



Algoritmos e Estruturas de Dados

TAD Map – Capítulo 10
2019/20

Package dataStructures: interfaces (TADs)

- **Dicionário** (interface *Map*)
 - Coleção de elementos não repetidos, identificados por uma chave.
 - Acesso por chave (inserção, remoção, pesquisa).



TAD *Map* (1)

```
package dataStructures;
```

```
public interface Map<K, V> {
```

```
    // Returns true iff the map contains no entries.
```

```
    boolean isEmpty( );
```

```
    // Returns the number of entries in the map.
```

```
    int size( );
```

```
    // Returns an iterator of the keys in the map.
```

```
    Iterator<K> keys( ) throws NoSuchElementException;
```

```
    // Returns an iterator of the values in the map.
```

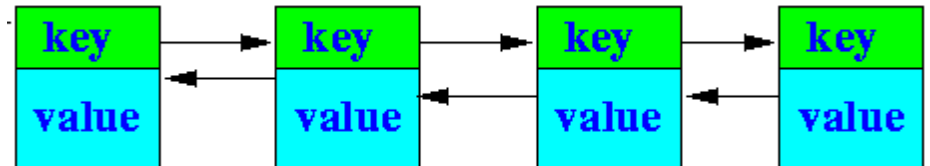
```
    Iterator<V> values( ) throws NoSuchElementException;
```

```
    // Returns an iterator of the entries in the map.
```

```
    Iterator<Entry<K,V>> iterator() throws NoSuchElementException;
```

```
    ...
```

```
}
```



TAD *Map* (2)

```
package dataStructures;
```

```
public interface Map<K, V> {
```

```
...
```

```
// If there is an entry in the map whose key is the specified key,  
// returns its value; otherwise, returns null.
```

```
V find( K key );
```

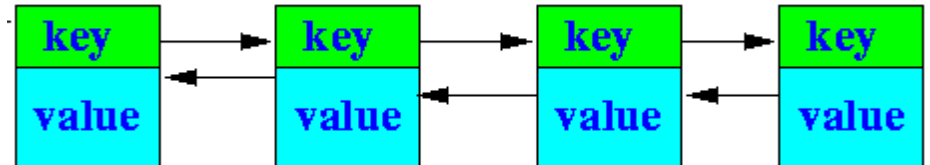
```
// If there is an entry in the map whose key is the specified key,  
// replaces its value by the specified value and returns the old value;  
// otherwise, inserts the entry (key, value) and returns null.
```

```
V insert( K key, V value );
```

```
// If there is an entry in the map whose key is the specified key,  
// removes it from the map and returns its value; otherwise, returns null.
```

```
V remove( K key );
```

```
}
```



Interface Entry

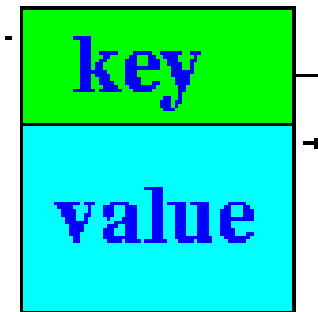
```
package dataStructures;

public interface Entry<K, V> {

    // Returns the key in the entry.
    K getKey( );

    // Returns the value in the entry.
    V getValue( );

}
```





TAD $\text{Map}\langle K, V \rangle$

Opções de implementação

- A implementação pode ser realizada usando uma das seguintes estruturas de dados:
 - Vector;
 - Lista ligada (simples ou dupla);

TAD **Map**<K,V> com **Vector** (1)

- Esta implementação pode ser realizada usando a classe **Array**<E>, onde E é uma **Entry**<K,V>.
 - Será uma boa implementação?

Key,value	Key,value	Key,value	Key,value	Key,value			
-----------	-----------	-----------	-----------	-----------	--	--	--

TAD $\text{Map}\langle K, V \rangle$ com **Vector** (2)

- Esta implementação pode ser realizada usando a classe **Array** $\langle E \rangle$, onde E é uma **Entry** $\langle K, V \rangle$.
 - Será uma boa implementação?

Key,value	Key,value	Key,value	Key,value	Key,value			
-----------	-----------	-----------	-----------	-----------	--	--	--

Operação Com - Sem sucesso	Caso Médio	Pior Caso
Inserção	$O(n)$ $n - n/2$ comparações	$O(n)$ n comparações
Remoção	$O(n)$ $n/2 - n$ comparações	$O(n)$ n comparações
Pesquisa	$O(n)$ $n/2 - n$ comparações	$O(n)$ n comparações
Percurso	$O(n)$ n iterações	$O(n)$ n iterações



TAD Map<K,V>

com **Lista Ligada (simples ou dupla)** (1)

- Estas implementações podem ser realizadas usando as classes **SinglyLinkedList<E>** ou **DoublyLinkedList<E>**, onde E é uma **Entry<K,V>**.
 - Será uma boa implementação?



TAD Map<K,V>

com **Lista Ligada (simples ou dupla)** (2)

- Estas implementações podem ser realizadas usando as classes **SinglyLinkedList<E>** ou **DoublyLinkedList<E>**, onde E é uma **Entry<K,V>**.

- Será uma boa implementação?



Operação Com - Sem sucesso	Caso Médio	Pior Caso
Inserção	O(n) n - n/2 comparações	O(n) n comparações
Remoção	O(n) n/2 - n comparações	O(n) n comparações
Pesquisa	O(n) n/2 - n comparações	O(n) n comparações
Percurso	O(n) n iterações	O(n) n iterações



TAD $\text{Map}\langle K, V \rangle$

Opções de implementação

- A implementação pode ser realizada usando uma das seguintes estruturas de dados:
 - Vector;
 - Lista ligada (simples ou dupla).

Não conseguimos melhor?

Uma estrutura de dados que consiga
 $O(1)$ na pesquisa



TAD Map<K,V>

Implementação com as classes do Java

```
package dataStructures;

public class MapWithJavaClass<K, V> implements Map<K, V> {

    protected java.util.Map<K,V> elementos;
    protected int capPrevista;

    public MapWithJavaClass(int prevusers) {
        elementos = new java.util.HashMap<K,V>(prevusers);
        capPrevista =prevusers;
    }
}
```

- O que será uma **tabela de dispersão**?

Uma estrutura de dados que particione o conjunto a pesquisar, ou seja, que distribua os elementos uniformemente em subconjuntos disjuntos (generalização dum vector).



O(1) na pesquisa

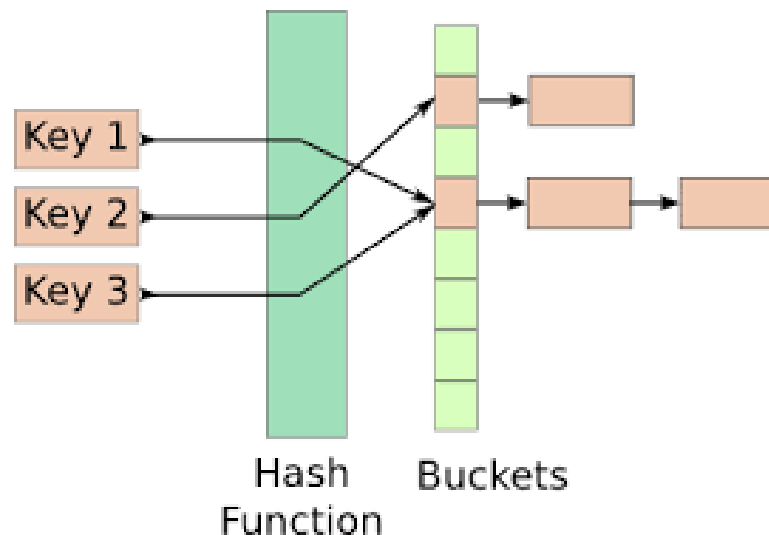
TAD Map<K,V>

Tabela de dispersão (1)

O que é uma **tabela de dispersão**?

- Uma tabela de dispersão é uma generalização de um vector que, em determinadas condições, permite aceder rapidamente à informação.

Numa inserção, a posição onde se insere o elemento é função do valor da chave do elemento, independentemente da ordem de inserção.

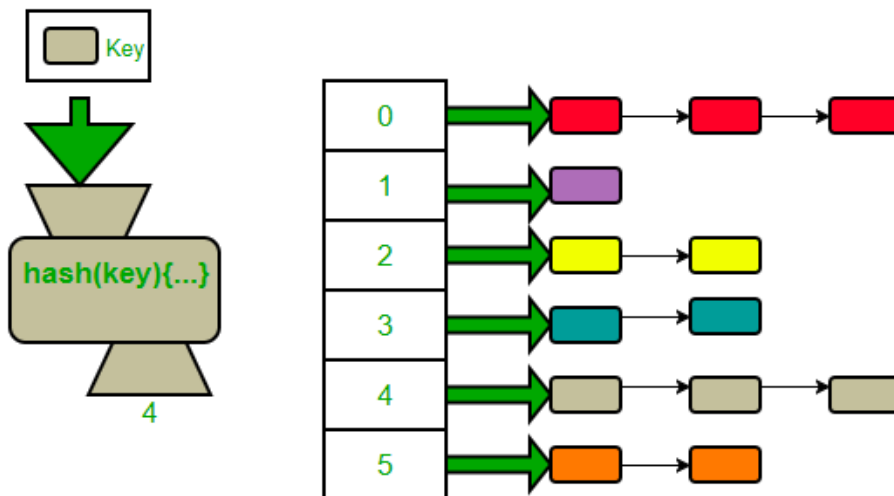


TAD Map<K,V>

Tabela de dispersão (1)

O que é uma **função de dispersão**?

- A função de dispersão converte a chave do elemento num índice da tabela (a entrada no vector).

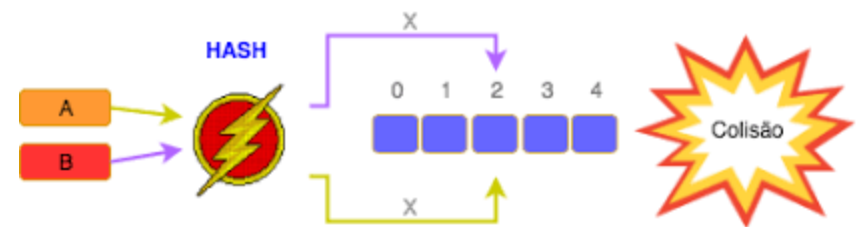
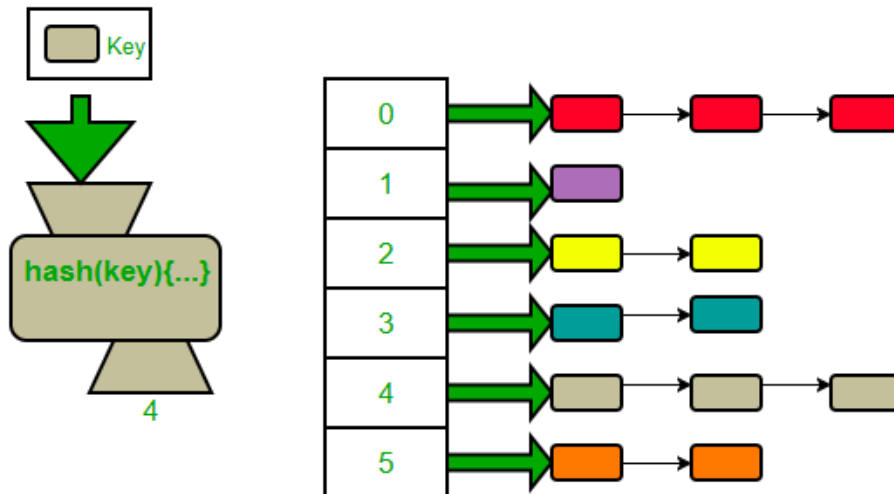


TAD Map<K,V>

Tabela de dispersão (2)

O que é uma **função de dispersão**?

- A função de dispersão converte a chave do elemento num índice da tabela (a entrada no vector).
- Se a função de dispersão indicar a mesma entrada para duas chaves distintas, então surge um problema de colisão.



TAD $\text{Map}\langle K, V \rangle$

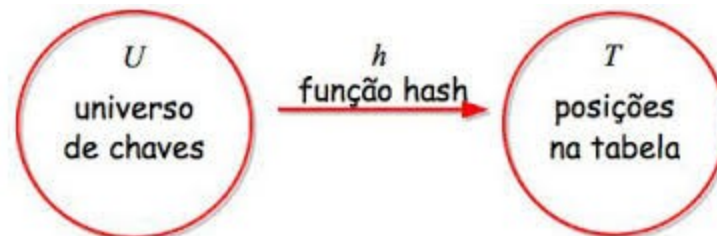
Função de dispersão (1)

Objectivos

- Deve ser eficiente.
- Deve distribuir as chaves uniformemente por todas as posições da tabela.

Regras Práticas

- A função de dispersão deve ser simples.
- A dimensão da tabela (dim) deve ser um número primo.
- Se as chaves forem grandes, deve-se considerar apenas uma parte, oriunda de vários pontos.



TAD Map<K,V>

Função de dispersão (2)

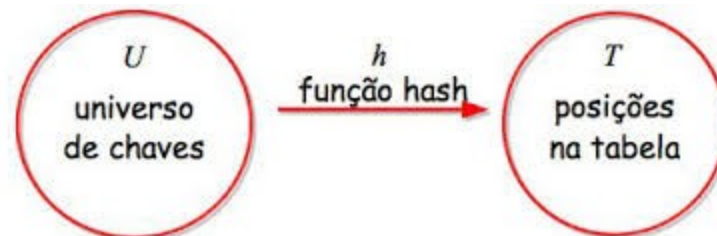
Calculo da função de dispersão (2 passos)

Passo 1: Cada chave sabe calcular o seu código de dispersão.

```
public int hashCode( );
```

Passo 2: A tabela de dispersão converte o código de dispersão da chave num índice da tabela.

```
Math.abs( key.hashCode() ) % table.length
```



TAD Map<K,V>

Função de dispersão (3)

Exemplos de funções de dispersão

- Se a chave é um número inteiro n : n .
- Se a chave é uma cadeia de caracteres $s_0 s_1 \cdots s_{n-1}$:
 - i. $s_0 + s_1 + \cdots + s_{n-1}$;
 - ii. $(s_0 a^{n-1} + s_1 a^{n-2} + \cdots + s_{n-1}) \% b$, com a e b primos.

Regra de Horner

$$\text{códigoDisp}(s_0 s_1 \cdots s_{n-1}) = (s_0 a^{n-1} + s_1 a^{n-2} + \cdots + s_{n-1}) \% b$$

TAD Map<K,V>

Função de dispersão (4)

Função de dispersão (sequência de caracteres)

$$\text{códigoDisp}(s_0 s_1 \cdots s_{n-1}) = (s_0 a^{n-1} + s_1 a^{n-2} + \cdots + s_{n-1}) \% b$$

Regra de Horner

$$v \leftarrow 0;$$

$$v \leftarrow (v * a + s_0) \% b = (s_0) \% b$$

$$v \leftarrow (v * a + s_1) \% b = (s_0 a + s_1) \% b$$

$$v \leftarrow (v * a + s_2) \% b = (s_0 a^2 + s_1 a + s_2) \% b$$

$$v \leftarrow (v * a + s_3) \% b = (s_0 a^3 + s_1 a^2 + s_2 a + s_3) \% b$$

.....

TAD Map<K,V>

Função de dispersão (5)

Função de dispersão (sequência de caracteres)

$$\text{códigoDisp}(s_0 s_1 \cdots s_{n-1}) = (s_0 a^{n-1} + s_1 a^{n-2} + \cdots + s_{n-1}) \% b$$

Regra de Horner

```
public static int hash( String key ){
    int a = 127;
    // a is a prime number.
    int b = 2147483647;
    // b is a prime number.
    int hashCode = 0;

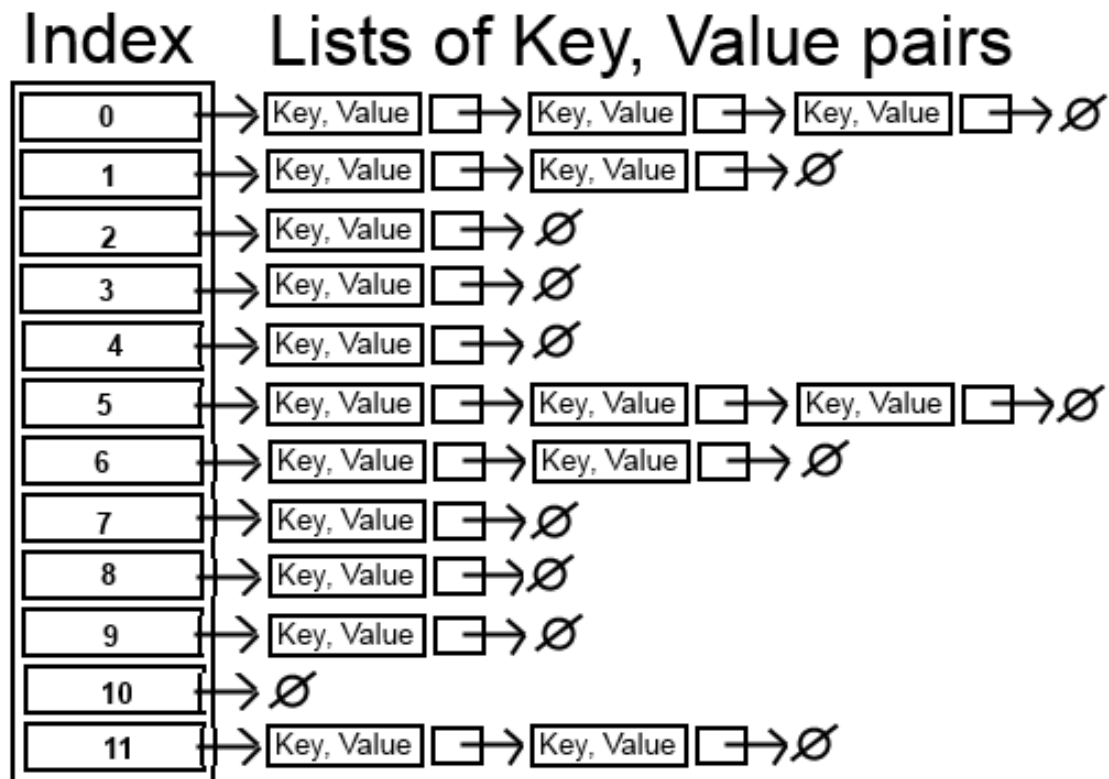
    for ( int i = 0; i < key.length(); i++ )
        hashCode = ( hashCode * a + key.charAt(i) ) % b;
    return hashCode;
}
```

TAD Map<K,V>

Colisões (1)

Como tratar as **colisões**?

A) Tabela de dispersão aberta.



TAD Map<K,V>

Colisões (2)

Como tratar as **colisões**?

B) Tabela de dispersão fechada.

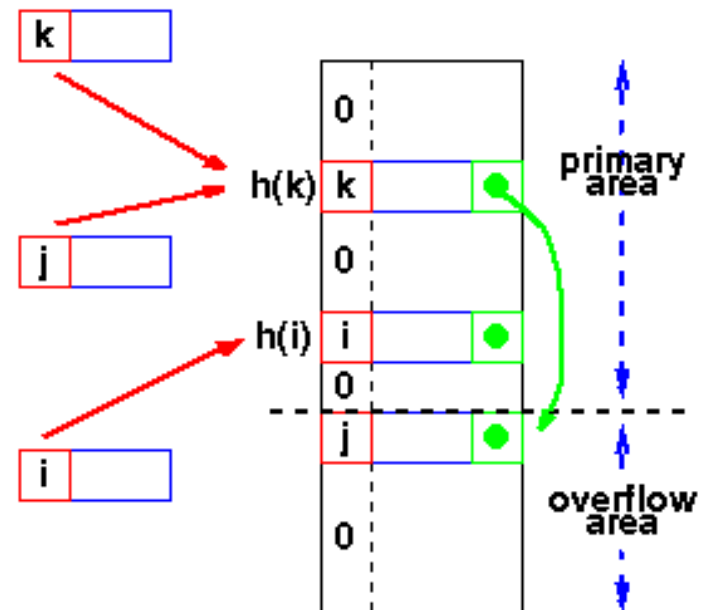
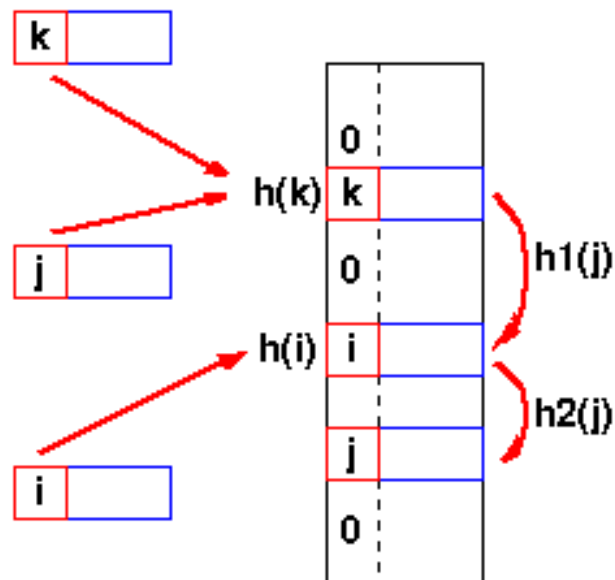
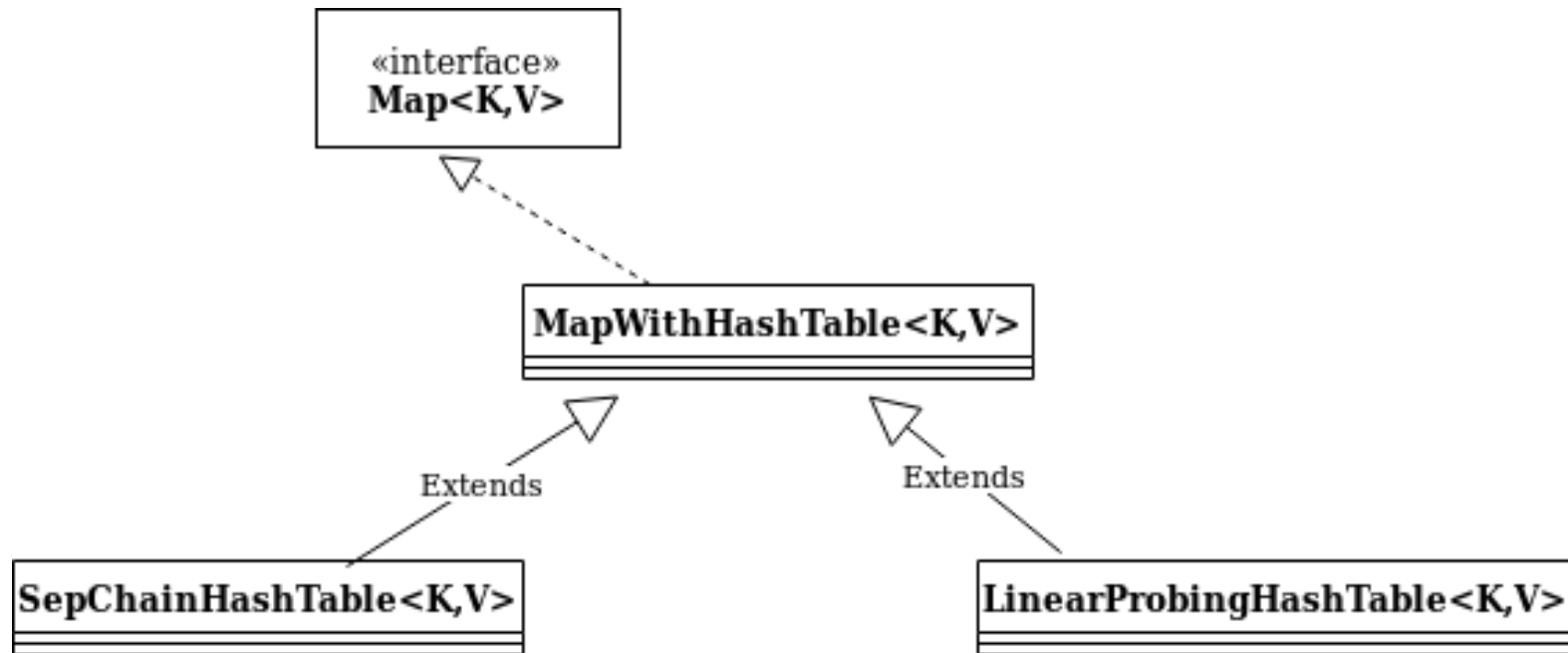


Diagrama de classes (interface **Map**)

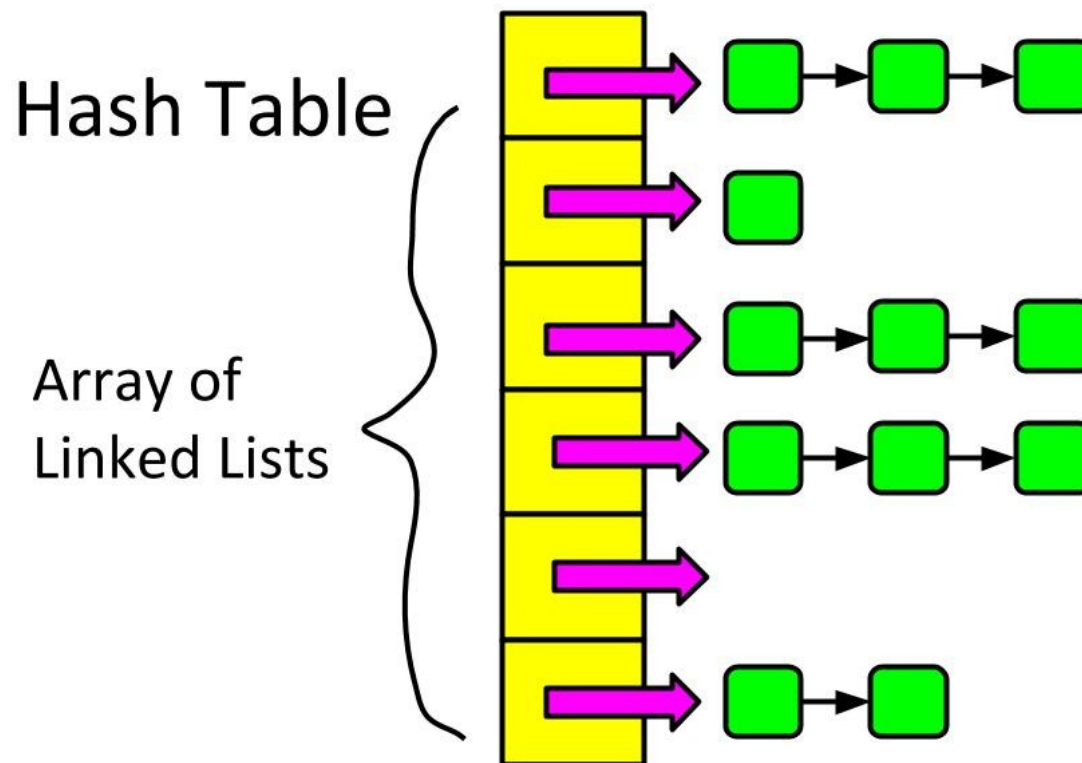


TAD $\text{Map}\langle K, V \rangle$

Tabela de dispersão aberta (1)

Como implementar a **tabela de dispersão aberta**?

- ✓ Vector de listas ligadas

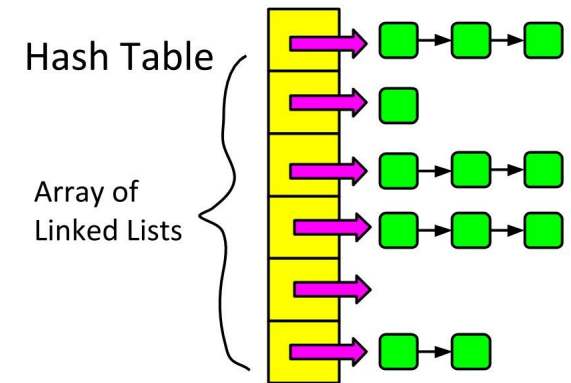


TAD Map<K,V>

Tabela de dispersão aberta (2)

Qual o tamanho do vector na **tabela de dispersão aberta**?

- ✓ Sejam
 - ✓ n – o número de entradas na tabela;
 - ✓ dim – a dimensão do vector
 - ✓ λ - o comprimento médio das listas de colisões (deve ser sempre inferior a 1)



Factor de ocupação da tabela de dispersão

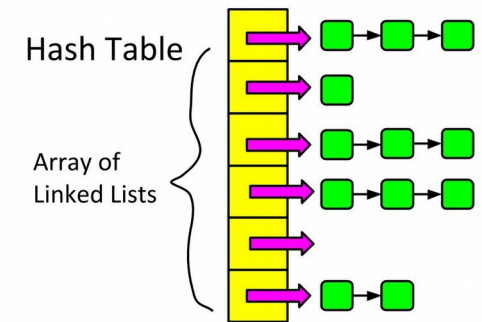
$$\lambda = \frac{n}{dim}$$

TAD Map<K,V>

Tabela de dispersão aberta (3)

Criação: criar o vector de listas ligadas e criar todas as listas de colisões (que podem ser simples ou duplas)

Pesquisa	Melhor Caso	Pior Caso	Caso Esperado
Com sucesso	$O(1)$	$O(n)$	$O(1+\lambda)$
Sem sucesso	$O(1)$	$O(n)$	$O(1+\lambda)$



Considerações sobre o factor de ocupação λ

- ✓ Se $\lambda < 1$ e a função de dispersão for apropriada, o preenchimento da tabela será esparsa e a dimensão das listas curta. Permitindo assim uma pesquisa $O(1)$ no caso médio.
- ✓ Quando $\lambda > 0.9$ deve-se fazer uma redispersão (criar uma nova tabela maior e inserir os elementos).
 - A classe HashMap do Java tem como valor por omissão o $\lambda = 0.75$.

TAD Map<K,V>

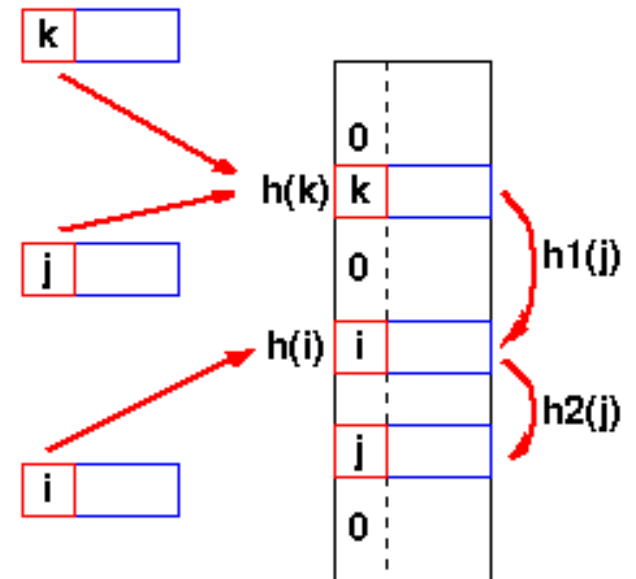
Tabela de dispersão fechada (1)

Como implementar a **tabela de dispersão fechada**?

- ✓ Vector de **Entry<K,V>**

Como tratar as colisões?

- ✓ Sondagem linear
- ✓ Dispersão dupla
- ✓ Várias funções de dispersão, ...



TAD Map<K,V>

Tabela de dispersão fechada (2)

Sondagem Linear

$\text{dispersão} : K \rightarrow \{0, 1, 2, \dots, \text{dim}-1\}$

$\text{dispersão}(k) = i$

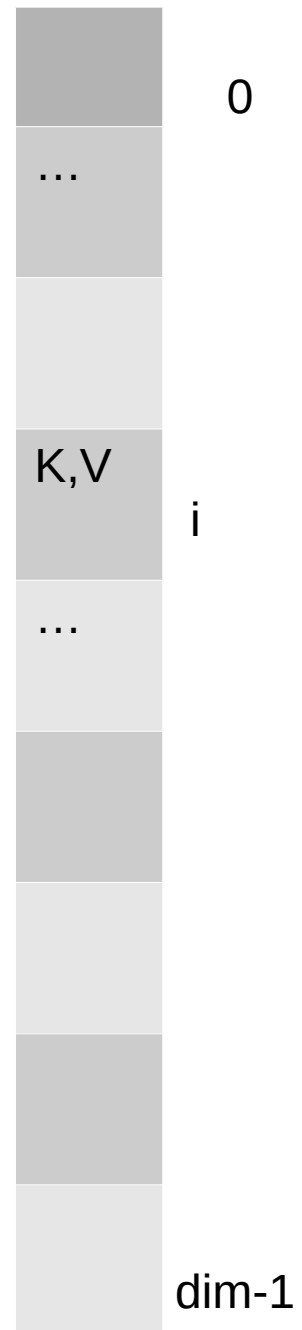
$\text{dispersão}(k') = i$

Na pesquisa da chave k' , quando a posição da tabela indicada pela função de dispersão - $\text{dispersão}(k')$ - já está ocupada com uma entrada cuja chave é diferente de k' , recorre-se a uma segunda função - **a função de sondagem** - que indica outra posição da tabela.

Visitam-se as posições $p_0(k')$, $p_1(k')$, $p_2(k')$, ... tais que:

$$p_j(k') = (\text{dispersão}(k') + \text{sondagem}(j)) \% \text{dim}$$

Tendo-se sempre $\text{sondagem}(0) = 0$



TAD Map<K,V>

Tabela de dispersão fechada (3)

Sondagem Linear

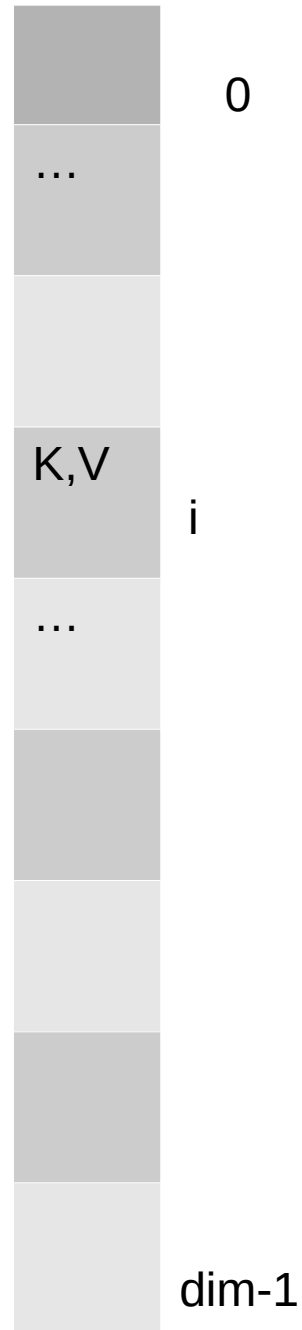
Neste método a função de sondagem é representada por:

$$\text{sondagem}(j) = j$$

$$\begin{array}{ll} \text{sondagem}(0) = 0 & p_0 = (\text{dispersão}(k) + 0) \% \text{dim} \\ \text{sondagem}(1) = 1 & p_1 = (\text{dispersão}(k) + 1) \% \text{dim} \\ \text{sondagem}(2) = 2 & p_2 = (\text{dispersão}(k) + 2) \% \text{dim} \\ \text{sondagem}(3) = 3 & p_3 = (\text{dispersão}(k) + 3) \% \text{dim} \end{array}$$

Primeira Posição: $\text{pos} \leftarrow \text{dispersão}(k)$

Posição Seguinte: $\text{pos} \leftarrow (\text{pos} + 1) \% \text{dim}$



TAD Map<K,V>

Tabela de dispersão fechada (4)

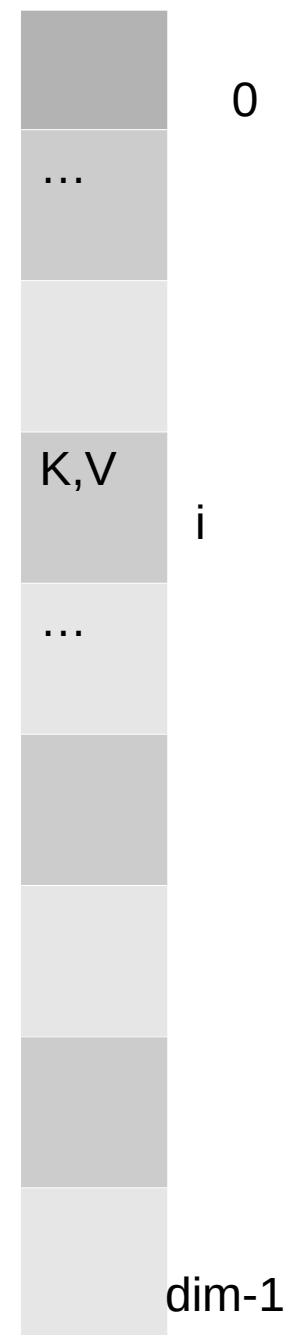
Problemas da Sondagem Linear

- Existência de blocos contíguos de posições ocupadas de grandes dimensões (em geral, quando $\lambda \geq 0.5$).
- A sondagem é igual para chaves que colidem.

Método de resolução de Colisões	λ ideal	λ máximo
Sondagem Linear	0.5	0.8

Qual o tamanho do vector na **tabela de dispersão fechada**?

Criação: Deve-se pré-dimensionar a tabela tendo em vista o fator de ocupação ideal, permitindo inserções apenas enquanto não for atingido o fator de ocupação máximo. Ou fazendo um redispersão (criar uma nova tabela maior e inserir os elementos).



TAD Map<K,V>

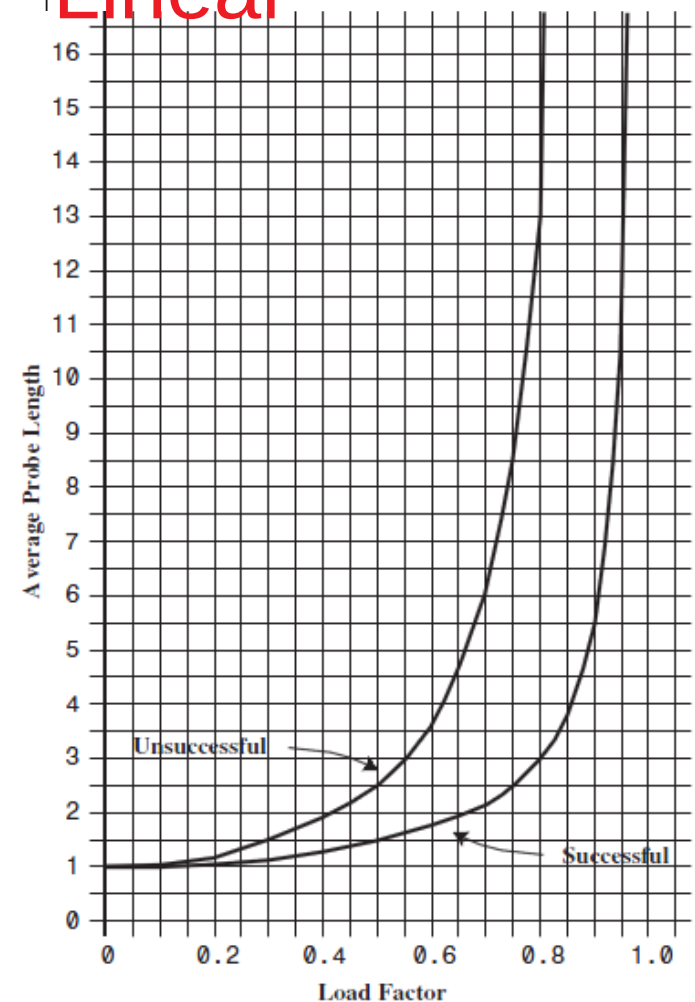
Tabela de dispersão fechada (5)

Pesquisa	Melhor Caso	Pior Caso	Caso Esperado
com sucesso	1	n	$\frac{1}{2} \left(1 + \frac{1}{1 - \lambda}\right)$
sem sucesso	1	n	$\frac{1}{2} \left(1 + \frac{1}{(1 - \lambda)^2}\right)$

(n é o número de entradas na tabela)

Pesquisa no Caso Esperado	Factor de Ocupação				
	0.25	0.5	0.75	0.8	0.9
com sucesso	1.2	1.5	2.5	3.0	5.5
sem sucesso	1.4	2.5	8.5	13.0	50.5

Sondagem
Linear



TAD Map<K,V>

Classe abstracta Tabela de dispersão (1)

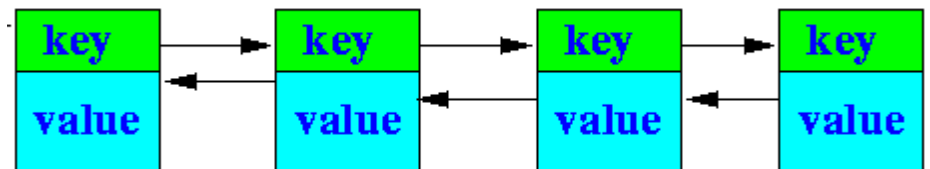
```
package dataStructures;

public abstract class MapWithHashTable<K, V> implements Map<K, V> {

    // Default size of the hash table.
    protected static final int DEFAULTCAPACITY = 50;

    // Number of entries in the hash table.
    protected int currentSize;

    // Maximum number of entries.
    protected int maxSize;
```



TAD Map<K,V>

Classe abstracta Tabela de dispersão (2)

```
// Public Static Methods
```

```
// Returns the hash code of the specified key,  
// which is an integer in the range 0, ..., b-1.
```

```
public static int hash( String key )
```

```
// Returns a prime number that is not less than the specified number;  
// or zero if all such primes are greater than Integer.MAX VALUE.
```

```
protected static int nextPrime( int number ){
```

```
    for ( int i = 0; i < PRIMES.length; i++ )
```

```
        if ( PRIMES[i] >= number )
```

```
            return PRIMES[i];
```

```
    return 0;
```

```
}
```

```
protected static final int[]
```

```
PRIMES={11, 19, 31, 47, 73,
```

```
113, 181, 277, 421, 643, 967,
```

```
1451, 2179, 3299, 4951, 7433,
```

```
11173, 16763, 25163, 37747, 56671, 85009,
```

```
127529, 191299, 287003, 430517, 645787, 968689,
```

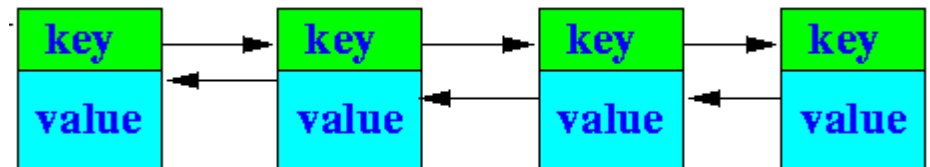
```
1453043, 2179571, 3269377, 4904077, 7356119,
```

```
11034223, 16551361, 24827059, 37240597, 55860923, 83791441,
```

```
125687173, 188530777, 282796177, 424194271, 636291413, 954437161,
```

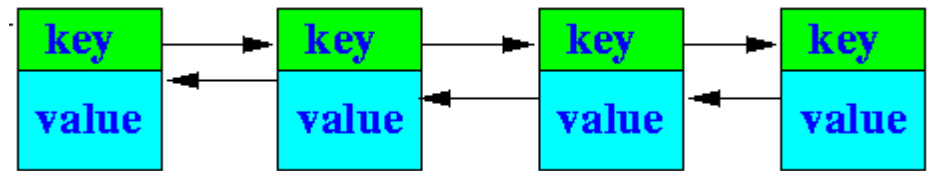
```
1431655751, 2147483647
```

```
};
```



TAD Map<K,V>

Classe abstracta Tabela de dispersão (3)

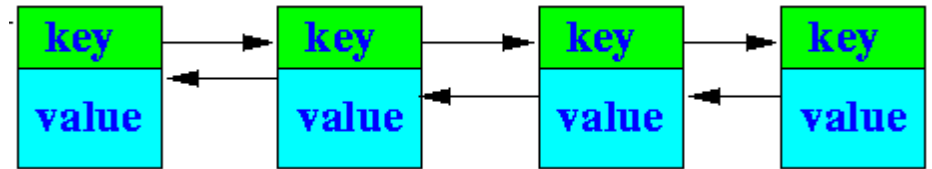


```
@Override
// Returns true iff the dictionary contains no entries.
public boolean isEmpty() {
    return currentSize == 0;
}

@Override
// Returns the number of entries in the dictionary.
public int size() {
    return currentSize;
}
```

TAD Map<K,V>

Classe abstracta Tabela de dispersão (4)



@Override

// If there is an entry in the map whose key is the specified key,
// returns its value; otherwise, returns null.

public abstract V find(K key);

@Override

// If there is an entry in the map whose key is the specified key,
// replaces its value by the specified value and returns the old value;
// otherwise, inserts the entry (key, value) and returns null.

public abstract V insert(K key, V value);

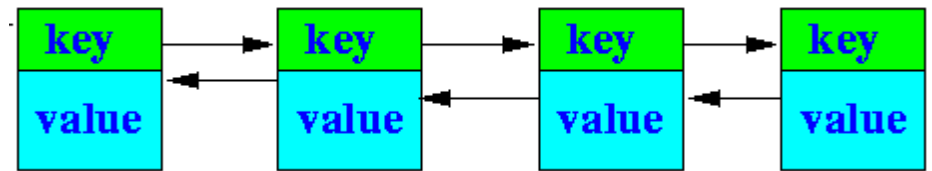
@Override

// If there is an entry in the map whose key is the specified key,
// removes it from the map and returns its value; otherwise, returns null.

public abstract V remove(K key);

TAD Map<K,V>

Classe abstracta Tabela de dispersão (5)



@Override

// Returns an iterator of the keys in the map.

public abstract Iterator<K> keys() **throws** NoSuchElementException;

@Override

// Returns an iterator of the values in the map.

public abstract Iterator<V> values() **throws** NoSuchElementException;

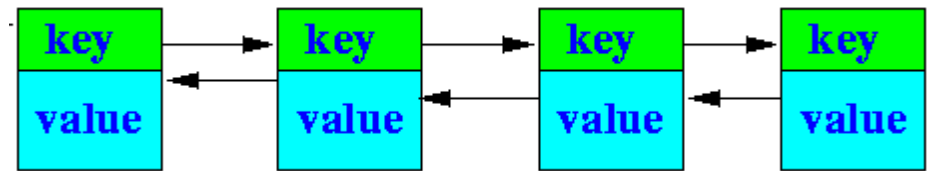
@Override

// Returns an iterator of the entries in the map.

public abstract Iterator<Entry<K,V>> iterator() **throws** NoSuchElementException;

TAD Map<K,V>

Classe abstracta Tabela de dispersão (6)



```
// Protected Instance Methods
```

```
// Returns true iff the hash table cannot contain more entries.
```

```
protected boolean isFull( ){  
    return currentSize == maxSize;  
}
```

```
}
```

TAD Map<K,V>

Classe Tabela de dispersão aberta (1)

```
package dataStructures;
```

```
public class SepChainHashTable<K, V> extends MapWithHashTable<K,V>{
```

```
// The array of maps.
```

```
protected Map<K,V>[] table;
```

```
public SepChainHashTable(){  
    this(DEFAULTCAPACITY);  
}
```

```
@SuppressWarnings("unchecked")
```

```
public SepChainHashTable(int capacity) {
```

```
// Load factor is 1/1.3 (0.75)
```

```
int arraySize = MapWithHashTable.nextPrime((int) (1.3 * capacity));
```

```
// Compiler gives a warning.
```

```
table = (Map<K,V>[]) new Map[arraySize];
```

```
for ( int i = 0; i < arraySize; i++ )
```

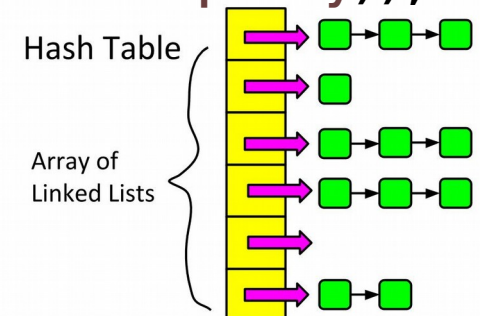
```
    table[i] = new MapWithSinglyLinkedList<K,V>();
```

```
maxSize = capacity;
```

```
currentSize = 0;
```

```
}
```

Exercício da aula prática



TAD Map<K,V>

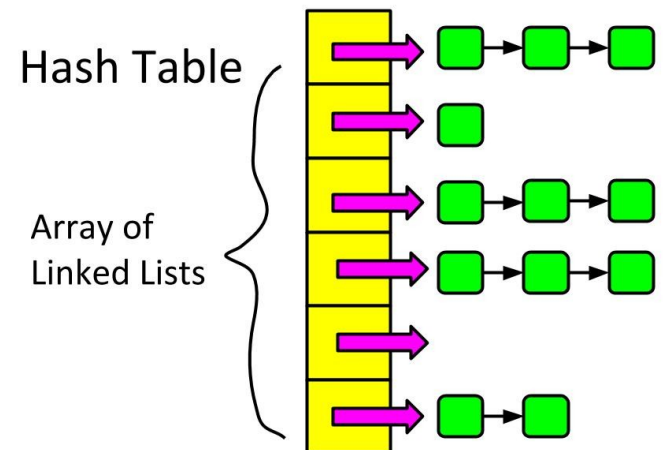
Classe Tabela de dispersão aberta (2)

```
// Returns the hash value of the specified key.  
protected int hash( K key ){  
    return Math.abs( key.hashCode() ) % table.length;  
}
```

```
// If there is an entry in the map whose key is the specified key,  
// returns its value; otherwise, returns null.
```

```
@Override
```

```
public V find( K key ){  
    return table[ this.hash(key) ].find(key);  
}
```



TAD Map<K,V>

Tabela de dispersão aberta

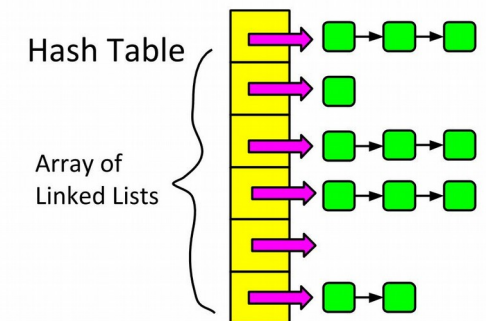
Exemplo inserção (1)

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

1. Inserir $(14, V_{14})$

$$\text{Dispersão}(14) = 3$$

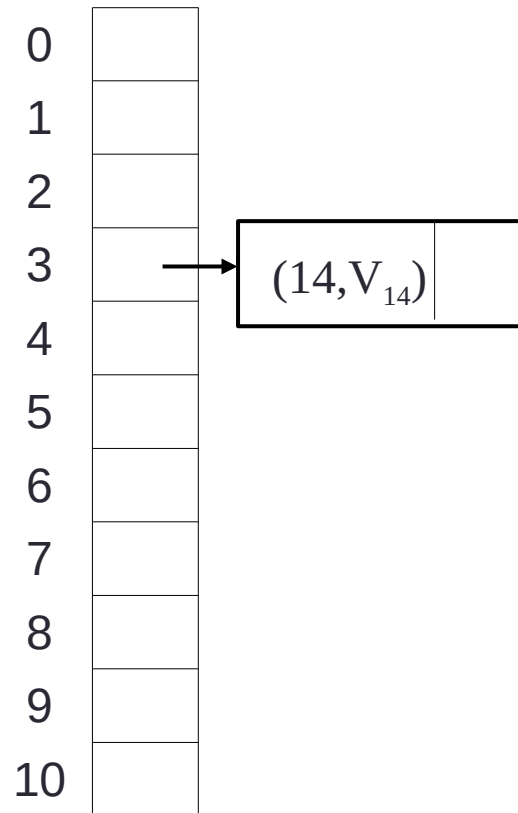
$$\text{Dispersão}(k) = k \% 11$$



TAD Map<K,V>

Tabela de dispersão aberta

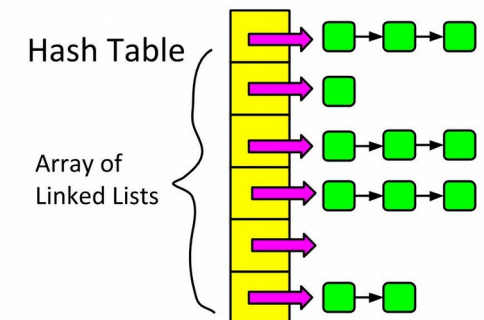
Exemplo inserção (2)



1. Inserir $(14, V_{14})$

$$\text{Dispersão}(14) = 3$$

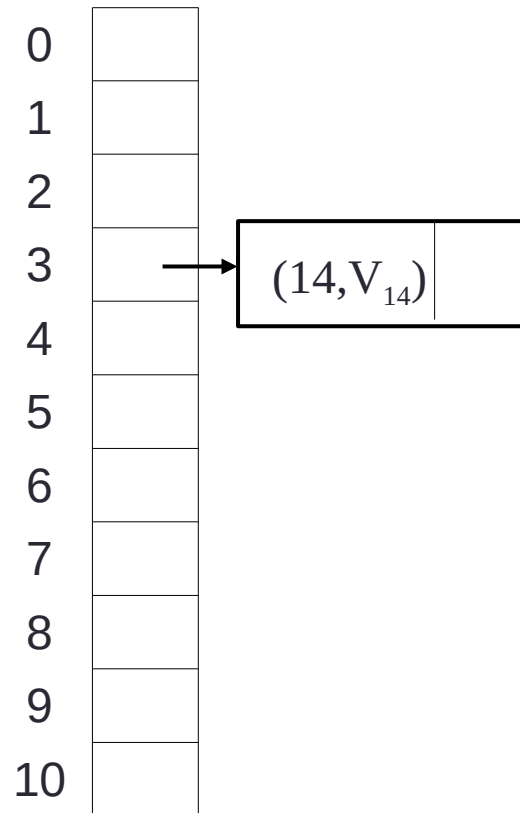
$$\text{Dispersão}(k) = k \% 11$$



TAD Map<K,V>

Tabela de dispersão aberta

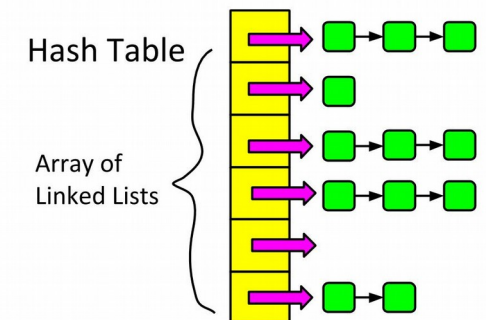
Exemplo inserção (3)



1. Inserir $(14, V_{14})$
 $\text{Dispersão}(14) = 3$

2. Inserir $(29, V_{29})$
 $\text{Dispersão}(29) = 7$

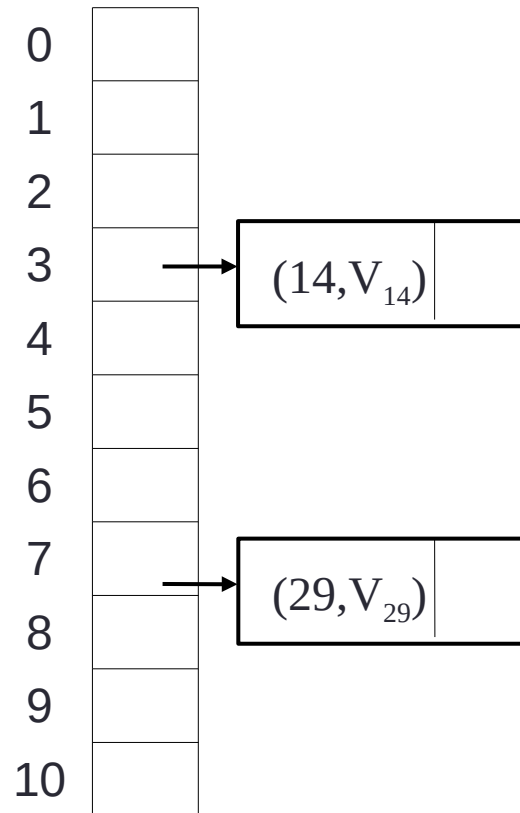
$$\text{Dispersão}(k) = k \% 11$$



TAD Map<K,V>

Tabela de dispersão aberta

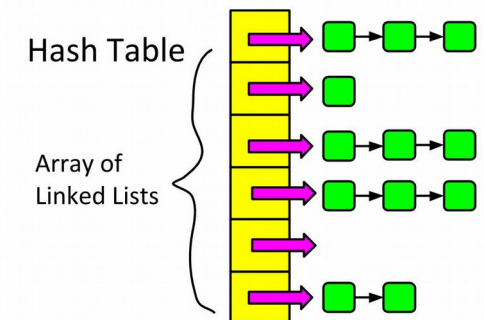
Exemplo inserção (4)



1. Inserir $(14, V_{14})$
 $\text{Dispersão}(14) = 3$

2. Inserir $(29, V_{29})$
 $\text{Dispersão}(29) = 7$

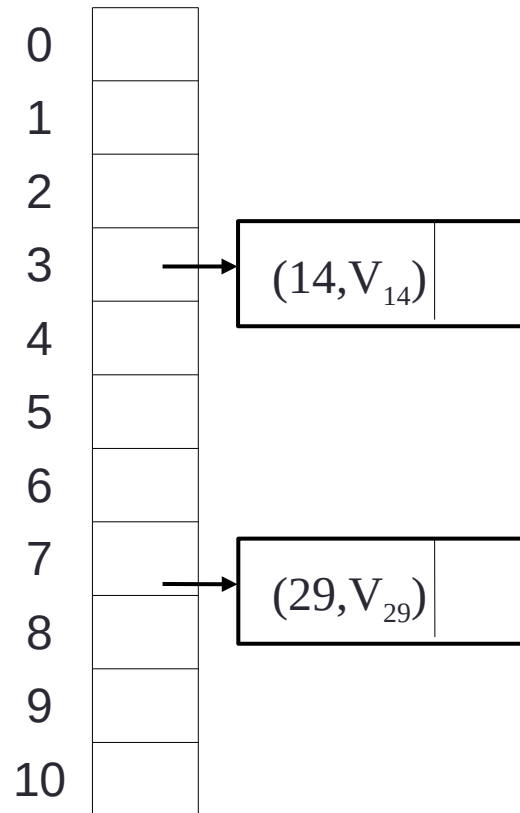
$$\text{Dispersão}(k) = k \% 11$$



TAD Map<K,V>

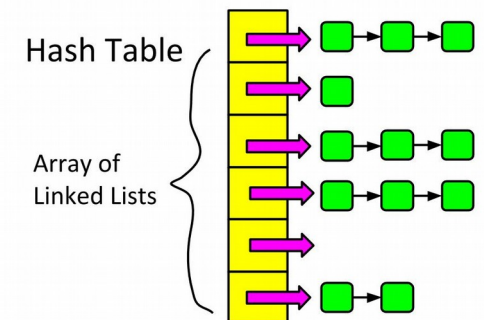
Tabela de dispersão aberta

Exemplo inserção (5)



$$\text{Dispersão}(k) = k \% 11$$

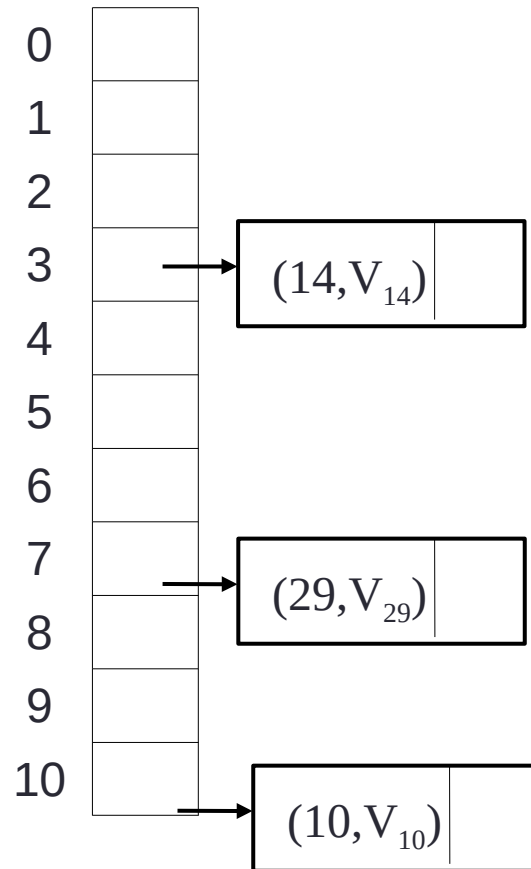
1. Inserir $(14, V_{14})$
 $\text{Dispersão}(14) = 3$
2. Inserir $(29, V_{29})$
 $\text{Dispersão}(29) = 7$
3. Inserir $(10, V_{10})$
 $\text{Dispersão}(10) = 10$



TAD Map<K,V>

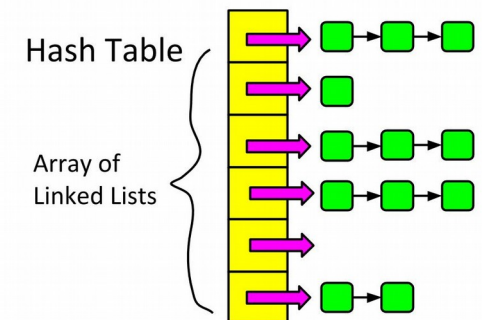
Tabela de dispersão aberta

Exemplo inserção (6)



$$\text{Dispersão}(k) = k \% 11$$

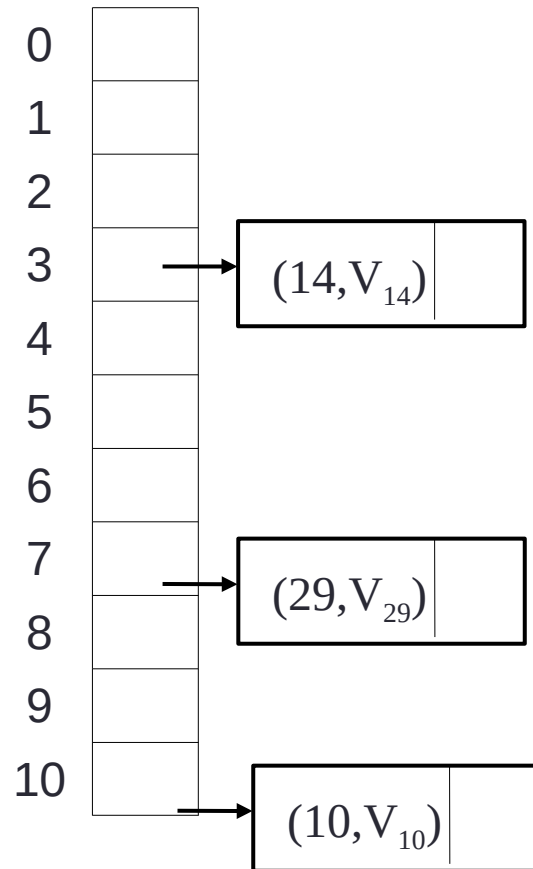
1. Inserir (14, V₁₄)
 $\text{Dispersão}(14) = 3$
2. Inserir (29, V₂₉)
 $\text{Dispersão}(29) = 7$
3. Inserir (10, V₁₀)
 $\text{Dispersão}(10) = 10$



TAD Map<K,V>

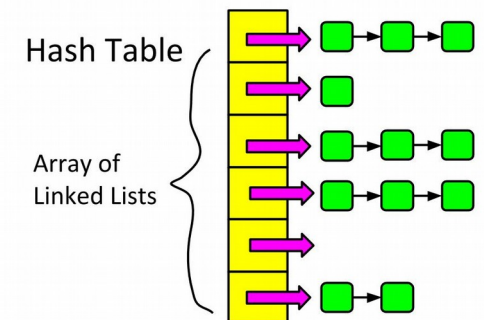
Tabela de dispersão aberta

Exemplo inserção (7)



$$\text{Dispersão}(k) = k \% 11$$

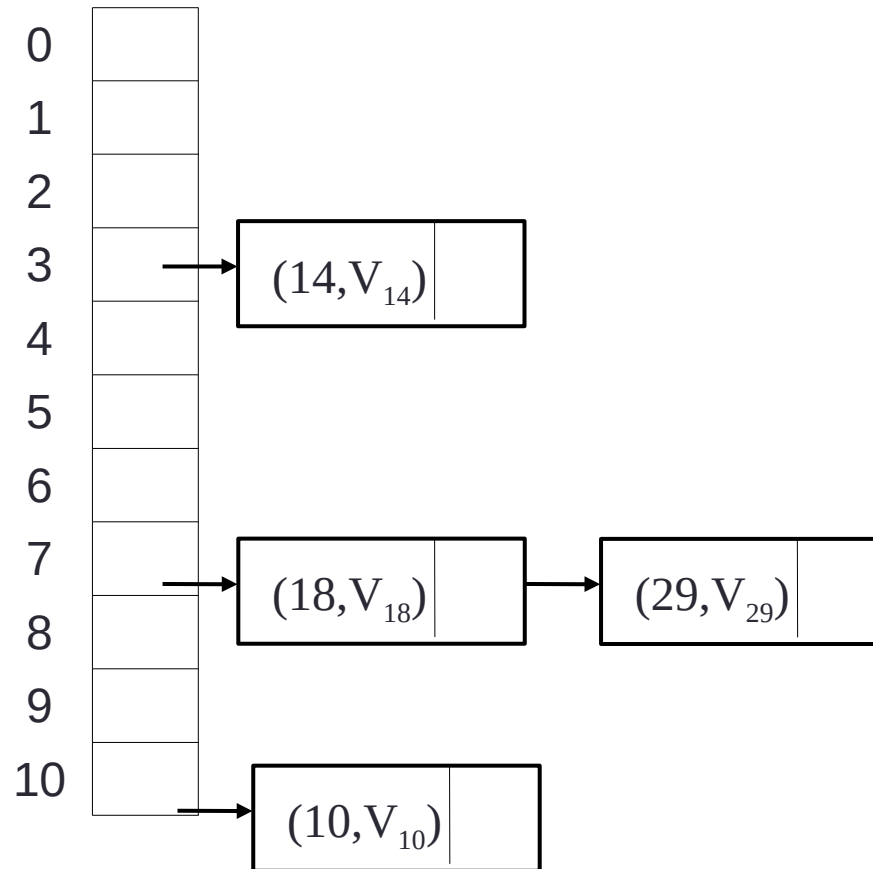
1. Inserir (14, V₁₄)
 $\text{Dispersão}(14) = 3$
2. Inserir (29, V₂₉)
 $\text{Dispersão}(29) = 7$
3. Inserir (10, V₁₀)
 $\text{Dispersão}(10) = 10$
4. Inserir (18, V₁₈)
 $\text{Dispersão}(18) = 7$



TAD Map<K,V>

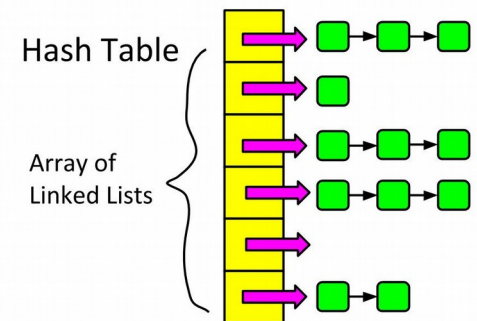
Tabela de dispersão aberta

Exemplo inserção (8)



$$\text{Dispersão}(k) = k \% 11$$

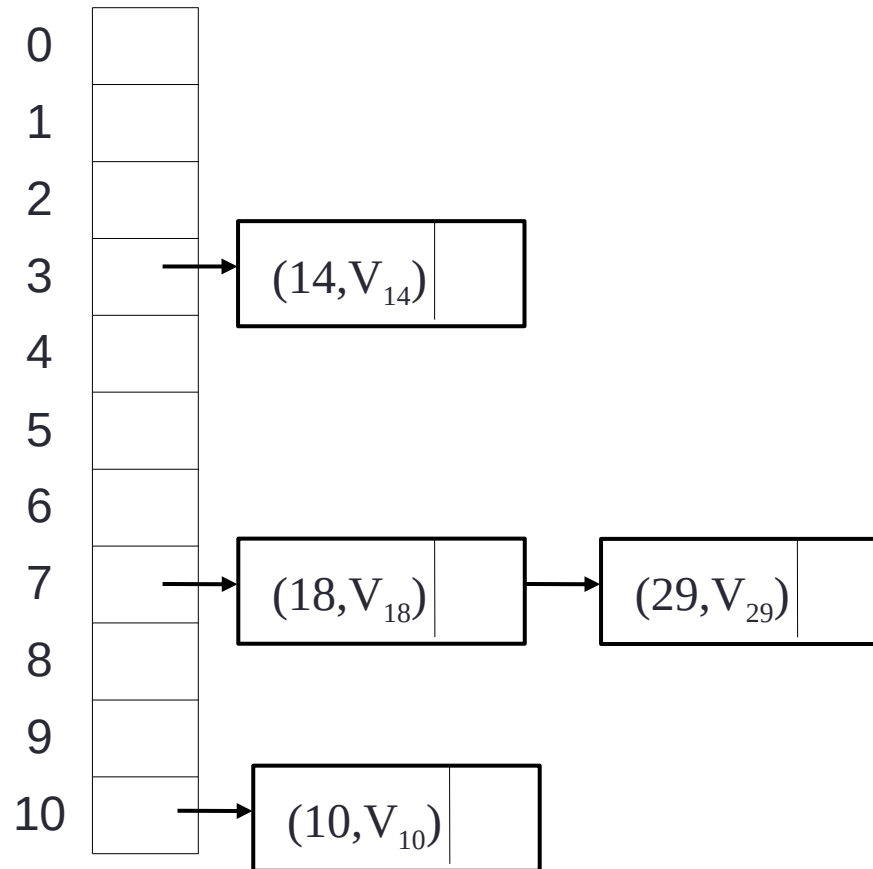
1. Inserir $(14, V_{14})$
 $\text{Dispersão}(14) = 3$
2. Inserir $(29, V_{29})$
 $\text{Dispersão}(29) = 7$
3. Inserir $(10, V_{10})$
 $\text{Dispersão}(10) = 10$
4. Inserir $(18, V_{18})$
 $\text{Dispersão}(18) = 7$



TAD Map<K,V>

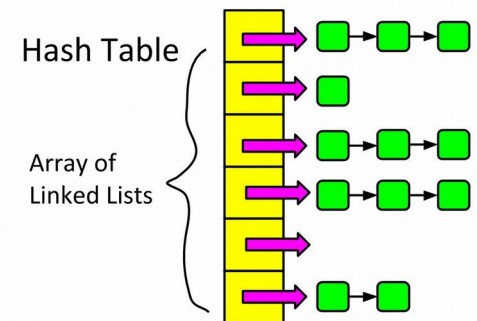
Tabela de dispersão aberta

Exemplo inserção (9)



$$\text{Dispersão}(k) = k \% 11$$

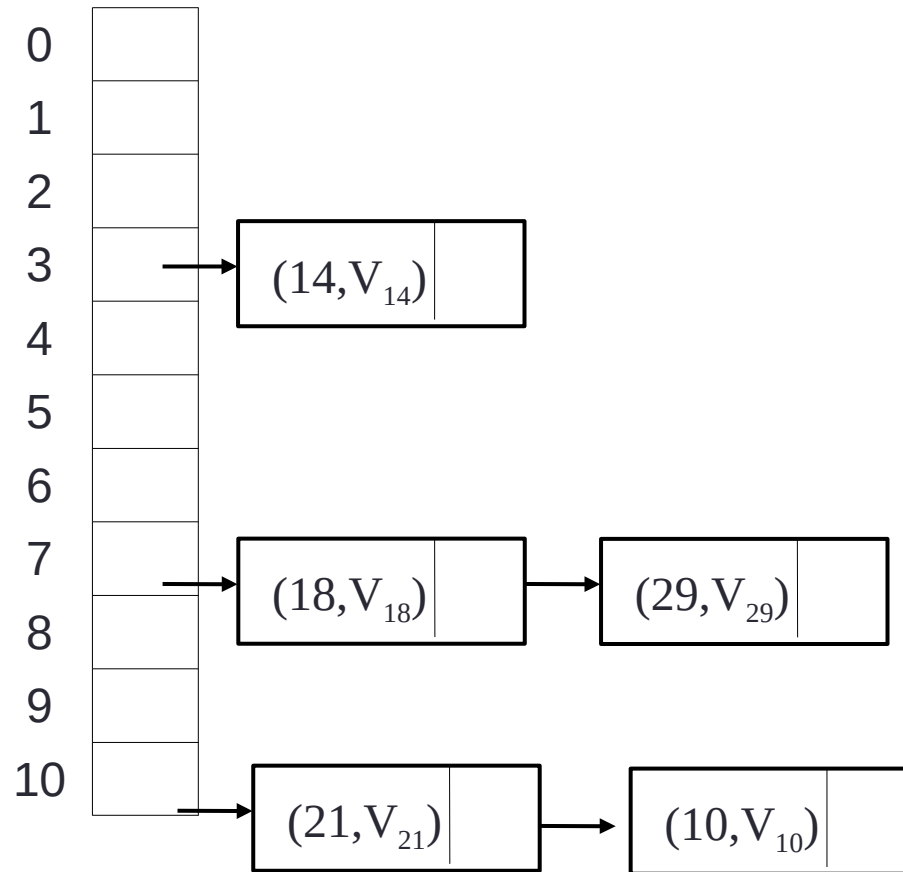
1. Inserir (14, V_{14})
 $\text{Dispersão}(14) = 3$
2. Inserir (29, V_{29})
 $\text{Dispersão}(29) = 7$
3. Inserir (10, V_{10})
 $\text{Dispersão}(10) = 10$
4. Inserir (18, V_{18})
 $\text{Dispersão}(18) = 7$
5. Inserir (21, V_{21})
 $\text{Dispersão}(21) = 10$



TAD Map<K,V>

Tabela de dispersão aberta

Exemplo inserção (10)



$$\text{Dispersão}(k) = k \% 11$$

1. Inserir (14, V_{14})

$$\text{Dispersão}(14) = 3$$

2. Inserir (29, V_{29})

$$\text{Dispersão}(29) = 7$$

3. Inserir (10, V_{10})

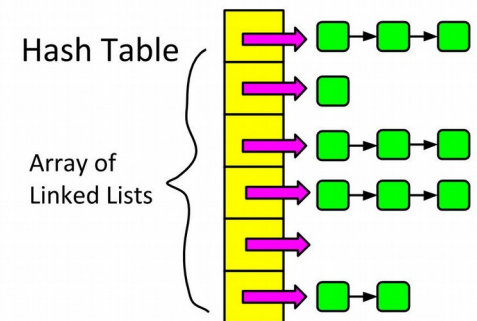
$$\text{Dispersão}(10) = 10$$

4. Inserir (18, V_{18})

$$\text{Dispersão}(18) = 7$$

1. Inserir (21, V_{21})

$$\text{Dispersão}(21) = 10$$



TAD Map<K,V>

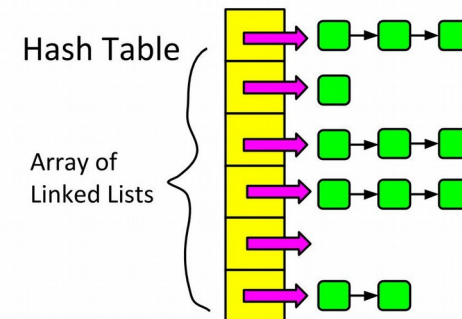
Classe Tabela de dispersão aberta (3)

```
// If there is an entry in the map whose key is the specified key,  
// replaces its value by the specified value and returns the old value;  
// otherwise, inserts the entry (key, value) and returns null.
```

```
@Override
```

```
public V insert( K key, V value ){  
    if ( this.isFull() )  
        this.rehash();  
    V valueOld=table[this.hash(key)].insert(key, value);  
    if (valueOld == null)  
        currentSize++;  
    return valueOld;  
}
```

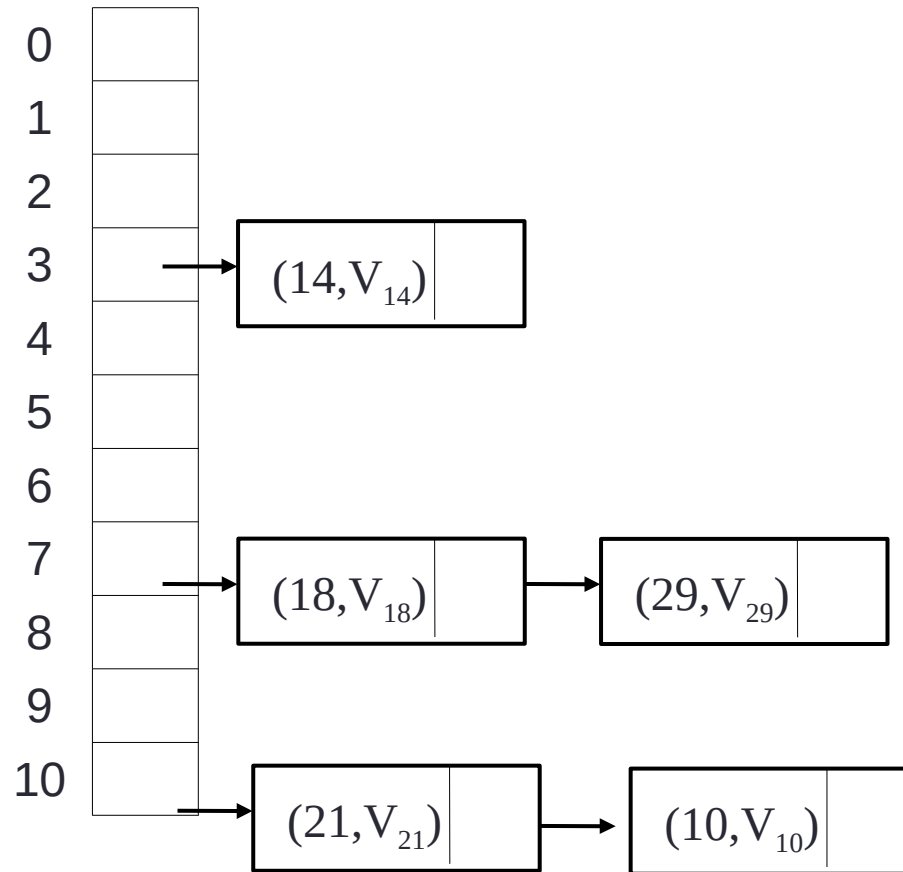
```
private void rehash() {  
    //TODO  
}
```



TAD Map<K,V>

Tabela de dispersão aberta

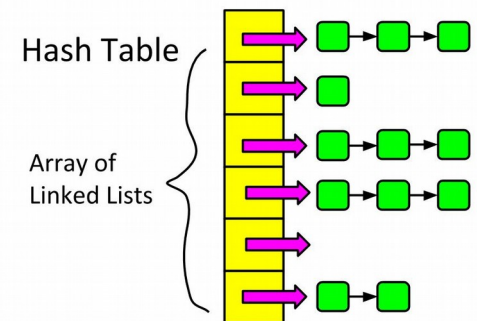
Exemplo remoção (1)



1. Remover (14)

$$\text{Dispersão}(14) = 3$$

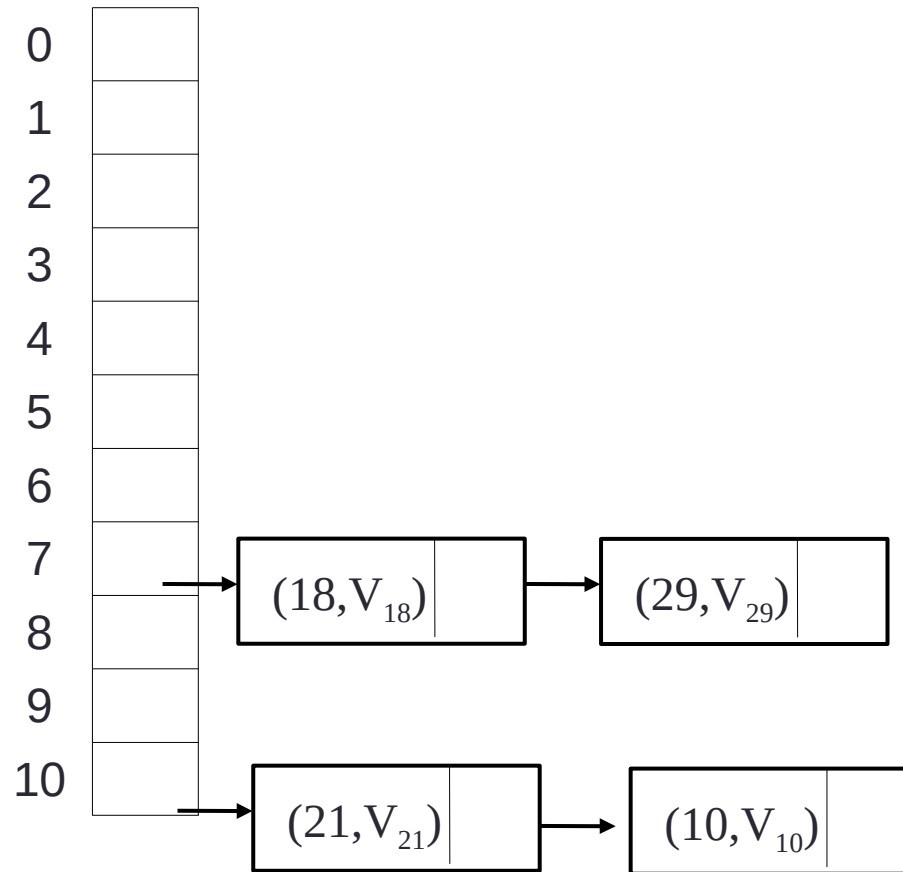
$$\text{Dispersão}(k) = k \% 11$$



TAD Map<K,V>

Tabela de dispersão aberta

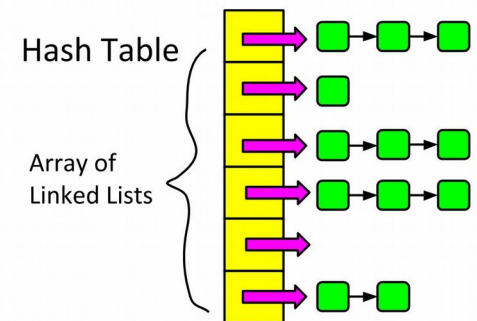
Exemplo remoção (2)



1. Remover (14)

$$\text{Dispersão}(14) = 3$$

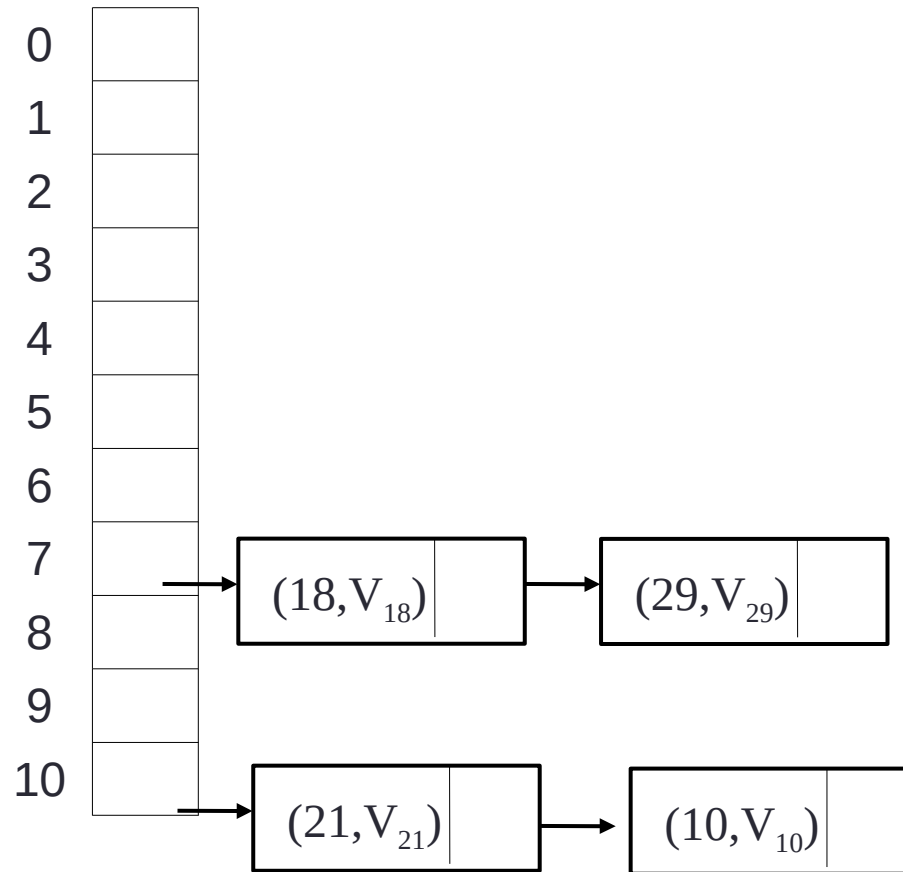
$$\text{Dispersão}(k) = k \% 11$$



TAD Map<K,V>

Tabela de dispersão aberta

Exemplo remoção (3)



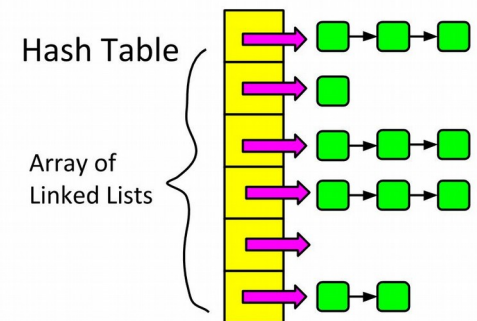
$$\text{Dispersão}(k) = k \% 11$$

1. Remover (14)

$$\text{Dispersão}(14) = 3$$

2. Remover (10)

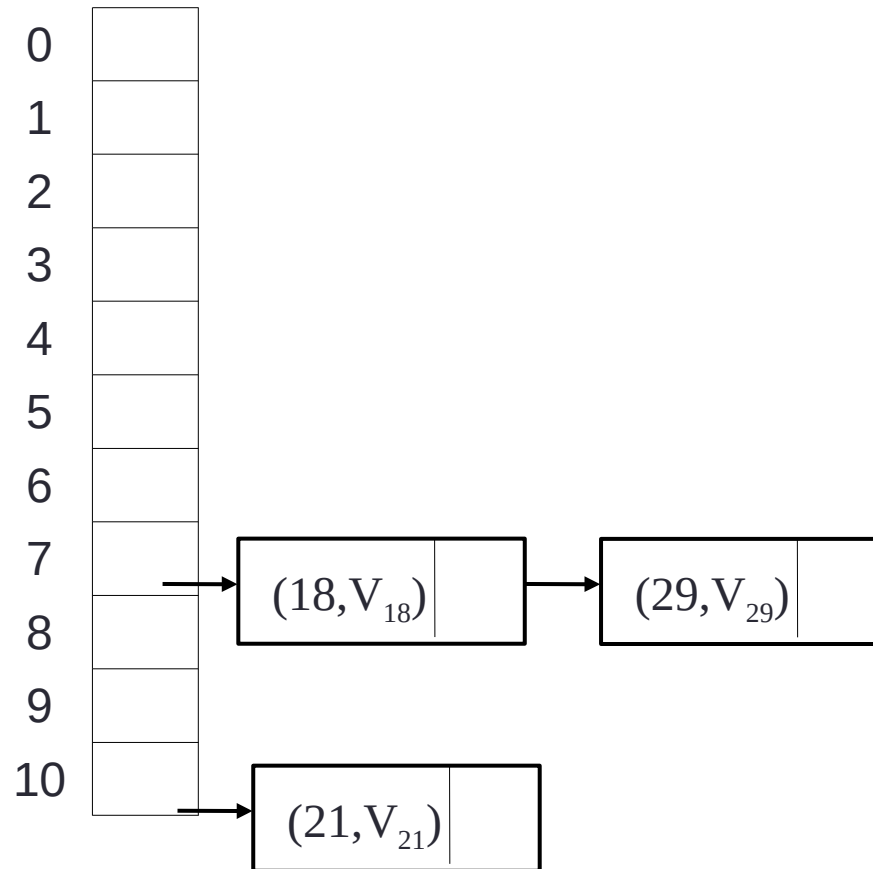
$$\text{Dispersão}(10) = 10$$



TAD Map<K,V>

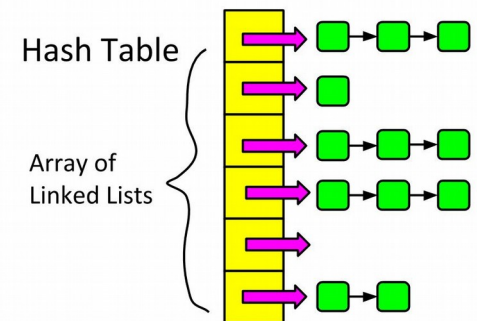
Tabela de dispersão aberta

Exemplo remoção (4)



$$\text{Dispersão}(k) = k \% 11$$

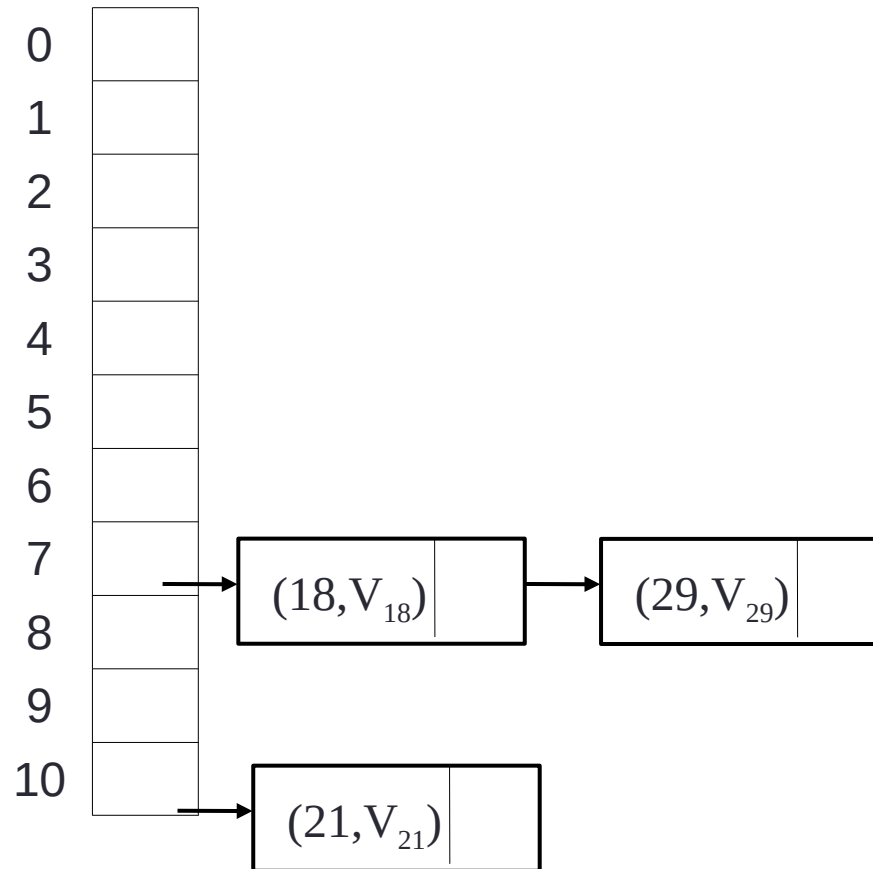
1. Remover (14)
 $\text{Dispersão}(14) = 3$
2. Remover (10)
 $\text{Dispersão}(10) = 10$



TAD Map<K,V>

Tabela de dispersão aberta

Exemplo remoção (5)



$$\text{Dispersão}(k) = k \% 11$$

1. Remover (14)

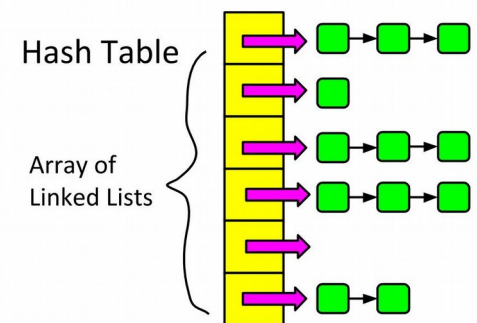
$$\text{Dispersão}(14) = 3$$

2. Remover (10)

$$\text{Dispersão}(10) = 10$$

3. Remover (18)

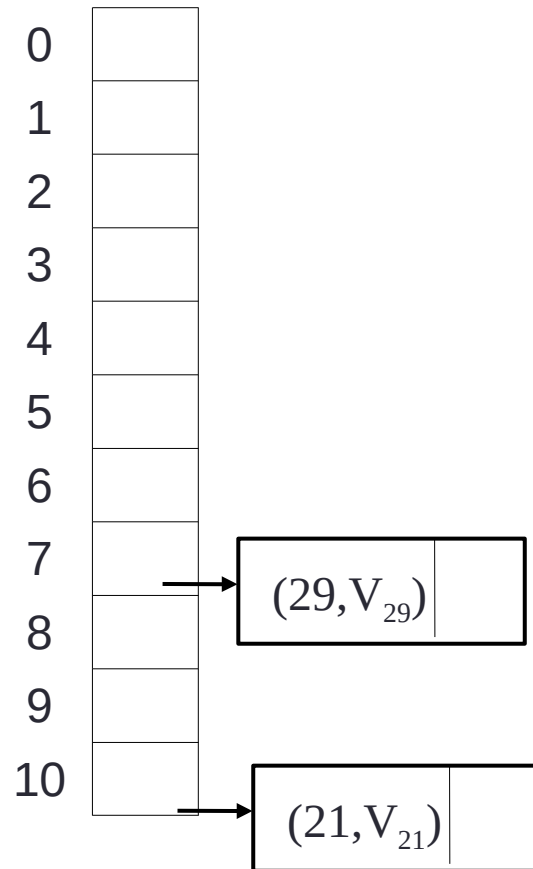
$$\text{Dispersão}(18) = 7$$



TAD Map<K,V>

Tabela de dispersão aberta

Exemplo remoção (6)



$$\text{Dispersão}(k) = k \% 11$$

1. Remover (14)

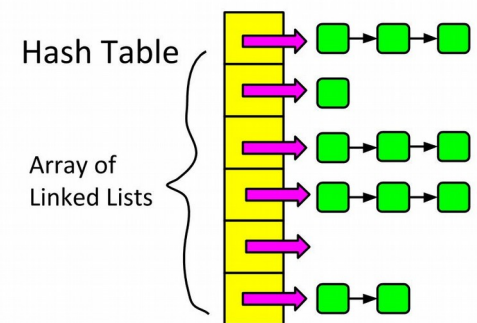
$$\text{Dispersão}(14) = 3$$

2. Remover (10)

$$\text{Dispersão}(10) = 10$$

3. Remover (18)

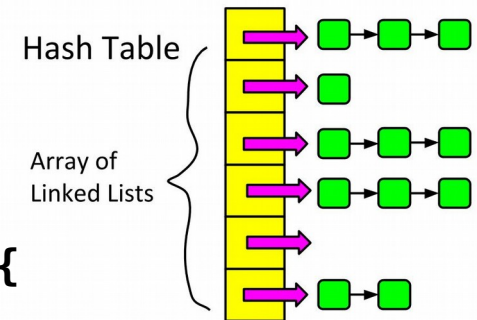
$$\text{Dispersão}(18) = 7$$



TAD Map<K,V>

Classe Tabela de dispersão aberta (4)

```
@Override
public V remove(K key) {
    //TODO
}
@Override
public Iterator<K> keys() throws NoSuchElementException {
    if (isEmpty())
        throw new NoSuchElementException("Map is empty.");
    //TODO
}
@Override
public Iterator<V> values() throws NoSuchElementException {
    if (isEmpty())
        throw new NoSuchElementException("Map is empty.");
    //TODO
}
@Override
public Iterator<Entry<K,V>> iterator() throws NoSuchElementException {
    if (isEmpty())
        throw new NoSuchElementException("Map is empty.");
    //TODO
}
```

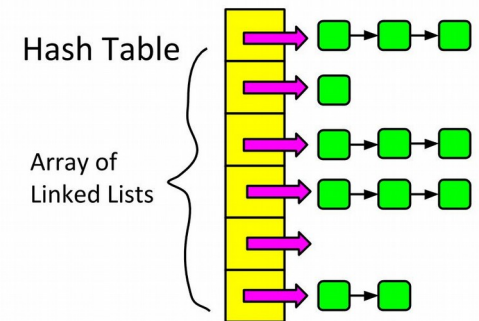


TAD $\text{Map}\langle K, V \rangle$

com Tabela de dispersão aberta (5)

- Será uma boa implementação?

Operação Com - Sem sucesso	Caso Médio	Pior Caso
Inserção	$O(1 + \lambda)$	$O(n)$
Remoção	$O(1 + \lambda)$	$O(n)$
Pesquisa	$O(1 + \lambda)$	$O(n)$
Percurso	$O(n)$	$O(n)$



Sendo n o número de elementos; e λ o factor de ocupação



TAD Map<K,V>

Classe Tabela de dispersão fechada (1)

```
public class LinearProbingHashTable<K, V> extends MapWithHashTable<K,V>{

    // The array of entries.
    protected Entry<K,V>[] table;

    public LinearProbingHashTable(){
        this(DEFAULTCAPACITY);
    }

    @SuppressWarnings("unchecked")
    public LinearProbingHashTable(int capacity) {
        // Load factor is 1/2 (0.5)
        int arraySize = MapWithHashTable.nextPrime((int) (2 * capacity));
        // Compiler gives a warning.
        table = (Entry<K,V>[]) new Entry[arraySize];
        for ( int i = 0; i < arraySize; i++ )
            table[i] = null;
        maxSize = capacity;
        currentSize = 0;
    }
}
```

TAD Map<K,V>

Tabela de dispersão fechada

Exemplo inserção (1)

Sondagem linear

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

1. Inserir $(14, V_{14})$

Dispersão(14) = 3

$$\text{Dispersão}(k) = k \% 11$$

TAD Map<K,V>

Tabela de dispersão fechada

Exemplo inserção (2)

Sondagem linear

0	
1	
2	
3	(14, V ₁₄)
4	
5	
6	
7	
8	
9	
10	

1. Inserir (14, V₁₄)

Dispersão(14) = 3

$$\text{Dispersão}(k) = k \% 11$$

TAD Map<K,V>

Tabela de dispersão fechada

Exemplo inserção (3)

Sondagem linear

0	
1	
2	
3	(14, V ₁₄)
4	
5	
6	
7	
8	
9	
10	

1. Inserir (14, V₁₄)

Dispersão(14) = 3

2. Inserir (5, V₅)

Dispersão(5) = 5

$$\text{Dispersão}(k) = k \% 11$$

TAD Map<K,V>

Tabela de dispersão fechada

Exemplo inserção (4)

Sondagem linear

0	
1	
2	
3	(14, V ₁₄)
4	
5	(5, V ₅)
6	
7	
8	
9	
10	

1. Inserir (14, V₁₄)

Dispersão(14) = 3

2. Inserir (5, V₅)

Dispersão(5) = 5

$$\text{Dispersão}(k) = k \% 11$$

TAD Map<K,V>

Tabela de dispersão fechada

Exemplo inserção (5)

Sondagem linear

0	
1	
2	
3	(14, V ₁₄)
4	
5	(5, V ₅)
6	
7	
8	
9	
10	

1. Inserir (14, V₁₄)

Dispersão(14) = 3

2. Inserir (5, V₅)

Dispersão(5) = 5

3. Inserir (3, V₃)

Dispersão(3) = 3

$$\text{Dispersão}(k) = k \% 11$$

TAD Map<K,V>

Tabela de dispersão fechada

Exemplo inserção (6)

Sondagem linear

0	
1	
2	
3	(14, V ₁₄)
4	(3, V ₃)
5	(5, V ₅)
6	
7	
8	
9	
10	

1. Inserir (14, V₁₄)

Dispersão(14) = 3

2. Inserir (5, V₅)

Dispersão(5) = 5

3. Inserir (3, V₃)

Dispersão(3) = 3

$$\text{Dispersão}(k) = k \% 11$$

TAD Map<K,V>

Tabela de dispersão fechada

Exemplo inserção (7)

Sondagem linear

0	
1	
2	
3	(14, V ₁₄)
4	(3, V ₃)
5	(5, V ₅)
6	
7	
8	
9	
10	

1. Inserir (14, V₁₄)

Dispersão(14) = 3

2. Inserir (5, V₅)

Dispersão(5) = 5

3. Inserir (3, V₃)

Dispersão(3) = 3

4. Inserir (15, V₁₅)

Dispersão(15) = 4

$$\text{Dispersão}(k) = k \% 11$$

TAD Map<K,V>

Tabela de dispersão fechada

Exemplo inserção (8)

Sondagem linear

0	
1	
2	
3	(14, V ₁₄)
4	(3, V ₃)
5	(5, V ₅)
6	(15, V ₁₅)
7	
8	
9	
10	

1. Inserir (14, V₁₄)

Dispersão(14) = 3

2. Inserir (5, V₅)

Dispersão(5) = 5

3. Inserir (3, V₃)

Dispersão(3) = 3

4. Inserir (15, V₁₅)

Dispersão(15) = 4

$$\text{Dispersão}(k) = k \% 11$$

TAD Map<K,V>

Tabela de dispersão fechada

Exemplo pesquisa (sem remoções) (1)

0	
1	
2	
3	(14, V ₁₄)
4	(3, V ₃)
5	(5, V ₅)
6	(15, V ₁₅)
7	
8	
9	
10	

Sondagem linear

1. Pesquisa (14)

$$\text{Dispersão}(14) = 3$$

2. Pesquisa (3)

$$\text{Dispersão}(3) = 3$$

3. Pesquisa (15)

$$\text{Dispersão}(15) = 4$$

4. Pesquisa (5)

$$\text{Dispersão}(5) = 5$$

Procura até conseguir:

- O elemento, ou
- Uma posição vazia

$$\text{Dispersão}(k) = k \% 11$$

TAD Map<K,V>

Classe Tabela de dispersão fechada (2)

@Override

```
public V find(K key) {
    int pos=findPos(key);
    if (pos == -1 || isEmpty(pos))return null;
    return table[pos].getValue();
}
// returns the position (a) where is the element with this key
// or (b) of an empty position
// or (c) -1 the map is full and there is not element with this key
private int findPos(K key) {
    int pos = this.hash(key);
    int hashKey=pos;
    if (!isEmpty(pos) && table[pos].getKey().equals(key) )
        return pos;
    pos = ( pos + 1 ) % table.length;
    while (hashKey!=pos && !isEmpty(pos) &&
           !table[pos].getKey().equals(key) )
        pos = ( pos + 1 ) % table.length;
    if (hashKey==pos) return -1;
    return pos;
}
```

TAD Map<K,V>

Classe Tabela de dispersão fechada (3)

```
//without remove
private boolean isEmpty(int pos) {
    return table[pos]==null;
}

@Override
//without remove
public V insert(K key, V value) {
    if ( this.isFull() ) this.rehash();
    int pos=findPos(key); // nunca será -1
    V valueOld;
    if (!isEmpty(pos))
        valueOld= table[pos].getValue();
    else {
        valueOld=null;
        currentSize++;
    }
    table[pos] = new EntryClass<K,V>(key,value);
    return valueOld;
}
```

TAD Map<K,V>

Tabela de dispersão fechada

Exemplo remoção (1)

Sondagem linear

0	
1	
2	
3	(14, V ₁₄)
4	(3, V ₃)
5	(5, V ₅)
6	(15, V ₁₅)
7	
8	
9	
10	

1. Remover (3)

Dispersão(3) = 3

$$\text{Dispersão}(k) = k \% 11$$

TAD Map<K,V>

Tabela de dispersão fechada

Exemplo remoção (2)

Sondagem linear

0	
1	
2	
3	(14, V ₁₄)
4	
5	(5, V ₅)
6	(15, V ₁₅)
7	
8	
9	
10	

1. Remover (3)

Dispersão(3) = 3

$$\text{Dispersão}(k) = k \% 11$$

TAD Map<K,V>

Tabela de dispersão fechada

Exemplo remoção (3)

Sondagem linear

0	
1	
2	
3	(14, V ₁₄)
4	
5	(5, V ₅)
6	(15, V ₁₅)
7	
8	
9	
10	

1. Remover (3)

Dispersão(3) = 3

2. Remover (15)

Dispersão(15) = 4

Como executar Pesquisa K?

$$\text{Dispersão}(k) = k \% 11$$

TAD Map<K,V>

Tabela de dispersão fechada

Exemplo remoção (3)

Sondagem linear

0	
1	
2	
3	(14, V ₁₄)
4	REMOVIDO
5	(5, V ₅)
6	(15, V ₁₅)
7	
8	
9	
10	

1. Remover (3)

Dispersão(3) = 3

2. Remover (15)

Dispersão(15) = 4

Como executar Pesquisa K?

- As remoções tem que ser marcadas.

$$\text{Dispersão}(k) = k \% 11$$

TAD Map<K,V>

Tabela de dispersão fechada

Exemplo remoção (4)

Sondagem linear

0	
1	
2	
3	(14, V ₁₄)
4	REMOVIDO
5	(5, V ₅)
6	REMOVIDO
7	
8	
9	
10	

1. Remover (3)

$\text{Dispersão}(3) = 3$

2. Remover (15)

$\text{Dispersão}(15) = 4$

Na **pesquisa K** procura-se até encontrar o elemento ou chegar a uma posição vazia (com null).

E as posições com o estado REMOVIDO, quando serão outra vez ocupadas?

- Na inserção

$$\text{Dispersão}(k) = k \% 11$$

TAD Map<K,V>

Tabela de dispersão fechada

Exemplo inserção (1)

Sondagem linear

0	
1	
2	
3	(14, V ₁₄)
4	REMOVIDO
5	(5, V ₅)
6	REMOVIDO
7	
8	
9	
10	

1. Inserir (15, V₁₅)

Dispersão(15) = 4

Na pesquisa procura-se da posição 4 até à posição 7.

Inserir na posição removida (posição 4).

$$\text{Dispersão}(k) = k \% 11$$

TAD Map<K,V>

Tabela de dispersão fechada

Exemplo inserção (2)

Sondagem linear

0	
1	
2	
3	(14, V ₁₄)
4	(15, V ₁₅)
5	(5, V ₅)
6	REMOVIDO
7	
8	
9	
10	

1. Inserir (15, V₁₅)

Dispersão(15) = 4

Na pesquisa procura-se da posição 4 até à posição 7.

Inserir na posição removida (posição 4).

$$\text{Dispersão}(k) = k \% 11$$

TAD Map<K,V>

Classe Tabela de dispersão fechada (4)

```
//with remove
private boolean isEmpty(int pos) {
    return table[pos]==null || table[pos].getKey()==null;
}

public V insert(K key, V value) {
    //TODO
    return null;
}
```

TAD Map<K,V>

Classe Tabela de dispersão fechada (5)

//with remove

```
public V insert(K key, V value) {  
    //TODO  
    return null;  
}
```

Inserir k, v : envolve dois ciclos de pesquisa antes de fazer a inserção

O primeiro ciclo termina quando

se encontra k , uma posição **Vazia** (null) ou uma posição **Removida** (key==null)

K: Substituir valor v nessa posição

Vazia: Insere-se k, v nessa posição


Removida: Guarda-se essa posição (posição-se-inserir) e executa-se o segundo ciclo

O segundo ciclo termina quando

se encontra K ou uma posição **Vazia**

K: Substituir valor v nessa posição

Vazia: Insere-se k, v em posição-se-inserir



Dispersão aberta vs Dispersão fechada

- Vantagens da dispersão aberta
 - Suporta a operação de remoção.
 - Os diferentes conjuntos de chaves que colidem (as “listas de colisões”) não se misturam.
- Vantagens da dispersão fechada
 - Gasta menos memória (alguns bytes por entrada).
- Desvantagens da dispersão
 - Não é uma estrutura totalmente dinâmica.
 - Não suporta operações que se baseiam na relação entre chaves.

Diagrama de classes (interface **Map**)

