

# Aula anterior

- Introdução a Recursividade
- Fundamentos da Recursividade
  - Implementação de Recursividade
  - As três regras de Recursão
  - Quando Não usar Recursividade

# Métodos de Ordenação Simples

---

Prof. Diego Silva Caldeira Rocha

---

# Objetivos

- Métodos de Ordenação Simples
  - Ordenação Método Bolha (Bubble-Sort)
  - Ordenação Método Seleção (Selection-Sort)
  - Ordenação Método de Inserção (insertion-Sort)
  - Ordenação Método de Contagem (Counting-Sort)
  - Ordenação Método de Chave (Radix-Sort)

# Ordenação

- Ordenação é o processo de rearranjar um conjunto de dados, seja em ordem ascendente ou descendente
- O objetivo é facilitar a recuperação dos dados (informação)
- A maior parte dos algoritmos de ordenação é baseada na comparação das chaves e o número de trocas entre os itens
- Métodos de ordenação simples: possuem códigos de fácil compreensão; são adequados para pequenos conjuntos de dados; usam menos comparações; tem complexidade  $O(n^2)$
- Métodos de ordenação eficientes: possuem códigos mais complexos; são adequados para grandes conjuntos de dados; usam mais comparações; e tem complexidade  $O(n \log n)$

```
public int[] BubbleSort( int[] vet ){
    int aux, n= vet.length;

    for ( int i = n-1; i>0; i-- ){
        for ( int j = n-1; j >n-i-1; j-- ){
            if( vet[ j ] < vet[ j-1 ] ){
                aux  = vet[ j ];
                vet[ j ]    = vet[ j -1 ];
                vet[ j-1 ] = aux;
            }
        }
    }
    return( vet );
}
```

101    115    30    63    47    20

Bolha

# Exemplo

101    115    30    63

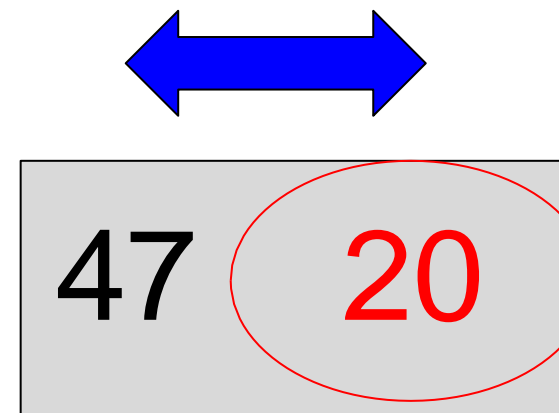
Comparação



Bolha

# Exemplo

101      115      30      63



Bolha

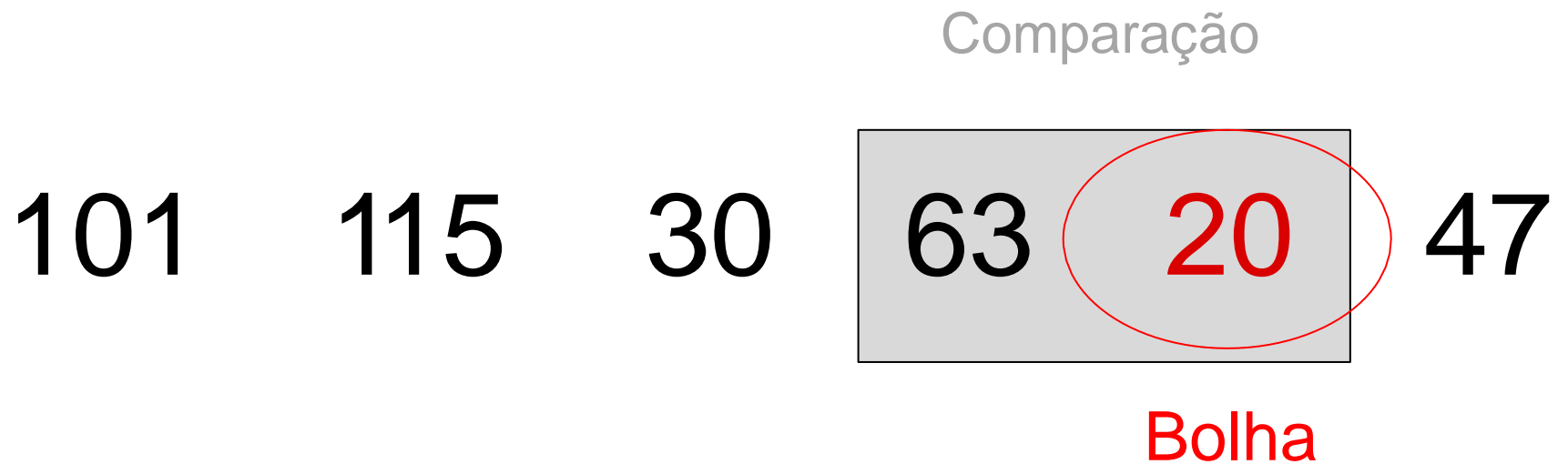


# Exemplo

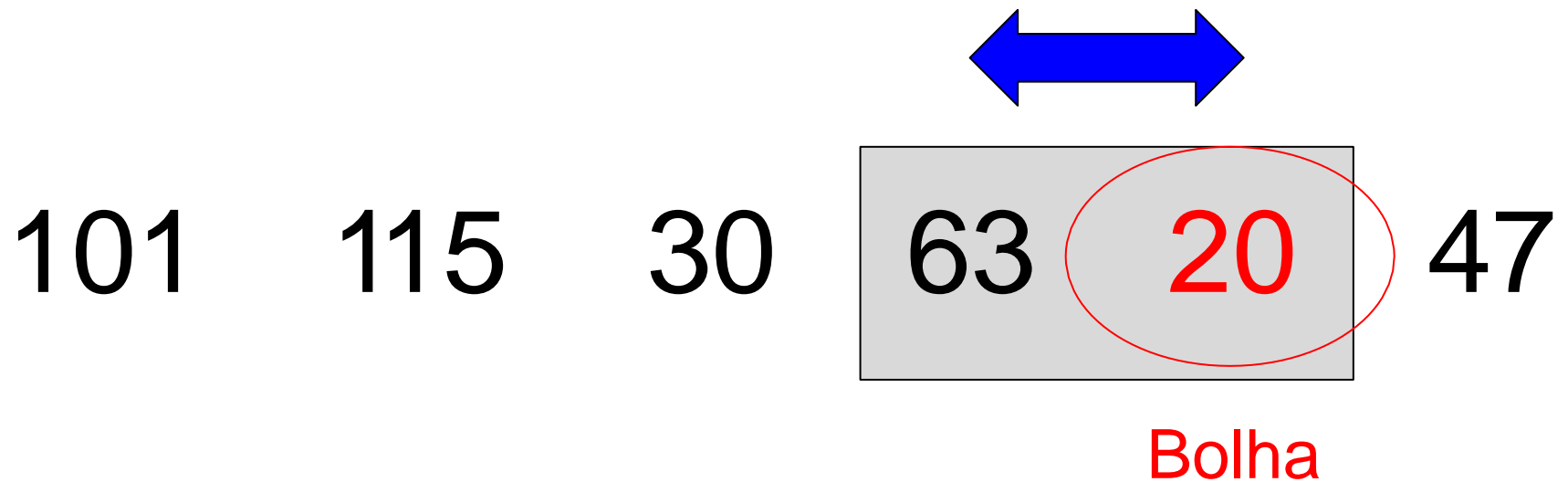
101    115    30    63    20    47

Bolha

# Exemplo



# Exemplo

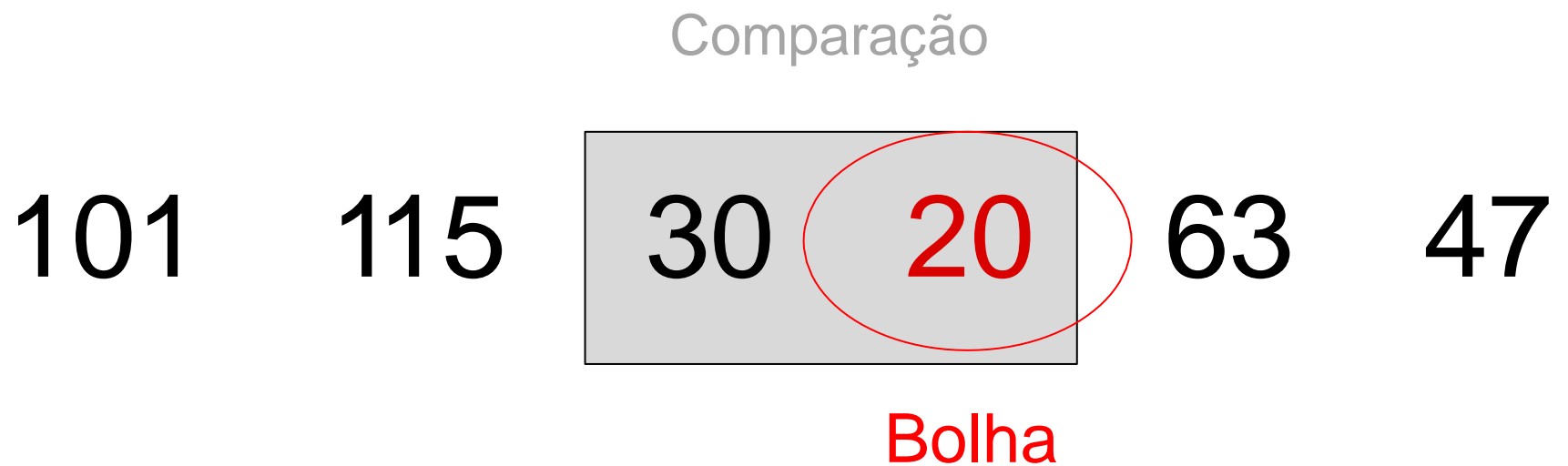


# Exemplo

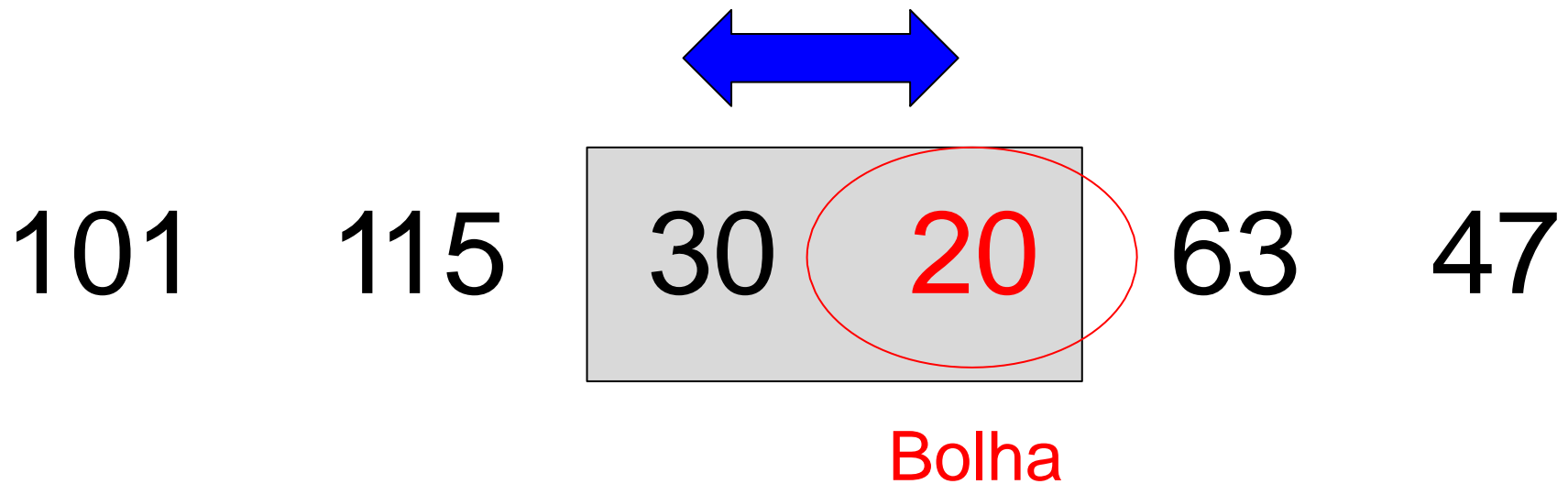
101    115    30    20    63    47

Bolha

## Exemplo



## Exemplo



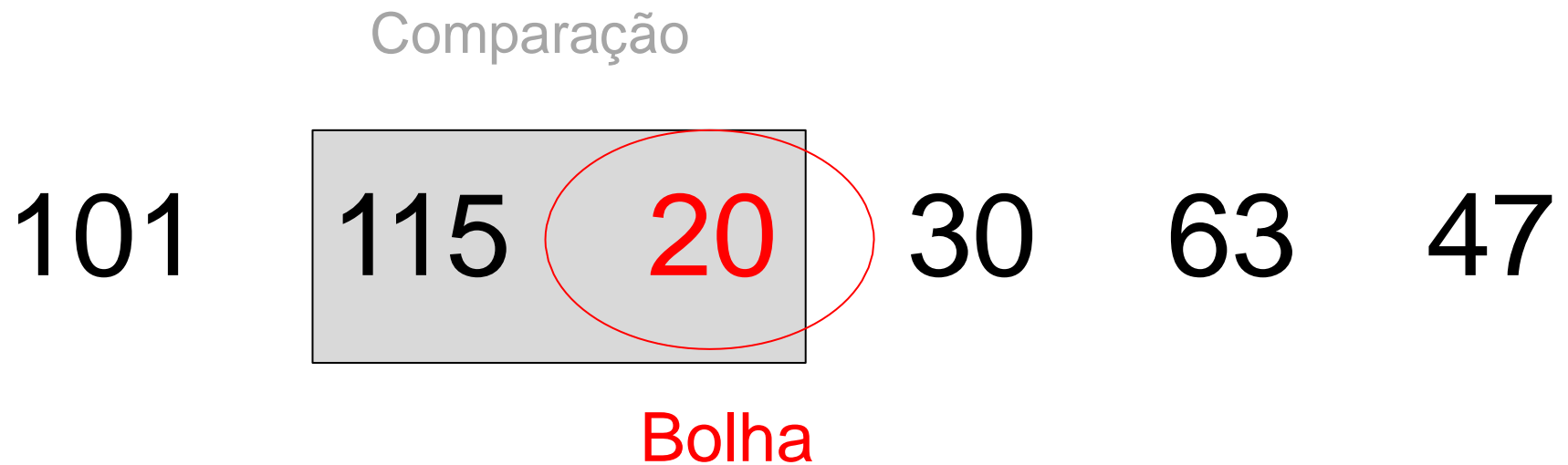
# Exemplo

101    115    20    30    63    47



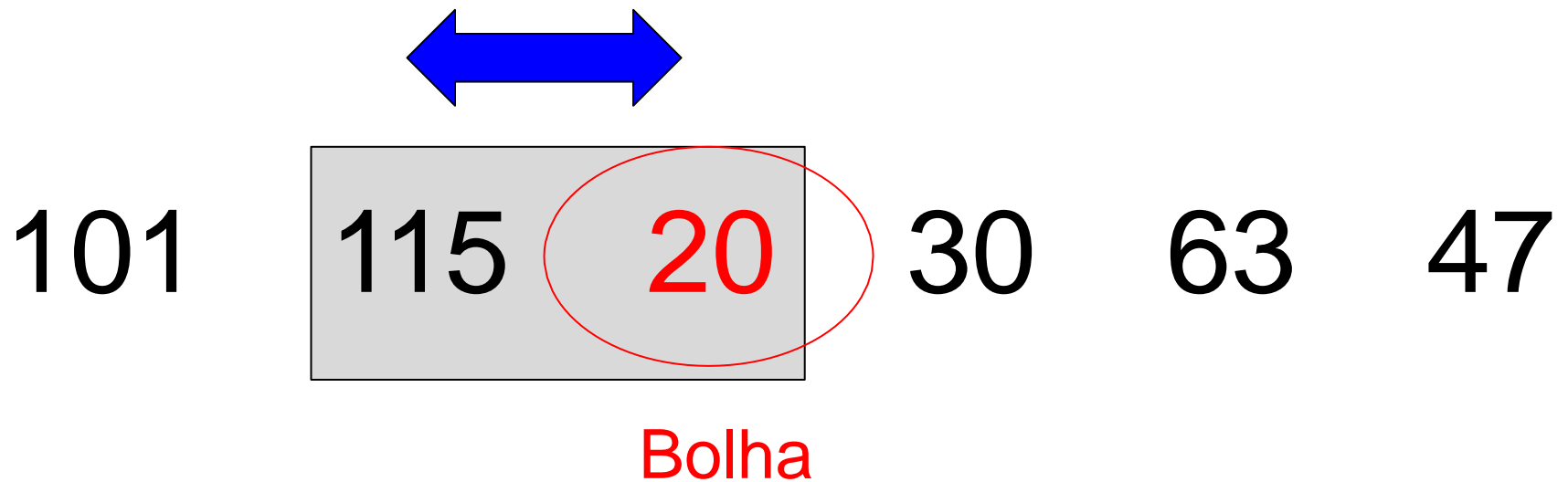
Bolha

## Exemplo





# Exemplo



# Exemplo

101 20 115 30 63 47

Bolha

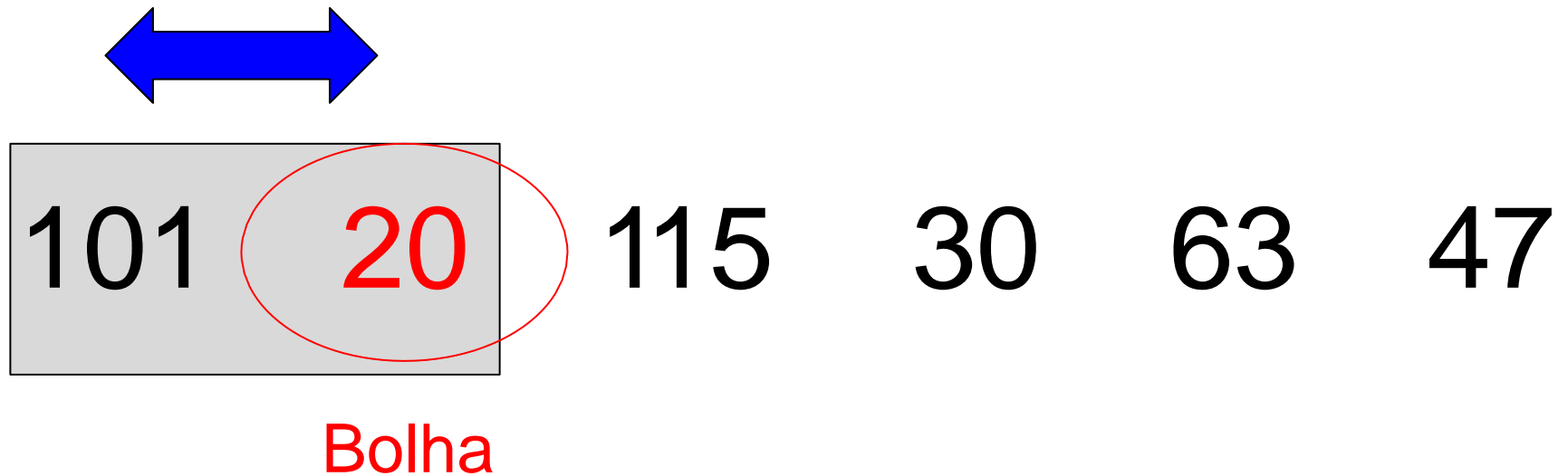
## Exemplo

Comparação



Bolha

# Exemplo



# Exemplo

20 101 115 30 63 47

Bolha

# Exemplo

20 101 115 30 63 47

Ordenado

# Exemplo

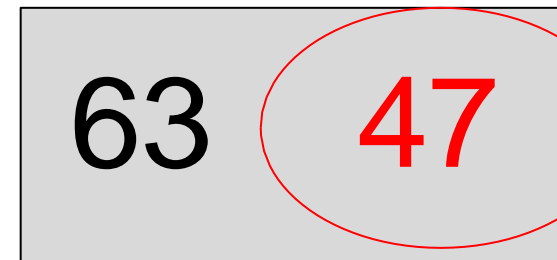
20 101 115 30 63 47

Bolha

## Exemplo

20 101 115 30

Comparação

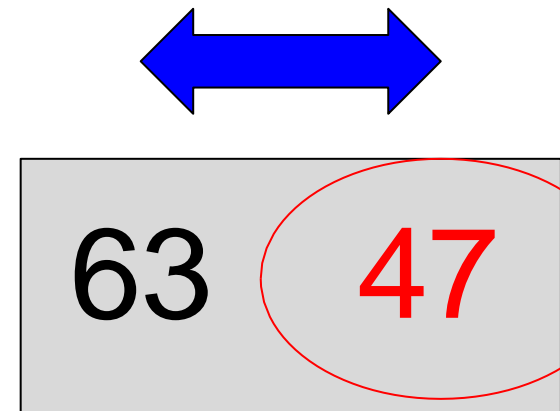


Bolha



# Exemplo

20 101 115 30



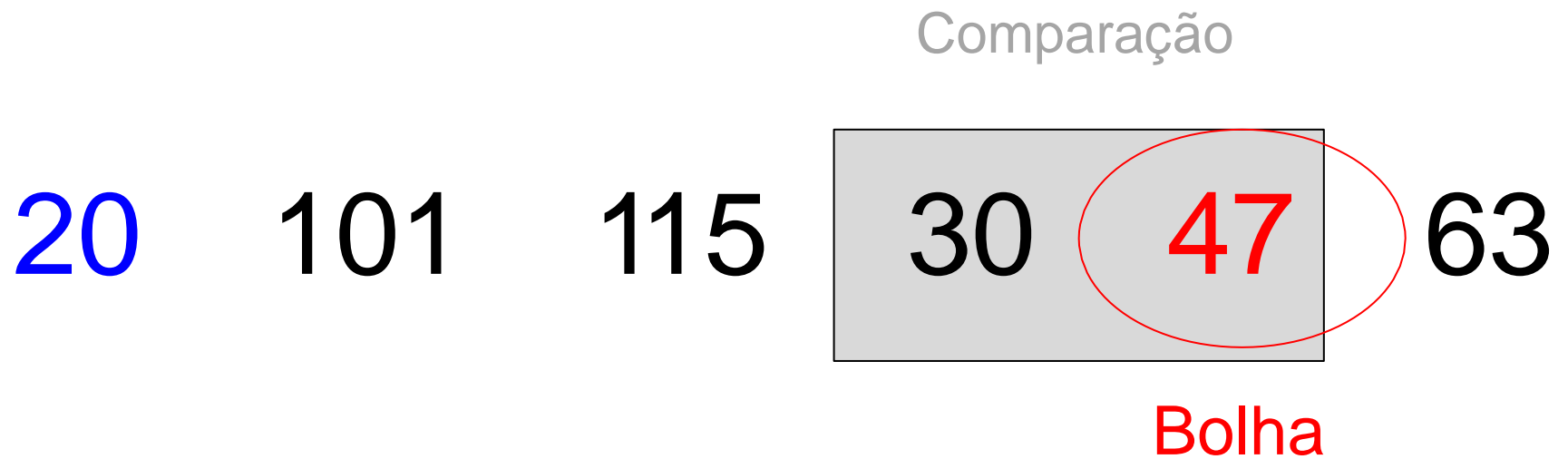
Bolha

# Exemplo

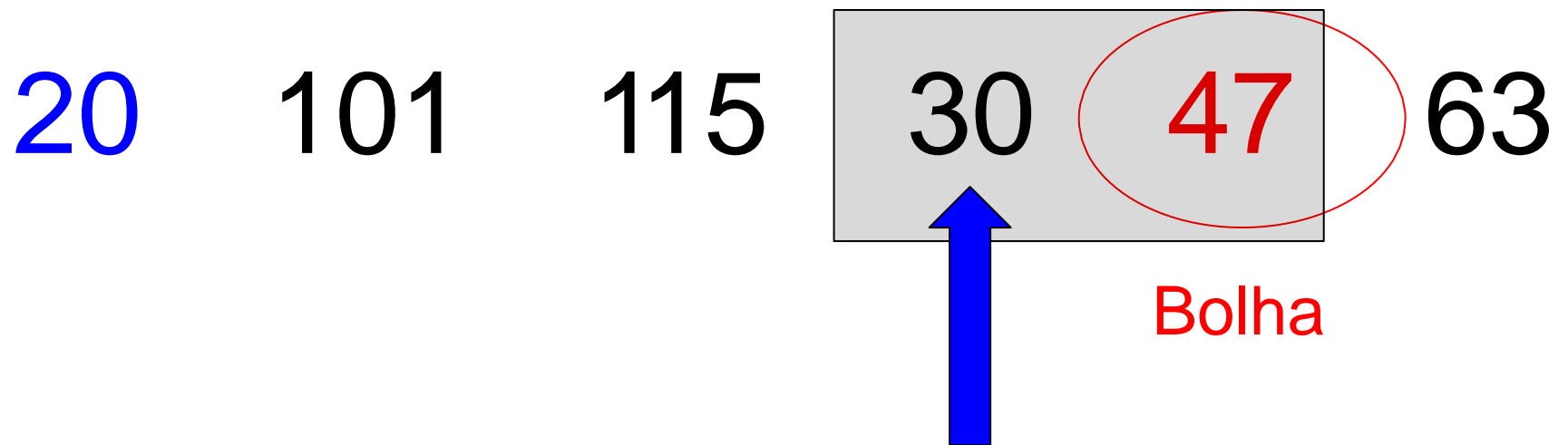
20 101 115 30 47 63

Bolha

# Exemplo



## Exemplo



Menor (Será o número da bolha)

# Exemplo

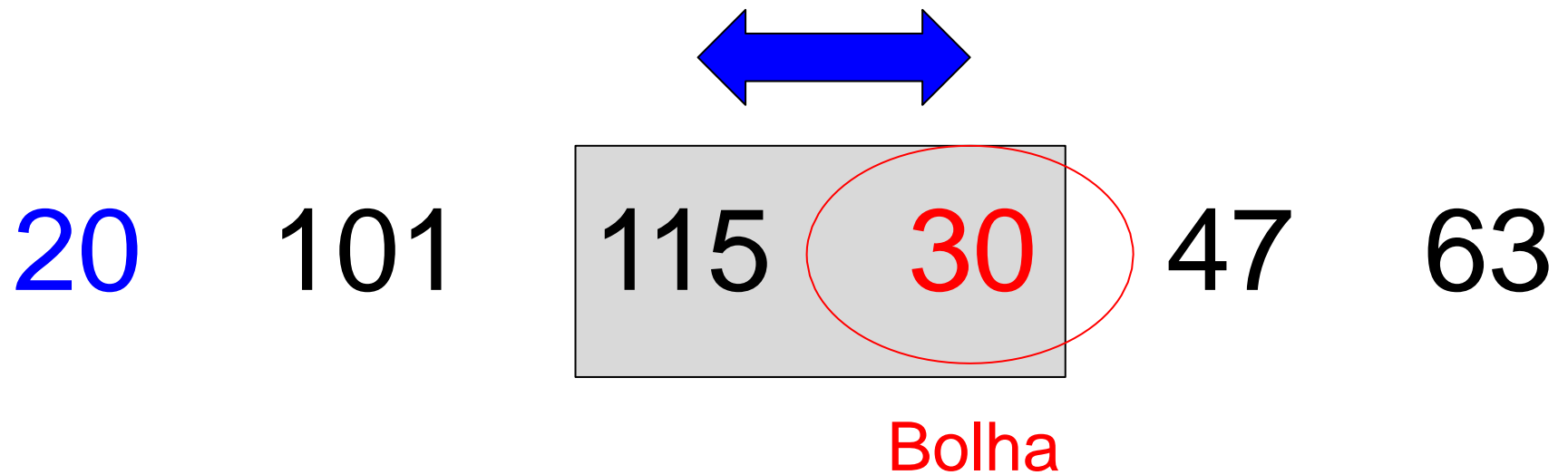
20 101 115 30 47 63

Bolha

# Exemplo



## Exemplo



# Exemplo

20 101 30 115 47 63

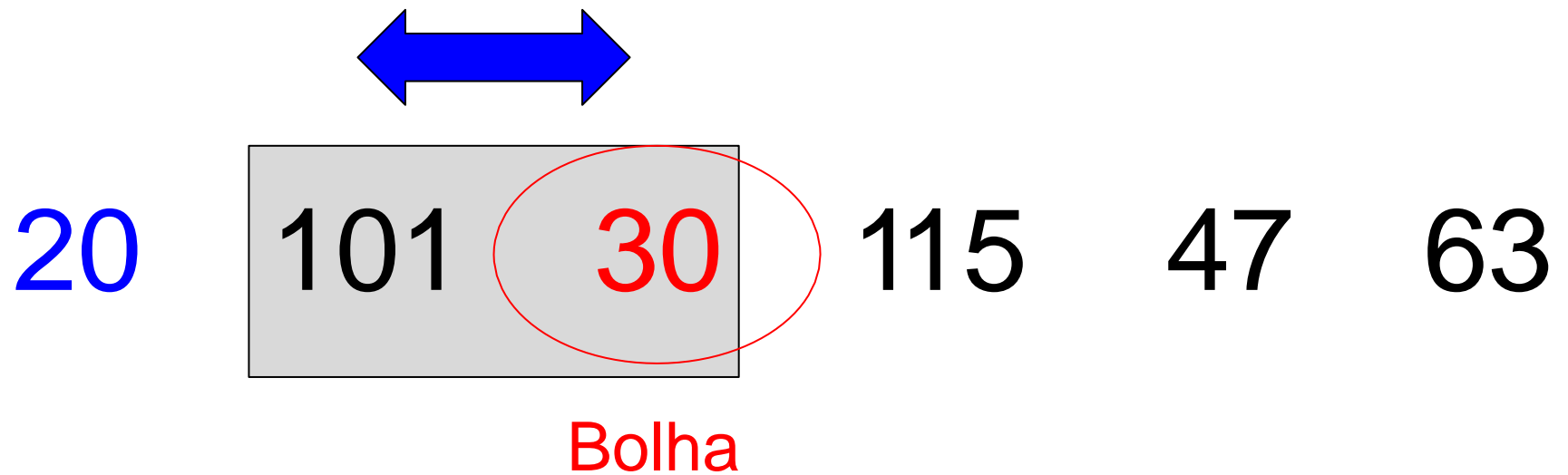
Bolha



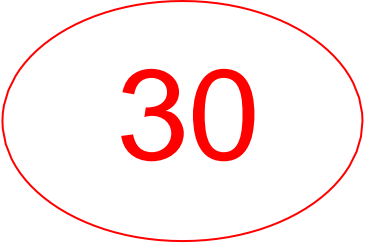
## Exemplo



# Exemplo



20 30 101 115 47 63



Bolha

# Exemplo

20 30 101 115 47 63

Ordenado

# Exemplo

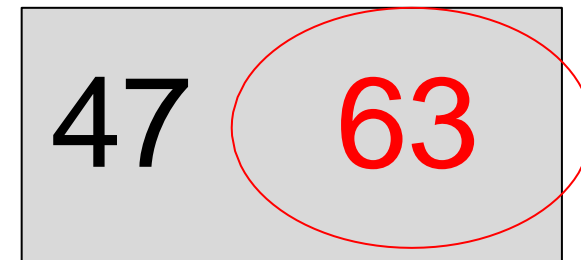
20 30 101 115 47 63

Bolha

# Exemplo

20 30 101 115

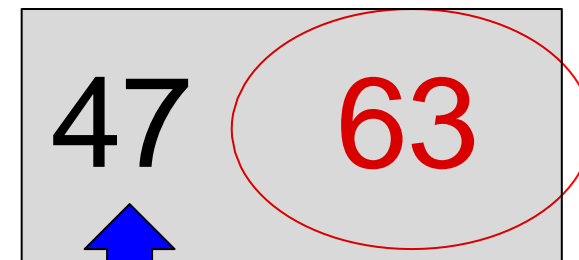
Comparação



Bolha

## Exemplo

20 30 101 115



Bolha

Menor (Será o número da bolha)

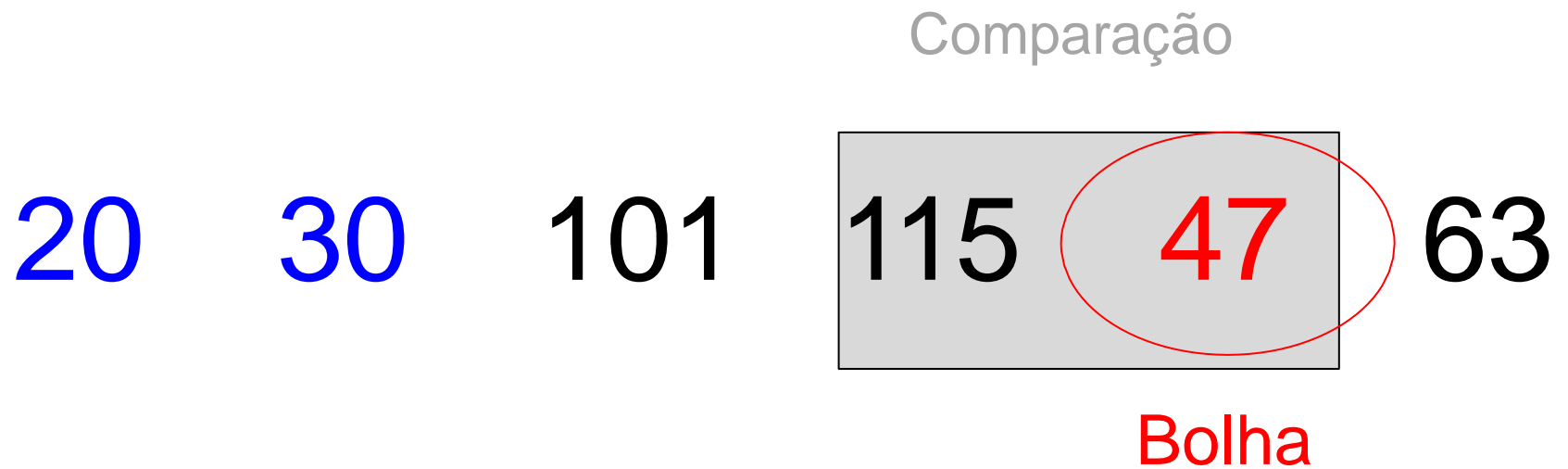
# Exemplo

20 30 101 115 47 63

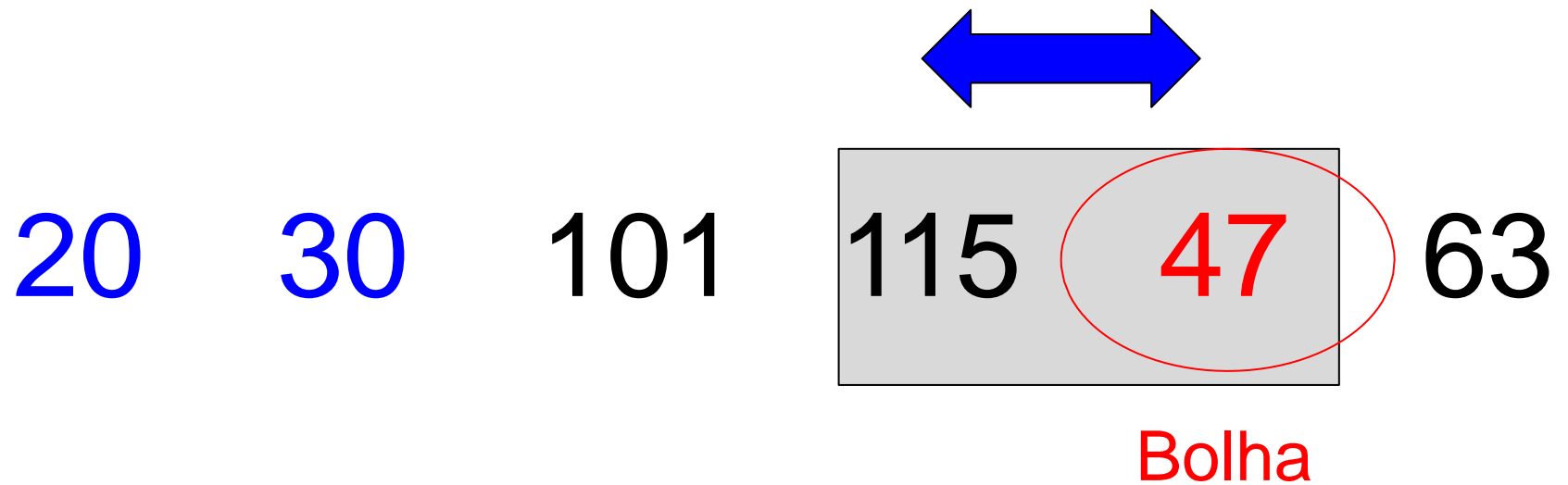
Bolha



# Exemplo



# Exemplo



# Exemplo

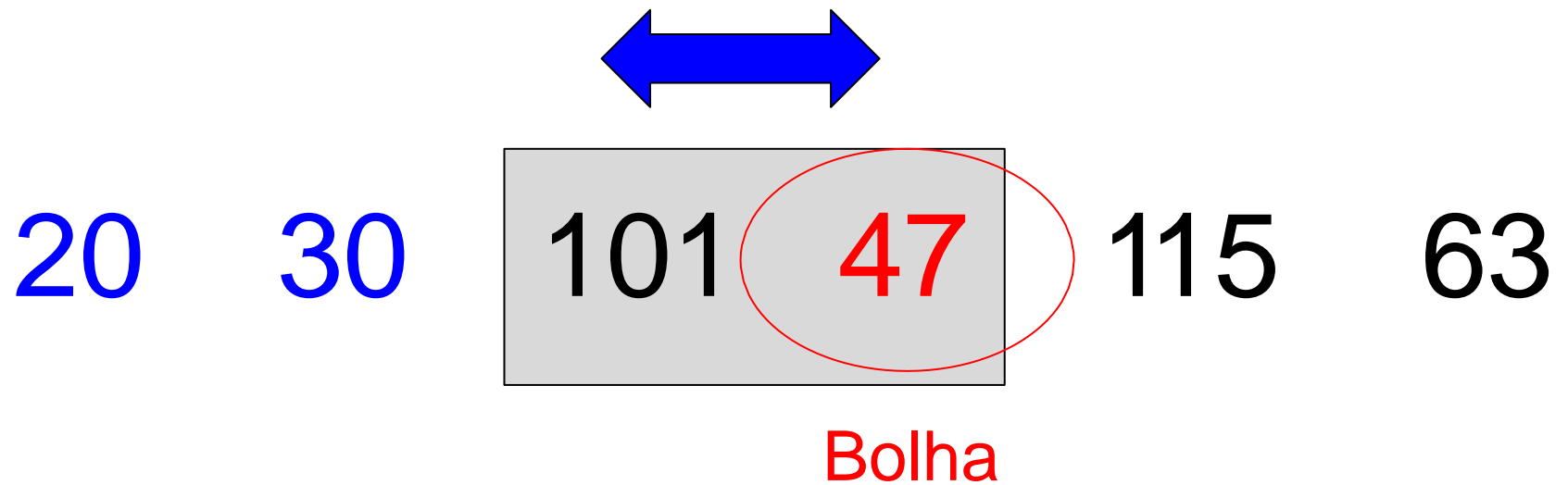
20 30 101 47 115 63

Bolha


## Exemplo



# Exemplo



20 30 47 101 115 63



Bolha

# Exemplo

20   30   47   101   115   63

Ordenado

# Exemplo

20 30 47 101 115 63

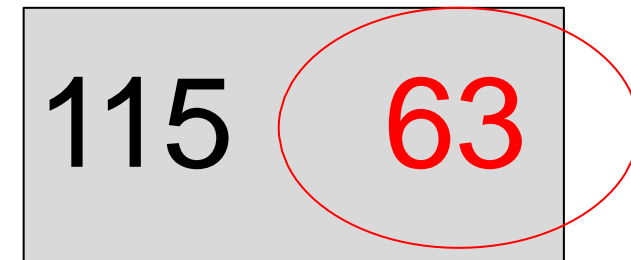
Bolha



# Exemplo

20 30 47 101

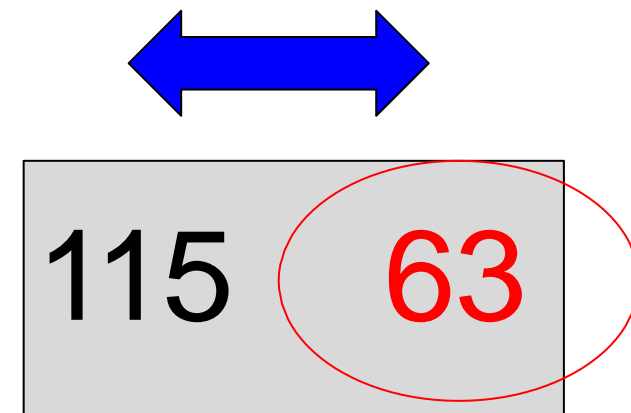
Comparação



Bolha

## Exemplo

20 30 47 101

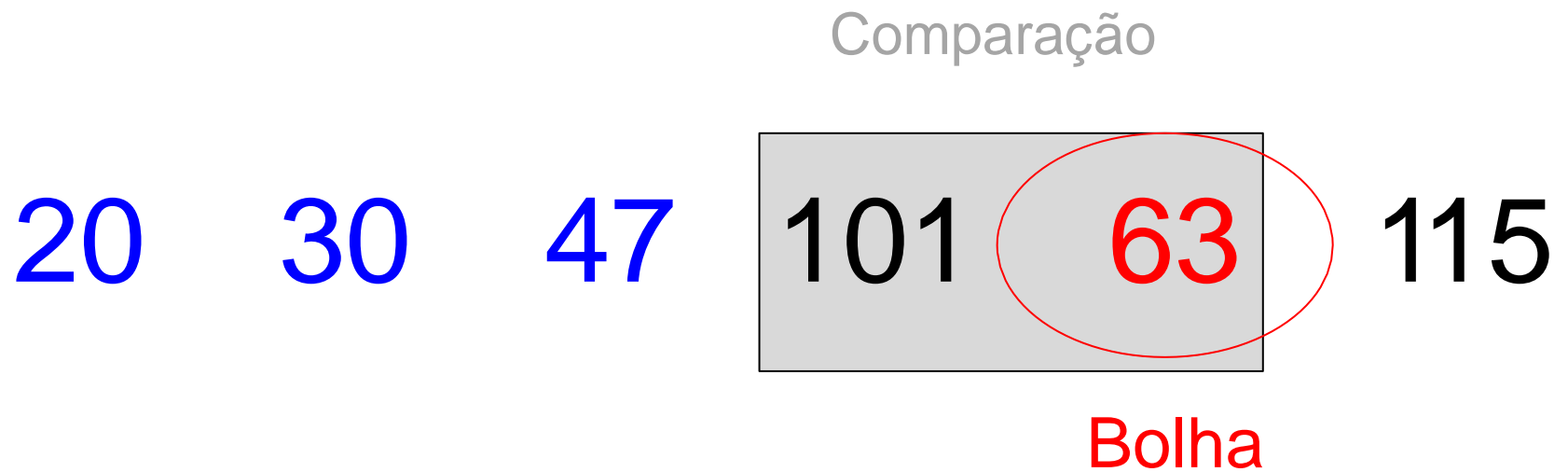


Bolha

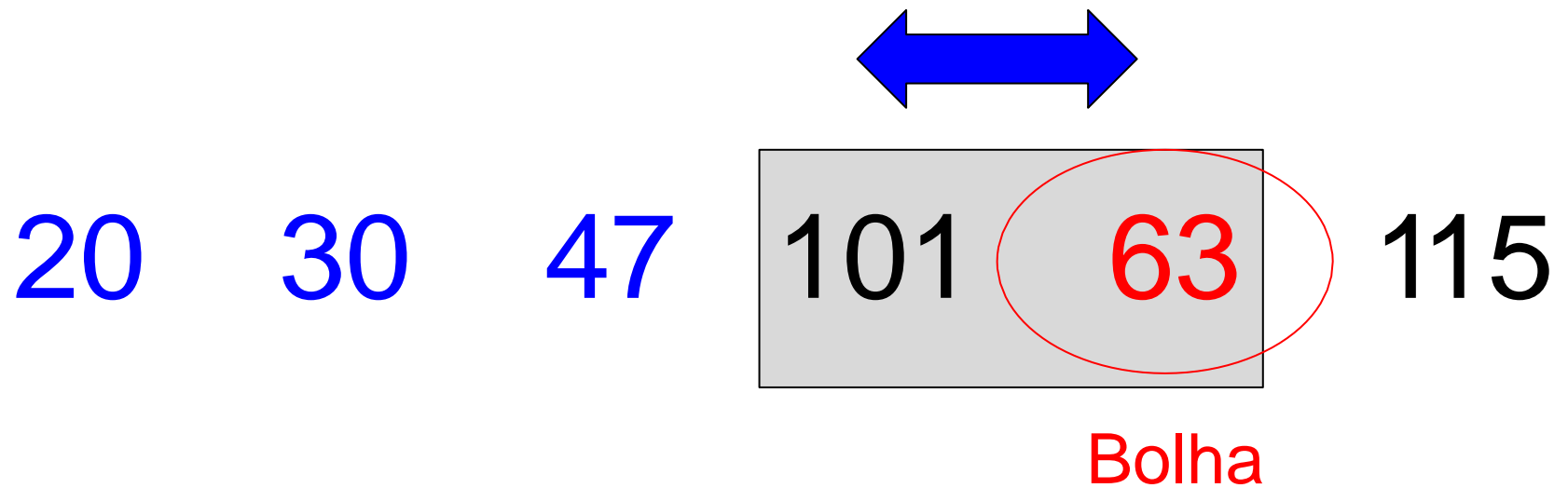
20 30 47 101 63 115

Bolha

# Exemplo



## Exemplo



# Exemplo

20 30 47 63 101 115



Bolha

# Exemplo

20 30 47 63 101 115

Ordenado

20    30    47    63    101    115

Ordenado

O algoritmo terminou? Por que?



# Exemplo

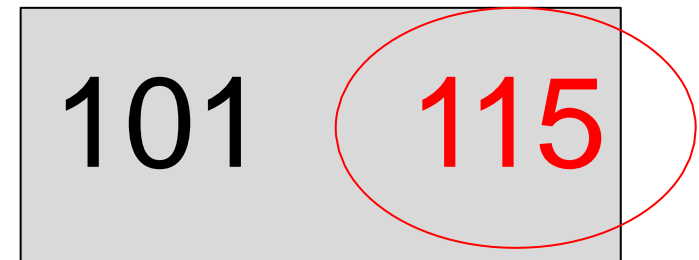
20 30 47 63 101 115

Bolha

# Exemplo

20 30 47 63

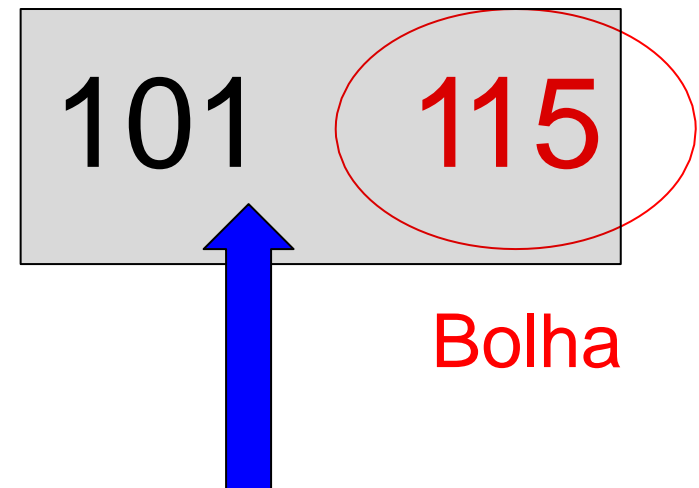
Comparação



Bolha

# Exemplo

20 30 47 63



Bolha

Menor (Será o número da bolha)

# Exemplo

20 30 47 63 101 115

Bolha

20    30    47    63    101    115

Ordenado

O algoritmo terminou? Por que?

# Exemplo

20 30 47 63 101 115

Ordenado

# Análise do Número de Movimentações

- Pior caso: o *array* está ordenado de forma decrescente

$$M_{Max}(n) = 3 * \sum_{i=1}^{n-1} (n - i) = 3 * \frac{n(n - 1)}{2}$$

- Caso médio: depende do número de inversões em todas as permutações do *array*

$$M_{Med}(n) = 3 * \frac{n(n - 1)}{4}$$

# Código

```
public int[] SelectionSort(int[] vet){

    int menor;
    for( int fixo = 0; fixo < vet.length-1; fixo++ ){
        menor = fixo;
        for( int i = menor + 1; i < vet.length; i++ ){
            if( vet[ i ] < vet[ menor ]){
                menor = i;
            }
        }
        if( menor != fixo ){
            int aux      = vet[ fixo ];
            vet[ fixo ]  = vet[ menor ];
            vet[ menor ] = aux;
        }
    }
    return( vet );
}
```



# Análise do Número de Comparações

- Método de ordenação por seleção em que os registros são comparados, dois a dois e o menor é movimentado para o início do *array*

$$C(n) = \frac{n(n-1)}{2}, \text{ para os três casos}$$

# Código

## Ordenação por **Seleção**

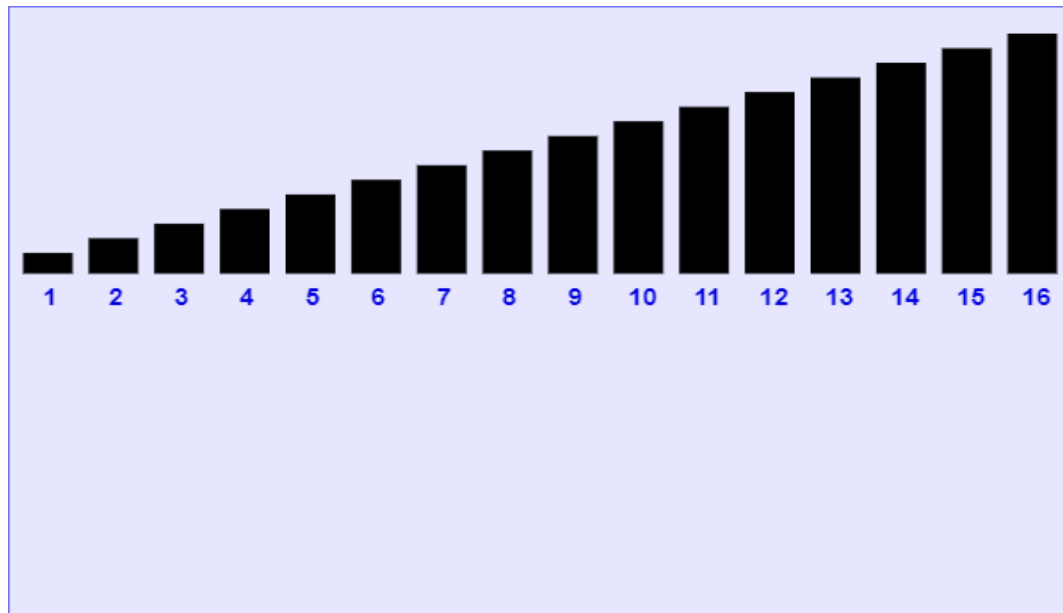
### **Selection-Sort**

Método **ineficiente** e **instável**

Recomendado para **conjuntos pequenos**

### Desvantagens

Caso o **vetor** esteja **ordenado**, as **comparações serão realizadas** do mesmo modo



	8
	5
	2
	6
	9
	3
	1
	4
	0
	7

# Código

```
public int[] InsertionSort( int[] vet ){
    int chave, j;
    for ( int i = 1; i < vet.length; i++ ){
        chave = vet[ i ];
        j      = i - 1;
        while( j >= 0 && vet[ j ] > chave ){
            vet[ j+1 ] = vet[ j ];
            j          = j - 1;
        }
        vet[ j+1 ] = chave;
    }
    return( vet );
}
```

101      115      30      63      47      20

101 115 30 63 47 20

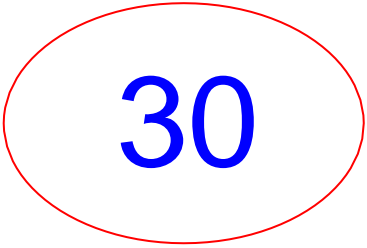
Inicialmente, temos um elemento,  
logo, ele está na posição correta

101 115 30 63 47 20



Comparamos o 101 e 115 e,  
como o novo elemento é o  
maior, os dois estão em ordem

101    115    30    63    47    20



Comparamos o 115 e 30 e, como o novo elemento é o menor, vamos procurar sua posição na lista ordenada

**30** variável  
temporária

101      115      30      63      47      20

Enquanto procuramos,  
deslocamos os elementos  
maiores que o 30



30 variável  
temporária

101      115      63      47      20

---

Enquanto procuramos,  
deslocamos os elementos  
maiores que o 30



30

variável  
temporária

101

115

63

47

20

Enquanto procuramos,  
deslocamos os elementos  
maiores que o 30

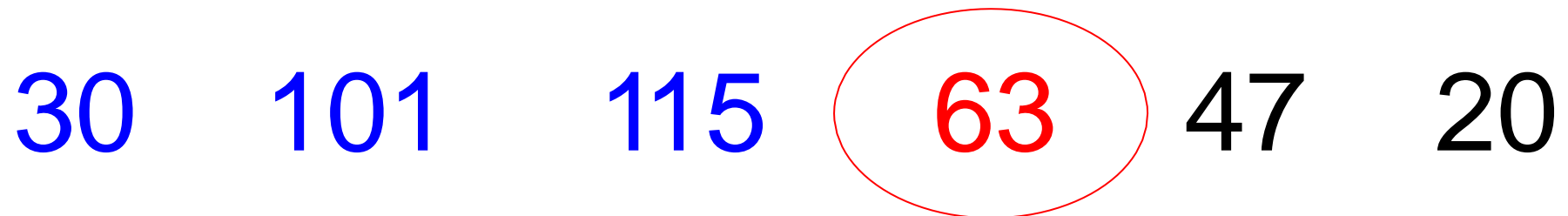


Encontramos a  
posição do 30

30    101    115    63    47    20

Encontramos a  
posição do 30

30    101    115    63    47    20



Comparamos o 115 e 63 e, como o novo elemento é o menor, vamos procurar sua posição na lista ordenada

63variável  
temporária

30    101    115    \_\_\_\_\_    47    20

Enquanto procuramos,  
deslocamos os elementos  
maiores que o 63

63

variável  
temporária

30    101    \_\_\_\_\_    115    47    20

Enquanto procuramos,  
deslocamos os elementos  
maiores que o 63

63variável  
temporária

30    \_\_\_\_\_    101    115    47    20

Enquanto procuramos,  
deslocamos os elementos  
maiores que o 63





Encontramos a  
posição do 63

30    63    101    115    47    20

Encontramos a  
posição do 63

30    63    101    115    47    20



Comparamos o 115 e 47 e, como o novo elemento é o menor, vamos procurar sua posição na lista ordenada

47 variável  
temporária

30    63    101    115    \_\_\_\_\_    20

Enquanto procuramos,  
deslocamos os elementos  
maiores que o 47

47 variável  
temporária

30    63    101    \_\_\_\_\_    115    20

Enquanto procuramos,  
deslocamos os elementos  
maiores que o 47

47 variável  
temporária

30    63    \_\_\_\_\_    101    115    20

Enquanto procuramos,  
deslocamos os elementos  
maiores que o 47

47variável  
temporária

30                63    101    115    20

Enquanto procuramos,  
deslocamos os elementos  
maiores que o 47



Encontramos a  
posição do 47



30    47    63    101    115    20

Encontramos a  
posição do 47

30    47    63    101    115    20

Comparamos o 115 e 20 e, como o novo elemento é o menor, vamos procurar sua posição na lista ordenada

20 variável  
temporária

30    47    63    101    115    \_\_\_\_\_

Enquanto procuramos,  
deslocamos os elementos  
maiores que o 20

20 variável  
temporária

30    47    63    101    \_\_\_\_\_    115

Enquanto procuramos,  
deslocamos os elementos  
maiores que o 20

20 variável  
temporária

30    47    63                101    115

Enquanto procuramos,  
deslocamos os elementos  
maiores que o 20

20 variável  
temporária

30    47    \_\_\_\_\_    63    101    115

Enquanto procuramos,  
deslocamos os elementos  
maiores que o 20

20 variável  
temporária

30              47    63    101    115

Enquanto procuramos,  
deslocamos os elementos  
maiores que o 20

20

variável  
temporária

30

47

63

101

115

Enquanto procuramos,  
deslocamos os elementos  
maiores que o 20





Encontramos a  
posição do 20

20    30    47            63    101    115

Encontramos a  
posição do 20

### Ordenação por **Inserção**

#### **Insertion-Sort**

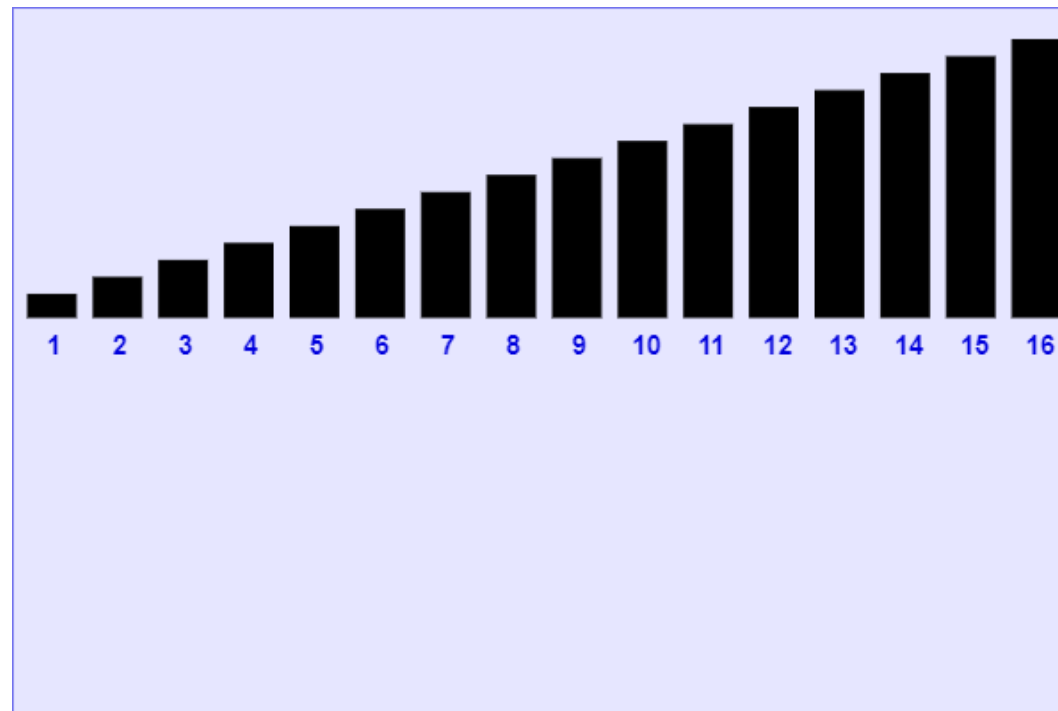
Método **ineficiente** mas **estável**

Os **itens à esquerda** estarão sempre **ordenados**

Vantagem

Caso o **vetor** esteja **ordenado**, haverá o **mínimo** de **comparações** e **movimentos** entre os dados

6 5 3 1 8 7 2 4



• Sendo  $M_i(n) = C_i(n) + 1$ , no **melhor caso**, temos:

- $C(n) = 1 + 1 + 1 + \dots + 1$ ,  $n-1$  vezes  $= (n-1)$
- $M(n) = 2 + 2 + 2 + \dots + 2$ ,  $n-1$  vezes  $= 2(n-1) = \Theta(n)$
- Sendo  $M_i(n) = C_i(n) + 1$ , no **pior caso**, temos:

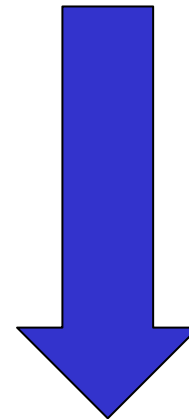
$$M(n) = -1 + \sum_{0 \leq i \leq n} i = \frac{n(n+1) - 2}{2} = \Theta(n^2)$$

## Ideia Básica

- Triplicamos o número de *arrays* (entrada, contagem e saída)

**Array de entrada**  
(a ser ordenado)

0	1	2	3	4	5	6	7



**Array de contagem**  
(mapeamento de elementos)

**Array de saída**  
(ordenado)

0	1	2	3	4	5	6	7

# Ideia Básica

- Cada posição do contagem armazena o número de elementos menores ou iguais a ela no entrada. Por exemplo, se a entrada tem 3 zeros, 1 um e 2 dois, então o contagem tem 3, 4 e 6, respectivamente

## Array de entrada

0		a	b	c		d	e
1	...	2	0	2	...	0	0

## Array de contagem

0	1	2	3	4	5
3	4	6			

***Array de entrada***

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

O *array* de contagem terá seis posições (0 à 5)

O *array* de saída terá oito posições



**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5

**Array de saída**

0	1	2	3	4	5	6	7

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	0	0	0	0	0

Inicializar todas as posições  
do *array* de contagem com zero

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	0	0	0	0	0

Para cada elemento do *array* de entrada,  
incrementá-lo no de contagem

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	0	1	0	0	0

Para cada elemento do *array* de entrada,  
incrementá-lo no de contagem

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	0	1	0	0	1

Para cada elemento do *array* de entrada,  
incrementá-lo no de contagem

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	0	1	1	0	1

Para cada elemento do *array* de entrada,  
incrementá-lo no de contagem

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
1	0	1	1	0	1

Para cada elemento do *array* de entrada,  
incrementá-lo no de contagem



**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
1	0	2	1	0	1

Para cada elemento do *array* de entrada,  
incrementá-lo no de contagem

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
1	0	2	2	0	1

Para cada elemento do *array* de entrada,  
incrementá-lo no de contagem

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	0	2	2	0	1

Para cada elemento do *array* de entrada,  
incrementá-lo no de contagem

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	0	2	3	0	1

Para cada elemento do *array* de entrada,  
incrementá-lo no de contagem

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	0	2	3	0	1

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	0	2	3	0	1

Fazer com que o *array* de contagem seja acumulativo de tal forma que cada posição  $i$  armazene o número de elementos menores ou iguais a  $i$

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	0	2	3	0	1

Fazer com que o *array* de contagem seja acumulativo de tal forma que cada posição  $i$  armazene o número de elementos menores ou iguais a  $i$

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	2	2	3	0	1

Fazer com que o *array* de contagem seja acumulativo de tal forma que cada posição  $i$  armazene o número de elementos menores ou iguais a  $i$



**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	2	4	3	0	1

Fazer com que o *array* de contagem seja acumulativo de tal forma que cada posição  $i$  armazene o número de elementos menores ou iguais a  $i$

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	2	4	7	0	1

Fazer com que o *array* de contagem seja acumulativo de tal forma que cada posição  $i$  armazene o número de elementos menores ou iguais a  $i$

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	2	4	7	7	1

Fazer com que o *array* de contagem seja acumulativo de tal forma que cada posição  $i$  armazene o número de elementos menores ou iguais a  $i$

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	2	4	7	7	8

Fazer com que o *array* de contagem seja acumulativo de tal forma que cada posição  $i$  armazene o número de elementos menores ou iguais a  $i$

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	2	4	7	7	8

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	2	4	7	7	8

**Array de saída**

0	1	2	3	4	5	6	7

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	2	4	7	7	8

**Array de saída**

0	1	2	3	4	5	6	7

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	2	4	7	7	8

7ª posição

**Array de saída**

0	1	2	3	4	5	6	7



Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	2	4	7	7	8

7ª posição

**Array de saída**

0	1	2	3	4	5	6	7
						3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	2	4	6	7	8

Atualizar *array* de contagem

**Array de saída**

0	1	2	3	4	5	6	7
						3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	2	4	6	7	8

**Array de saída**

0	1	2	3	4	5	6	7
						3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	2	4	6	7	8

2ª posição

**Array de saída**

0	1	2	3	4	5	6	7
						3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
2	2	4	6	7	8

2ª posição

**Array de saída**

0	1	2	3	4	5	6	7
	0					3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
1	2	4	6	7	8

Atualizar *array* de  
contagem

**Array de saída**

0	1	2	3	4	5	6	7
	0					3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
1	2	4	6	7	8

**Array de saída**

0	1	2	3	4	5	6	7
	0					3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
1	2	4	6	7	8

6ª posição

**Array de saída**

0	1	2	3	4	5	6	7
	0					3	



Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
1	2	4	6	7	8

6ª posição

**Array de saída**

0	1	2	3	4	5	6	7
	0				3	3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
1	2	4	5	7	8

Atualizar *array* de contagem

**Array de saída**

0	1	2	3	4	5	6	7
	0				3	3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
1	2	4	5	7	8

**Array de saída**

0	1	2	3	4	5	6	7
	0				3	3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
1	2	4	5	7	8

**Array de saída**

0	1	2	3	4	5	6	7
	0				3	3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
1	2	4	5	7	8

4ª posição

**Array de saída**

0	1	2	3	4	5	6	7
	0		2		3	3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
1	2	3	5	7	8

Atualizar *array* de contagem

**Array de saída**

0	1	2	3	4	5	6	7
	0		2		3	3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
1	2	3	5	7	8

**Array de saída**

0	1	2	3	4	5	6	7
	0		2		3	3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
1	2	3	5	7	8

1ª posição



**Array de saída**

0	1	2	3	4	5	6	7
	0		2		3	3	



Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
1	2	3	5	7	8

1ª posição



**Array de saída**

0	1	2	3	4	5	6	7
0	0		2		3	3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	2	3	5	7	8

Atualizar array de contagem

**Array de saída**

0	1	2	3	4	5	6	7
0	0		2		3	3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	2	3	5	7	8

**Array de saída**

0	1	2	3	4	5	6	7
0	0		2		3	3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	2	3	5	7	8

**Array de saída**

0	1	2	3	4	5	6	7
0	0		2		3	3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	2	3	5	7	8

5ª posição

**Array de saída**

0	1	2	3	4	5	6	7
0	0		2		3	3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	2	3	5	7	8

5ª posição

**Array de saída**

0	1	2	3	4	5	6	7
0	0		2	3	3	3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	2	3	4	7	8

Atualizar array de contagem

**Array de saída**

0	1	2	3	4	5	6	7
0	0		2	3	3	3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	2	3	4	7	8

**Array de saída**

0	1	2	3	4	5	6	7
0	0		2	3	3	3	



Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	2	3	4	7	8

**Array de saída**

0	1	2	3	4	5	6	7
0	0		2	3	3	3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	2	3	4	7	8

8ª posição

**Array de saída**

0	1	2	3	4	5	6	7
0	0		2	3	3	3	

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	2	3	4	7	8

8ª posição

**Array de saída**

0	1	2	3	4	5	6	7
0	0		2	3	3	3	5

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	2	3	4	7	7

Atualizar array de contagem

**Array de saída**

0	1	2	3	4	5	6	7
0	0		2	3	3	3	5

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	2	3	4	7	7

**Array de saída**

0	1	2	3	4	5	6	7
0	0		2	3	3	3	5

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	2	3	4	7	7

**Array de saída**

0	1	2	3	4	5	6	7
0	0		2	3	3	3	5

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	2	3	4	7	7

3ª posição



**Array de saída**

0	1	2	3	4	5	6	7
0	0		2	3	3	3	5

Preencher o *array* de saída, copiando os elementos da entrada de trás para frente nas suas respectivas posições

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	2	3	4	7	7

3ª posição



**Array de saída**

0	1	2	3	4	5	6	7
0	0	2	2	3	3	3	5



## Exercício Resolvido (1)

- Em nosso exemplo, o algoritmo terminou sua execução?

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

**Array de contagem**

0	1	2	3	4	5
0	2	3	4	7	7

3ª posição



**Array de saída**

0	1	2	3	4	5	6	7
0	0	2	2	3	3	3	5

## Exercício Resolvido (1)

- Em nosso exemplo, o algoritmo terminou sua execução?

**Array de entrada**

0	1	2	3	4	5	6	7
2	5	3	0	2	3	0	3

Falso, pois ainda precisamos atualizar o array de contagem

**Array de contagem**

0	1	2	3	4	5
0	2	2	4	7	7

Atualizar array de contagem

**Array de saída**

0	1	2	3	4	5	6	7
0	0	2	2	3	3	3	5

## Exercício Resolvido (3)

- O Counting Sort pode ser aplicado adequadamente na ordenação de strings e números reais?

Falso. No caso das *strings*, temos um problema combinatório para identificar a posição de cada *string* no *array* de Contagem. No caso dos números reais, temos infinitos valores entre dois números inteiros.

## Exercício Resolvido (4)

- Nosso dinheiro é um número real. Conseguimos utilizar adequadamente o Counting Sort para ordenar valores financeiros?

## Exercício Resolvido (4)

- Nosso dinheiro é um número real. Conseguimos utilizar adequadamente o Counting Sort para ordenar valores financeiros?

Verdadeiro. Basta multiplicarmos os valores por cem e considerar somente a parte inteira para a ordenação. No final, basta dividir os valores ordenados por cem (considere a divisão no ambiente de números reais).

# Algoritmo em Java

```
public static int[] countSort(int[] inputArray) {  
    int N = inputArray.length;  
    int M = 0;  
  
    for (int i = 0; i < N; i++) {  
        M = Math.max(M, inputArray[i]);  
    }  
  
    int[] countArray = new int[M + 1];  
  
    for (int i = 0; i < N; i++) {  
        countArray[inputArray[i]]++;  
    }  
  
    for (int i = 1; i <= M; i++) {  
        countArray[i] += countArray[i - 1];  
    }  
  
    int[] outputArray = new int[N];  
    for (int i = N - 1; i >= 0; i--) {  
        outputArray[countArray[inputArray[i]] - 1] = inputArray[i];  
        countArray[inputArray[i]]--;  
    }  
  
    return outputArray;  
}
```

# Análise das Operações com Elementos do *Array*

- Inicializar todas as posições do *array* de contagem com zero
- Para cada elemento do *array* de entrada, incrementá-lo no de contagem
- Fazer com que o *array* de contagem seja acumulativo de tal forma que cada posição  $i$  armazene o número de elementos menores ou iguais a  $i$
- Sabendo o número de elementos menores ou iguais a  $i$ , preencher o *array* de saída

# Análise das Operações com Elementos do *Array*

- Inicializar todas as posições do *array* de contagem com zero  $\Theta(n)$
- Para cada elemento do *array* de entrada, incrementá-lo no de contagem
- Fazer com que o *array* de contagem seja acumulativo de tal forma que cada posição  $i$  armazene o número de elementos menores ou iguais a  $i$
- Sabendo o número de elementos menores ou iguais a  $i$ , preencher o *array* de saída



# Análise das Operações com Elementos do *Array*

- Inicializar todas as posições do *array* de contagem com zero  $\Theta(n)$
- Para cada elemento do *array* de entrada, incrementá-lo no de contagem  $\Theta(n)$
- Fazer com que o *array* de contagem seja acumulativo de tal forma que cada posição  $i$  armazene o número de elementos menores ou iguais a  $i$
- Sabendo o número de elementos menores ou iguais a  $i$ , preencher o *array* de saída

# Análise das Operações com Elementos do *Array*

- Inicializar todas as posições do *array* de contagem com zero  $\Theta(n)$
- Para cada elemento do *array* de entrada, incrementá-lo no de contagem  $\Theta(n)$
- Fazer com que o *array* de contagem seja acumulativo de tal forma que cada posição  $i$  armazene o número de elementos menores ou iguais a  $i$   $\Theta(n)$
- Sabendo o número de elementos menores ou iguais a  $i$ , preencher o *array* de saída

# Análise das Operações com Elementos do *Array*

- Inicializar todas as posições do *array* de contagem com zero  $\Theta(n)$
- Para cada elemento do *array* de entrada, incrementá-lo no de contagem  $\Theta(n)$
- Fazer com que o *array* de contagem seja acumulativo de tal forma que cada posição  $i$  armazene o número de elementos menores ou iguais a  $i$   $\Theta(n)$
- Sabendo o número de elementos menores ou iguais a  $i$ , preencher o *array* de saída  $\Theta(n)$

# Análise das Operações com Elementos do *Array*

- Análise da complexidade para operações com elementos do array:

$$\Theta(n) + \Theta(n) + \Theta(n) + \Theta(n) = \Theta(n)$$

# *Funcionamento Básico*

- Os métodos de ordenação apresentados comparam as chaves de pesquisa como um todo
- Outra opção é comparar as chaves por parte. Por exemplo, em uma lista de nomes, ordenamos os mesmos pelas primeiras letras

# *Ideia Básica*

- Para cada caractere da chave primária (do menos para o mais significativo), ordene as chaves

<i>array</i>
329
457
657
839
436
720
355

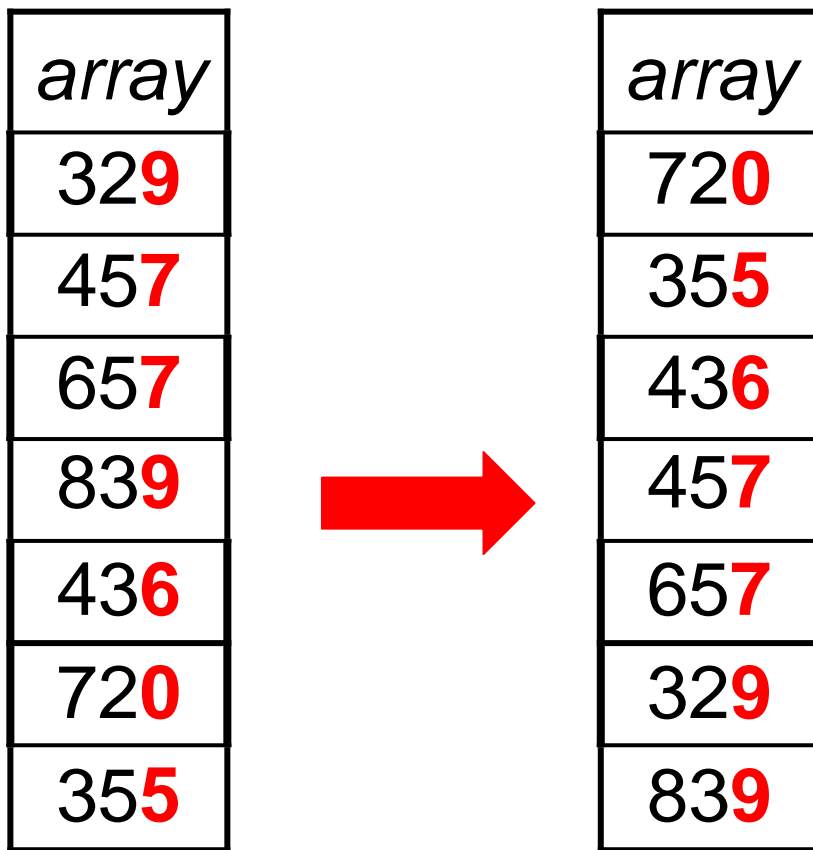
# *Ideia Básica*

- Para cada caractere da chave primária (do menos para o mais significativo), ordene as chaves

<i>array</i>
32 <b>9</b>
45 <b>7</b>
65 <b>7</b>
83 <b>9</b>
43 <b>6</b>
72 <b>0</b>
35 <b>5</b>

# Ideia Básica

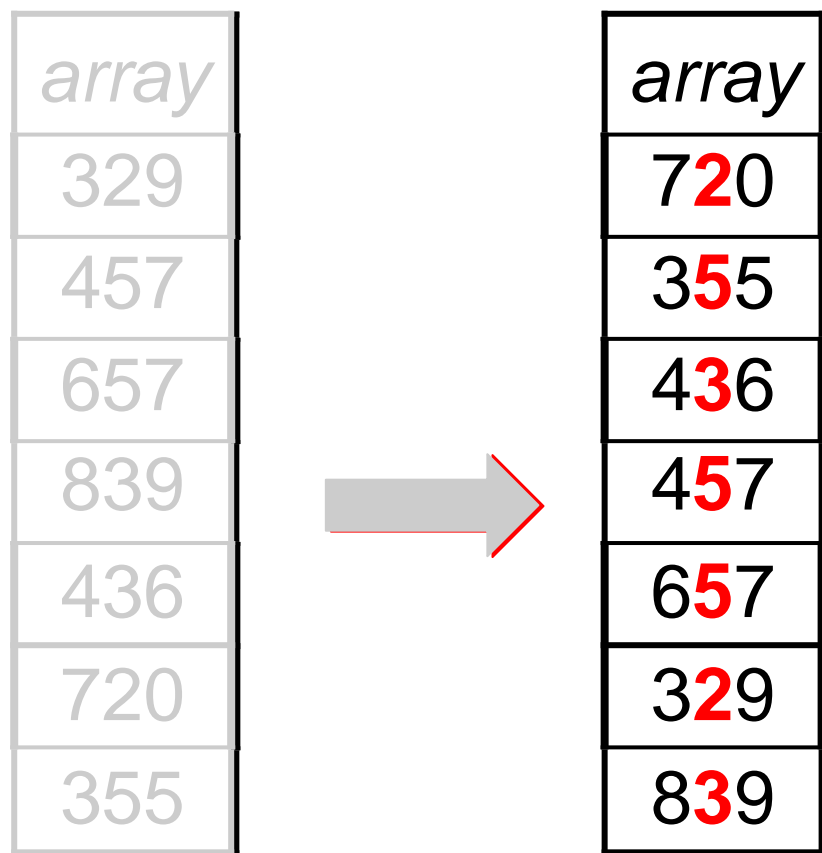
- Para cada caractere da chave primária (do menos para o mais significativo), ordene as chaves





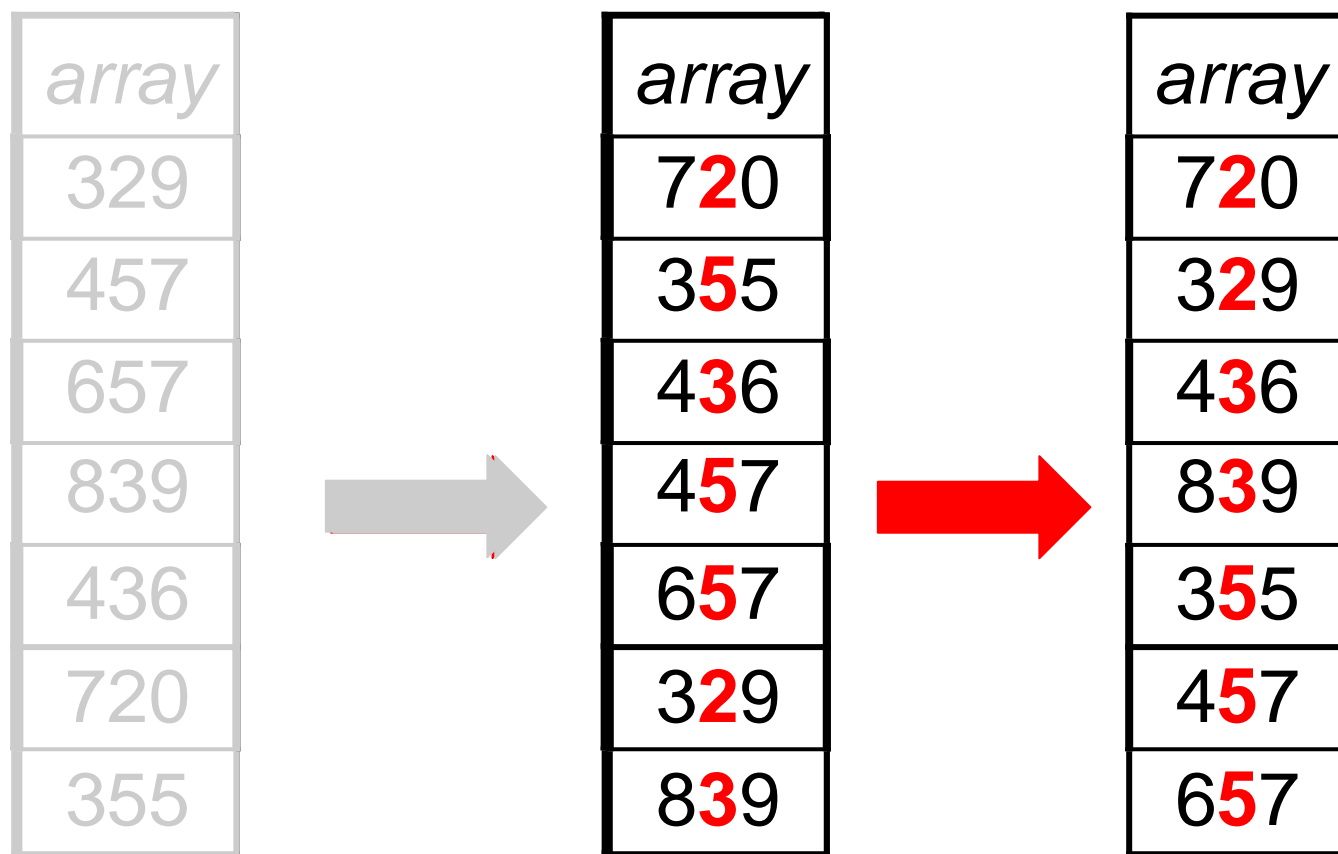
# Ideia Básica

- Para cada caractere da chave primária (do menos para o mais significativo), ordene as chaves

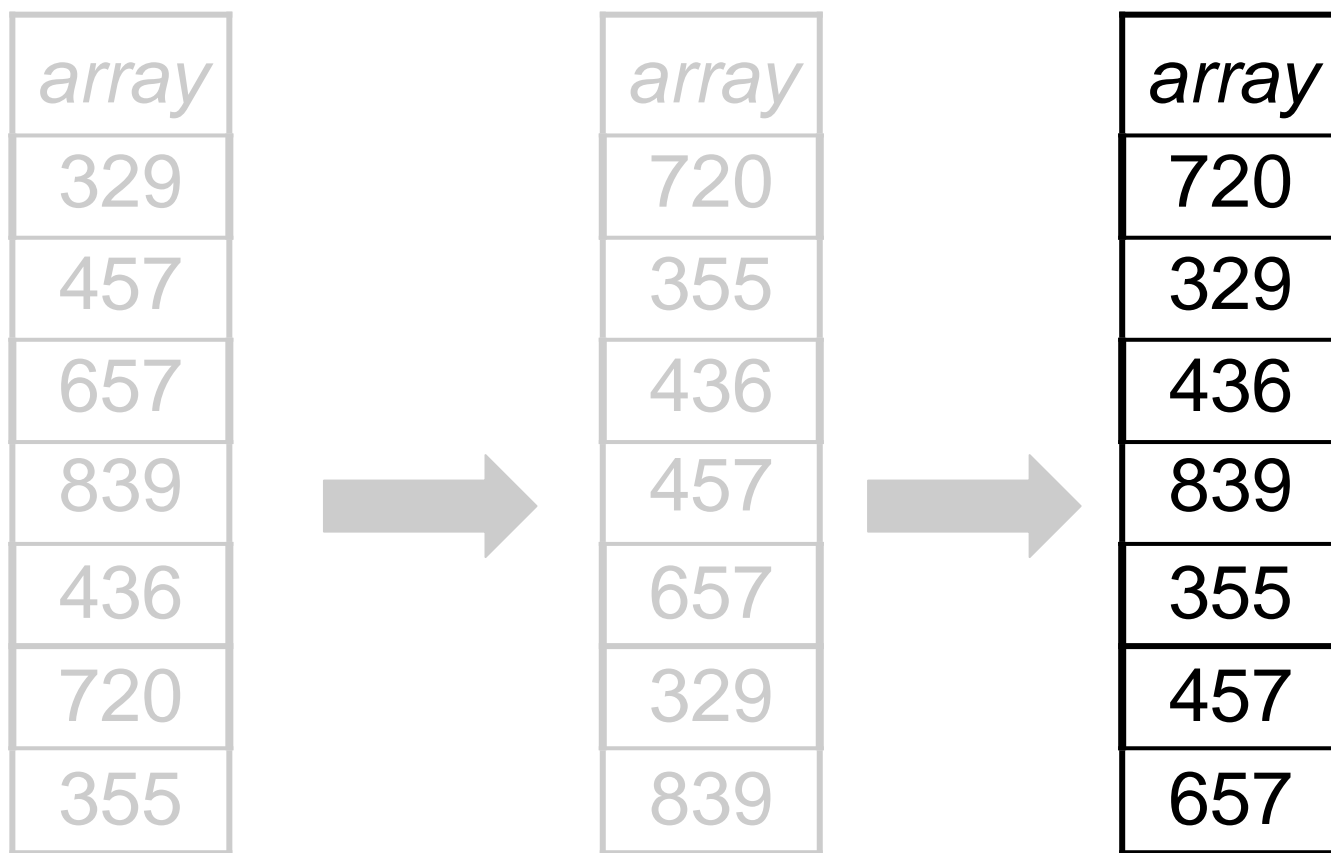


# Ideia Básica

- Para cada caractere da chave primária (do menos para o mais significativo), ordene as chaves

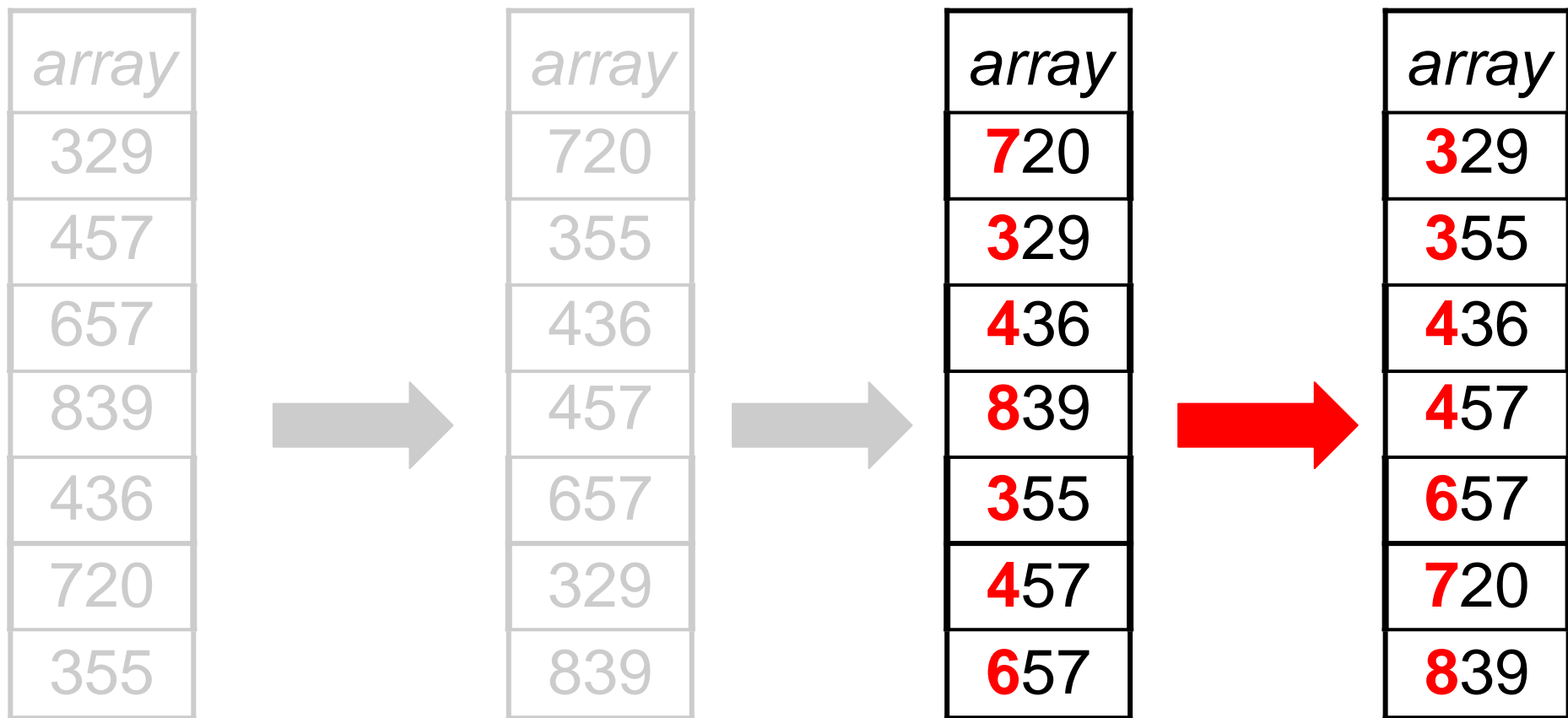


- Para cada caractere da chave primária (do menos para o mais significativo), ordene as chaves



# Ideia Básica

- Para cada caractere da chave primária (do menos para o mais significativo), ordene as chaves



# Próxima aula

Métodos de Ordenação Eficientes:

- Método Quick-Sort
- Método Shell-Sort
- Método Merge-Sort
- Método Heap-Sort

Conclusão dos métodos de Ordenação