Hashing

Introdução

- Uma Tabela Hash, também conhecida como **tabela de dispersão** ou **tabela de espalhamento**, é uma estrutura de dados especial, que associa chaves e valores.

- A ideia é que a partir de uma chave simples, fazer uma busca rápida e obter o valor desejado.

- Na tabela hash tem-se:
  - Que o valor absoluto de cada chave é interpretado como um valor numérico
  - Através da aplicação de uma função conveniente, a chave é transformada em um endereço de uma tabela

Funcionamento

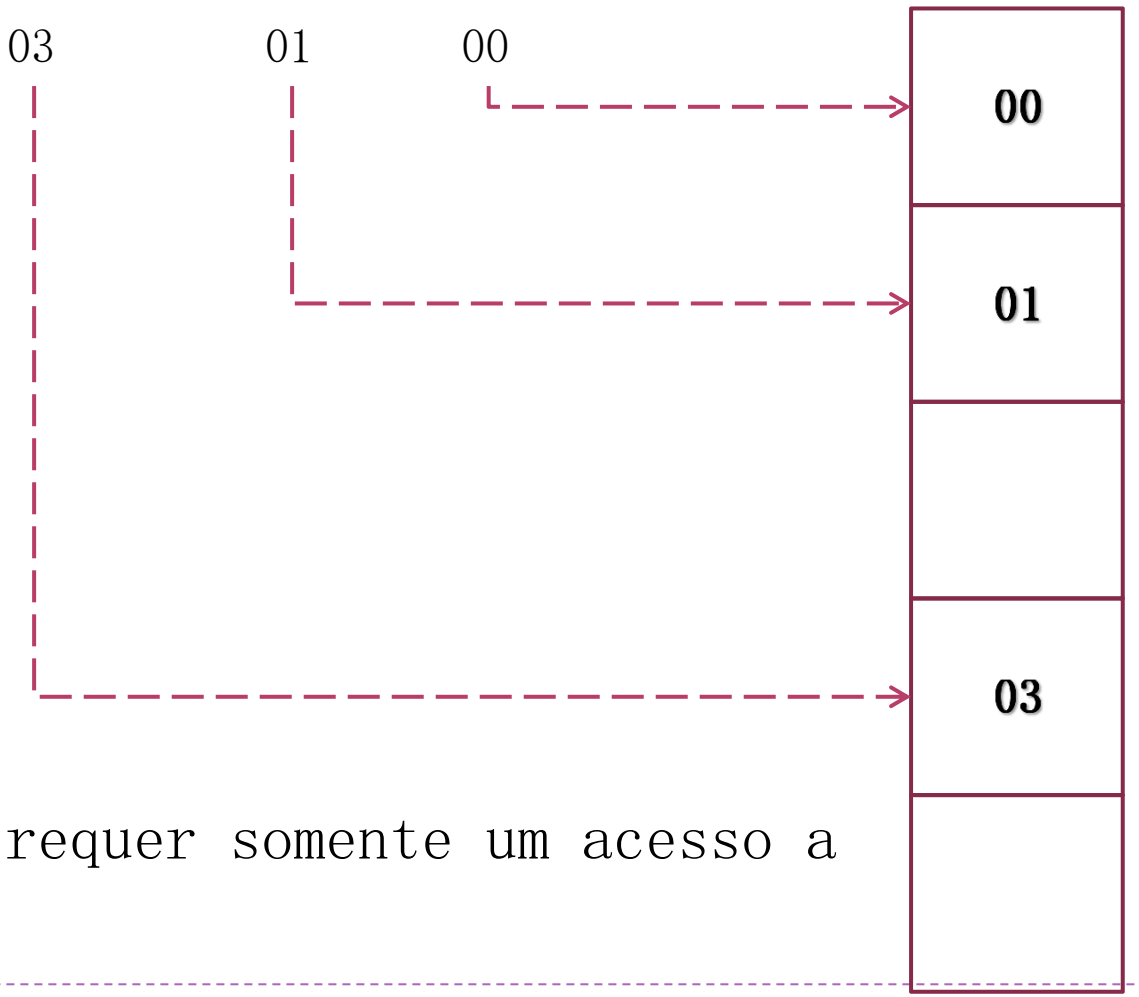
- Suponha que existam **n** chaves a serem armazenadas em uma tabela T, sequencial e de tamanho m.

- As posições da tabela se situam no intervalo [0, m-1].

  - a tabela é particionada em **m** compartimentos
  - cada um corresponde a um endereço e pode armazenar **r** nós distintos.
  - O objetivo é armazenar cada chave no bloco referente ao seu endereço.

Funcionamento

- Valor da chave como seu índice na tabela.
- Cada chave x é adicionada no compartimento x.



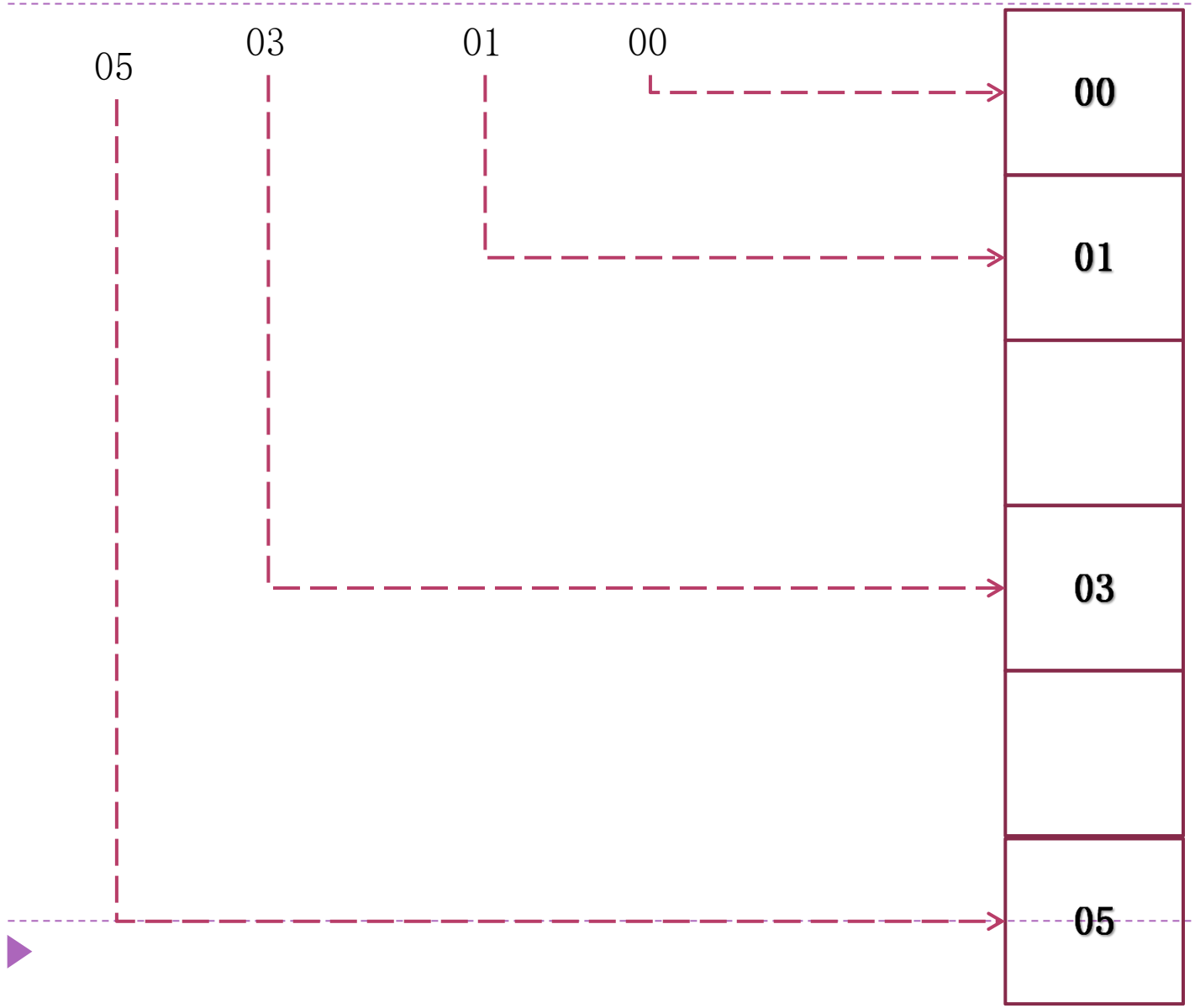
A busca, assim, requer somente um acesso a um bloco.

Funcionamento

- Chaves nem sempre são valores numéricos.
  - As chaves podem consistir em nomes de pessoas.

- Solução: Todo dado não numérico corresponde uma representação numérica no computador.
  - todas as chaves são consideradas numéricas.

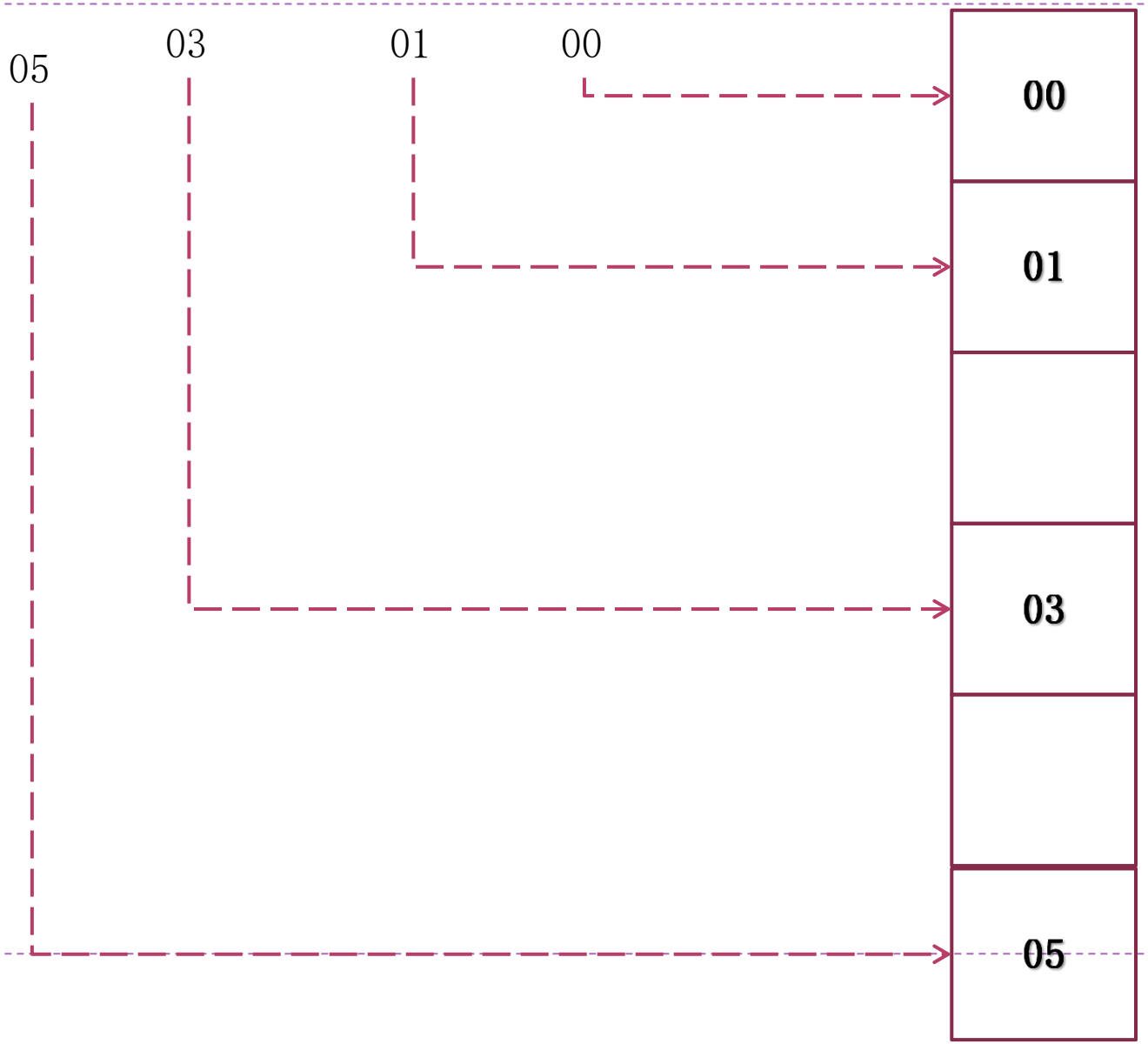
Funcionamento



Qual é o problema com essa abordagem?

Como resolver?

Funcionamento



Qual é o problema com essa abordagem?

Como resolver?

- Transformar cada chave x em um valor no intervalo [0, m-1].
- Através de uma função de dispersão h.

Funcionamento

- Dada uma chave **x**, determina-se o valor **h(x)**: endereço-base.
  - Se o compartimento **h(x)** estiver desocupado, poderá ser utilizado para armazenar **x**.
  - Uma função de dispersão **h** transforma uma chave **x** em um endereço-base **h(x)** da tabela de dispersão.
- Uma função de dispersão deve satisfazer às seguintes condições:
  - produzir um número baixo de colisão;
  - ser facilmente computável;
  - ser uniforme.

Função hash

As mais comuns são:

- Método da Divisão
- Método da Multiplicação

Função hash – Método da divisão

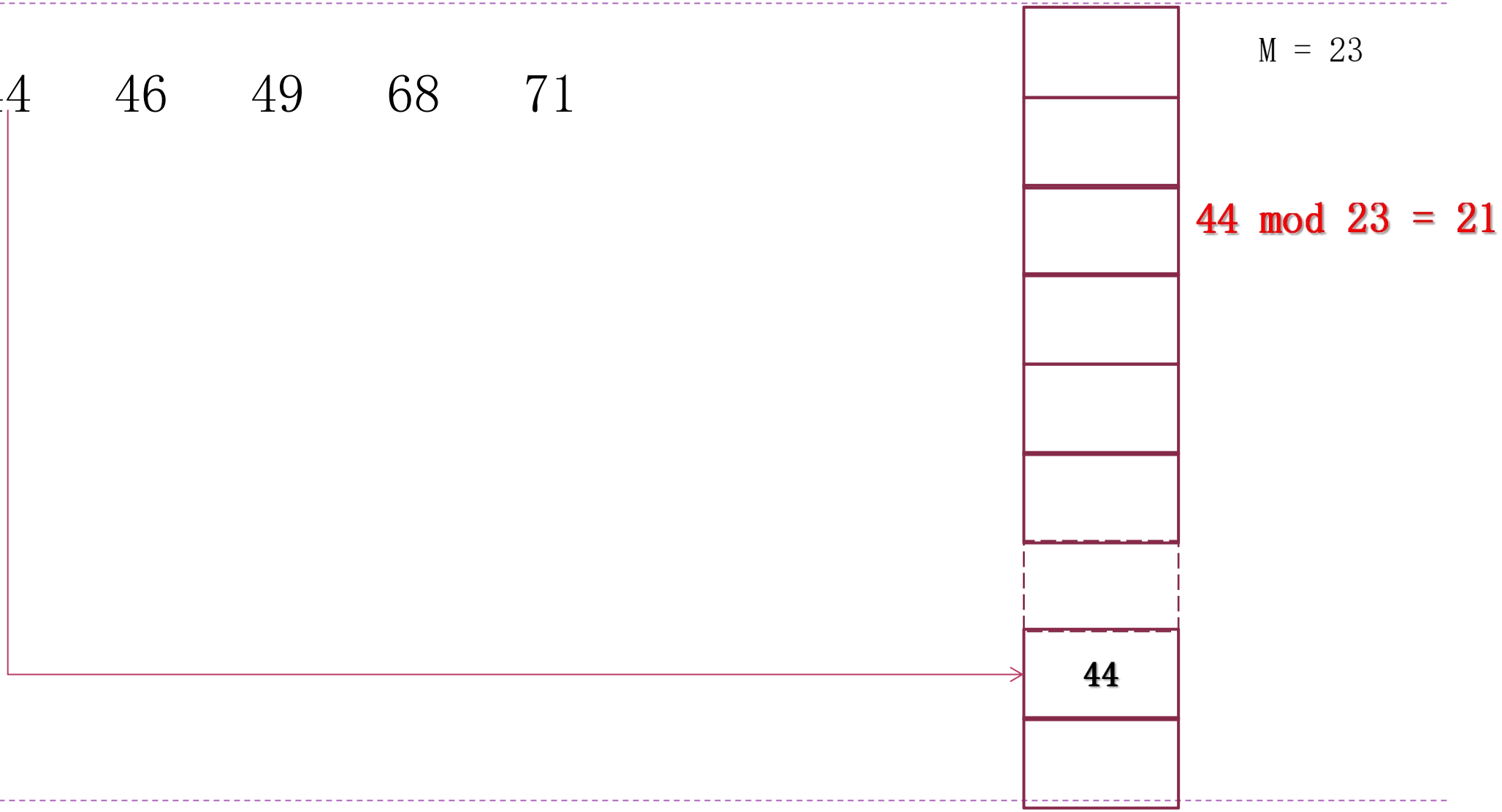
- Princípio: a chave **x** é dividida pela dimensão da tabela **m**, e o resto da divisão é usado com endereço chave.

- Endereços no intervalo

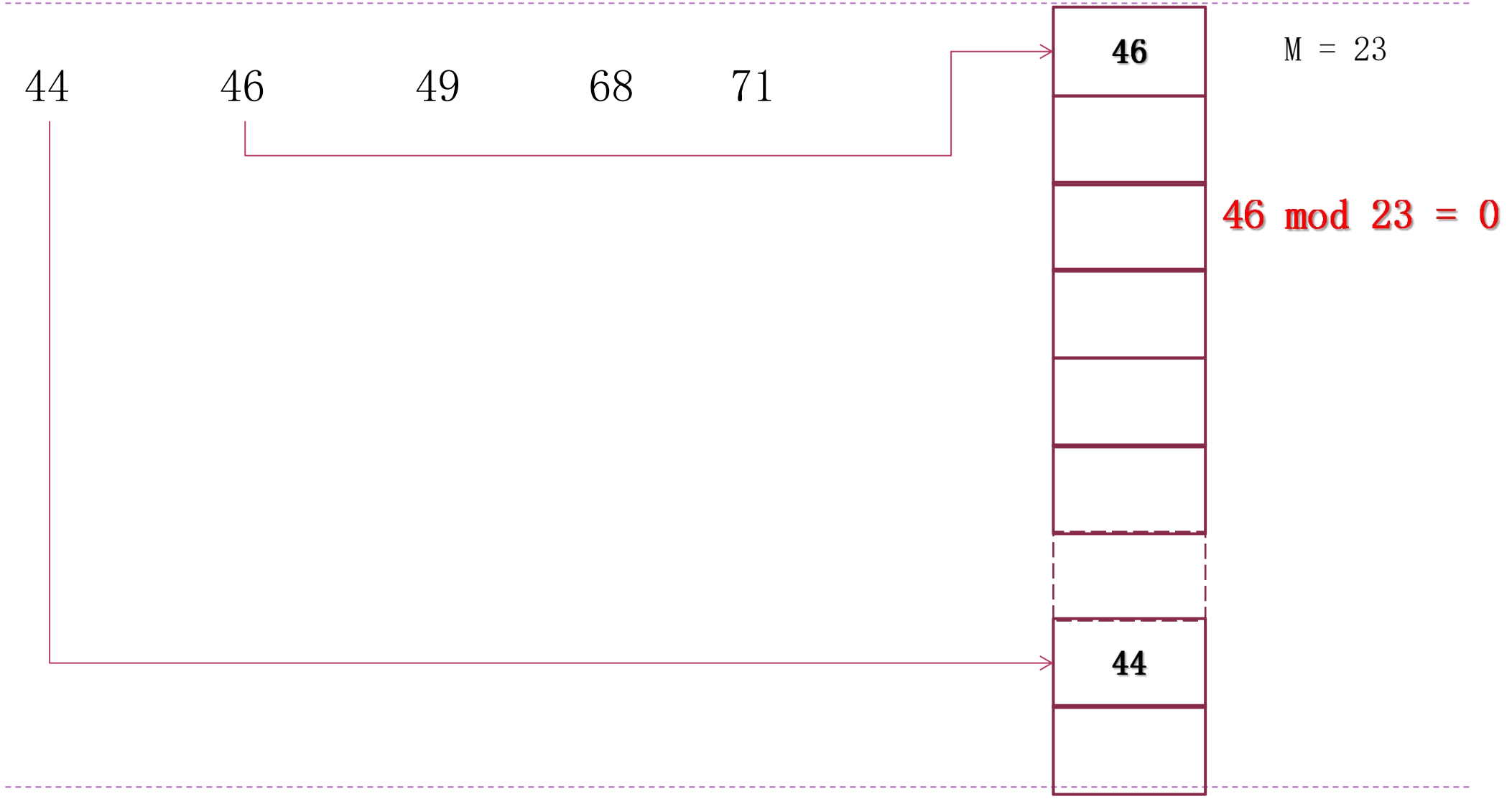
$$h(x) = x \ mod \ m$$

Função hash – Método da divisão

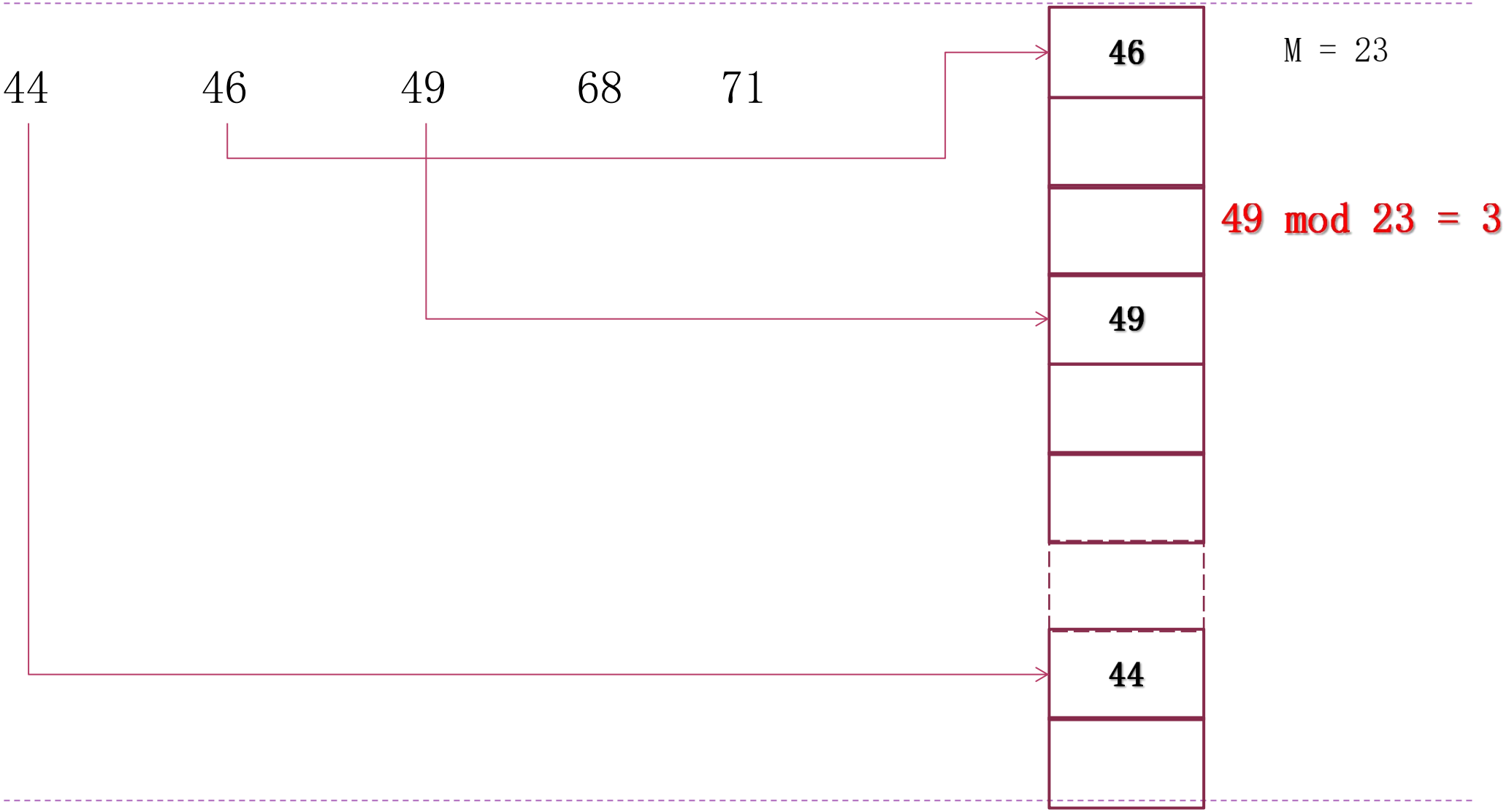
44 46 49 68 71



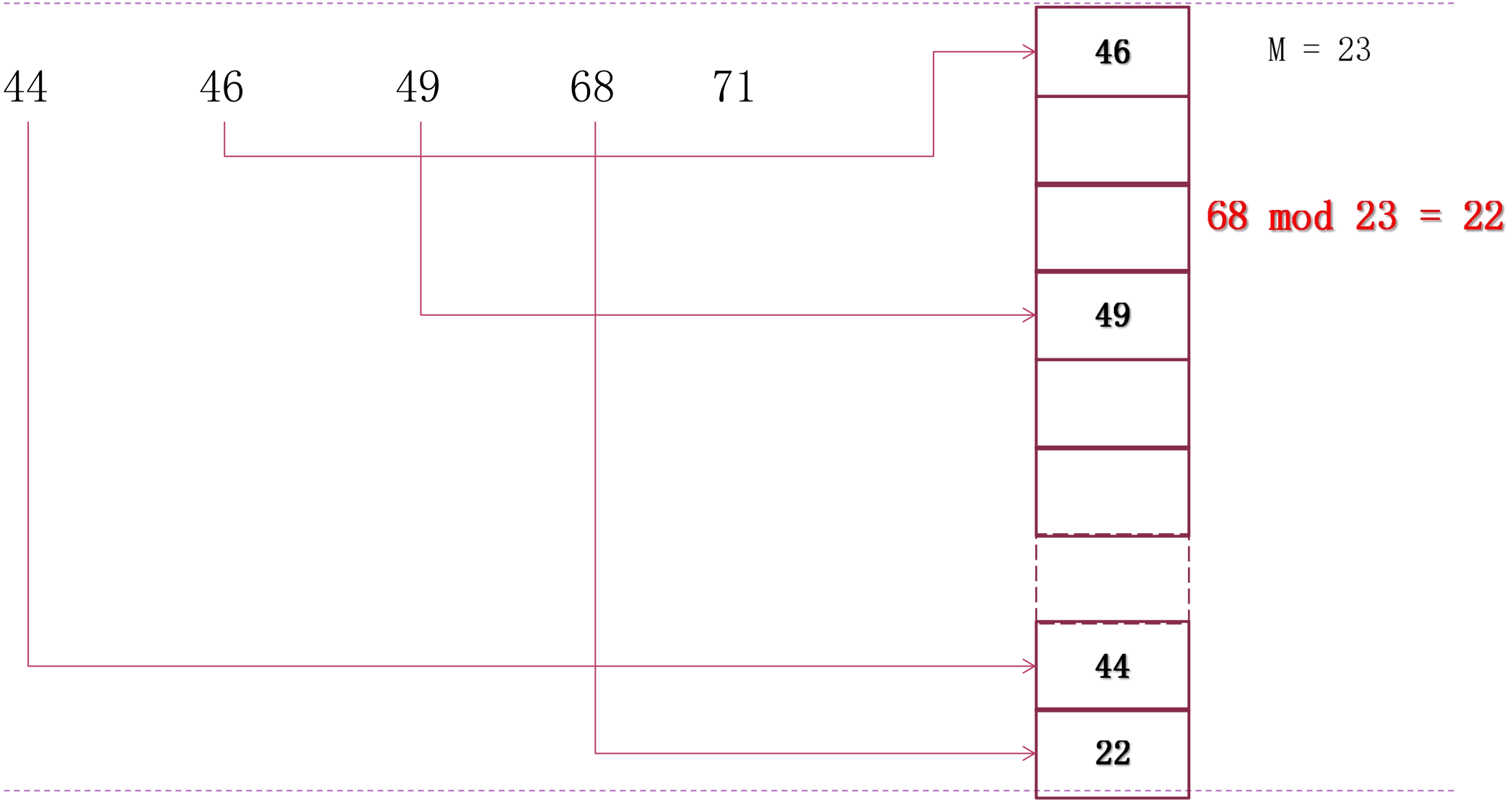
Função hash – Método da divisão



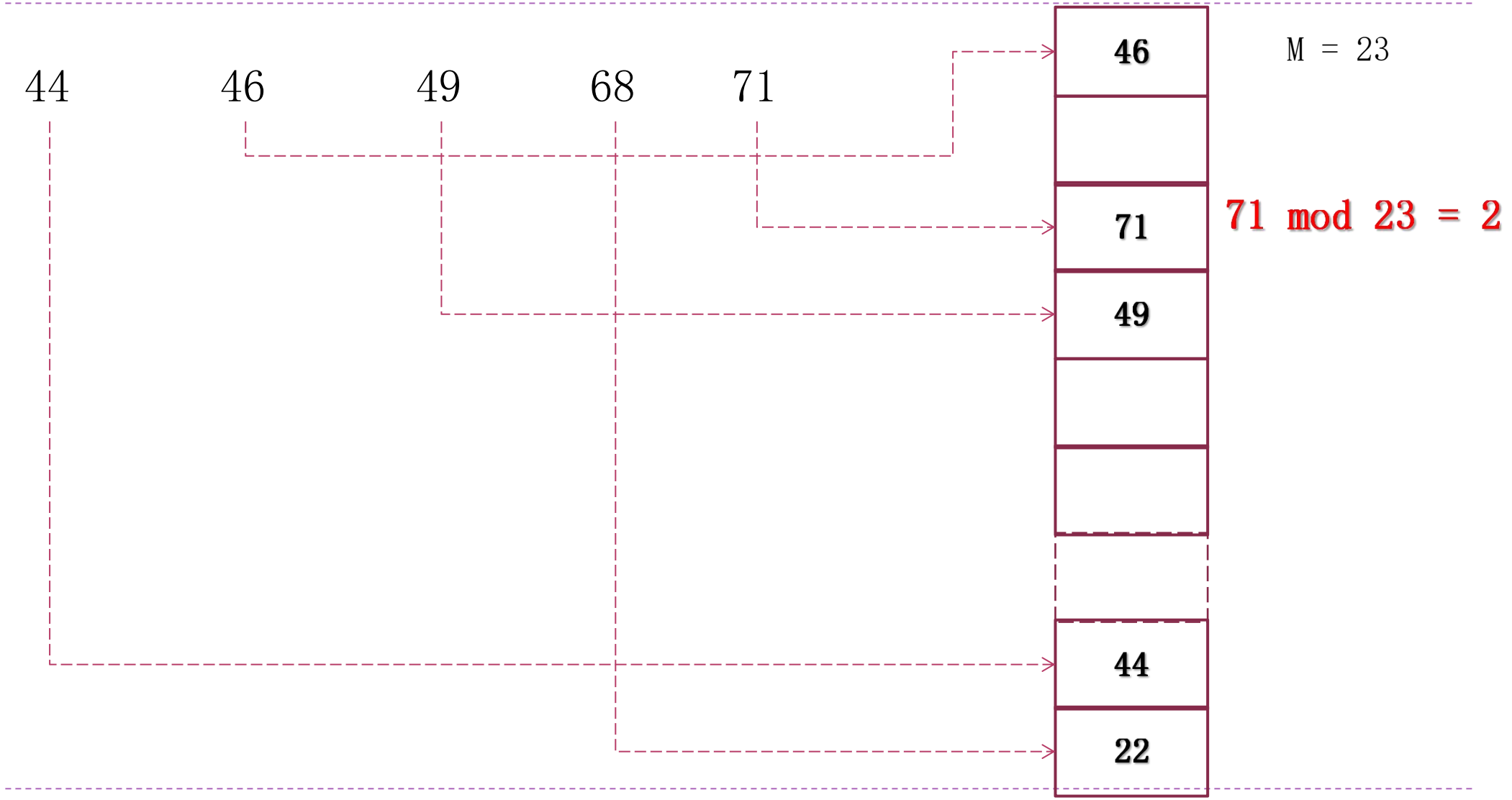
Função hash – Método da divisão



Função hash – Método da divisão



Função hash – Método da divisão



Função hash – Método da multiplicação

- Método **meio do quadrado**

- Princípio: A chave é multiplicada por ela mesma.
  - O resultado é armazenada em palavra de memória de **b** bits.
  - O número de bits necessário para formar o endereço-base de uma chave é então retirado dos **b** bits.
  - Descartando-se os bits excessivos da extrema direita e da extrema esquerda.

Tratamento de colisão

- É possível a existência de outra chave **y** $\neq$ **x**, tal que h(y) = h(x).
- Esse fenômeno é denominado de colisão.
- As chaves **x** e **y** são sinônimas em relação a **h**.
- Para tratar esse problema utiliza-se o procedimento de tratamento de colisão.

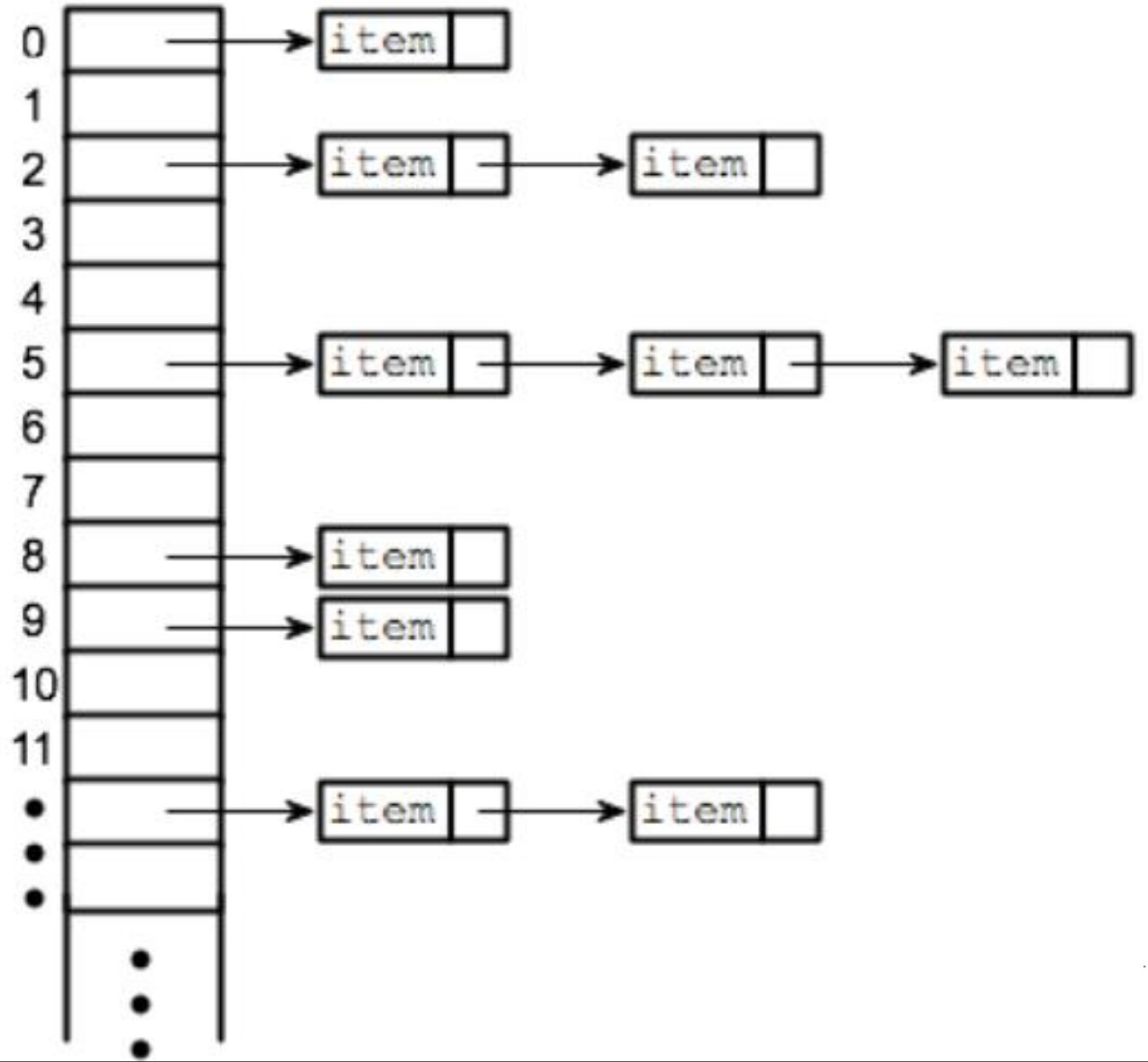
Tratamento de colisão

- Uma ideia para tratar colisões consiste em armazenar chaves sinônimas em listas encadeadas:
  - Encadeamento exterior
  - Encadeamento interior

Tratamento de colisão – encadeamento exterior

- Consiste em manter **m** listas encadeadas, uma para cada possível endereço base.
  - Um campo para o encadeamento dever ser acrescentado a cada nó.
  - O nós correspondentes ao endereço base serão apenas nós-cabeças para essas listas.

Tratamento de colisão – encadeamento exterior



Tratamento de colisão – encadeamento interior

- Usa-se listas encadeadas, desde que estas compartilhem o mesmo espaço de memória que a tabela de dispersão.

- O encadeamento interior prevê a divisão da tabela **T** em duas zonas, uma de endereço-base, de tamanho **p**, e outra reservada aos sinônimos, de tamanho s.

Tratamento de colisão – encadeamento interior

- Cada nó da tabela possui dois campos, um para armazenar o valor, e outro um ponteiro que indica o próximo elemento da lista de sinônimos correspondentes ao endereço-base em questão.

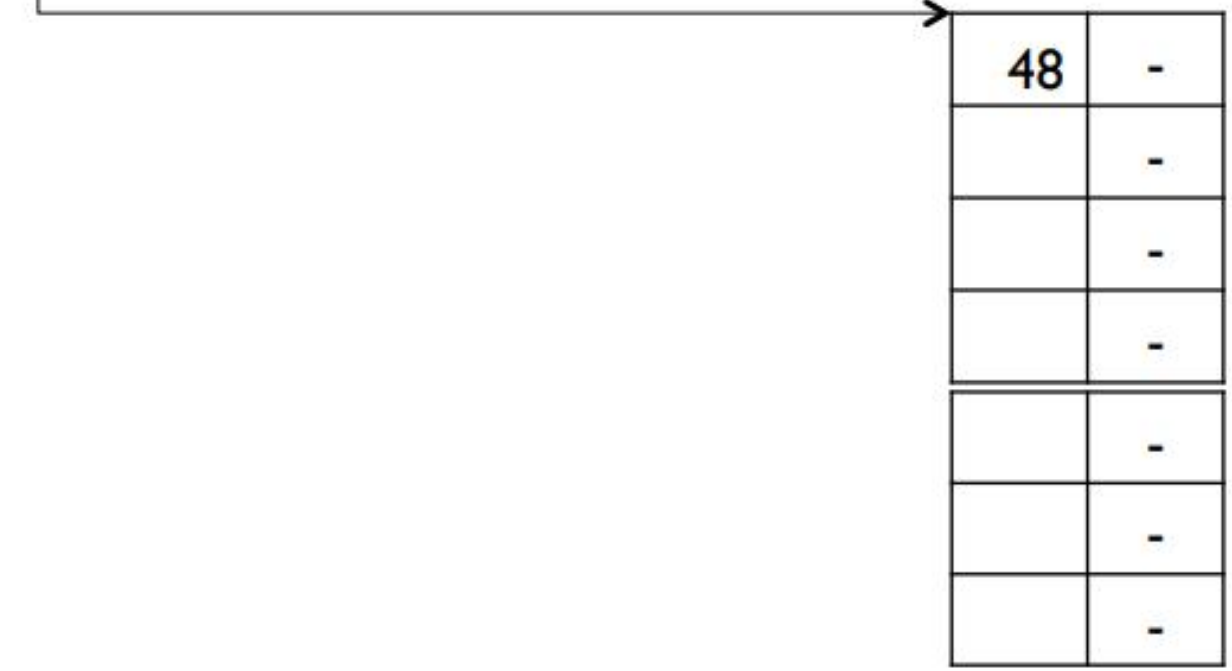
- n = 5
- m = 7
- p = 4 e s = 3
- Função de dispersão: h(x) = x mod 4 M

Tratamento de colisão – encadeamento interior

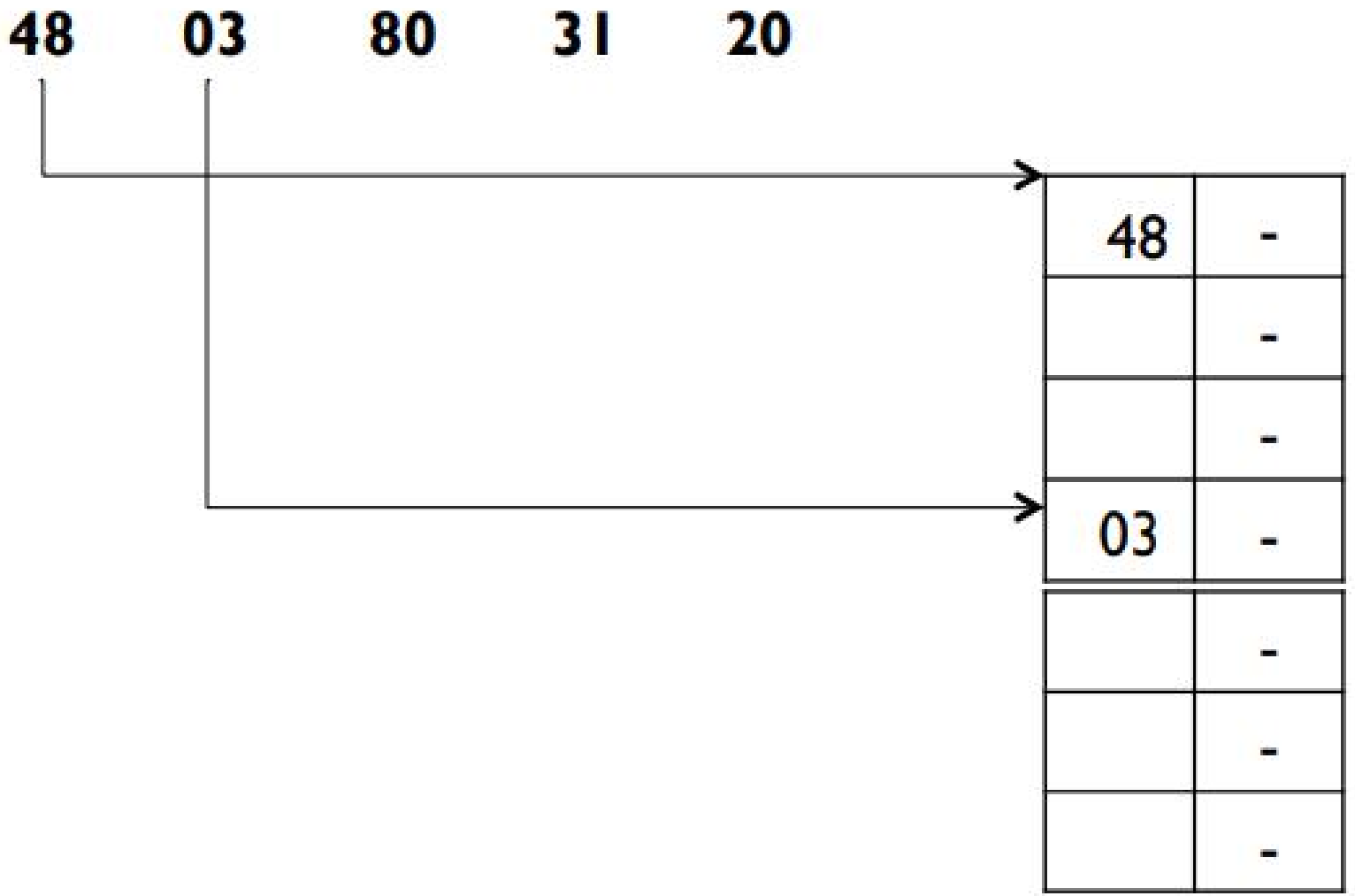
m = 7

48 mod 4= 0

48 03 80 31 20



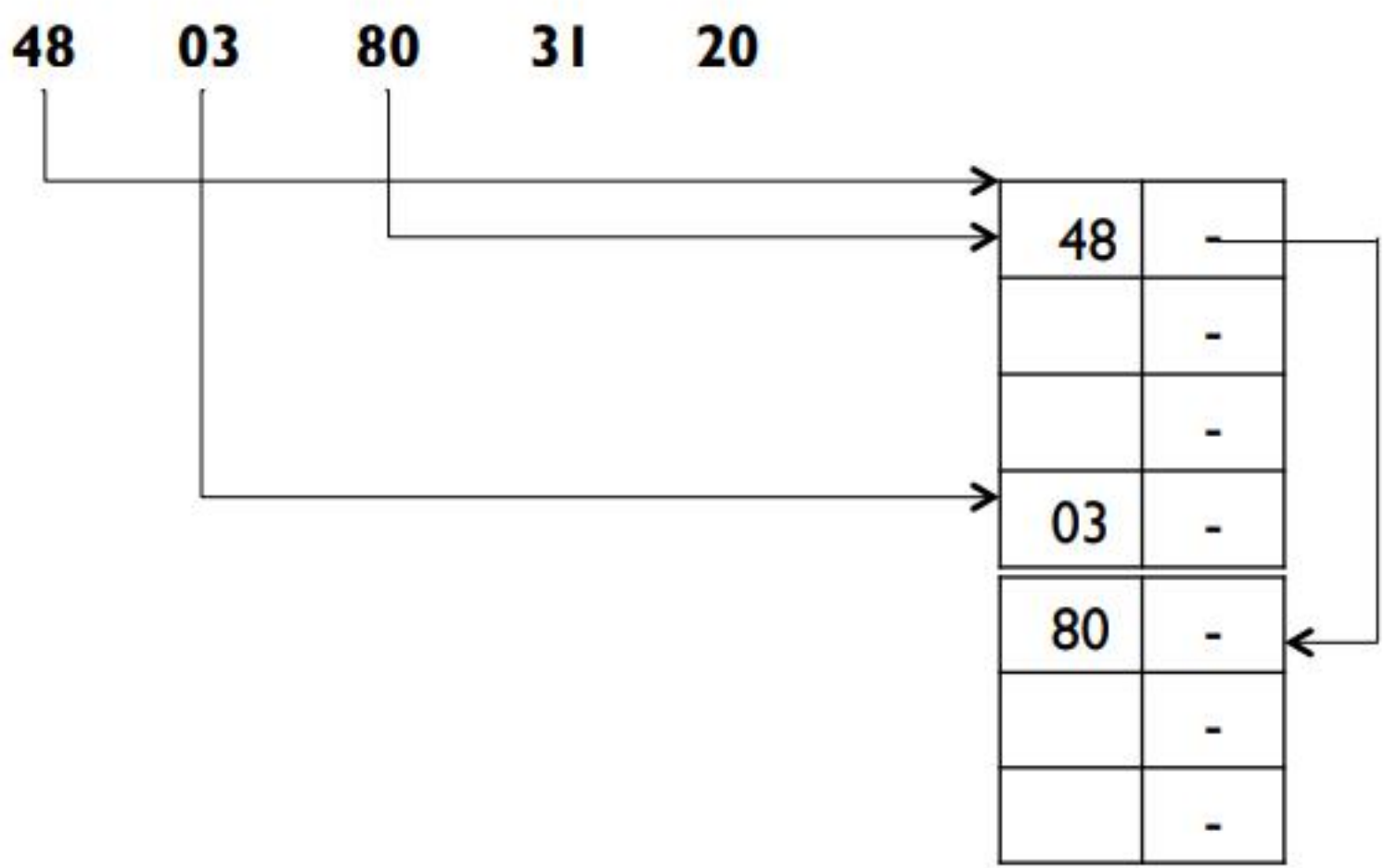
Tratamento de colisão – encadeamento interior



Tratamento de colisão – encadeamento interior

m = 7

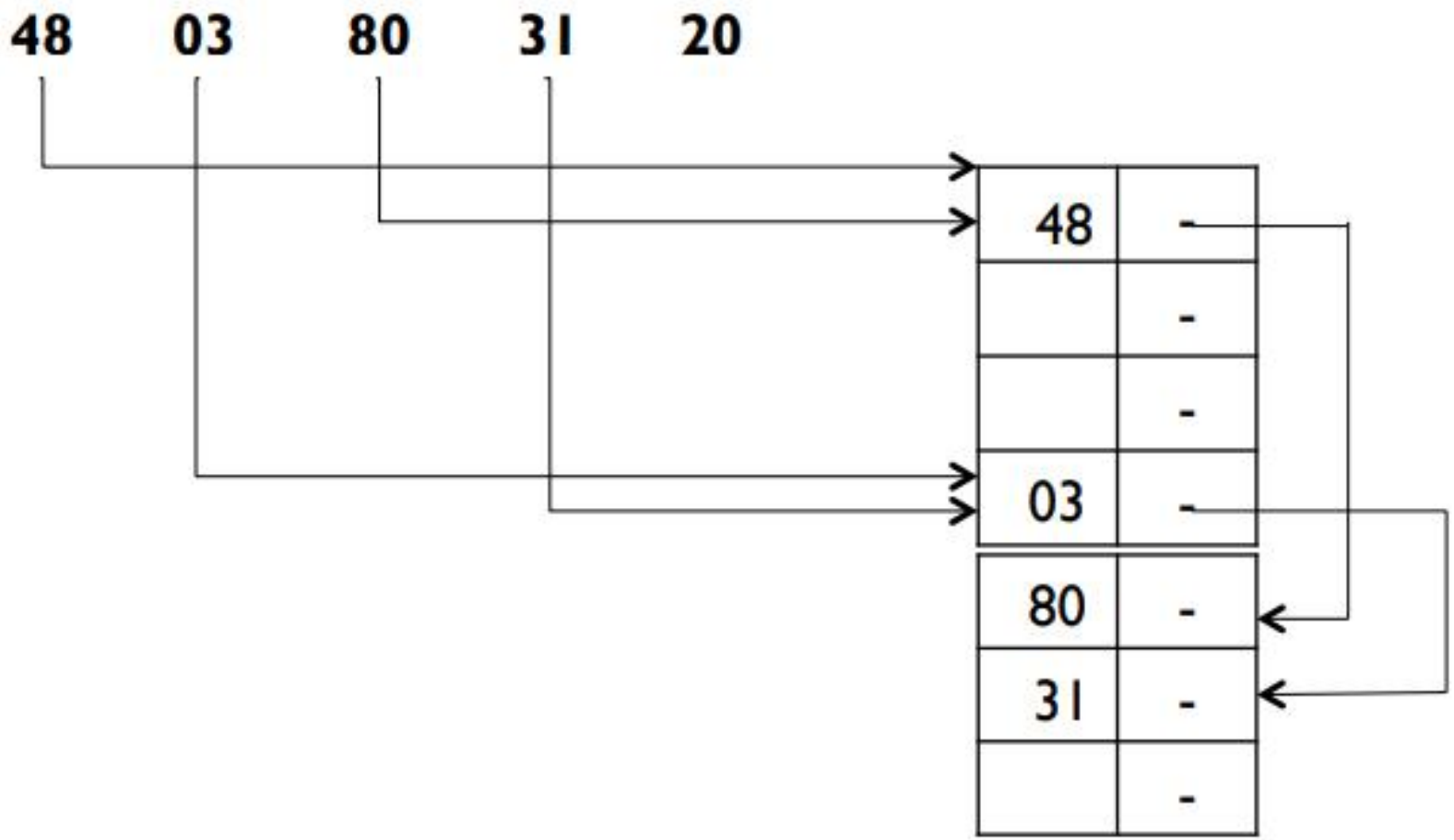
80 mod 4= 0



Tratamento de colisão – encadeamento interior

m = 7

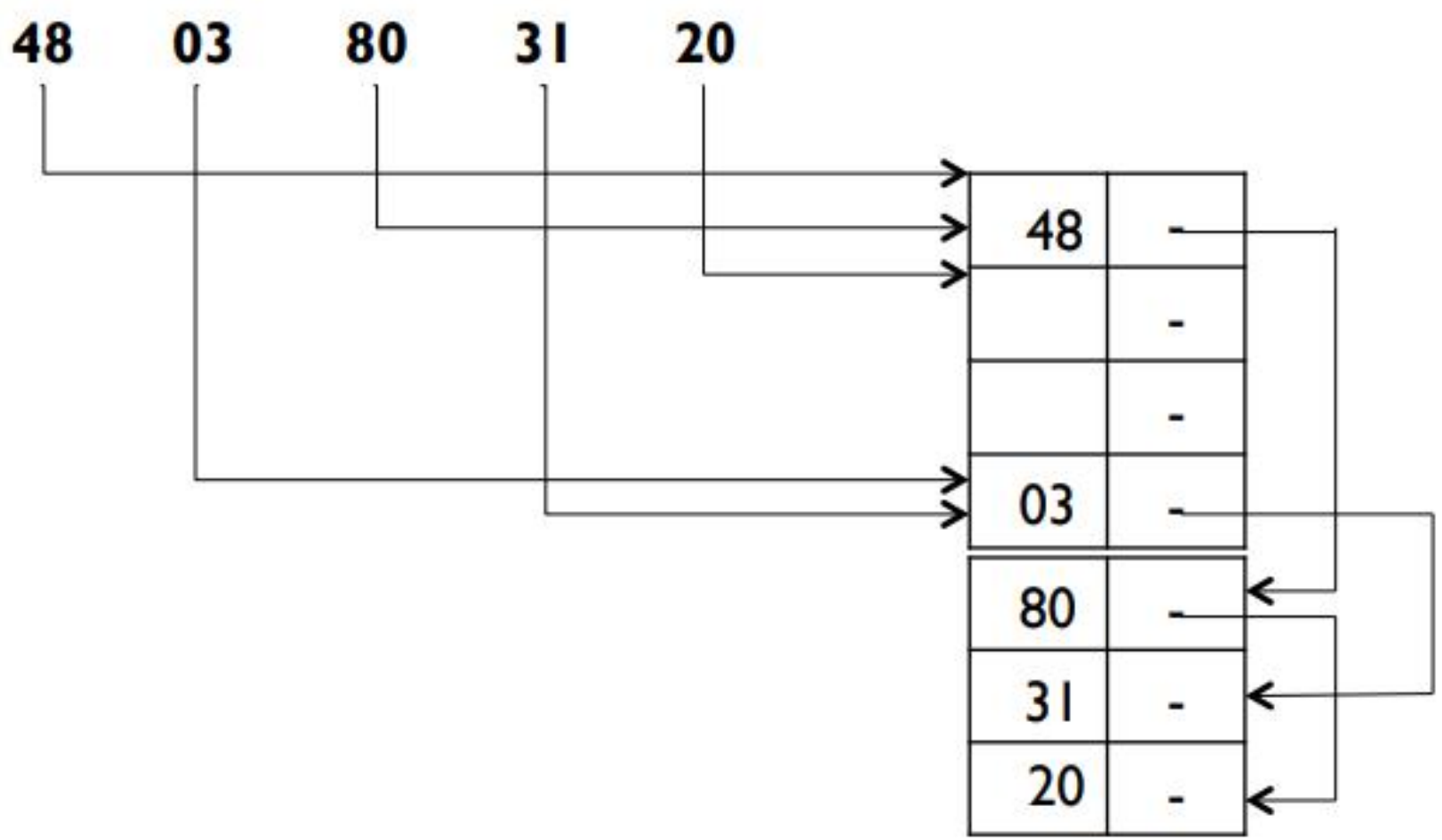
31 mod 4 = 3



Tratamento de colisão – encadeamento interior

m = 7

20 mod 4 = 0



```c
#include <stdio.h>
#include <stdlib.h>

// constante repesenta o tamanho da tabela
#define M 19

// estrutura Pessoa com nome e uma matrícula
typedef struct{
    int matricula;
    char nome[50];
}Pessoa;

// nossa tabela hash do tipo Pessoa
Pessoa tabelaHash[M];

// inicializa nossa tabela com o valor de codigo -1
void inicializarTabela(){
    int i;
    for(i = 0; i < M; i++)
        tabelaHash[i].matricula = -1;
}

// função de espalhamento (resto da divisão da chave por M)
int gerarCodigoHash(int chave){
    return chave % M;
}

```

```
29.  Pessoa lerPessoa(){
30.      Pessoa p;
31.      printf("Digite a matricula: ");
32.      scanf("%d", &p.matricula);
33.      scanf("%*c");
34.      printf("Digite o nome: ");
35.      fgets(p.nome, 50, stdin);
36.      return p;
37.  }
38.
39.  // inserir uma pessoa na tabela
40.  void inserir(){
41.      Pessoa pes = lerPessoa();
42.      int indice = gerarCodigoHash(pes.matricula);
43.      while(tabelaHash[indice].matricula != -1)
44.          indice = gerarCodigoHash(indice + 1);
45.      tabelaHash[indice] = pes;
46.  }
47.
48.  Pessoa* buscar(int chave){
49.      int indice = gerarCodigoHash(chave);
50.      while(tabelaHash[indice].matricula != -1){
51.          if(tabelaHash[indice].matricula == chave)
52.              return &tabelaHash[indice];
53.          else
54.              indice = gerarCodigoHash(indice + 1);
55.      }
56.      return NULL;
57.  }
```

Atividade

1. Implementar uma tabela hash em C.

2 .Implemente nesta tabela pelo menos três tratamentos de colisão.

- https://wagnergaspar.com/tabela-hash/
- https://www.geeksforgeeks.org/hashing-data-structure/

https://youtube.com/playlist?list=PL8iN9FQ7_jt7GZiYfxGIb7sZJOQhw3Xr7&si=V-eMSTEMwbiI9Ggk