

Algoritmos e Estruturas de Dados III

Aula 2.1 - Dispositivos de Armazenamento

Prof. Hayala Curto
2022



PUC Minas

Roteiro do Conteúdo



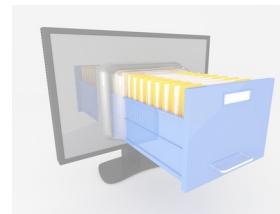
2.1 Dispositivos de Armazenamento

- Memória primária x Memória secundária
- Disco Rígido x SSD
- Estrutura dos discos rígidos
- Estrutura dos SSD
- Tempo de operações

2.2 Manipulando a Memória secundária

- Arquivos
- Fluxos de entrada e saída
 - Arquivos como vetores de bytes
 - CRUD
 - Implementando em JAVA

Roteiro do Conteúdo



2.1 Dispositivos de Armazenamento

- Memória primária x Memória secundária
- Disco Rígido x SSD
- Estrutura dos discos rígidos
- Estrutura dos SSD
- Tempo de operações

2.2 Manipulando a Memória secundária

- Arquivos
- Fluxos de entrada e saída
 - Arquivos como vetores de bytes
 - CRUD
 - Implementando em JAVA

Roteiro da Aula

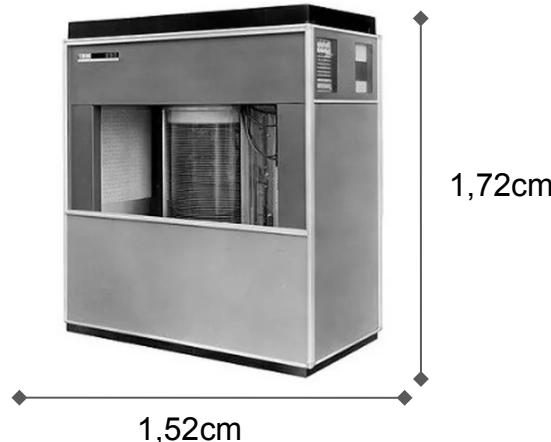
- Dispositivos de Armazenamento
 - **Memória primária x Memória secundária**
 - Disco Rígido x SSD
 - Estrutura dos discos rígidos
 - Estrutura dos SSD
 - Tempo de operações

Memória Secundária

A contínua evolução do hardware tem permitido que as memórias sejam cada vez:

- menores, mais rápidas, de maior capacidade e mais baratas.

RAMAC 305 - 1º HD - 1956
5 MB



Dispositivos atuais



Memória Primária x Memória Secundária

Memória primária ou principal



Memória secundária



- memória de alta velocidade
- tempo de acesso muito baixo
- custo por byte alto
- São memórias voláteis

- mais lenta que a primária
- tempo de acesso mais alto
- custo por byte mais baixo
- São memórias não voláteis

Memória Primária x Memória Secundária

Memória primária ou principal



Memória secundária



O custo do GB em memória RAM é
aproximadamente 140 vezes superior ao custo do
GB em um HDD ou SSD

Vamos calcular?

16 GB

- Preço aproximado: R\$ 450,00

HD - 2 TB

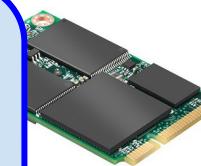
- Preço aproximado: R\$ 350,00

Memória Primária x Memória Secundária

Memória primária ou principal



Memória secundária



O custo do GB em memória RAM é
aproximadamente 140 vezes superior ao custo do
GB em um HDD ou SSD

Vamos calcular?

16 GB

- Preço aproximado: R\$ 450,00

SSD - 480 GB

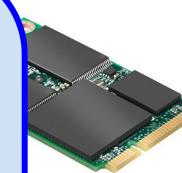
- Preço aproximado: R\$ 260,00

Memória Primária x Memória Secundária

Memória primária ou principal



Memória secundária



Quando um determinado sistema requer o acesso e/ou a manipulação de um grande volume de dados, que não cabem na memória primária, ele deve usar uma memória secundária para armazenamento desses dados.

- memória de alta velocidade
- tempo de acesso menor
- custo por byte maior
- São memórias voláteis

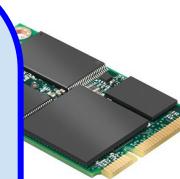
- memória de baixa velocidade
- tempo de acesso maior
- custo por byte menor
- São memórias não voláteis

Memória Primária x Memória Secundária

Memória primária ou principal



Memória secundária



O disco rígido gasta muito mais tempo para acessar os dados que a memória principal.

Além de serem dispositivos mais lentos, eles não são acessados diretamente pela CPU, mas por meio de algum tipo de interface.

Assim, precisamos escrever algoritmos específicos para armazenamento

- memória de alta velocidade
- tempo de acesso menor
- custo por byte menor
- São memórias voláteis

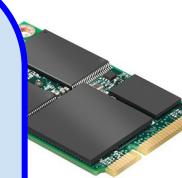
e a primária
tem custo mais alto
tempo de acesso mais baixo
e é de armazenamento não voláteis

Memória Primária x Memória Secundária

Memória primária ou principal



Memória secundária



O disco rígido gasta muito mais tempo para acessar os dados que a memória principal.

Além de serem dispositivos mais lentos, eles não são acessados diretamente pela CPU, mas por meio de algum tipo de interface.

- memória de alta velocidade
- tempo de acesso menor
- custo por byte maior
- São memórias voláteis

e a primária
tempo de acesso mais alto
custo por byte mais baixo
São memórias não voláteis

Roteiro da Aula

- Dispositivos de Armazenamento
 - Memória primária x Memória secundária
 - **Disco Rígido x SSD**
 - Estrutura dos discos rígidos
 - Estrutura dos SSD
 - Tempo de operações

Disco Rígido



Possuem partes mecânicas. Essas peças, que precisam se mover com o apoio de motores, tornam o dispositivo bem mais lento do que as placas de memória.

Não há contato entre os cabeçotes e as superfícies do disco, para evitar problemas de choque entre eles. Uma fina camada de ar os separa.

SSD

Os SSDs mais comuns no mercado possuem dois componentes fundamentais:

Memória flash: guarda todos os dados e todas as operações são feitas eletricamente.

Controlador: gerencia a troca de dados entre o computador e a memória flash. Formado por um processador que executa diversas tarefas no drive, é um dos principais responsáveis pela performance de um SSD.





SSD

vs



HDD

mais rápido



mais lento

menor duração



maior duração

mais caro



mais econômico

não mecânico (flash)



mecânico (partes móveis)

resistente a impactos



frágil

indicado para o
armazenamento de sistemas
operacionais, jogos e
arquivos de uso frequente

indicado para armazenar
dados adicionais, como
filmes, fotos e documentos

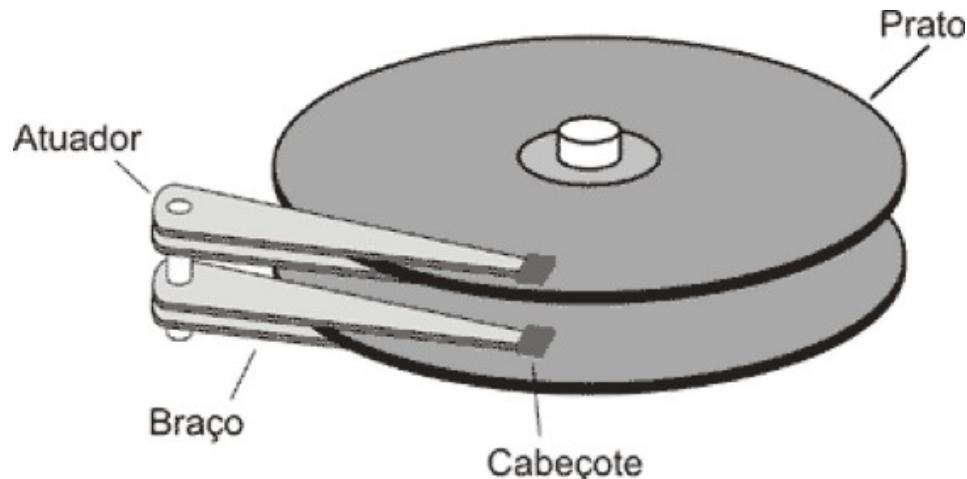
Roteiro da Aula

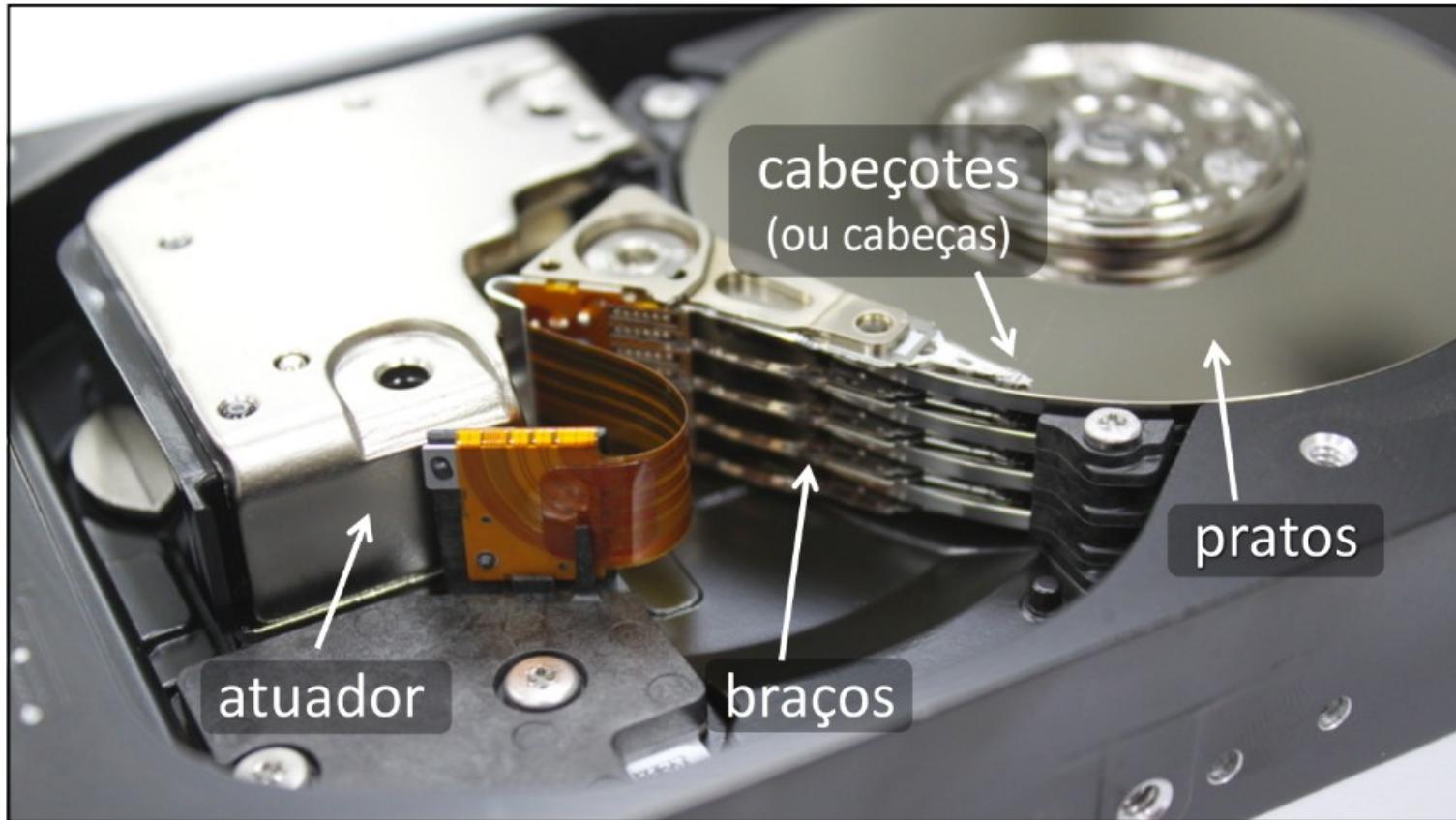
- Dispositivos de Armazenamento
 - Memória primária x Memória secundária
 - Disco Rígido x SSD
 - **Estrutura dos discos rígidos**
 - Estrutura dos SSD
 - Tempo de operações

Disco Rígido

O disco rígido é composto por:

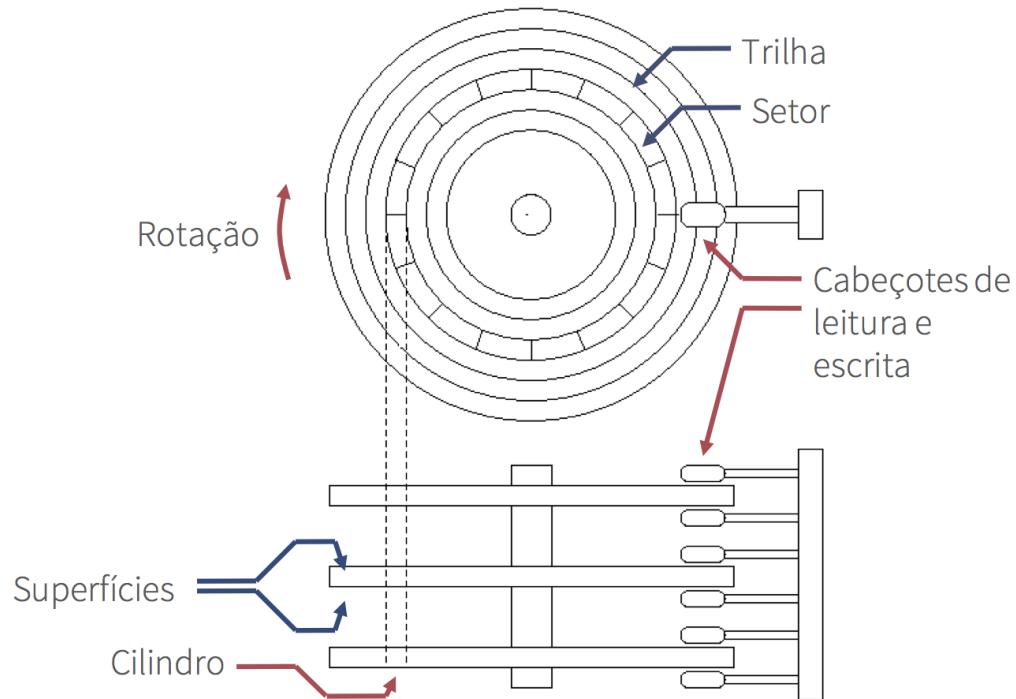
- Pratos
- Cabeçotes
- Braços
- Atuador





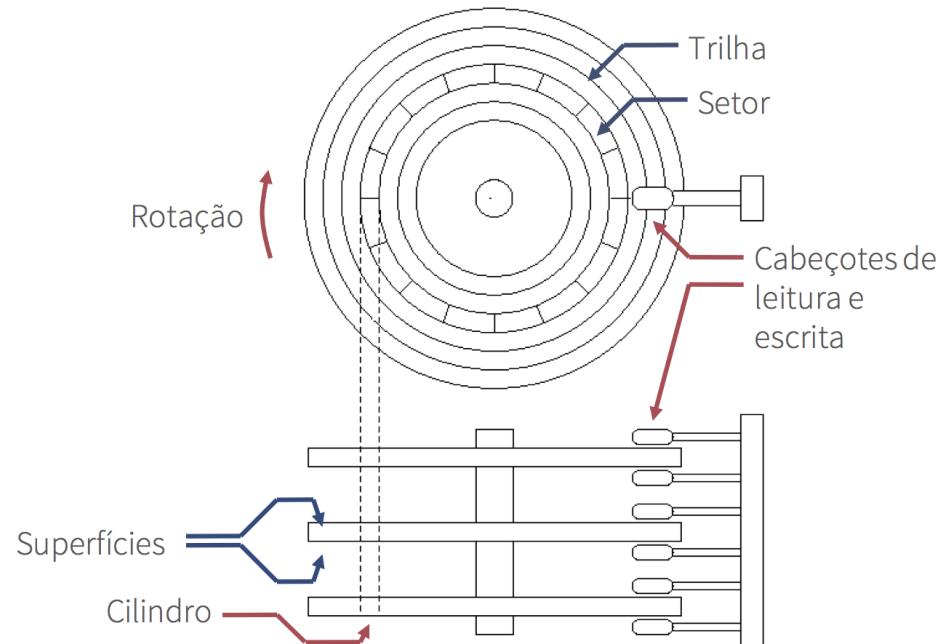
Geometria do Disco Rígido

Cada prato de um disco rígido é dividido em anéis concêntricos chamados de trilhas. Cada trilha é composta por vários setores, nos quais as informações são gravadas.



Leitura e escrita no Disco Rígido

A leitura ou escrita são feitos através da transferência dos dados armazenados nas trilhas já acessíveis em todas as superfícies, antes de qualquer movimento mecânico.

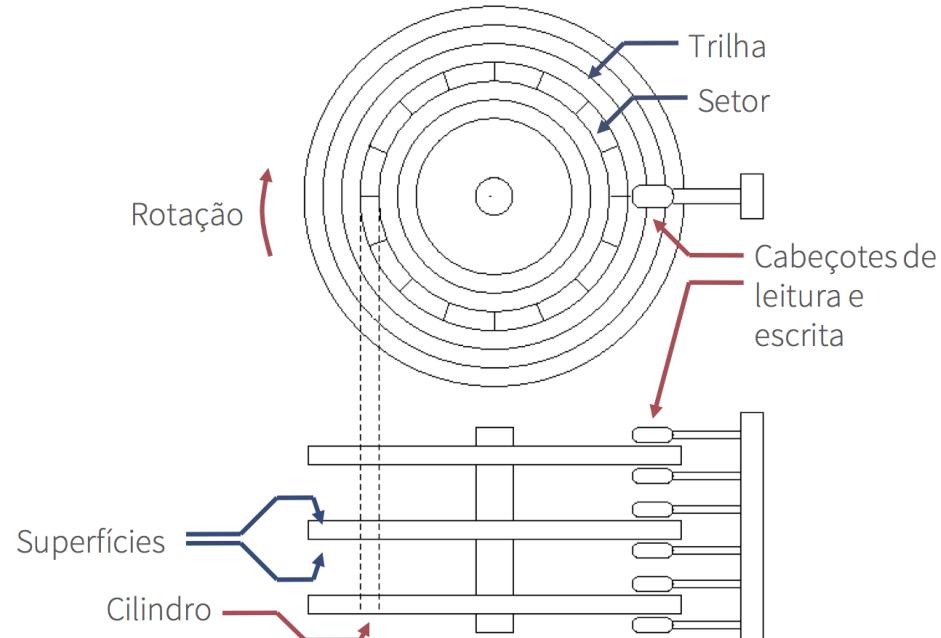


Leitura e escrita no Disco Rígido

Os cabeçotes são movimentados juntos.

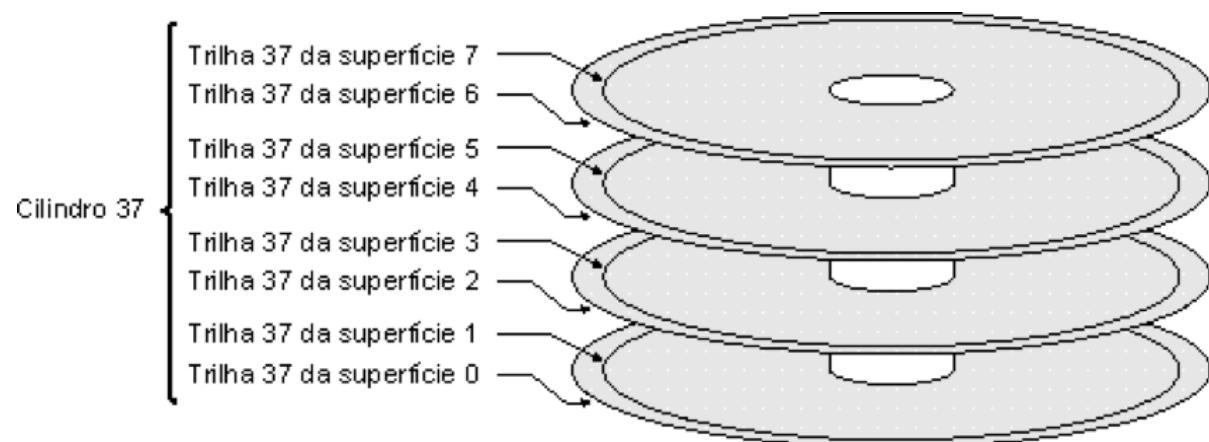
Quando o cabeçote da primeira superfície estiver posicionado em uma trilha, todos os outros cabeçotes também estarão nesse mesmo cilindro.

Por causa dessa forma de operação, o conjunto de trilhas de mesma posição é chamado de cilindro.



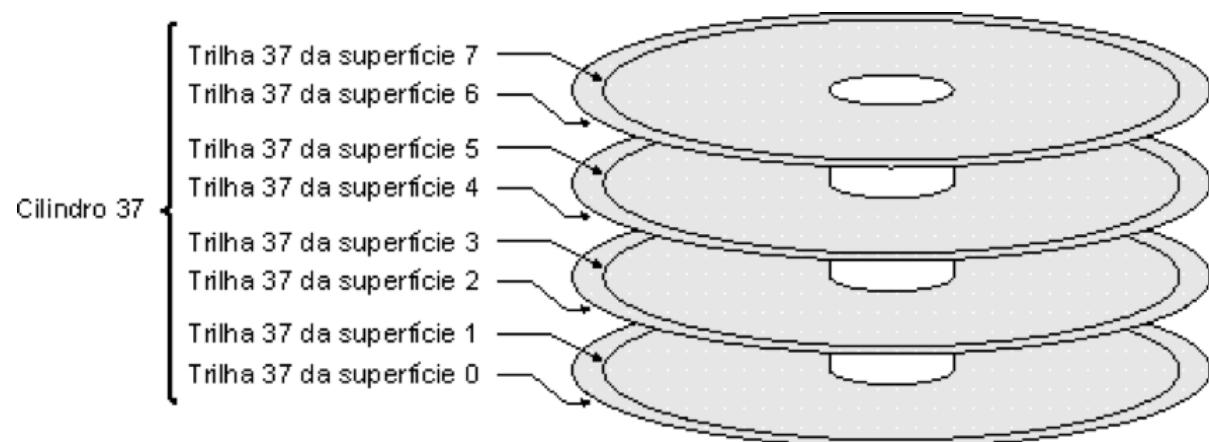
Leitura e escrita no Disco Rígido

O cilindro 37, por exemplo, é composto por todas as trilhas 37 das superfícies.



Leitura e escrita no Disco Rígido

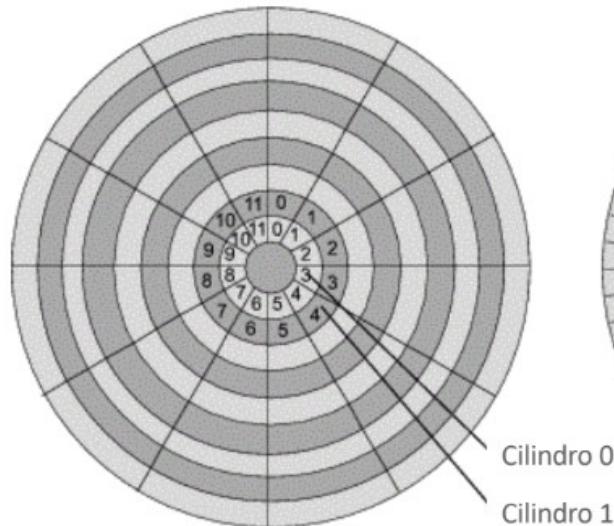
O cilindro 37, por exemplo, é composto por todas as trilhas 37 das superfícies.



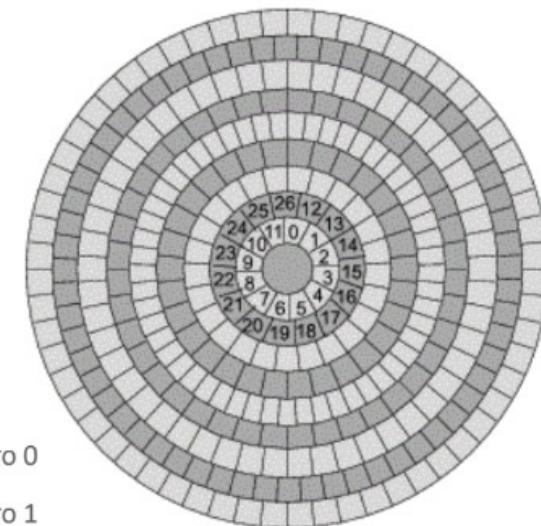
Leitura e escrita no Disco Rígido

Um cilindro não existe fisicamente, é usado na forma de endereçamento de setores conhecida como CHS (Cylinders, Heads and Sectors).

Essa forma de endereçamento deixou de ser usada passando apenas por um número sequencial.



Endereçamento CHS

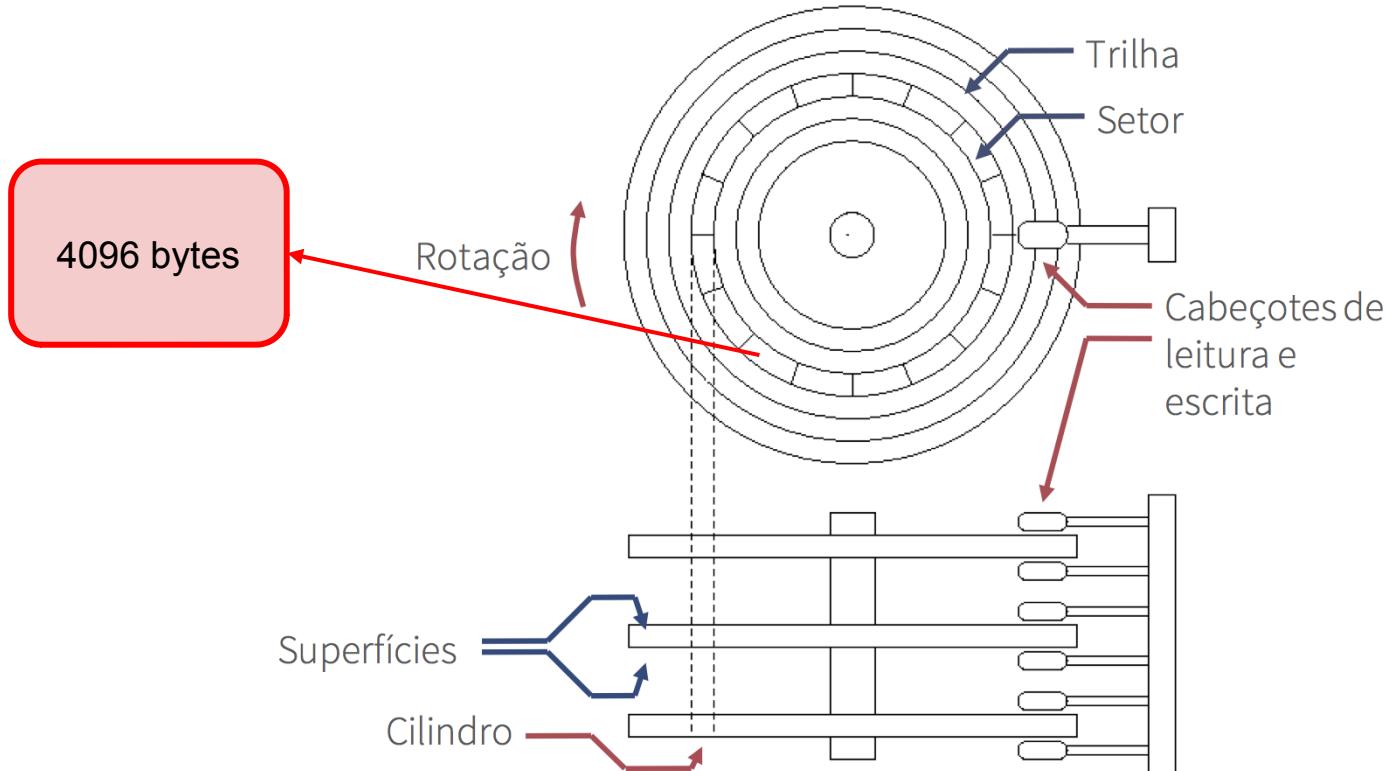


Endereçamento LBA

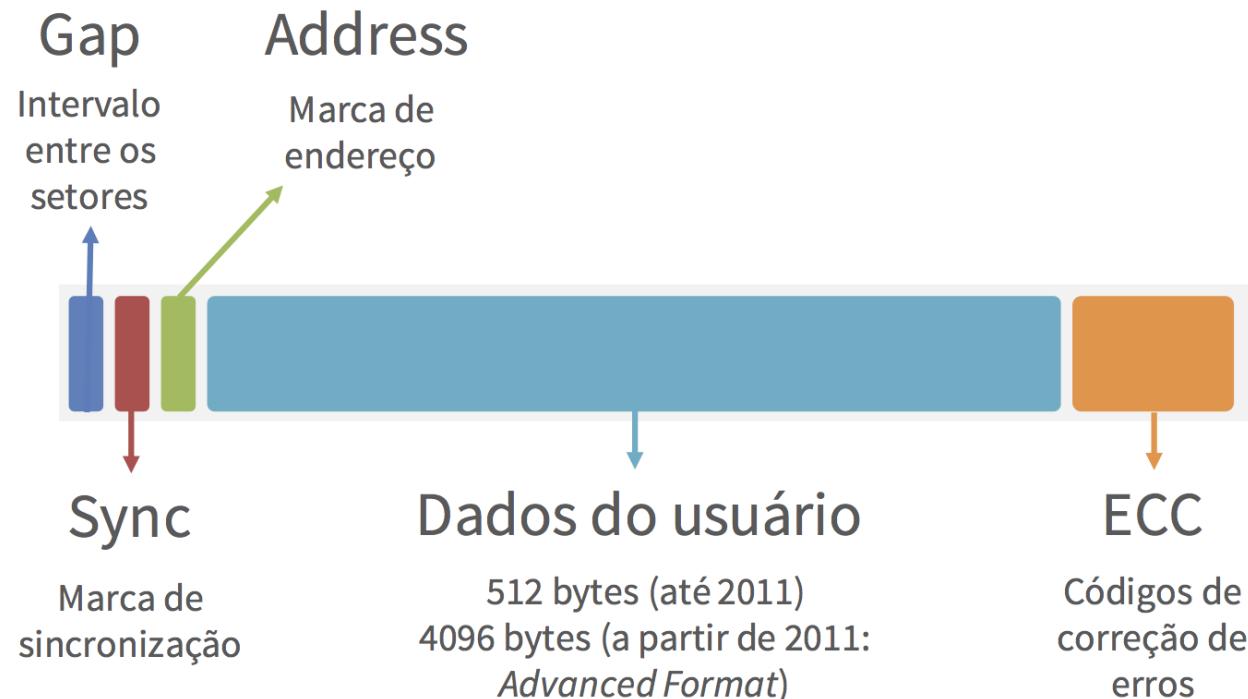
Leitura e escrita nos discos rígidos

Até 2011, em muitos HDs, os setores tinham 512 bytes de espaço para armazenamento de dados. A partir de 2011, os HDs passaram a adotar setores de 4096 bytes, em uma estrutura conhecida como Advanced Format. Tipicamente, os setores contém a seguinte estrutura:

- Identificação – localização física do setor (número, cilindro e prato).
- Campos de controle – dados utilizados internamente para orientação no processo de leitura.
- Dados – dados do usuário.
- ECC (Error Correction Code) – código para correção de bits lidos incorretamente. Esse código é gerado na gravação dos dados.
- Espaço (Gap) – espaço entre setores para dar tempo à controladora processar o que já leu e também para diminuir a necessidade de precisão nas leituras.
- Essa estrutura pode ser vista na figura abaixo.



Leitura e escrita no Disco Rígido



Leitura e escrita no Disco Rígido

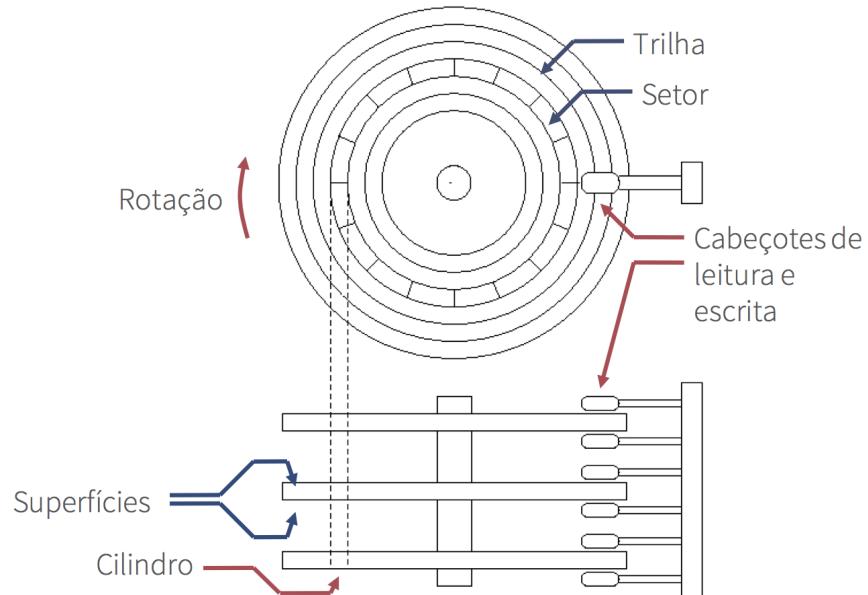
O campo Identificação também é usado para indicar se o setor é defeituoso ou se ele foi remapeado. Setores remapeados são setores defeituosos que foram substituídos por setores reserva para manter a capacidade total do disco. Todo disco rígido moderno é fabricado com vários setores reserva disponíveis para essas substituições e esse processo é transparente para o usuário.

Capacidade de um disco rígido

Na antiga forma de endereçamento por CHS (Cylinders, Heads and Sectors), calculamos a capacidade do disco rígido como:

- Tamanho do setor = 512 bytes
- Tamanho da trilha = tamanho do setor * número de setores por trilha
- Tamanho do cilindro = tamanho da trilha * número de trilhas por cilindro
- Tamanho do disco = tamanho do cilindro * número de cilindros

O número de trilhas por cilindro é o mesmo número de superfícies. Como há um cabeçote por superfície, esse valor também poderia ser especificado como o número de cabeçotes.

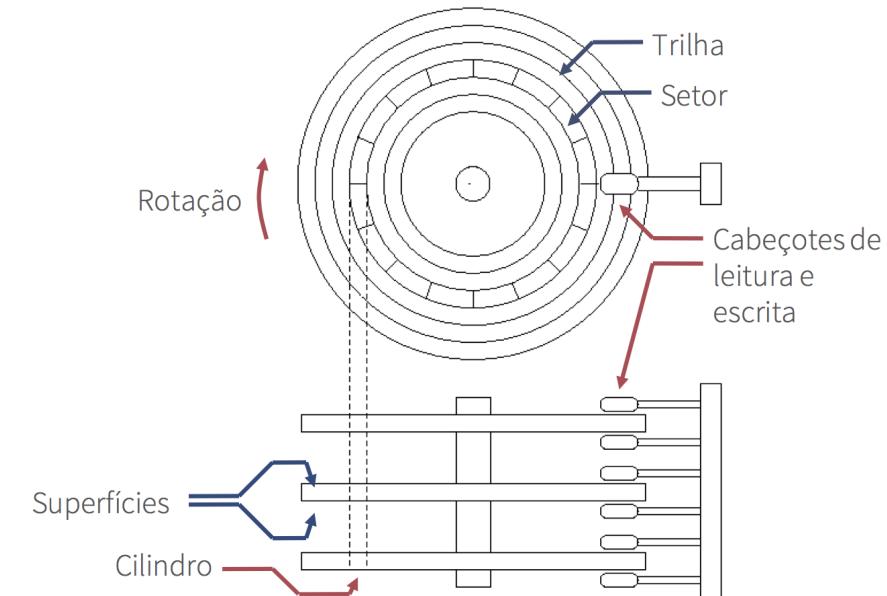


Capacidade de um disco rígido

Na antiga forma de endereçamento por CHS (Cylinders, Heads and Sectors), calculamos a capacidade do disco rígido como:

- Tamanho do setor = 512 bytes
- Tamanho da trilha = tamanho do setor * número de setores por trilha
- Tamanho do cilindro = tamanho da trilha * número de trilhas por cilindro
- Tamanho do disco = tamanho do cilindro * número de cilindros

E se considerarmos, por exemplo, um disco cuja geometria seja 63 setores por trilha, 16 cabeçotes e 1060 cilindros, teríamos?

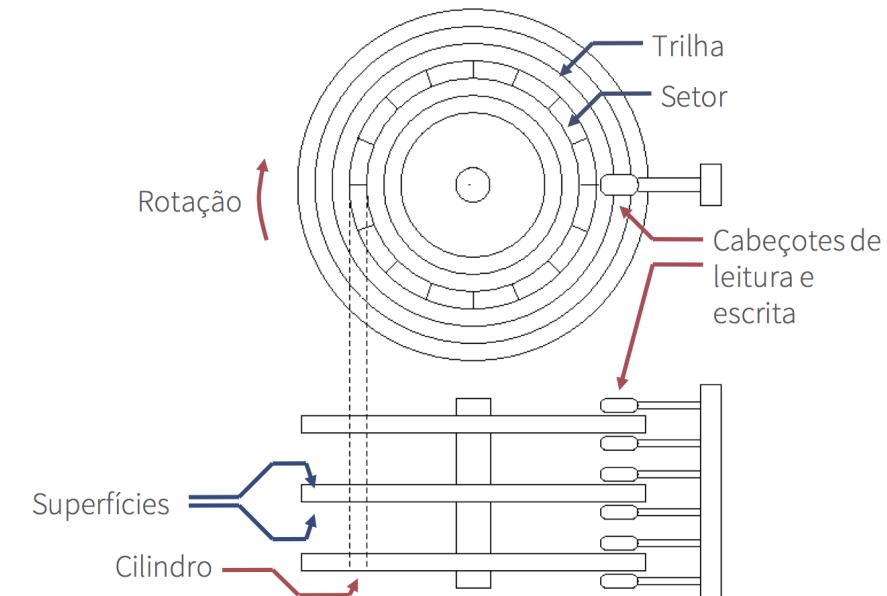


Capacidade de um disco rígido

Na antiga forma de endereçamento por CHS (Cylinders, Heads and Sectors), calculamos a capacidade do disco rígido como:

- Tamanho do setor = 512 bytes
- Tamanho da trilha = tamanho do setor * número de setores por trilha = **32.256 B**
- Tamanho do cilindro = tamanho da trilha * número de trilhas por cilindro = **516.096 B**
- Tamanho do disco = tamanho do cilindro * número de cilindros = **547.061.760 B**

E se considerarmos, por exemplo, um disco cuja geometria seja 63 setores por trilha, 16 cabeçotes e 1060 cilindros, teríamos?



Endereçamento de Blocos Lógicos

O modo de endereçamento chamado de LBA (Logical Block Addressing) se tornou o padrão utilizado por fabricantes de discos há vários anos. O LBA endereça os setores através de um número sequencial, iniciado em zero. Assim, cada setor possui um número único, independente do cilindro, da superfície ou da posição na trilha em que esteja.

É possível fazer um mapeamento entre os modos de endereçamento CHS e LBA, por meio da seguinte fórmula:

$$\text{LBA} = (C \times \text{MAX_H} + H) \times \text{MAX_S} + (S - 1)$$

C, H e S são, respectivamente, cilindros (C), cabeçotes (H) e setores (S),

MAX_H é o número máximo de cabeçotes por cilindro e

MAX_S é o número máximo de setores por trilha.

Endereçamento de Blocos Lógicos

O disco rígido é um gargalo do computador e é o seu desempenho que define, de uma forma geral, o desempenho geral dos sistemas (já que programas e dados estão armazenados em disco).

Obviamente, esse impacto é maior nos programas que acessam mais arquivos em disco, pois a CPU executa mais de um milhão de instruções enquanto o disco rígido executa um único acesso aleatório.



Tempo da operação em disco



Tempo de acesso

Tempo necessário para o cabeçote ser posicionado no setor desejado.



Tempo de busca

Tempo para posicionamento no cilindro correto. Em média, corresponde a $\frac{2}{3}$ do tempo máximo.



Latência rotacional

Tempo para posicionamento no setor correto. Em média, corresponde a $\frac{1}{2}$ rotação.



Tempo de transferência

Tempo de leitura ou escrita dos dados em disco. Depende da velocidade de rotação do disco e da densidade de dados da mídia.

taxa transferência =

$$\frac{\text{nº setores por trilha} * \text{tamanho do setor}}{60 / \text{velocidade em RPM}}$$

Eventualmente é necessário considerar o tempo de mudança de cabeçote e o tempo de mudança de cilindro.

Exemplo

Considere um disco cuja geometria seja de 1060 cilindros,

- 16 cabeçotes e 63 setores por trilha, sendo cada setor de 512 bytes.

Considere também:

- tempo médio de busca do disco rígido = 13,5 ms
- latência rotacional média = 6,7 ms
- velocidade rotacional = 4.500 RPM
- tempo de mudança de cabeçote = 1 ms
- tempo de mudança de cilindro = 2,5 ms
- Qual o tempo de leitura de um arquivo de 10 MB (20.480 setores)?



Exemplo 1 – Setores separados

- | | | |
|---|-------------------------------------|--------------------------------|
| ✓ | Tempo médio de busca por setor | 13,5 ms |
| ✓ | Latência rotacional média | 6,7 ms |
| ✓ | Transferência dos dados de um setor | 0,2 ms |
| ✓ | Tempo total por setor | 20,4 ms |
| ✓ | Tempo total dos 20.480 setores | 417,8 segundos
(±7 minutos) |

TEMPO DAS OPERAÇÕES



Exemplo 2 – Blocos de 8 setores

- ✓ Tempo médio de busca por setor 13,5 ms
- ✓ Latência rotacional média 6,7 ms
- ✓ Transferência dos dados de 8 setores 1,7 ms
- ✓ Tempo total por **bloco** 21,9 ms
- ✓ Tempo total dos 2.560 blocos 56,1 segundos

TEMPO DAS OPERAÇÕES



Exemplo 3 – Todos setores contíguos

- ✓ Tempo médio de busca por setor 13,5 ms
- ✓ Latência rotacional média 6,7 ms
- ✓ Transferência dos dados de todos setores 4.334,4 ms
- ✓ Tempo de 325 mudanças de cabeçote 325,0 ms
- ✓ Tempo de 20 mudanças de cilindro 50,0 ms
- ✓ Tempo total 4,7 segundos

Algoritmos e Estruturas de Dados III

Aula 3.1 – Arquivos Sequenciais

Prof. Hayala Curto
2022



PUC Minas

Roteiro do Conteúdo



3.1 Arquivos

- Arquivos Sequenciais
- Chaves de Ordenação
- Chave Primária
- Operações em Arquivos - CRUD

3.2 Ordenação Externa

- Ordenação Externa
- Intercalação Balanceada
- Análise de Complexidade
- Segmento de Tamanho Variável
- Seleção por Substituição

Roteiro do Conteúdo



3.1 Arquivos

- Arquivos Sequenciais
- Chaves de Ordenação
- Chave Primária
- Operações em Arquivos

3.2 Ordenação Externa

- Ordenação Externa
- Intercalação Balanceada

Roteiro da Aula

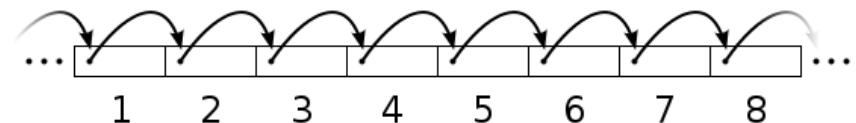
3.1 Arquivos

- **Arquivos Sequenciais**
- Chaves de Ordenação
- Chave Primária
- Operações em Arquivos

Arquivos Sequenciais

- Arquivos em que os registros são acessados na ordem em que estão armazenados.
- Normalmente são usados quando há poucas (ou nenhuma) movimentação de registros.
- O objetivo é o acesso rápido a um conjunto de registros.
- Não são bons para acessos aleatórios

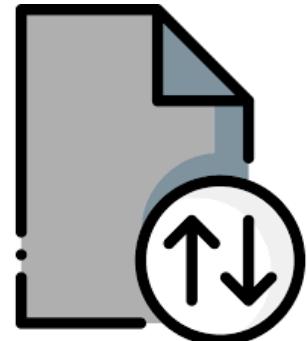
Sequential access



Arquivos Sequenciais - Ordenação

Os registros nos arquivos sequenciais podem estar ordenados por:

- Momento (data/hora) de criação
- Chave primária (atributo ou conjunto de atributos que identifica a entidade de forma exclusiva)
- Qualquer outra chave de ordenação
- Nenhuma ordem



Roteiro da Aula

3.1 Arquivos

- Arquivos Sequenciais
- **Chaves de Ordenação**
- Chave Primária
- Operações em Arquivos

Arquivos Sequenciais – Chave de Ordenação

A chave de ordenação é o atributo ou conjunto de atributos que estabelece a ordem dos registros

Tipos de chave

- Atributo (ex.: ID, CPF, CNPJ, RG, ...)
- Combinação de campos (ex.: estado+cidade)
- Processamento de campos (ex.: inversão de data)

Arquivos Sequenciais – Chaves de Ordenação

ID	NOME	VALOR (US\$)	TIPO	DATA LANC.
1	Black Lotus	150.000	Artifact	01/01/1992
2	Mox Sapphire	30.000	Artifact	01/04/1992
3	Ancestral Recall	18.000	Instant	01/01/1993
4	Time Twister	16.800	Sorcery	04/05/1994
5	Underground Sea	25.000	Land	01/06/1995

Arquivos Sequenciais – Chaves de Ordenação

ID	NOME	VALOR (US\$)	TIPO	DATA LANC.
1	Black Lotus	150.000	Artifact	01/01/1992
2	Mox Sapphire	30.000	Artifact	01/04/1992
3	Ancestral Recall	18.000	Instant	01/01/1993
4	Time Twister	16.800	Sorcery	04/05/1994
5	Underground Sea	25.000	Land	01/06/1995

Arquivos Sequenciais – Chaves de Ordenação

ID	NOME	VALOR (US\$)	TIPO	DATA LANC.
3	Ancestral Recall	18.000	Instant	01/01/1993
1	Black Lotus	150.000	Artifact	01/01/1992
2	Mox Sapphire	30.000	Artifact	01/04/1992
4	Time Twister	16.800	Sorcery	04/05/1994
5	Underground Sea	25.000	Land	01/06/1995

Arquivos Sequenciais – Chaves de Ordenação

ID	NOME	VALOR (US\$)	TIPO	DATA LANC.	Chave de Ordenação
1	Black Lotus	150.000	Artifact	01/01/1992	Lotus
3	Ancestral Recall	18.000	Instant	01/01/1993	Recall
2	Mox Sapphire	30.000	Artifact	01/04/1992	Sapphire
5	Underground Sea	25.000	Land	01/06/1995	Sea
4	Time Twister	16.800	Sorcery	04/05/1994	Twister

Arquivos Sequenciais – Chaves de Ordenação

ID	NOME	VALOR (US\$)	TIPO	DATA LANC.	Chave de Ordenação
1	Black Lotus	150.000	Artifact	01/01/1992	1992 01 01
2	Mox Sapphire	30.000	Artifact	01/04/1992	1992 04 01
3	Ancestral Recall	18.000	Instant	01/01/1993	1993 01 01
4	Time Twister	16.800	Sorcery	04/05/1994	1994 05 04
5	Underground Sea	25.000	Land	01/06/1995	1995 06 01

Arquivos Sequenciais - Observações

- O arquivo eventualmente precisará ser reordenado para se manter a ordem dos registros
 - Arquivos sequenciais são usados quando sofrem poucas alterações
- Arquivos sequenciais geralmente são usados como arquivos temporários
 - Existem outras estruturas bem melhores para acesso aleatório como os arquivos indexados e os arquivos diretos

Roteiro da Aula

3.1 Arquivos

- Arquivos Sequenciais
- Chaves de Ordenação
- **Chave Primária**
- Operações em Arquivos

Arquivos Sequenciais – Chaves Primárias

Como escolher a chave primária?

- Chaves candidatas
 - Chaves que identificam cada entidade de forma exclusiva
 - Ex.: CPF, CNPJ, RG, Matrícula, E-Mail, Telefone, etc.
- Chave primária
 - Chave candidata escolhida

Arquivos Sequenciais – Chaves Primárias

Problemas das chaves candidatas

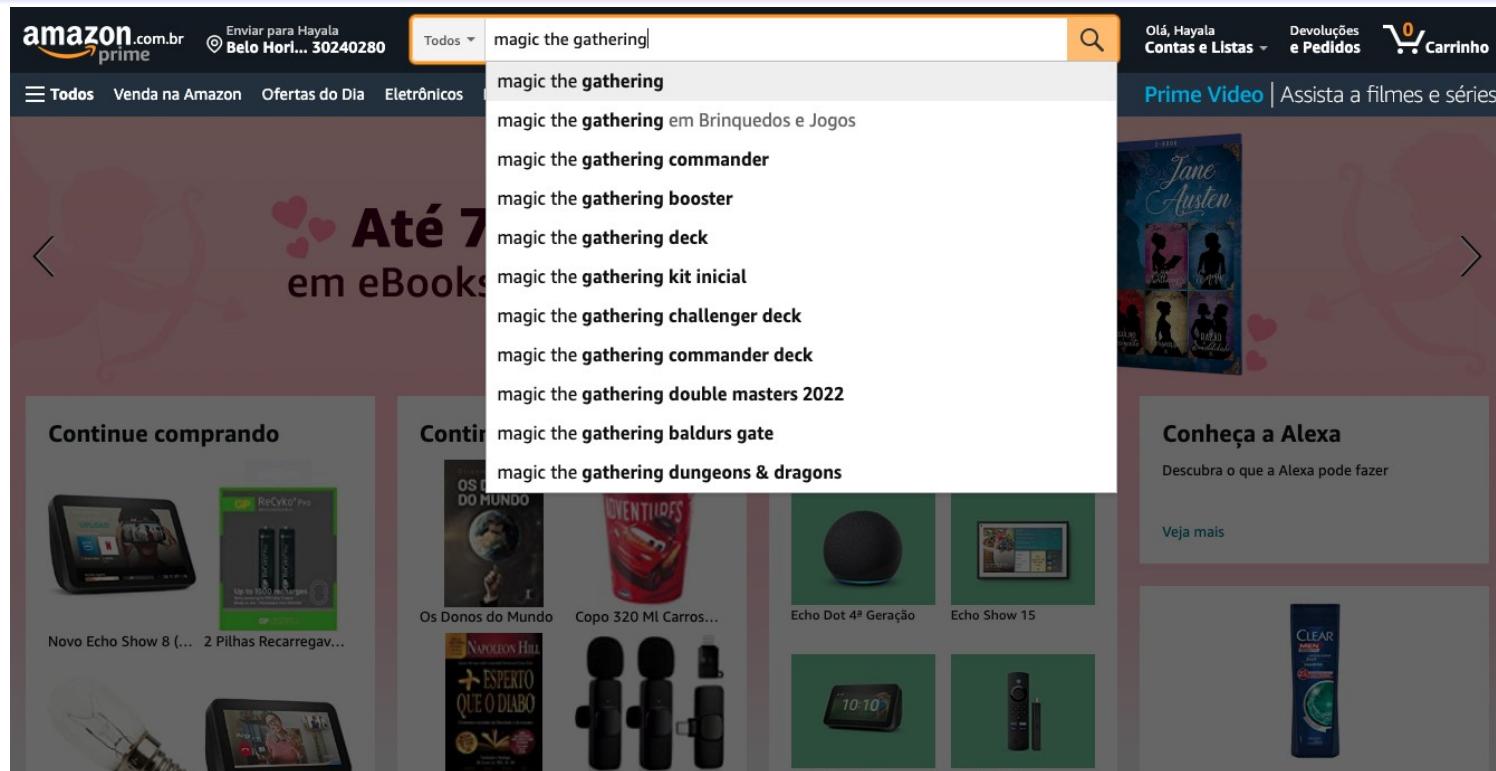
- Nome: Algumas pessoas têm o mesmo nome e outras mudam de nome quando se casam
- E-mail: As pessoas trocam seus e-mails
- Telefone: São reutilizáveis
- CPF e CNPJ: São muito longos

Arquivos Sequenciais – Chaves Primárias

Características Identificadores (ID):

- Numéricos (sem sinal) melhor aproveitamento dos bytes
- Sequenciais evita o desperdício de valores numéricos
- Exclusivos não podem ser ambíguos
- Não significativos não podem ser alterados
- Não reusáveis nunca são reaproveitados
- Os identificadores são chaves internas, isto é, só são usadas pelo sistema e não devem ser usados pelos usuários
- Os usuários continuarão buscando as entidades por meio de e-mails, CPFs, CNPJs, etc.

Arquivos Sequenciais – Identificadores



Arquivos Sequenciais – Identificadores

amazon.com.br Enviar para Hayala
Belo Horizonte 30240280 Todos magic the gathering Olá, Hayala Contas e Listas Devoluções e Pedidos Carrinho

Todos Venda na Amazon Ofertas do Dia Eletrônicos Ferramentas e Construção Games eBooks Kindle Computadores Conheça os Dispositivos Fire TV Ordenar por: Destaques

1-48 de 361 resultados para "magic the gathering"

Amazon Prime prime

Departamento

- Jogos e Acessórios
 - Decks e Conjuntos para Jogo de Cartas Colecionáveis
 - Jogos de Tabuleiro
 - Pacotes de Booster para Jogos de Cartas Colecionáveis
- Cartas Colecionáveis e Acessórios
- Livros
 - Importados de Artes, Cinema e Fotografia
 - Importados sobre Jogos de Cartas
 - Importados sobre Cultura Popular
 - Antiguidades e Colecionáveis

Ver todos os 5 Departamentos

Avaliação do Cliente

- 5 estrelas acima
- 4 estrelas acima
- 3 estrelas acima
- 2 estrelas acima

Marca Magic The Gathering

RESULTADOS



Patrocinados

Magic: The Gathering – Deck de Commander de Ruas de Nova Capenna – Mediadores Engalanados+ Pacote de amostra

R\$ 733⁵³

Econome 10%

R\$ 319¹⁹ R\$ 355,52

em até 10x de R\$ 32,00 sem juros



Patrocinados

Magic The Gathering Kamigawa: Dinastia Neon - Commander Deck - Português

R\$ 733⁵³

em até 10x de R\$ 73,38 sem juros



Patrocinados

Magic: The Gathering 2021 - Challenger Deck – Mono Green Stompy

5 estrelas 8

R\$ 178⁴⁰

em até 5x de R\$ 35,68 sem juros



Patrocinados

Magic: The Gathering - Caixa de Boosters de Draft de Espiral Temporal Remasterizada | 36 boosters (540 cards de Magic)

5 estrelas 2

R\$ 1.430⁹⁹

em até 10x de R\$ 143,18 sem juros

Arquivos Sequenciais – Identificadores

The screenshot shows a product page on the Brazilian Amazon website (https://www.amazon.com.br/Magic-Gathering-Challenger-Green-Stompy/dp/B08XJVK81F/ref=sr_1_3_sspa?keywords=magic+the+gathering). A blue arrow points from the identifier **B08XJVK81F** in the top right corner of the slide to the ASIN number displayed above the product title on the page.

B08XJVK81F

amazon.com.br Enviar para Hayala
Belo Horizonte 30240280 Todos magic the gathering Olá, Hayala Contas e Listas Devoluções e Pedidos Carrinho

Todos Venda na Amazon Ofertas do Dia Eletrônicos Ferramentas e Construção Games eBooks Kindle Computadores

Brinquedos e Jogos Mais Vendidos Ofertas Atividades ao Ar Livre Bonecas e Acessórios Brinquedos de Montar Brinquedos educativos Bonecos Colecionáveis e Miniaturas

Dia dos Pais Ainda dá tempo de presentear Confira ▶

« Voltar aos resultados

Magic: The Gathering 2021 - Challenger Deck – Mono Green Stompy
Visite a loja Magic The Gathering

R\$ 178⁴⁰
prime

Em até 5x R\$ 35,68 sem juros Ver parcelas disponíveis

Pagamentos e Segurança Enviado pela Amazon Política de devolução

Marca Magic The Gathering
Componentes MTG incluídos
Cor Preto
Material Paper

R\$ 178⁴⁰
Entrega GRÁTIS: Amanhã, 10 de Agosto
Ou Entrega mais rápida: Hoje.
Se pedir dentro de 5 hrs 3 mins

Enviar para Hayala - Belo Horizonte 30240280

Em estoque.
Quantidade: 1

Adicionar ao carrinho Comprar agora

Transação segura
Enviado por Amazon.com.br
Vendido por Amazon.com.br

Arquivos Sequenciais – Identificadores



B09WB5RRQ6

https://www.amazon.com.br/Magic-Gathering-Mediadores-Engalanados-colecionador/dp/B09WB5RRQ6/ref=sr_1_1_sspa?keywords=magic-the-gathering

amazon.com.br Envíar para Hayala Belo Horizonte 30240280 Todos magic the gathering Olá, Hayala Contas e Listas Devoluções e Pedidos Carrinho

Todos Venda na Amazon Ofertas do Dia Eletrônicos Ferramentas e Construção Games eBooks Kindle Computadores Copa do Brasil | Assistir AO VIVO

Brinquedos e Jogos Mais Vendidos Ofertas Atividades ao Ar Livre Bonecas e Acessórios Brinquedos de Montar Brinquedos educativos Bonecos Colecionáveis e Miniaturas

Dia dos Pais Ainda dá tempo de presentear Confira ▶

< Voltar aos resultados



Magic: The Gathering – Deck de Commander de Ruas de Nova Capenna – Mediadores Engalanados+ Pacote de amostra de booster de colecionador

Visite a loja Magic The Gathering

Econome 10% Menor preço dos últimos 30 dias

De: R\$355,52 Saiba mais

Por: R\$319,19 prime

Você economiza: R\$36,33 (10%)

Em até 10x R\$ 32,00 sem juros Ver parcelas disponíveis

Pagamentos e Segurança Enviado pela Amazon Política de devolução

R\$319,19 prime

Entrega GRÁTIS: Quinta-feira, 11 de Agosto. Se pedir dentro de 15 hrs 33 mins

Enviar para Hayala - Belo Horizonte 30240280

Em estoque.

Quantidade: 1

Adicionar ao carrinho Comprar agora

Transação segura

Enviado por Amazon.com.br
Vendido por Amazon.com.br

Comprar este item como presente

Arquivos Sequenciais – Identificadores

The screenshot shows an Instagram post from the account `netprojectppm`. The post features a photograph of a person's hand writing in a notebook with a green pen. The `netproject` logo is visible in the background. The caption discusses the importance of defining specific and direct objectives for projects. A blue callout box highlights the Instagram handle `ChAD-0ks7jH`.

Instagram Pesquisar

netprojectppm ...

netprojectppm Primeiramente, precisamos diferenciar metas e objetivos do projeto.

Os objetivos são mais específicos e diretos, eles influenciam todas as decisões e podem ser tratados como etapas para alcançar uma visão de alto nível das metas do projeto.

Isso significa que definir o objetivo é garantir que haja uma métrica para medir seu progresso para que você possa saber se está atendendo às suas expectativas de linha de base.

As metas podem ser um pouco...

[Ver insights](#)

13 visualizações HÁ 21 HORAS [Ver tradução](#)

Adicione um comentário... [Publicar](#)

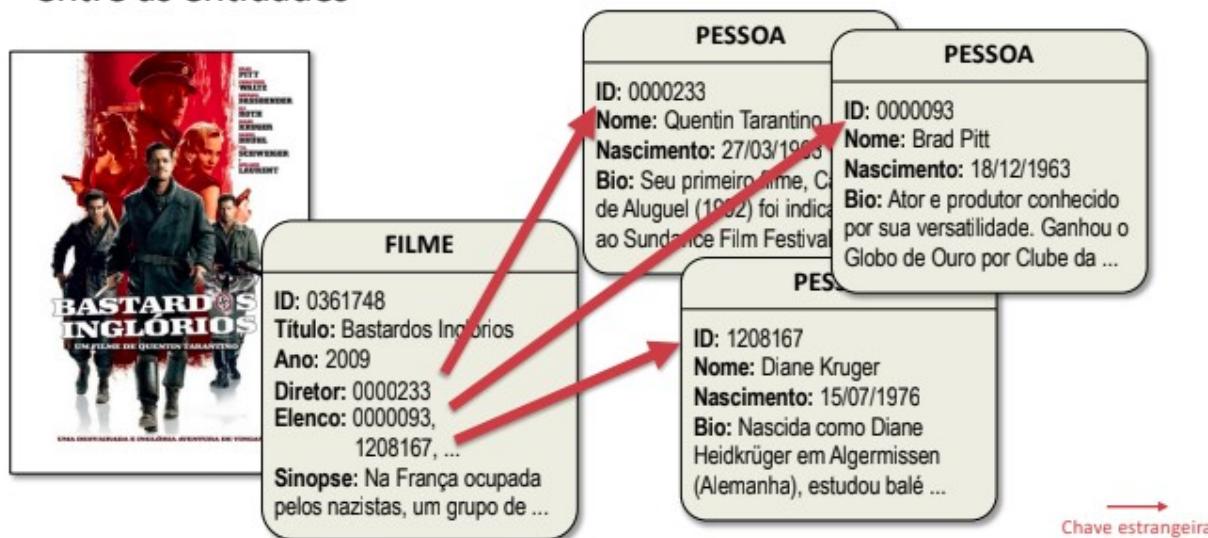
Como definir objetivos eficazes para um projeto?

[Leia na legenda](#)

ChAD-0ks7jH

Arquivos Sequenciais – Identificadores

Os identificadores ajudam a estabelecer os **relacionamentos** entre as entidades



Arquivos Sequenciais – Identificadores

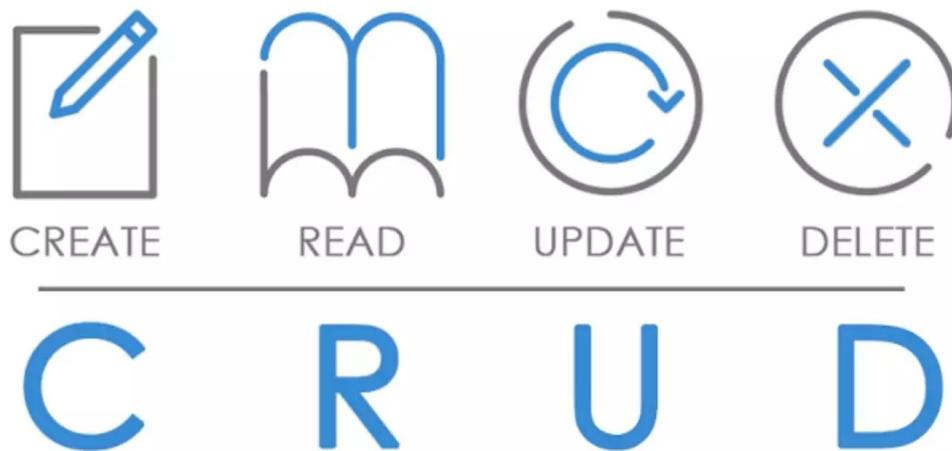


Roteiro da Aula

3.1 Arquivos

- Arquivos Sequenciais
- Chaves de Ordenação
- Chave Primária
- **Operações em Arquivos**

CRUD



CRUD

Operação	SQL	HTTP
Create	INSERT	POST / PUT
Read	SELECT	GET
Update	UPDATE	PUT / PATCH
Delete	DELETE	DELETE

Interface do CRUD

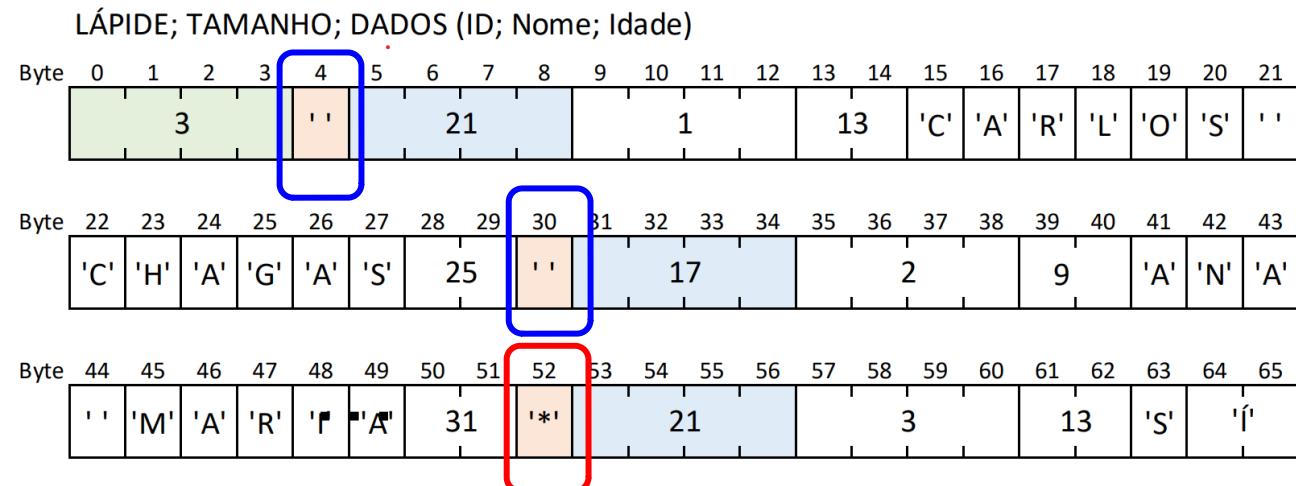
- $ID \leftarrow \text{arquivo.create}(novo_objeto)$
- $\text{objeto} \leftarrow \text{arquivo.read}(ID)$
- $\text{ok} \leftarrow \text{arquivo.update}(objeto_atualizado)$
- $\text{ok} \leftarrow \text{arquivo.delete}(ID)$

O ID passado nesse novo objeto será ignorado (usar -1)

O ID deve permanecer o mesmo, pois IDs nunca são alterados.

CRUD

- Lápide
 - Marca de exclusão
 - Campo (1 byte) que indica se o registro foi excluído ou permanece válido
 - Os registros só são realmente apagados do arquivo quando há uma reordenação



CRUD

- A operação 'excluir' não é física, mas lógica.
- Após a exclusão de um arquivo, o que o sistema operacional faz é eliminar de uma tabela o ponteiro que aponta para a posição do arquivo.
- O espaço do disco aparentemente excluído fica livre para armazenar novos dados
- Dados antigos irão persistir no HD até que sejam sobrescritos.
- Padrões magnéticos podem deixar vestígios de dados mais antigos na mídia física, ainda que os dados tenham sido sobrescritos diversas vezes.
- Programas de recuperação de dados se aproveitam desses mecanismos para resgatar arquivos excluídos do sistema.

CRUD

LÁPIDE; TAMANHO; DADOS (ID; Nome; Idade)

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
					3				21					1		13	'C'	'A'	'R'	'L'	'O'	'S'	''

Byte	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	
	'C'	'H'	'A'	'G'	'A'	'S'		25							17			2		9	'A'	'N'	'A'

Byte	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65		
									''						31	'*'		21			3	13	'S'	'Í'

Byte	66	67	68	69	70	71	72	73	74	75	76	77
	'L'	'V'	'I'	'O'	''	'C'	'O'	'S'	'T'	'A'		27

CRUD - Algoritmo Create (inclusão)

```
01: algoritmo create(objeto)
02:   |   mover o ponteiro para início do arquivo (cabeçalho)
03:   |   ler últimoID
04:   |   objeto.ID ← últimoID + 1
05:   |   mover o ponteiro para início do arquivo
06:   |   escrever objeto.ID
07:   |   criar registro para o objeto
08:   |   mover para o fim do arquivo
09:   |   escrever registro
10: fim-algoritmo
```

CRUD - Algoritmo Read (busca sequencial)

```
01: algoritmo read(ID)
02:   mover o ponteiro para o primeiro registro (após o cabeçalho)
03:   enquanto não atingir o fim do arquivo
04:     | ler próximo registro
05:     | se registro.lapide ≠ '*'
06:       | então extrair objeto do registro
07:         | se objeto.ID = ID
08:           |   | então retornar objeto e terminar
09:           |   | fim-se
10:     | fim-se
11:   fim-enquanto
12:   retornar objeto vazio // null
13: fim-algoritmo
```

CRUD - Algoritmo Read (busca sequencial)

Obs: leitura de um conjunto de objetos

```
01: algoritmo read(critérios)
02:     |   criar conjunto vazio
03:     |   mover o ponteiro para o primeiro registro (após o cabeçalho)
04:     |   enquanto não atingir o fim do arquivo
05:         |       |   ler próximo registro
06:         |       |   se registro.lapide ≠ '*'
07:         |           |   então extrair objeto do registro
08:         |               |   se registro atender aos critérios
09:         |                   |   então adicionar objeto ao conjunto
10:             |               |   fim-se
11:         |       |   fim-se
12:     |   fim-enquanto
13:     |   retornar conjunto
14: fim-algoritmo
```

CRUD - Algoritmo Update (atualização)

Se a alteração não implicar em mudança de tamanho

- Escrever o registro alterado na mesma posição

Se o registro diminuir de tamanho

- Escrever o registro na mesma posição, mas mantendo o mesmo indicador de tamanho

Se o registro aumentar de tamanho

- Marcar o registro atual como excluído e criar um novo registro no fim do arquivo

```
01: algoritmo update(novoObjeto)
02:   mover para o primeiro registro do arquivo (após cabeçalho)
03:   enquanto não atingir o fim do arquivo
04:     pos ← posição do ponteiro
05:     ler próximo registro
06:     se registro.lapide ≠ '*'
07:       então extraír objeto do registro
08:       se objeto.ID = novoObjeto.ID
09:         então criar novoRegistro para novoObjeto
10:         se novoRegistro.tamanho ≤ registro.tamanho
11:           então mover para pos
12:             escrever novoRegistro mantendo ind.tam.
13:           senão mover para pos
14:             escrever lápide como excluído
15:             mover para fim do arquivo
16:             escrever novoRegistro
17:         fim-se
18:         retornar verdadeiro e terminar
19:     fim-se
20:   fim-enquanto
21:   retornar falso
22: fim-algoritmo
```

CRUD - Algoritmo Delete (exclusão)

```
01: algoritmo delete(ID)
02:   mover o ponteiro para o primeiro registro (após o cabeçalho)
03:   enquanto não atingir o fim do arquivo
04:     pos ← posição do ponteiro
05:     ler próximo registro
06:     se registro.lapide ≠ '*'
07:       então extrair objeto do registro
08:         se objeto.ID = ID
09:           então mover para pos
10:             escrever lápide como excluído
11:             retornar verdadeiro e terminar
12:         fim-se
13:     fim-se
14:   fim-enquanto
15:   retornar falso
16: fim-algoritmo
```

.

CRUD - Considerações finais

- Update e Delete geralmente passam por um Read anterior
- Arquivos sequenciais dependem de acesso sequencial, o que significa que eles não são bons para quaisquer operações de acesso aleatório
- Os espaços deixados pelos registros excluídos são espaços que podem ser reaproveitados (desde que exista uma lógica para esse reaproveitamento)
- O arquivo deve ser reordenado sempre que necessário (ordenação externa)

Dúvidas?



hayala.curto@sga.pucminas.br

Algoritmos e Estruturas de Dados III

Aula 4.1 Arq. Indexados

Prof. Hayala Curto
2022



PUC Minas

Arquivo Indexado

- Arquivos indexados são arquivos em que os registros são acessados de forma aleatória. Para que a busca seja eficiente, esses arquivos contam com estruturas de dados adicional de apoio (chamadas índices).
- Assim, a busca não é realizada diretamente no arquivo de dados, mas no tal índice, que retornará quais registros atendem ao critério de busca e onde eles estão localizadas no arquivo de dados.
- Um arquivo indexado é um arquivo que possui um ou mais índices que permitem acesso aleatório a um registro, dada uma determinada chave.

Arquivo Indexado

Um arquivo pode ter vários índices, cada um baseado em um atributo (ou conjunto de atributos) diferente.

Em um sistema de gerenciamento de clientes, é possível termos, por exemplo, um índice baseado no ID de cliente, outro baseado no nome do cliente e ainda um terceiro baseado no e-mail do cliente.

Arquivo Indexado

Pos.	ID	Título	Plataforma	Preço
4	2	Minecraft	Xbox	79,00
60	3	GTA V	PS4	120,00
90	7	Mario	Nintendo	295,00
135	8	Fortnite	PC	48,00
182	13	Zelda	Nintendo	300,00
253	15	The Sims 4	PC	99,00

Arquivo Indexado pela chave primária

Pos.	ID	Título	Plataforma	Preço
4	2	Minecraft	Xbox	79,00
60	3	GTA V	PS4	120,00
90	15	Mario	Nintendo	295,00
135	8	Fortnite	PC	48,00
182	7	Zelda	Nintendo	300,00
253	13	The Sims 4	PC	99,00

Pos.	ID	Pos.
0	2	4
12	3	60
24	7	182
36	8	135
48	13	253
60	15	90

Arquivo Indexado pelo título

Pos.	ID	Título	Plataforma	Preço
4	2	Minecraft	Xbox	79,00
60	3	GTA V	PS4	120,00
90	15	Mario	Nintendo	295,00
135	8	Fortnite	PC	48,00
182	7	Zelda	Nintendo	300,00
253	13	The Sims 4	PC	99,00

Pos.	Título	Pos.
0	Fortnite	135
12	GTA V	60
24	Mario	90
36	Minecraft	4
48	The Sims 4	253
60	Zelda	182

Tipos de Índices



PUC Minas

Tipos de índices

- Primários ou secundários
- Diretos ou indiretos
- Densos ou esparsos

Índices primários ou secundários

- **Índices primários:** seguem a mesma ordem do arquivo de dados
- **Índices secundários:** não seguem a mesma ordem do arquivo de dados

Índices primários ou secundários

- **Índices primários: seguem a mesma ordem do arquivo de dados**
- Índices secundários: não seguem a mesma ordem do arquivo de dados

	ID	End		ID	Dados
0	10	4		4	10
12	20	60		60	20
24	30	96		96	30
36	40	137	→	137	40
48	50	182		182	50
60	60	223		223	60
72	70	252		252	70
84	80	360		360	80
96	90	415		415	90
108	100	484		484	100

Índices primários ou secundários

- Índices primários: seguem a mesma ordem do arquivo de dados
- **Índices secundários:** não seguem a mesma ordem do arquivo de dados

	ID	End		ID	Dados
0	10	415		4	20
12	20	4		60	50
24	30	96		96	30
36	40	484		137	90
48	50	60		182	100
60	60	360		223	80
72	70	252		252	70
84	80	223		360	60
96	90	137		415	10
108	100	182		484	40

Índices diretos ou indiretos

- **Índices diretos:** apontam diretamente para a posição do registro no arquivo de dados
- **Índices indiretos:** apontam para um índice direto, normalmente, baseado na chave primária (que, por sua vez, aponta para o arquivo de dados)

Índices diretos ou indiretos

- **Índices diretos:** apontam diretamente para a posição do registro no arquivo de dados
- Índices indiretos: apontam para um índice direto, normalmente, baseado na chave primária (que, por sua vez, aponta para o arquivo de dados)

	ID	End
0	10	415
12	20	4
24	30	96
36	40	484
48	50	60
60	60	360
72	70	252
84	80	223
96	90	137
108	100	182

A red curved arrow points from the value 484 in the End column of the first table to the ID column of the second table.

	ID	Dados
4	20	
60	50	
96	30	
137	90	
182	100	
223	80	
252	70	
360	60	
415	10	
484	40	

Índices diretos ou indiretos

- Índices diretos: apontam diretamente para a posição do registro no arquivo de dados
- **Índices indiretos: apontam para um índice direto, normalmente, baseado na chave primária (que, por sua vez, aponta para o arquivo de dados)**

Chave	ID
Ana	50
Beatriz	90
Carlos	60
Daniel	80
Fernando	30
Gabriela	70
Paulo	40
Renato	10
Simone	20
Teresa	100

ID	End
10	415
20	4
30	96
40	484
50	60
60	360
70	252
80	223
90	137
100	182

ID	Dados
4	20
60	50
96	30
137	90
182	100
223	80
252	70
360	60
415	10
484	40

The diagram illustrates the structure of an indirect index. It consists of three tables:

- Primary Key Table:** Chave (Name) and ID (Value). The entries are: Ana (50), Beatriz (90), Carlos (60), Daniel (80), Fernando (30), Gabriela (70), Paulo (40), Renato (10), Simone (20), and Teresa (100).
- Index Table:** ID (Record ID) and End (Starting Address). The entries are: 10 (415), 20 (4), 30 (96), 40 (484), 50 (60), 60 (360), 70 (252), 80 (223), 90 (137), and 100 (182).
- Data Table:** ID (Record ID) and Dados (Data Value). The entries are: 4 (20), 60 (50), 96 (30), 137 (90), 182 (100), 223 (80), 252 (70), 360 (60), 415 (10), and 484 (40).

Red arrows indicate the pointers: one arrow points from the 'Simone' row in the primary key table to the '20' entry in the index table; another arrow points from the '20' entry in the index table to the '4' entry in the data table.

Índices densos ou esparsos

- **Índices densos:** possuem uma entrada para cada registro no arquivo de dados
- **Índices esparsos:** possuem entradas para apenas alguns registros

Índices densos ou esparsos

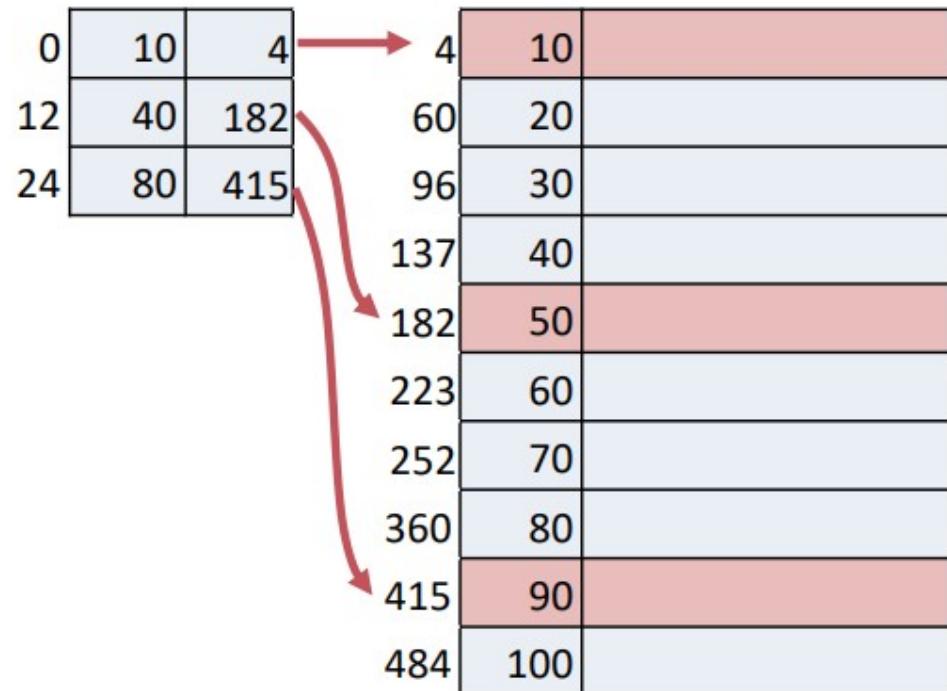
- **Índices densos:** possuem uma entrada para cada registro no arquivo de dados
- Índices esparsos: possuem entradas para apenas alguns registros

0	10	4
12	20	60
24	30	96
36	40	137
48	50	182
60	60	223
72	70	252
84	80	360
96	90	415
108	100	484

4	10	
60	20	
96	30	
137	40	
182	50	
223	60	
252	70	
360	80	
415	90	
484	100	

Índices densos ou esparsos

- **Índices densos:** possuem uma entrada para cada registro no arquivo de dados
- **Índices esparsos:** possuem entradas para apenas alguns registros



Operações em Arquivos Indexados

CRUD



PUC Minas

- As operações realizadas em arquivos indexados são as mesmas realizadas nos arquivos sequenciais.
- A principal diferença aqui, porém, é que as buscas por quaisquer entidades não serão feitas sequencialmente no arquivo dados, mas serão feitas em um dos índices.
 - Para que isso funcione corretamente, os índices devem sempre estar atualizados,

Interface do CRUD

- `ID ← arquivo.create(novo_objeto)`
- `objeto ← arquivo.read(ID)`
- `ok ← arquivo.update(objeto_atualizado)`
- `ok ← arquivo.delete(ID)`

Observação: A interface do CRUD não deve oferecer acesso direto aos índices usados

CREATE

```
01: algoritmo create(objeto)
02:     mover o ponteiro para início do arquivo (cabeçalho)
03:     ler últimoID
04:     objeto.ID ← últimoID + 1
05:     mover o ponteiro para início do arquivo
06:     escrever objeto.ID
07:     criar registro para o objeto
08:     mover para o fim do arquivo
09:     pos ← posição do ponteiro
10:     escrever registro
11:     inserir o par (objeto.ID, pos) no índice
12: fim-algoritmo
```

READ

```
01: algoritmo read(ID)
02:   pos ← buscar o ID no índice
03:   se pos ≠ -1
04:     mover ponteiro para pos
05:     ler registro
06:     se registro.lapide ≠ '*' // checagem dupla
07:       então extrair objeto do registro
08:       retornar objeto e terminar
09:     fim-se
10:   fim-se
11:   retornar objeto vazio // null
12: fim-algoritmo
```

DELETE

```
01: algoritmo delete(ID)
02:     pos ← buscar o ID no índice
03:     se pos ≠ -1
04:         então mover ponteiro para pos
05:             ler registro
06:             se registro.lapide ≠ '*'
07:                 então extrair objeto do registro
08:                     mover para pos
09:                     escrever lápide como excluído
10:                     remover o ID do índice
11:                     retornar verdadeiro e terminar
12:             fim-se
13:     fim-se
14:     retornar falso
15: fim-algoritmo
```

UPDATE

```
01: algoritmo update(novoObjeto)
02:     pos ← buscar o ID no índice
03:     se pos ≠ -1
04:         então mover ponteiro para pos
05:             ler registro
06:             se registro.lapide ≠ '*'
07:                 então extrair objeto do registro
08:                     criar novoRegistro para novoObjeto
09:                     se novoRegistro.tamanho ≤ registro.tamanho
10:                         então mover para pos
11:                             escrever novoRegistro (manter ind.tam.)
12:                         senão mover para pos
13:                             escrever lápide como excluído
14:                             mover para fim do arquivo
15:                             pos ← posição do ponteiro
16:                             escrever novoRegistro
17:                             atualizar o endereço para o ID no índice
18:                             retornar verdadeiro e terminar
19:             fim-se
20:         fim-se
21:     fim-se
22:     retornar falso
23: fim-algoritmo
```

Upd

Considerações importantes

- O algoritmo apresentado possui apenas um índice secundário (por ID), direto e denso. Se os arquivos tiverem mais índices, esse algoritmo deve ser modificado.
- Os índices podem ser gerenciados fora do CRUD, mas isso requer muita atenção extra.

Considerações sobre Relacionamentos



PUC Minas

Relacionamento

Um relacionamento é uma associação entre duas entidades em um banco de dados

Exemplos:

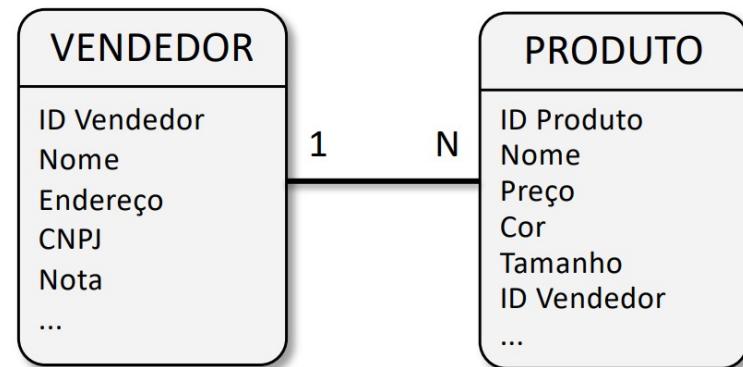
- Produtos de um vendedor
- Compromissos de um usuário
- Operações bancárias de uma conta corrente (de um correntista)

Relacionamento

Um relacionamento é uma associação entre duas entidades em um banco de dados

RELACIONAMENTO 1:N

- Cada vendedor possui N produtos
- Cada produto pertence a 1 vendedor
- O relacionamento é feito por meio de chaves estrangeiras



Tipos de relacionamento

- **Um para um (1:1)**

- Liga uma entidade de um tipo a uma outra entidade de outro tipo
 - Exemplo: pessoa 1:1 habitação

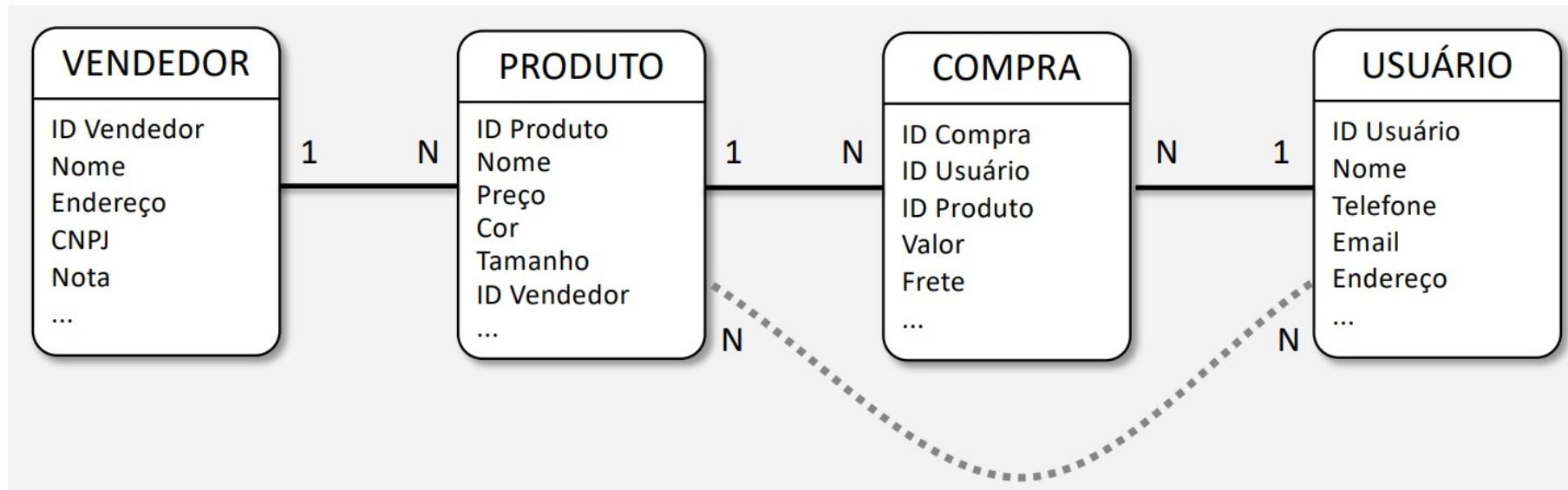
- **Um para muitos (1:N)**

- Uma entidade de um tipo está ligada a várias outras entidades de outro tipo
 - Exemplo: categoria 1:N produto

- **Muitos para muitos (N:N)**

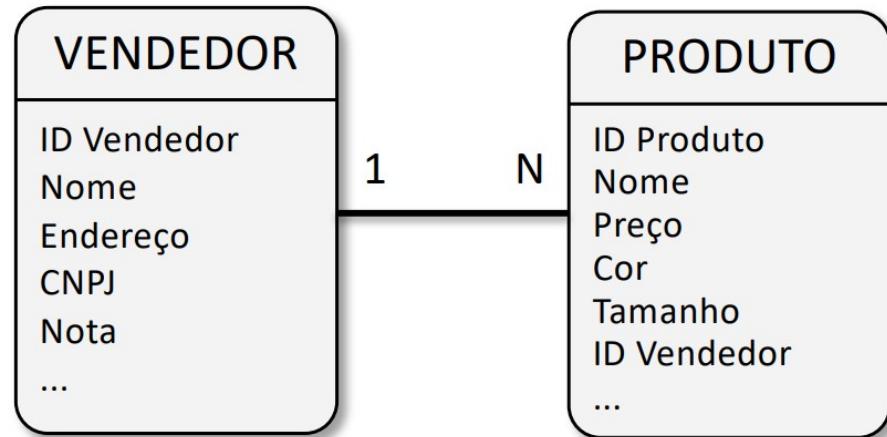
- Várias entidades de um tipo estão ligadas a várias entidades de outro tipo
 - São relacionamentos que geralmente dependem de uma entidade intermediária.
 - Exemplo: ator N:N filme

Relacionamentos



Observações importantes

- Quanto usamos relacionamentos, precisamos nos preocupar com a **integridade dos dados**.
- Os códigos do CRUD precisam ser alterados para garantir essa integridade.
- Muitas vezes, a **exclusão** será substituída por uma desativação.
 - ex.: vendas realizadas / produtos excluídos



Não podemos excluir um vendedor que tenha produtos cadastrados

Algoritmos e Estruturas de Dados III

4.2 Árvores B

Prof. Hayala Curto
2022



PUC Minas

Estrutura do Índice

- Problema
 - Uma busca sequencial em um arquivo de índice pode não ter muita valia
 - Pode-se fazer busca um pouco mais eficiente (ex. busca binária), se o arquivo de índice estiver ordenado
- Solução:
 - Assumir que os índices não são estruturas sequenciais, e sim hierárquicas
 - Melhor opção: Árvores de Múltiplos Caminhos (Árvore B, Árvore B*, Árvore B+)

Árvores de Múltiplos Caminhos

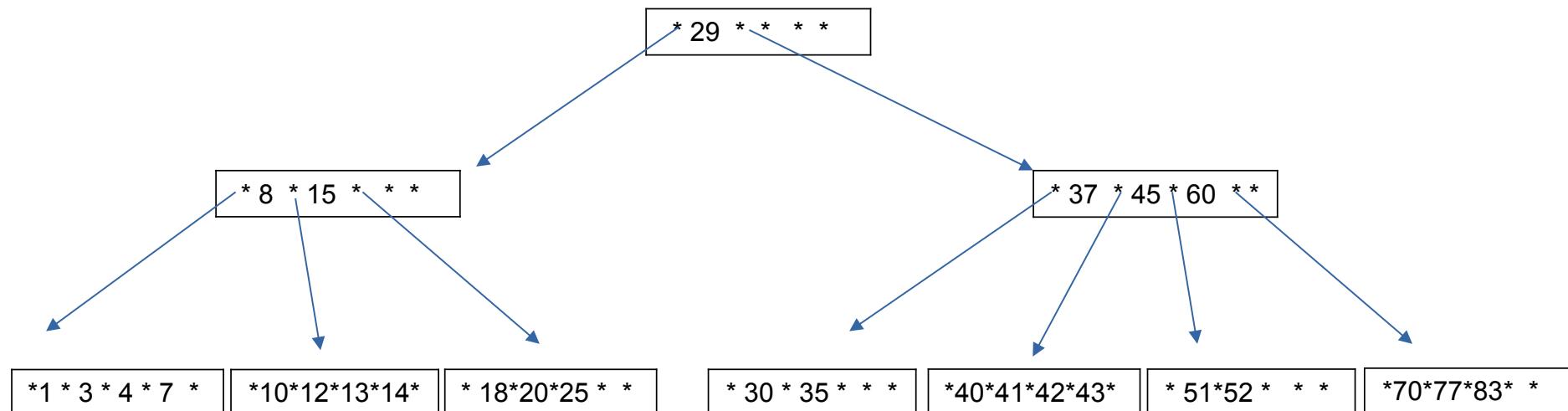
- Têm altura bem menor que as árvores binárias
- As árvores são “baixas”, sendo necessários poucos acessos em disco até chegar ao ponteiro para o bloco que contém o registro desejado
- Ideais para uso como índice de arquivos em disco

Árvore B

- Conceitos
 - **Ordem:**
 - Diz qual é o número máximo de filhos que cada página pode ter
 - **Página:**
 - Nó da árvore B
- Regras da árvore B:
 - Cada página (exceto a raiz) deve ter pelo menos 50% de ocupação
 - O número de filhos de cada página (exceto as folhas) deve ser igual ao seu número de chaves mais um
 - Todas as folhas estão no mesmo nível (a árvore cresce para cima).

Árvore B

Árvore de busca em que cada nodo (ou página) contém mais de 1 elemento



Árvore B

Árvore de busca em que cada nodo (ou página) contém mais de 1 elemento

* 29 * * * *

Página
permite a manipulação
de um bloco de informações.
Grande o suficiente para
ocupar
muita memória.

* 8 * 15 * * *

* 37 * 45 * 60 * *

*1 * 3 * 4 * 7 *

*10*12*13*14*

* 18*20*25 * *

* 30 * 35 * * *

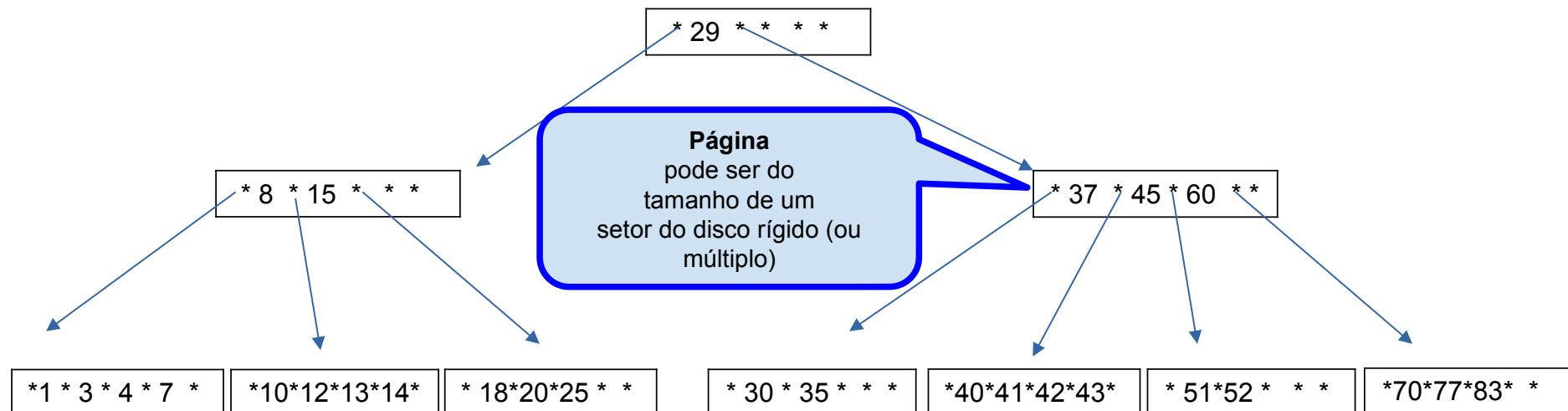
*40*41*42*43*

* 51*52 * * *

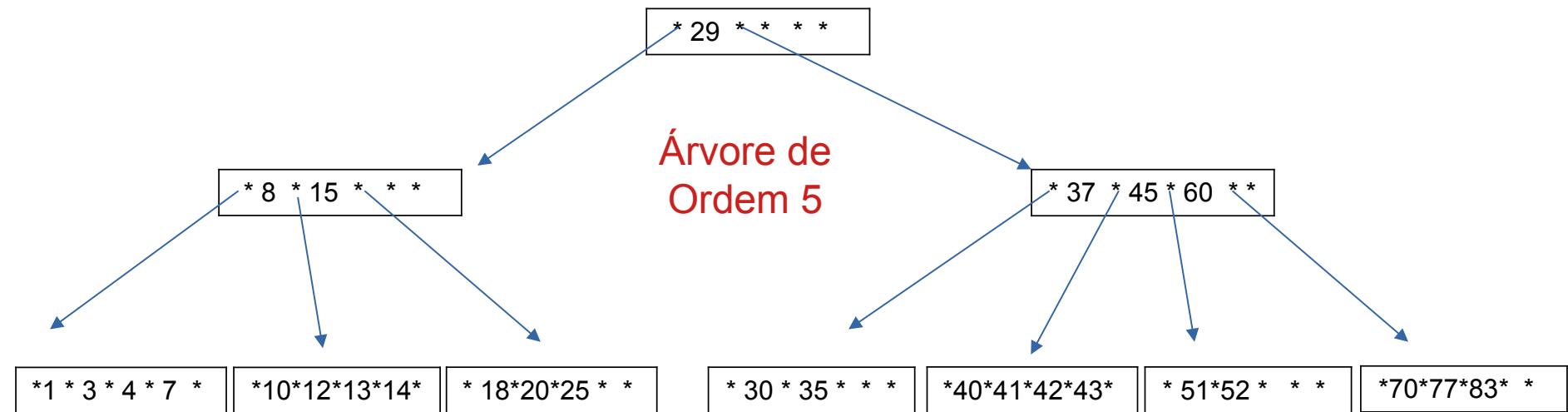
*70*77*83* *

Árvore B

Árvore de busca em que cada nodo (ou página) contém mais de 1 elemento



Árvore B - Ordem

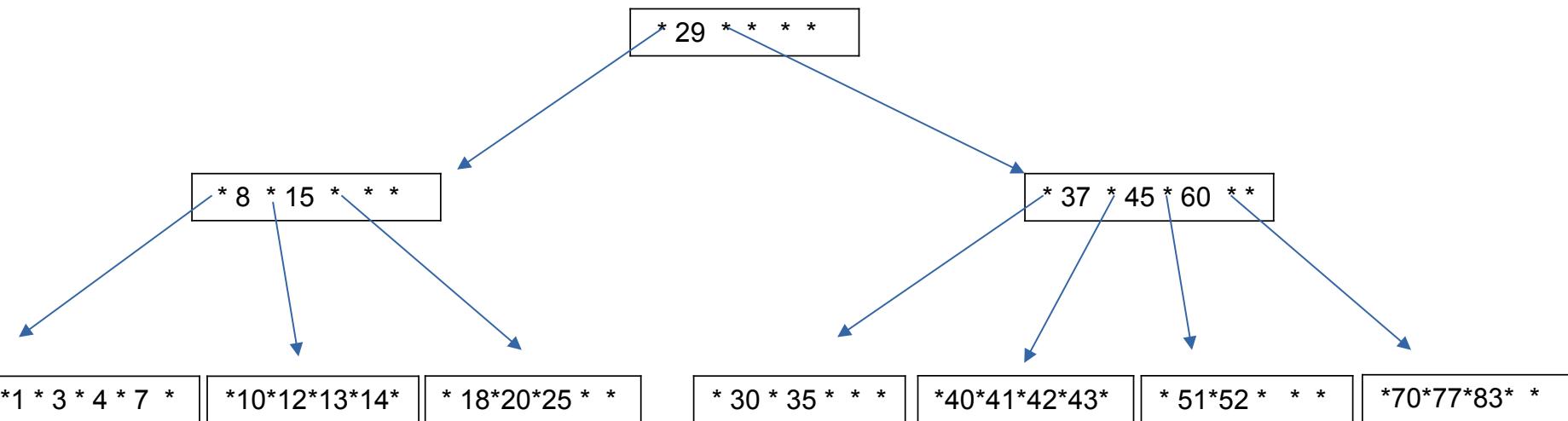


Número mínimo de elementos que cada página (exceto raiz) pode ter (Cormen, 2001; Bayer e McCreight, 1972)



Número de filhos que cada página pode ter (Knuth, 1978)

Árvore B - Regras



- Cada página deve ter pelo menos 50% de ocupação (considerar ordem da árvore), exceto a raiz: Otimização
- O número de filhos (exceto folha) deve ser o número de chaves + 1: Balanceamento
- Todas as folhas estão no mesmo nível (o crescimento é para cima)

Árvore B - Propriedades

Para uma Árvore-B de ordem m

- Cada página tem, no máximo, m descendentes
- Cada página, exceto a raiz e as folhas, tem no mínimo teto($m/2$) descendentes
- A raiz tem, no mínimo, dois descendentes – a menos que seja uma folha
- Todas as folhas estão no mesmo nível
- Uma página não folha que possui k descendentes contém $k-1$ chaves
- Uma página folha contém, no mínimo teto($m/2$) -1 e, no máximo, $m-1$ chaves

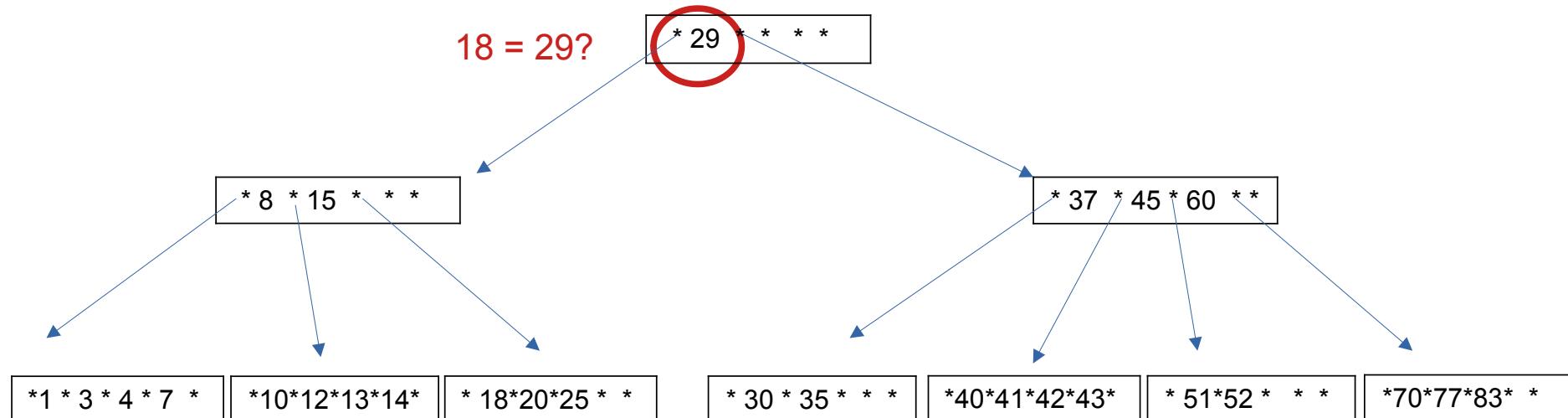
Busca em Árvore B



PUC Minas

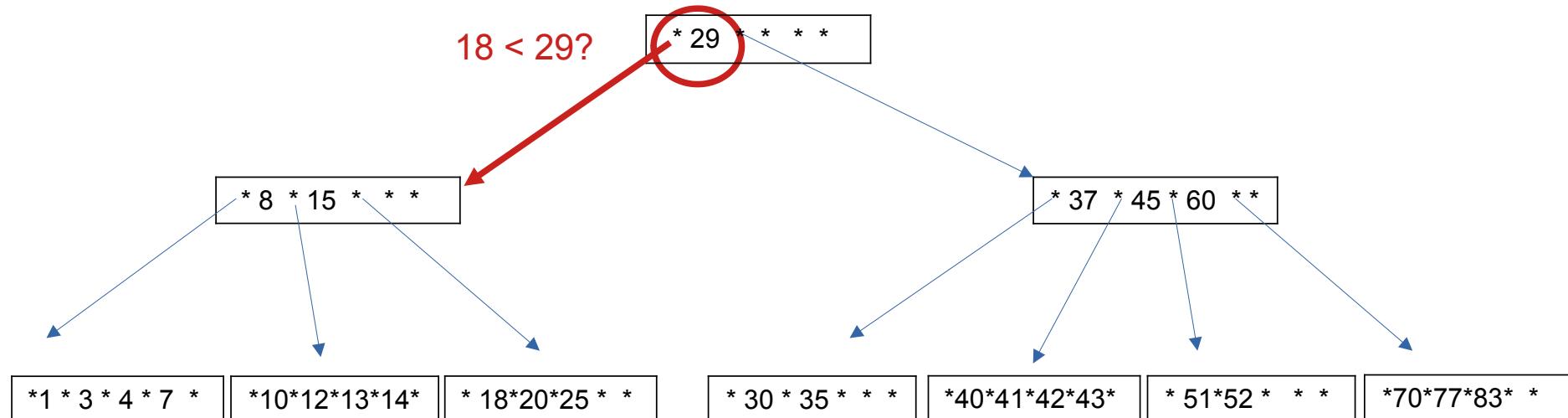
Árvore B - Busca

Exemplo: Localizar a chave 18



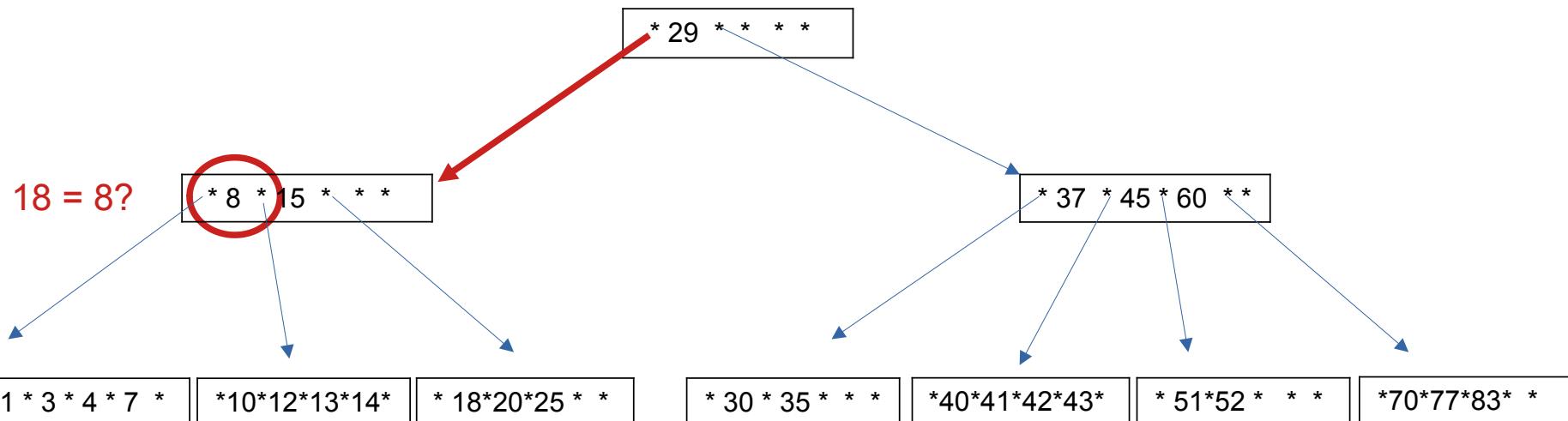
Árvore B - Busca

Exemplo: Localizar a chave 18



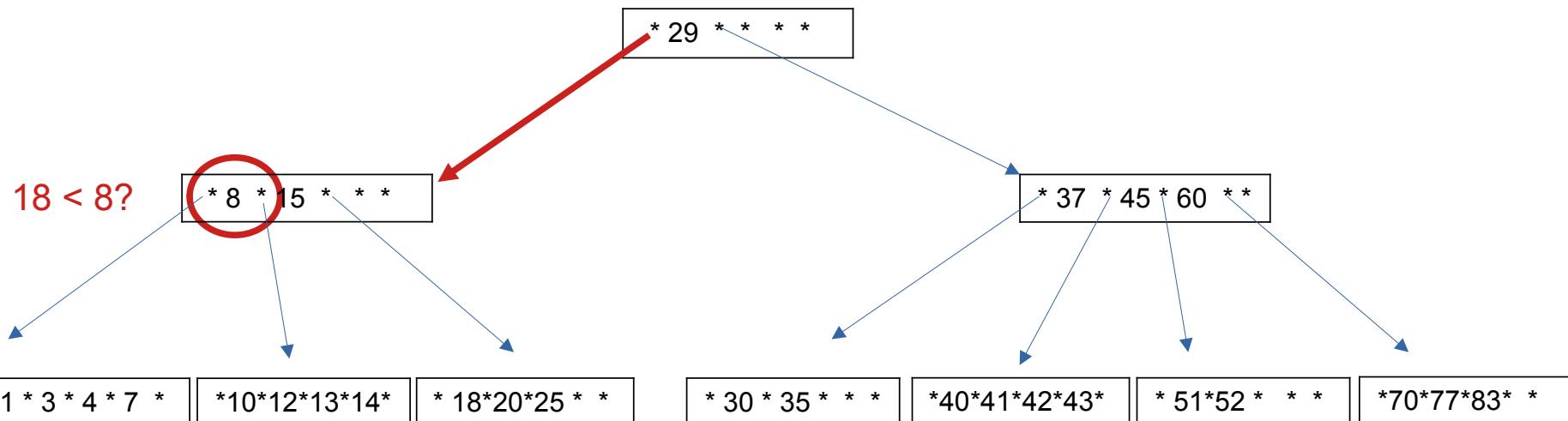
Árvore B - Busca

Exemplo: Localizar a chave 18



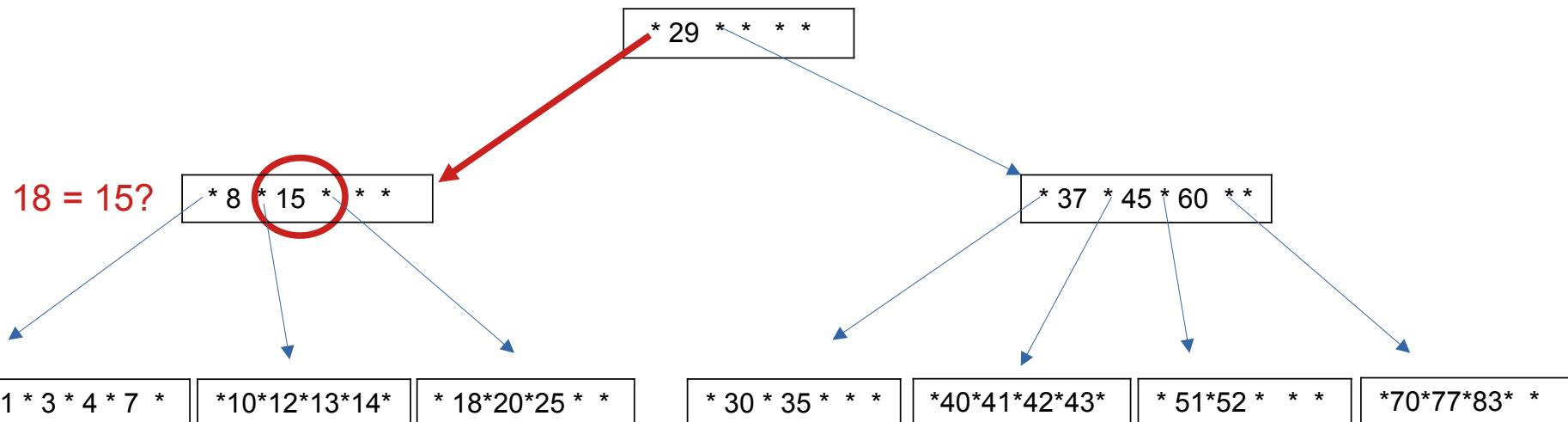
Árvore B - Busca

Exemplo: Localizar a chave 18



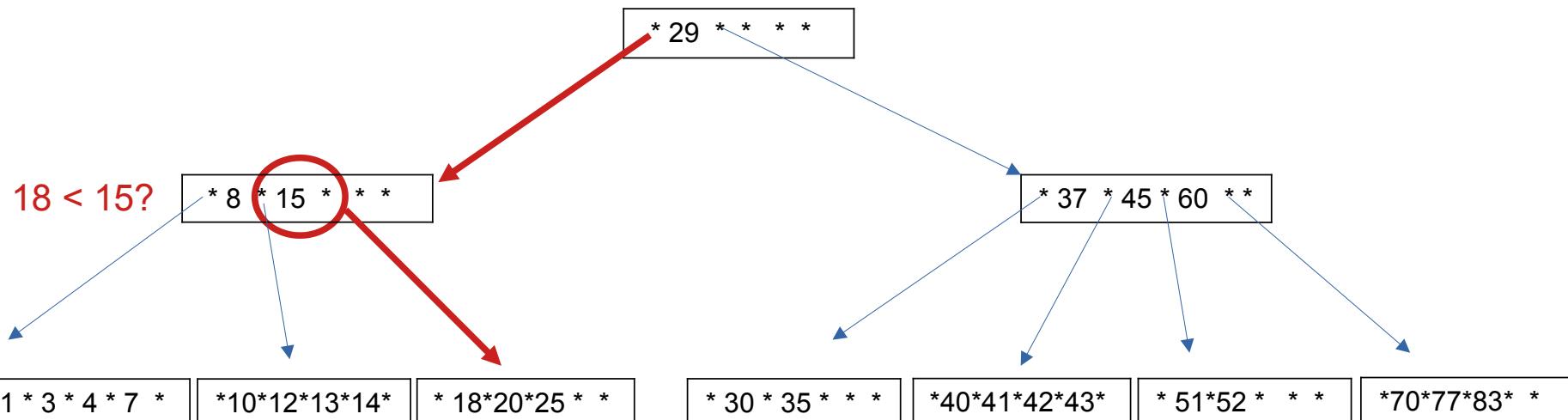
Árvore B - Busca

Exemplo: Localizar a chave 18



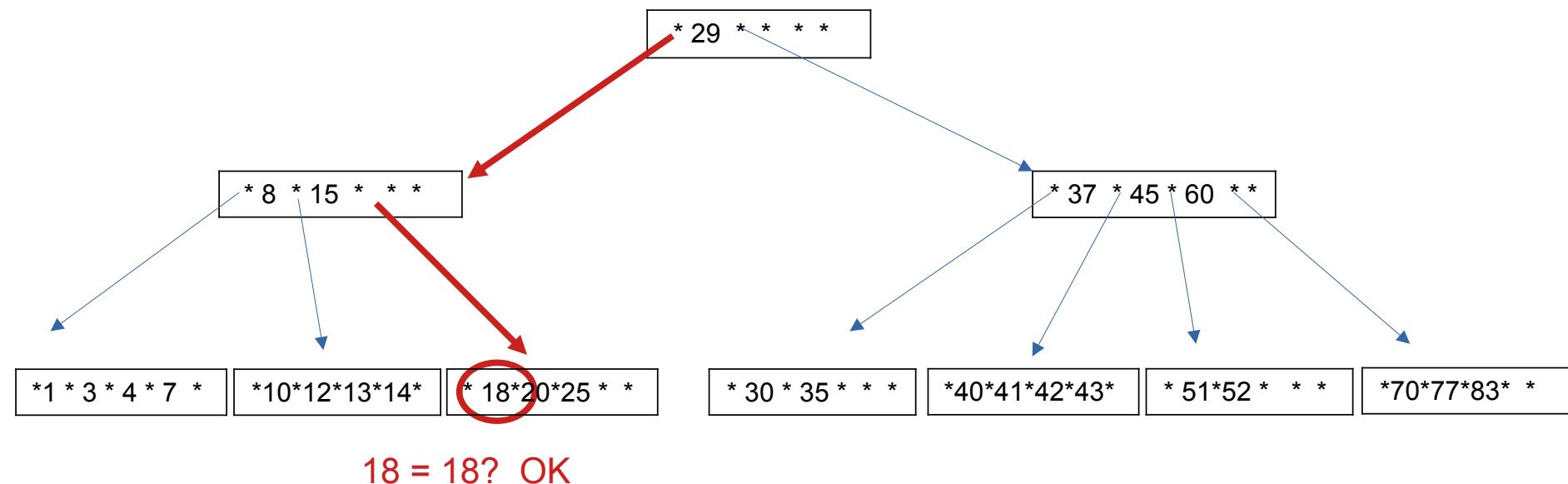
Árvore B - Busca

Exemplo: Localizar a chave 18



Árvore B - Busca

Exemplo: Localizar a chave 18



Inserção em Árvore B



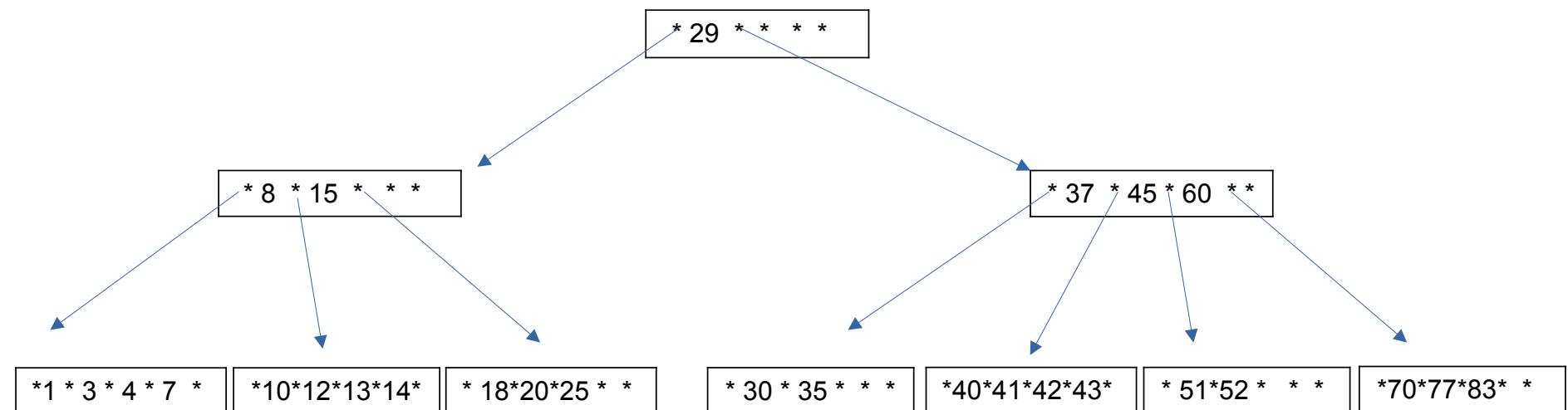
PUC Minas

Árvore B - Inserção

- Se o elemento couber na página, basta inclui-lo de forma ordenada
- Se não couber, a página deve ser dividida em duas (split) e o elemento do meio deve ser promovido

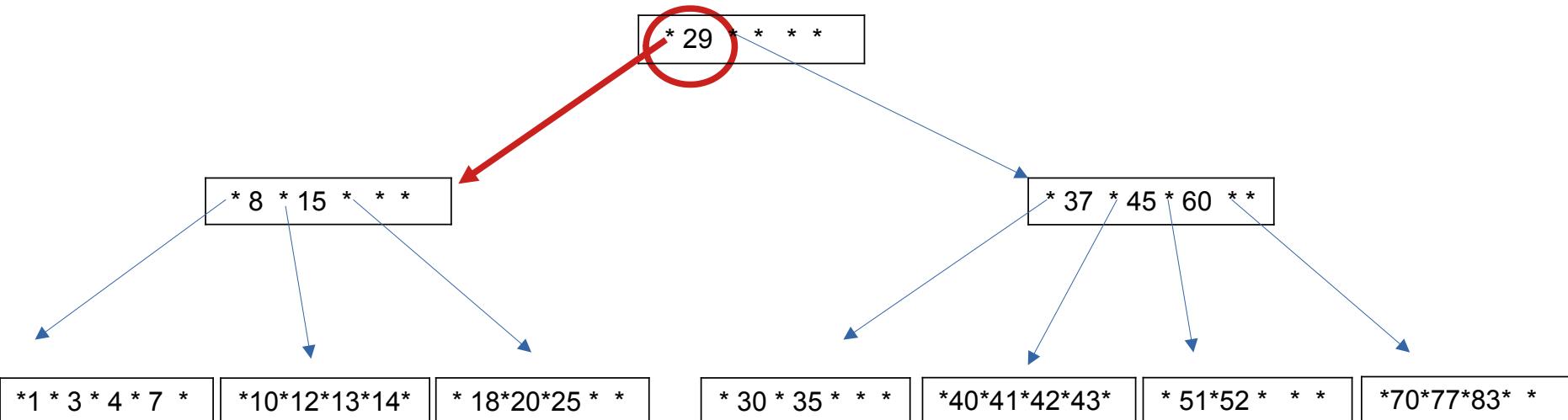
Árvore B - Inserção

Exemplo: Inserir a chave 11



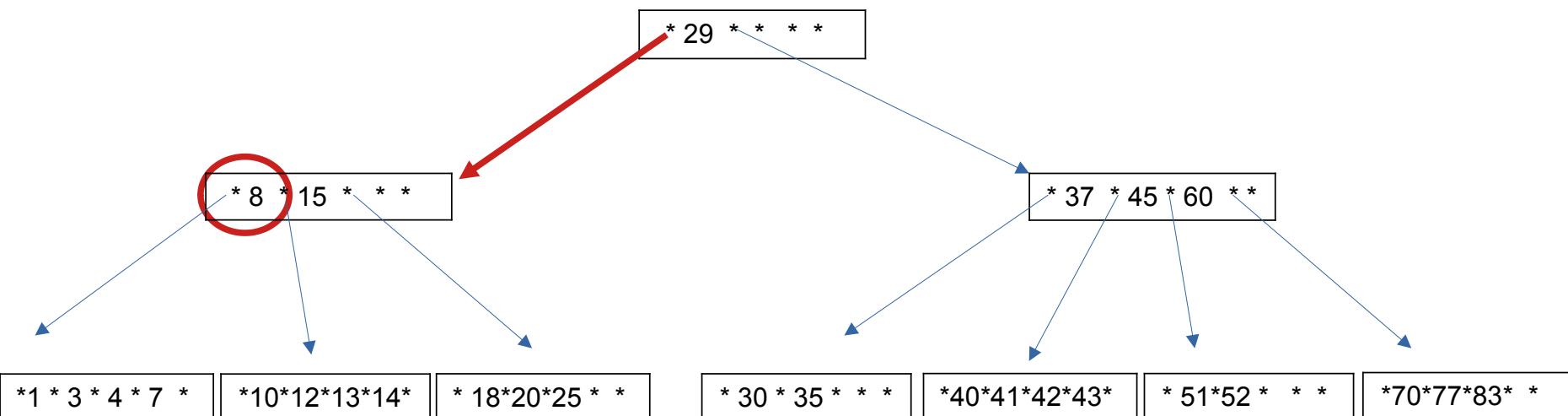
Árvore B - Inserção

Exemplo: Inserir a chave 11



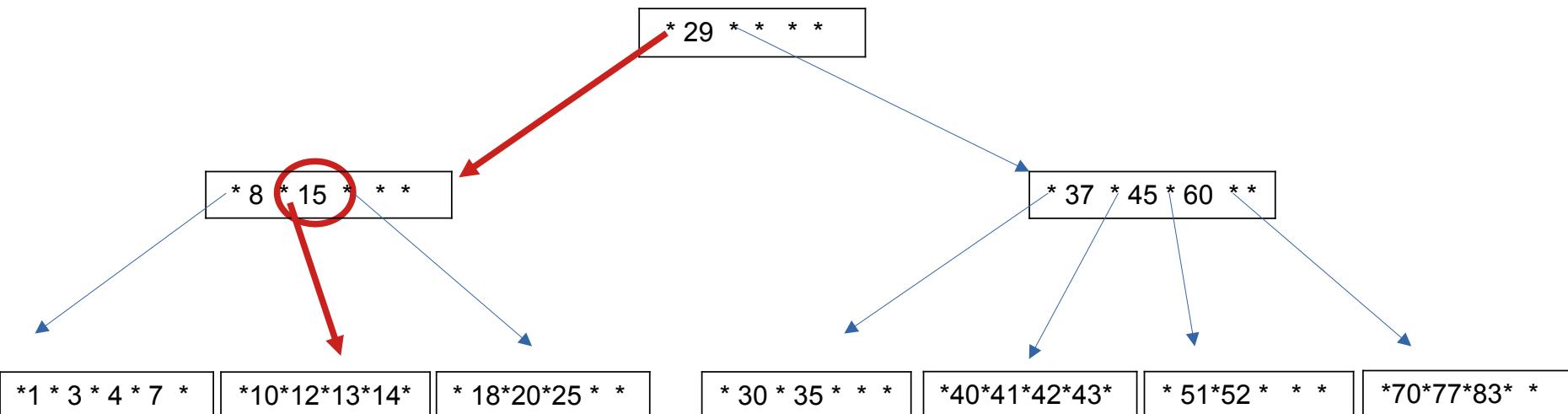
Árvore B - Inserção

Exemplo: Inserir a chave 11



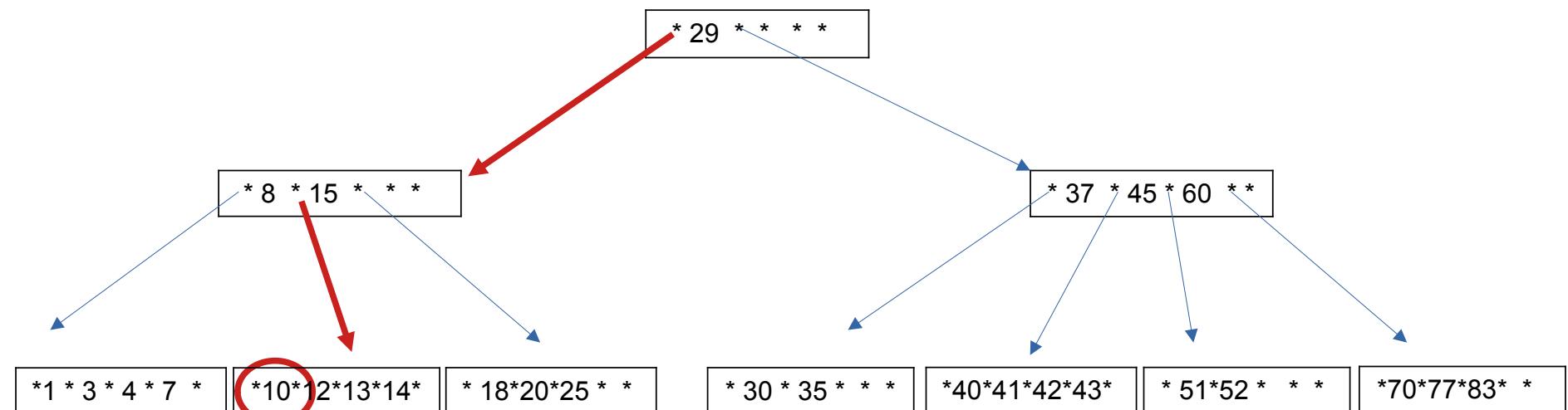
Árvore B - Inserção

Exemplo: Inserir a chave 11



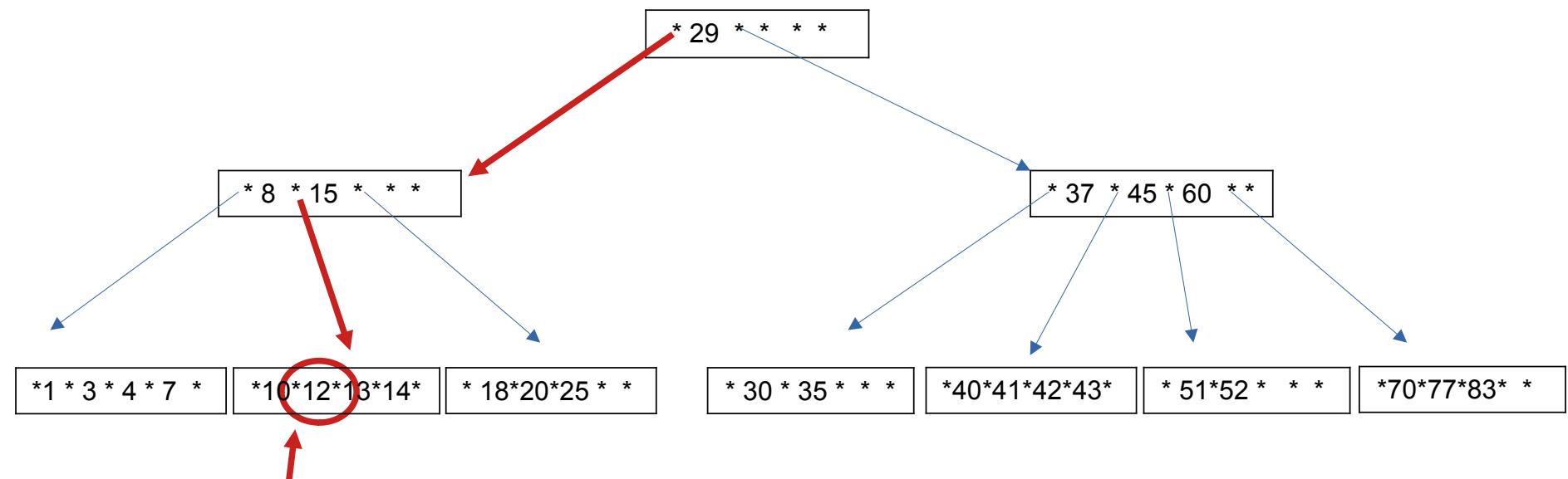
Árvore B - Inserção

Exemplo: Inserir a chave 11



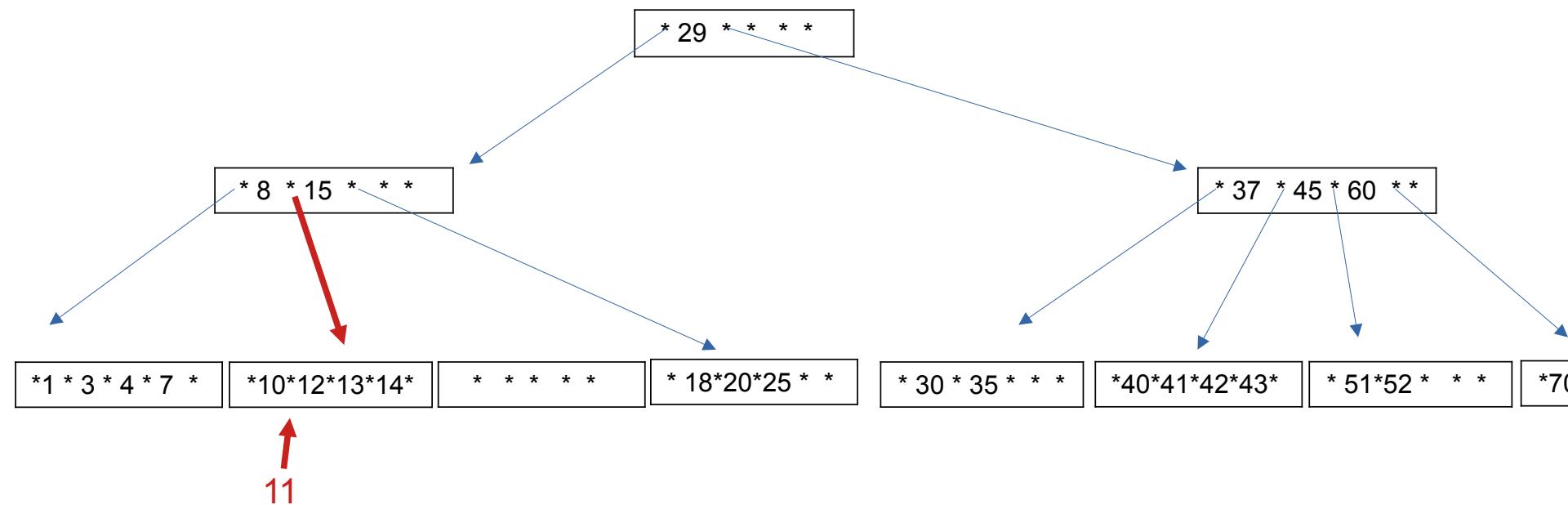
Árvore B - Inserção

Exemplo: Inserir a chave 11



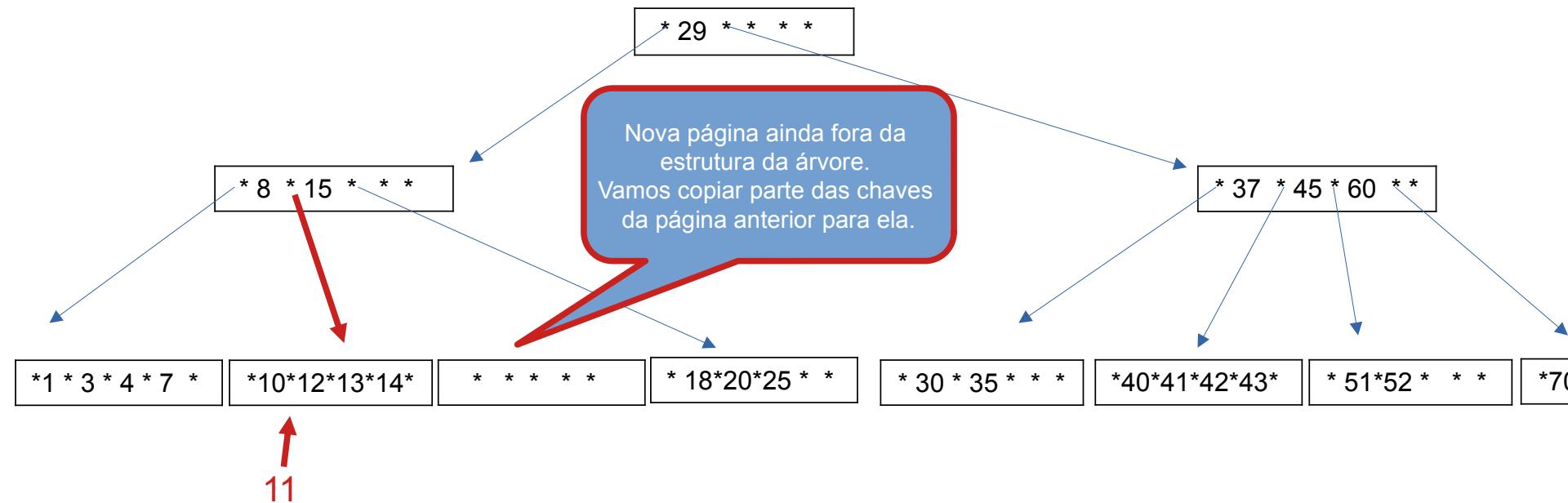
Árvore B - Inserção

Exemplo: Inserir a chave 11



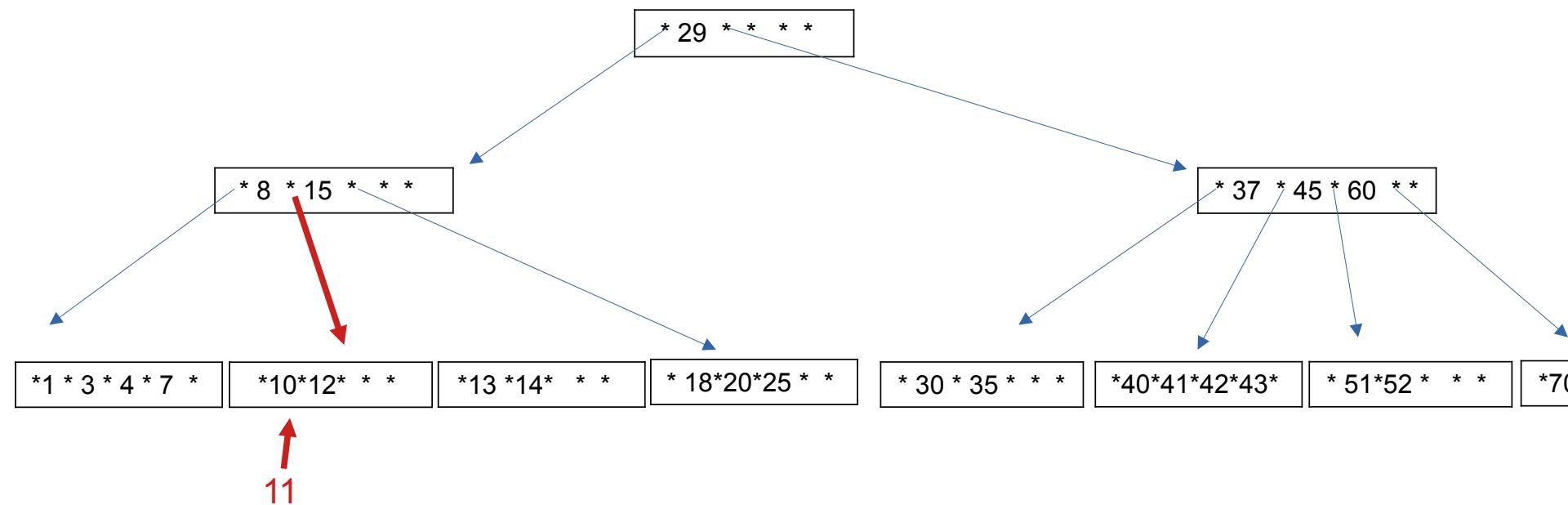
Árvore B - Inserção

Exemplo: Inserir a chave 11



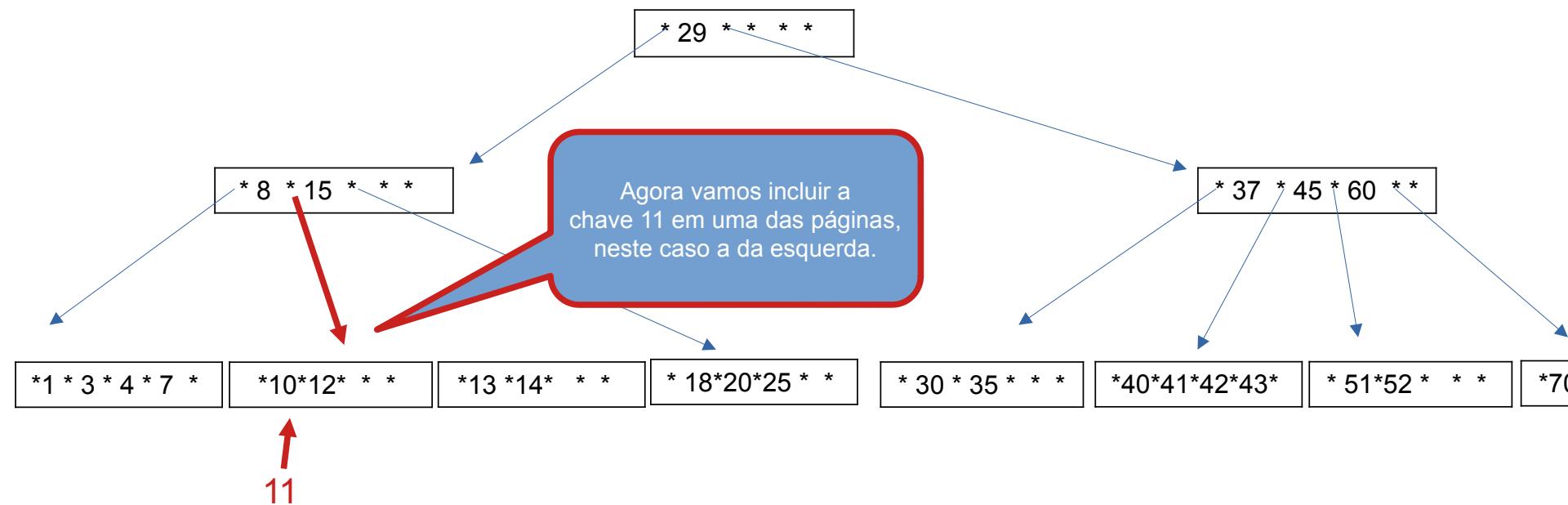
Árvore B - Inserção

Exemplo: Inserir a chave 11



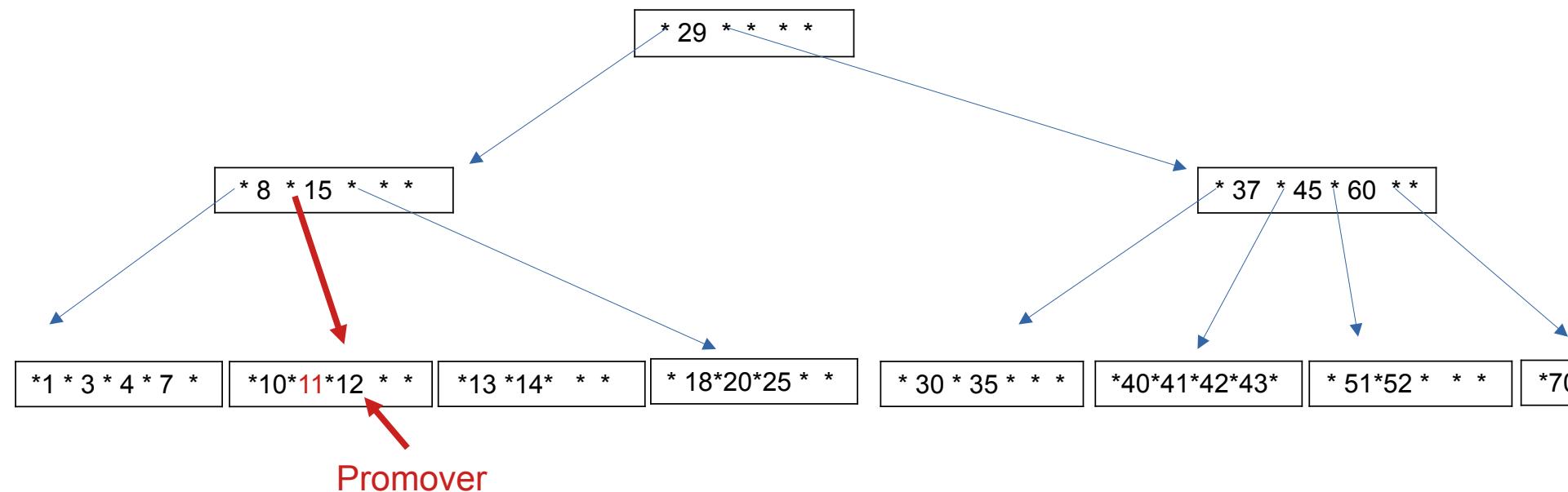
Árvore B - Inserção

Exemplo: Inserir a chave 11



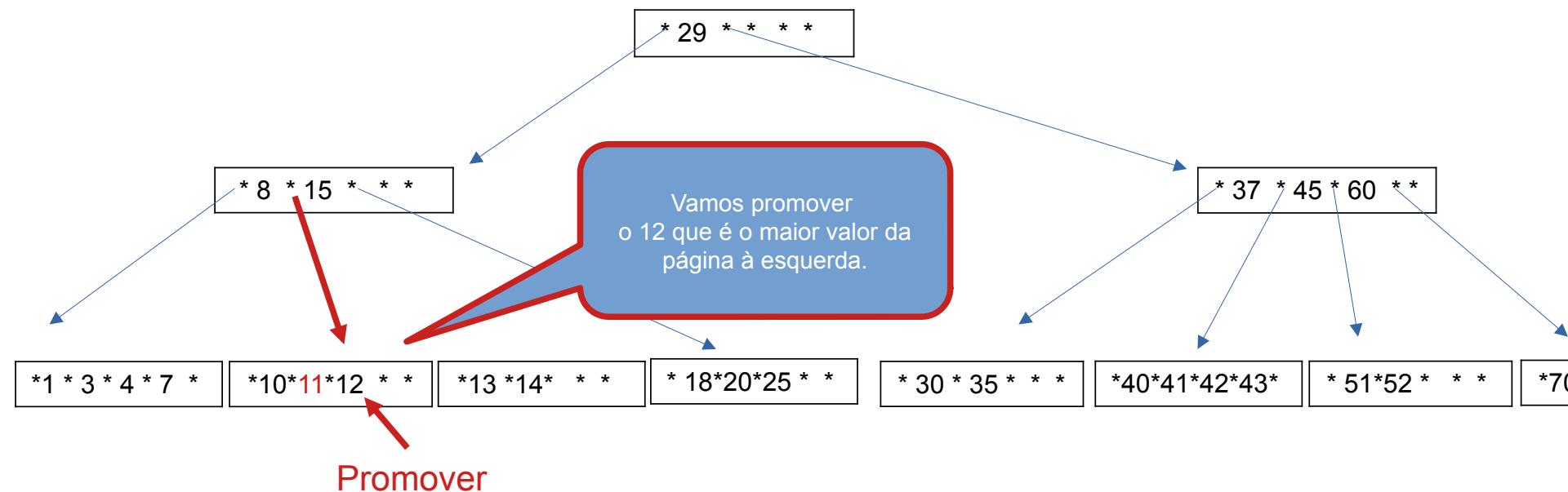
Árvore B - Inserção

Exemplo: Inserir a chave 11



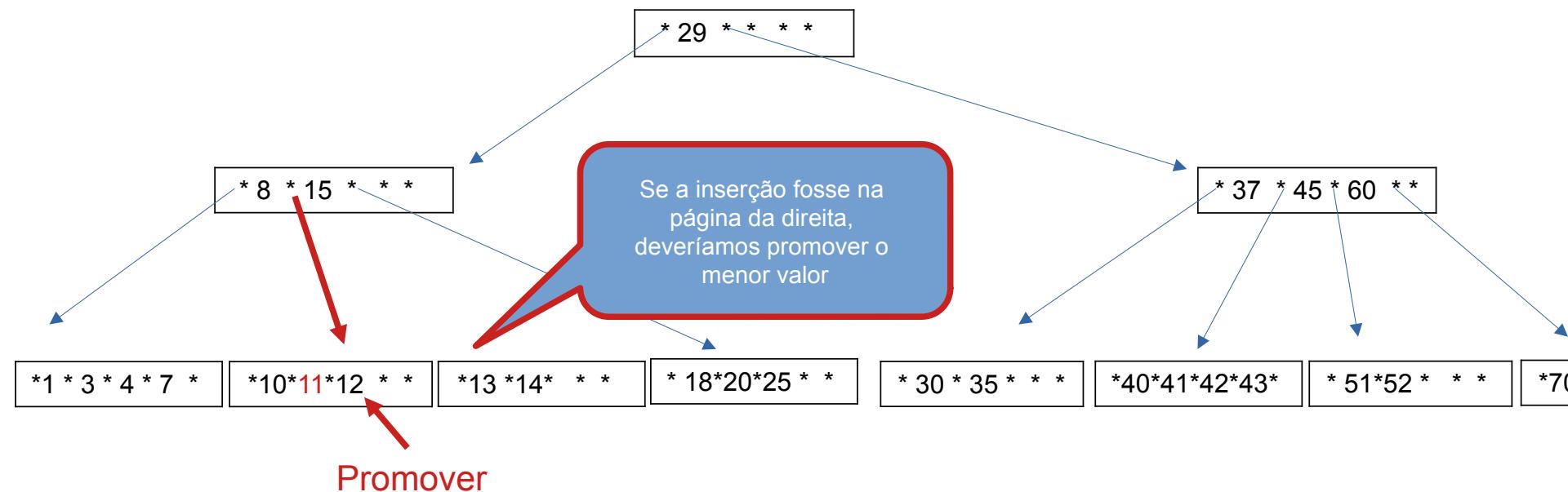
Árvore B - Inserção

Exemplo: Inserir a chave 11



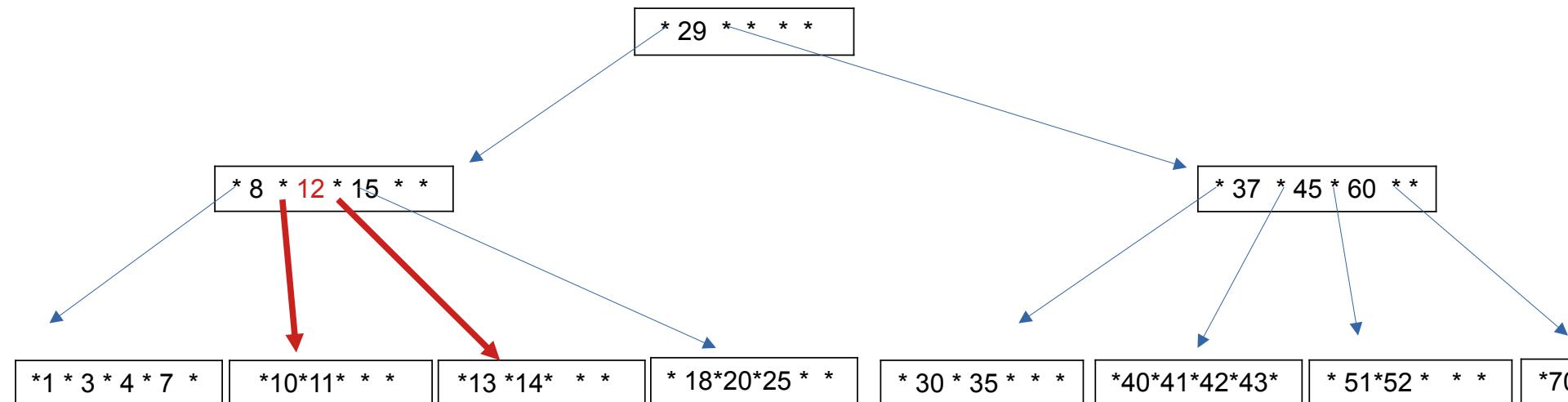
Árvore B - Inserção

Exemplo: Inserir a chave 11



Árvore B - Inserção

Exemplo: Inserir a chave 11



Construindo árvore



PUC Minas

Características

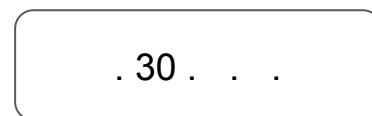
Árvore de ordem 4 que recebe 10 inserções

Inserir os seguintes elementos:

30, 40, 50, 20, 10, 5, 15, 2, 7, 1

Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 30

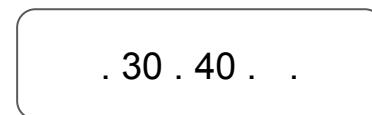


Total de elementos:

1

Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 40



Total de elementos:

2

Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 50

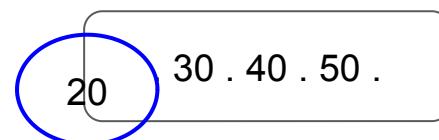
. 30 . 40 . 50 .

Total de elementos:

3

Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 20



Total de elementos:

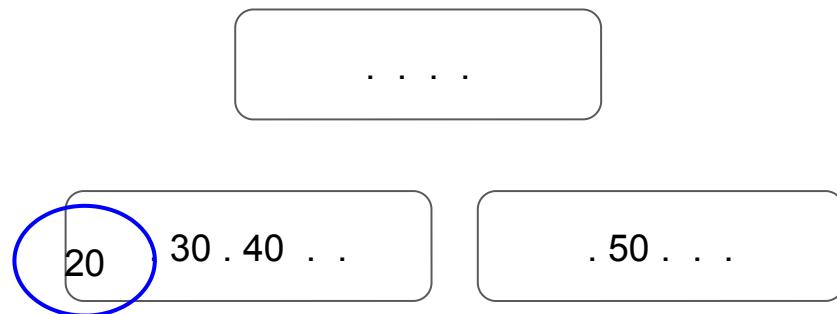
4

Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 20

Total de elementos:

4

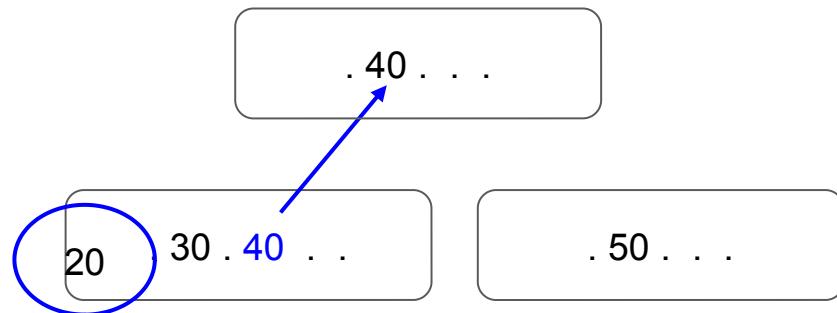


Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 20

Total de elementos:

4

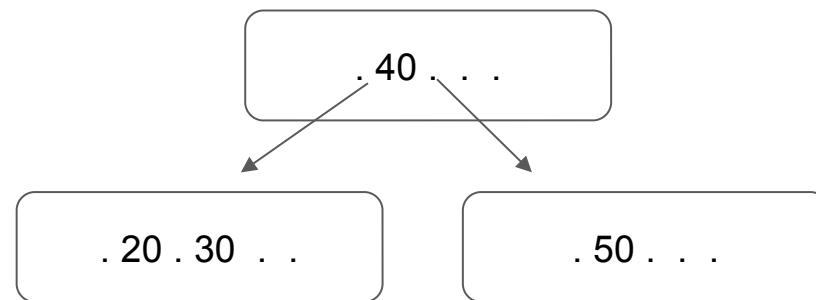


Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 20

Total de elementos:

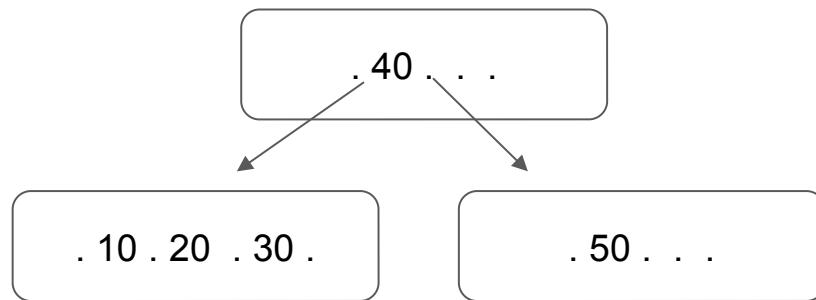
4



Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 10

Total de elementos:	5
---------------------	---

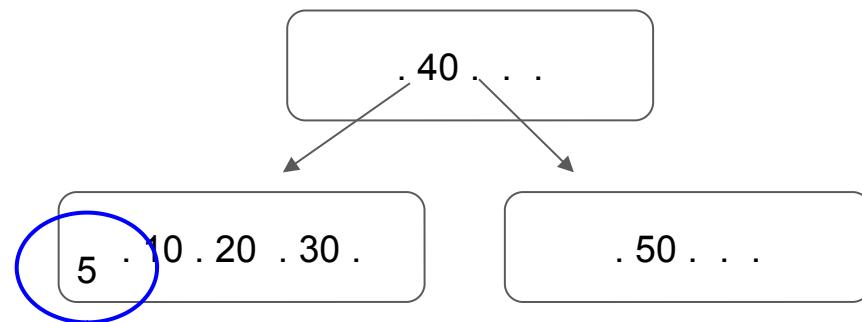


Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 5

Total de elementos:

6

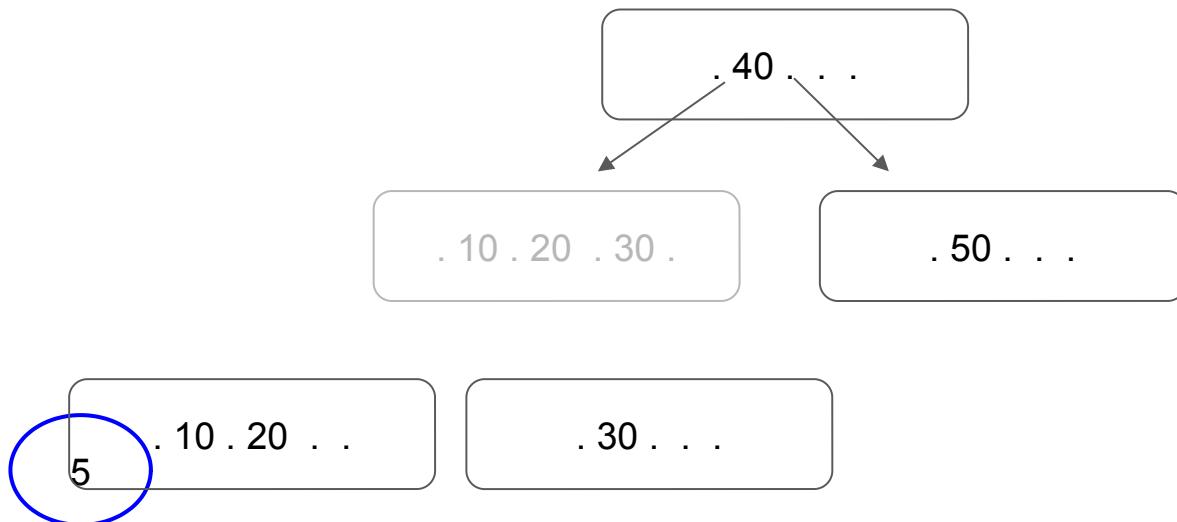


Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 5

Total de elementos:

6

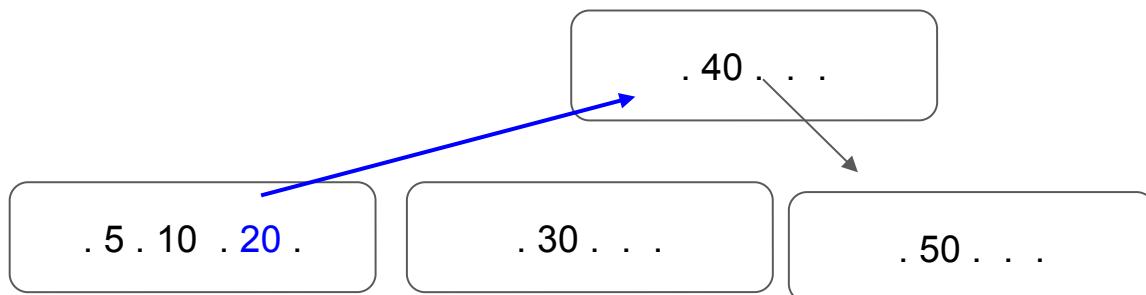


Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 5

Total de elementos:

6

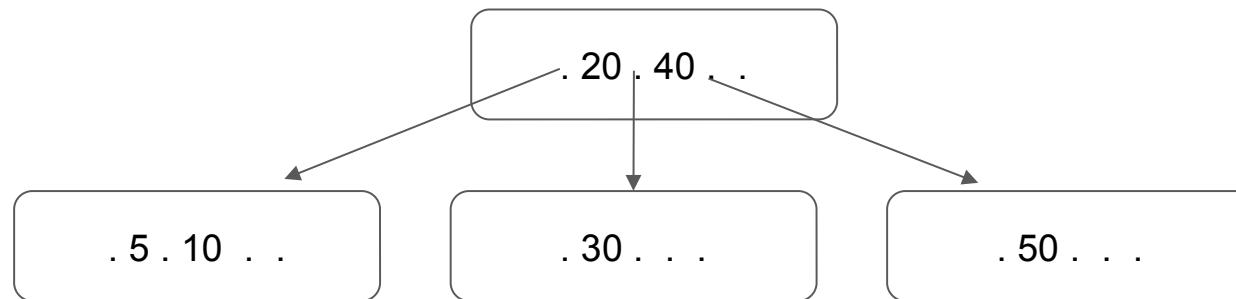


Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 5

Total de elementos:

6

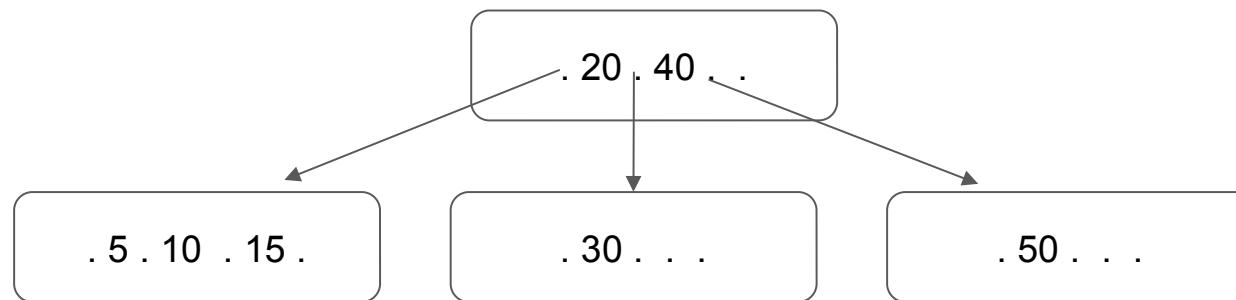


Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 15

Total de elementos:

7

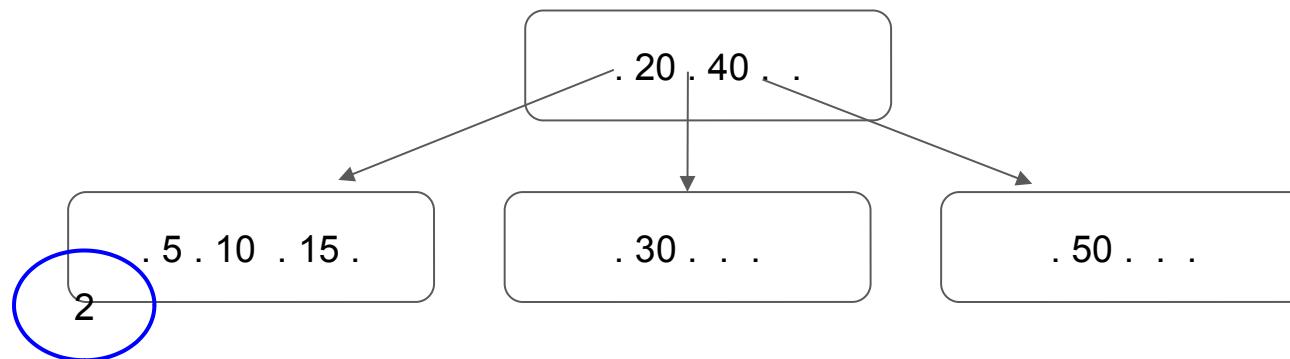


Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 2

Total de elementos:

8

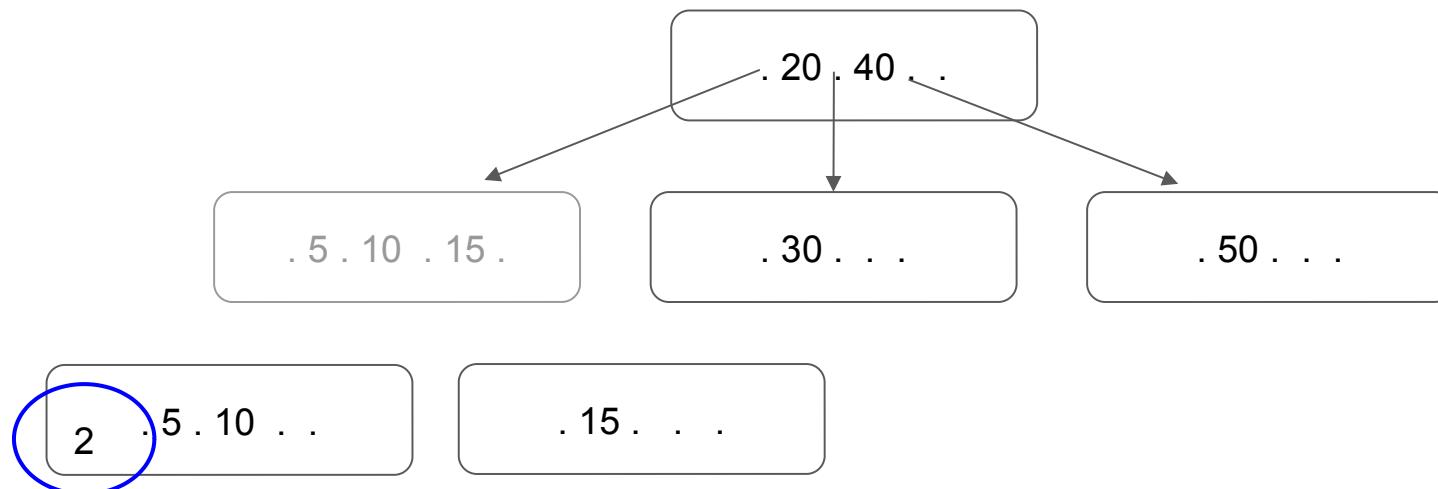


Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 2

Total de elementos:

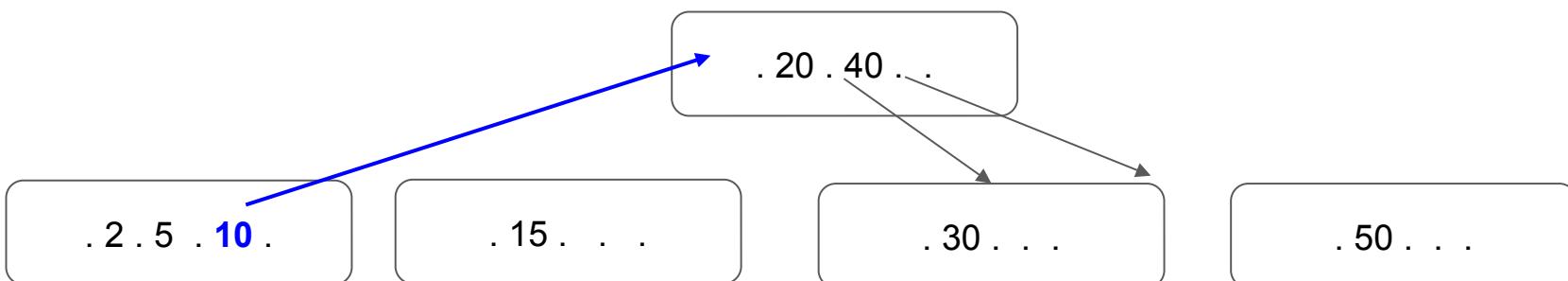
8



Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 2

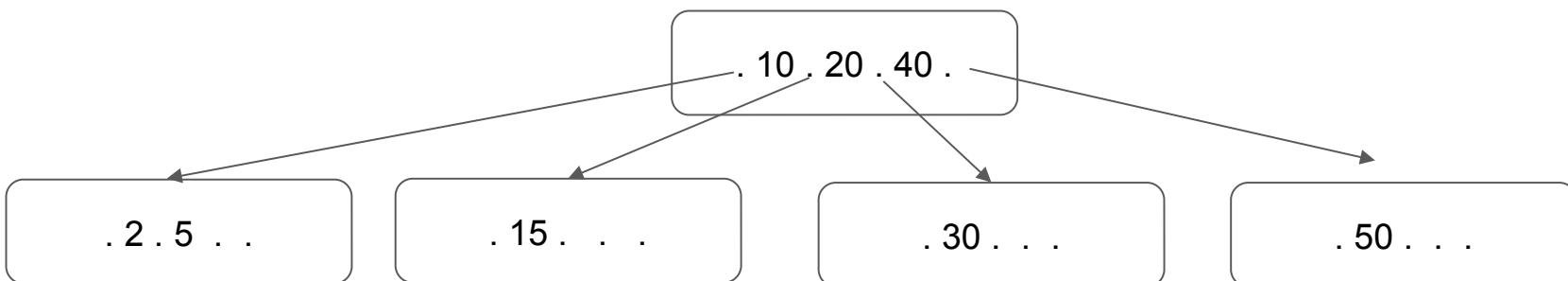
Total de elementos: 8



Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 2

Total de elementos: 8

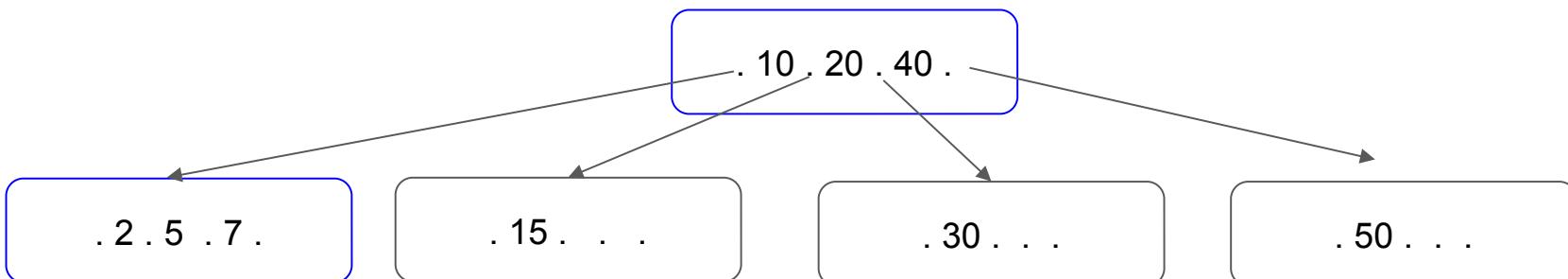


Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 7

Total de elementos:

9

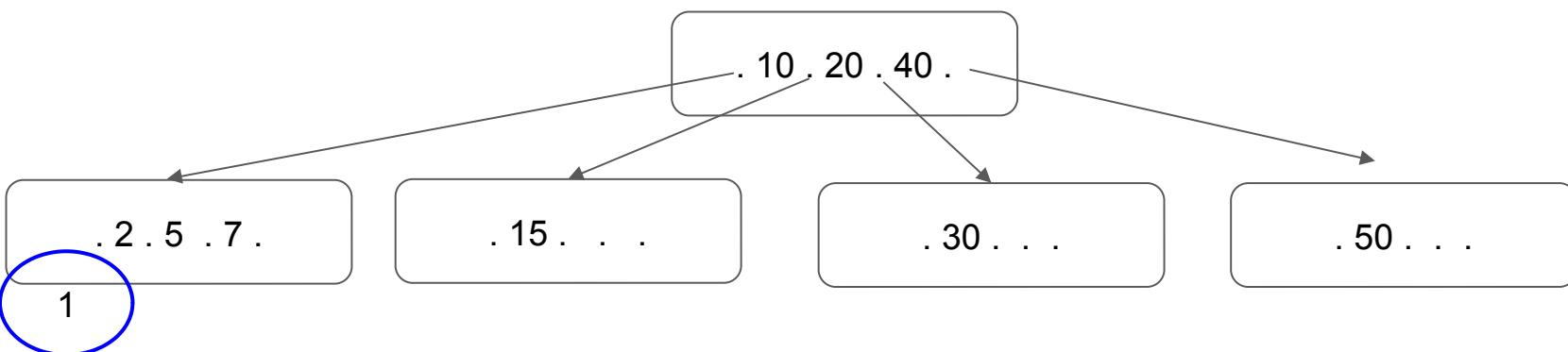


Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 1

Total de elementos:

10

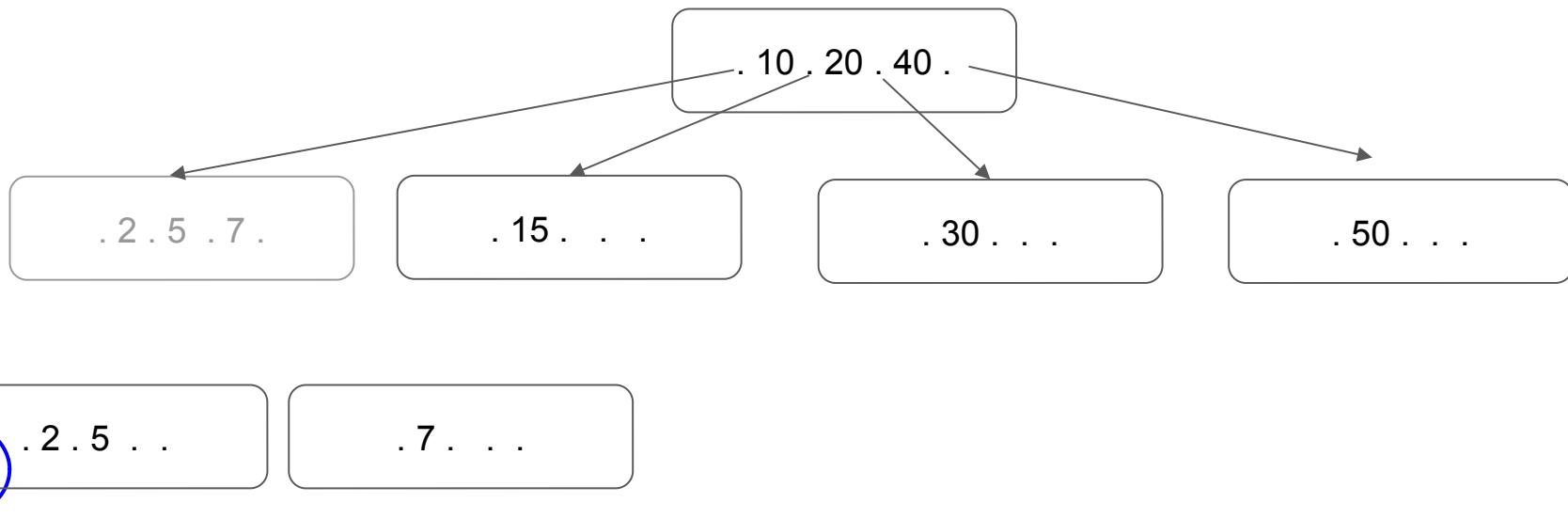


Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 1

Total de elementos:

10

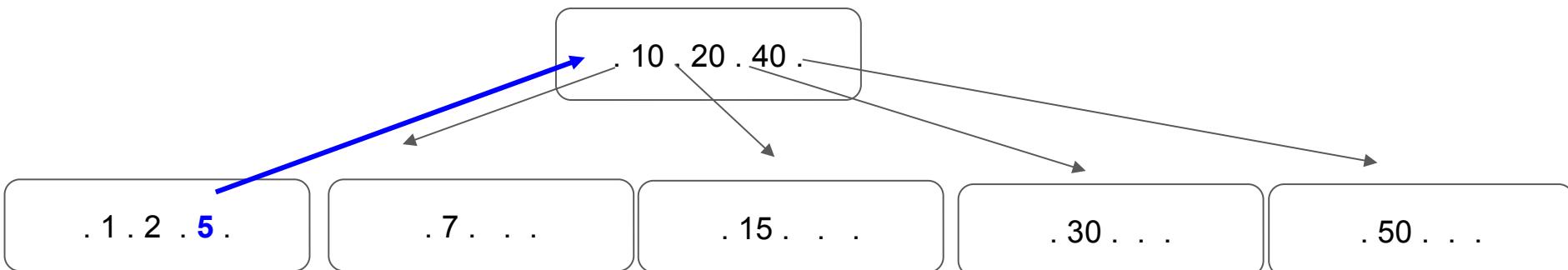


Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 1

Total de elementos:

10

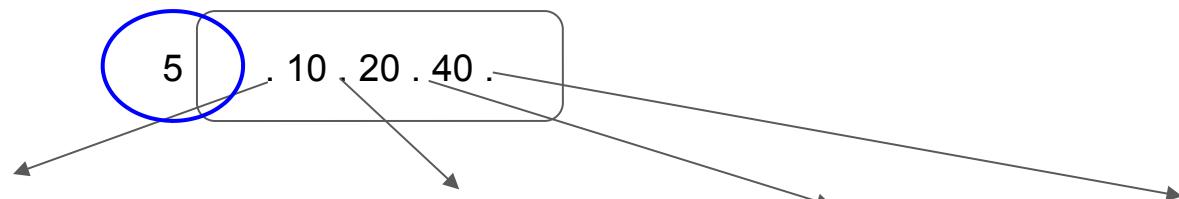


Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 1

Total de elementos:

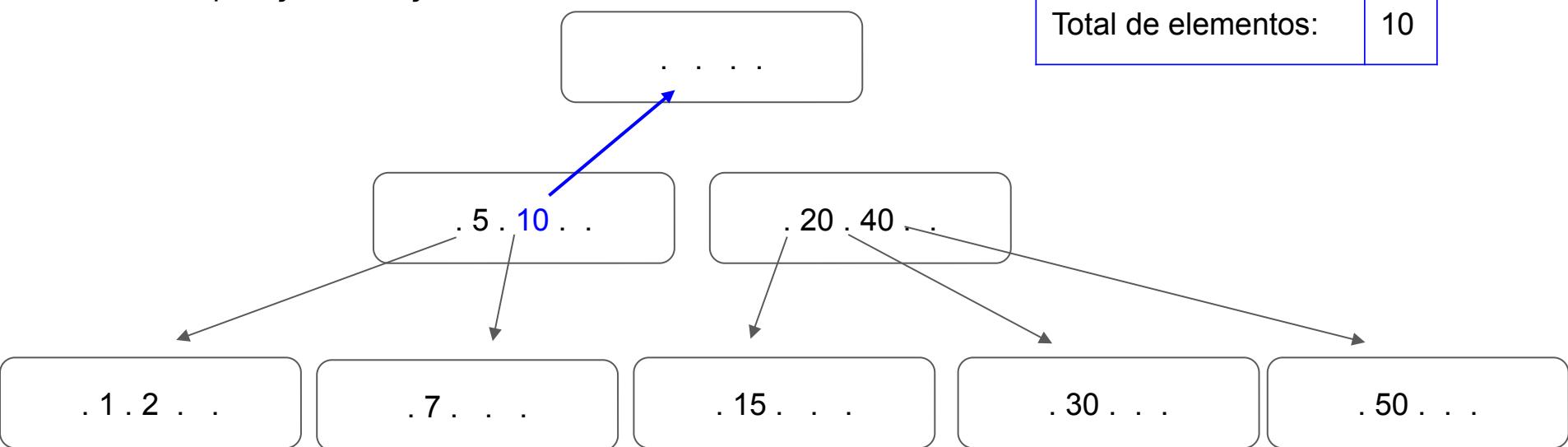
10



Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 1

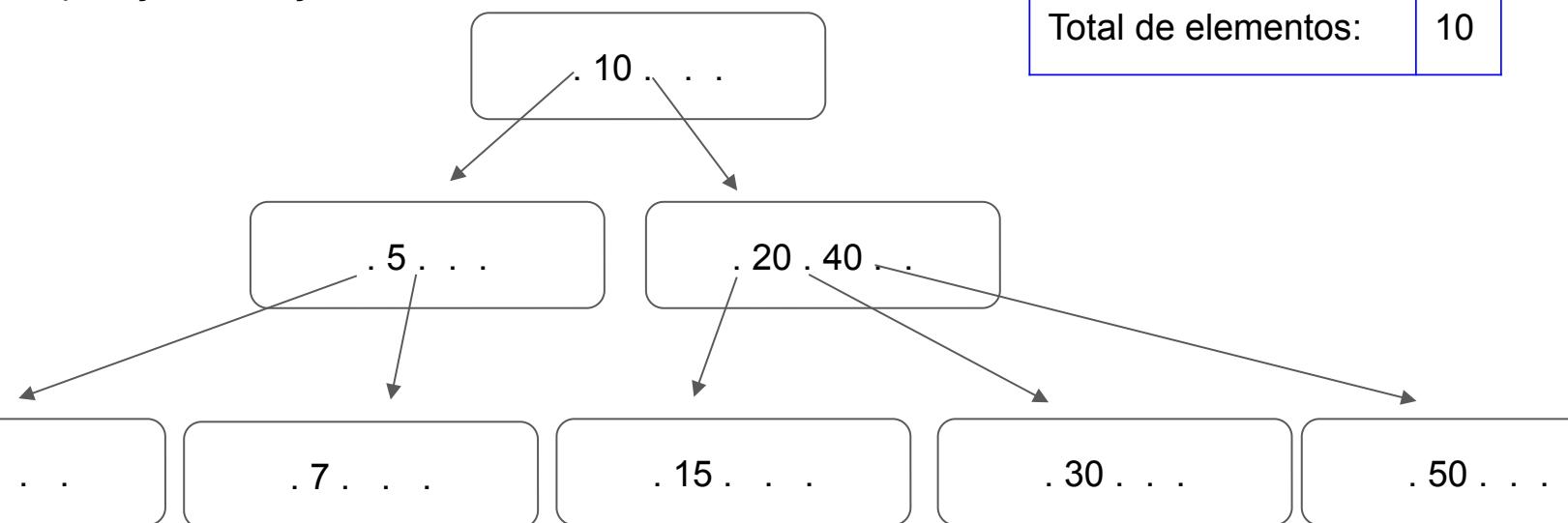
Total de elementos:	10
---------------------	----



Árvore de ordem 4 que recebe 10 inserções

Operação: inserção do elemento de chave 1

Total de elementos:	10
---------------------	----



Remoção em Árvore B



PUC Minas

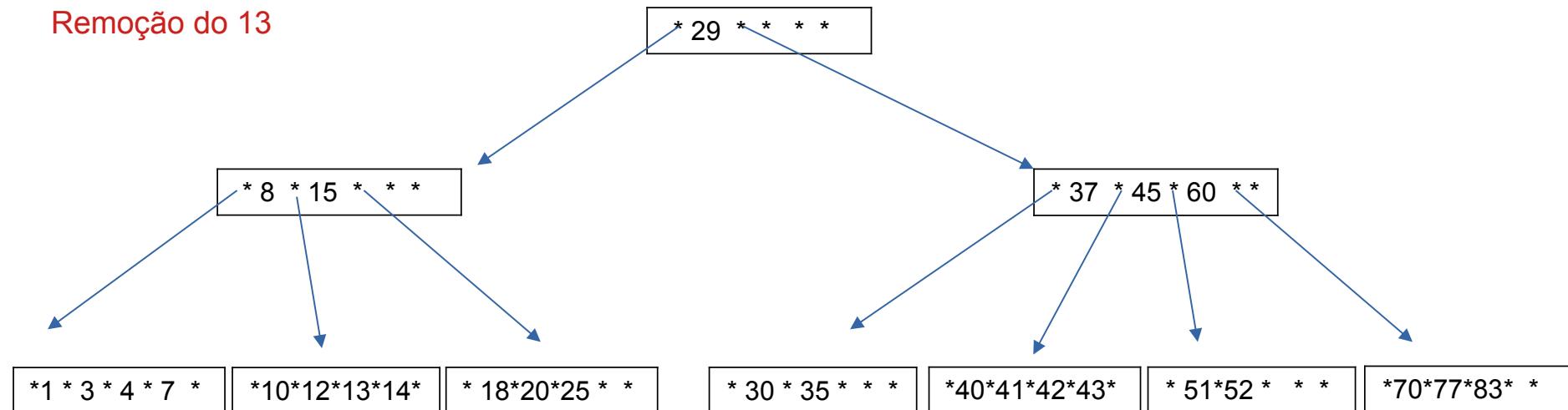
Árvore B - Removendo elemento

- Caso 1: se o elemento ESTIVER em uma folha e a folha mantiver 50% de ocupação, basta removê-lo
- Caso 2: se o elemento NÃO ESTIVER em uma folha, trocá-lo pelo seu antecessor
- Caso 3: se a folha ficar com menos de 50% de ocupação, mas a página irmã puder ceder uma chave
- Caso 4: se a folha ficar com menos de 50% de ocupação e as páginas irmãs não puderem ceder uma chave

Árvore B - Remoção

Caso 1: se o elemento ESTIVER em uma folha e a folha mantiver 50% de ocupação, basta removê-lo

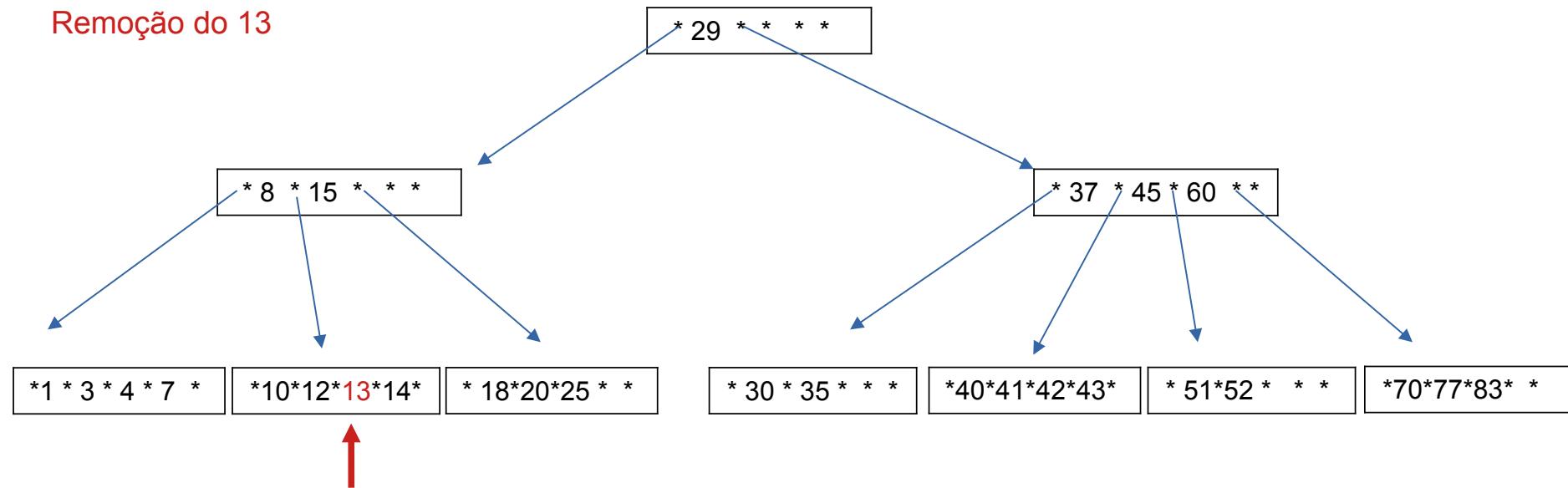
Remoção do 13



Árvore B - Remoção

Caso 1: se o elemento ESTIVER em uma folha e a folha mantiver 50% de ocupação, basta removê-lo

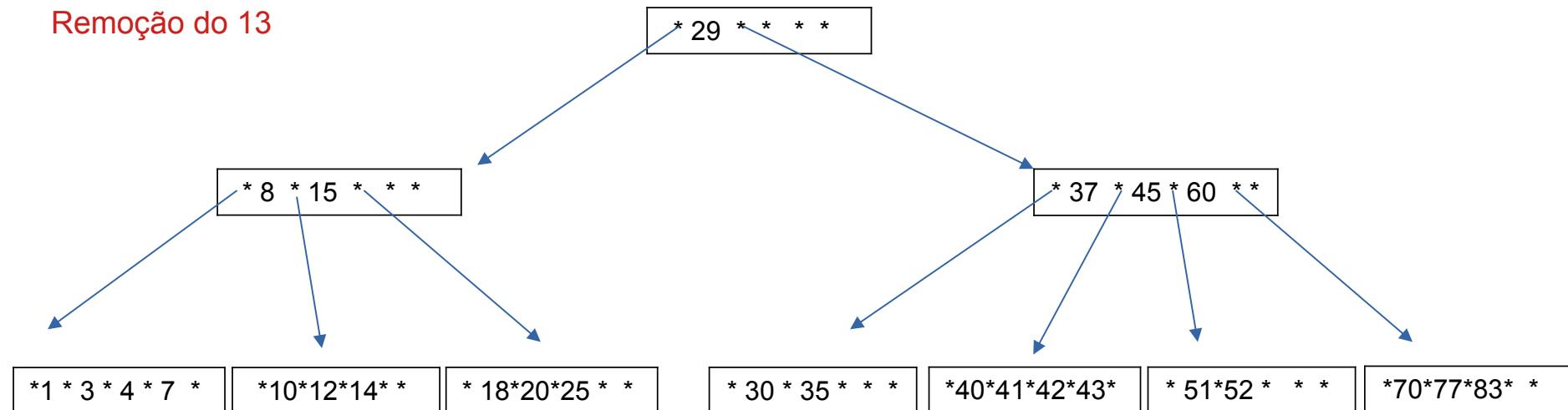
Remoção do 13



Árvore B - Remoção

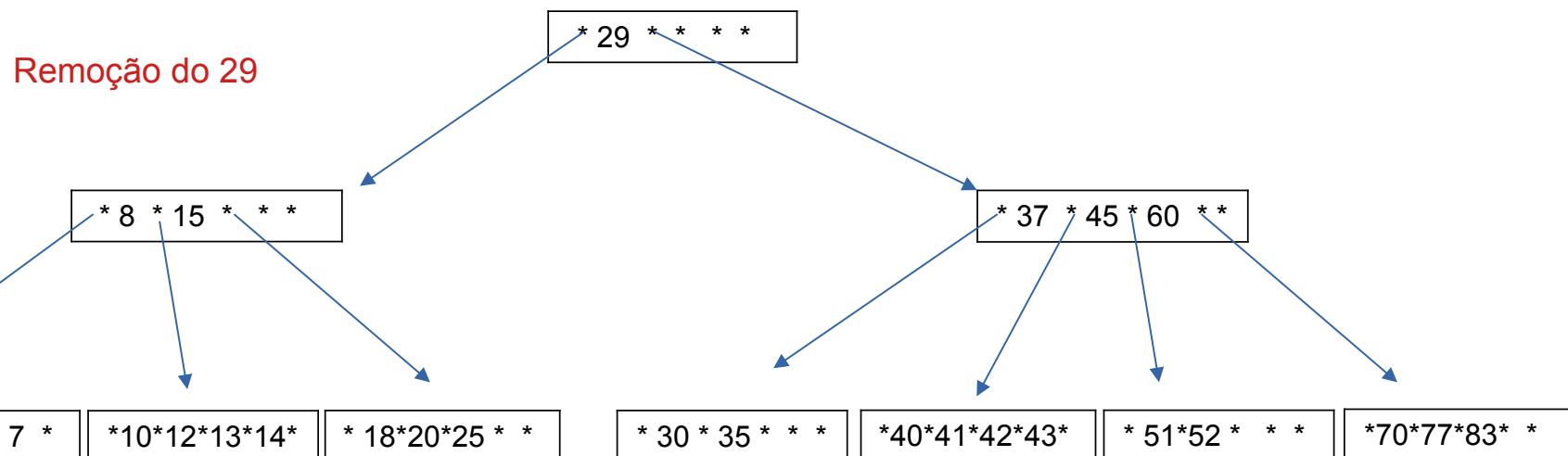
Caso 1: se o elemento ESTIVER em uma folha e a folha mantiver 50% de ocupação, basta removê-lo

Remoção do 13



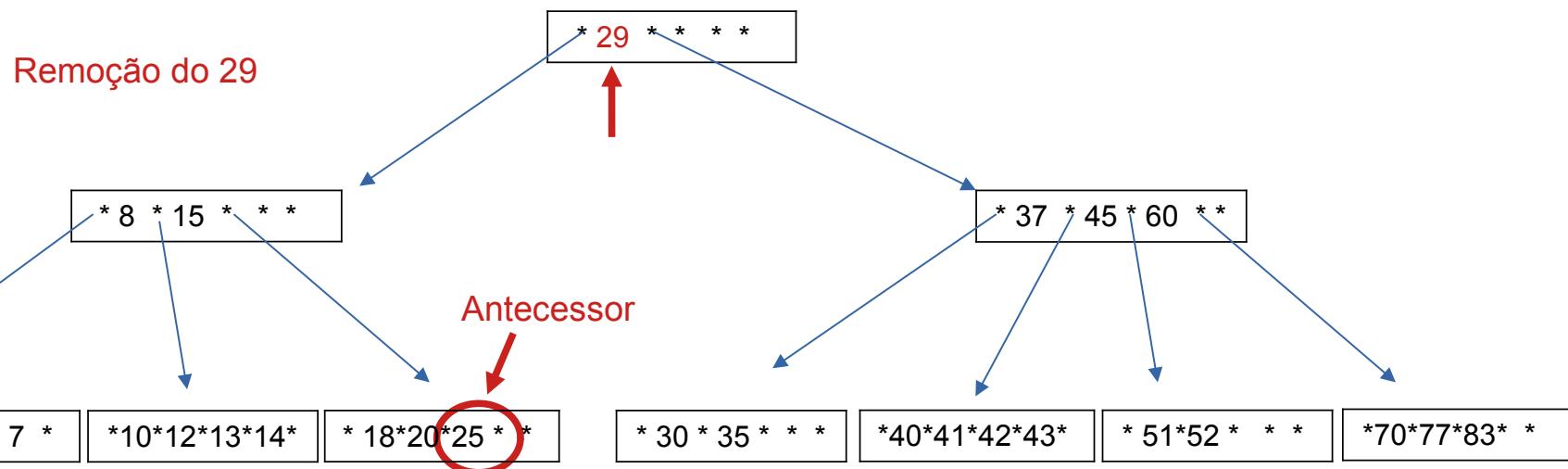
Árvore B - Remoção

Caso 2: se o elemento NÃO ESTIVER em uma folha, trocá-lo pelo seu antecessor



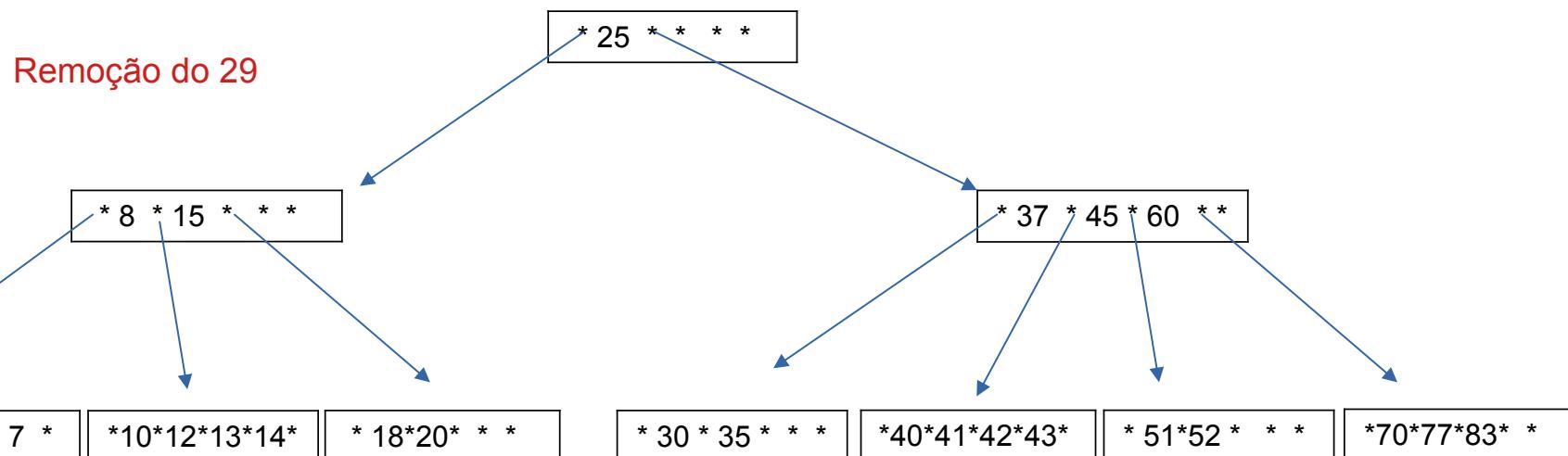
Árvore B - Remoção

Caso 2: se o elemento NÃO ESTIVER em uma folha, trocá-lo pelo seu antecessor



Árvore B - Remoção

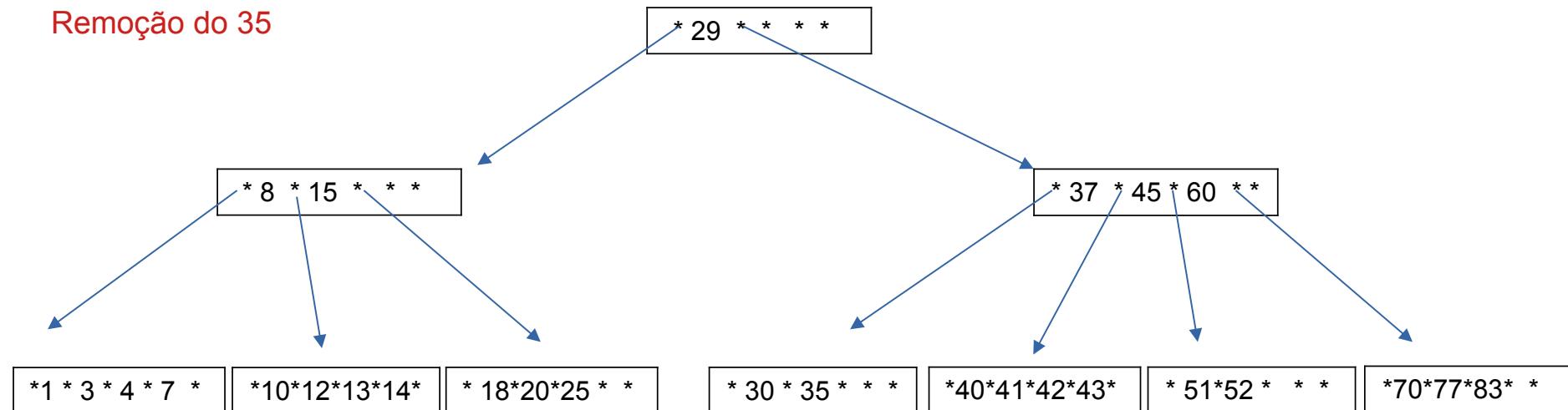
Caso 2: se o elemento NÃO ESTIVER em uma folha, trocá-lo pelo seu antecessor



Árvore B - Remoção

Caso 3: Se a folha ficar com menos de 50% de ocupação, mas a **página irmã puder ceder** uma chave

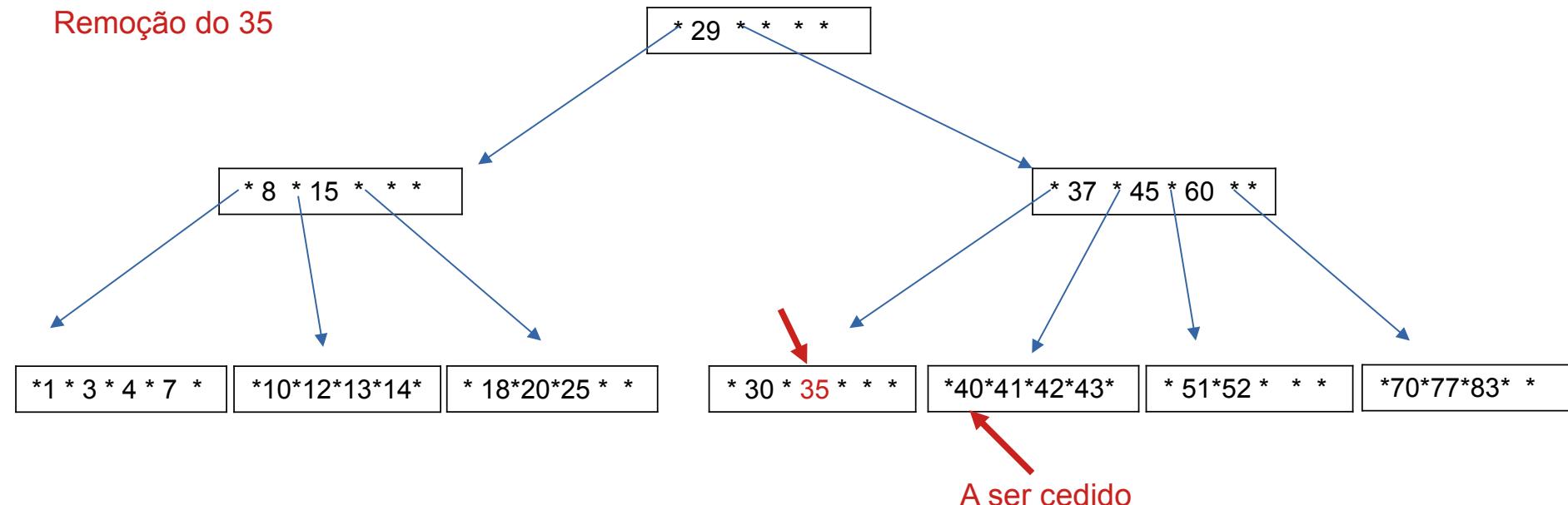
Remoção do 35



Árvore B - Remoção

Caso 3: Se a folha ficar com menos de 50% de ocupação, mas a página irmã puder ceder uma chave

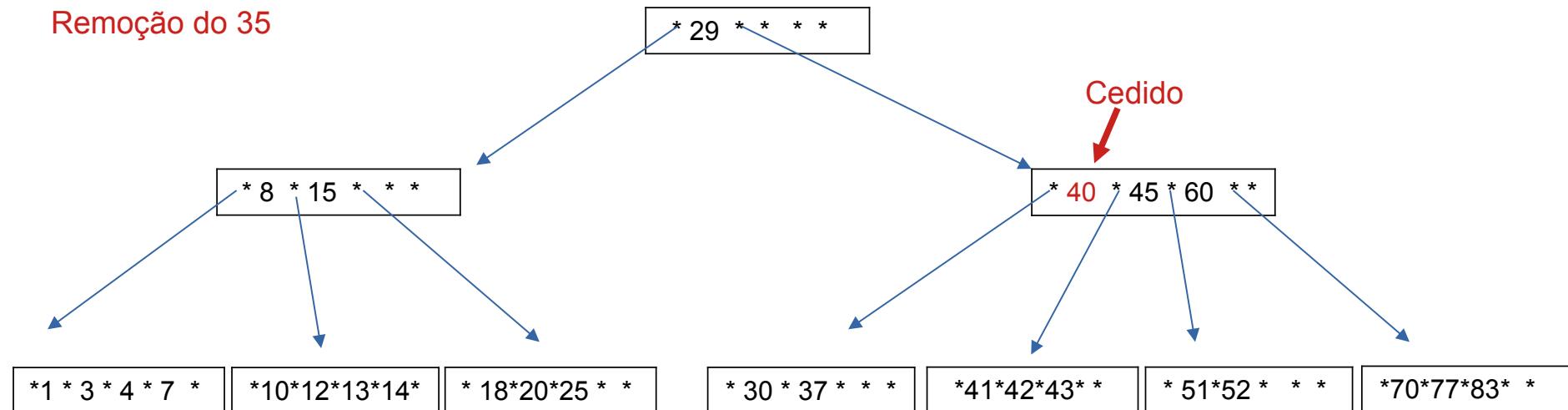
Remoção do 35



Árvore B - Remoção

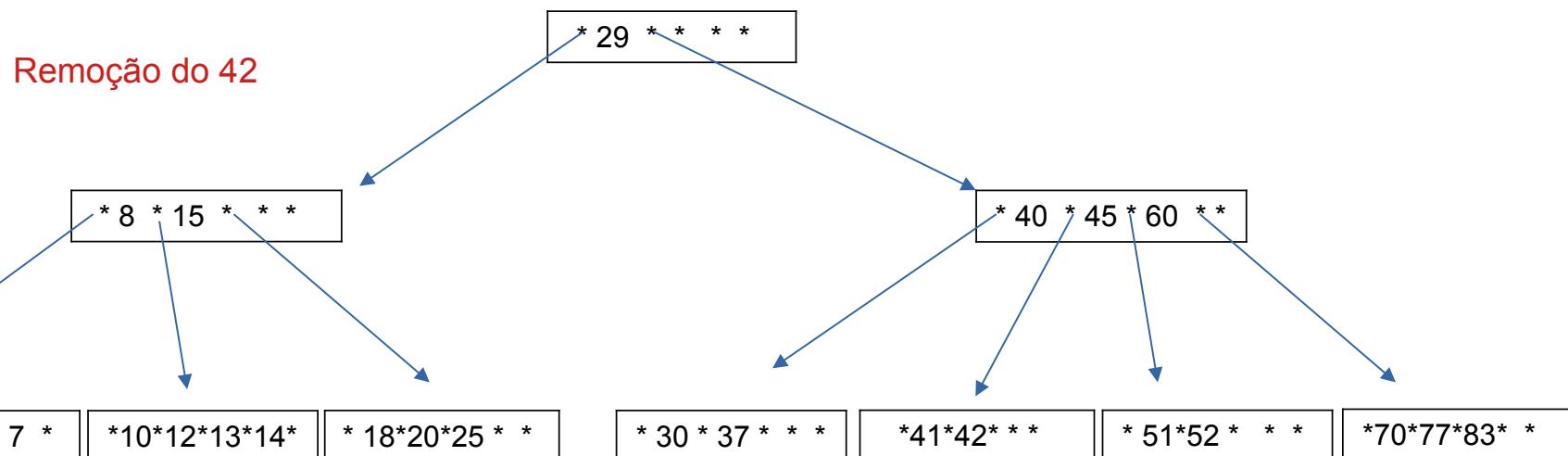
Caso 3: Se a folha ficar com menos de 50% de ocupação, mas a página irmã puder ceder uma chave

Remoção do 35



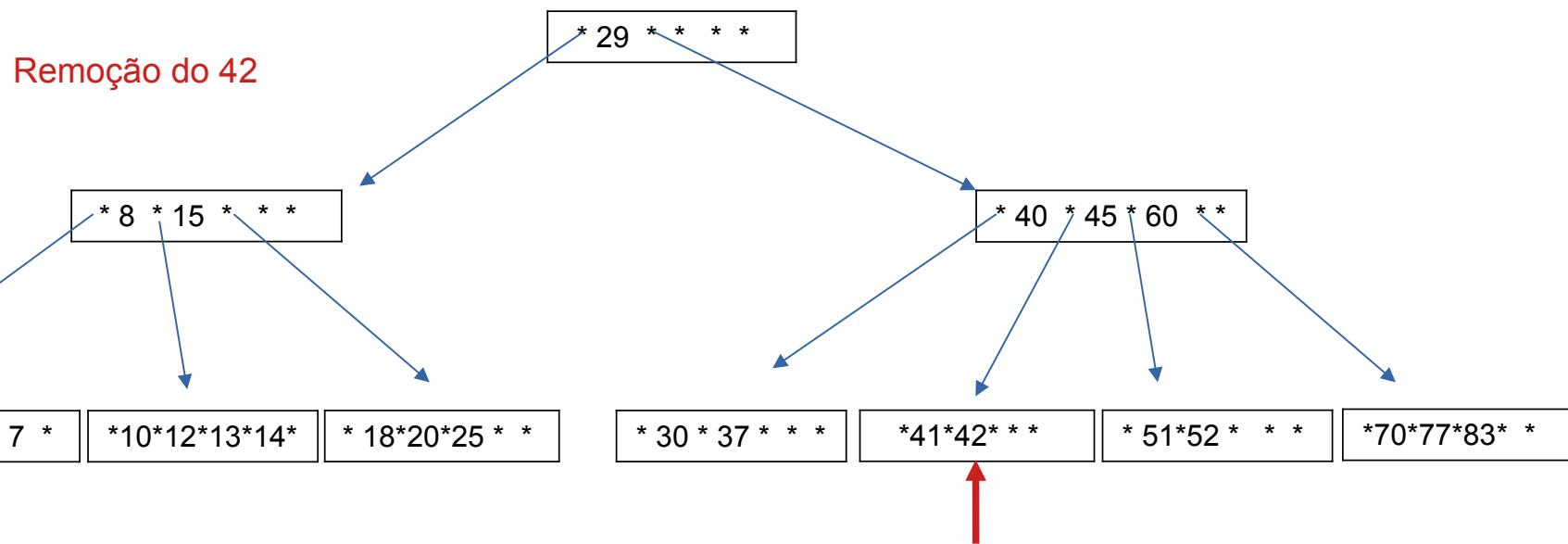
Árvore B - Remoção

Caso 4: se a folha ficar com menos de 50% de ocupação e as páginas irmãs não puderem ceder uma chave



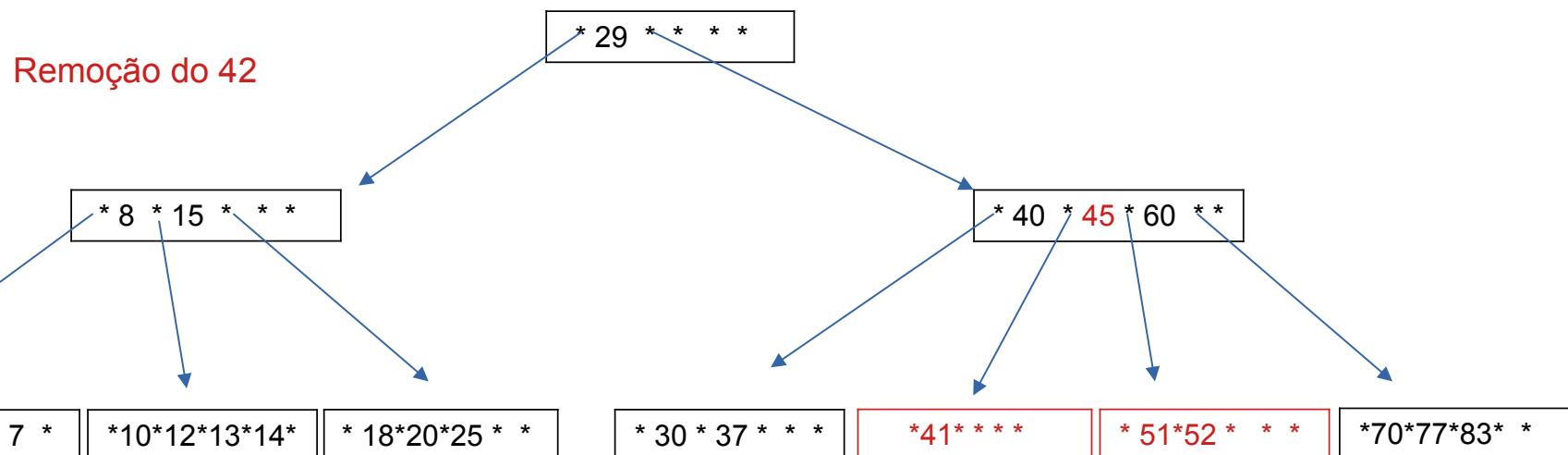
Árvore B - Remoção

Caso 4: se a folha ficar com menos de 50% de ocupação e as páginas irmãs não puderem ceder uma chave



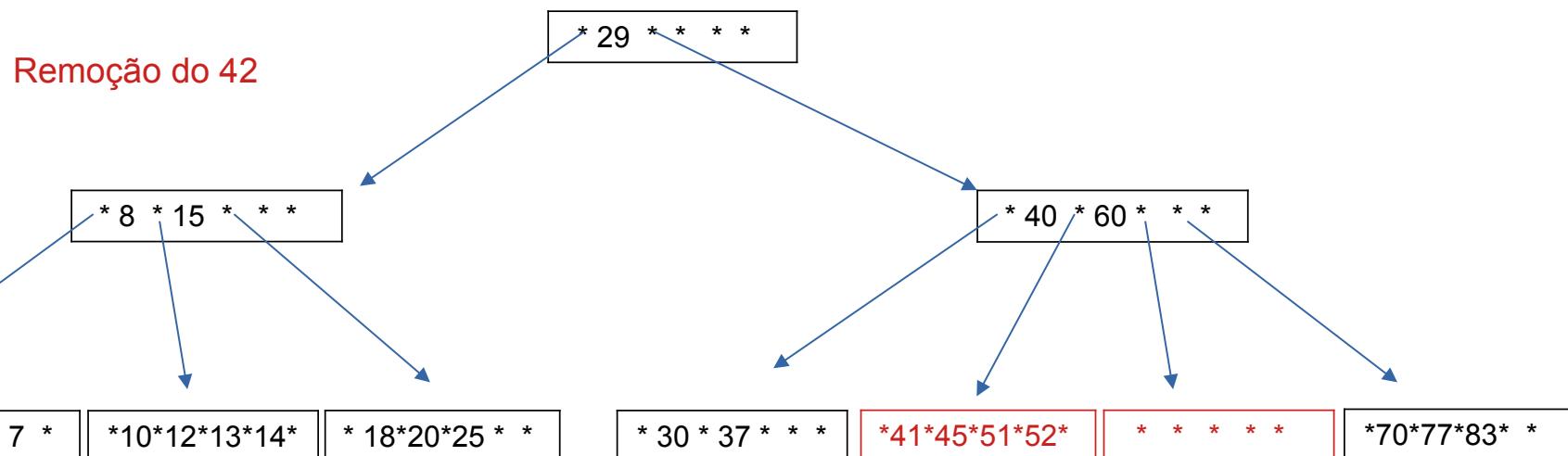
Árvore B - Remoção

Caso 4: se a folha ficar com menos de 50% de ocupação e as páginas irmãs não puderem ceder uma chave



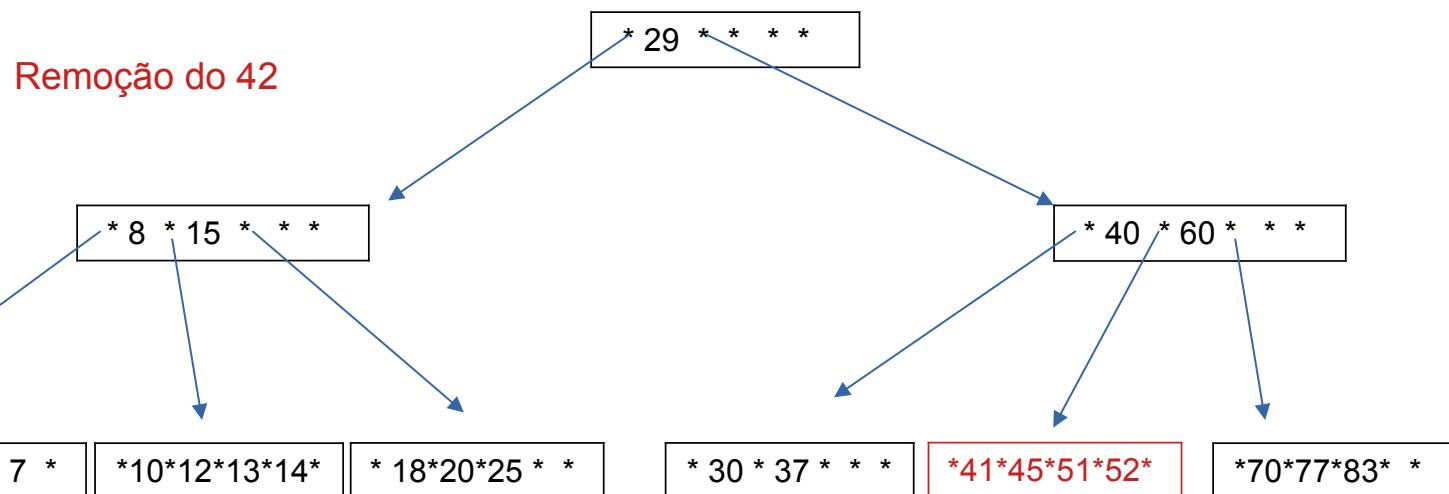
Árvore B - Remoção

Caso 4: se a folha ficar com menos de 50% de ocupação e as páginas irmãs não puderem ceder uma chave



Árvore B - Remoção

Caso 4: se a folha ficar com menos de 50% de ocupação e as páginas irmãs não puderem ceder uma chave



Estrutura da Página (tamanho fixo)



PUC Minas

Estrutura do Índice

N	P0	C0D0	P1	C1D1	P2	C2D2	P3	C3D3	Pn-1	Cn-1 Dn-1	Pn
---	----	------	----	------	----	------	----	------	-------	------	-----------	----

Em que:

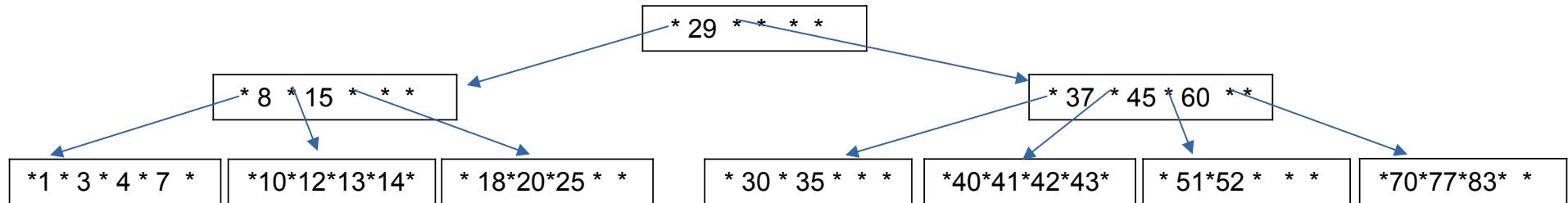
- N número de elementos presentes na página
- Ci chave do registro (geralmente um código)
- Di dados (ex.: endereço do registro no arquivo)
- Pi ponteiro para o i-ésimo filho

Obs: Registros de tamanho fixo armazenados em um arquivo

Estrutura do Índice

Ponteiro da raiz

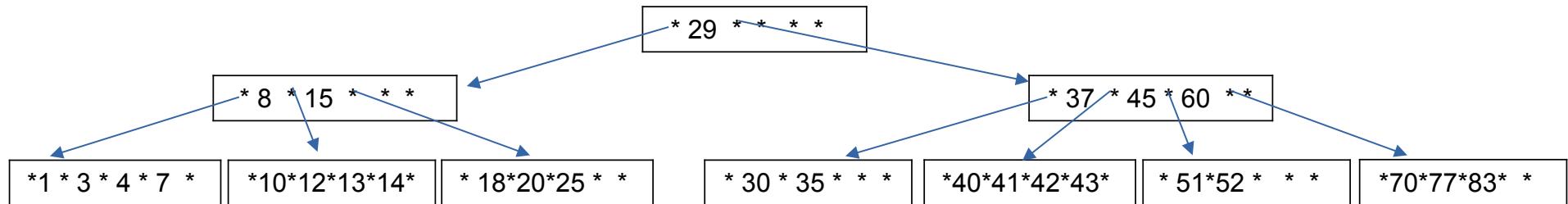
0	296													
8	2	104	8	D0	200	15	D1	392			-1			-1
104	4	-1	1	D0	-1	3	D1	-1	4	D2	-1	7	D3	-1
200	4	-1	10	D0	-1	12	D1	-1	13	D2	-1	14	D3	-1
296	1	8	29	D0	488			-1			-1			-1
392	3	-1	18	D0	-1	20	D1	-1	25	D2	-1			-1
488	3	584	37	D0	680	45	D1	776	60	D2	872			-1
584	2	-1	30	D0	-1	35	D1	-1			-1			-1
680	4	-1	40	D0	-1	41	D1	-1	42	D2	-1	43	D3	-1
776	2	-1	51	D0	-1	52	D1	-1			-1			-1
872	3	-1	70	D0	-1	77	D1	-1	83	D2	-1			-1



Estrutura do Índice

Número de elementos na página

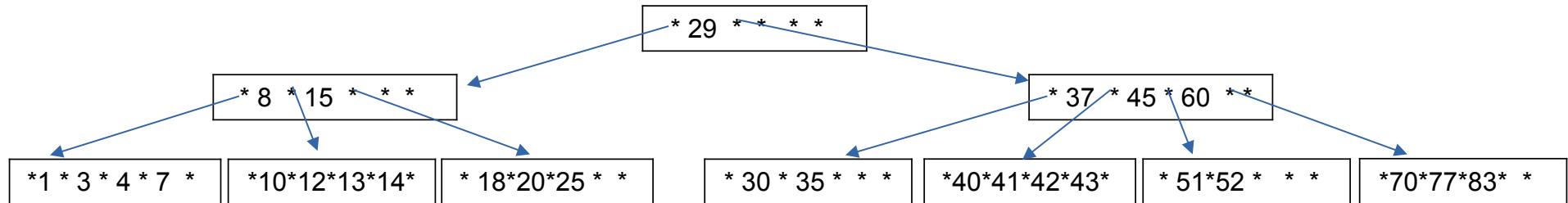
0	296													
8	2	104	8	D0	200	15	D1	392		-1			-1	
104	4	-1	1	D0	-1	3	D1	-1	4	D2	-1	7	D3	-1
200	4	-1	10	D0	-1	12	D1	-1	13	D2	-1	14	D3	-1
296	1	8	29	D0	488			-1			-1			-1
392	3	-1	18	D0	-1	20	D1	-1	25	D2	-1			-1
488	3	584	37	D0	680	45	D1	776	60	D2	872			-1
584	2	-1	30	D0	-1	35	D1	-1			-1			-1
680	4	-1	40	D0	-1	41	D1	-1	42	D2	-1	43	D3	-1
776	2	-1	51	D0	-1	52	D1	-1			-1			-1
872	3	-1	70	D0	-1	77	D1	-1	83	D2	-1			-1



Estrutura do Índice

Pi ponteiro para
o i-ésimo filho

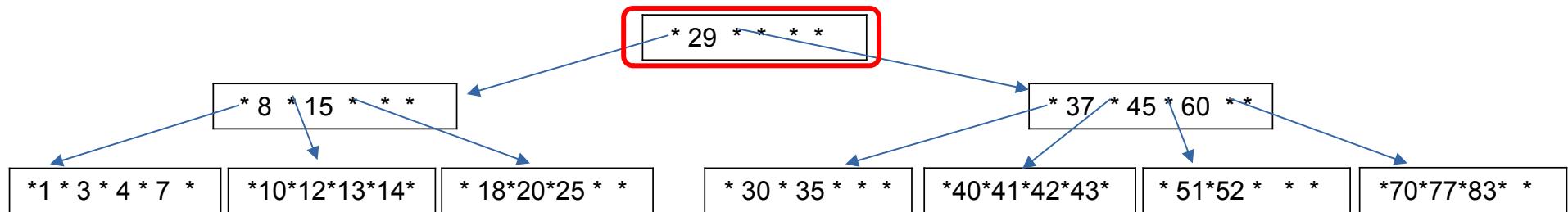
0	296													
8	2	104	8	D0	200	15	D1	392			-1			-1
104	4	-1	1	D0	-1	3	D1	-1	4	D2	-1	7	D3	-1
296	4	-1	10	D0	-1	12	D1	-1	13	D2	-1	14	D3	-1
392	1	8	29	D0	488			-1			-1			-1
488	3	584	37	D0	680	45	D1	776	60	D2	872			-1
584	2	-1	30	D0	-1	35	D1	-1			-1			-1
680	4	-1	40	D0	-1	41	D1	-1	42	D2	-1	43	D3	-1
776	2	-1	51	D0	-1	52	D1	-1			-1			-1
872	3	-1	70	D0	-1	77	D1	-1	83	D2	-1			-1



Estrutura do Índice

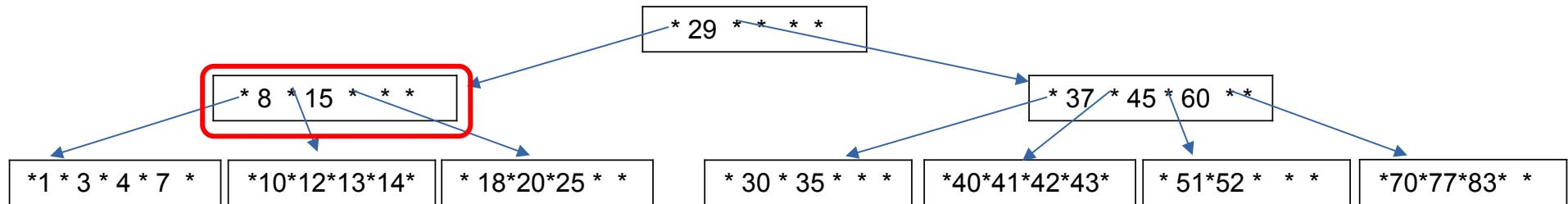


0	296													
8	2	104	8	D0	200	15	D1	392			-1		-1	
104	4	-1	1	D0	-1	3	D1	-1	4	D2	-1	7	D3	-1
200	4	-1	10	D0	-1	12	D1	-1	13	D2	-1	14	D3	-1
296	1	8	29	D0	488			-1			-1			-1
392	3	-1	18	D0	-1	20	D1	-1	25	D2	-1			-1
488	3	584	37	D0	680	45	D1	776	60	D2	872			-1
584	2	-1	30	D0	-1	35	D1	-1			-1			-1
680	4	-1	40	D0	-1	41	D1	-1	42	D2	-1	43	D3	-1
776	2	-1	51	D0	-1	52	D1	-1			-1			-1
872	3	-1	70	D0	-1	77	D1	-1	83	D2	-1			-1



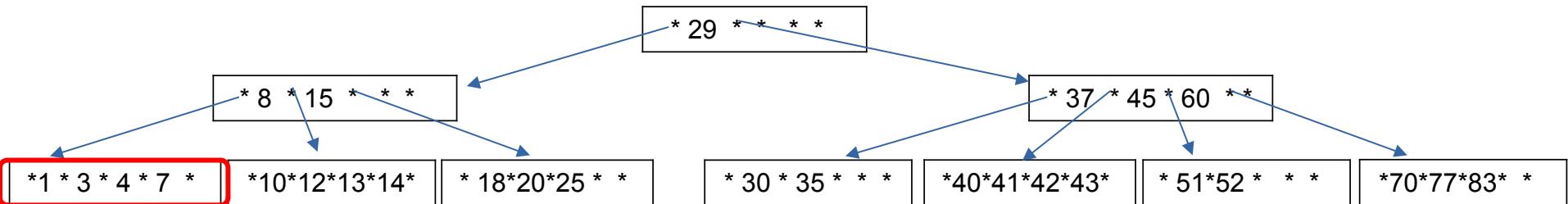
Estrutura do Índice

0	296													
8	2	104	8	D0	200	15	D1	392		-1			-1	
104	4	-1	1	D0	-1	3	D1	-1	4	D2	-1	7	D3	-1
200	4	-1	10	D0	-1	12	D1	-1	13	D2	-1	14	D3	-1
296	1	8	29	D0	488			-1			-1			-1
392	3	-1	18	D0	-1	20	D1	-1	25	D2	-1			-1
488	3	584	37	D0	680	45	D1	776	60	D2	872			-1
584	2	-1	30	D0	-1	35	D1	-1			-1			-1
680	4	-1	40	D0	-1	41	D1	-1	42	D2	-1	43	D3	-1
776	2	-1	51	D0	-1	52	D1	-1			-1			-1
872	3	-1	70	D0	-1	77	D1	-1	83	D2	-1			-1



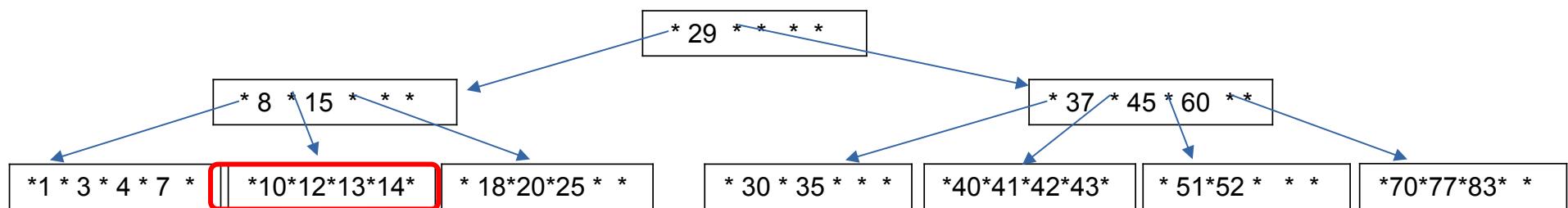
Estrutura do Índice

0	296													
8	2	104	8	D0	200	15	D1	392			-1			-1
104	4	-1	1	D0	-1	3	D1	-1	4	D2	-1	7	D3	-1
200	4	-1	10	D0	-1	12	D1	-1	13	D2	-1	14	D3	-1
296	1	8	29	D0	488			-1			-1			-1
392	3	-1	18	D0	-1	20	D1	-1	25	D2	-1			-1
488	3	584	37	D0	680	45	D1	776	60	D2	872			-1
584	2	-1	30	D0	-1	35	D1	-1			-1			-1
680	4	-1	40	D0	-1	41	D1	-1	42	D2	-1	43	D3	-1
776	2	-1	51	D0	-1	52	D1	-1			-1			-1
872	3	-1	70	D0	-1	77	D1	-1	83	D2	-1			-1



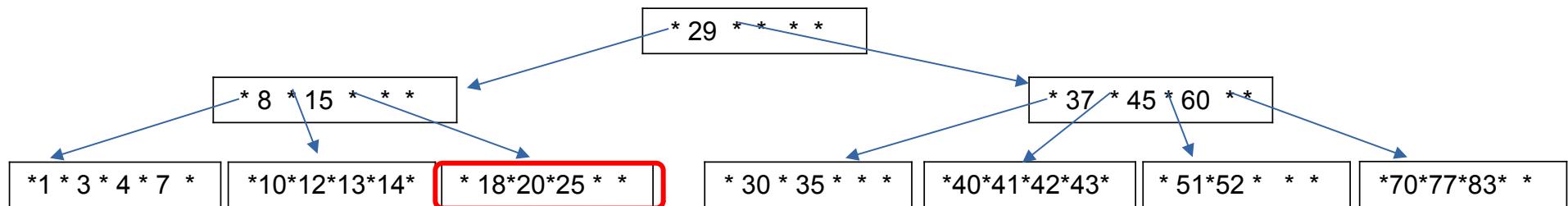
Estrutura do Índice

0	296																			
8	2	104	8	D0	200	15	D1	392				-1							-1	
104	4	-1	1	D0	-1	3	D1	-1	4	D2	-1	7	D3	-1						
200	4	-1	10	D0	-1	12	D1	-1	13	D2	-1	14	D3	-1						
296	1	8	29	D0	488			-1			-1								-1	
392	3	-1	18	D0	-1	20	D1	-1	25	D2	-1								-1	
488	3	584	37	D0	680	45	D1	776	60	D2	872								-1	
584	2	-1	30	D0	-1	35	D1	-1			-1								-1	
680	4	-1	40	D0	-1	41	D1	-1	42	D2	-1	43	D3	-1						
776	2	-1	51	D0	-1	52	D1	-1			-1								-1	
872	3	-1	70	D0	-1	77	D1	-1	83	D2	-1								-1	



Estrutura do Índice

0	296																			
8	2	104	8	D0	200	15	D1	392				-1							-1	
104	4	-1	1	D0	-1	3	D1	-1	4	D2	-1	7	D3	-1						
200	4	-1	10	D0	-1	12	D1	-1	13	D2	-1	14	D3	-1						
296	1	8	29	D0	488			-1			-1			-1						-1
392	3	-1	18	D0	-1	20	D1	-1	25	D2	-1									-1
488	3	584	37	D0	680	45	D1	776	60	D2	872									-1
584	2	-1	30	D0	-1	35	D1	-1			-1			-1						-1
680	4	-1	40	D0	-1	41	D1	-1	42	D2	-1	43	D3	-1						
776	2	-1	51	D0	-1	52	D1	-1			-1			-1						-1
872	3	-1	70	D0	-1	77	D1	-1	83	D2	-1									-1



Estrutura do Índice

0	296																			
8	2	104	8	D0	200	15	D1	392				-1							-1	
104	4	-1	1	D0	-1	3	D1	-1	4	D2	-1	7	D3	-1						
200	4	-1	10	D0	-1	12	D1	-1	13	D2	-1	14	D3	-1						
296	1	8	29	D0	488			-1			-1			-1					-1	
392	3	-1	18	D0	-1	20	D1	-1	25	D2	-1								-1	
488	3	584	37	D0	680	45	D1	776	60	D2	872								-1	
584	2	-1	30	D0	-1	35	D1	-1			-1			-1					-1	
680	4	-1	40	D0	-1	41	D1	-1	42	D2	-1	43	D3	-1						
776	2	-1	51	D0	-1	52	D1	-1			-1			-1					-1	
872	3	-1	70	D0	-1	77	D1	-1	83	D2	-1								-1	

* 29 * * * *

* 8 * 15 * * *

* 37 * 45 * 60 * *

*1 * 3 * 4 * 7 *

*10*12*13*14*

* 18*20*25 * * *

* 30 * 35 * * *

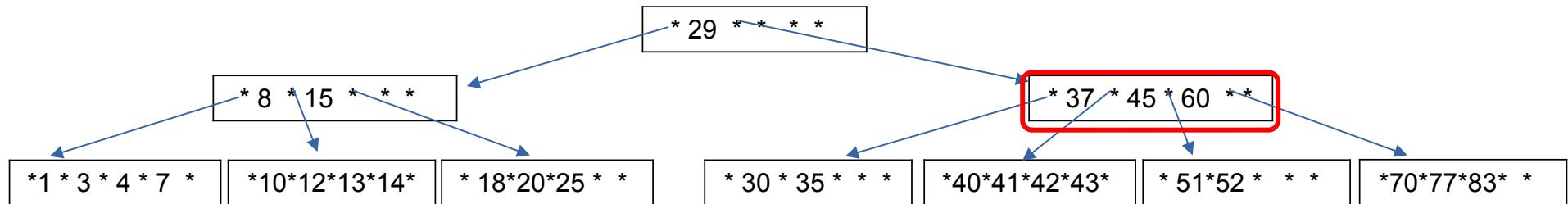
*40*41*42*43*

* 51*52 * * *

*70*77*83* *

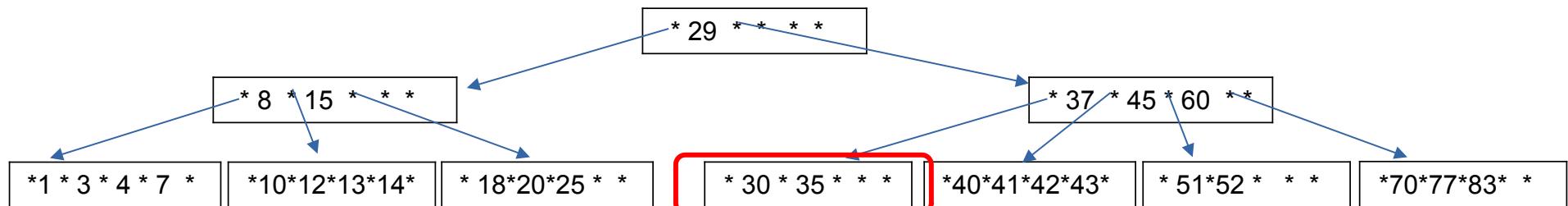
Estrutura do Índice

0	296																			
8	2	104	8	D0	200	15	D1	392				-1							-1	
104	4	-1	1	D0	-1	3	D1	-1	4	D2	-1	7	D3	-1						
200	4	-1	10	D0	-1	12	D1	-1	13	D2	-1	14	D3	-1						
296	1	8	29	D0	488			-1			-1			-1					-1	
392	3	-1	18	D0	-1	20	D1	-1	25	D2	-1								-1	
488	3	584	37	D0	680	45	D1	776	60	D2	872								-1	
584	2	-1	30	D0	-1	35	D1	-1				-1			-1				-1	
680	4	-1	40	D0	-1	41	D1	-1	42	D2	-1	43	D3	-1						
776	2	-1	51	D0	-1	52	D1	-1			-1			-1					-1	
872	3	-1	70	D0	-1	77	D1	-1	83	D2	-1								-1	



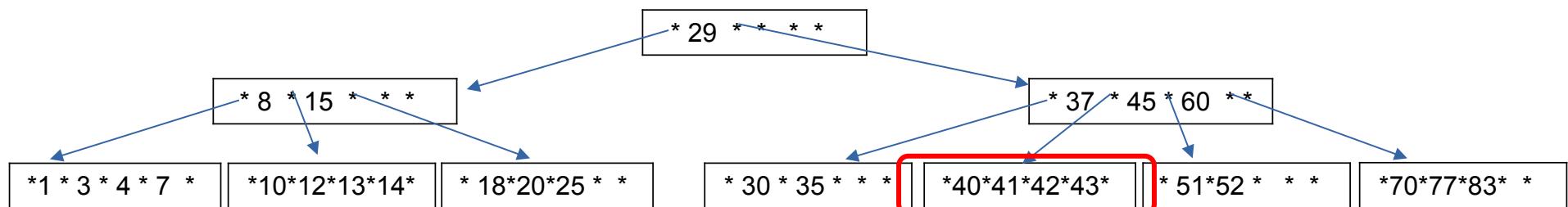
Estrutura do Índice

0	296																			
8	2	104	8	D0	200	15	D1	392				-1							-1	
104	4	-1	1	D0	-1	3	D1	-1	4	D2	-1	7	D3	-1						
200	4	-1	10	D0	-1	12	D1	-1	13	D2	-1	14	D3	-1						
296	1	8	29	D0	488			-1			-1			-1					-1	
392	3	-1	18	D0	-1	20	D1	-1	25	D2	-1								-1	
488	3	584	37	D0	680	45	D1	776	60	D2	872								-1	
584	2	-1	30	D0	-1	35	D1	-1			-1			-1					-1	
680	4	-1	40	D0	-1	41	D1	-1	42	D2	-1	43	D3	-1						
776	2	-1	51	D0	-1	52	D1	-1			-1			-1					-1	
872	3	-1	70	D0	-1	77	D1	-1	83	D2	-1								-1	



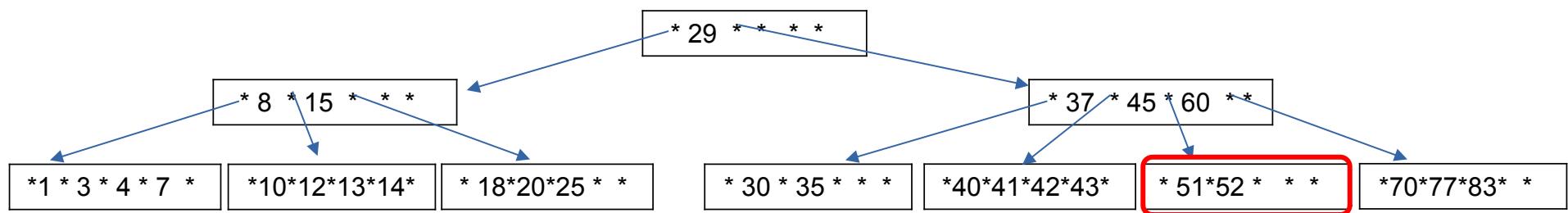
Estrutura do Índice

0	296													
8	2	104	8	D0	200	15	D1	392			-1		-1	
104	4	-1	1	D0	-1	3	D1	-1	4	D2	-1	7	D3	-1
200	4	-1	10	D0	-1	12	D1	-1	13	D2	-1	14	D3	-1
296	1	8	29	D0	488			-1			-1			-1
392	3	-1	18	D0	-1	20	D1	-1	25	D2	-1			-1
488	3	584	37	D0	680	45	D1	776	60	D2	872			-1
584	2	-1	30	D0	-1	35	D1	-1			-1			-1
680	4	-1	40	D0	-1	41	D1	-1	42	D2	-1	43	D3	-1
776	2	-1	51	D0	-1	52	D1	-1			-1			-1
872	3	-1	70	D0	-1	77	D1	-1	83	D2	-1			-1



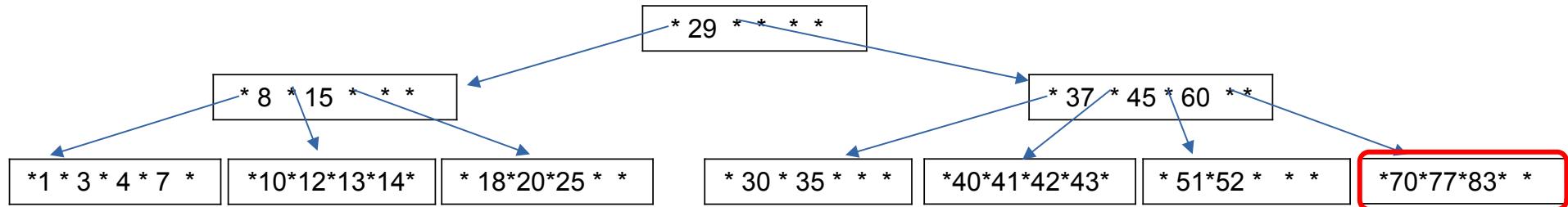
Estrutura do Índice

0	296																			
8	2	104	8	D0	200	15	D1	392				-1							-1	
104	4	-1	1	D0	-1	3	D1	-1	4	D2	-1	7	D3	-1						
200	4	-1	10	D0	-1	12	D1	-1	13	D2	-1	14	D3	-1						
296	1	8	29	D0	488			-1			-1			-1						-1
392	3	-1	18	D0	-1	20	D1	-1	25	D2	-1									-1
488	3	584	37	D0	680	45	D1	776	60	D2	872									-1
584	2	-1	30	D0	-1	35	D1	-1			-1			-1						-1
680	4	-1	40	D0	-1	41	D1	-1	42	D2	-1	43	D3	-1						
776	2	-1	51	D0	-1	52	D1	-1			-1			-1						-1
872	3	-1	70	D0	-1	//	D1	-1	83	D2	-1									-1



Estrutura do Índice

0	296													
8	2	104	8	D0	200	15	D1	392			-1			-1
104	4	-1	1	D0	-1	3	D1	-1	4	D2	-1	7	D3	-1
200	4	-1	10	D0	-1	12	D1	-1	13	D2	-1	14	D3	-1
296	1	8	29	D0	488			-1			-1			-1
392	3	-1	18	D0	-1	20	D1	-1	25	D2	-1			-1
488	3	584	37	D0	680	45	D1	776	60	D2	872			-1
584	2	-1	30	D0	-1	35	D1	-1			-1			-1
680	4	-1	40	D0	-1	41	D1	-1	42	D2	-1	43	D3	-1
776	2	-1	51	D0	-1	52	D1	-1			-1			-1
872	3	-1	70	D0	-1	77	D1	-1	83	D2	-1			-1



Algoritmos e Estruturas de Dados III

4.2 Árvores B

Prof. Hayala Curto
2022



PUC Minas

Árvore B+

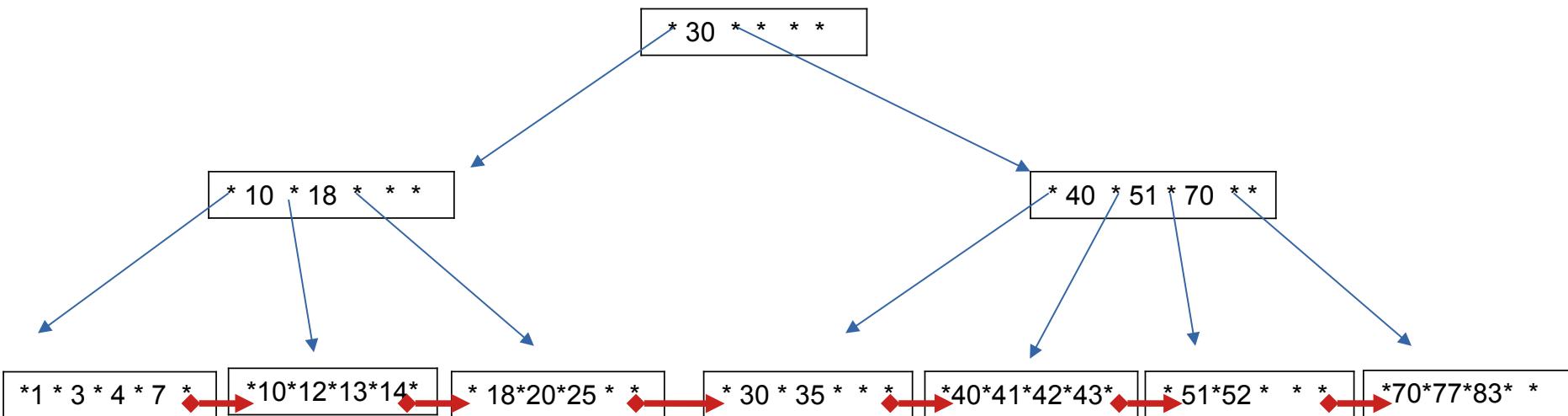


PUC Minas

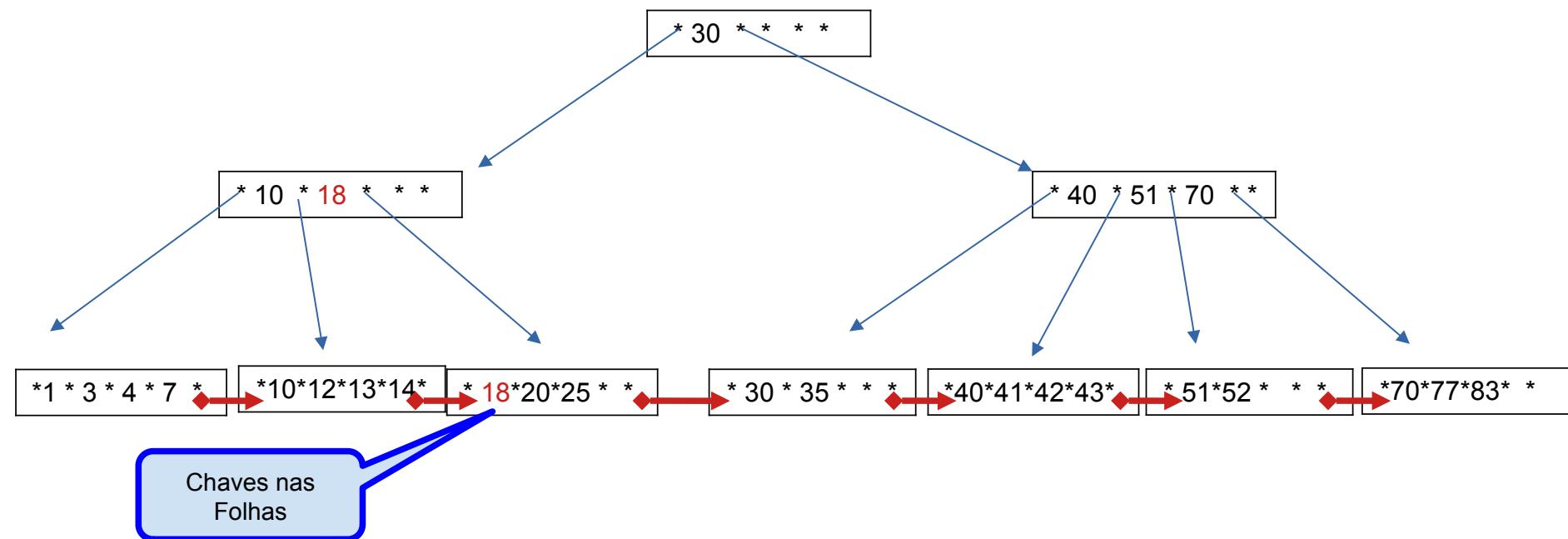
Árvore B+

- Todas as chaves são armazenadas nas folhas
- Cada folha aponta para a próxima folha (para permitir a leitura sequencial)
- As folhas podem possuir uma estrutura diferente das páginas não folhas, por serem as únicas páginas a carregarem dados
- Vantagens
 - Mantém a eficiência da busca e da inserção da árvore B
 - Aumenta a eficiência da localização do próximo registro na árvore de $O(\log n)$ para $O(1)$
 - Não é necessário manter nenhum ponteiro de registro em nós não-folha

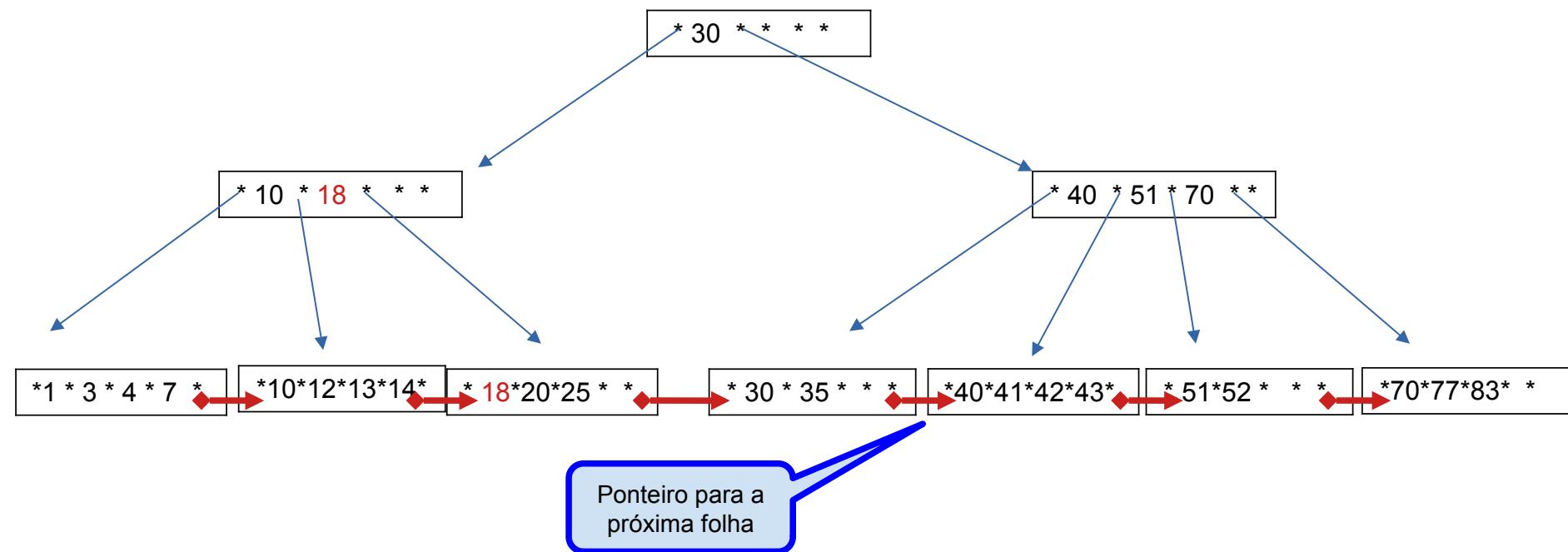
Árvore B+



Árvore B+



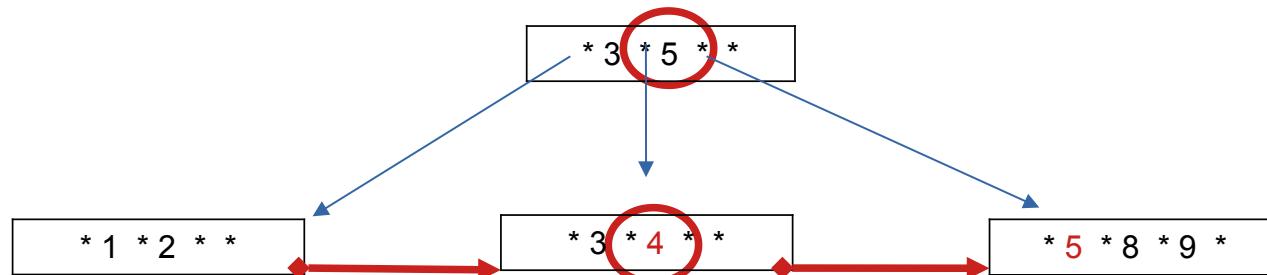
Árvore B+



Árvore B+ Operações

- As operações de busca, inserção e remoção são efetuadas de modo similar à Árvore B
- Uma busca por conjunto de chaves é simplificada à encontrar a primeira chave de interesse, seguida por uma operação de busca linear na lista ligada de folhas

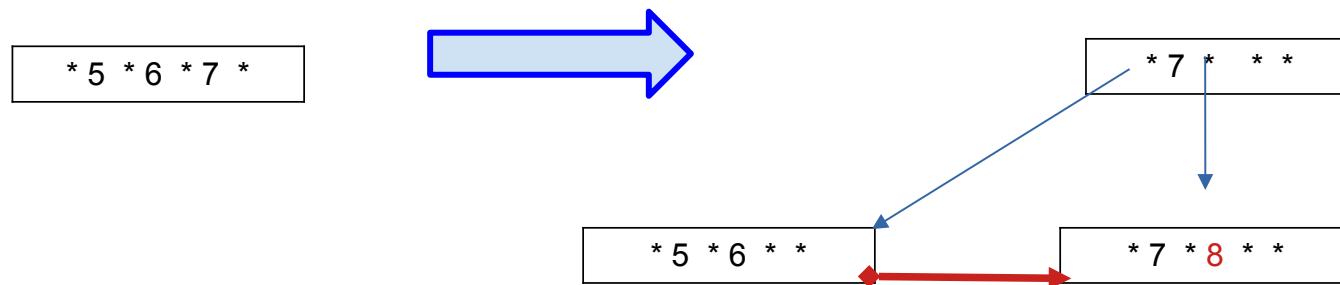
Exemplo: Retorne todos os registros entre [4, 5]



Árvore B+ Inserção

- Todas as chaves devem existir tanto em nó interno quanto em nó folha
- Ao ocorrer uma divisão de nó a chave mediana deve ser copiada para o novo nó pai e mantida no novo nó folha

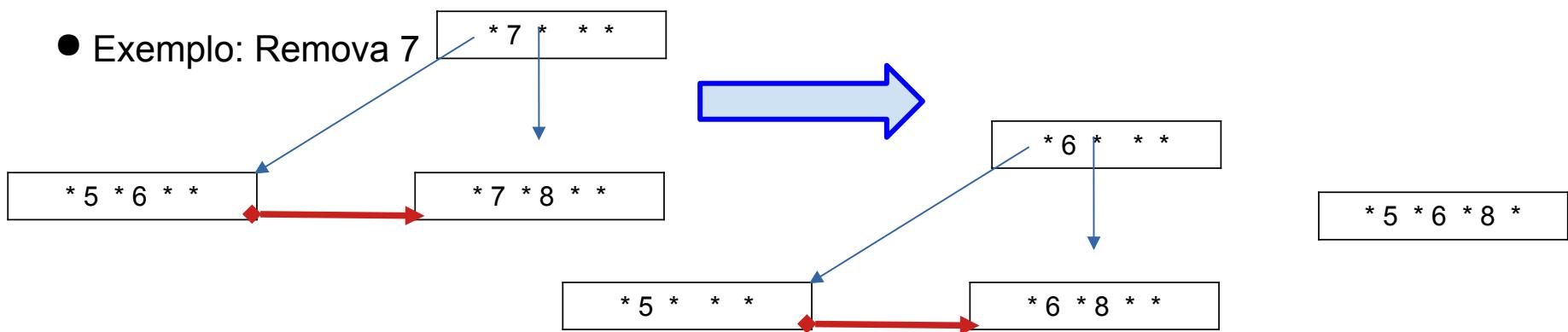
Exemplo: Insira 8



Árvore B+ Remoção

- Se chave a ser removida estiver em um nó folha: basta executar a remoção normalmente
- Se chave a ser removida estiver em um nó interno: ela deve ser substituída por seu antecessor ou sucessor (depende do algoritmo). Buscá-lo em seu filho esquerdo ou direito (depende do algoritmo), fazer a substituição e então chamar a função de remoção recursivamente para a subárvore à direita.

- Exemplo: Remova 7



0	96											
8	2	-1	10	E5	-1	15	E16	-1				
96	1	96	9	E0	8							
184	2	-1	1	E3	-1	2	E4	-1				



Árvore B \times Árvore B+

0	96											
8	2	-1	10	E5	-1	15	E16	-1				
96	1	184	9	8								
184	2	-1	1	E3	-1	2	E4	8				



Árvore B*



PUC Minas

Árvore B*

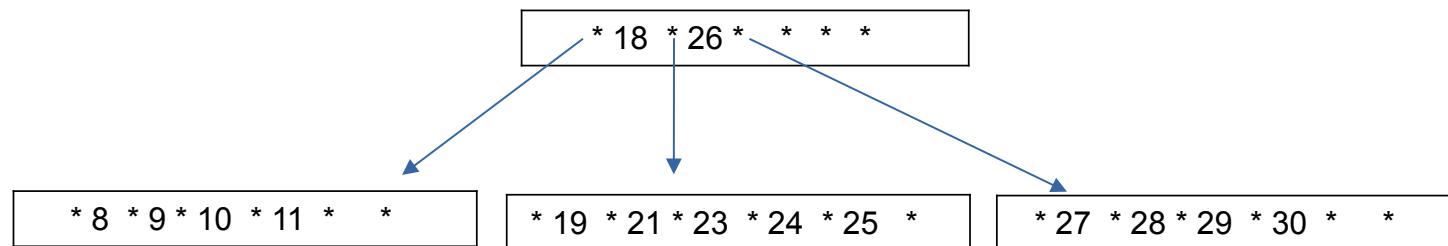
- Variação da árvore B proposta em 1973 por Knuth
- Técnica de redistribuição de chaves também é empregada durante as operações de inserção.
- Operação de split pode ser adiada até que duas páginas irmãs estejam completamente cheias

Para lembrar, na Operação de Inserção na Árvore B:

- Se o elemento couber na página, basta incluí-lo de forma ordenada
- Se não couber, a página deve ser dividida em duas (split) e o elemento do meio deve ser promovido

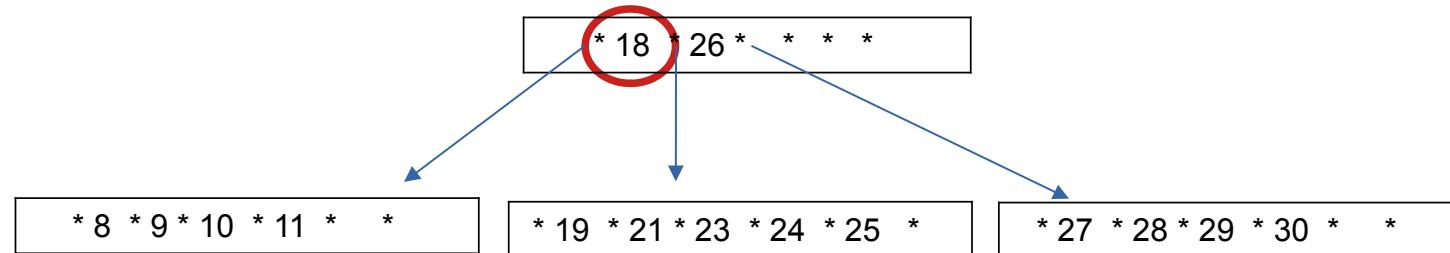
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B*:
Redistribuição de Chaves entre irmãos



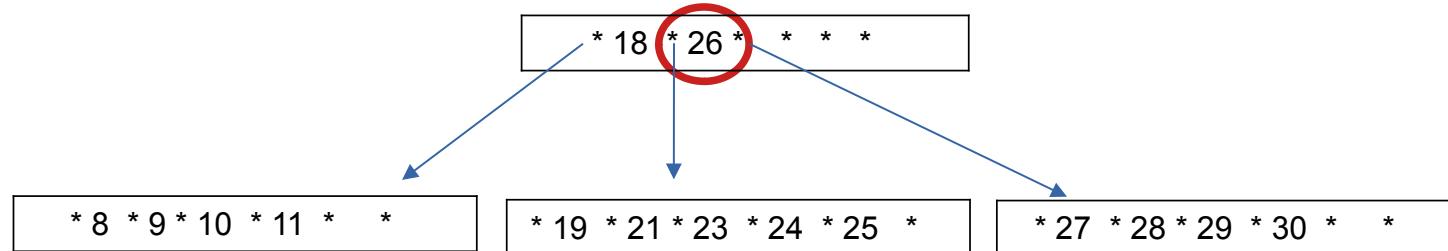
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B*:
Redistribuição de Chaves entre irmãos



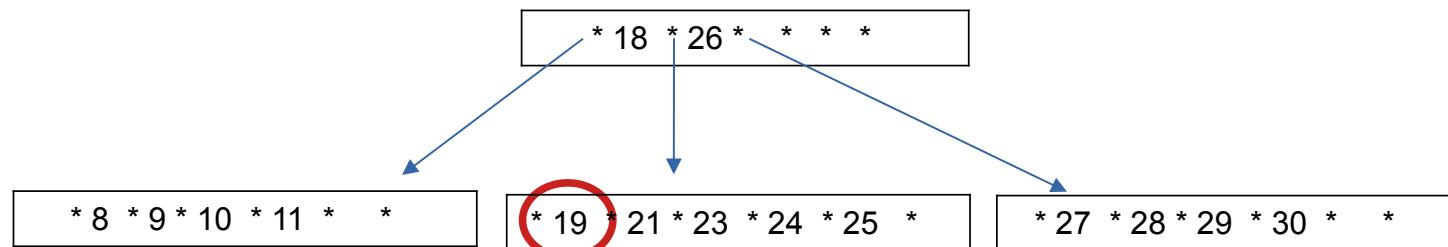
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B*:
Redistribuição de Chaves entre irmãos



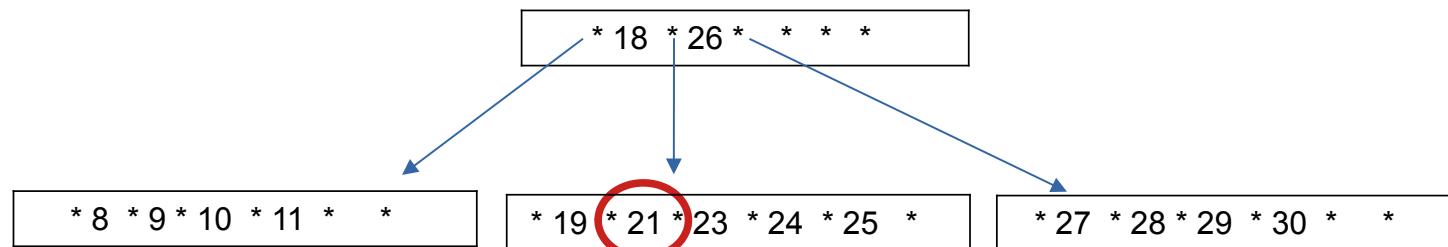
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B*:
Redistribuição de Chaves entre irmãos



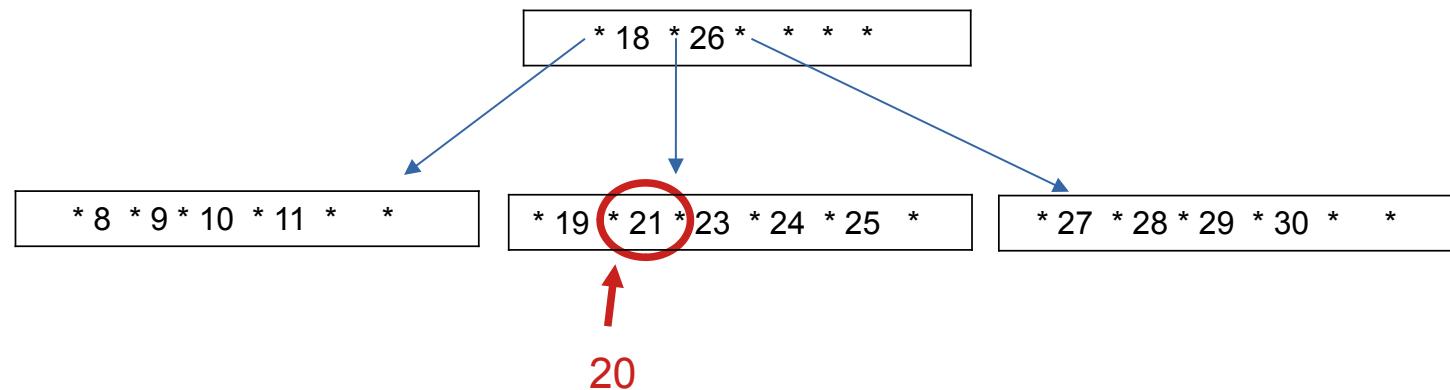
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B*:
Redistribuição de Chaves entre irmãos



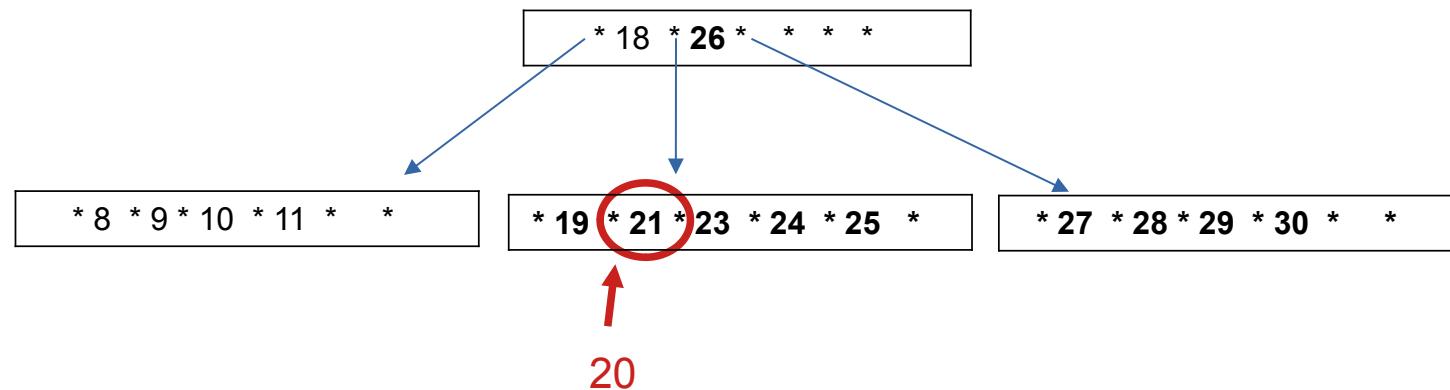
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B*:
Redistribuição de Chaves entre irmãos



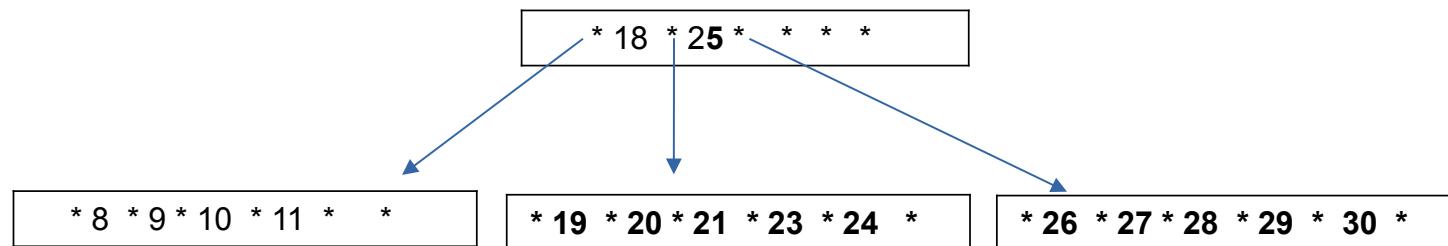
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B*:
Redistribuição de Chaves entre irmãos



Árvore B* – Inclusão com Split

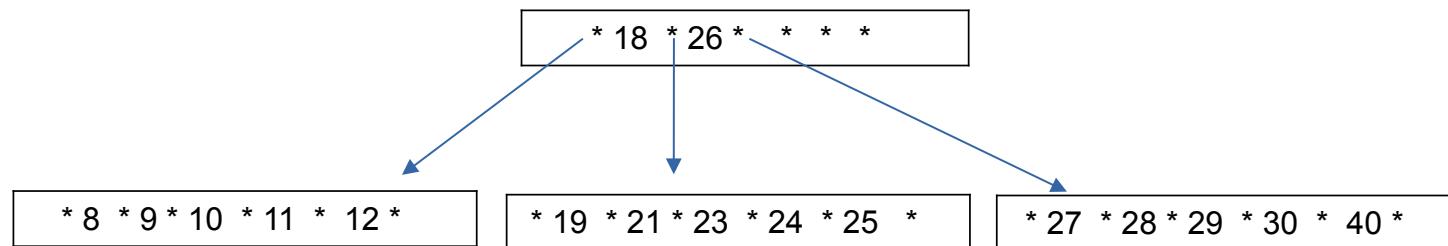
Inclusão da chave 20 em uma Árvore B*:
Redistribuição de Chaves entre irmãos





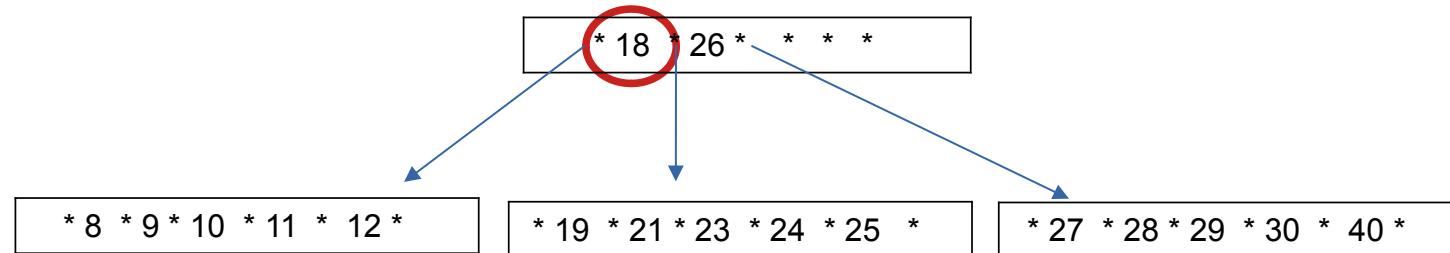
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B:
Divisão de Um para Dois (one-to-two split)



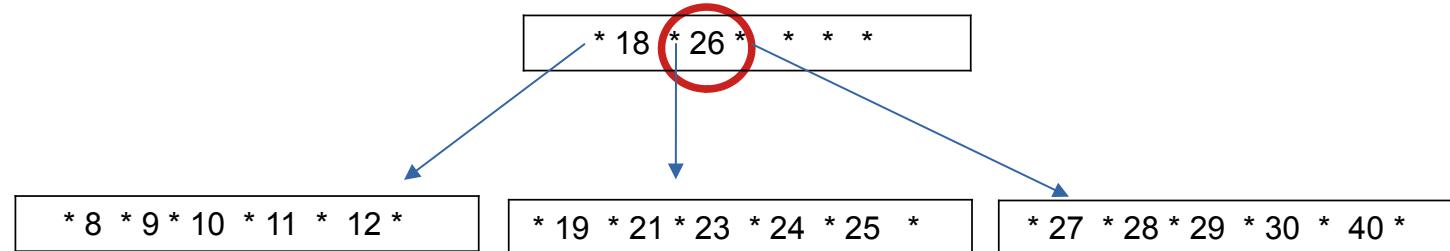
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B:
Divisão de Um pra Dois (one-to-two split)



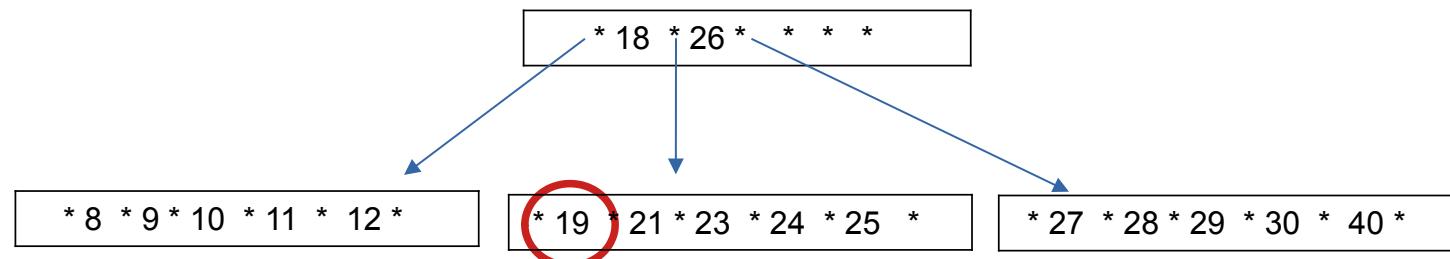
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B:
Divisão de Um pra Dois (one-to-two split)



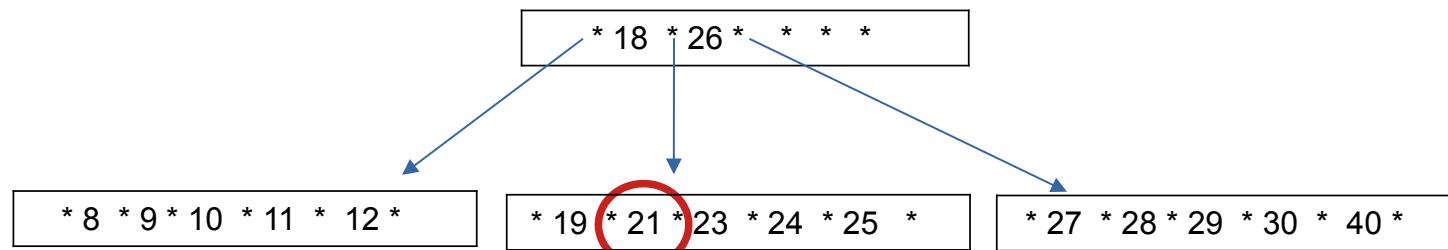
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B:
Divisão de Um pra Dois (one-to-two split)



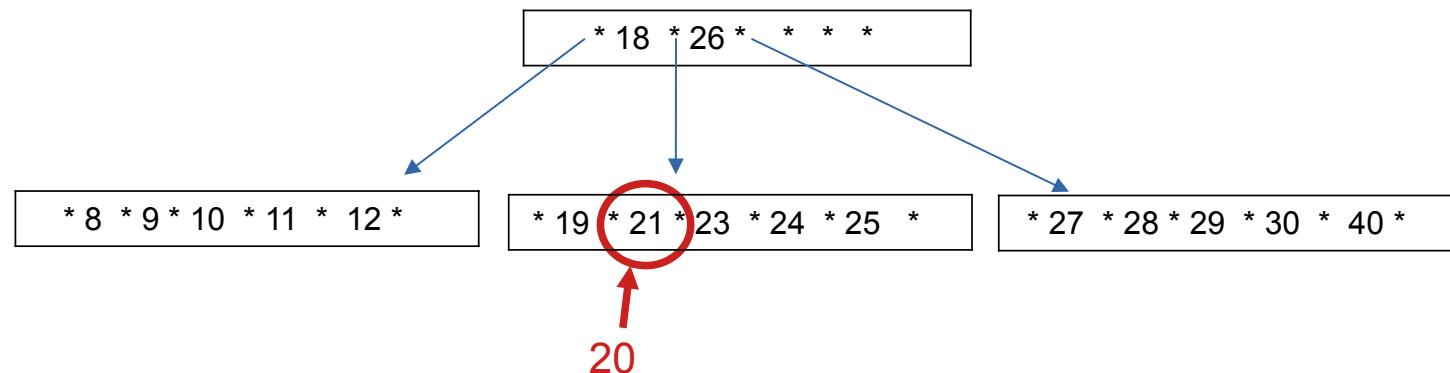
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B:
Divisão de Um pra Dois (one-to-two split)



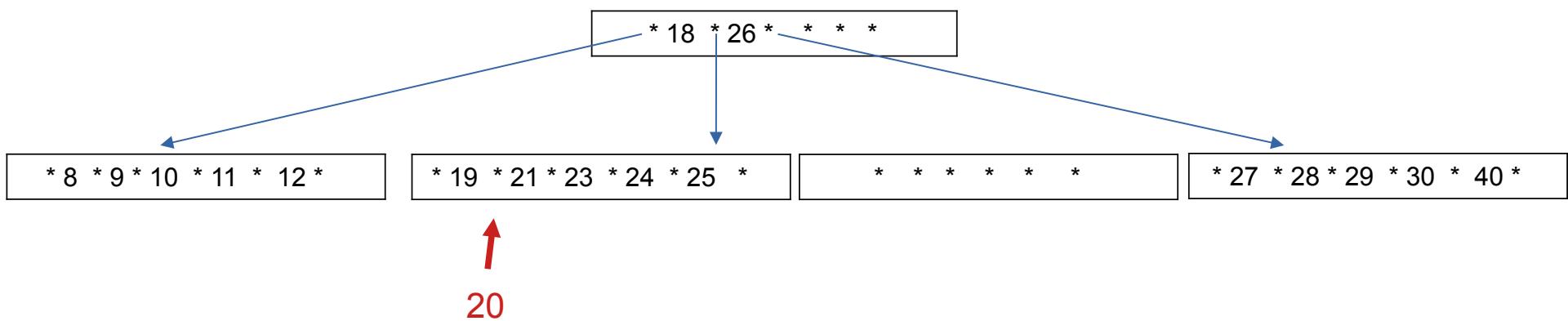
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B:
Divisão de Um pra Dois (one-to-two split)



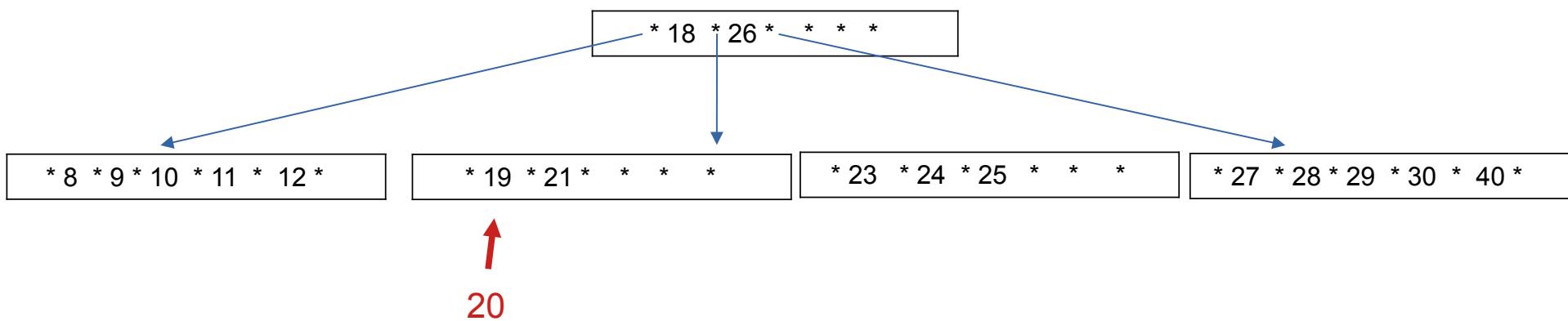
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B:
Divisão de Um pra Dois (one-to-two split)



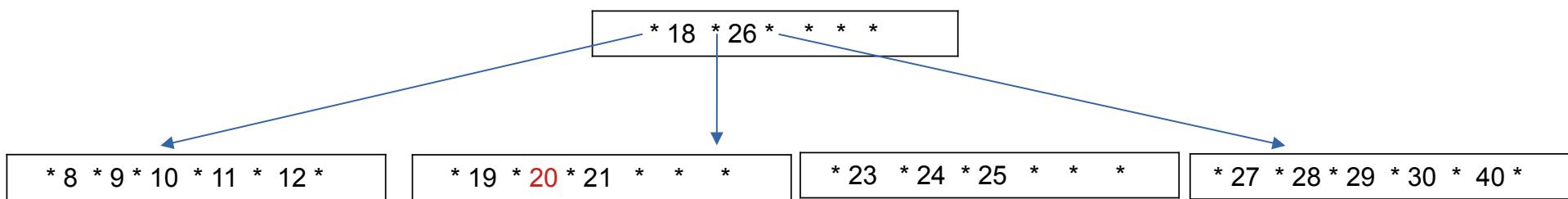
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B:
Divisão de Um pra Dois (one-to-two split)



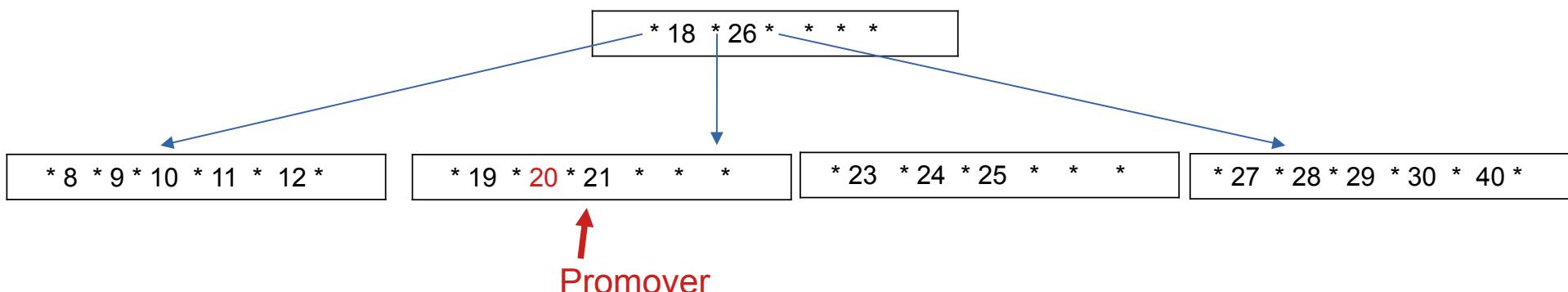
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B:
Divisão de Um pra Dois (one-to-two split)



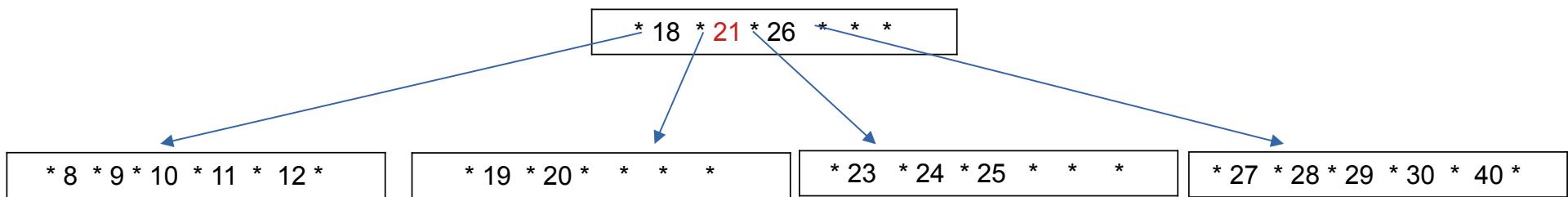
Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B:
Divisão de Um pra Dois (one-to-two split)



Árvore B* – Inclusão com Split

Inclusão da chave 20 em uma Árvore B:
Divisão de Um pra Dois (one-to-two split)



Árvore B* - Propriedades

- Em uma Árvore B* de ordem m
- Cada página tem no máximo m descendentes
- Toda página, exceto a raiz e as folhas, tem no mínimo $(2m-1)/3$ descendentes
- A raiz tem pelo menos 2 descendentes, a menos que seja uma folha
- Todas as folhas estão no mesmo nível
- Uma página não-folha com k descendentes contém $k-1$ chaves
- Uma página folha contém no mínimo piso($(2m-1)/3$) e no máximo $m-1$ chaves

Árvore B* - Propriedades

- As principais alterações estão na segunda e na última regra
- Esta propriedade afeta as regras para remoção e redistribuição
- Deve-se tomar cuidado na implementação, uma vez que a raiz nunca tem irmã, e portanto requer tratamento especial
- Uma solução é dividir a raiz usando a divisão convencional (one-to-two split), outra é permitir que a raiz seja maior

Algoritmos e Estruturas de Dados III

6 Hashing

Prof. Hayala Curto
2022



PUC Minas

Hashing

Tabelas de Dispersão

As tabelas de dispersão (hash) em disco também podem ser usadas como índices, ao invés das árvores.

- Nessas tabelas, o custo de acesso é **O(1)** → **Principal Vantagem**
- Custo para isso: sobrecarga, maior consumo de disco ou processamento extra
- A posição do registro é determinada por uma função de dispersão (função hash).

Hashing – Função de Dispersão

Função de Dispersão

$h(\text{chave}) \rightarrow \text{endereço}$

$$H(3204) = 504 \quad \text{endereço no arquivo de índice}$$

↑ ↑
Chave Endereço

- Depende do número de endereços e da natureza da chave.
- Chave pode ser um campo, uma combinação de campos ou um pedaço de campo
- Registros do índice devem ser de tamanho fixo.
- Quantidade fixa de endereços (depende do tratamento de colisões)
- Quanto mais **disperso**, melhor
- 504 é uma posição relativa. Devemos multiplicar pelo tamanho do registro

Hashing – Função de Dispersão

Índice

	Código	Endereço
0	5	432
1		
2	1	0
3		
4		
5	4	324
6	3	216
7		
8	2	108
9		

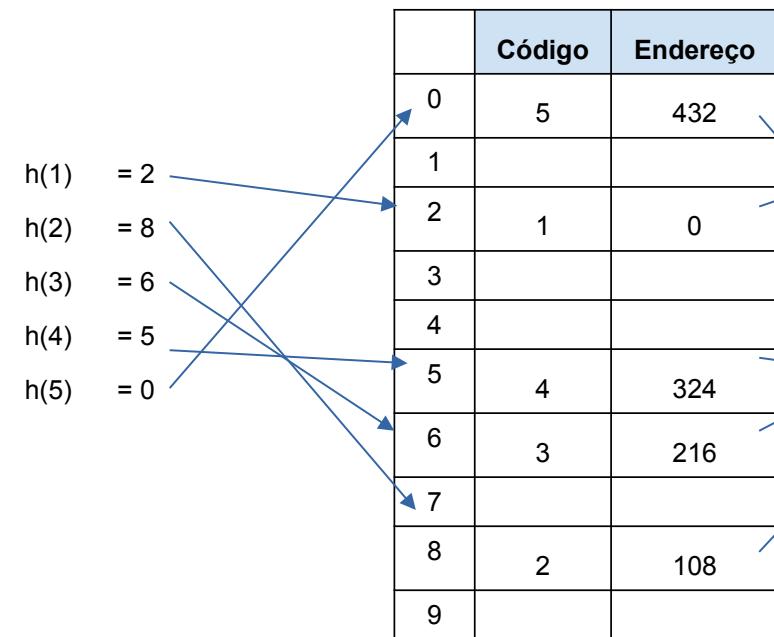
$h(1) = 2$

$h(2) = 8$

$h(3) = 6$

$h(4) = 5$

$h(5) = 0$



Arquivo de Dados

Código	Título	Autor	Preço
1	Java Web Services	Martin Kalin	34,87
2	Web Design	Mauríci o Samy	45,5
3	Web Services em PHP	Lorena Jane	33,9
4	Programaç ão Java	Décio	93,22
5	Desenvolvi mento WEB	Fulano	118,9

Como escolher uma boa função de dispersão?



PUC Minas

Hashing – Função de Dispersão

- **Como escolher uma boa função de dispersão?**

- Operação matemática
- Conversão de base
- Campos não numéricos
- Método da dobra
- Método da divisão
- Outros...

Hashing – Função de Dispersão

- Como escolher uma boa função de dispersão?

- **Operação matemática**
- Conversão de base
- Campos não numéricos
- Método da dobra
- Método da divisão
- Outros...

Operação Matemática

Exemplos de função de dispersão – Operação matemática

Elevar a chave ao **quadrado** e pegar um grupo de dígitos do meio:

- $A = h(453) \rightarrow 453^2 = 205209 \rightarrow A = 52$
- Dois dígitos foram escolhidos pois o arquivo possui apenas 100 endereços
- Resultado fica então dentro da faixa dos endereços disponíveis para o índice

Hashing – Função de Dispersão

- Como escolher uma boa função de dispersão?
 - Operação matemática
 - **Conversão de base**
 - Campos não numéricos
 - Método da dobra
 - Método da divisão
 - Outros...

Conversão de Base

Exemplos de função de dispersão – Conversão de Base

Mudar a chave para outra base:

$$A = h(453) \rightarrow 453_{10} = 382_{11} \rightarrow 382 \bmod 99 = 85 \rightarrow A = 85$$

(99 é a quantidade de endereços no arquivo)

Hashing – Função de Dispersão

- Como escolher uma boa função de dispersão?
 - Operação matemática
 - Conversão de base
 - **Campos não numéricos**
 - Método da dobra
 - Método da divisão
 - Outros...

Campos não numéricos

Exemplos de função de dispersão – Campos não numéricos

Multiplicar o valor ASCII das letras e usar o resto da divisão pelo número de endereços.

Chave	Cálculo	Endereço
JOÃO	$74 \times 79 = 5846$	846
CARLOS	$67 \times 65 = 4355$	355
GILBERTO	$71 \times 73 = 5183$	183

- Três dígitos foram escolhidos pois o arquivo possui apenas 1000 endereços
- Resultado fica então dentro da faixa dos endereços disponíveis para o índice

Hashing – Função de Dispersão

- Como escolher uma boa função de dispersão?
 - Operação matemática
 - Conversão de base
 - Campos não numéricos
 - **Método da dobra**
 - Método da divisão
 - Outros...

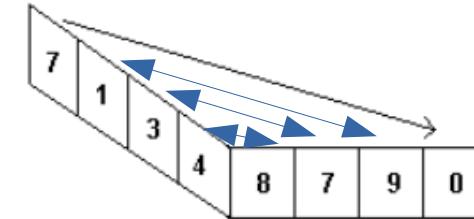
Método da Dobra

A chave é interpretada como uma sequência de dígitos escritos em um “pedaço papel”.

O método consiste em “dobrar esse papel”, de maneira que os dígitos se sobreponham.

O processo é repetido até que os dígitos formem um número menor que o tamanho da tabela hash.

7	1	3	4	8	7	9	0
---	---	---	---	---	---	---	---



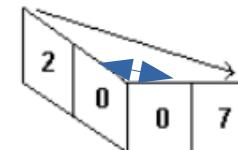
$$7 + 0 = 7$$

$$1 + 9 = \cancel{1}0$$

$$3 + 7 = \cancel{1}0$$

$$4 + 8 = \cancel{1}2$$

2	0	0	7
---	---	---	---



$$2 + 7 = 9$$

$$0 + 0 = 0$$

0	9
---	---

Método da Dobra - Exemplo

Supondo que cada chave ocupa 10 bits e a dimensão da tabela é 32 (2^5).

Deve-se transformar as chaves em endereços que ocupam 5 bits.

Exemplo:

$71 = 00010\ 00111 \Rightarrow ?$

$46 = 00001\ 01110 \Rightarrow ?$

Método da Dobra - Exemplo

Supondo que cada chave ocupa 10 bits e a dimensão da tabela é 32 (2^5).

Deve-se transformar as chaves em endereços que ocupam 5 bits.

Exemplo:

$$71 = 00010\ 00111 \Rightarrow 01111 \Rightarrow 15$$

$$46 = 00001\ 01110 \Rightarrow 11110 \Rightarrow 30$$

Hashing – Função de Dispersão

- Como escolher uma boa função de dispersão?
 - Operação matemática
 - Conversão de base
 - Campos não numéricos
 - Método da dobra
 - **Método da divisão**
 - Outros...

Método da Divisão

$$h(x) = x \bmod m$$

- Fácil e eficiente
- Alguns valores de m são melhores que outros

m sendo um **número primo** ajuda a “espalhar” os valores

Estudos apontam bons valores de m :

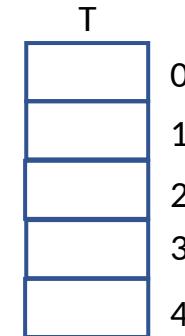
- Escolher m de modo que seja um **número primo não próximo a uma potência de 2**; ou
- Escolher m tal que **não possua divisores primos menores do que 20**

Método da Divisão

Transformar a chave x em um endereço-base $h(x)$, que é uma valor entre 0 e m .

Exemplo:

$$h(x) = x \bmod 5$$



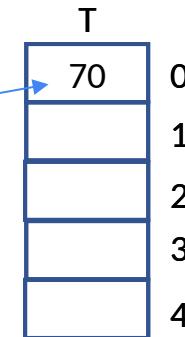
Método da Divisão

Transformar a chave x em um endereço-base $h(x)$, que é uma valor entre 0 e m .

Exemplo:

$$h(x) = x \bmod 5$$

Chaves: 70



Método da Divisão

Transformar a chave x em um endereço-base $h(x)$, que é uma valor entre 0 e m .

Exemplo:

$$h(x) = x \bmod 5$$

Chaves: 70, 51

T	
70	0
51	1
	2
	3
	4

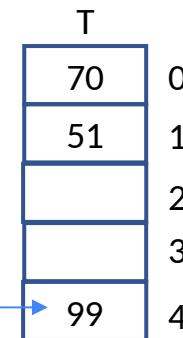
Método da Divisão

Transformar a chave x em um endereço-base $h(x)$, que é uma valor entre 0 e m .

Exemplo:

$$h(x) = x \bmod 5$$

Chaves: 70, 51, 99



Método da Divisão

Transformar a chave x em um endereço-base $h(x)$, que é uma valor entre 0 e m .

Exemplo:

$$h(x) = x \bmod 5$$

Chaves: 70, 51, 99, 17

T	
70	0
51	1
17	2
	3
99	4

Método da Divisão

Transformar a chave x em um endereço-base $h(x)$, que é uma valor entre 0 e m .

Exemplo:

$$h(x) = x \bmod 5$$

Chaves: 70, 51, 99, 17, 15

T	0
15, 70	0
51	1
17	2
	3
99	4

colisão

Método da Divisão

Transformar a chave x em um endereço-base $h(x)$, que é uma valor entre 0 e m .

Exemplo:

$$h(x) = x \bmod 5$$

Chaves: 70, 51, 99, 17, 15

**COMO
RESOLVER?**

colisão

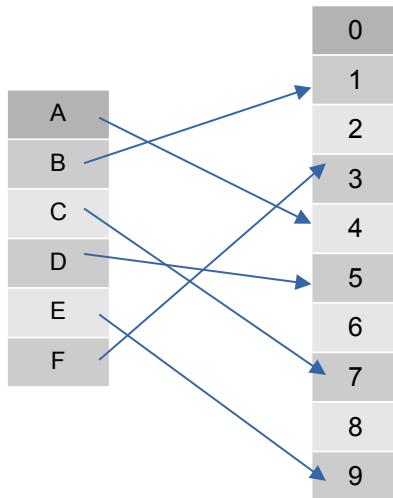
T	15, 70	0
	51	1
	17	2
		3
	99	4

Colisões

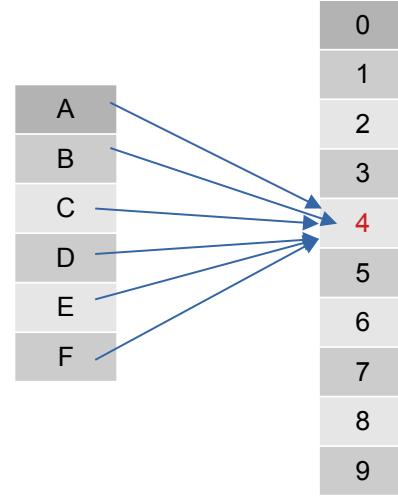


PUC Minas

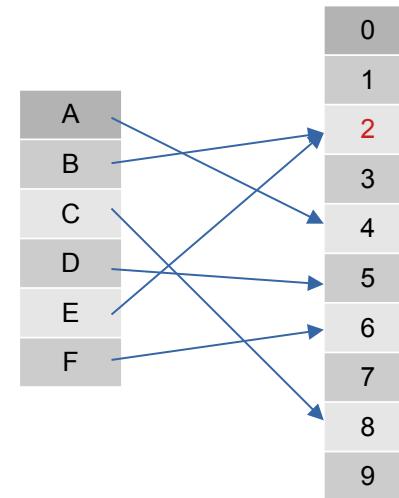
Hashing – Colisões



Desejável

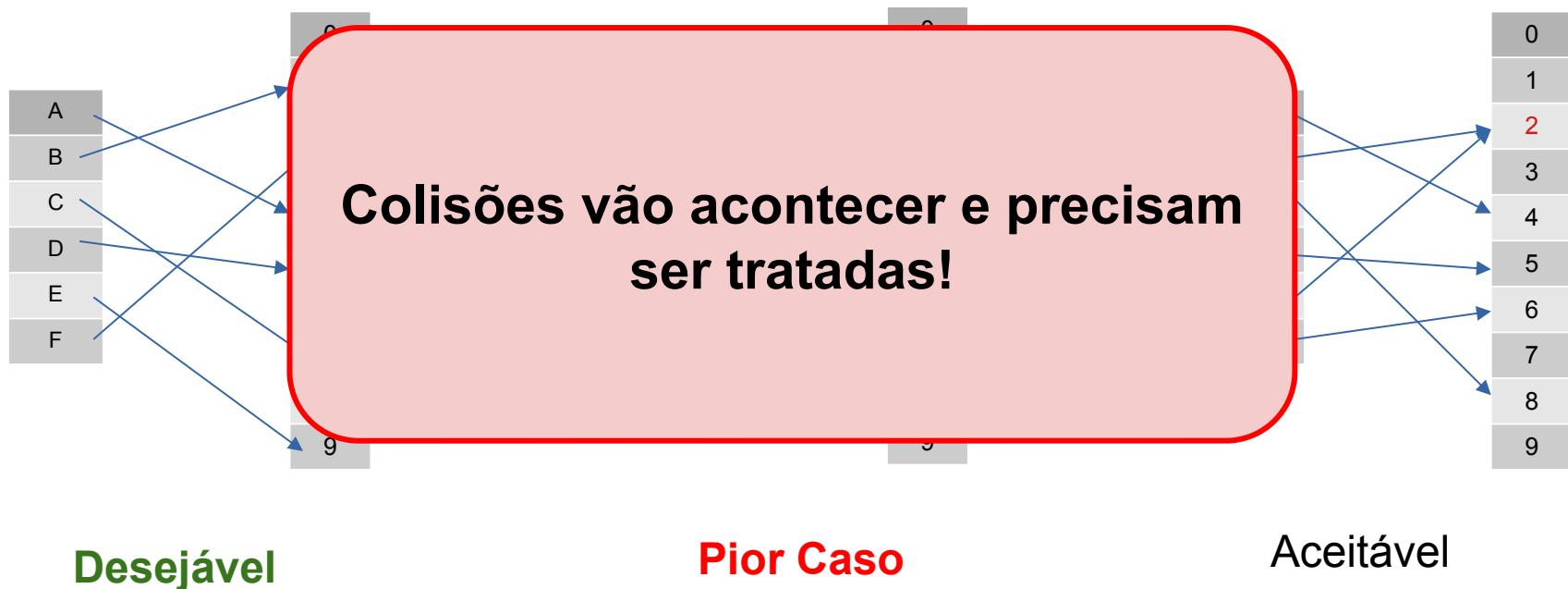


Pior Caso



Aceitável

Hashing – Colisões



Desejável

Pior Caso

Aceitável

Colisões - Tratamento de Colisões

Alternativas para o tratamento de colisões:

- **Endereçamento aberto** – usa outras posições vazias dentro da própria da tabela hash
- **Encadeamento interno** – usa uma área extra dentro da própria da tabela hash
- **Encadeamento externo** – usa uma área externa, fora da tabela (p.ex., um segundo arquivo).

Colisões - Tratamento de Colisões

Alternativas para o tratamento de colisões:

- **Endereçamento aberto** – usa outras posições vazias dentro da própria da tabela hash
- **Encadeamento interno** – usa uma área extra dentro da própria da tabela hash
- **Encadeamento externo** – usa uma área externa, fora da tabela (p.ex., um segundo arquivo).

Endereçamento aberto

Uma nova posição **dentro** da área da tabela será procurada

Fórmula matemática para encontrar nova posição

- Sondagem linear
- Sondagem quadrática
- Duplo hash (double hashing)

Endereçamento aberto

Uma nova posição **dentro** da área da tabela será procurada

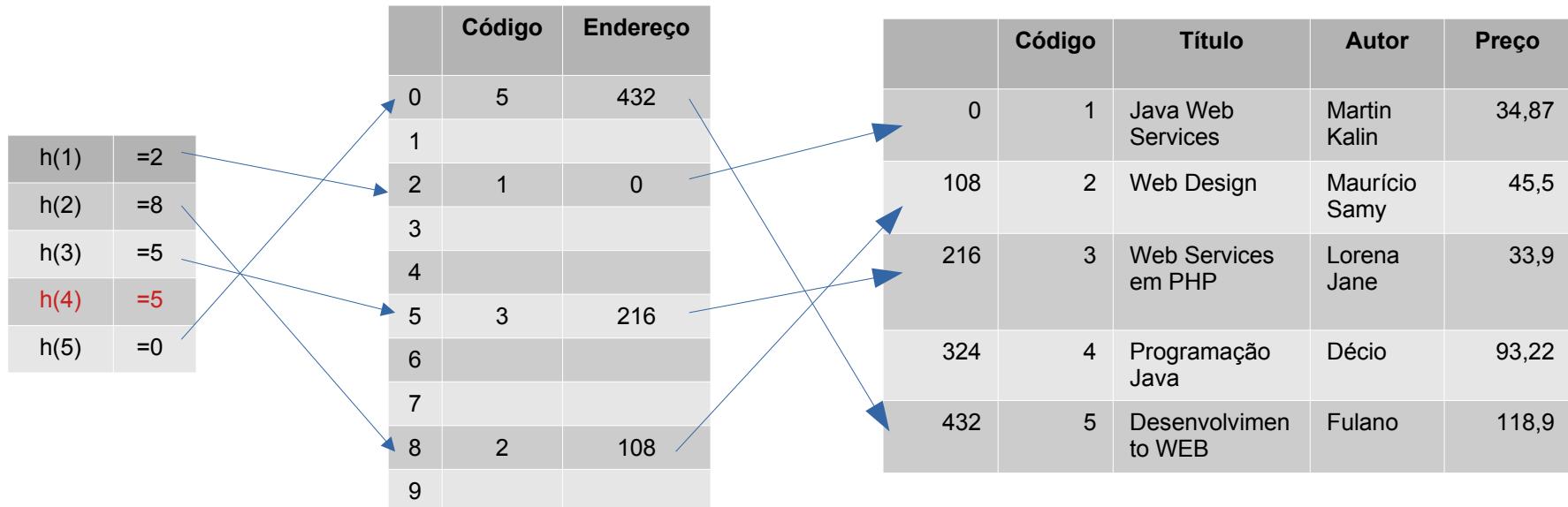
Fórmula matemática para encontrar nova posição

- **Sondagem linear**
- Sondagem quadrática
- Duplo hash (double hashing)

Endereçamento aberto

Sondagem linear – as próximas posições são sondadas (circularmente), até que uma posição livre seja encontrada. Regra:

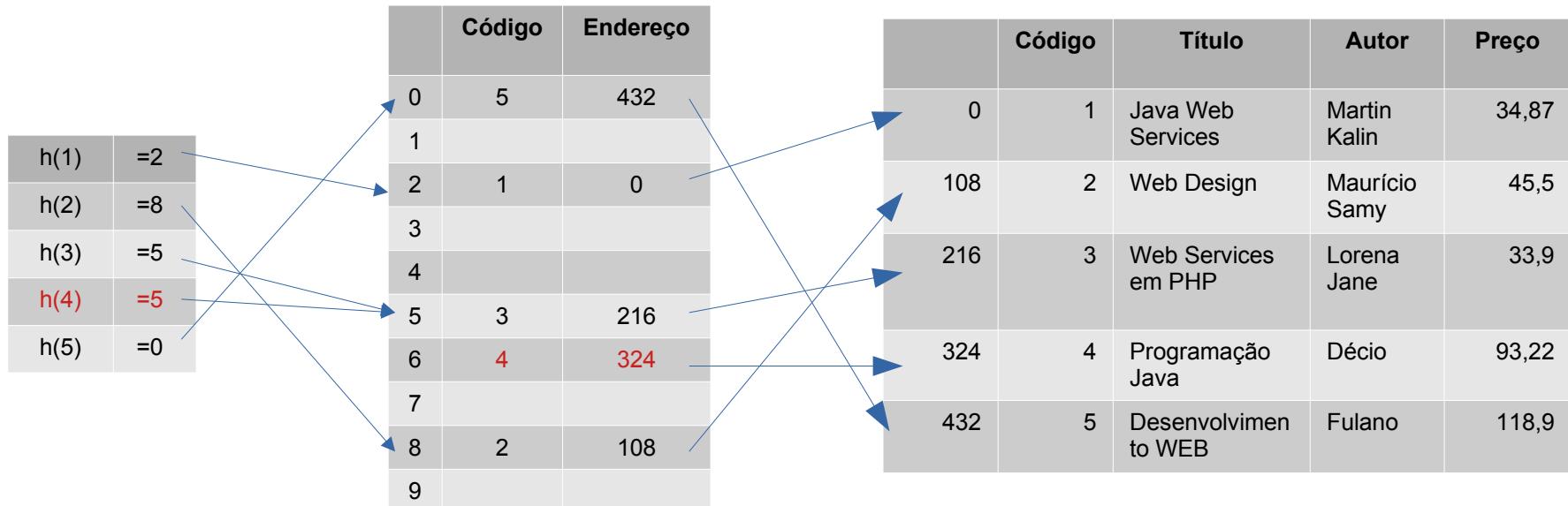
$$h(k, i) = [h(k) + i] \bmod n$$



Endereçamento aberto

Sondagem linear – as próximas posições são sondadas (circularmente), até que uma posição livre seja encontrada. Regra:

$$h(k, i) = [h(k) + i] \bmod n$$



Endereçamento aberto

Sondagem linear – as próximas posições são sondadas (circularmente), até que uma posição livre seja encontrada. Regra: $h(k, i) = [h(k) + i] \bmod n$

- Avanço unitário → +i
- Avanço circular → mod n, em que n é o número de posições da tabela
- Até encontrar uma posição **vazia**
- Retorno ao mesmo índice indica que tabela hash está cheia
- Busca deve ler registros de forma sequencial até encontrar uma posição **vazia**
- Uso de indicador de **lápide** pode ser necessário caso exclusões ocorram, para atender a estratégia de busca acima (sequencial até encontrar posição vazia sem marcação de lápide)
- Se ocorrerem muitas colisões, pode ser criado um agrupamento (clustering) de chaves em uma certa área. _x0001_ Isso pode fazer com que sejam necessários muitos acessos para recuperar um certo registro.

Endereçamento aberto

Uma nova posição **dentro** da área da tabela será procurada

Fórmula matemática para encontrar nova posição

- Sondagem linear
- **Sondagem quadrática**
- Duplo hash (double hashing)

Endereçamento aberto

Sondagem quadrática – a distância até a próxima posição a ser sondada é determinada pelo quadrado da tentativa, Regra:

$$h(k, i) = [h(k) + i^2] \bmod n$$

$h(1)$	=2
$h(2)$	=8
$h(3)$	=5
$h(4)$	=2
$h(5)$	=2

	Código	Endereço
0		
1		
2	1	0
3		
4		
5	3	216
6		
7		
8	2	108
9		

	Código	Título	Autor	Preço
0	1	Java Web Services	Martin Kalin	34,87
108	2	Web Design	Maurício Samy	45,5
216	3	Web Services em PHP	Lorena Jane	33,9
324	4	Programação Java	Décio	93,22
432	5	Desenvolvimento WEB	Fulano	118,9

Endereçamento aberto

Sondagem quadrática – a distância até a próxima posição a ser sondada é determinada pelo quadrado da tentativa, Regra:

$$h(k, i) = [h(k) + i^2] \bmod n$$

$h(1)$	=2
$h(2)$	=8
$h(3)$	=5
$h(4)$	=2
$h(5)$	=2

	Código	Endereço
0		
1		
2	1	0
3	4	324
4		
5	3	216
6		
7		
8	2	108
9		

	Código	Título	Autor	Preço
0	1	Java Web Services	Martin Kalin	34,87
108	2	Web Design	Maurício Samy	45,5
216	3	Web Services em PHP	Lorena Jane	33,9
324	4	Programação Java	Décio	93,22
432	5	Desenvolvimento WEB	Fulano	118,9

Endereçamento aberto

Sondagem quadrática – a distância até a próxima posição a ser sondada é determinada pelo quadrado da tentativa, Regra:

$$h(k, i) = [h(k) + i^2] \bmod n$$

$h(1)$	=2
$h(2)$	=8
$h(3)$	=5
$h(4)$	=2
$h(5)$	=2

	Código	Endereço
0		
1		
2	1	0
3	4	324
4		
5	3	216
6	5	432
7		
8	2	108
9		

	Código	Título	Autor	Preço
0	1	Java Web Services	Martin Kalin	34,87
108	2	Web Design	Maurício Samy	45,5
216	3	Web Services em PHP	Lorena Jane	33,9
324	4	Programação Java	Décio	93,22
432	5	Desenvolvimento WEB	Fulano	118,9

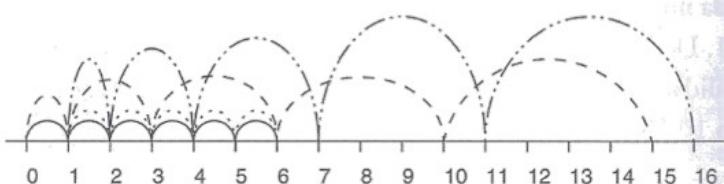
Endereçamento aberto

Sondagem quadrática – a distância até a próxima posição a ser sondada é determinada pelo quadrado da tentativa, Regra: $h(k, i) = [h(k) + i^2] \bmod n$

- Avanço quadrático → $+ i^2$
- Avanço circular → $\bmod n$, em que n é o número de posições da tabela
- Até encontrar uma posição **vazia**

tentativa linear:
endereço-base 0
endereço-base 1
.....

tentativa quadrática:
endereço-base 0
endereço-base 1
.....



Endereçamento aberto

Uma nova posição **dentro** da área da tabela será procurada

Fórmula matemática para encontrar nova posição

- Sondagem linear
- Sondagem quadrática
- **Duplo hash (double hashing)**

Endereçamento aberto

Duplo hash – a distância até a próxima posição a ser sondada é determinada por uma segunda função hash. Regra:

$$h(k, i) = [h(k) + i * h'(k)] \bmod n$$

$h(1)$	=2
$h(2)$	=8
$h(3)$	=5
$h(4)$	=2
$h'(4)$	=4
$h(5)$	=2
$h'(5)$	=1

	Código	Endereço
0		
1		
2	1	0
3		
4		
5	3	216
6		
7		
8	2	108
9		

	Código	Título	Autor	Preço
0	1	Java Web Services	Martin Kalin	34,87
108	2	Web Design	Maurício Samy	45,5
216	3	Web Services em PHP	Lorena Jane	33,9
324	4	Programação Java	Décio	93,22
432	5	Desenvolvimento WEB	Fulano	118,9

Endereçamento aberto

Duplo hash – a distância até a próxima posição a ser sondada é determinada por uma segunda função hash. Regra:

$$h(k, i) = [h(k) + i * h'(k)] \bmod n$$

$h(1)$	=2
$h(2)$	=8
$h(3)$	=5
$h(4)$	=2
$h'(4)$	=4
$h(5)$	=2
$h'(5)$	=1

	Código	Endereço
0		
1		
2	1	0
3		
4		
5	3	216
6	4	324
7		
8	2	108
9		

	Código	Título	Autor	Preço
0	1	Java Web Services	Martin Kalin	34,87
108	2	Web Design	Maurício Samy	45,5
216	3	Web Services em PHP	Lorena Jane	33,9
324	4	Programação Java	Décio	93,22
432	5	Desenvolvimento WEB	Fulano	118,9

Endereçamento aberto

Duplo hash – a distância até a próxima posição a ser sondada é determinada por uma segunda função hash. Regra:

$$h(k, i) = [h(k) + i * h'(k)] \bmod n$$

$h(1)$	=2
$h(2)$	=8
$h(3)$	=5
$h(4)$	=2
$h'(4)$	=4
$h(5)$	=2
$h'(5)$	=1

	Código	Endereço
0		
1		
2	1	0
3	5	432
4		
5	3	216
6	4	324
7		
8	2	108
9		

	Código	Título	Autor	Preço
0	1	Java Web Services	Martin Kalin	34,87
108	2	Web Design	Maurício Samy	45,5
216	3	Web Services em PHP	Lorena Jane	33,9
324	4	Programação Java	Décio	93,22
432	5	Desenvolvimento WEB	Fulano	118,9

Endereçamento aberto

Duplo hash – a distância até a próxima posição a ser sondada é determinada por uma segunda função hash. Regra: $h(k, i) = [h(k) + i * h'(k)] \bmod n$

- Avanço com segunda função hash → $+i * h'(k)$
- Avanço circular → mod n, em que n é o número de posições da tabela
- Avanço da segunda função hash a partir da posição atual dada pela primeira função hash.
- Vantagem: tende a espalhar melhor as chaves pelos endereços.
- Desvantagem: os endereços podem estar muito distantes um do outro (o princípio da localidade é violado), provocando seekings adicionais.

Colisões - Tratamento de Colisões

Alternativas para o tratamento de colisões:

- **Endereçamento aberto** – usa outras posições vazias dentro da própria da tabela hash
- **Encadeamento interno** – usa uma área extra dentro da própria da tabela hash
- **Encadeamento externo** – usa uma área externa, fora da tabela (p.ex., um segundo arquivo).

Encadeamento



PUC Minas

Hashing – Encadeamento

Encadeamento Interno

Os registros colididos são armazenados
Em uma área extra (geralmente
separada da área principal)

$h(1)$	=2
$h(2)$	=8
$h(3)$	=5
$h(4)$	=5
$h(5)$	=2

	Código	Endereço	Próximo
0			-1
1			-1
2	1	0	-1
3			-1
4			-1
5	3	216	-1
6			-1
7			-1
8	2	108	-1
9			-1
10			-1
11			-1

Hashing – Encadeamento

Encadeamento Interno

Os registros colididos são armazenados
Em uma área extra (geralmente
separada da área principal)

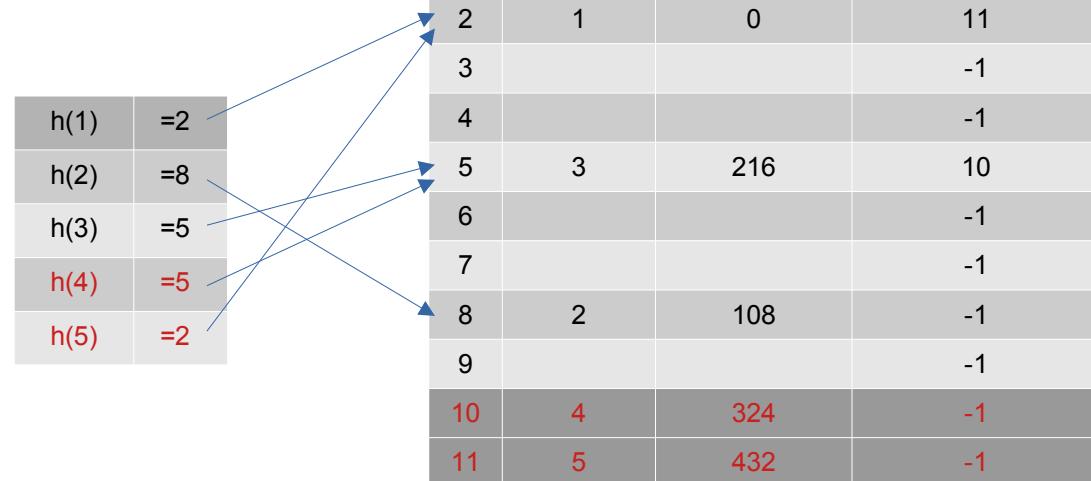
$h(1)$	=2
$h(2)$	=8
$h(3)$	=5
$h(4)$	=5
$h(5)$	=2

	Código	Endereço	Próximo
0			-1
1			-1
2	1	0	11
3			-1
4			-1
5	3	216	10
6			-1
7			-1
8	2	108	-1
9			-1
10	4	324	-1
11			-1

Hashing – Encadeamento

Encadeamento Interno

Os registros colididos são armazenados
Em uma área extra (geralmente
separada da área principal)



Hashing – Encadeamento

Encadeamento

Análise: Com uma “boa” função hash, assume-se que qualquer item do conjunto de chaves tem igual **probabilidade** de ser endereçado para qualquer entrada da tabela.

Logo, o comprimento esperado de cada lista encadeada é n/m , chamado de **fator de carga**, em que n representa o número de registros na tabela e m o tamanho da tabela.

As operações pesquisa, inserção e remoção custam $O(1 + n/m)$ em média, sendo que a constante 1 representa o tempo para encontrar a entrada da tabela (ou seja, calcular a função hash), e n/m o tempo para percorrer a lista.

Hashing – Encadeamento

Encadeamento Externo

Lista encadeada – todos os registros são armazenados em uma lista encadeada (outro arquivo)

$h(1)$	=2
$h(2)$	=8
$h(3)$	=5
$h(4)$	=5
$h(5)$	=2

	Cabeça da lista
0	-1
1	-1
2	0
3	-1
4	-1
5	2
6	-1
7	-1
8	1
9	-1

	Código	Endereço	Próximo
0	1	0	4
1	2	108	-1
2	3	216	3
3	4	324	-1
4	5	432	-1

Hashing – Encadeamento

Encadeamento Externo

Lista encadeada – todos os registros são armazenados em uma **lista encadeada** (outro arquivo)

$h(1)$	=2
$h(2)$	=8
$h(3)$	=5
$h(4)$	=5
$h(5)$	=2

	Cabeça da lista
0	-1
1	-1
2	0
3	-1
4	-1
5	2
6	-1
7	-1
8	1
9	-1

Índice

	Código	Endereço	Próximo
0	1	0	4
1	2	108	-1
2	3	216	3
3	4	324	-1
4	5	432	-1

Hashing – Encadeamento

Encadeamento Externo

Preenchimento da lista encadeada

$h(1)$	=2
$h(2)$	=8
$h(3)$	=5
$h(4)$	=5
$h(5)$	=2

	Cabeça da lista
0	-1
1	-1
2	0
3	-1
4	-1
5	2
6	-1
7	-1
8	1
9	-1

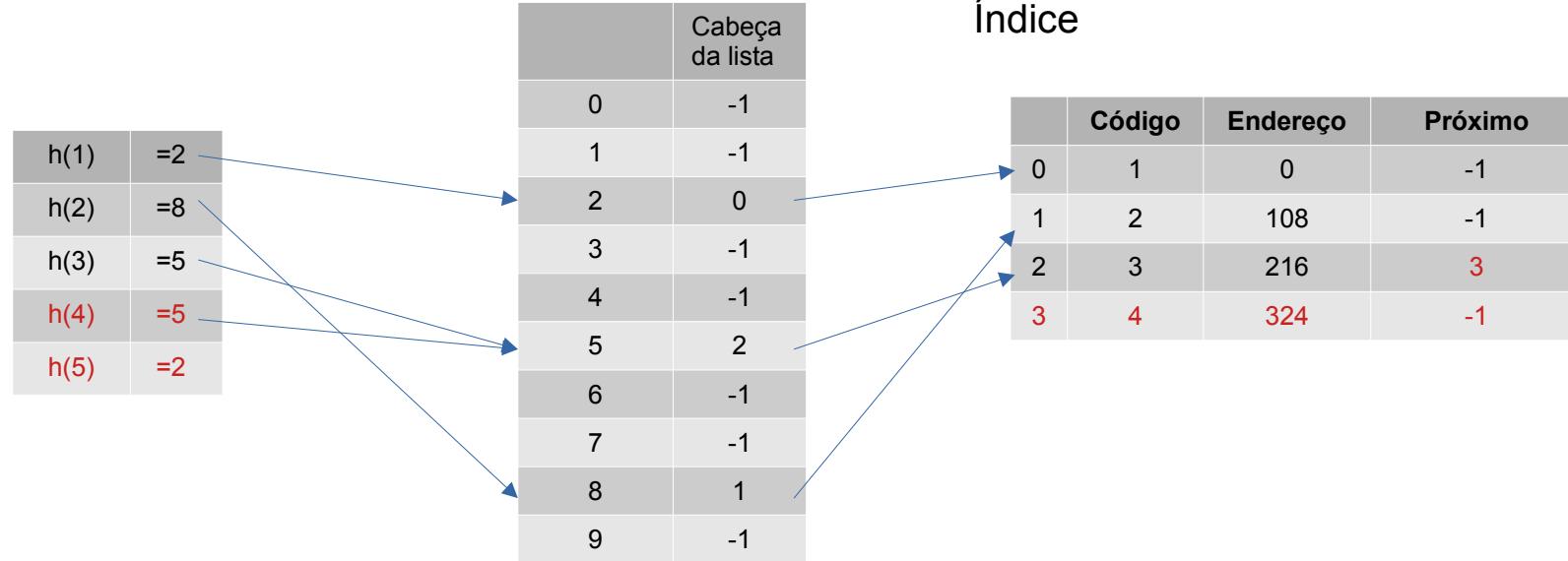
Índice

	Código	Endereço	Próximo
0	1	0	-1
1	2	108	-1
2	3	216	-1

Hashing – Encadeamento

Encadeamento Externo

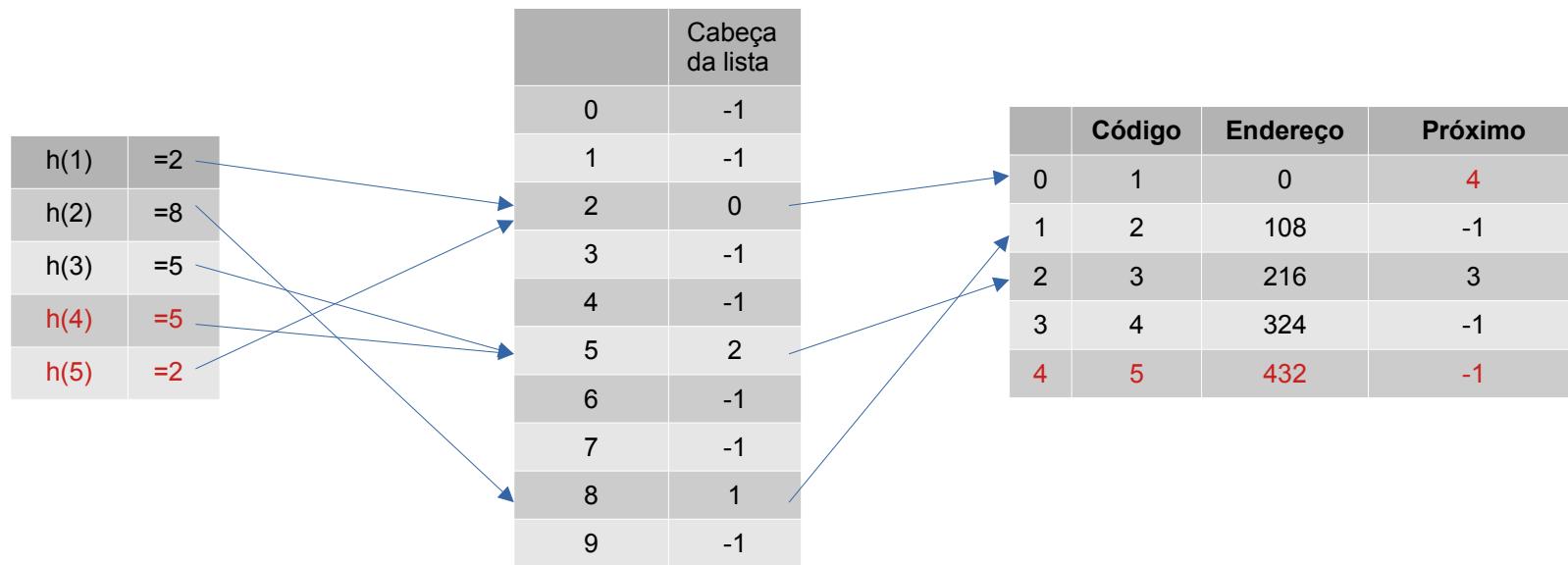
Preenchimento da lista encadeada



Hashing – Encadeamento

Encadeamento Externo

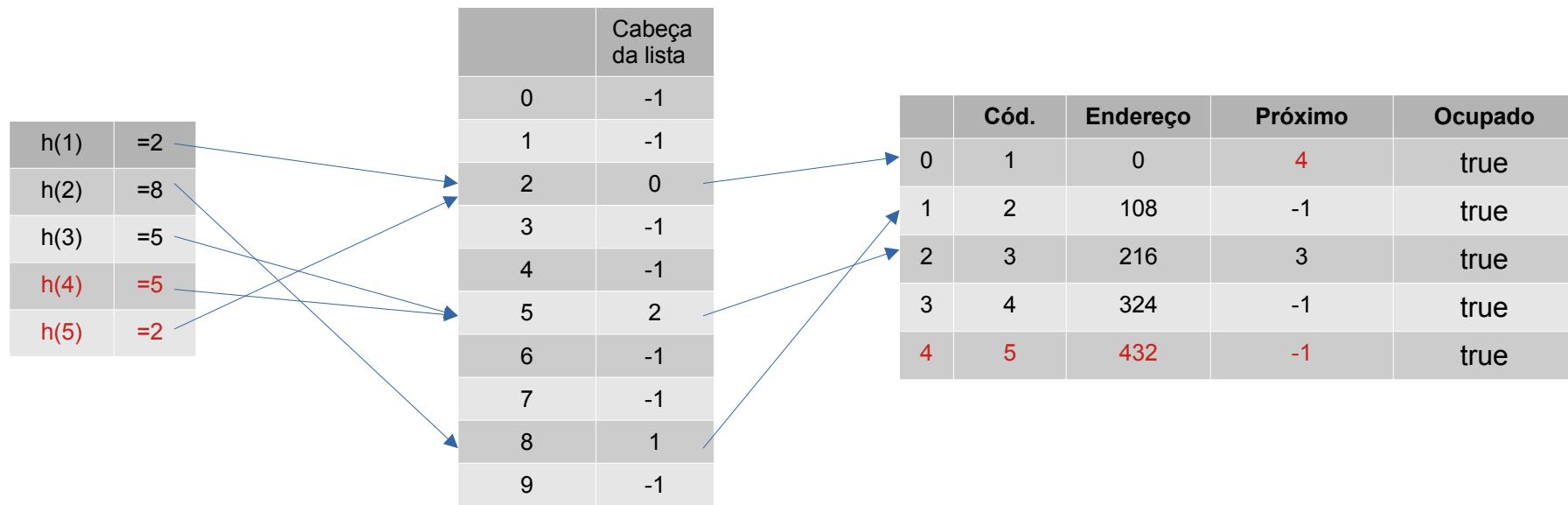
Preenchimento da lista encadeada



Hashing – Encadeamento

Encadeamento Externo

Preenchimento da lista encadeada



Hashing – Encadeamento

Para facilitar a manutenção da lista encadeada, pode-se adicionar um **flag** indicador de **status** a cada registro

No exemplo do slide anterior, esse flag é chamado **ocupado**

O flag ocupado pode ter os seguintes valores:

TRUE: quando o compartimento tem um registro

FALSE: quando o registro que estava no compartimento foi excluído

Hashing – Encadeamento

Encadeamento Externo

Lista encadeada – todos os registros são armazenados em uma **lista encadeada**

- Lista encadeada pode ser outro arquivo
- Lista encadeada pode estar no mesmo arquivo da tabela hash
- Parte fixa, constante, reservada para a tabela
- Parte crescente com a lista encadeada
- Cresce de acordo com a necessidade
- Aparentemente a melhor solução, mas ainda precisamos melhorar e evitar a leitura de apenas 1 registro a cada busca em disco → Buckets

Buckets



PUC Minas

Hashing – Buckets

Balde → Cesto

- Da mesma forma que no caso da árvore B, é importante otimizar o acesso ao disco. Árvore Binária X Árvore B
- Assim, cada posição no índice, pode conter mais de uma entrada (ou registro). Hash sem Buckets X Hash com Buckets

Exemplo:

- Registro no índice = 12 bytes
- Setor do HD = 4096 bytes = 341,33 registros

Hashing – Buckets

Buckets com 4 pares de chave/endereço

$h(1)$	=2
$h(2)$	=8
$h(3)$	=5
$h(4)$	=5
$h(5)$	=2

	Código	Endereço	Código	Endereço	Código	Endereço
0						
1						
2	1	0	5	432		
3						
4						
5	3	216	3	324		
6						
7						
8	2	108				
9						

Hashing – Buckets

Tratamento de colisões (quando o cesto está cheio)

- Alocar o registro no próximo cesto em que houver espaço disponível (usando endereçamento aberto)
- Usar uma das técnicas anteriores (considerando que cada posição equivale a um novo cesto)

Hashing – Buckets encadeados

Buckets com 4 pares de chave/endereço

		Código	Endereço	Código	Endereço	Código	Endereço	Próx
	0							
	1							
$h(1)$	=2							
$h(2)$	=8							
$h(3)$	=5							
$h(4)$	=5							
$h(5)$	=2							
	2	1	0	5	432			
	3							
	4							
	5	3	216	4	324			
	6							
	7							
	8	2	108					
	9							

The diagram illustrates the mapping of five hash function outputs (h(1) to h(5)) to a 10-element array of buckets. The array has 4 slots per element. Blue arrows point from each row to its corresponding index in the array.

Hashing – Buckets

Funções de Dispersão e Buckets

- A **pior função** de dispersão mapeia todos os valores das chaves de busca para o mesmo bucket; isto faz o tempo de acesso proporcional ao número de valores das chaves de busca no arquivo.
- Uma função de dispersão ideal é **uniforme** isto é, a cada bucket é atribuído o mesmo número de valores das chaves de busca do conjunto de todos os possíveis valores.
- Uma função de dispersão ideal é **aleatória**, assim cada bucket terá quase o mesmo número de valores atribuídos a ele, independente da distribuição real dos valores da chave de busca no arquivo

Hashing – Estático X Dinâmico

- Tabelas Hash podem ter uma capacidade fixa ou crescer de acordo com os dados armazenados nas mesmas
- Hash Estático: A capacidade da tabela é fixa e determinada em sua criação. Se os dados crescem além da capacidade, novo hash é necessário
- Hash Dinâmico: A capacidade da tabela é pequena para acomodar poucos dados no início, mas aumenta (ou diminui) a medida que novos dados são inseridos (ou removidos)

Algoritmos e Estruturas de Dados III

Aula 6.3 – Multilistas e Índices Invertidos

Prof. Hayala Curto
2022



PUC Minas

Índices Secundários

- Estruturas de dados tipo arquivo não são usadas estritamente para disponibilizar consultas a partir de chaves primárias, que identificam unicamente os registros
- Consultas cujas respostas envolvem mais de um registro são formuladas a partir de possíveis chaves secundárias
- Chaves Secundárias são chaves cujos valores identificam mais de um registro, diferente das chaves primárias, a partir das quais obtém-se um único registro

Índices Secundários

- Vários problemas apresentam chaves com valores idênticos (elas não distinguem unicamente os registros)
- As buscas resultam em subconjuntos de registros
- EX:

Posição	Nome	Idade	Cidade
01	João	19	BH
02	Ana	18	RJ
03	Pedro	19	SP
04	Maria	20	BH
05	Carlos	18	SP

Índices Secundários

Um arquivo em que são feitas buscas a partir de chaves secundárias deve ser organizado para minimizar o esforço de pesquisa. Possibilidades:

Arquivos/Índices multilista: mantém-se um link para cada chave secundária C, através dos quais, registros que apresentam mesmo valor de C são encadeados.

Arquivos/Índices invertidos: mantém-se arquivos adicionais que indexam os registros que apresentam mesmo valor de chave secundária

Multilistas



PUC Minas

Multilistas

- Arquivos representem links para cada chave secundária, usando ponteiros para encadear os registros que apresentam o mesmo valor de chave secundária.
- Para cada chave secundária é elaborado um índice correspondente e todos os índices são mantidos numa área denominada diretório

Multilistas

Arquivos Multilistas são compostos por:

- diretório contendo 1 ou mais índices
- área de registro de dados

Multilistas

Posição	Nome	Idade	Link I	Cidade
01	João	19	03	BH
02	Ana	18	05	RJ
03	Pedro	19	-1	SP
04	Maria	20	-1	BH
05	Carlos	18	-1	SP

Idade	Quantidade	Pos. Inicial
18	02	02
19	02	01
20	01	04

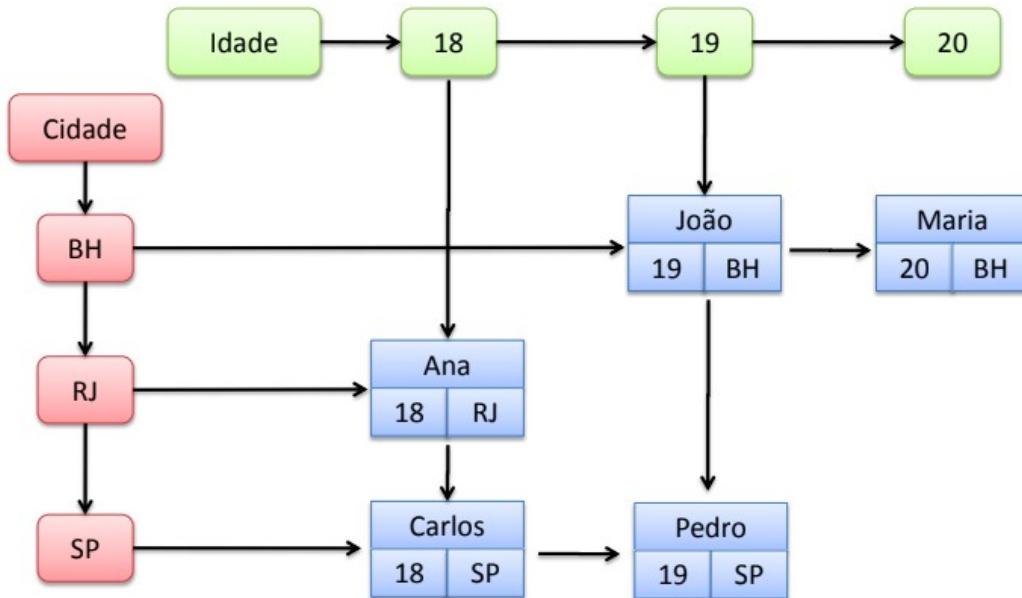
Multilistas

Posição	Nome	Idade	Link I.	Cidade	• Link C.
01	João	19	03	BH	04
02	Ana	18	05	RJ	-1
03	Pedro	19	-1	SP	05
04	Maria	20	-1	BH	-1
05	Carlos	18	-1	SP	-1

Idade	Quantidade	Pos. Inicial
18	02	02
19	02	01
20	01	04

Cidade	Quantidade	Pos. Inicial
BH	02	01
RJ	01	02
SP	02	03

Multilistas



Posição	Nome	Idade	Cidade
01	João	19	BH
02	Ana	18	RJ
03	Pedro	19	SP
04	Maria	20	BH
05	Carlos	18	SP

Índice Invertido



PUC Minas

Índice Invertido

- Um índice invertido é um índice em que uma parte do conteúdo de um registro (como uma palavra de um campo) é usada na localização do próprio registro.
- Essa é uma solução para permitir, entre outras, a busca de texto em arquivos, como fazem as máquinas de busca na Web.

Índice Invertido

Cód	Título	
1	Implementação de sistemas de bancos de dados	...
2	Sistemas de bancos de dados	...
3	Estruturas de dados e seus algoritmos	...
4	Dominando algoritmos	...
5	Estruturas de dados em java	...
6	Core Java	...
7	Biblioteca do programador Java	...

Índice Invertido

Cód	Título	.
1	Implementação de sistemas de bancos de dados	...
2	Sistemas de bancos de dados	...
3	Estruturas de dados e seus algoritmos	...
4	Dominando algoritmos	...
5	Estruturas de dados em java	...
6	Core Java	...
7	Biblioteca do programador Java	...

Dados

Índice Invertido

- Busca Sequencial?
- A busca sequencial (testando cada registro) é lenta demais para um grande volume de dados
- Google: pesquisa em bilhões de páginas em uma fração de segundo

Índice Invertido

Google house

Aproximadamente 14.600.000.000 resultados (0,82 segundos)

https://pt.wikipedia.org/wiki/House,_M.D._

House, M.D. – Wikipédia, a encyclopédia livre

Gregory House, interpretado pelo ator inglês Hugh Laurie. A série passa-se no fictício hospital universitário Princeton-Plainsboro Teaching Hospital, em ...

Temporadas: 8 Episódios: 177 (Lista de episódios)

Elenco: Hugh Laurie; Lisa Edelstein; Omar E... Tema de abertura: "Teardrop"

Gregory House - Lista de episódios de House... · Lisa Edelstein · Kal Penn

Principais notícias

Notícias sobre A Casa do Dragão

 **TecMundo** House of the Dragon: fãs encontram erro de CGI no 3º episódio; veja! 5 horas atrás

 **GZH** Com caprichos de Rhaenyra e batalha sangrenta, saiba como foi o terceiro episódio... 10 horas atrás

 **iG Queer** 'House of the Dragon' apresenta primeiro personagem gay 10 horas atrás

 Mais imagens

House, M.D.

2004 · Drama · 8 temporadas

 Assistir agora Assinatura  Assistir Quero assistir

▼ Todas as opções para assistir

89% gostaram desse programa de TV

Usuários do Google

No fictício Hospital-Escola Plainsboro de Princeton,

Patente
Google

Índice Invertido

- Todos os "termos" são identificados e, para cada um deles, criamos uma lista dos registros em que aparecem
- Índice invertido = listas invertidas

Índice Invertido

Cód	Título	.
1	Implementação de sistemas de bancos de dados	...
2	Sistemas de bancos de dados	...
3	Estruturas de dados e seus algoritmos	...
4	Dominando algoritmos	...
5	Estruturas de dados em java	...
6	Core Java	...
7	Biblioteca do programador Java	...

Termos
Implementação
de
sistemas
bancos
dados
estruturas
e
seus
algoritmos
dominando
em
java
core
biblioteca
do
programador

Índice Invertido

Cód	Título	.
1	Implementação de sistemas de bancos de dados	...
2	Sistemas de bancos de dados	...
3	Estruturas de dados e seus algoritmos	...
4	Dominando algoritmos	...
5	Estruturas de dados em java	...
6	Core Java	...
7	Biblioteca do programador Java	...

Termos
Implementação
de
sistemas
bancos
dados
estruturas
e
seus
algoritmos
dominando
em
java
core
biblioteca
do
programador

Retirar termos
não
significativos e
ordenar conjunto

Índice Invertido

Cód	Título	.
1	Implementação de sistemas de bancos de dados	...
2	Sistemas de bancos de dados	...
3	Estruturas de dados e seus algoritmos	...
4	Dominando algoritmos	...
5	Estruturas de dados em java	...
6	Core Java	...
7	Biblioteca do programador Java	...

Termos
Implementação
de
sistemas
bancos
dados
estruturas
e
seus
algoritmos
dominando
em
java
core
biblioteca
do
programador

Retirar termos
não
significativos e
ordenar conjunto

Índice Invertido

Cód	Título	.
1	Implementação de sistemas de bancos de dados	...
2	Sistemas de bancos de dados	...
3	Estruturas de dados e seus algoritmos	...
4	Dominando algoritmos	...
5	Estruturas de dados em java	...
6	Core Java	...
7	Biblioteca do programador Java	...

Termos
implementação
sistemas
bancos
dados
estruturas
seus
algoritmos
dominando
java
core
biblioteca
programador

Índice Invertido

Cód	Título	.
1	Implementação de sistemas de bancos de dados	...
2	Sistemas de bancos de dados	...
3	Estruturas de dados e seus algoritmos	...
4	Dominando algoritmos	...
5	Estruturas de dados em java	...
6	Core Java	...
7	Biblioteca do programador Java	...

Termos
algoritmos
bancos
biblioteca
core
dados
dominando
estruturas
implementação
java
programador
seus
sistemas

Índice Invertido

Cód	Título	.
1	Implementação de sistemas de bancos de dados	...
2	Sistemas de bancos de dados	...
3	Estruturas de dados e seus algoritmos	...
4	Dominando algoritmos	...
5	Estruturas de dados em java	...
6	Core Java	...
7	Biblioteca do programador Java	...

Termos	Registros
algoritmos	3 4
bancos	1 2
biblioteca	7
core	6
dados	1 2 3 5
dominando	4
estruturas	3 5
implementação	1
java	5 6 7
programador	7
seus	3
sistemas	1 2

Índice Invertido - Consultas

Termos	Registros
algoritmos	3 4
bancos	1 2
biblioteca	7
core	6
dados	1 2 3 5
dominando	4
estruturas	3 5
implementação	1
java	5 6 7
programador	7
seus	3
sistemas	1 2

Consulta: “estrutura **de** dados”

Primeiro Termo: “estruturas”
Conjunto: {3,5}

Repetir:
Próximo termo: “dados”
Conjunto teste: {1,2,3,5}
Conjunto atual: {3,5}
Conjunto resposta: {3,5}
(interseção)

Índice Invertido

- Outro exemplo:

Posição	Nome	Idade	Cidade
01	João	19	BH
02	Ana	18	RJ
03	Pedro	19	SP
04	Maria	20	BH
05	Carlos	18	SP

- Inversão Por Idade

Idade	Referências
18	02 05
19	01 03
20	04

Inversão Por Cidade

Cidade	Referências
BH	01 04
RJ	02
SP	03 05

Índice Invertido - Vantagens

- Os índices invertidos (ou listas invertidas) podem ser construídos para qualquer conjunto de informações dos registros.
- Esses índices são estruturas adequadas para consultas combinadas (vários campos).
- Implementação relativamente fácil
- É a estrutura de dados mais popular usada em sistemas de recuperação de documentos, usada em larga escala, por exemplo, em mecanismos de busca

Índice Invertido - Desvantagens

- Grande sobrecarga de armazenamento e altos custos de manutenção na atualização, exclusão e inserção.

Índice Invertido - TP1

Orientações sobre a criação da lista invertida.

- Deve-se criar um arquivo contendo a lista invertida para o nome da pessoa.
- Deve-se criar um arquivo contendo a lista invertida para a cidade da pessoa.
- O sistema deverá realizar alterações nas listas invertidas sempre que novos registros forem inseridos, alterados ou deletados no arquivo de dados.
- O sistema deve ser capaz de receber uma busca por nome e/ou cidade.
- A partir do nome e/ou cidade digitados, o sistema deve ser capaz de retornar em quais ids de registros a pesquisa foi capaz de encontrar os termos informados.