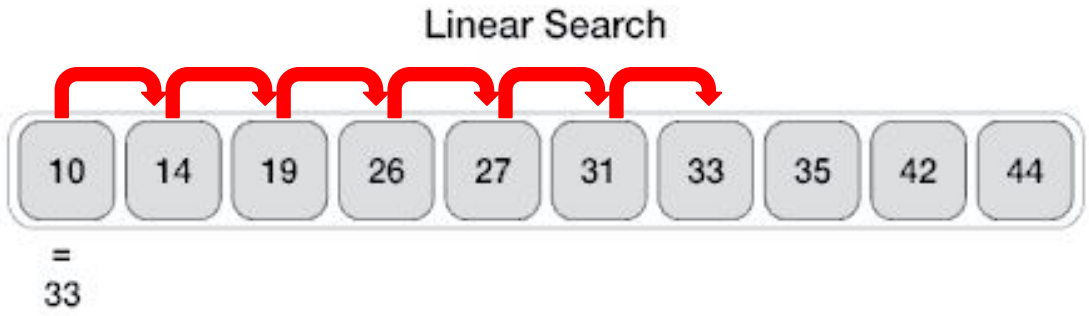
Tabelas *Hash*

Helena Graziottin Ribeiro
hgrib@ucs.br

UCS
UNIVERSIDADE
DE CAXIAS DO SUL

Métodos de pesquisa ou busca

- Os métodos de **pesquisa** vistos até agora (sequencial, binária, por dicionários) buscam informações armazenadas com base na comparação de suas **chaves** (chave pode ser o próprio elemento procurado):



Métodos de pesquisa ou busca

- Chave de um registro (a chave pode ser numérica ou não):

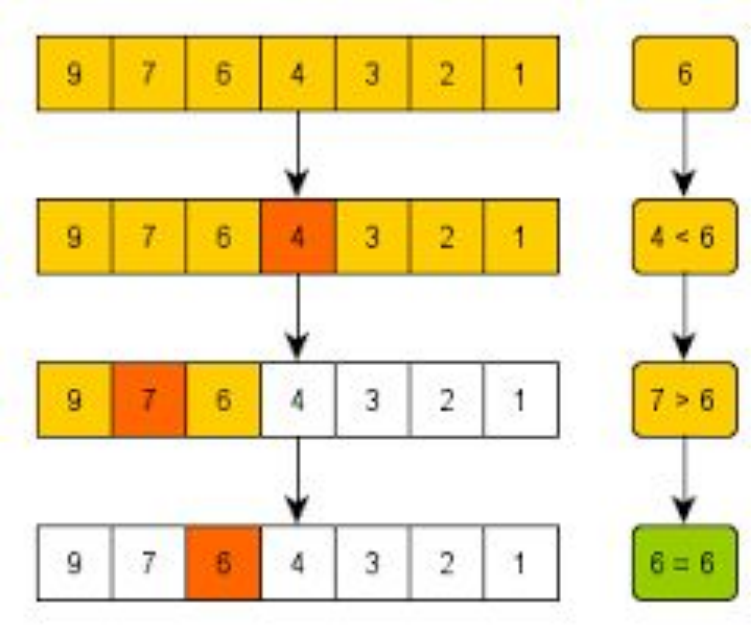
Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Métodos de pesquisa ou busca

- **Chaves** podem ser **nomes** de pessoas, por exemplo
- Solução: todo dado não numérico corresponde uma representação numérica no computador;
  - uma **função** pode converter o dado não numérico em numérico
- Assim, **todas as chaves são consideradas numéricas**

Métodos de pesquisa ou busca

- Para obtermos algoritmos eficientes, armazenamos os elementos ordenados e tiramos proveito dessa ordenação

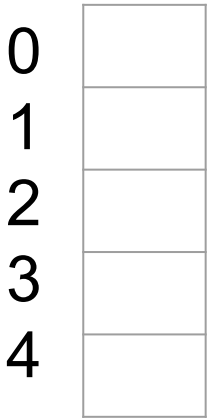


O que é?

- Uma **tabela *Hash***, também conhecida como **tabela de dispersão** ou **tabela de espalhamento**, é uma estrutura de dados especial, que associa **chaves** e valores, através de uma **função hash** (ou função de dispersão):
  - a partir de uma **chave simples**, essa função permite fazer uma busca rápida e obter o valor desejado
  - **forneço a chave**, e a **função hash retorna a posição** (ou endereço) da localização do valor

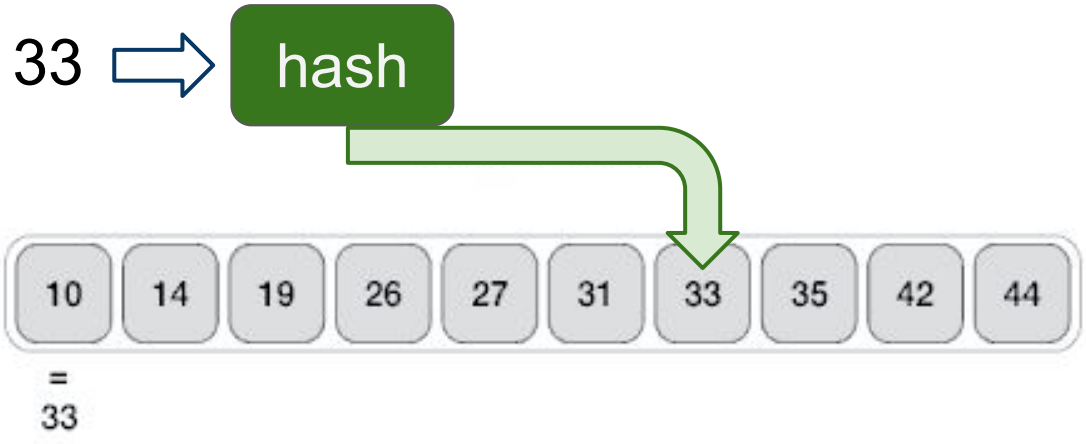
O que é?

- A **tabela *Hash*** é um conjunto de elementos, como um vetor:



O que é?

- **forneço a chave**, e a **função hash** calcula e **retorna a posição** (ou endereço) da localização do valor



O que é? Exemplo

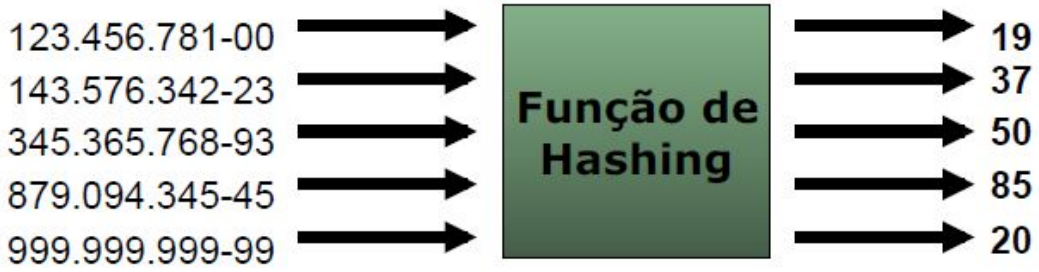
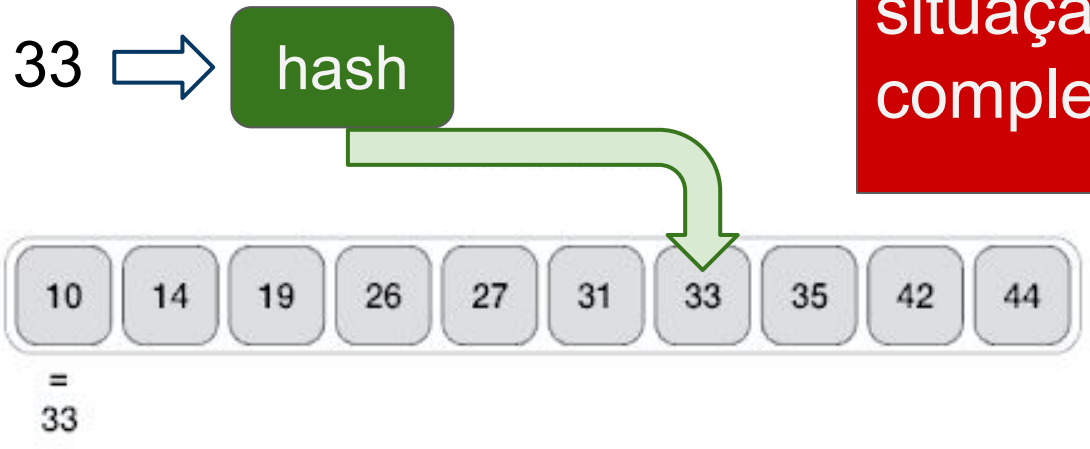


Tabela Hash

19	**123.456.781-00; Fausto Silva; Av. Canal. Nº 45.**
20	
...	
37	**143.576.342-23; Carla Perez; Rua Celso Oliva. Nº 27.**
...	
50	**345.365.768-93; Gugu Liberato; Av. Atlântica. S/N.**
...	
85	**879.094.345-45 ; Hebe Camargo; Rua B. Nº 100.**
...	

O que é?

- **forneço a chave**, e a **função hash** calcula e **retorna a posição** (ou endereço, ou índice) da localização do valor



situação ideal:
complexidade **O(1)**

Problema: definir uma boa função de *hash*

- a **função *hash*** faz um cálculo para definir uma posição possível na tabela *hash*:
  - exemplo de **função *hash*** “clássica” para um vetor:

```
chave % tamanho
```

resto da divisão inteira da **chave** pelo **tamanho do vetor**

Problema: definir uma boa função de *hash*

- a **função *hash*** pode ser qualquer cálculo:
  - o ideal é que a função forneça índices únicos para o conjunto das chaves de entrada possíveis:
    - sem repetir valor gerado
    - fácil de computar
    - uniforme (todos os locais da tabela sejam igualmente utilizados)

Problema: definir uma boa função de *hash*

- a **função *hash*** pode ser qualquer cálculo:
  - o que se quer é que ela distribua **o melhor possível** os valores dentro da tabela

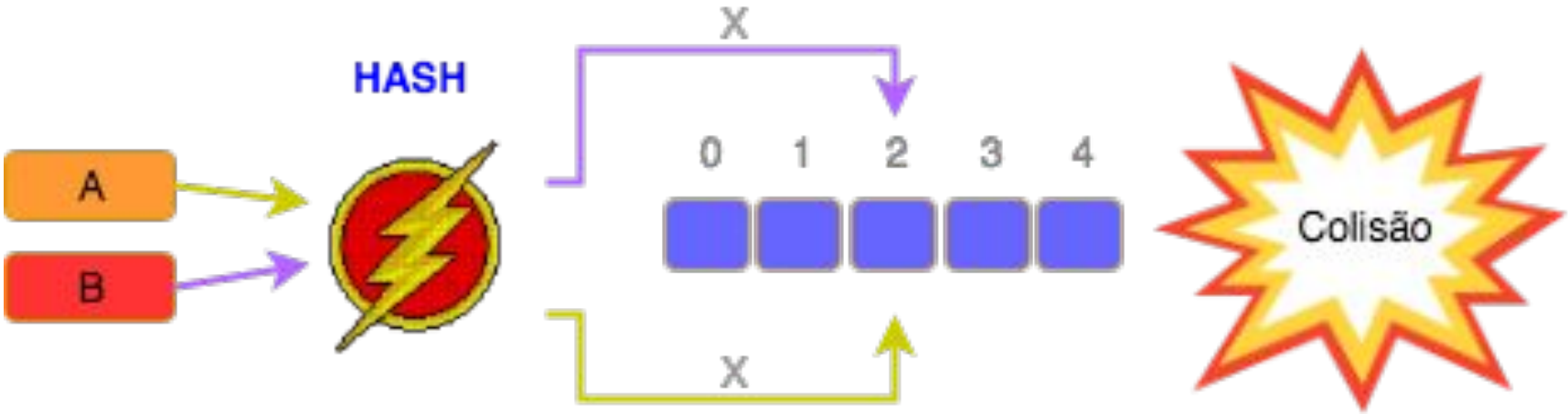
espalhar bem os valores

Problema: definir uma boa função de *hash*

- a **função *hash*** pode ser qualquer cálculo:
  - o que se quer é que ela distribua **o melhor possível** os valores dentro da tabela

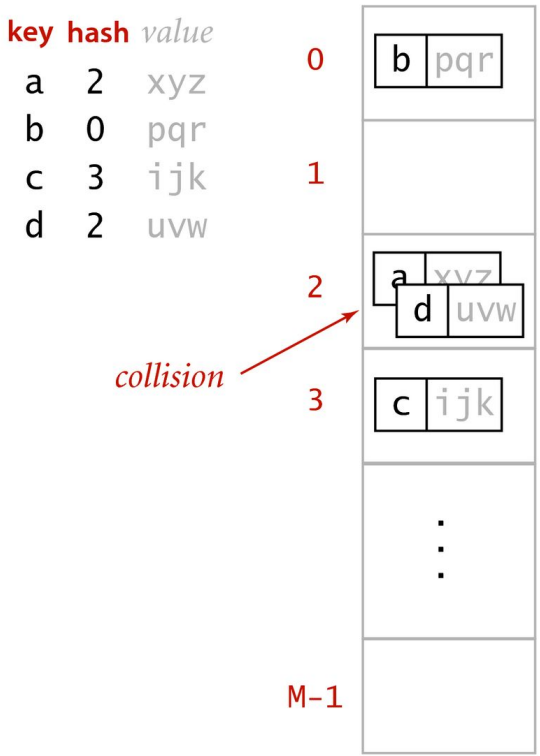
Problema: colisões

- Porém, se não forem, pode haver geração de **duas posições iguais** para **duas chaves diferentes:**



Problema: colisões

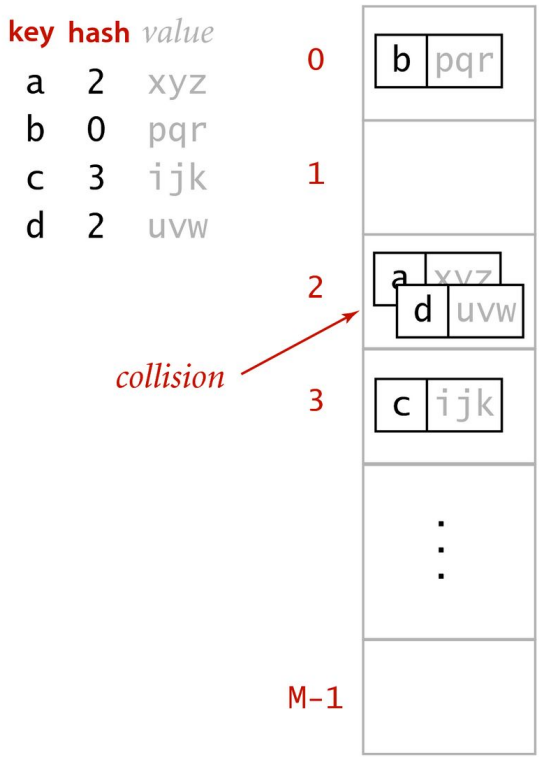
- pode haver geração de **duas (ou mais) posições iguais** para **duas (ou mais) chaves diferentes:**



Hashing: the crux of the problem

Problema: tamanho da tabela?

- é preciso saber o tamanho máximo da tabela: **M**
- ou “estimar” o tamanho máximo da tabela



Hashing: the crux of the problem

Implementações de tabelas hash

- As **diferentes implementações** de tabelas *hash* são **diferentes formas de resolver colisões:**
  - **tabelas hash fechadas:**
    - **linear** (*hashing* para busca linear)
    - **por buckets** (“depósitos”)
  - **tabelas hash abertas:**
    - **hashing aberto** (ou por encadeamento separado)

Implementações de tabelas hash: linear

- função ***hash***: calcula a posição
- se der colisão, usa função de ***rehash*** (ou reespalhamento):
  - procura próxima posição livre

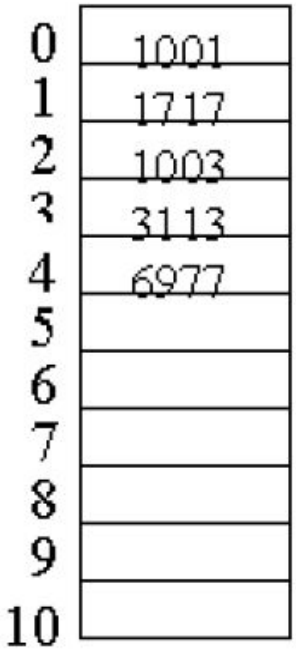
Exemplo:

hash = chave % tamanho

rehash = (hash + 1) % tamanho

Implementações de tabelas hash: linear

Exemplo: tabela ***hash*** com 11 posições



1001: H=**0**

1003: H=**2**

1717: H=**1**

3113: H=0 (colisão); RH=1 (colisão); RH=2 (colisão); RH=**3**

6977: H=3; RH=**4**

1001 % 11 = 0

(1001 / 11 = 91)

Implementações de tabelas hash: linear

O que seria um bom tamanho de tabela (**tam**)?

- Alguns valores de **tam** são melhores que outros
- Se **tam** é um número par, *hash(x)* será par quando x for par e ímpar quando x for ímpar
  - Não é uma boa solução
- Se **tam** for uma potência de 2, *hash(x)* dependerá apenas de alguns dígitos de x.
  - Não é uma boa solução

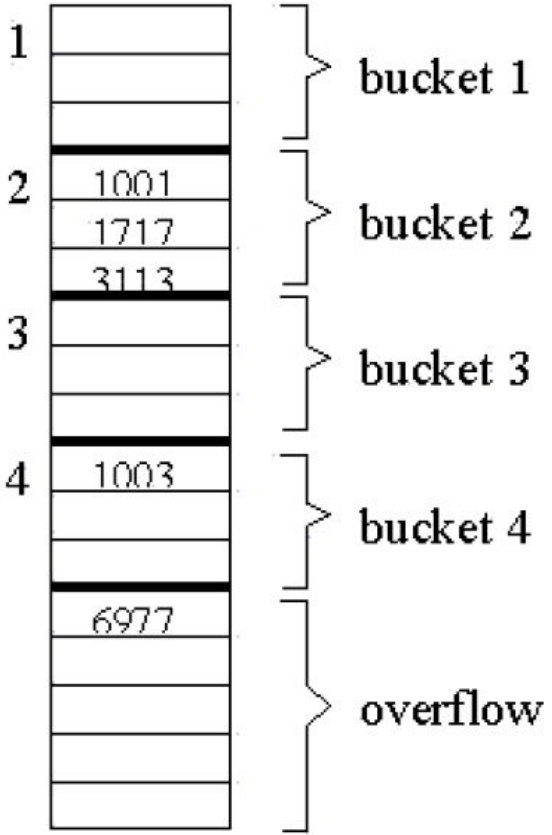
Implementações de tabelas hash: linear

O que seria um bom tamanho de tabela (**tam**)?

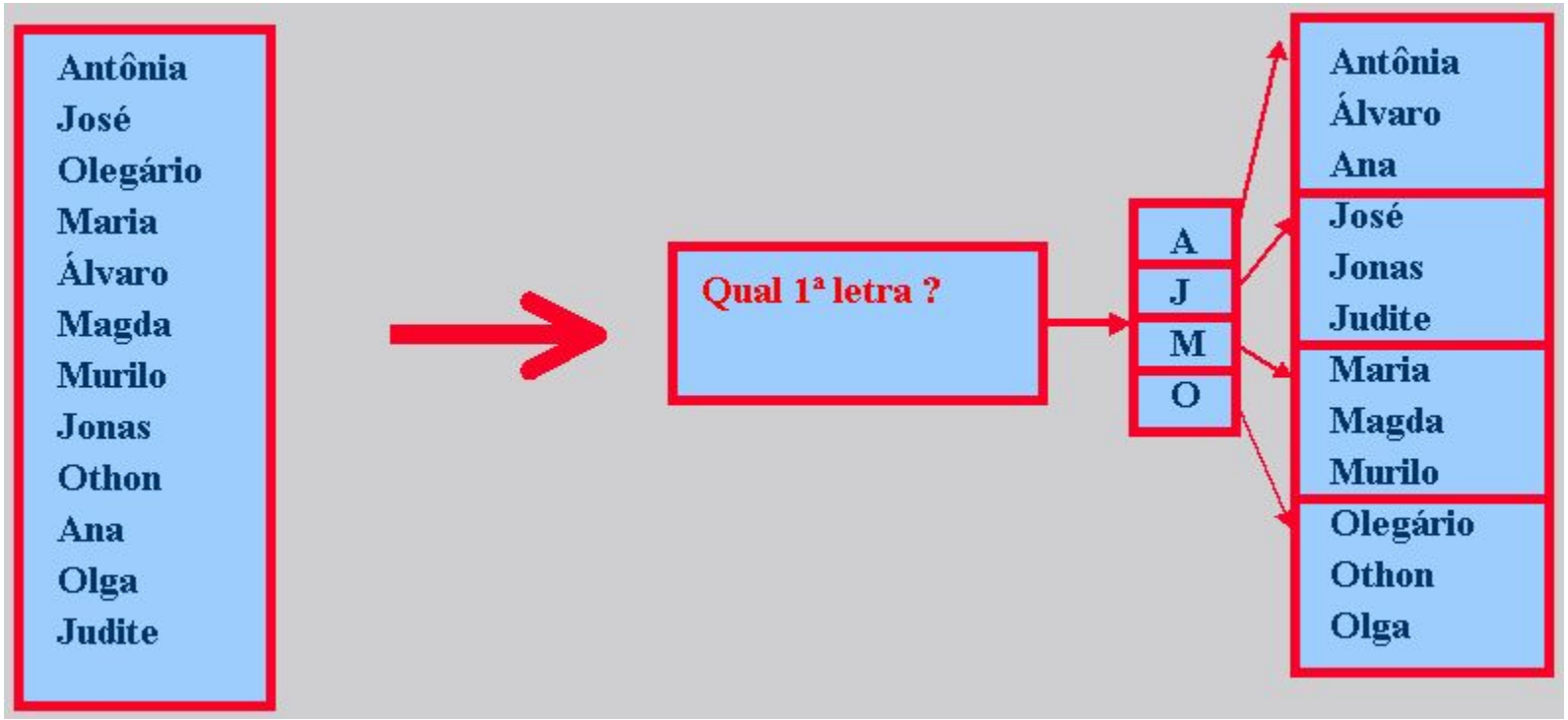
- Existem alguns critérios que têm sido aplicados com bons resultados práticos:
  - Escolher **tam** de modo que seja um número primo não próximo a uma potência de 2
  - Escolher **tam** tal que não possua divisores primos menores que 20
  - Exemplo: **tam = 23**

Implementações de tabelas hash: por *buckets*

- A tabela é dividida em compartimentos (*buckets*)
- a função *hash* retorna o valor de um dos compartimentos
- **Exemplo:**
  - tabela ***hash*** com 4 buckets, de 3 posições cada
  - overflow de 5 posições



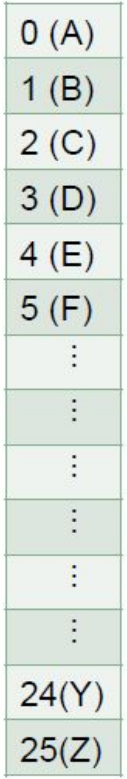
Implementações de tabelas hash: por *buckets*



Implementações de tabelas hash: *hashing aberto*

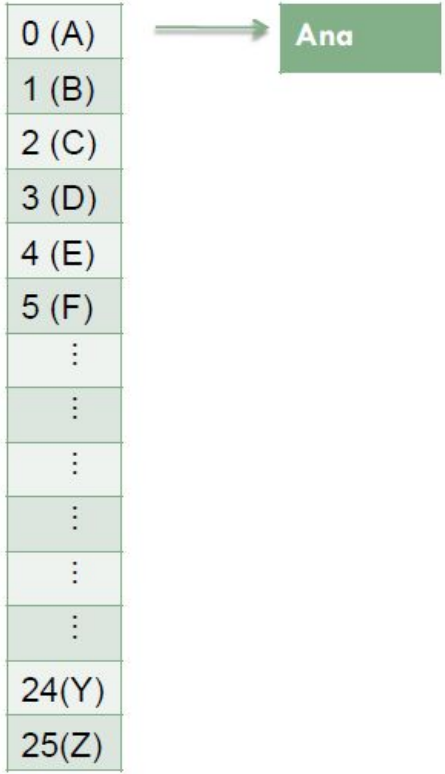
- Colisões: valores com mesmo ***hash*** ficam em lista encadeada associada à mesma posição

Implementações de tabelas hash: *hashing aberto*

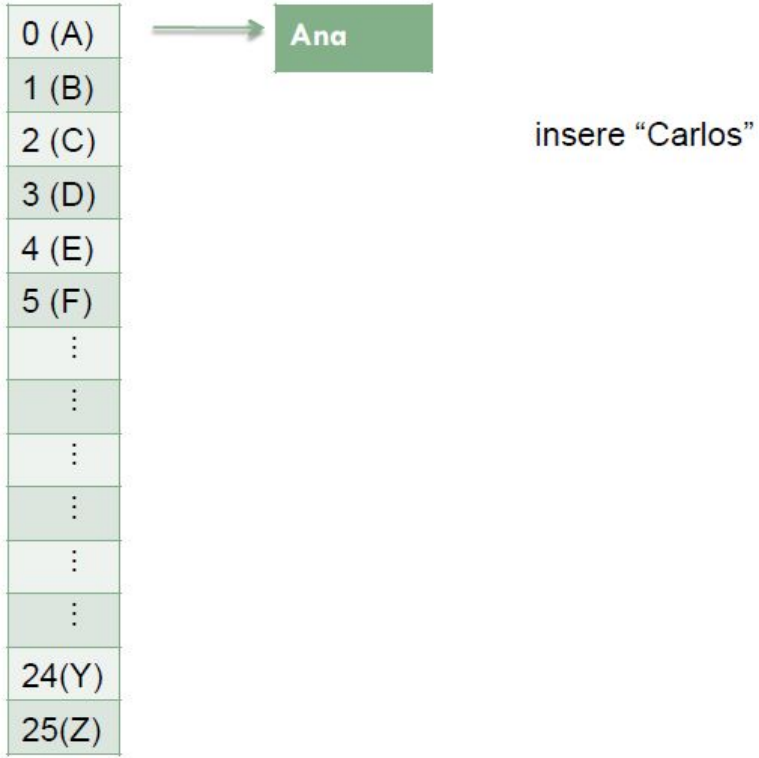


insere “Ana”

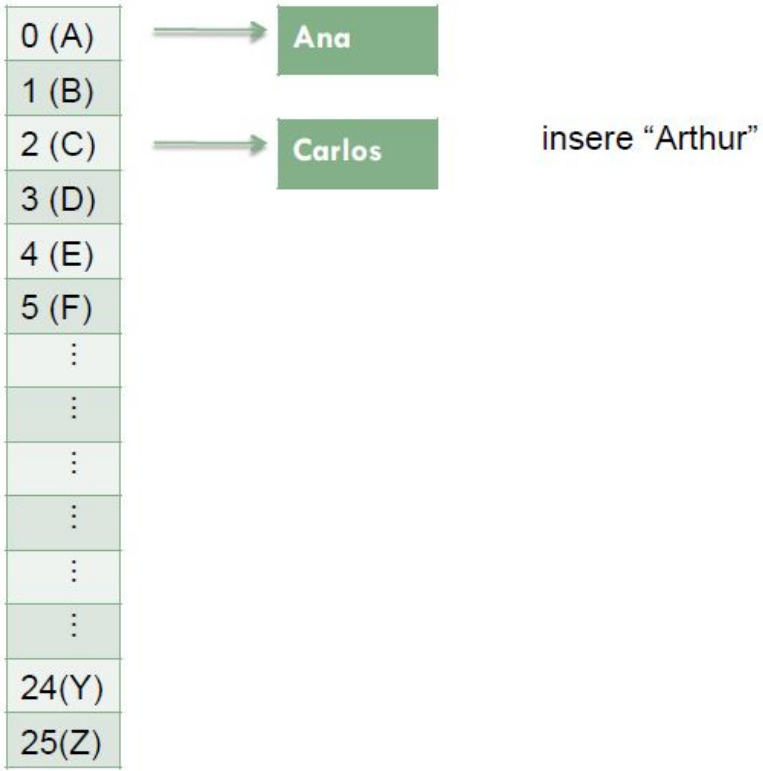
Implementações de tabelas hash: *hashing aberto*



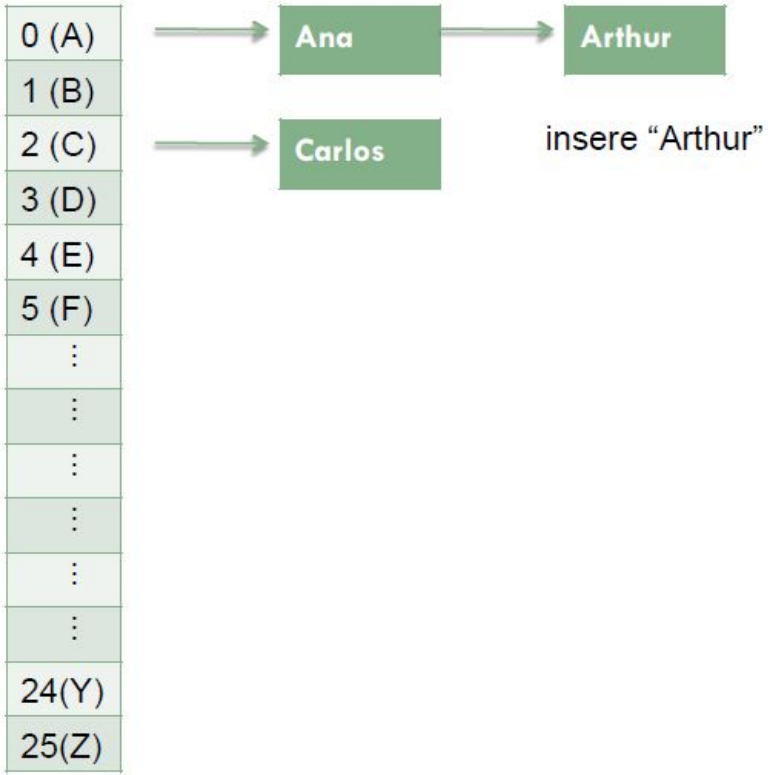
Implementações de tabelas hash: *hashing aberto*



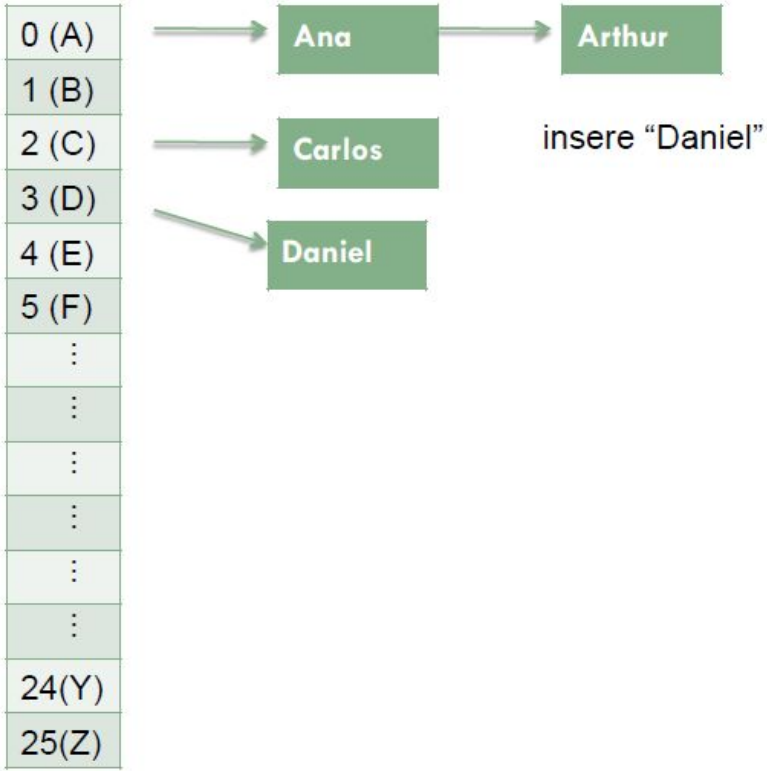
Implementações de tabelas hash: *hashing aberto*



Implementações de tabelas hash: *hashing aberto*



Implementações de tabelas hash: *hashing aberto*



Implementações de tabelas hash: *hashing aberto*