UNIVERSIDADE DO SUL DE SANTA CATARINA

Aqui o seu futuro acontece

Estrutura de Dados

Aula 12 – Hashing

Prof. Flávio Ceci, Dr.

flavio.ceci@unisul.br

Introdução

- O método de cálculo de endereço ("**Hashing**") não é apenas um método de **pesquisa**, mas também um método de **organização** (**classificação**).

- Os endereços gerados são colocados em uma tabela chamada hashing, dispersão ou espalhamento.

Introdução

- Mas por que Hash se temos Listas e Arvores como estrutura de dados?
  - As Listas são simples de se implementar, mas, com um tempo médio de acesso;
  - As Árvores são estruturas mais complexas, mas que possuem um tempo médio de acesso melhor que os das listas.
    - A organização de uma árvore depende da ordem de entrada dos dados.

Visão Geral

- **Hash**: é uma estrutura de dados de simples entendimento, de implementação razoável e intuitiva de se organizar grandes quantidades de dados.

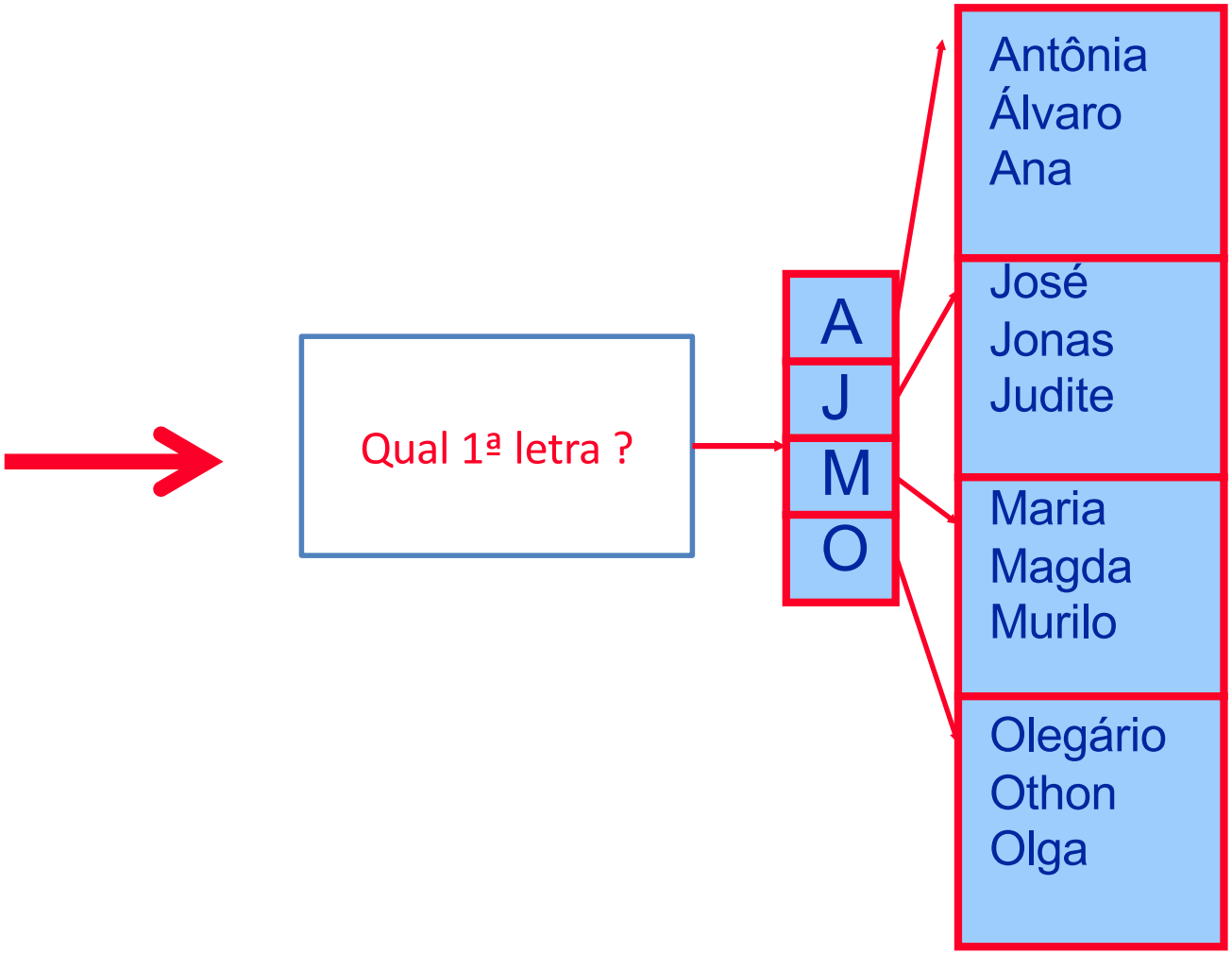
  - Trabalha com a abstração chave/valor que possibilita o acesso direto a partir de um termo ou valor numero, utilizado como chave.

Visão Geral

- O Hash tem como função principal a divisão de um universo de dados a ser organizado em subconjuntos mais gerenciáveis:
  - Permitindo um acesso mais rápido;
  - Organizando melhor os dados.

Exemplo

Antônia
José
Olegário
Maria
Álvaro
Magda
Murilo
Jonas
Othon
Ana
Olga
Judite



Mas será que utilizar a 1ª letra é uma boa solução?

Antônia
José
Olegário
Maria
Álvaro
Magda
Murilo
Jonas
Othon
Ana
Olga
Judite

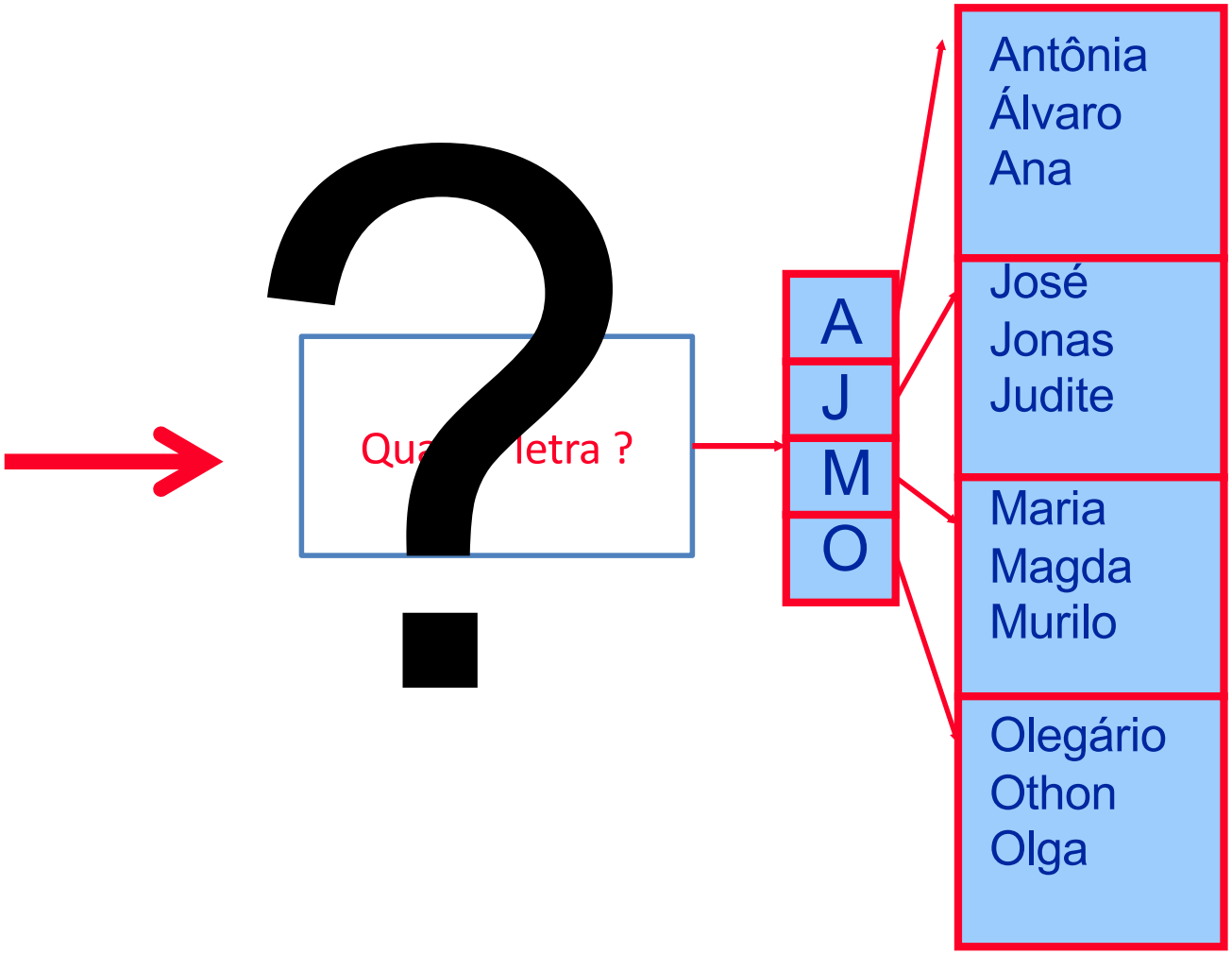


Tabela e Função Hash

- **Tabela Hash**: Estrutura que permite o acesso aos subconjuntos.
- **Função Hash**: Heurística responsável por distribuir e recuperar os valores pelas chaves.

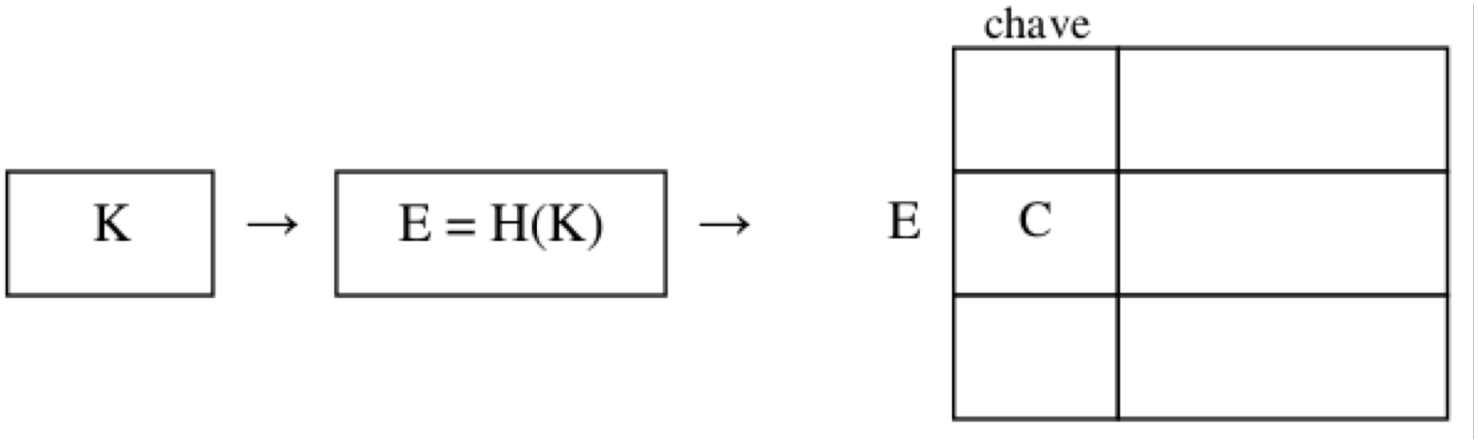


Tabela e Função Hash

- Observe que o hashing de duas chaves K1 e K2 podem levar ao mesmo endereço E.
- Nesse momento, está caracterizada uma **colisão**.
- A função de hashing ideal é aquela que não permite que ocorram colisões.
- Se não for possível identificar uma função hashing ideal, as colisões devem ser tratadas.

Função Hash

- Tem o objetivo de transformar o valor da chave de um elemento de dados em uma posição para este elemento na tabela hash.

- Chamada de função de dispersão ou espalhamento.

Função Hash

- Uma função de hashing ideal deve satisfazer às seguintes condições:

  - Produzir um número baixo de colisões - Depende de se conhecer algo sobre a distribuição das chaves sendo acessadas.
  - Ser facilmente computável - Tipicamente, funções contendo poucas operações aritméticas;
  - Ser uniforme - Idealmente, o número máximo de chaves que são mapeadas num mesmo índice deve ser |N|/|M|.

Função Hash – Método da Divisão

- A função hashing mais difundida é o método da divisão, a qual é definida como sendo:

$$H(K) = K \textbf{ mod } N$$

  - onde K é a chave a ser transformada, N é o número de possíveis elementos da tabela e **mod** calcula o resto da divisão inteira de **K** por **N**.

- A função **mod** pode ser substituída pela divisão por inteiro para conseguir o resto.

Função Hash – Método da Divisão

- Tomemos, á título de ilustração, uma tabela que admita um máximo de 53 entradas e que tenham o campo chave apresentando valores no intervalo de [0..1000].
- Tomemos também a título de exemplo a seguinte função de cálculo de endereço:

H(K) = (K mod 53)

Função Hash – Método da Divisão

- Portanto, a função H recebe um argumento que é um valor entre 0 e 1000, e calcula um endereço entre 0 e 52. A seguir, consideramos os seguintes valores de chave e os endereços correspondentes:

Chave	383	487	235	527	510	320	563	500	646	103	63
Endereço	11	9	22	49	32	1	32	22	9	49	9

Função Hash – Método da Divisão

Chave	383	487	235	527	510	320	563	500	646	103	63
Endereço	11	9	22	49	32	1	32	22	9	49	9

- Neste exemplo, notamos que o endereço 32 é gerando para as chaves 510 e 563, enquanto o endereço 9 é gerando para as chaves 487, 646 e 63:

Tratamento de Colisões

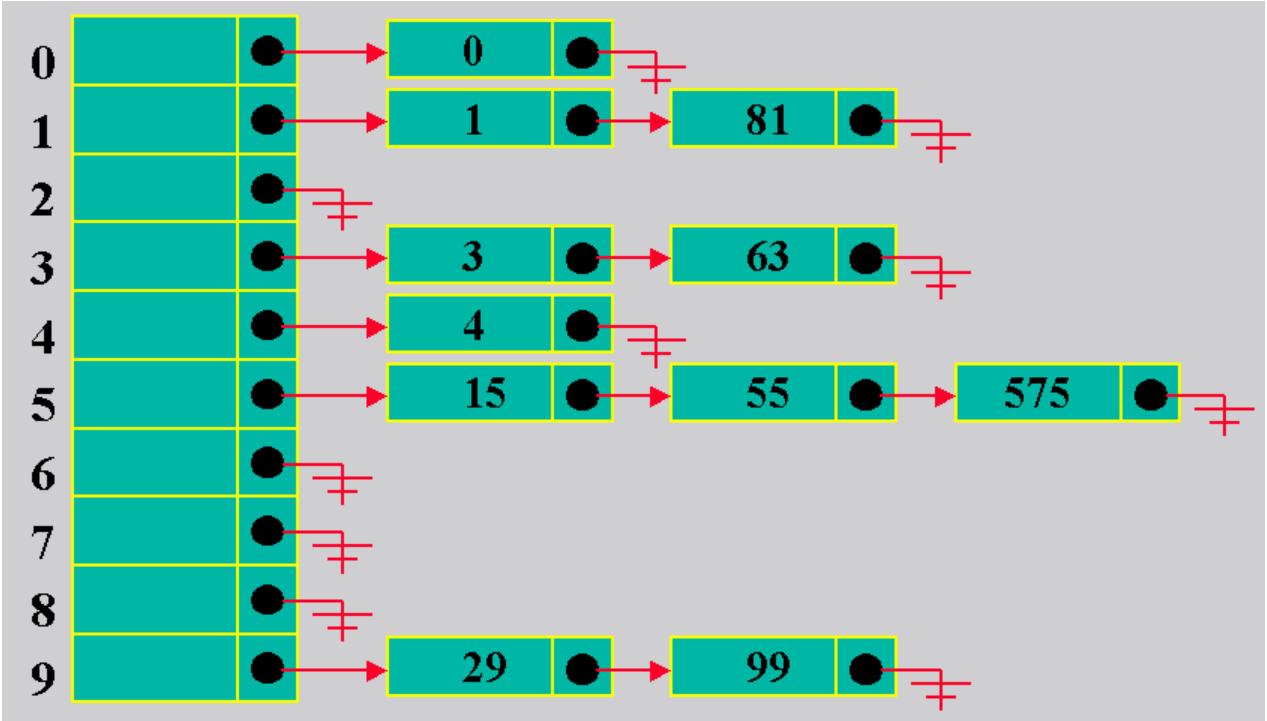
Tratamento de Colisões

- Já foi observado que o mesmo endereço base pode ser encontrado para chaves diferentes, como resultado da função de hash, o que é chamado uma colisão.

- Define-se como fator de carga de uma tabela de hashing o valor a = N/M, onde N é o número de chaves armazenadas e M o tamanho da tabela.

Tratamento de Colisões

- Os mecanismos mais comuns para tratamento de colisões são: endereçamento aberto e **encadeamento**.



Fonte: http://www.inf.ufsc.br/~ine5384-hp/Hashing/

Exercício

Exercício

- Desenvolva uma aplicação em Java que implemente um HASH que suporte 25 chaves distintas;

- Utilize a seguinte **Função Hash**:

K(valor) = valor % 25;

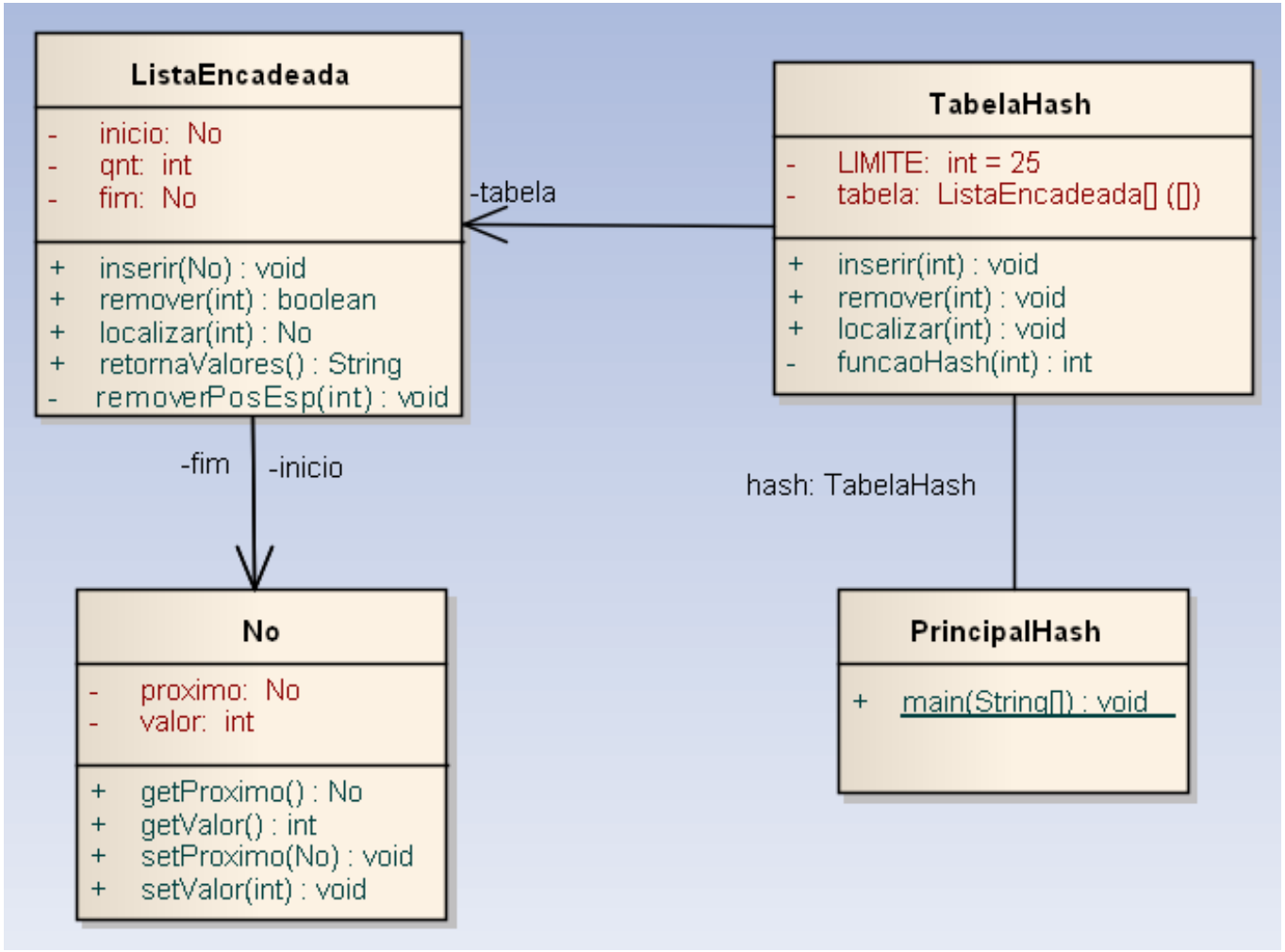
Exercício

- **Requisitos:**
  - O Hash deve permitir o cadastro de valores repedidos;
  - Essa implementação deve ser baseada em uma tabela hash aberta;
  - O suporte a colisões deve ser implementado utilizando como base uma **lista encadeada**.

Exercício

- Sobre a lista encadeada:
  - A mesma deve possuir descritor já que as inserções acontecerão no **final da lista**;
  - Deve-se construir um método para **remover** um elemento utilizando como **base no seu valor**;
  - Deve-se construir um método para verificar se o valor existe na HASH;

Exercício



Obs. A solução desenvolvida deve seguir esse desenho de solução.

Exemplo da implementação

Valores para teste: 72, 2, 29, 25, 82, 100, 85, 42, 39, 45