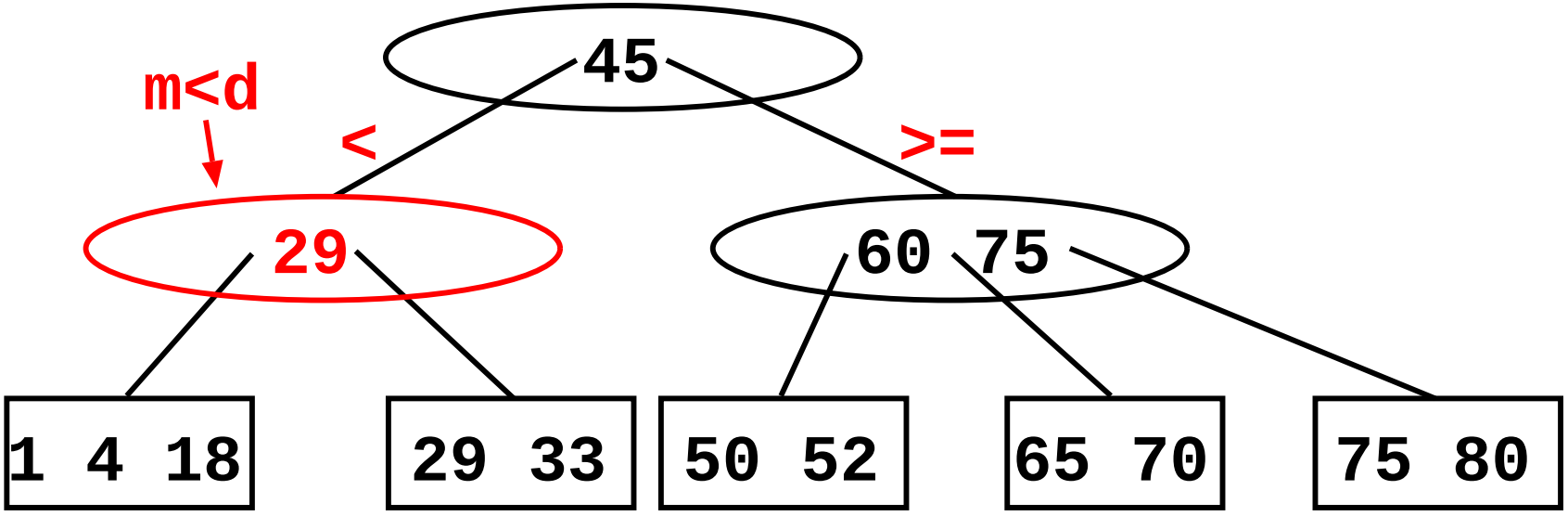
Remoção em Árvores B*

- Atenção à concatenação e redistribuição de chaves em páginas índice e páginas folha!
- **Removendo 9:**



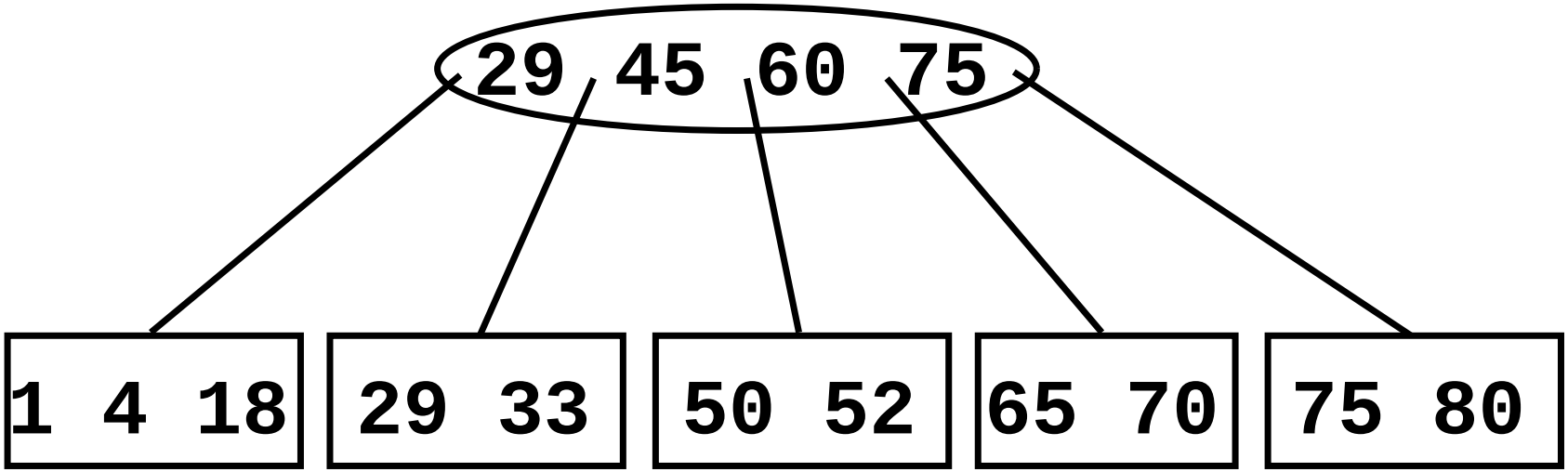
Remoção em Árvores B*

- Atenção à concatenação e redistribuição de chaves em páginas índice e páginas folha!
- **Removendo 9:**



Remoção em Árvores B*

- Atenção à concatenação e redistribuição de chaves em páginas índice e páginas folha!
- **Removendo 9:**



Acesso Concorrente em Árvores B*

Acesso Concorrente em Árvores B*

- acesso simultâneo ao banco de dados;

Acesso Concorrente em Árvores B*

- acesso simultâneo ao banco de dados;
- **gargalo:** se é um processo por vez acessando BD;

Acesso Concorrente em Árvores B*

- acesso simultâneo ao banco de dados;
- **gargalo:** se é um processo por vez acessando BD;
- **protocolos:** para garantir integridade dos dados e da estrutura;

Acesso Concorrente em Árvores B*

- acesso simultâneo ao banco de dados;
- **gargalo:** se é um processo por vez acessando BD;
- **protocolos:** para garantir integridade dos dados e da estrutura;
- **problema:** um processo busca e outro insere ao mesmo tempo, modificando estrutura da árvore;

Acesso Concorrente em Árvores B*

- acesso simultâneo ao banco de dados;
- **gargalo:** se é um processo por vez acessando BD;
- **protocolos:** para garantir integridade dos dados e da estrutura;
- **problema:** um processo busca e outro insere ao mesmo tempo, modificando estrutura da árvore;
- **resultado:** processo que busca pode obter ponteiro para subárvore errada ou para endereço inexistente;

Acesso Concorrente em Árvores B*

- acesso simultâneo ao banco de dados;
- **gargalo:** se é um processo por vez acessando BD;
- **protocolos:** para garantir integridade dos dados e da estrutura;
- **problema:** um processo busca e outro insere ao mesmo tempo, modificando estrutura da árvore;
- **resultado:** processo que busca pode obter ponteiro para subárvore errada ou para endereço inexistente;
- **página segura:** não tem possibilidade de modificar estrutura da árvore;

Acesso Concorrente em Árvores B*

- acesso simultâneo ao banco de dados;
- **gargalo:** se é um processo por vez acessando BD;
- **protocolos:** para garantir integridade dos dados e da estrutura;
- **problema:** um processo busca e outro insere ao mesmo tempo, modificando estrutura da árvore;
- **resultado:** processo que busca pode obter ponteiro para subárvore errada ou para endereço inexistente;
- **página segura:** não tem possibilidade de modificar estrutura da árvore;
  1. **busca:** não modifica;

Acesso Concorrente em Árvores B*

- acesso simultâneo ao banco de dados;
- **gargalo:** se é um processo por vez acessando BD;
- **protocolos:** para garantir integridade dos dados e da estrutura;
- **problema:** um processo busca e outro insere ao mesmo tempo, modificando estrutura da árvore;
- **resultado:** processo que busca pode obter ponteiro para subárvore errada ou para endereço inexistente;
- **página segura:** não tem possibilidade de modificar estrutura da árvore;
  1. **busca:** não modifica;
  2. **inserção:** se $m < 2d$;

Acesso Concorrente em Árvores B*

- acesso simultâneo ao banco de dados;
- **gargalo:** se é um processo por vez acessando BD;
- **protocolos:** para garantir integridade dos dados e da estrutura;
- **problema:** um processo busca e outro insere ao mesmo tempo, modificando estrutura da árvore;
- **resultado:** processo que busca pode obter ponteiro para subárvore errada ou para endereço inexistente;
- **página segura:** não tem possibilidade de modificar estrutura da árvore;
  1. **busca:** não modifica;
  2. **inserção:** se $m < 2d$;
  3. **remoção:** se $m > d$;

Acesso Concorrente em Árvores B*

Acesso Concorrente em Árvores B*

- **protocolo para processos leitores:**

Acesso Concorrente em Árvores B*

- **protocolo para processos leitores:**
  1. coloque travamento para leitura na raiz;

Acesso Concorrente em Árvores B*

- **protocolo para processos leitores:**
  1. coloque travamento para leitura na raiz;
  2. leia página raiz e faça-a página corrente;

Acesso Concorrente em Árvores B*

- **protocolo para processos leitores:**
  1. coloque travamento para leitura na raiz;
  2. leia página raiz e faça-a página corrente;
  3. enquanto página corrente não é folha:

Acesso Concorrente em Árvores B*

- **protocolo para processos leitores:**
  1. coloque travamento para leitura na raiz;
  2. leia página raiz e faça-a página corrente;
  3. enquanto página corrente não é folha:
     - 3.1 coloque travamento para leitura no descendente apropriado;

Acesso Concorrente em Árvores B*

- **protocolo para processos leitores:**
  1. coloque travamento para leitura na raiz;
  2. leia página raiz e faça-a página corrente;
  3. enquanto página corrente não é folha:
     - 3.1 coloque travamento para leitura no descendente apropriado;
     - 3.2 libere travamento para leitura na página corrente;

Acesso Concorrente em Árvores B*

- **protocolo para processos leitores:**
  1. coloque travamento para leitura na raiz;
  2. leia página raiz e faça-a página corrente;
  3. enquanto página corrente não é folha:
     - 3.1 coloque travamento para leitura no descendente apropriado;
     - 3.2 libere travamento para leitura na página corrente;
     - 3.3 leia descendente e faça-a página corrente;

Acesso Concorrente em Árvores B*

Acesso Concorrente em Árvores B*

- **protocolo para processos modificadores:**

Acesso Concorrente em Árvores B*

- **protocolo para processos modificadores:**
  1. coloque travamento exclusivo na raiz;

Acesso Concorrente em Árvores B*

- **protocolo para processos modificadores:**
  1. coloque travamento exclusivo na raiz;
  2. leia página raiz e faça-a página corrente;

Acesso Concorrente em Árvores B*

- **protocolo para processos modificadores:**
  1. coloque travamento exclusivo na raiz;
  2. leia página raiz e faça-a página corrente;
  3. enquanto página corrente não é folha:

Acesso Concorrente em Árvores B*

- **protocolo para processos modificadores:**
  1. coloque travamento exclusivo na raiz;
  2. leia página raiz e faça-a página corrente;
  3. enquanto página corrente não é folha:
     - 3.1 coloque travamento exclusivo no descendente apropriado;

Acesso Concorrente em Árvores B*

- **protocolo para processos modificadores:**
  1. coloque travamento exclusivo na raiz;
  2. leia página raiz e faça-a página corrente;
  3. enquanto página corrente não é folha:
     - **3.1** coloque travamento exclusivo no descendente apropriado;
     - **3.2** leia descendente e faça-a página corrente;

Acesso Concorrente em Árvores B*

- **protocolo para processos modificadores:**
  1. coloque travamento exclusivo na raiz;
  2. leia página raiz e faça-a página corrente;
  3. enquanto página corrente não é folha:
     - **3.1** coloque travamento exclusivo no descendente apropriado;
     - **3.2** leia descendente e faça-a página corrente;
     - **3.3** se página corrente é segura, libere todos travamentos sobre páginas antecessoras;

Exercícios Propostos

1. Desenhe uma árvore B* de ordem 3 que contenha as seguintes chaves: 2, 4, 7, 9, 15, 33, 37, 39, 40, 42, 45.

Exercícios Propostos

1. Desenhe uma árvore B* de ordem 3 que contenha as seguintes chaves: 2, 4, 7, 9, 15, 33, 37, 39, 40, 42, 45.
2. Dê exemplo de uma cisão de página em árvores B* que se propaga até à raiz.

Exercícios Propostos

1. Desenhe uma árvore B* de ordem 3 que contenha as seguintes chaves: 2, 4, 7, 9, 15, 33, 37, 39, 40, 42, 45.
2. Dê exemplo de uma cisão de página em árvores B* que se propaga até à raiz.
3. Dê exemplo de uma concatenação em árvores B* que se propaga até à raiz.

Exercícios Propostos

1. Desenhe uma árvore B* de ordem 3 que contenha as seguintes chaves: 2, 4, 7, 9, 15, 33, 37, 39, 40, 42, 45.
2. Dê exemplo de uma cisão de página em árvores B* que se propaga até à raiz.
3. Dê exemplo de uma concatenação em árvores B* que se propaga até à raiz.
4. Dê exemplo de uma redistribuição em árvores B*.

Exercícios Propostos

1. Desenhe uma árvore B* de ordem 3 que contenha as seguintes chaves: 2, 4, 7, 9, 15, 33, 37, 39, 40, 42, 45.
2. Dê exemplo de uma cisão de página em árvores B* que se propaga até à raiz.
3. Dê exemplo de uma concatenação em árvores B* que se propaga até à raiz.
4. Dê exemplo de uma redistribuição em árvores B*.
5. Escreva o algoritmo de inserção em uma árvore B*.

Exercícios Propostos

1. Desenhe uma árvore B* de ordem 3 que contenha as seguintes chaves: 2, 4, 7, 9, 15, 33, 37, 39, 40, 42, 45.
2. Dê exemplo de uma cisão de página em árvores B* que se propaga até à raiz.
3. Dê exemplo de uma concatenação em árvores B* que se propaga até à raiz.
4. Dê exemplo de uma redistribuição em árvores B*.
5. Escreva o algoritmo de inserção em uma árvore B*.
6. Escreva o algoritmo de remoção em uma árvore B*.

Exercícios Propostos

1. Desenhe uma árvore B* de ordem 3 que contenha as seguintes chaves: 2, 4, 7, 9, 15, 33, 37, 39, 40, 42, 45.
2. Dê exemplo de uma cisão de página em árvores B* que se propaga até à raiz.
3. Dê exemplo de uma concatenação em árvores B* que se propaga até à raiz.
4. Dê exemplo de uma redistribuição em árvores B*.
5. Escreva o algoritmo de inserção em uma árvore B*.
6. Escreva o algoritmo de remoção em uma árvore B*.
7. Escreva o algoritmo de busca em uma árvore B*.

Bibliografia Utilizada

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