

# ENGENHARIA INFORMÁTICA E DE COMPUTADORES

## Algoritmos e Estruturas de Dados

(parte 12 – Tabelas de Dispersão)

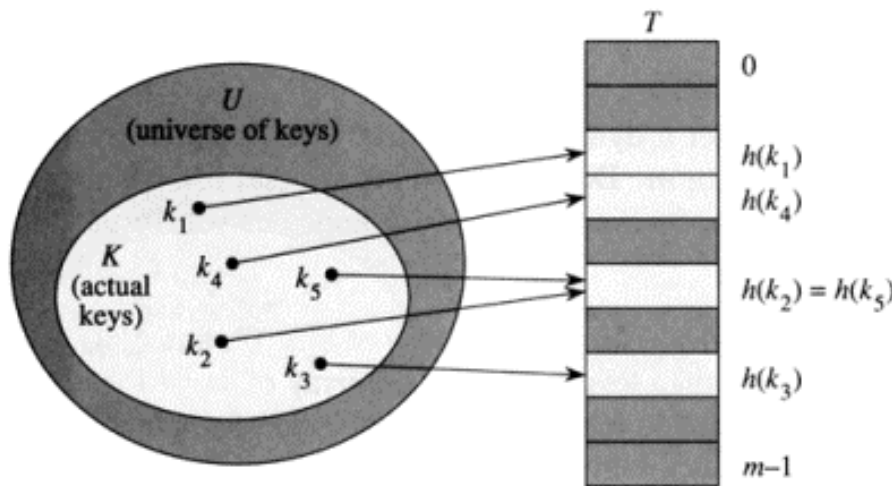
2º Semestre 2022/2023

Instituto Superior de Engenharia de Lisboa

Paula Graça

# TABELAS DE ENDEREÇAMENTO DIRETO

- O **endereçoamento direto** é uma técnica que funciona bem quando o universo de chaves é relativamente pequeno, ou são números sequenciais seguidos ( $0..m-1$ )
    - O conjunto é representado através de uma tabela (**direct-address table**)  $T[0..m-1]$ , em que cada posição corresponde a uma chave do universo  $U$
    - Se não existir a chave  $k$ , então  $T[k] = \text{NULL}$
- Tempo de execução  $O(1)$



Direct-Address-Search( $T, k$ )  
return  $T[k]$

Direct-Address-Add( $T, x$ )  
 $T[x.key] = x$

Direct-Address-Remove( $T, x$ )  
 $T[x.key] = \text{NULL}$

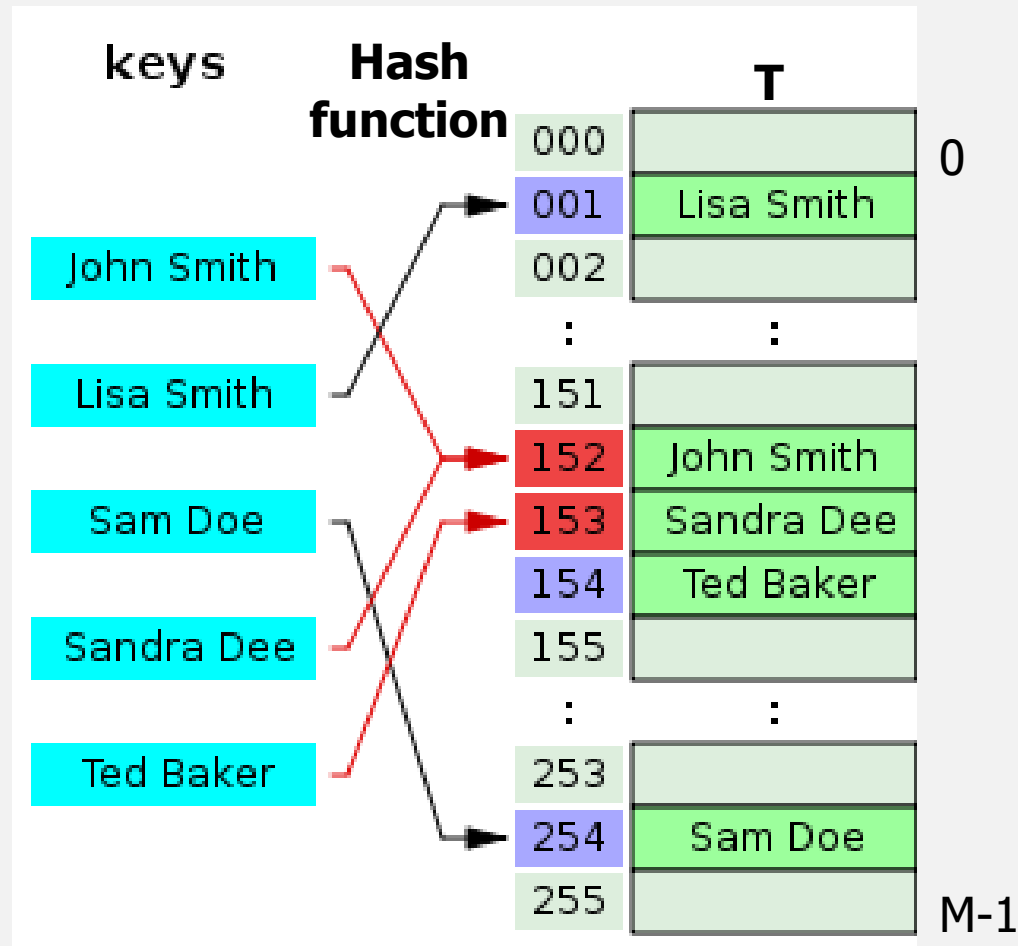
# TABELAS DE ENDEREÇAMENTO DIRETO

- Se o universo  $U$  é muito grande, armazenar numa tabela  $T$  de dimensão  $|U|$  é impraticável
- O conjunto  $K$  de chaves armazenado pode ser tão pequeno relativamente à dimensão do universo  $U$  que a maior parte do espaço alocado para  $T$  é desperdiçado
- Alternativa: **Tabelas de Dispersão**
  - Pesquisa de um elemento no pior caso pode ser  $O(N)$
  - Assumindo determinadas condições no dimensionamento da tabela, o tempo expectável para encontrar um elemento é  $O(1)$

# TABELAS DE DISPERSÃO

- Dispersão (*hashing*) é baseada na ideia de distribuição de chaves por um *array* unidimensional  $T[0..M-1]$  designado por *tabela de dispersão* (*hash table*)
- A distribuição é feita calculando para cada uma das chaves, o valor de uma função pré-definida  $h$  designada *função de dispersão* (*hash function*)
- Esta função atribui a cada chave, um valor inteiro entre  $0$  e  $M-1$  chamado *endereço de dispersão* (*hash address*)

# TABELAS DE DISPERSÃO



# FUNÇÃO DE DISPERSÃO

- Em geral, uma **função de dispersão** deve obedecer aos seguintes requisitos:
  - Transforma a chave num inteiro  **$[0..M-1]$** , sendo **M** a dimensão da tabela de dispersão
  - Deve distribuir o melhor possível as chaves pelas posições livres da tabela de dispersão (por esta razão o valor de **M** escolhido, é usualmente primo)
  - Deve ser de cálculo simples
- Função de dispersão ideal
  - A probabilidade de ocorrer o mesmo valor para duas chaves distintas é de  **$1/M$**

# FUNÇÃO DE DISPERSÃO

- As **funções de dispersão** podem ser da forma:
  1. Se as chaves (K) são números inteiros (se forem negativos passam-se a positivos), então,

$$h(K) = K \% M$$

2. Se as chaves são *strings* ( $K = c_0c_1...c_{s-1}$ ), então,

$$h(K) = ( \sum_{i=0}^{s-1} \text{Unicode}(c_i) ) \% M$$

# ADICIONAR ELEMENTOS

- Para adicionar elementos a uma tabela de dispersão, coloca-se cada uma das chaves numa posição calculada em função do seu valor (através da **função de dispersão**), independentemente da ordem de chegada desses elementos

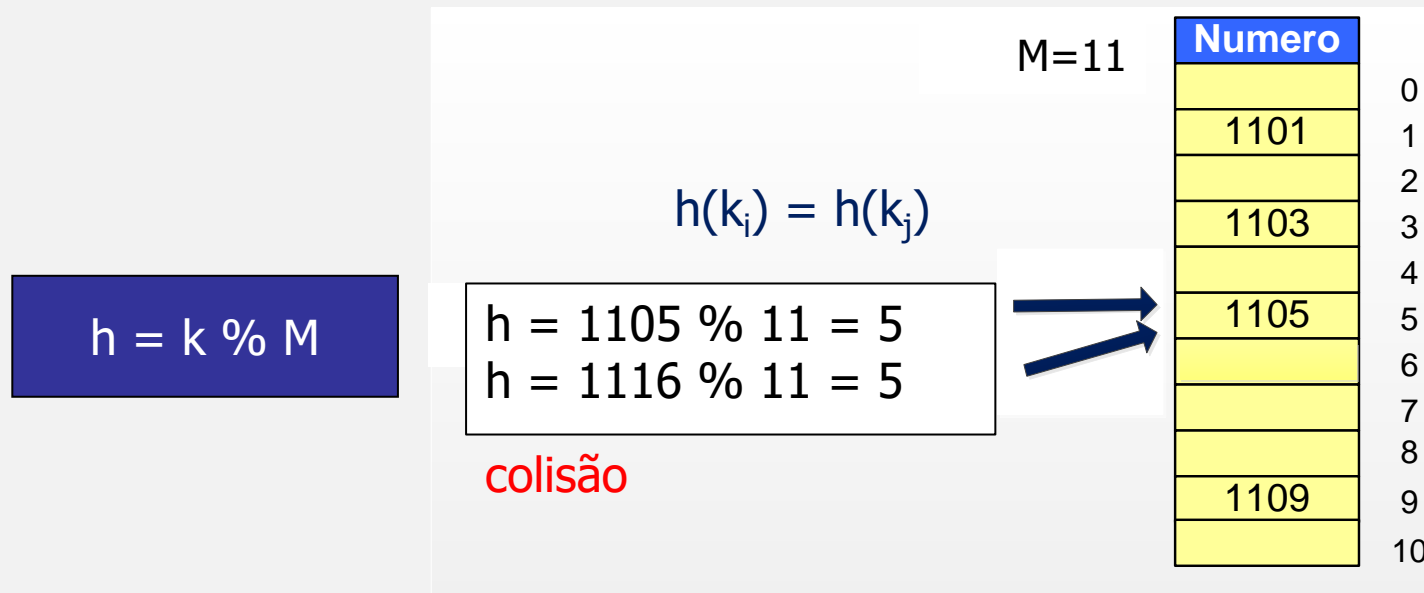
$$h = \text{key} \% M$$

M=11		
$h = 1101 \% 11 = 1$	➡	1101
$h = 1103 \% 11 = 3$	➡	1103
$h = 1105 \% 11 = 5$	➡	1105



# COLISÕES

- A função de dispersão pode devolver o mesmo valor para duas ou mais chaves distintas, o que se designa por **colisão**
- No pior caso, todas as chaves podem colidir no mesmo endereço da tabela de dispersão
- Com a escolha de uma dimensão adequada da tabela e uma boa função de dispersão, as colisões ocorrem raramente



# RESOLUÇÃO DE COLISÕES

- Existem vários métodos de resolução de colisões
  - *Separate Chaining* (Encadeamento Externo)
    - Todos os elementos que colidem no mesmo endereço da tabela de dispersão, são colocados na mesma lista ligada
    - A posição endereçada na tabela, fica com o apontador para a cabeça da lista
  - *Open Addressing* (Encadeamento Interno)
    - Todos os elementos são colocados dentro da tabela de dispersão. Deste modo, esta contém em cada posição um elemento válido ou NULL
    - Suporta várias técnicas de arrumar as colisões:
      - *Linear Probing* (Procura Linear)
      - *Quadratic Probing* (Procura Quadrática)
      - *Double Hashing* (Dupla Dispersão)

# ENCADEAMENTO INTERNO

- Uma tabela de dispersão por encadeamento interno nunca deve estar muito cheia, sendo conveniente que a função de dispersão distribua bem as chaves pela tabela, para que as posições livres também fiquem bem distribuídas
- Por muito boa que seja uma função de dispersão, existem sempre colisões
- Quanto maior for a taxa de ocupação da tabela, maior a probabilidade de colisões, assim como será maior o número médio de acessos para encontrar uma chave

# COLISÕES - ENCADEAMENTO INTERNO

- Procura Linear

- Quando a posição dada pela função de dispersão ( $h'$ ) está ocupada, a chave é arrumada na primeira posição livre adjacente

$$h'(k) = k \% M$$

$$h(k, i) = (h'(k) + i) \% M$$

$$h'(1116, 0) = 1116 \% 11 = 5$$

$$h(1116, 1) = (5 + 1) \% 11 = 6$$

$$h(1116, 2) = (5 + 2) \% 11 = 7$$

M=11	
0	
1	1101
2	
3	
4	
5	1105
6	1106
7	1116
8	
9	1109
10	

# COLISÕES - ENCADEAMENTO INTERNO

- Procura Quadrática

- Quando a posição dada pela função de dispersão (**h'**) está ocupada, a chave é arrumada na posição dada por uma função quadrática

$$h(k, i) = (h'(k) + c_1 i + c_2 i^2) \% M$$

Ex:  $c_1 = 0, c_2 = 1$

$$h'(k) = k \% M$$

$$h(k, i) = (h'(k) + i^2) \% M$$

$$\begin{aligned} h'(1112, 0) &= 1112 \% 11 = 1 \\ h(1112, 1) &= (1 + 1) \% 11 = 2 \\ h(1112, 2) &= (1 + 2^2) \% 11 = 5 \\ h(1112, 3) &= (1 + 3^2) \% 11 = 10 \end{aligned}$$

M=11	
0	
1	1101
2	1102
3	1103
4	
5	1105
6	
7	
8	
9	1109
10	1112

# COLISÕES - ENCADEAMENTO INTERNO

- **Dupla Dispersão**

- Quando a posição dada pela primeira função de dispersão está ocupada, calcula-se e aplica-se um deslocamento através da segunda função de dispersão, até encontrar uma posição livre
- $M = 13$  (deve ser primo mas próximo de  $M$ )
- $M' = 11$  (deve ser menor mas próximo de  $M$ )

$$h_1 = k \% M$$
$$h_2 = 1 + (k \% M')$$

$$h_1(14) = 14 \% 13 = 1$$
$$h_2(14) = 1 + (14 \% 11) = 4$$

14

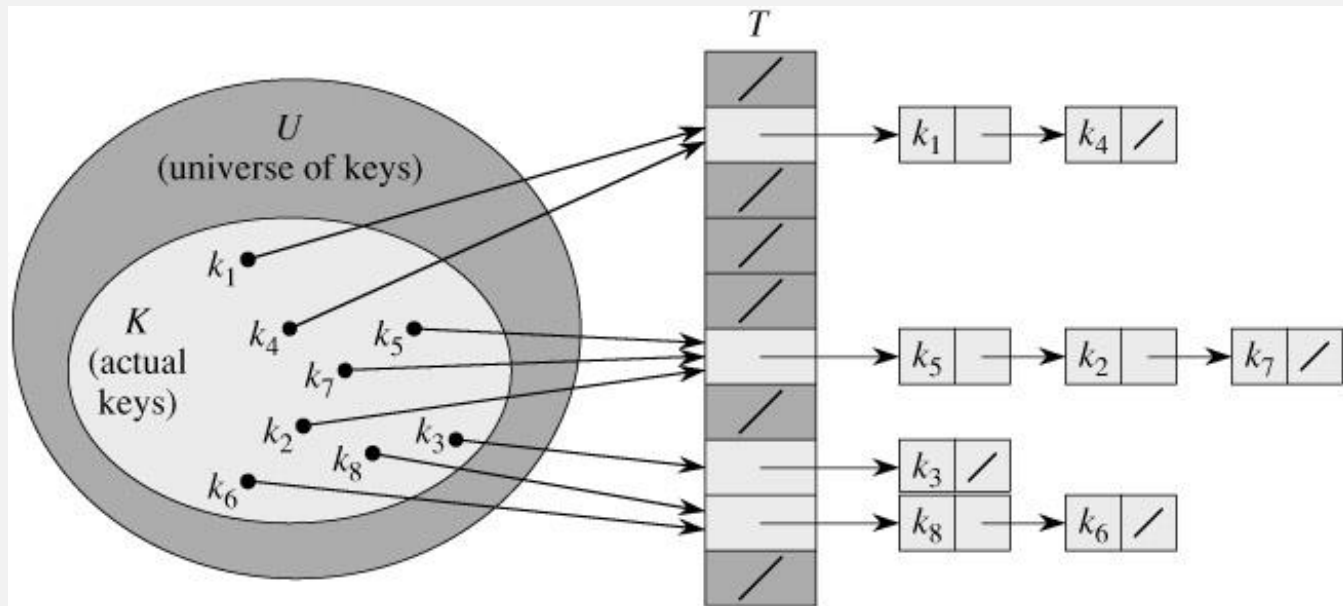
M=13	
0	
1	79
2	
3	
4	69
5	98
6	
7	72
8	
9	14
10	
11	50
12	

4

4

# COLISÕES – ENCADEAMENTO EXTERNO

- Encadeamento Externo (*Separate Chaining*)
  - Todos os itens que colidem na mesma posição, são arrumados numa lista ligada, ficando a cabeça da lista na posição dada pela função de dispersão



# OPERAÇÕES – ENCADEAMENTO EXTERNO

- Operações numa tabela de dispersão **T**, onde as colisões são resolvidas por encadeamento externo
  - Tempo de execução  **$O(1)$**

Chained-Hash-Add( $T, x$ )  
insert  $x$  at the head of list  $T[h(x.key)]$

- Tempo de execução proporcional à dimensão da lista


Chained-Hash-Contains( $T, k$ )  
search for an element with key  $k$  in list  $T[h(k)]$


- Tempo de execução proporcional à dimensão da lista

Chained-Hash-Remove( $T, x$ )  
delete  $x$  from the list  $T[h(x.key)]$



# ANÁLISE – ENCADEAMENTO EXTERNO

- **Melhor Caso** 
  - A função de dispersão devolve uma posição  $h(k)$  diferente para cada chave
  - Existe um endereço para cada chave
  - O tempo de execução é  $\Theta(1)$

- **Pior Caso** 
  - A função de dispersão devolve a mesma posição  $h(k)$  para todas as chaves, criando uma lista de comprimento **N**
  - Todas as chaves colidem no mesmo endereço
  - O tempo de execução é  $\Theta(N)$

# ANÁLISE – ENCADEAMENTO EXTERNO

- Caso Médio 😊

- Se a função de dispersão distribui  $N$  chaves por  $M$  posições da tabela, cada lista terá cerca de  $N/M$  chaves, considerando uma **distribuição uniforme simples** por todas as posições
  - $\alpha = N/M$  (fator de carga)
  - Se  $\alpha$  é muito pequeno, significa muito espaço desperdiçado na tabela
  - Se  $\alpha$  é muito grande, significa listas grandes com pesquisa ineficiente
- O tempo de execução é  $\Theta(1 + \alpha)$  (função de *hash* +  $N/M$ )
- Se  $N = O(M)$  então  $\alpha = N/M = O(M)/M = O(1)$
- Fator de carga bom:  $\alpha = N/M = 0.75$