Tabela de dispersão

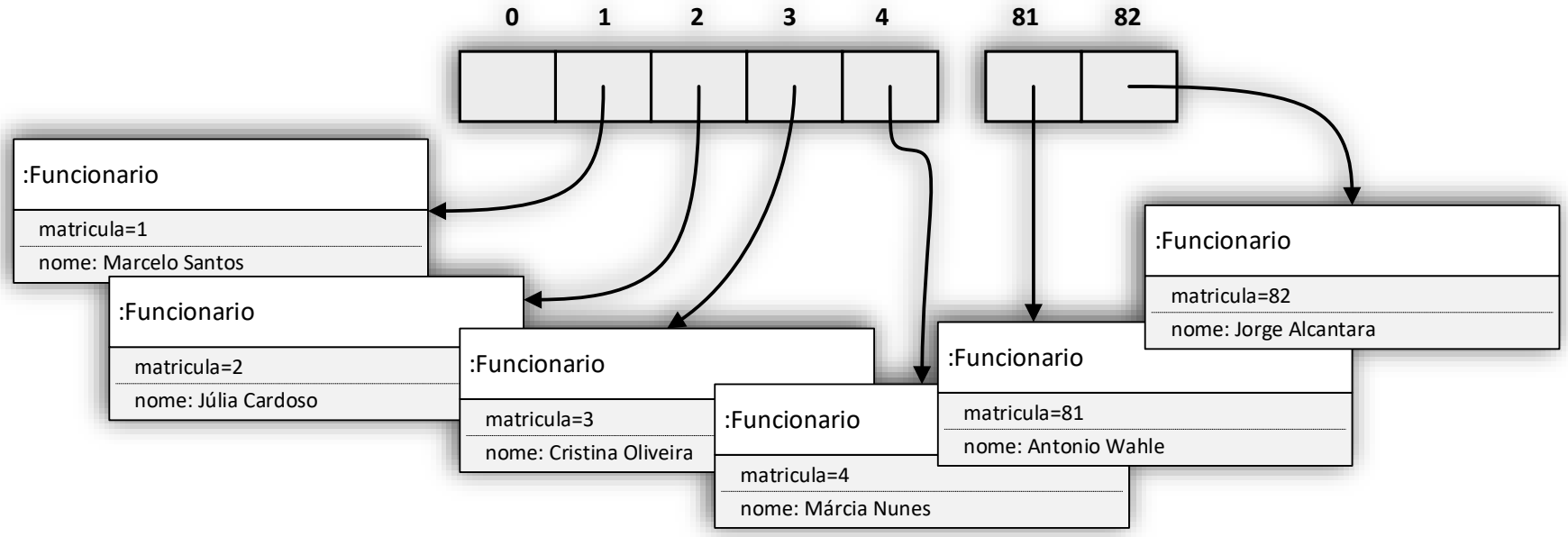
Definições

- Tabelas de dispersão oferecem inserção, busca e remoção muito rápidas:
  - Podem executar em *tempo constante*

- Também conhecidas como:
  - Mapas de dispersão
  - Tabelas de espalhamento
  - Tabela *hash* ou
  - Mapa *hash*
  - *Hash* map
  - *Hash* table

Busca em vetores

Considerar que os funcionários de uma empresa foram armazenados num vetor:



Para localizar o funcionário com número de matrícula *n* é preciso fazer uma pesquisa linear?

Usando vetores

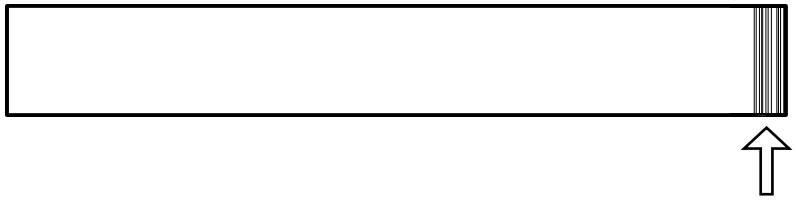
- Uso de vetores é atraente, porém:
  - Nem sempre as chaves de acesso são “bem organizadas”
  - Se houver exclusões de funcionários existirão intervalos perdidos

Uma tabela de dispersão simples

Sequencia	Matrícula	Nome
1	176489	Adriana Freitas
2	176203	Beatriz Souza
3	176770	Bruno Silva
4	181539	Douglas de Oliveira
5	181656	Eduardo Junior
6	181947	Fábio Costa
7	181766	Gabriel Xavier
8	185475	Gabriela Pereira
9	179264	Jonatan Becker
10	175416	Lucas Bocker
11	181616	Matheus Goias
12	182455	Miguel Radunz
13	176081	Paulo Wegner
14	185993	Renato Weber
15	176457	Ricardo Rocha
16	176278	Rodrigo Duarte

maior valor

Com um vetor de [0..185993] teríamos:



dados concentrados aqui

Taxa de ocupação: $\frac{16}{185.994}$ = 0,008 %

Uma tabela de dispersão simples

Sequencia	Matrícula	Nome	Índice
1	176489	Adriana Freitas	1073
2	176203	Beatriz Souza	787
3	176770	Bruno Silva	1354
4	181539	Douglas de Oliveira	6123
5	181656	Eduardo Junior	6240
6	181947	Fábio Costa	6531
7	181766	Gabriel Xavier	6350
8	185475	Gabriela Pereira	10059
9	179264	Jonatan Becker	3848
10	**175416**	Lucas Bocker	**0**
11	181616	Matheus Goias	6200
12	182455	Miguel Radunz	7039
13	176081	Paulo Wegner	665
14	185993	Renato Weber	**10577**
15	176457	Ricardo Rocha	1041
16	176278	Rodrigo Duarte	862

⇧

Matricula - 175416

Tamanho do vetor: 10578 elementos

Taxa de ocupação: 0,15%

Função de dispersão

- “Compactamos” a faixa de valores para utilizar uma faixa menor.

- *Função de dispersão* ou *função de hash* é uma função matemática que converte um número compreendido numa faixa de valores grande para uma faixa de valores menor
  - O resultado desta função é “valor de hash” e corresponde ao índice do vetor

Compactação de valores

- Considerar que ao invés de um vetor de 10 mil elementos, tenhamos apenas 16 elementos.
- Podemos “compactar” com a função:

Matricula % 16

Mátricula	Índice	Aluno
176489	9	Adr...
176203	11	Bea...
176770	2	Bru...
181539	3	Dou...
181656	8	Edu...
181947	11	Fáb...
181766	6	Gab...
185475	3	Gab...
179264	0	Jon...
176081	8	Luc...
181616	0	Mat...
182455	7	Mig...
175416	1	Pau...
185993	9	Ren...
176457	9	Ric...
176278	6	Rod...

Tabela de dispersão

- Um vetor no qual os dados são inseridos usando uma *função de hash* é chamado de *Tabela de dispersão*.

- Cada objeto armazenado na tabela de dispersão deve possuir uma chave de busca que é única para cada objeto.

- A chave de busca ou chave de pesquisa é utilizada nas operações de inclusão, exclusão e pesquisa

Tabela de dispersão

- Para **pesquisar** um elemento na tabela de dispersão:
  - Calcular o *valor de hash* da chave para identificar o índice do vetor
  - Se o índice estiver vazio, o elemento não foi adicionado
- Para **inserir** um elemento na tabela de dispersão
  - Calcular o *valor de hash* da chave para identificar o índice do vetor
  - Inserir o elemento no índice calculado
- Para **excluir** um elemento da tabela de dispersão
  - Calcular o *valor de hash* da chave para identificar o índice do vetor
  - Excluir o elemento no índice calculado

Colisões

Ao comprimir uma faixa de chaves com grande intervalo numa faixa de pequeno intervalo, não há garantias de que não haja duas chaves com o mesmo índice do vetor

Mátricula	Índice	Aluno
176489	9	Adr...
176203	11	Bea...
176770	2	Bru...
181539	3	Dou...
181656	8	Edu...
181947	11	Fáb...
181766	6	Gab...
185475	3	Gab...
179264	0	Jon...
176081	8	Luc...
181616	0	Mat...
182455	7	Mig...
175416	1	Pau...
185993	9	Ren...
176457	9	Ric...
176278	6	Rod...

Colisões

- Ao calcular o valor de *hash* para um elemento e identificar que a posição do vetor já está ocupada, afirmamos que ocorreu uma **colisão**.

- Para reduzir as colisões deve-se:
  - Aumentar o tamanho do vetor e
  - Criar um vetor cuja quantidade de elementos seja um número primo

- Existem algumas formas de resolver a colisão
  - Endereçamento aberto
  - Endereçamento separado

Fator de carga

- É a proporção do número de dados armazenados no mapa sobre o tamanho total do vetor.

$$f = \frac{N}{M}$$

  - N = quantidade de dados armazenados
  - M = tamanho do vetor

- Para minimizar colisões mas não desperdiçar muito espaço, utiliza-se um fator de carga igual à 0,75.

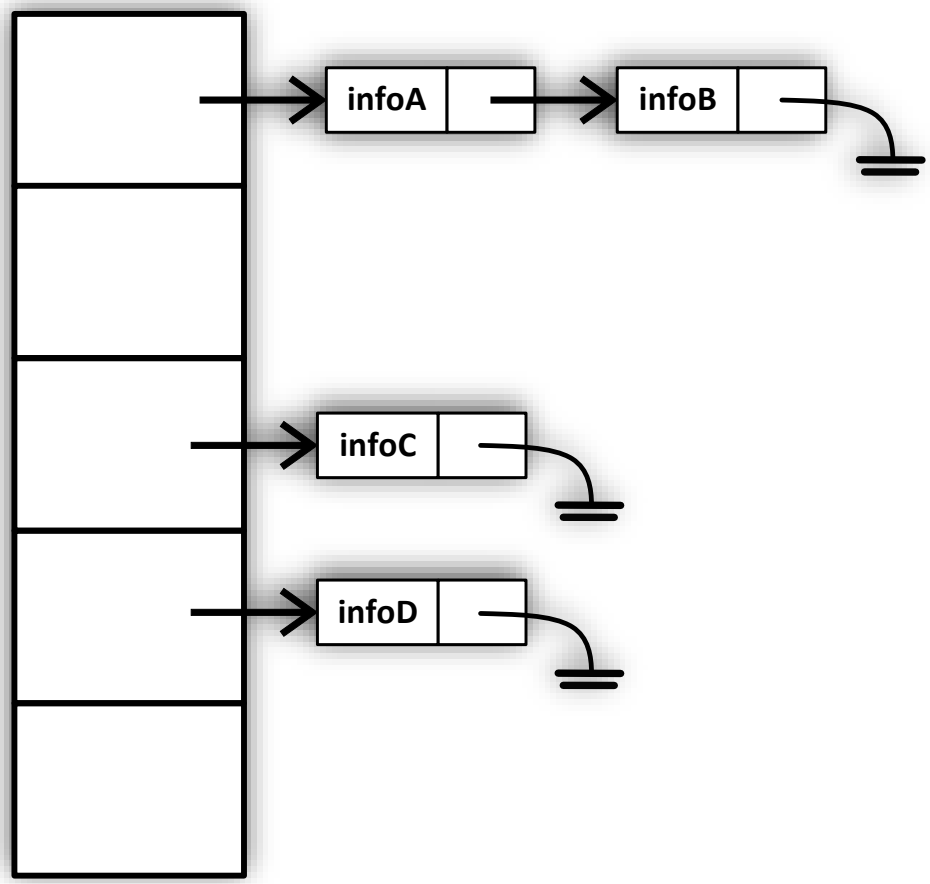
Funções de dispersão

- Uma boa função de dispersão tem as seguintes características:
  - Tende a espalhar as chaves pelo vetor
  - Simples, e por isso, rápida de computar

- Como a função de hash deriva o índice do vetor a partir de uma chave de busca, é esperado que a chave de busca seja imutável.

Tratamento de colisões por Endereçamento separado

Neste abordagem, o vetor não armazena os dados, ao invés disso, mantém listas encadeadas, para armazenar os dados.



Endereçamento separado

- A chave de um item de dados é convertida para o índice. O elemento é adicionado na lista encadeada daquele índice.
  - Não há necessidade de buscar posições vazias no vetor primário, como ocorre com o endereçamento aberto.

- A pesquisa requer efetuar uma busca linear na lista encadeada do índice calculado pela função de dispersão

- Esquema mais simples de implementar do que os algoritmos de endereçamento aberto.

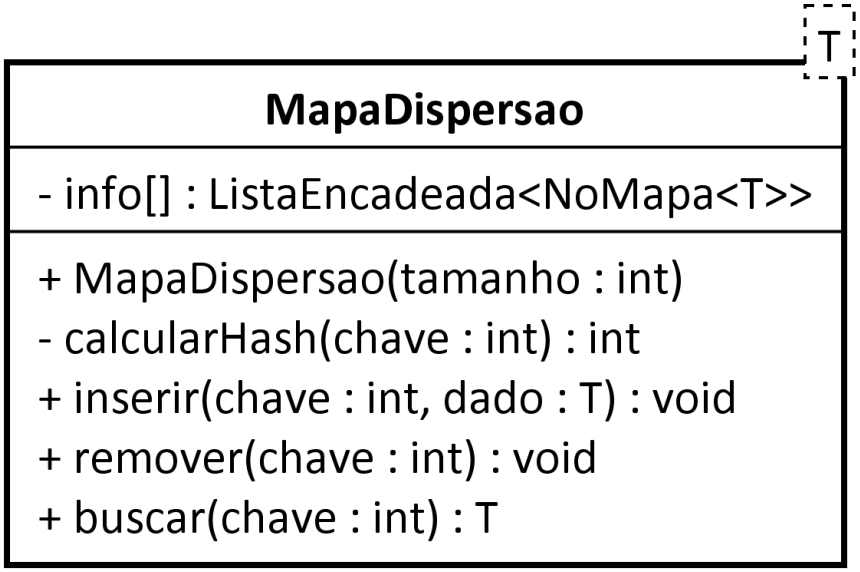
Desvantagens

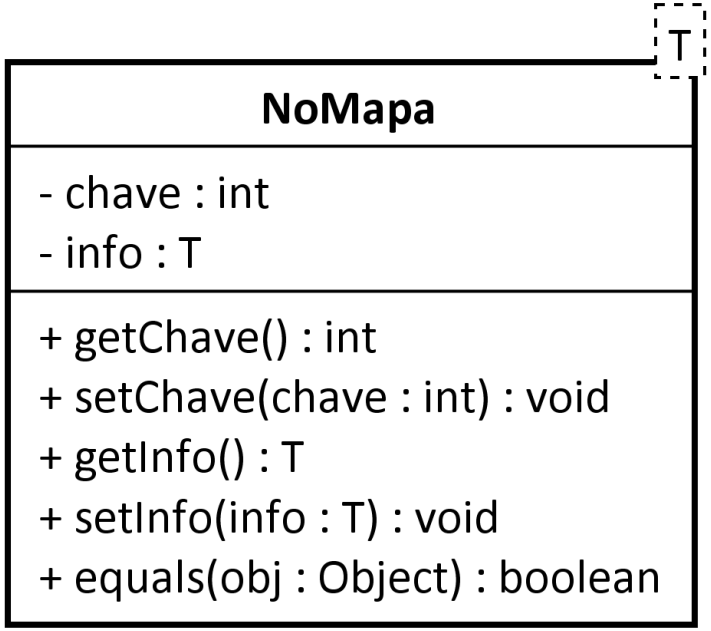
- Algumas desvantagens:
  - Baseiam-se em vetores
    - Difíceis de expandir
  - Podem ter desempenho reduzido de forma significativa
    - Requer que o programador saiba antecipadamente a quantidade aproximada de dados.
  - Não há uma maneira conveniente de visitar os itens de uma tabela de dispersão de forma ordenada

Projeto

FURB
UNIVERSIDADE DE BLUMENAU

Projeto

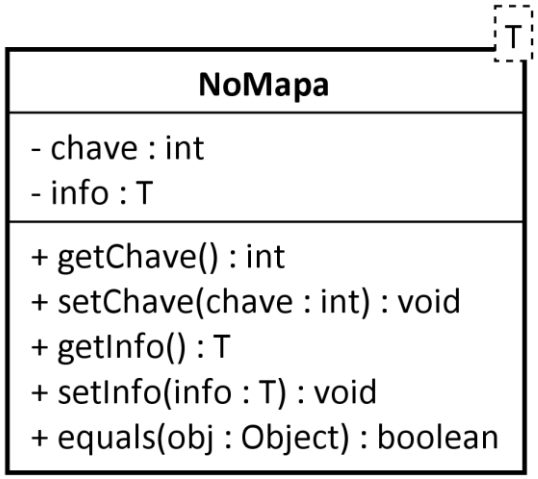




Mantém o par chave/objeto

Classe NoMapa

- Implementamos a *identidade por valor* na classe NoMapa como sendo realizada pelo atributo chave.




```
public boolean equals(Object obj) {
    if (this == obj)
      return true;
    if (obj == null)
      return false;
    if (getClass() != obj.getClass())
      return false;
    NoMapa other = (NoMapa) obj;
    if (chave != other.chave)
      return false;
    return true;
}
```

Manipulação do mapa de dispersão

- **Criação do mapa**
  - Cria um vetor capaz de armazenar listas encadeadas
  - O tamanho do vetor é predefinido pela classe cliente

```
Algoritmo: criarMapa(int tamanho)

info ← new ListaEncadeada[tamanho];
```

- **Cálculo de hash**
  - "Compacta" uma chave

```
Algoritmo: calcularHash(int chave)

tamanho ← size(info);
retornar chave mod tamanho;
```

Operador de
resto de divisão

Inserir um objeto no mapa

1. Deve calcular o *hash* para a chave de busca
2. Criar uma lista encadeada na posição calculada, caso ainda não houver
3. Criar um objeto que armazena o par chave/valor
4. Armazenar o objeto na lista

```
Algoritmo: inserir(int chave, int dado)

1  indice ← calcularHash(chave);

2  se (info[indice] = nulo) então
   inicio
     info[indice] ← new ListaEncadeada( );
   fim-se;

3  noMapa ← new NoMapa( );
   noMapa.chave ← chave;
   noMapa.info ← dado;

4  info[indice].inserir(noMapa);
```

Buscar um objeto no mapa

1. Calcula o *hash* a partir da chave de busca
2. Cria um objeto de mapeamento, estabelecendo apenas a *chave de busca*
3. Procura o objeto na lista encadeada, retornando o objeto armazenado, caso encontrado

```
Algoritmo: buscar(int chave)

1  indice ← calcularHash(chave);

   se (info[indice] ≠ nulo) então
   inicio

2     noMapa ← new NoMapa( );
      noMapa.chave ← chave;

3     no ←  info[indice].buscar(noMapa);
      se (no ≠ nulo) então
         retornar no.info.info;
      fim-se;
   fim-se;

   retornar nulo;
```