TABELAS HASH

Vanessa Braganholo

Estruturas de Dados e Seus Algoritmos

MOTIVAÇÃO

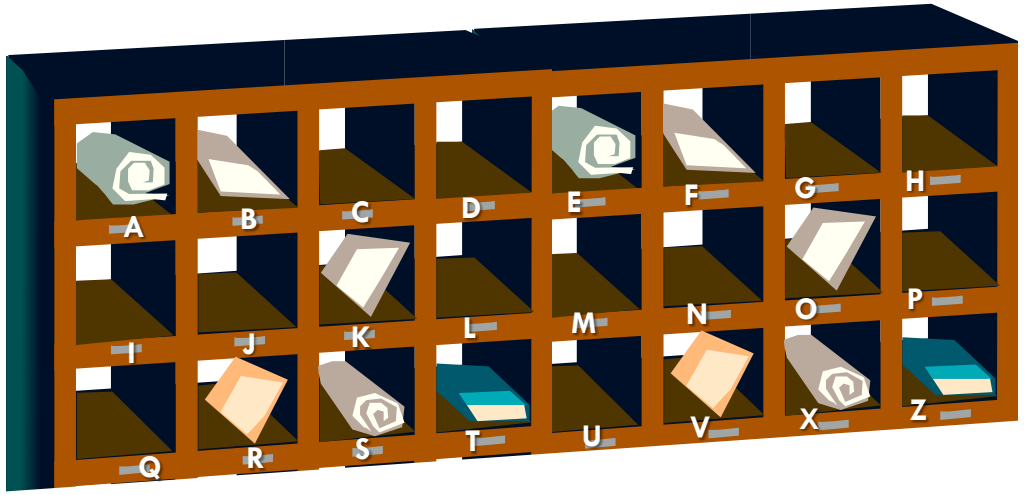
Alternativas para acelerar buscas em grandes volumes de dados:

- Usar um índice (ex. Árvore B, Árvore B+)
- Usar cálculo de endereço para acessar diretamente o registro procurado em O(1) ➔ **Tabelas Hash**

EXEMPLO MOTIVADOR

Distribuição de correspondências de funcionários numa empresa

- Um escaninho para cada inicial de sobrenome
- Todos os funcionários com a mesma inicial de sobrenome procuram sua correspondência dentro do mesmo escaninho
  - Pode haver mais de uma correspondência dentro do mesmo escaninho



HASHING: PRINCÍPIO DE FUNCIONAMENTO

Suponha que existem **n** chaves a serem armazenadas numa tabela de comprimento **m**

- Em outras palavras, a tabela tem **m** compartimentos
- Endereços possíveis: [**0, m-1**]

- Situações possíveis: cada compartimento da tabela pode armazenar **x** registros
- Para simplificar, vamos começar assumindo que **x = 1** (cada compartimento armazena apenas **1** registro)

COMO DETERMINAR M?

Uma opção é determinar **m** em função do número de valores possíves das chaves a serem armazenadas

HASHING: PRINCÍPIO DE FUNCIONAMENTO

Se os valores das chaves variam de [0, m-1], então podemos usar o valor da chave para definir o endereço do compartimento onde o registro será armazenado

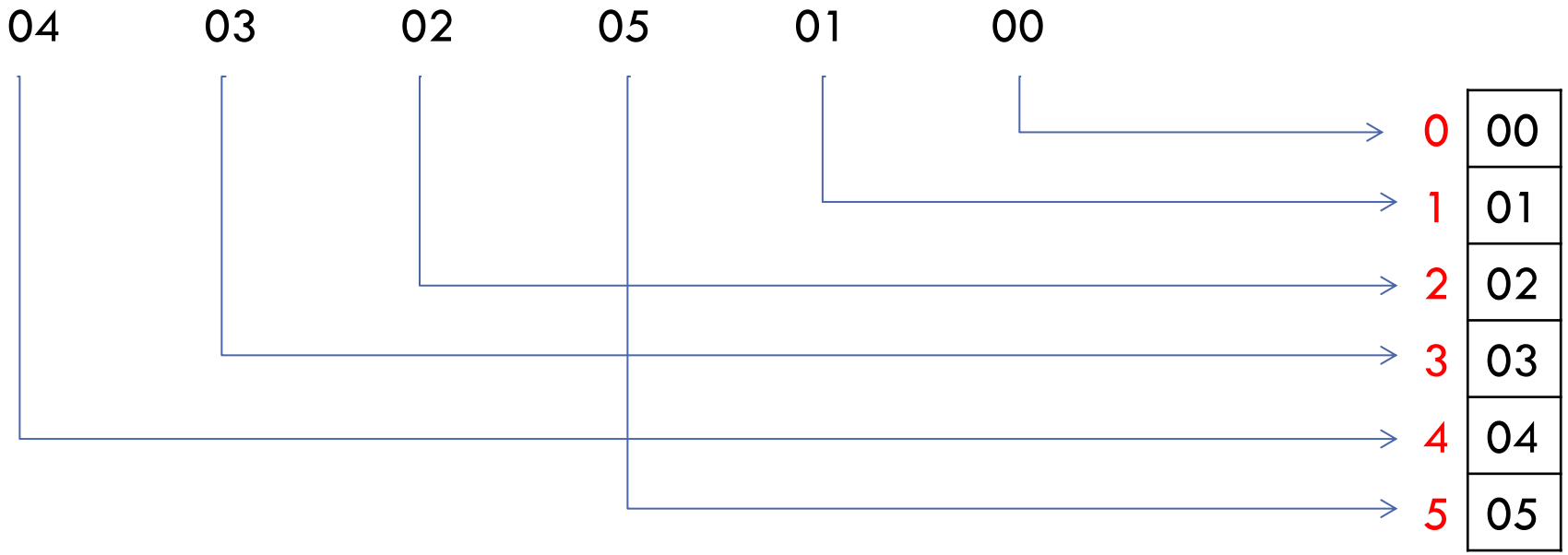
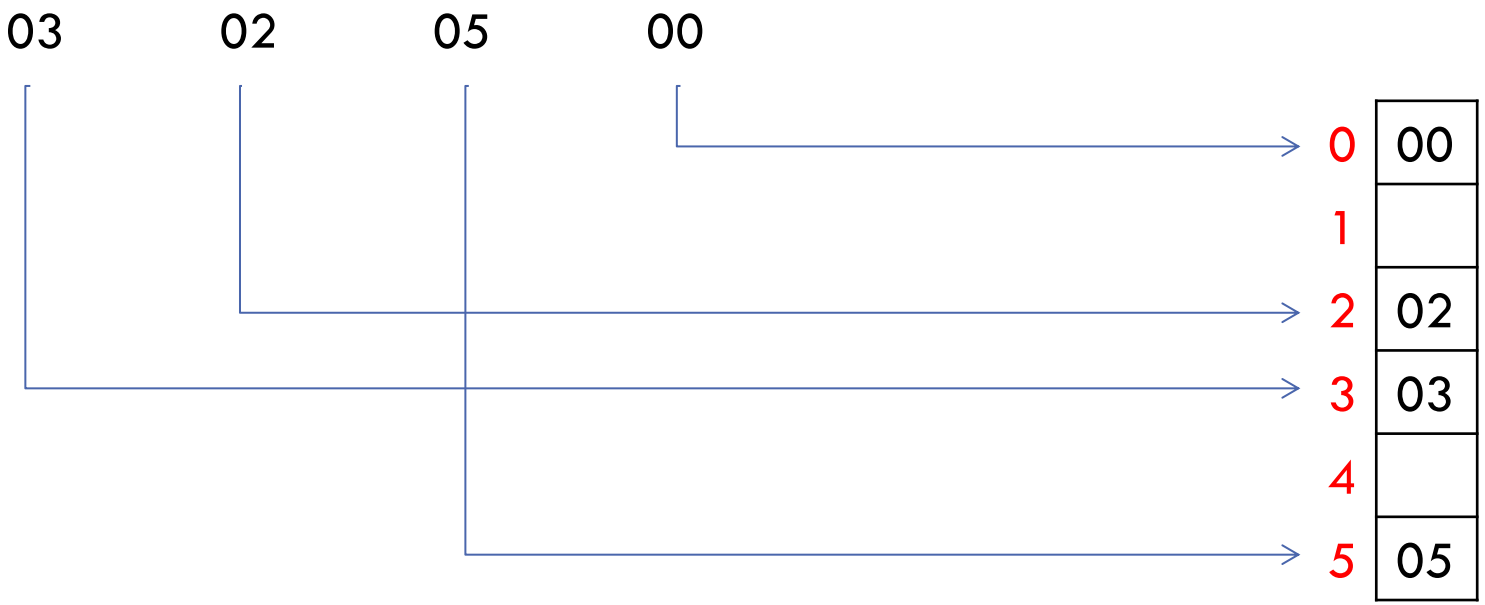


TABELA PODE TER ESPAÇOS VAZIOS

Se o número **n** de chaves a armazenar é menor que o número de compartimentos **m** da tabela



MAS...

Se o intervalo de valores de chave é muito grande, **m** é muito grande

Pode haver um número proibitivo de espaços vazios na tabela se houver poucos registros

Exemplo: armazenar 2 registros com chaves 0 e 999.999 respectivamente

- **m** = 1.000.000
- tabela teria 999.998 compartimentos vazios

SOLUÇÃO

Definir um valor de **m** menor que os valores de chaves possíveis

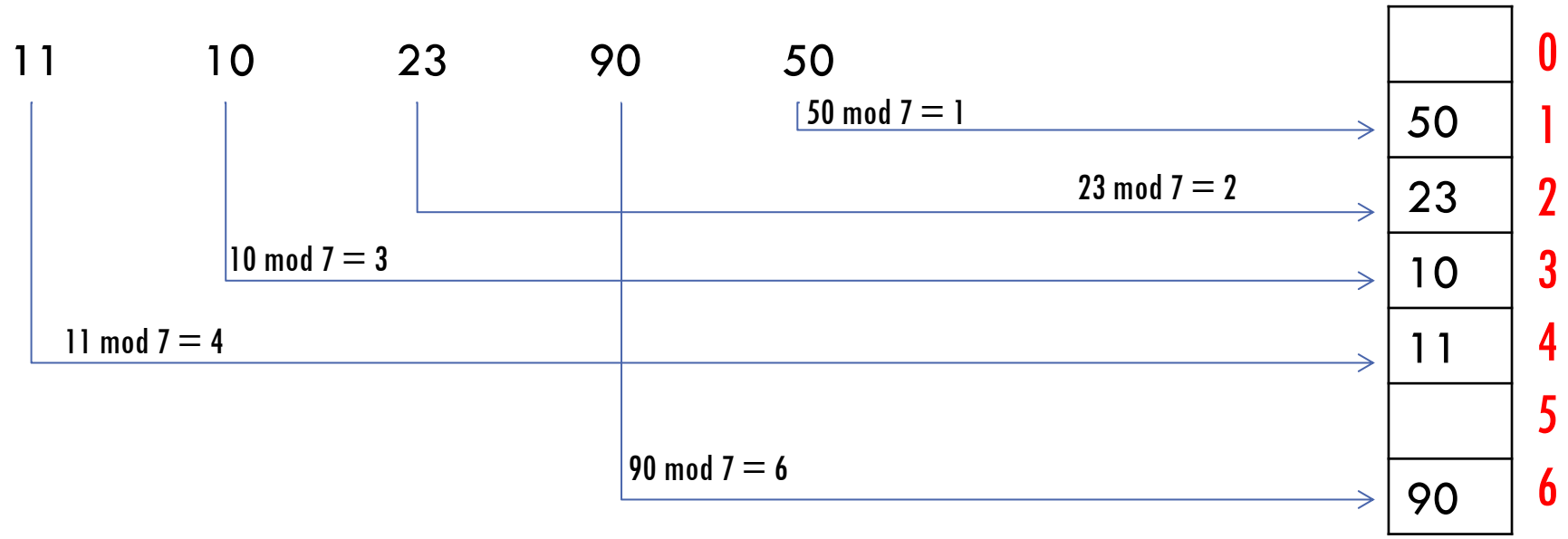
Usar uma função hash **h** que mapeia um valor de chave **x** para um endereço da tabela

Se o endereço **h(x)** estiver livre, o registro é armazenado no compartimento apontado por **h(x)**

Diz-se que **h(x)** produz um **endereço-base** para **x**

EXEMPLO

h(x) = x mod 7



FUNÇÃO HASH H

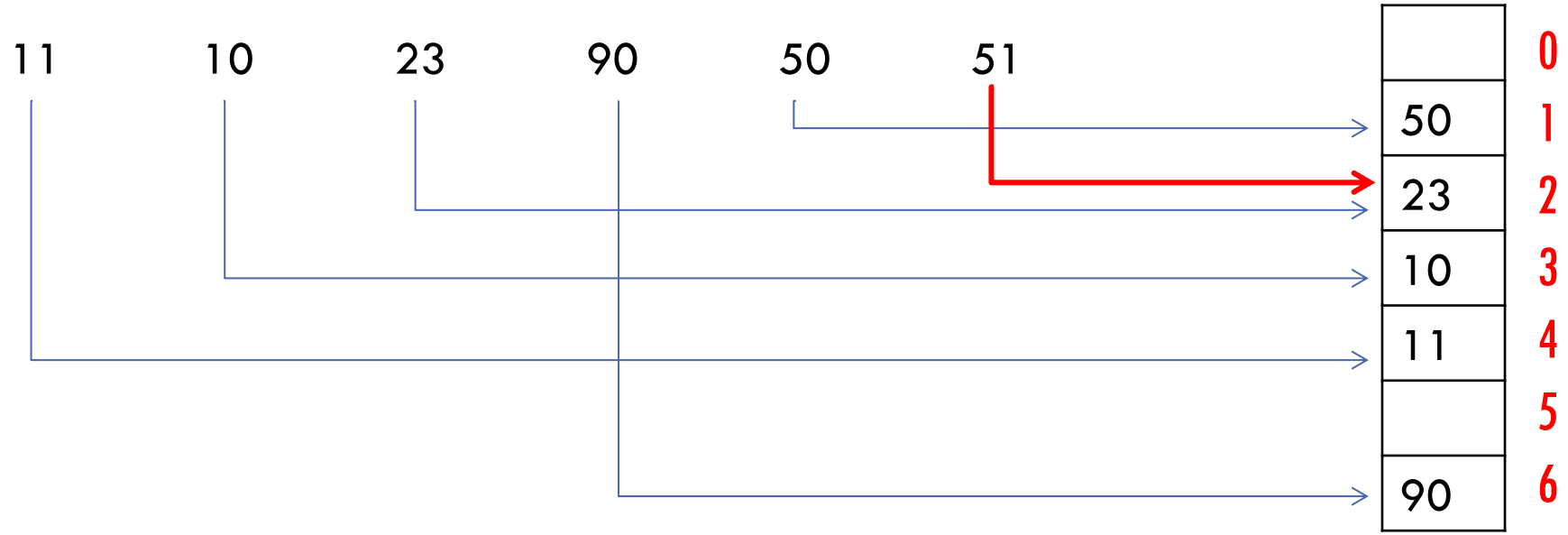
Infelizmente, a função pode não garantir injetividade, ou seja, é possível que **x ≠ y** e **h(x) = h(y)**

Se ao tentar inserir o registro de chave **x** o compartimento de endereço **h(x)** já estiver ocupado por **y**, ocorre uma **colisão**

- Diz-se que **x** e **y** são **sinônimos** em relação a **h**

EXEMPLO: COLISÃO

h(x) = x mod 7



A chave 51 colide com a chave 23 e não pode ser inserida no endereço 2!

Solução: uso de um procedimento especial para armazenar a chave 51 (tratamento de colisões)

CARACTERÍSTICAS DESEJÁVEIS DAS FUNÇÕES DE HASH

Produzir um número baixo de colisões

Ser facilmente computável

Ser uniforme

CARACTERÍSTICAS DESEJÁVEIS DAS FUNÇÕES DE HASH

Produzir um **número baixo de colisões**

- Difícil, pois depende da distribuição dos valores de chave
- Exemplo:
  - Pedidos que usam o ano e mês do pedido como parte da chave
  - Se a função **h** realçar estes dados, haverá muita concentração de valores nas mesmas faixas

CARACTERÍSTICAS DESEJÁVEIS DAS FUNÇÕES DE HASH

Ser **facilmente computável**

- Se a tabela estiver armazenada em disco, isso não é tão crítico, pois a operação de I/O é muito custosa, e dilui este tempo
- Das 3 condições, é a mais fácil de ser garantida

Ser **uniforme**

- Idealmente, a função **h** deve ser tal que todos os compartimentos possuam a mesma probabilidade de serem escolhidos
- Difícil de testar na prática

EXEMPLOS DE FUNÇÕES DE HASH

Algumas funções de hash são bastante empregadas na prática por possuírem algumas das características anteriores

- Método da Divisão
- Método da Dobra
- Método da Multiplicação

EXEMPLOS DE FUNÇÕES DE HASH

Método da Divisão

Método da Dobra

Método da Multiplicação

MÉTODO DA DIVISÃO

Uso da função mod:

h(x) = x mod m

onde **m** é a dimensão da tabela

Alguns valores de **m** são melhores do que outros

- Exemplo: se **m** for par, então **h(x)** será par quando **x** for par, e ímpar quando **x** for ímpar → indesejável

Atenção: na pag. 235 do livro, a fórmula contém um pequeno erro

MÉTODO DA DIVISÃO

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

EXEMPLOS DE FUNÇÕES DE HASH

Método da Divisão

Método da Dobra

Método da Multiplicação

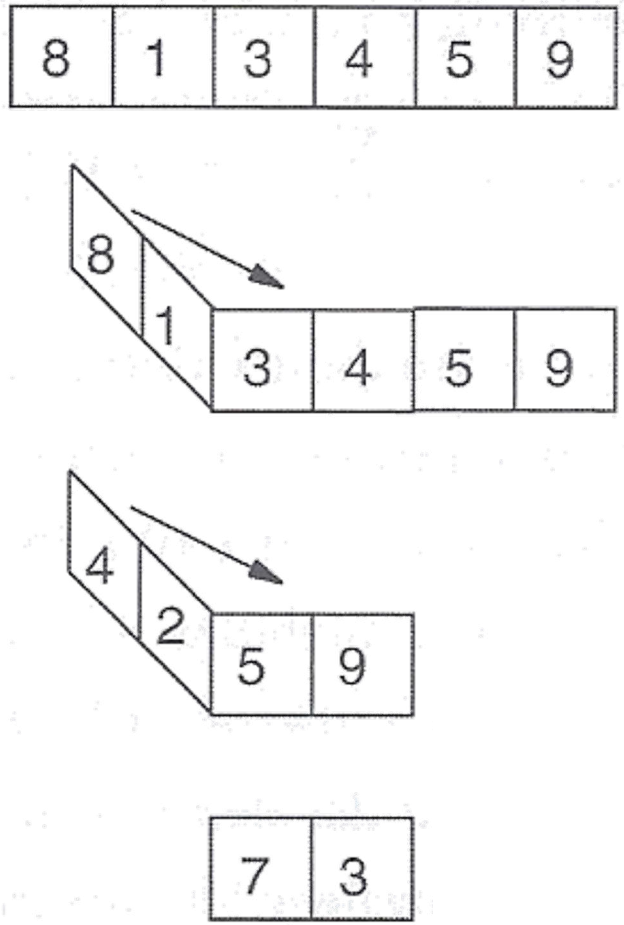
MÉTODO DA DOBRA

Suponha a chave como uma sequencia de dígitos escritos em um pedaço de papel

O método da dobra consiste em “dobrar” este papel, de maneira que os dígitos se superponham

Os dígitos então devem ser somados, sem levar em consideração o “vai-um”

EXEMPLO: MÉTODO DA DOBRA



8+4=~~1~~2

1+3=4

4+9=~~1~~3

2+5=7

Fonte: Fig. 10.4, pag 237

MÉTODO DA DOBRA

A posição onde a dobra será realizada, e quantas dobras serão realizadas, depende de quantos dígitos são necessários para formar o endereço base

O tamanho da dobra normalmente é do tamanho do endereço que se deseja obter

EXEMPLOS DE FUNÇÕES DE HASH

Método da Divisão

Método da Dobra

Método da Multiplicação

MÉTODO DA MULTIPLICAÇÃO

Multiplicar a chave por ela mesma

Armazenar o resultado numa palavra de **b** bits

Descartar os bits das extremidades direita e esquerda, um a um, até que o resultado tenha o tamanho de endereço desejado

MÉTODO DA MULTIPLICAÇÃO

Exemplo: chave 12

- 12 x 12 = 144
- 144 representado em binário: 10010000
- Armazenar em 10 bits: 0010010000
- Obter endereço de 6 bits (endereços entre 0 e 63)

MÉTODO DA MULTIPLICAÇÃO

Exemplo: chave 12

- 12 x 12 = 144
- 144 representado em binário: 10010000
- Armazenar em 10 bits: 0010010000
- Obter endereço de 6 bits (endereços entre 0 e 63)

MÉTODO DA MULTIPLICAÇÃO

Exemplo: chave 12

- 12 x 12 = 144
- 144 representado em binário: 10010000
- Armazenar em 10 bits: 0010010000
- Obter endereço de 6 bits (endereços entre 0 e 63)

MÉTODO DA MULTIPLICAÇÃO

Exemplo: chave 12

- 12 x 12 = 144
- 144 representado em binário: 10010000
- Armazenar em 10 bits: 0010010000
- Obter endereço de 6 bits (endereços entre 0 e 63)

MÉTODO DA MULTIPLICAÇÃO

Exemplo: chave 12

- 12 x 12 = 144
- 144 representado em binário: 10010000
- Armazenar em 10 bits: 0010010000
- Obter endereço de 6 bits (endereços entre 0 e 63)

MÉTODO DA MULTIPLICAÇÃO

Exemplo: chave 12

- 12 x 12 = 144
- 144 representado em binário: 10010000
- Armazenar em 10 bits: 0010010000
- Obter endereço de 6 bits (endereços entre 0 e 63)

USO DA FUNÇÃO DE HASH

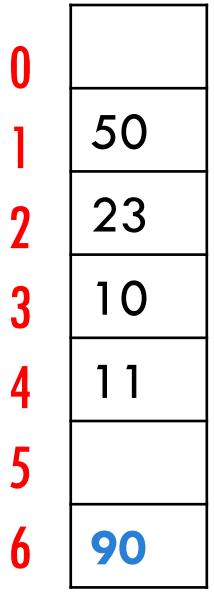
A mesma função de hash usada para inserir os registros é usada para buscar os registros

EXEMPLO: BUSCA DE REGISTRO POR CHAVE

h(x) = x mod 7

Encontrar o registro de chave 90

- 90 mod 7 = 6



EXEMPLO: BUSCA DE REGISTRO POR CHAVE

h(x) = x mod 7

Encontrar o registro de chave 7

- 7 mod 7 = 0
- Compartimento 0 está vazio: registro não está armazenado na tabela

0	
1	50
2	23
3	10
4	11
5	
6	90

EXEMPLO: BUSCA DE REGISTRO POR CHAVE

h(x) = x mod 7

Encontrar o registro de chave 8

- 8 mod 7 = 1
- Compartimento 1 tem um registro com chave diferente da chave buscada, e não existem registros adicionais: registro não está armazenado na tabela

0	
1	50
2	23
3	10
4	11
5	
6	90

IMPLEMENTAÇÃO BÁSICA EM MEMÓRIA PRINCIPAL

Ver código da implementação básica no site da disciplina

Observações:

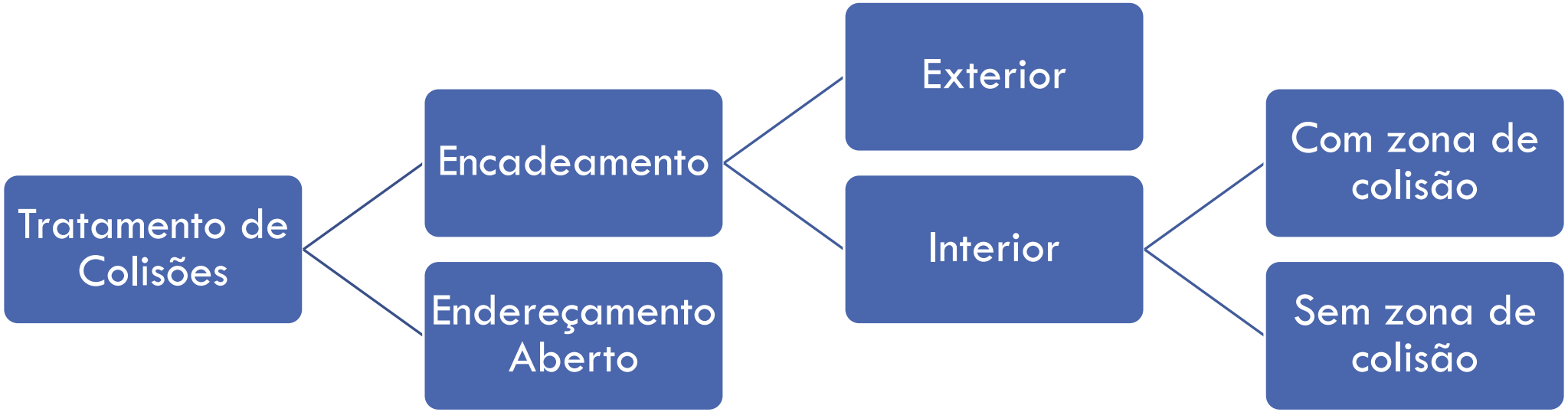
- Caso compartimento já esteja ocupado, **inserção é cancelada (não faz sentido na prática!!)**
- Para evitar isso, é necessário **tratar colisões**

FATOR DE CARGA

O fator de carga de uma tabela hash é $\alpha = n/m$, onde **n** é o número de registros armazenados na tabela

- O número de colisões cresce rapidamente quando o fator de carga aumenta
- Uma forma de diminuir as colisões é diminuir o fator de carga
- Mas **isso não resolve o problema**: colisões sempre podem ocorrer

Como tratar as colisões?


Tratamento de Colisões
Encadeamento
Endereçamento Aberto
Exterior
Interior
Com zona de colisão
Sem zona de colisão

REFERÊNCIA

Szwarcfiter, J.; Markezon, L. Estruturas de Dados e seus Algoritmos, 3a. ed. LTC. Cap. 10