Estrutura de Dados 2

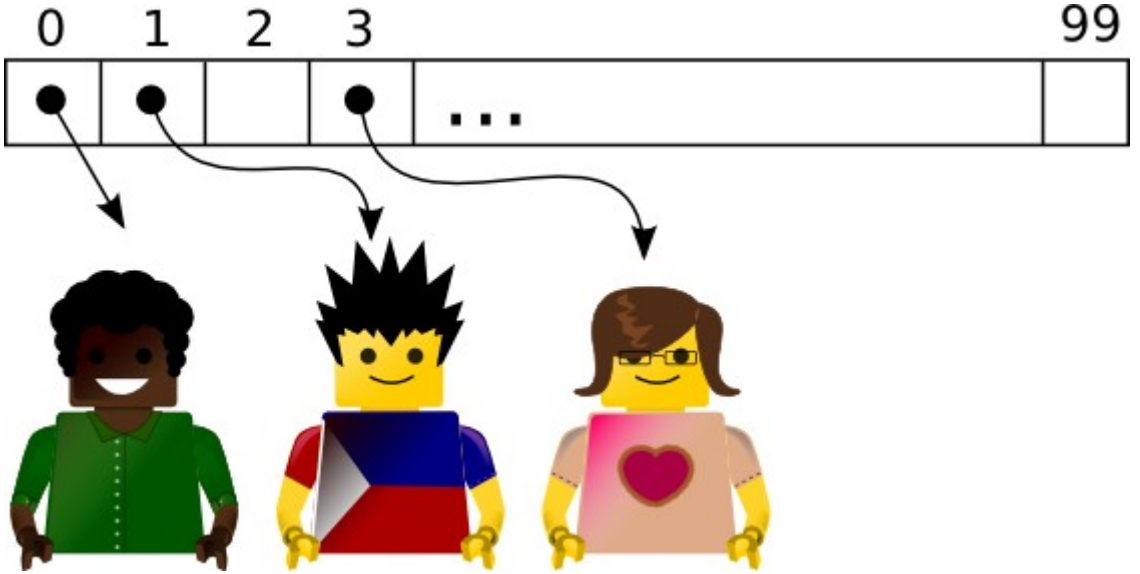
Tabelas de Dispersão (Hash Tables)

Motivação

- Exemplo clássico: escaninho
  - Professores recebem correspondências, comunicados, memorandos, etc. no escaninho abaixo.
  - O que aconteceria se o número de professores fosse (menor, igual, maior) que o número de compartimentos?

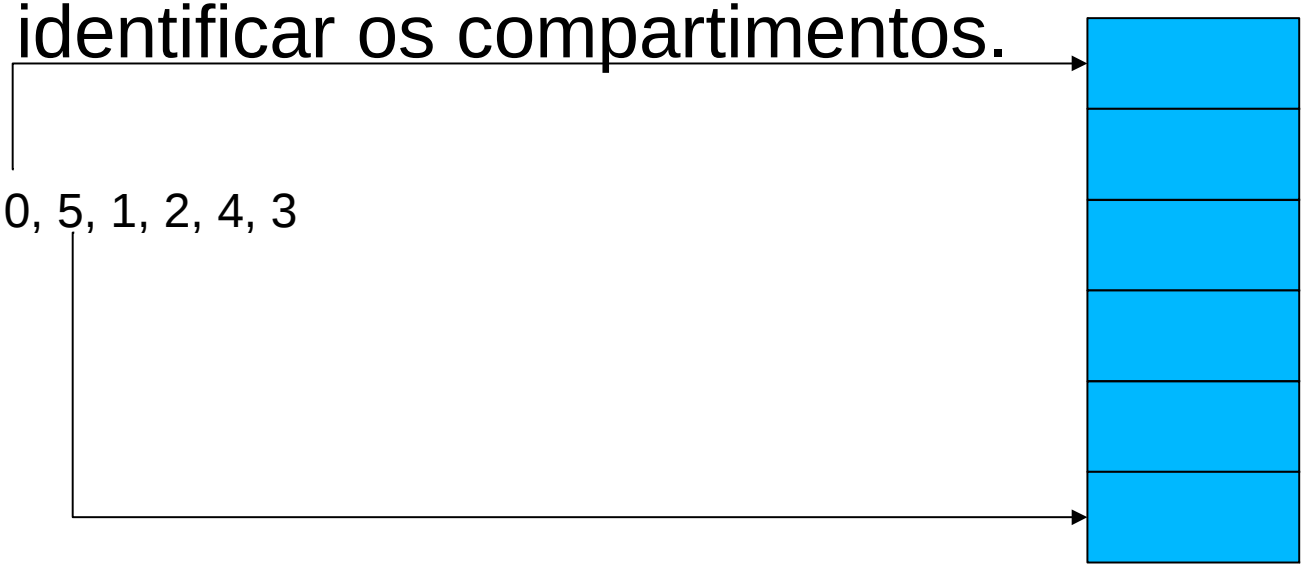
Motivação

- Podemos pensar no escaninho como um array com *m* compartimentos
  - Desta forma, os endereços possíveis são [0, *m* – 1]
  - Tarefa: distribuir correspondências de *n* professores neste array



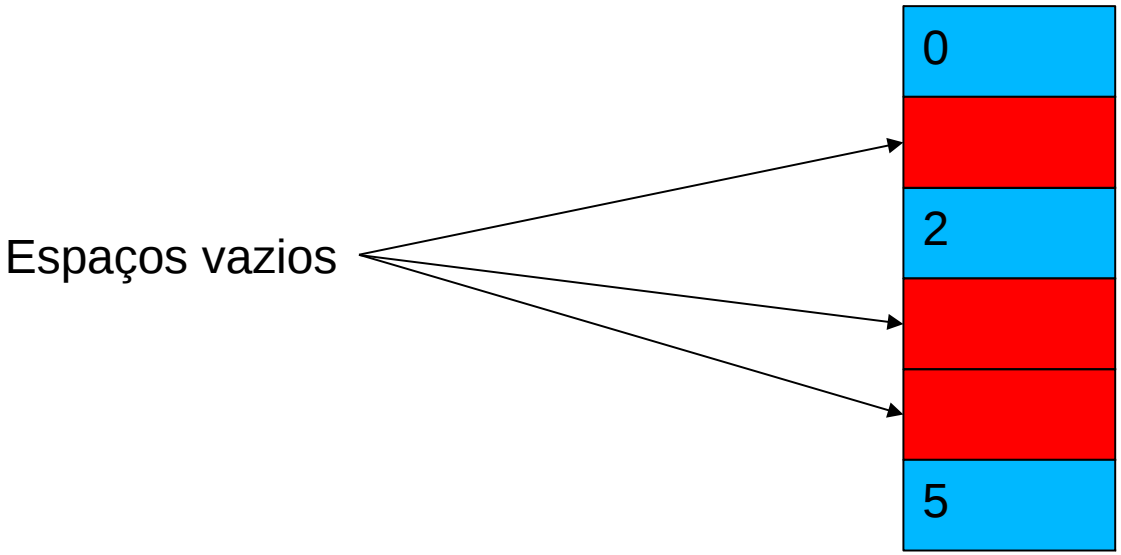
Hashing

- Como definir o tamanho *m* do array?
- Como distribuir eficientemente as *n* chaves pelos *m* compartimentos?
  - Se os valores que as chaves podem assumir variar entre [0, *m* – 1], podemos usar as próprias chaves para identificar os compartimentos.



Hashing

- Se o número de chaves *n* é menor que o número de compartimentos, a tabela pode conter espaços **vazios**.



UFRRJ

Hashing

- Exemplo:
  - Considere m = 1.000
  - Insere-se somente as chaves 0 e 999
  - 998 compartimentos ociosos, desperdiçando precioso espaço em memória
- Pense no desperdício de espaço com um escaninho com 1.000 compartimentos, onde 998 deles ficam vazios a maior parte das vezes.

UFRRJ

Hashing

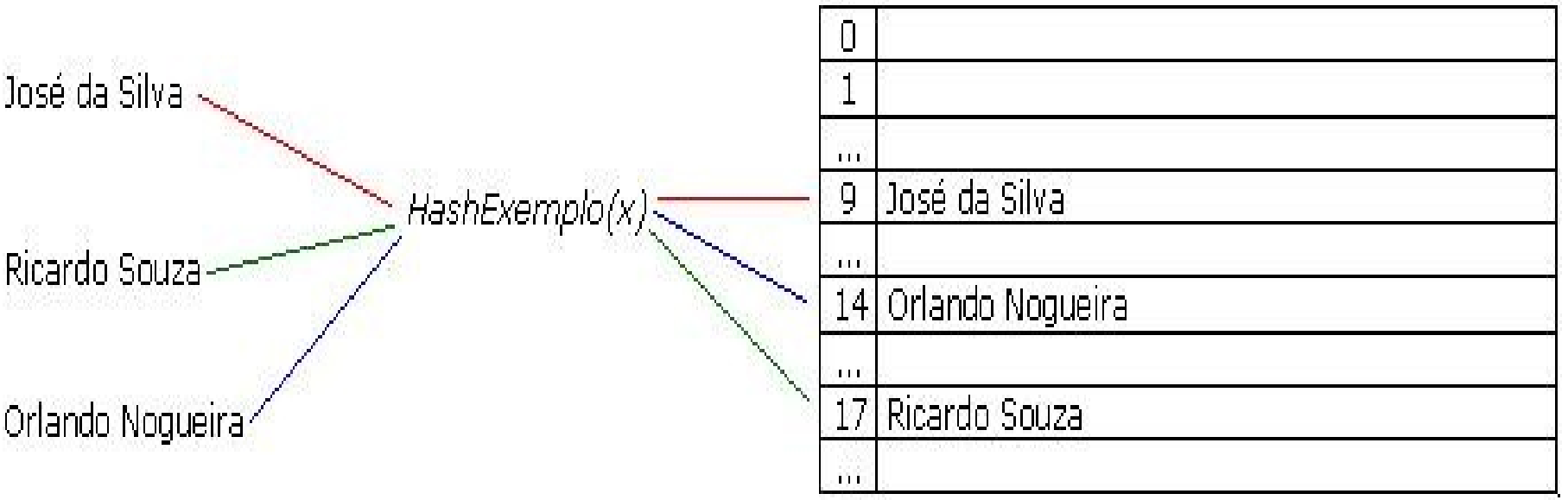
- A solução é criar uma tabela onde *m* é **menor** do que *n.*
- Elementos compartilharão compartimentos (Princípio da Casa dos Pombos)
- Desafio: manter o número de elementos em cada compartimento o mais uniforme possível

UFRRJ

Hashing

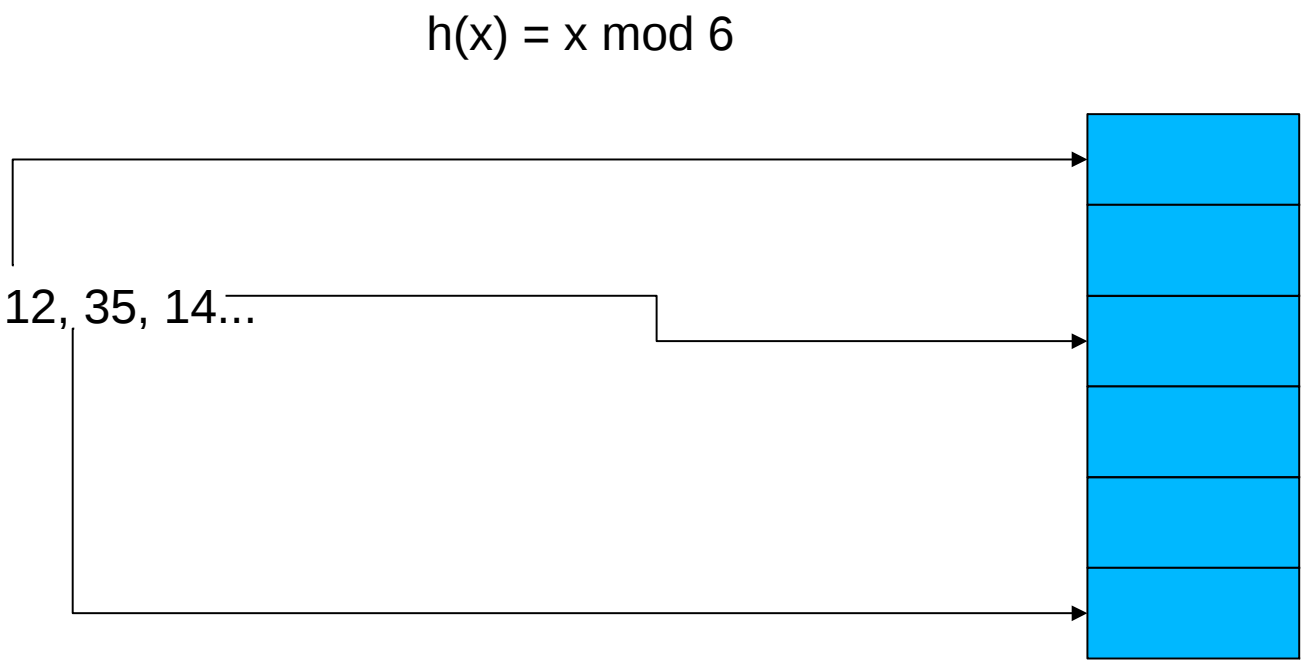
- Mas como mapear, neste caso, um elemento ao seu respectivo compartimento?
- Resposta: usando uma *função de dispersão* (hash function) **h(x)**.
- O objetivo de **h(x)** é distribuir os valores das chaves pelos compartimentos existentes
  - Diz-se que **h(x)** gera *endereço-base* para **x**
  - Array[x] □□ Array[**h(x)**]
- A mesma função **h(x)** será utilizada para buscar registros

Função de Dispersão



Função de Dispersão

- Exemplo de função de dispersão: **mod**
  - Fazer chave **mod** m



Função de Dispersão

- Uma função de dispersão não garante a injetividade
  - Ou seja, é possível **h(x) = h(y)** mesmo se **x ≠ y**
  - Exemplo:
    - h(x) = x mod 6
    - h(6) = h(0) = h(72) = ...
- Quando tenta-se inserir um registro em um compartimento já ocupado, ocorreu uma colisão
  - Diz-se que *x* e *y* são sinônimos em relação a **h**

Função de Dispersão

- É desejável que as funções de dispersão:
  - Criem um número baixo de colisões
  - Sejam facilmente computáveis
  - Sejam uniformes

Depende da distribuição dos valores das chaves

h(x) deve garantir que todos os compartimentos possuem a mesma probabilidade de serem escolhidos

Fácil de garantir caso a tabela seja utilizada para armazenar ponteiros para registros em disco

Função de Dispersão

- Métodos comuns para criação de funções de dispersão:
  - Método da Divisão
  - Método da Dobra
  - Método da Multiplicação
  - ...

Método da Divisão

- Utiliza a função *mod* (resto de uma divisão)
  - `h(x) = x` **`mod`** *`m`*
    - *m* é o tamanho da tabela

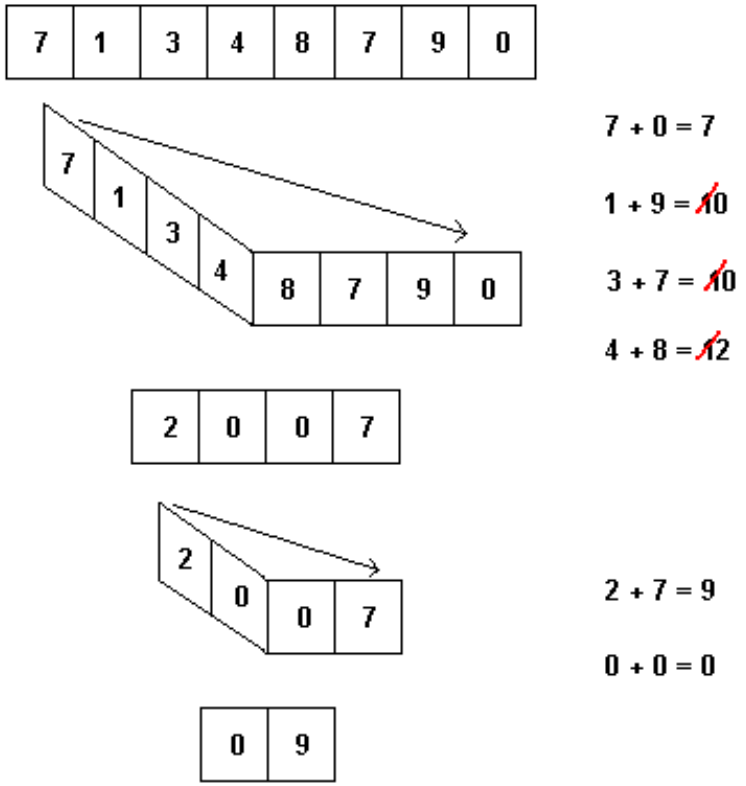
- Característica indesejável: dependência entre x e h(x) caso *m* seja par
  - x par h(x) par
  - x ímpar h(x) ímpar

Método da Divisão

- Estudos mostram bons valores de *m:*
  - Número primo distante de uma potência de 2
  - Não possua divisores primos menores que 20

Método da Dobra

- Este método consiste em “dobrar” a chave ao meio sucessivamente, somando os dígitos sobrepostos sem o vai-um
- Exemplo:



Método da Multiplicação

- Multiplica-se a chave por ela mesma
- Armazena-se o resultado em uma palavra de **b** bits
- Descarta-se os bits das extremidades direita e esquerda, sucessivamente, até que o resultado tenha o tamanho de endereço desejado

Método da Multiplicação

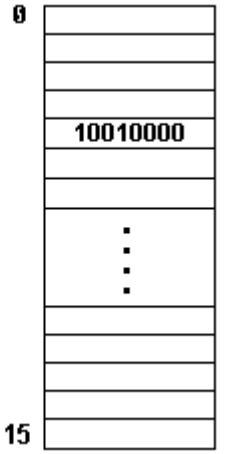
1100

1100 * 1100 = 10010000

1 0 0 1 0 0 0 0

4 bits

0100b = 4d



UFRRJ

Exercício

1. Implemente uma tabela de dispersão (m = 13) em Java onde seja possível escolher a função de dispersão a ser utilizada (utilizando os métodos vistos)
   - Permitir que o usuário digite as chaves a serem armazenadas na tabela
   - Para cada chave armazenada, exibir a posição dela na tabela (resultado da função de dispersão)
   - Caso ocorra uma colisão, exibir uma mensagem de erro.