



# ALGORITMOS

PROF<sup>º</sup>: SIMONE DOMINICO

BUSCA EM VETORES

---

## ALGORITMOS DE BUSCA

- Ato de procurar por um elemento em um conjunto de dados
  - Recuperação de dados armazenados em um repositório ou base de dados
- A operação de busca visa responder se um determinado valor está ou não presente em um conjunto de elementos
  - Por exemplo, em um vetor

## ALGORITMOS DE BUSCA

- Tipos de busca de dados:
  - Dados estão estruturados
  - Dados ordenados
  - Valores duplicados

## ALGORITMOS DE BUSCA

- Baseado em uma chave
  - A chave de busca é o campo do item utilizado para comparação
  - Valor armazenado em um array de inteiros
- É por meio dela que sabemos se dado elemento é o que buscamos

## ALGORITMOS DE BUSCA

- Tipos de buscas
  - Dados armazenados em um vetor
  - Dados ordenados
- Métodos
  - Busca linear
  - Ordenada
  - Binária



# BUSCA SECUENCIAL

---

## BUSCA SEQUENCIAL

- Basicamente, esse algoritmo percorre o array que contém os dados desde a sua primeira posição até a última.
- Assume que os dados não estão ordenados, por isso a necessidade de percorrer o array do seu início até o seu fim.

## BUSCA SEQUENCIAL

- Basicamente, esse algoritmo percorre o vetor que contém os dados desde a sua primeira posição até a última.
- Assume que os dados não estão ordenados, por isso a necessidade de percorrer o vetor do seu início até o seu fim .



## BUSCA SEQUÊNCIAL

- Para cada posição do vetor, o algoritmo compara se a posição atual do vetor é igual ao valor buscado.
  - Se os valores forem iguais, a busca termina
  - Caso contrário, a busca continua com a próxima posição do vetor

## BUSCA SEQUENCIAL

0	1	2	3	4	5
3	1	9	5	2	8

- O número inteiro 2 está no vetor?

## BUSCA SEQUENCIAL

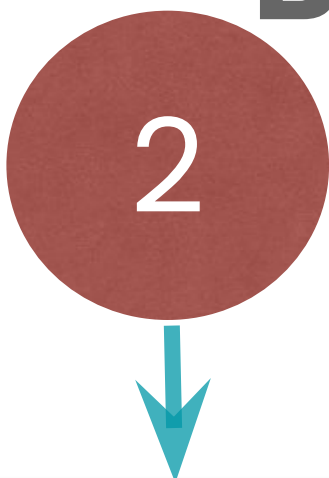
2



0	1	2	3	4	5
3	1	9	5	2	8

- O número inteiro 2 está no vetor?

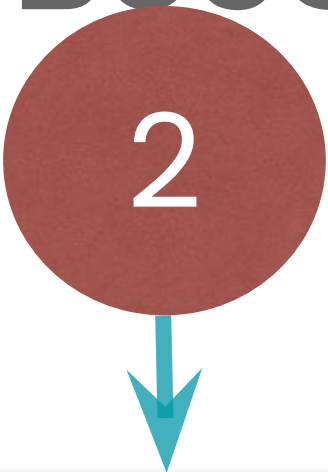
## BUSCA SEQUENCIAL



0	1	2	3	4	5
3	1	9	5	2	8

- O número inteiro 2 está no vetor?

## BUSCA SEQUENCIAL



0	1	2	3	4	5
3	1	9	5	2	8

- O número inteiro 2 está no vetor?
- Vai comparando até acabar o vetor ou achar o elemento.
- Neste caso está na posição 4 do vetor

## BUSCA secuencial

```
int busca_secuencial (int vector[size], int item)
{
    for (int i = 0; i < size; i++) {
        if (vector[i] == item) {
            return i;
        }
    }

    return -1;
}
```



**BUSCA  
ORDENADA**

---

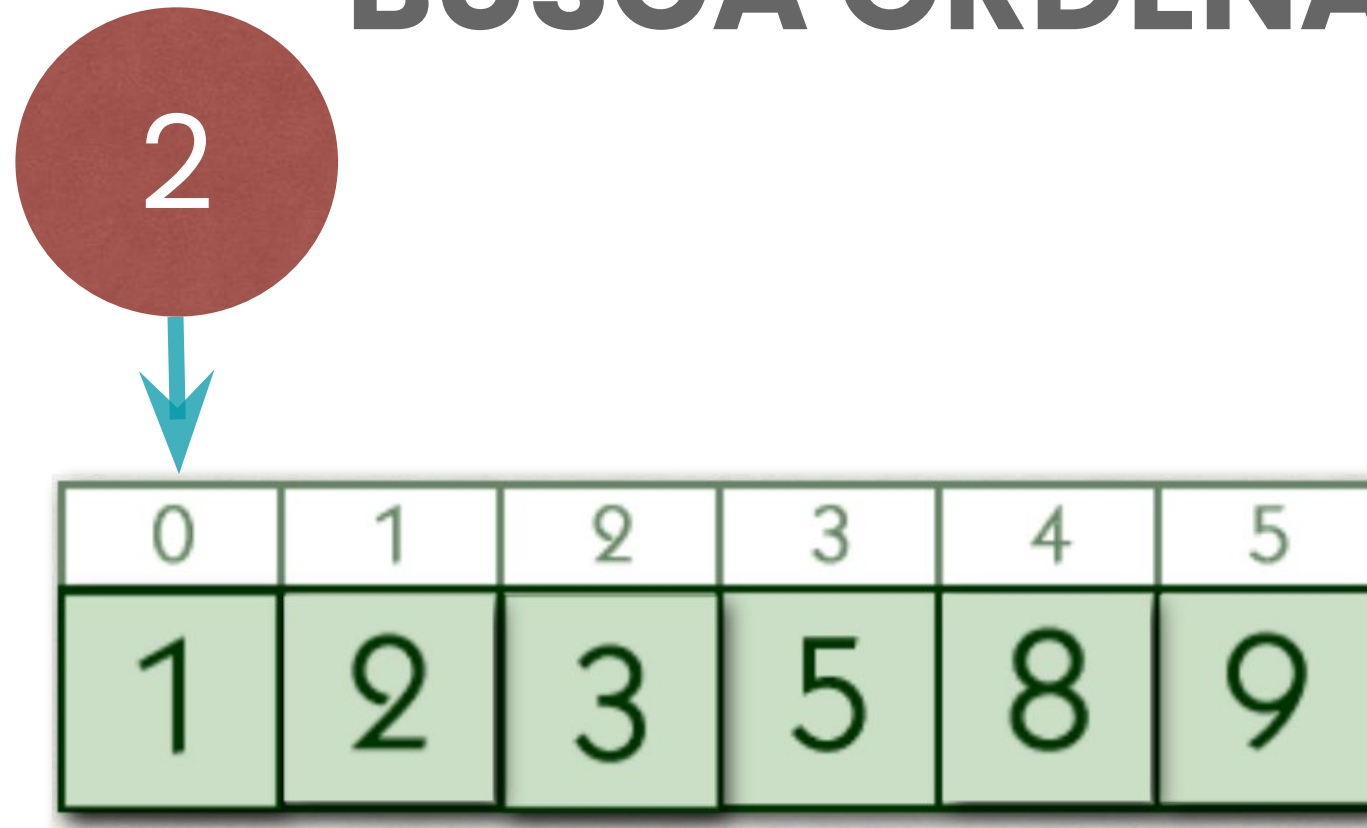
## BUSCA ORDENADA

- Assume que os dados estão ordenados.
- Se o elemento procurado for menor que o valor em uma determinada posição do vetor, temos a certeza de que ele não estará no restante do vetor
- Isso evita a necessidade de percorrer o vetor do seu início até o seu fim

0	1	2	3	4	5
1	2	3	5	8	9



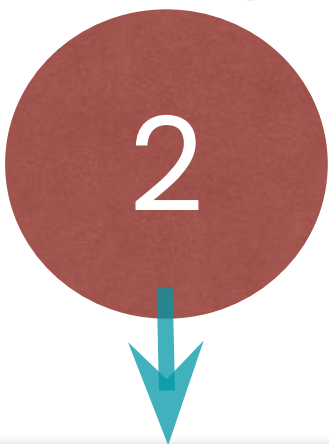
## BUSCA ORDENADA



0	1	2	3	4	5
1	2	3	5	8	9

- O elemento 2 está no vetor?

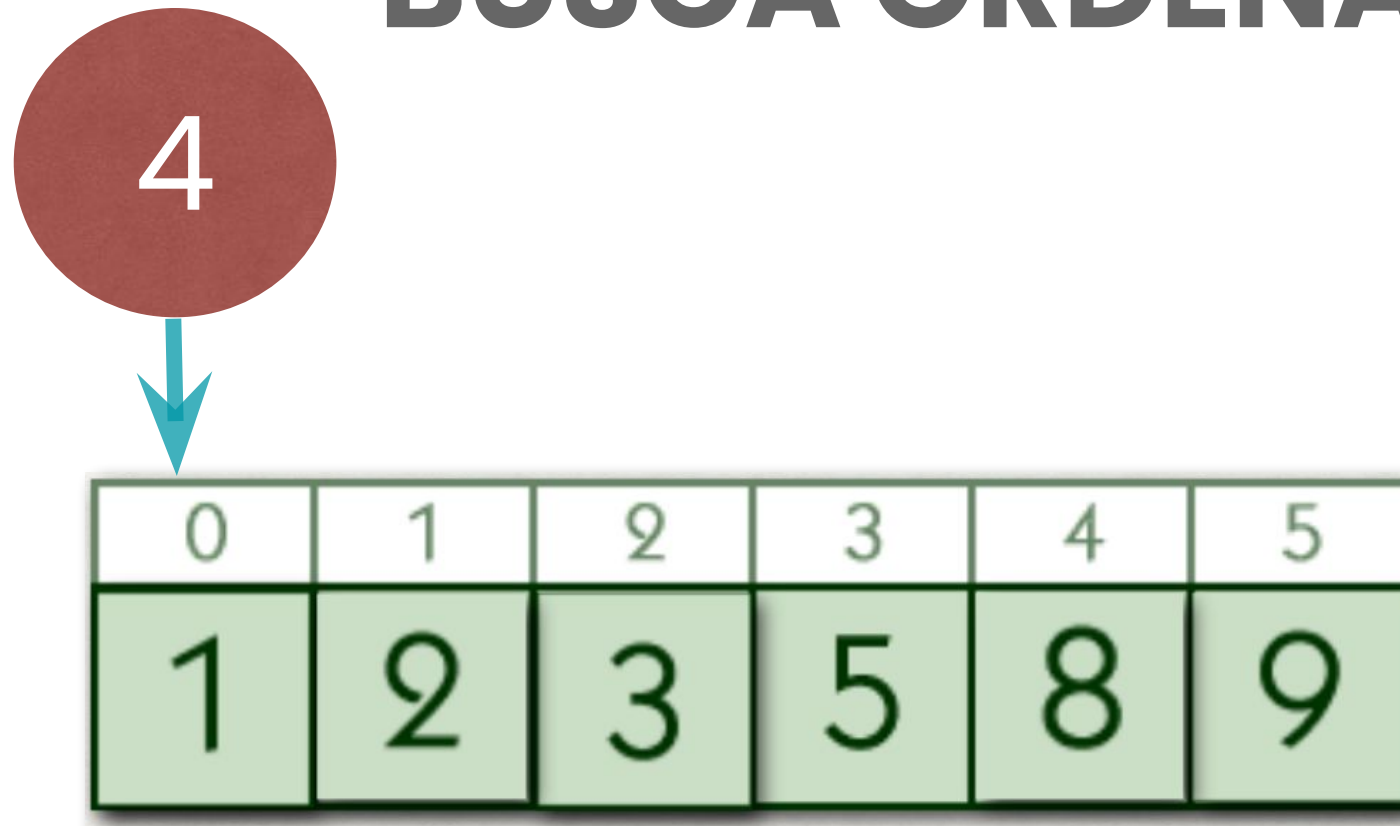
## BUSCA ORDENADA



0	1	2	3	4	5
1	2	3	5	8	9

- O elemento 2 está no vetor?


## BUSCA ORDENADA



0	1	2	3	4	5
1	2	3	5	8	9

- O elemento 4 está no vetor?

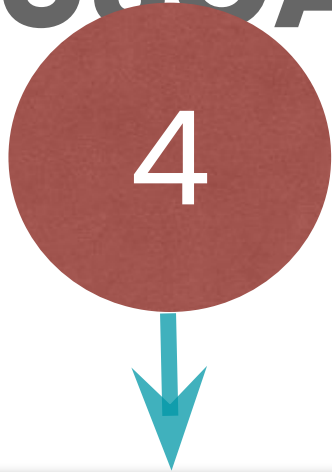
## BUSCA ORDENADA



0	1	2	3	4	5
1	2	3	5	8	9

- O elemento 4 está no vetor?

## BUSCA ORDENADA



0	1	2	3	4	5
1	2	3	5	8	9

- O elemento 4 está no vetor?

## BUSCA ORDENADA

4

TERMINA A BUSCA

0	1	2	3	4	5
1	2	3	5	8	9

- O elemento 4 está no vetor?

## BUSCA ORDENADA

- Desvantagem:
  - Ordenar o vetor tem um custo
    - Maior que realizar uma busca sequencial
  - Se for buscar um único elemento não vale a pena ordenar



**BUSCA BINÁRIA**

---



## BUSCA BINÁRIA

- A Busca Sequencial Ordenada é uma estratégia de busca extremamente simples.
- Ela percorre todo o array linearmente.
- Não utiliza adequadamente a ordenação dos dados.
- Uma estratégia de busca mais sofisticada é a Busca Binária
- Muito mais eficiente do que a Busca Sequencial Ordenada

## BUSCA BINÁRIA

- É uma estratégia baseada na idéia de dividir para conquistar.
  - A cada passo, esse algoritmo analisa o valor do meio do array.
  - Caso esse valor seja igual ao elemento procurado, a busca termina.
  - Caso contrário
    - Se o elemento do meio vier antes da chave, então a busca continua na metade posterior do vetor,
    - Caso contrário, a busca continua na metade anterior do vetor.

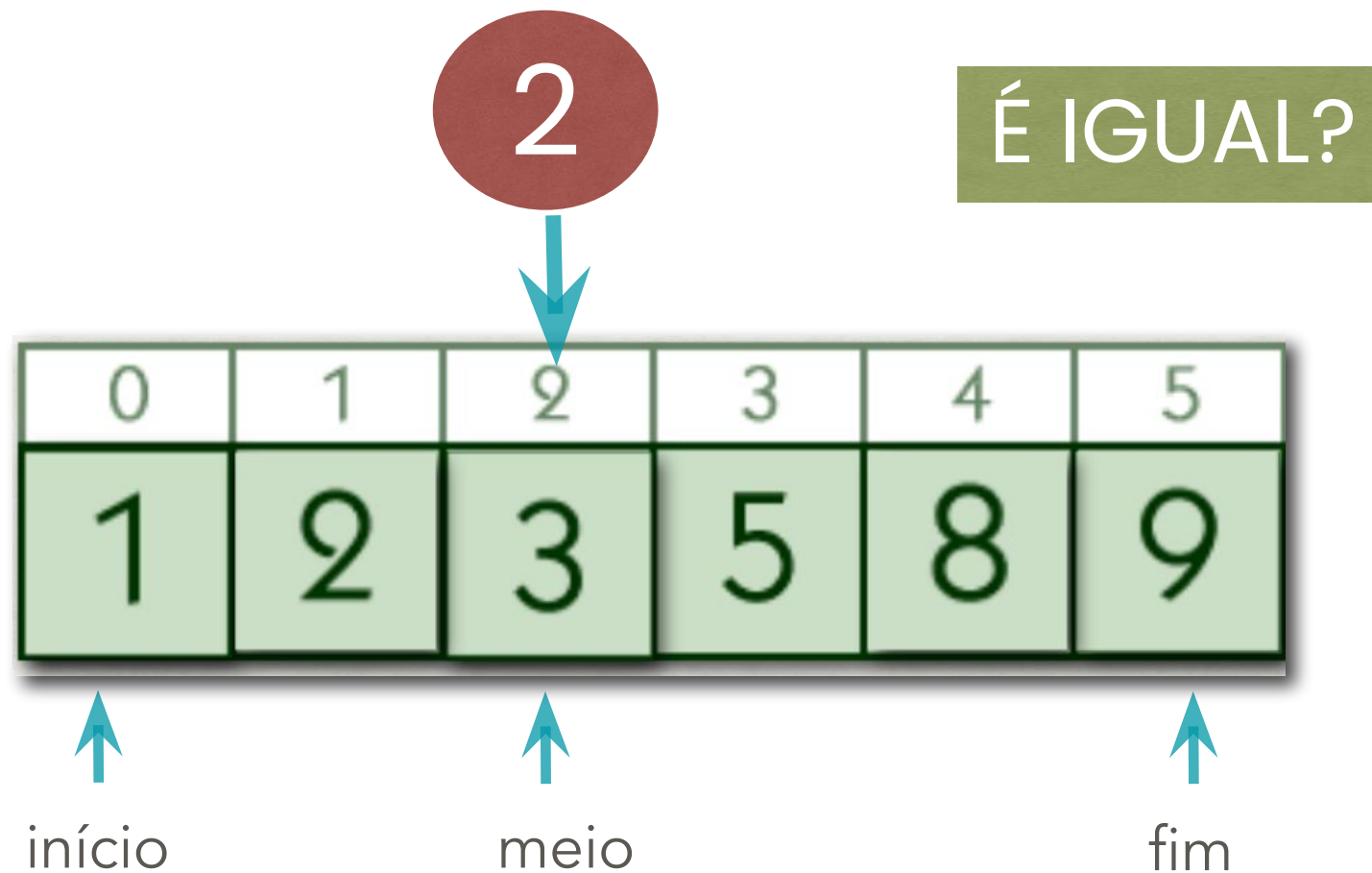
## BUSCA BINÁRIA

0	1	2	3	4	5
1	2	3	5	8	9

↑ início                      ↑ meio                                      ↑ fim

- O elemento 2 está no vetor?

## BUSCA BINÁRIA



- O elemento 2 está no vetor?

## BUSCA BINÁRIA

2

É MENOR?

0	1	2	3	4	5
1	2	3	5	8	9

↑ início      ↑ meio      ↑ fim

- O elemento 2 está no vetor?

## BUSCA BINÁRIA

2

É IGUAL?

0	1	2	3	4	5
1	2	3	5	8	9

↑  
início  
meio

↑  
fim

- O elemento 2 está no vetor?

## BUSCA BINÁRIA

2



É MAIOR?

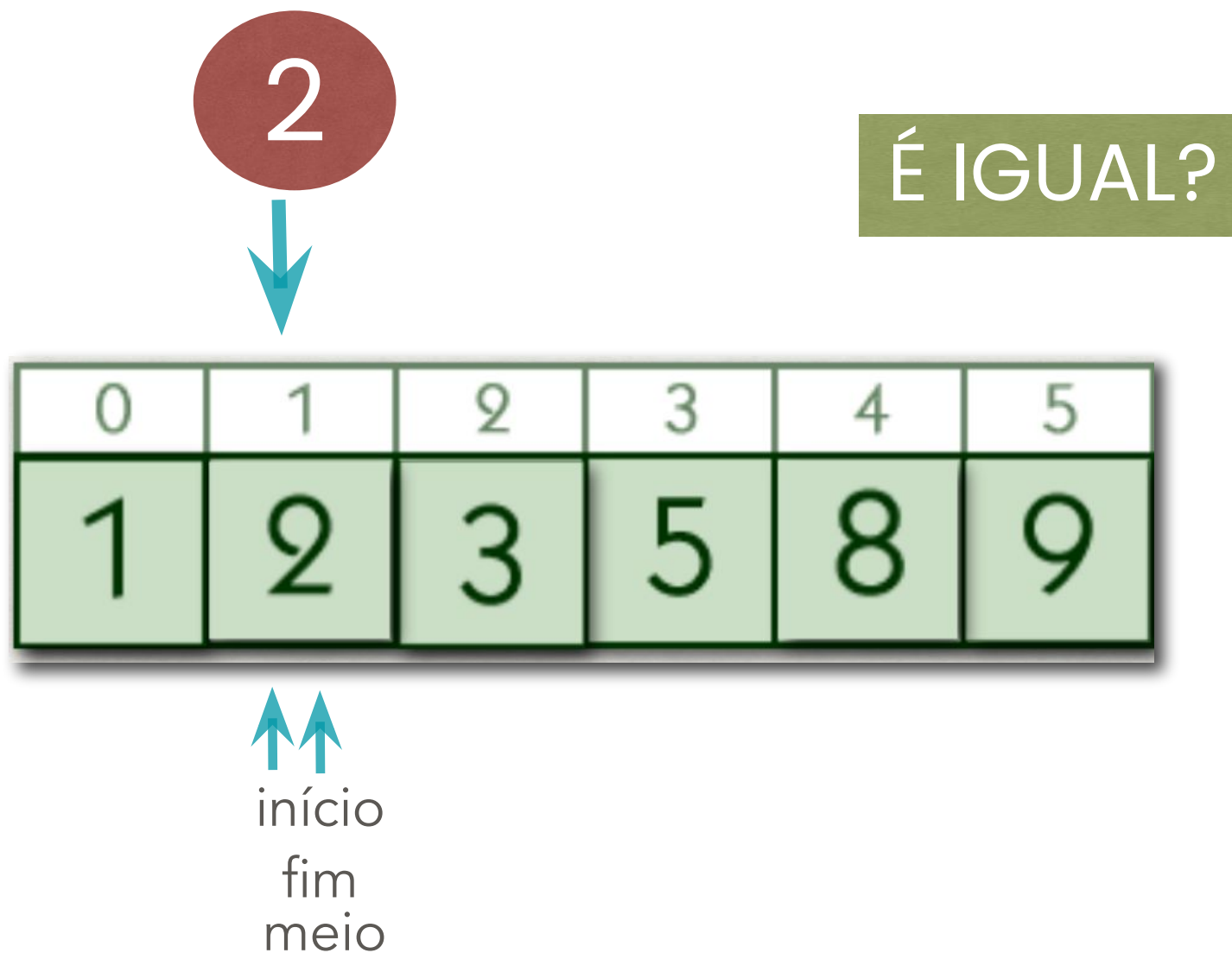
0	1	2	3	4	5
1	2	3	5	8	9



início  
fim  
meio

- O elemento 2 está no vetor?

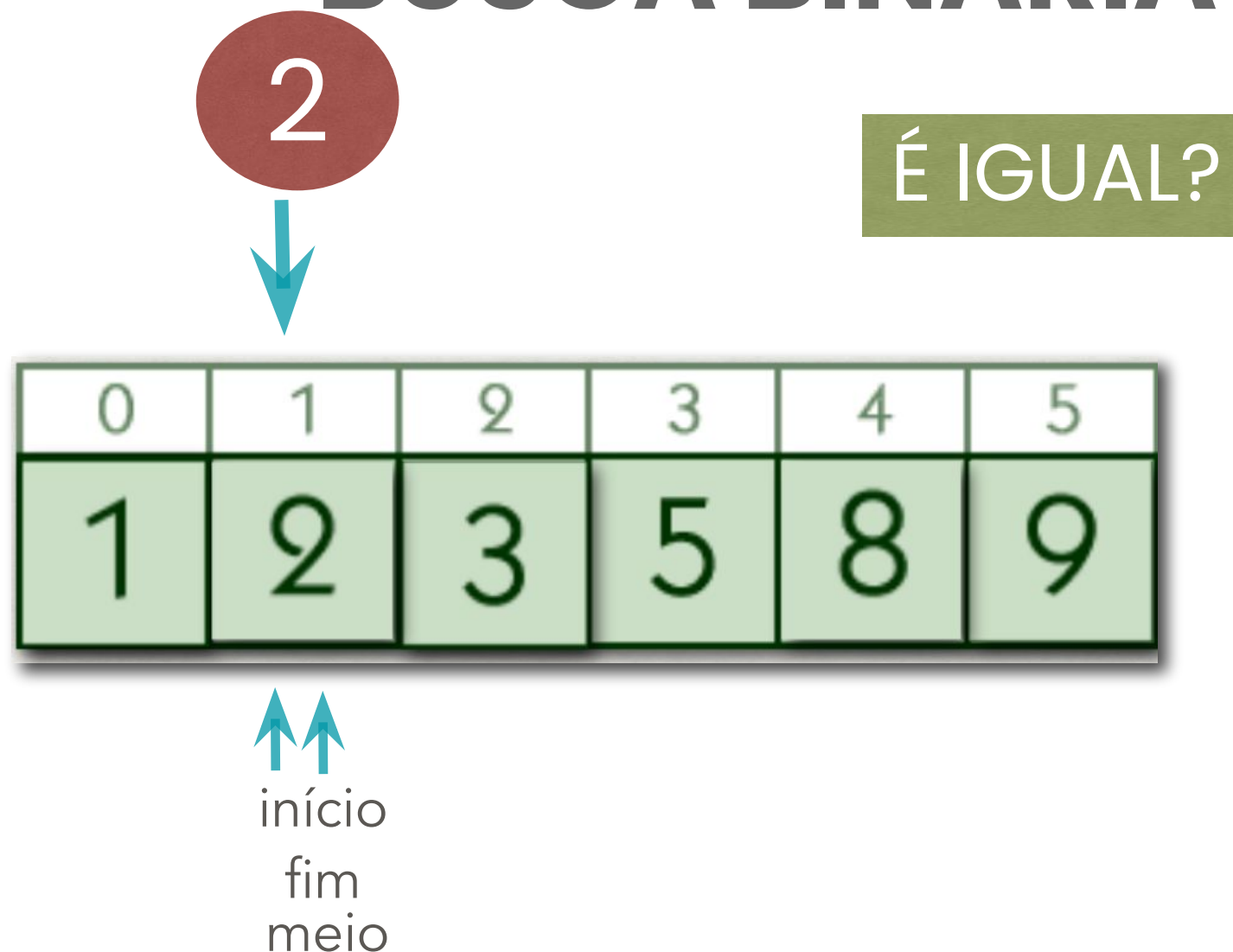
## BUSCA BINÁRIA



- O elemento 2 está no vetor?



## BUSCA BINÁRIA



- O elemento 2 está no vetor? Sim
- Condição de parada
  - Encontrar o valor ou início ser menor, ou igual ao fim

## BUSCA BINÁRIA

```
86  int busca_binaria(int vector[size], int item)
87  {
88      int begin = 0;
89      int end = size - 1;
90
91      while (begin <= end) { /* Condição de parada */
92
93          int i = (begin + end) / 2; /* Calcula o meio do sub-vetor */
94
95          if (vector[i] == item) { /* Item encontrado */
96              return i;
97          }
98
99          if (vector[i] < item) { /* Item está no sub-vetor à direita */
100              begin = i + 1;
101          } else { /* vector[i] > item. Item está no sub-vetor à esquerda */
102              end = i;
103          }
104      }
105
106      return -1;
107 }
```