

# Unidade V:


## Ordenação Interna - Mergesort



**PUC Minas**

Instituto de Ciências Exatas e Informática  
Departamento de Ciência da Computação

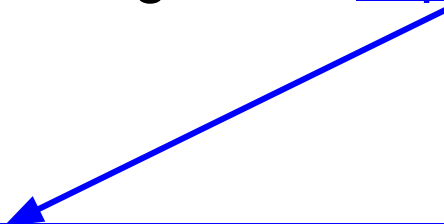
- Funcionamento básico
- Algoritmo em C#
- Análise dos número de comparações e movimentações
- Conclusão

- **Funcionamento básico** 
- Algoritmo em C#
- Análise dos número de comparações e movimentações
- Conclusão

# Introdução

- Ordenação por intercalação
- Algoritmo de ordenação do tipo dividir para conquistar
- Normalmente, implementado de forma recursiva e demandando um espaço adicional de memória (não é um algoritmo *in-place*)

- Ordenação por intercalação
- Algoritmo de ordenação do tipo dividir para conquistar
- Normalmente, implementado de forma recursiva e demandando um espaço adicional de memória (não é um algoritmo *in-place*)



Um **algoritmo** de ordenação é ***in-place*** se a memória adicional requerida é independente do tamanho do *array*

# Funcionamento Básico

- Dividir sistematicamente o *array* em *subarrays* até que os mesmos tenham tamanho um
- Conquistar através da intercalação (ordenada) sistemática de dois em dois *subarrays*

# Exercício Resolvido (1)

- Faça a intercalação (ordenada) dos dois vetores [1 2 3 4 9] e [3 5 6 7 8]

# Exercício Resolvido (1)

- Faça a intercalação (ordenada) dos dois vetores [1 2 3 4 9] e [3 5 6 7 8]

[ ]





# Exercício Resolvido (1)

- Faça a intercalação (ordenada) dos dois vetores [4 2 3 4 9] e [3 5 6 7 8]

[ ]

[ 1 ]



# Exercício Resolvido (1)

- Faça a intercalação (ordenada) dos dois vetores [~~1~~ 2 3 4 9] e [3 5 6 7 8]

[ ]

[ 1 ]

[ 1 2 ]



# Exercício Resolvido (1)

- Faça a intercalação (ordenada) dos dois vetores [~~1~~ 2 3 4 9] e [~~3~~ 5 6 7 8]

[ ]

[ 1 ]

[ 1 2 ]

[ 1 2 3 ]



# Exercício Resolvido (1)

- Faça a intercalação (ordenada) dos dois vetores [~~1~~ ~~2~~ ~~3~~ 4 9] e [~~3~~ 5 6 7 8]

[ ]

[ 1 ]

[ 1 2 ]

[ 1 2 3 ]

[ 1 2 3 3 ]



## Exercício Resolvido (1)

- Faça a intercalação (ordenada) dos dois vetores [~~1~~ ~~2~~ ~~3~~ ~~4~~ 9] e [~~3~~ 5 6 7 8]

[ ]

[ 1 ]

[ 1 2 ]

[ 1 2 3 ]

[ 1 2 3 3 ]

[ 1 2 3 3 4 ]



## Exercício Resolvido (1)

- Faça a intercalação (ordenada) dos dois vetores [~~1~~ ~~2~~ ~~3~~ ~~4~~ 9] e [~~3~~ ~~5~~ 6 7 8]

[ ]

[ 1 ]

[ 1 2 ]

[ 1 2 3 ]

[ 1 2 3 3 ]

[ 1 2 3 3 4 ]

[ 1 2 3 3 4 5 ]



## Exercício Resolvido (1)

- Faça a intercalação (ordenada) dos dois vetores [~~1~~ ~~2~~ ~~3~~ ~~4~~ 9] e [~~3~~ ~~5~~ ~~6~~ 7 8]

[ ]

[ 1 ]

[ 1 2 ]

[ 1 2 3 ]

[ 1 2 3 3 ]

[ 1 2 3 3 4 ]

[ 1 2 3 3 4 5 ]

[ 1 2 3 3 4 5 6 ]



## Exercício Resolvido (1)

- Faça a intercalação (ordenada) dos dois vetores [~~1~~ ~~2~~ ~~3~~ ~~4~~ 9] e [~~3~~ ~~5~~ ~~6~~ ~~7~~ 8]

[ ]

[ 1 ]

[ 1 2 ]

[ 1 2 3 ]

[ 1 2 3 3 ]

[ 1 2 3 3 4 ]

[ 1 2 3 3 4 5 ]

[ 1 2 3 3 4 5 6 ]

[ 1 2 3 3 4 5 6 7 ]





## Exercício Resolvido (1)

- Faça a intercalação (ordenada) dos dois vetores [~~1~~ ~~2~~ ~~3~~ ~~4~~ 9] e [~~3~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~]

[ ]

[ 1 ]

[ 1 2 ]

[ 1 2 3 ]

[ 1 2 3 3 ]

[ 1 2 3 3 4 ]

[ 1 2 3 3 4 5 ]

[ 1 2 3 3 4 5 6 ]

[ 1 2 3 3 4 5 6 7 ]

[ 1 2 3 3 4 5 6 7 8 ]



## Exercício Resolvido (1)

- Faça a intercalação (ordenada) dos dois vetores ~~[1 2 3 4 9]~~ e ~~[3 5 6 7 8]~~

[ ]

[ 1 ]

[ 1 2 ]

[ 1 2 3 ]

[ 1 2 3 3 ]

[ 1 2 3 3 4 ]

[ 1 2 3 3 4 5 ]

[ 1 2 3 3 4 5 6 ]

[ 1 2 3 3 4 5 6 7 ]

[ 1 2 3 3 4 5 6 7 8 ]

[ 1 2 3 3 4 5 6 7 8 9 ]



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----



Dividir



Conquistar (intercalação ou *merge*)



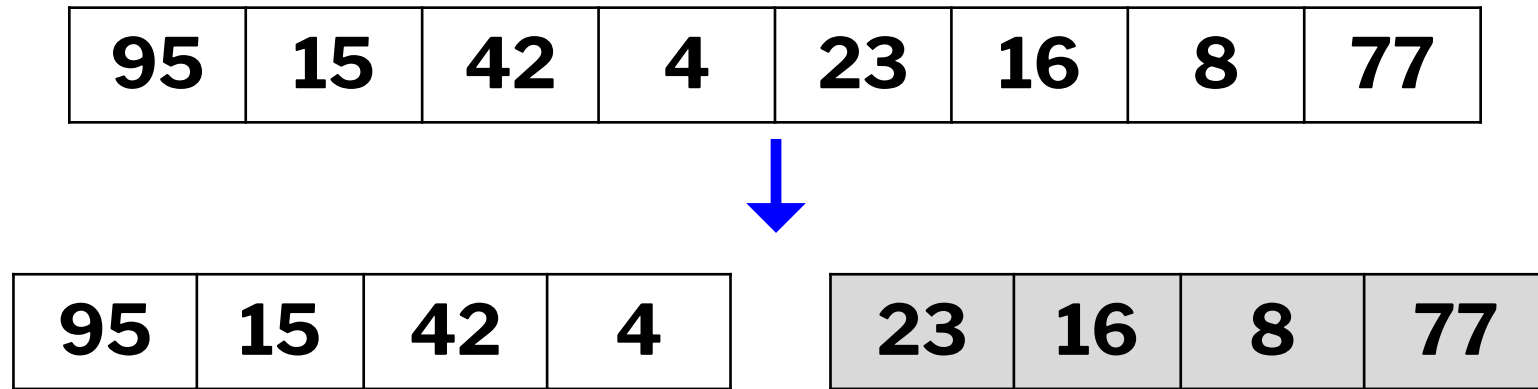
*Subarray* aguardando divisão



Elemento a ser intercalado no *subarray*

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----





95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----



95	15	42	4
----	----	----	---

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

95	15
----	----

42	4
----	---





95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

95	15
----	----

42	4
----	---



95	15
----	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

--	--

42	4
----	---

95	15
----	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	
----	--

42	4
----	---

95	<del>15</del>
----	---------------



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----


95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	95
----	----

42	4
----	---

<del>95</del>	<del>15</del>
---------------	---------------



## Exemplo

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	95
----	----

42	4
----	---

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	95
----	----

42	4
----	---



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	95
----	----

42	4
----	---



42	4
----	---

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	95
----	----

--	--



42	4
----	---



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	95
----	----

4	
---	--



42	<del>4</del>
----	--------------

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	95
----	----

4	42
---	----

<del>42</del>	<del>4</del>
---------------	--------------



## Exemplo

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	95
----	----

4	42
---	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

--	--	--	--

23	16	8	77
----	----	---	----

15	95	4	42
----	----	---	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4			
---	--	--	--

23	16	8	77
----	----	---	----

15	95	<del>4</del>	42
----	----	--------------	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15		
---	----	--	--

23	16	8	77
----	----	---	----



<del>15</del>	95	<del>4</del>	42
---------------	----	--------------	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	
---	----	----	--

23	16	8	77
----	----	---	----



<del>15</del>	95	<del>4</del>	<del>42</del>
---------------	----	--------------	---------------

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----



<del>15</del>	<del>95</del>
---------------	---------------

<del>4</del>	<del>42</del>
--------------	---------------



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----



23	16	8	77
----	----	---	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

23	16
----	----

8	77
---	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

23	16
----	----

8	77
---	----



23	16
----	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

--	--

8	77
---	----



23	16
----	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	
----	--

8	77
---	----



23	<del>16</del>
----	---------------

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	23
----	----

8	77
---	----



<del>23</del>	<del>16</del>
---------------	---------------



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	23
----	----

8	77
---	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	23
----	----

8	77
---	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	23
----	----

8	77
---	----



8	77
---	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	23
----	----

--	--

8	77
---	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	23
----	----

8	
---	--

<del>8</del>	77
--------------	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	23
----	----

8	77
---	----

<del>8</del>	<del>77</del>
--------------	---------------



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	23
----	----

8	77
---	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

--	--	--	--



16	23
----	----

8	77
---	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

8			
---	--	--	--

16	23
----	----

<del>8</del>	77
--------------	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

8	16		
---	----	--	--

<del>16</del>	23
---------------	----

<del>8</del>	77
--------------	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

8	16	23	
---	----	----	--

<del>16</del>	<del>23</del>	<del>8</del>	77
---------------	---------------	--------------	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

8	16	23	77
---	----	----	----

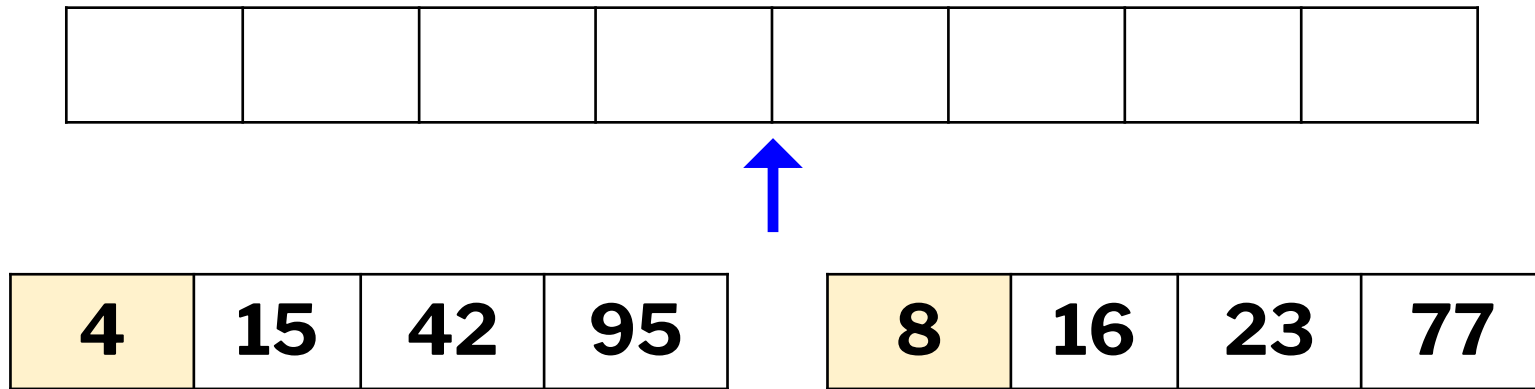


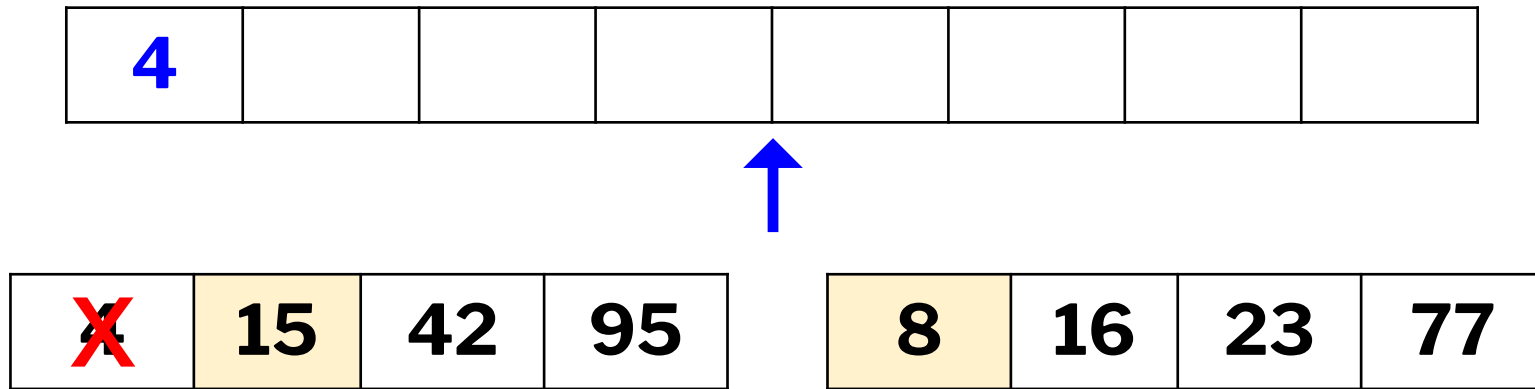
<del>16</del>	<del>23</del>	<del>8</del>	<del>77</del>
---------------	---------------	--------------	---------------

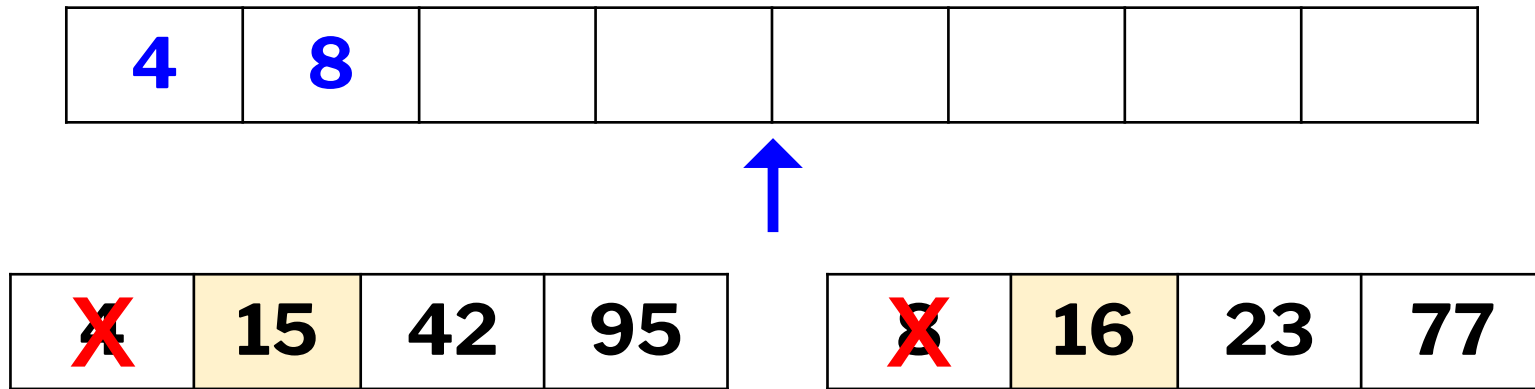
<b>95</b>	<b>15</b>	<b>42</b>	<b>4</b>	<b>23</b>	<b>16</b>	<b>8</b>	<b>77</b>
-----------	-----------	-----------	----------	-----------	-----------	----------	-----------

<b>4</b>	<b>15</b>	<b>42</b>	<b>95</b>
----------	-----------	-----------	-----------

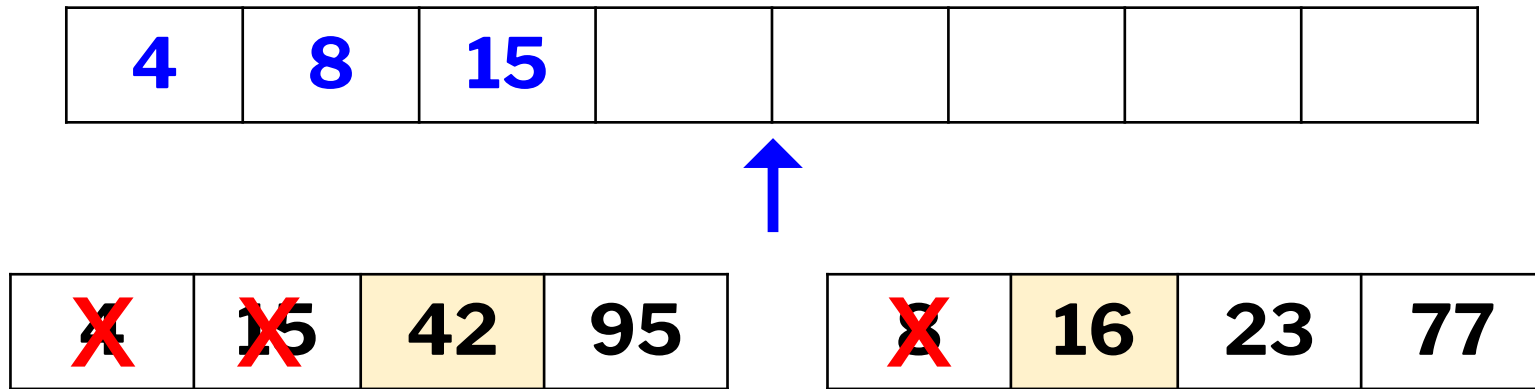
<b>8</b>	<b>16</b>	<b>23</b>	<b>77</b>
----------	-----------	-----------	-----------

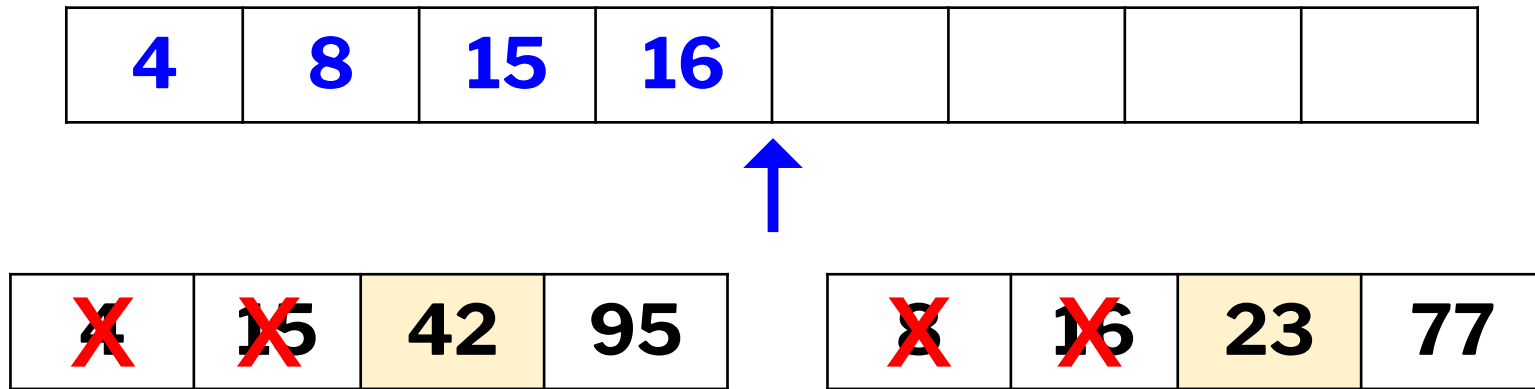


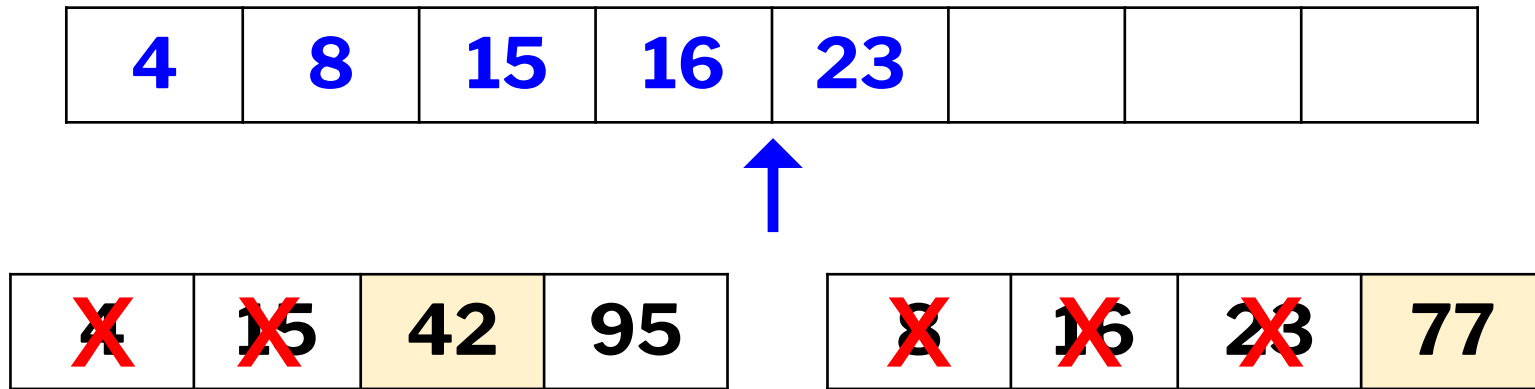


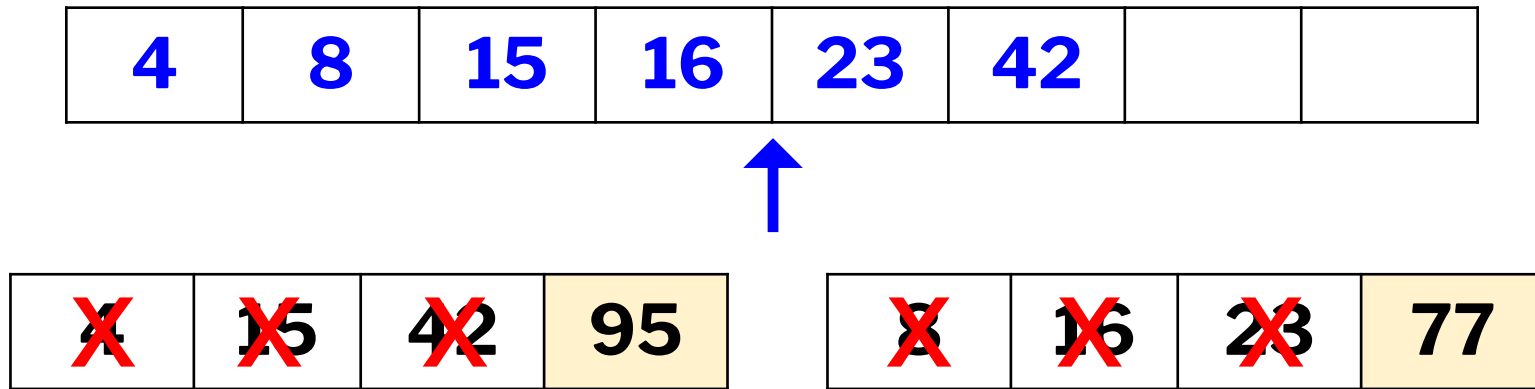


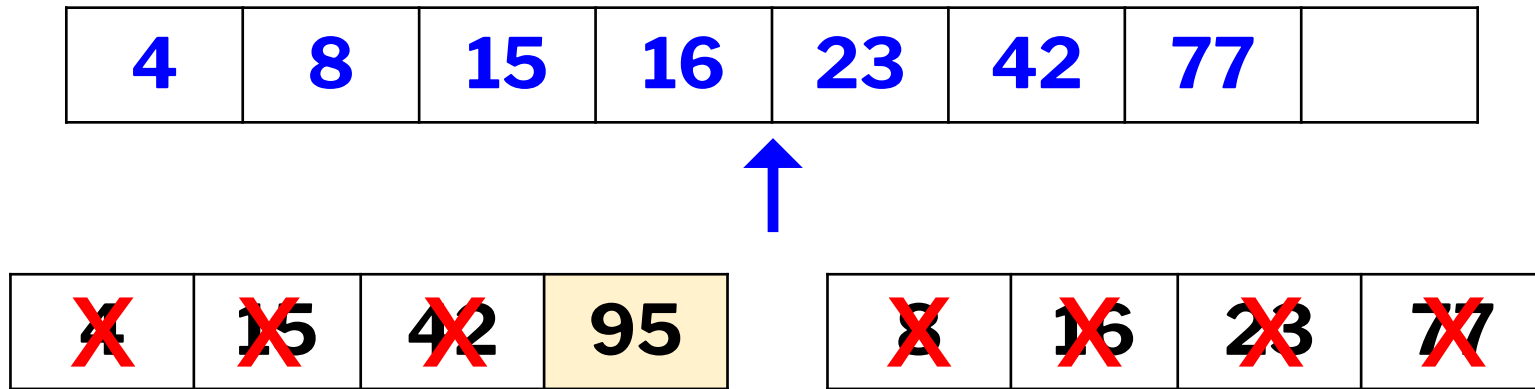


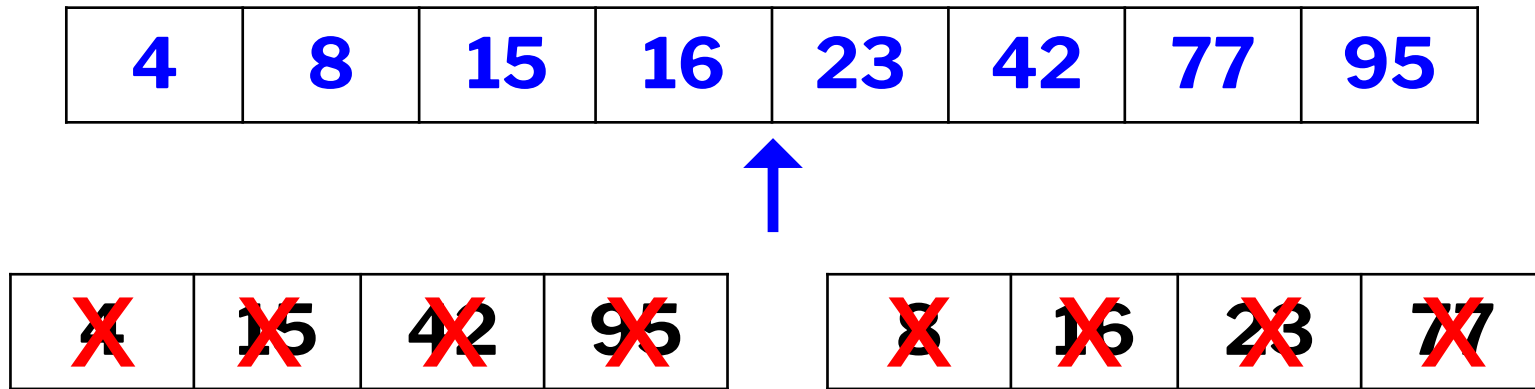













<b>4</b>	<b>8</b>	<b>15</b>	<b>16</b>	<b>23</b>	<b>42</b>	<b>77</b>	<b>95</b>
----------	----------	-----------	-----------	-----------	-----------	-----------	-----------

- Funcionamento básico
- **Algoritmo em C#** 
- Análise dos número de comparações e movimentações
- Conclusão



## Algoritmo em C#

```
void mergesort(int esq, int dir) {  
    if (esq < dir){  
        int meio = (esq + dir) / 2;  
        mergesort(esq, meio);  
        mergesort(meio + 1, dir);  
        intercalar(esq, meio, dir);  
    }  
}
```

## Algoritmo em C#

```
void intercalar(int esq, int meio, int dir){  
    //Definir tamanho dos dois subarrays  
    int nEsq = (meio+1)-esq;  
    int nDir = dir - meio;  
  
    int[] arrayEsq = new int[nEsq+1];  
    int[] arrayDir = new int[nDir+1];  
  
    //Sentinela no final dos dois arrays  
    arrayEsq[nEsq] = arrayDir[nDir] = 0x7FFFFFFF;
```

■ ■ ■

## Algoritmo em C#


■ ■ ■

```
int iEsq, iDir, i;

//Inicializar primeiro subarray
for (iEsq = 0; iEsq < nEsq; iEsq++){
    arrayEsq[iEsq] = array[esq+iEsq];
}

//Inicializar segundo subarray
for (iDir = 0; iDir < nDir; iDir++){
    arrayDir[iDir] = array[(meio+1)+iDir];
}

//Intercalacao propriamente dita
for (iEsq = iDir = 0, i = esq; i <= dir; i++){
    array[i] = (arrayEsq[iEsq] <= arrayDir[iDir]) ? arrayEsq[iEsq++] : arrayDir[iDir++];
}
}
```

- Funcionamento básico
- Algoritmo em C#
- **Análise dos número de comparações e movimentações** 
- Conclusão

# Análise do Número de Comparações


- Todos os casos:
  - Em cada *subarray* (tamanho  $k$ ), fazemos  $k - 1$  comparações
  - Supondo que o tamanho do *array* é uma potência de 2, fazemos  $\lg(n)$  passos

$$\begin{cases} C(1) = 0 \\ C(n) = 2C(n/2) + \Theta(n) \end{cases} \quad \Theta(n * \lg(n))$$

# Análise do Número de Movimentações

- Todos os casos:
  - Movimentamos os elementos de cada subarray duas vezes

$$\begin{cases} M(1) = 0 \\ M(n) = 2M(n/2) + \Theta(n) \end{cases} \quad \Theta(n \lg(n))$$

- Funcionamento básico
- Algoritmo em C#
- Análise dos número de comparações e movimentações
- **Conclusão** 

# Conclusão

- Método estável
- Normalmente, implementado de forma recursiva e demandando memória adicional
- Faz  $\Theta(n \lg(n))$  comparações nos três casos (melhor, médio e pior)



## Exercício (1)

- Mostre todas as comparações e movimentações do algoritmo anterior para o *array* abaixo:

12	4	8	2	14	17	6	18	10	16	15	5	13	9	1	11	7	3
----	---	---	---	----	----	---	----	----	----	----	---	----	---	---	----	---	---