

# Hashing

## Tabela de Dispersão

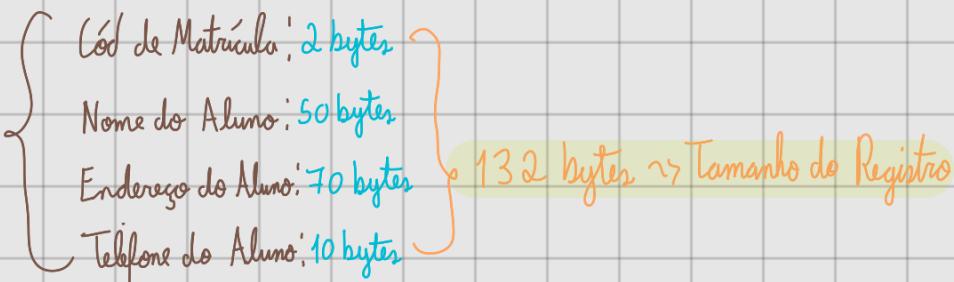
Conhecidas como Tabelas Hash

- Quando bem projetadas, podem ser usadas para se buscar um dado em uma tabela em tempo constante:  $O(1)$
- E por isso, acabam usando mais a memória.
- Implementam os arrays associativos ou dicionários (mapeamentos).

## Visão Geral

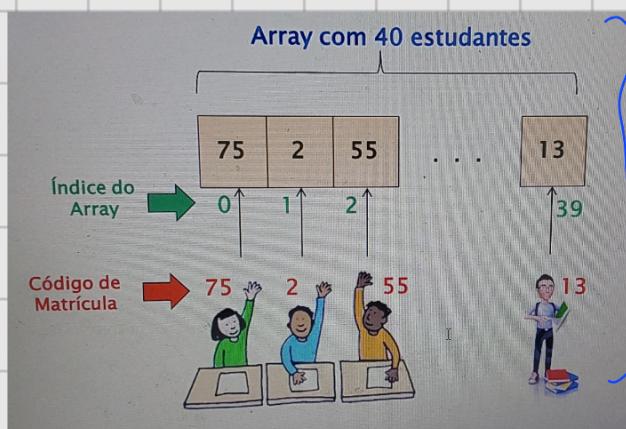
Escola com 80 estudantes, cada um com um código de matrícula de 2 dígitos

O estudante Paulo Alves tem o código de matrícula 55. Suponha que cada estudante tem os seguintes registros armazenados:



Como implementar uma estrutura de dados para armazenar os estudantes que têm a disciplina História, sendo 40 alunos matrículados nessa matéria?

## Usando Array Puro:



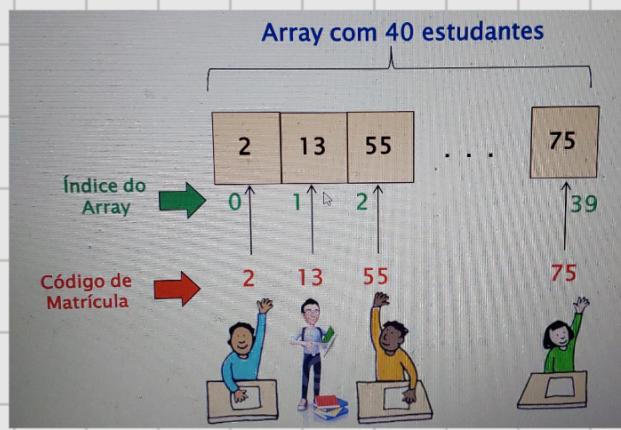
- Tem tamanho exato para alocar 40 estudantes
- Consumo de memória será  $40 \cdot 132 = 5280$  bytes  $\approx 5\text{Kb}$

## \* Operação de Busca no Array Puro

- Não há relacionamento entre o código de matrícula (chave) e o índice do array
- Dados estão desordenados
- Assim, a pesquisa será sequencial e o tempo é proporcional ao tamanho do array  $O(n)$
- Será necessário percorrer todo o array.

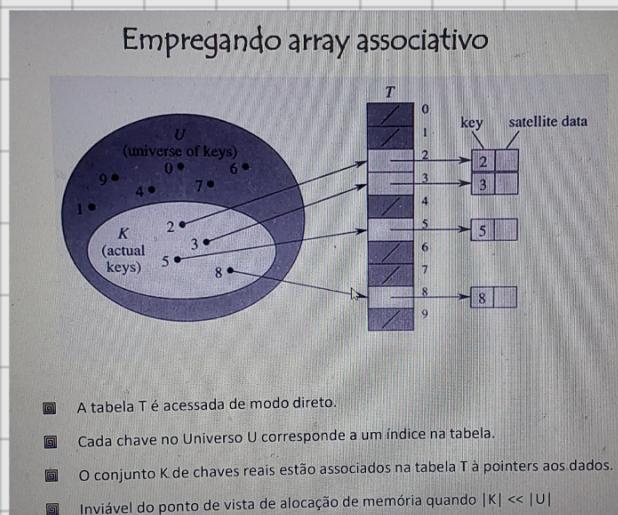
## \* Como Melhorar a eficiência da busca?

- Ordenando o array e fazendo uma busca binária
- Complexidade ainda será  $O(\log n)$



## \* Existe algum meio de se fazer uma busca com tempo constante $O(1)$ ? Acesso Direto ao Dado?

- Cada estudante tem um código de matrícula com dois dígitos, que daria um conjunto universo  $U = \{1, 2, \dots, 99\}$ .
- Podemos criar um array com 100 elementos e associar cada código de estudante ao índice do array.
- Esse array é chamado de array associativo



# Array Associativo

## Vantagens

- Acesso aos dados de forma direta, uma vez que o índice do array coincide com a chave do estudante
- O estudante de chave 55 está na posição do array
- O tempo para acessar o estudante não depende do tamanho do array. Esse tempo é constante e a complexidade é  $O(1)$ .



## Desvantagens

- Somente 40 alunos estão matriculados na disciplina, mas foi alocado na memória para 100 estudantes.
- A melhoria da eficiência da busca tem o preço de maior alocação de memória.

\* Qual o novo valor da memória alocada?

- $100 \cdot 132 = 13200 \text{ bytes} \approx 13 \text{ Kb}$
- Nesse novo modelo o consumo de memória foi de 5 Kb para 13 Kb
- Aumento de memória em 160%

## Quando Usar Arrays Associativos

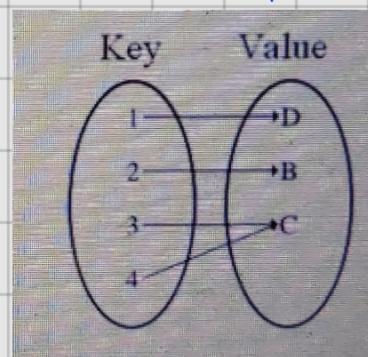
- São implementados por tabelas de acesso direto
- Aplicáveis quando o conjunto universal de chaves  $U$  for pequeno
- Operações de dicionários podem ser efetuadas em tempo constante  $O(1)$

Associative Arrays	
Index Key	Element Value
1	100
2	200
3	300
4	400
5	500
6	600
7	700

\* Como Obter eficiência na busca do dado sem comprometer a alocação de memória?

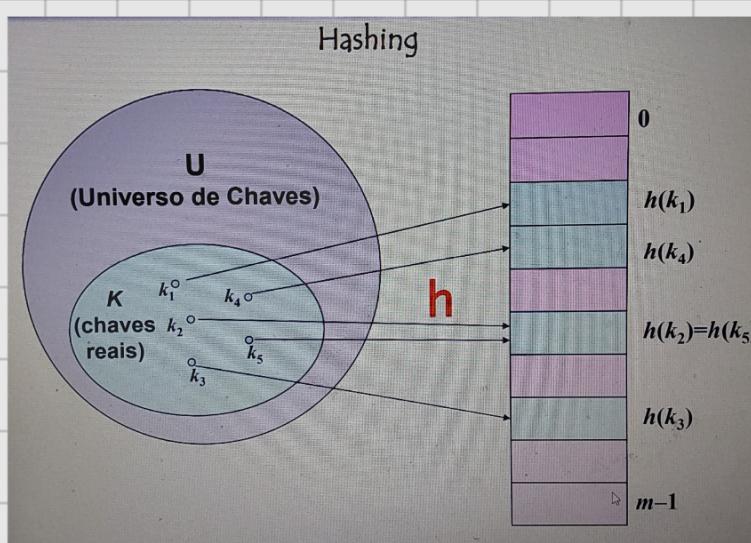
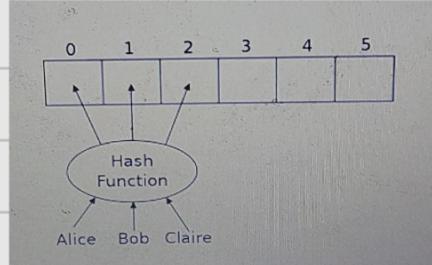
## Estrutura de Dados MAP

- Um mapa é uma estrutura de dados que estabelece uma relação de mapeamento entre dois conjuntos
- Podemos definir um mapa como sendo um conjunto de pares na forma (chave, valor) no qual cada chave está associada a um determinado valor.
- Mapas são também chamados de Estruturas de Dados Dicionário
- A implementação pode ser feita por arrays associativos ou por tabelas hash.



## Hash Tables

- Correspondem a tipos de arrays associativos que são implementados quando  $|K| \ll |U|$ , sendo K o conjunto de chaves válidas e U o conjunto universo de chaves.
- O mapeamento entre K e U é feito por meio de uma função.
- Esta função denomina-se Função de Hash

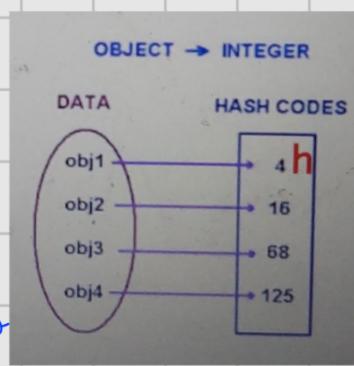


# Hashing

- Função Hash  $h$ : Efetuar o mapeamento de  $U$  para os slots da hash table  $T[0 \dots m-1]$

$$h: U \rightarrow \{0, 1, \dots, m-1\}$$

- Com arrays, a chave  $K$  é mapeada para o slot  $A[K]$ .
- Com hash tables, a chave  $K$  é mapeada para o slot  $T[h(K)]$
- $h(K)$  é o valor hash da chave  $K$ .
- Aplica uma fórmula para calcular um endereço, determinando assim o posicionamento de um dado.
- O cálculo é feito através de uma função que mapeia as chaves dentro do conjunto (Hashing).
- Permite "acesso direto" aos registros como um índice (endereço) dentro da tabela
- Útil quando a busca é feita sobre um número muito grande de dados que possuem faixas de valores muito variável.
- Exemplo: CPF em um arquivo de habitantes de uma cidade



## Gerando Funções Hashing Métodos da Divisão

**Princípio Básico:** O endereço do elemento na tabela é dado pelo resto da divisão da sua chave  $k$  por  $m$

( $h(k) = k \bmod m$ ),  
onde:  $m$  é o tamanho da tabela  
 $k$  é um inteiro correspondendo à chave

$$h(k) = k \bmod 10$$

