

# Classificação e Pesquisa de Dados

Cristiano Santos

*cristiano.santos@amf.edu.br*



# Counting Sort

# Counting Sort

- O Counting Sort (**Ordenação por Contagem**) é um algoritmo de ordenação de inteiros criado por Harold Seward **em 1954**.
- Ele é um **algoritmo de Programação Dinâmica**, que funciona construindo um **histograma cumulativo das chaves** do arranjo de entrada e o usando essa estrutura para construir um arranjo de saída ordenado.

# Counting Sort

- Características:
  - Um algoritmo iterativo.
  - Um algoritmo de **ordenação de inteiros**.
  - Um algoritmo de **ordenação estável**.
  - O desempenho é pouco sensível aos dados de entrada.
  - Necessita de uma **estrutura de dados auxiliar para funcionar**.

# Counting Sort

- Como é possível ordenar elementos sem utilizar comparação?

# Counting Sort

- Em geral, a ideia é valer-se do fato de que estamos ordenando números inteiros e que os índices dos arrays também são inteiros.

# Counting Sort

- Em geral, a ideia é valer-se do fato de que estamos ordenando números inteiros e que os índices dos arrays também são inteiros.
- Podemos mapear o valor presente em uma sequência para a posição de mesmo valor em um array auxiliar ( $\text{array}[i] = i$ ). Essa é a estratégia geral dos algoritmos de ordenação linear que se baseiam na contagem dos elementos da sequência a ser ordenada.

# Counting Sort

- **Antes de analisarmos os algoritmos de contagem em detalhes, vamos abordar um exemplo bem simples para entender esse conceito.** Para isso, vamos entrar em um mundo em que:



# Counting Sort

- **Antes de analisarmos os algoritmos de contagem em detalhes, vamos abordar um exemplo bem simples para entender esse conceito.** Para isso, vamos entrar em um mundo em que:
  - todos os **elementos do array** que vamos ordenar **são inteiros positivos** (1, 2, 3...k);
  - **não há repetição** de elementos no array que vamos ordenar;
  - **sabemos o maior valor desse array, o qual chamamos de k.**

# Counting Sort

- Desse modo, se quisermos ordenar o array  $A=[7,2,1,4]$
- Criarmos um array auxiliar C cujo tamanho é  $k$ , onde  $k$  é o maior elemento do array original, e iterarmos sobre  $A$
- Registramos a presença de seus elementos em  $C$  através da seguinte instrução  $C[A[i] - 1] = \text{true}$ .
- O índice é subtraído de 1, pois as posições de um array iniciam-se de 0

# Counting Sort

- Exemplo:

```
... boolean[] C = new boolean[k]; // registrando a presença de A[i] na sequência  
  
for (int i = 0; i < A.length; i++)  
    { C[A[i] - 1] = true; } ...
```

# Counting Sort

- **Exemplo:**

Se  $A=[7,2,1,4]$ , com  $k=7$ , temos  $C$  preenchido da seguinte maneira:

$C = [\text{true}, \text{true}, \text{false}, \text{true}, \text{false}, \text{false}, \text{true}]$

# Counting Sort

- **Exemplo:**

Se  $A=[7,2,1,4]$ , com  $k=7$ , temos  $C$  preenchido da seguinte maneira:

$C = [\text{true}, \text{true}, \text{false}, \text{true}, \text{false}, \text{false}, \text{true}]$

0,	1,	2,	3,	4,	5,	6
----	----	----	----	----	----	---

# Counting Sort

- **Exemplo:**

Agora, se criarmos um array B do tamanho de A e iterarmos sobre o array C preenchendo B com o valor do índice  $i+1$  em que  $C[i] == \text{true}$ , temos que B é a versão ordenada de A

# Counting Sort

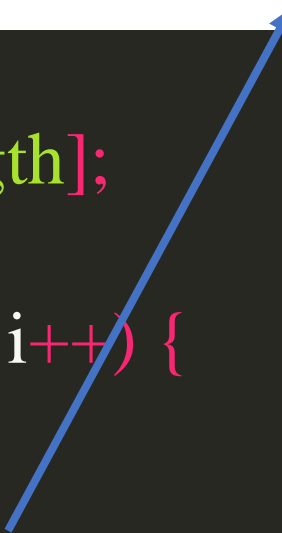
C = [true, true, false, true, false, false, true]

0, 1, 2, 3, 4, 5, 6

- Exemplo:

```
... int j = 0;
    int[] B = new int[A.length];

    for (int i = 0; i < C.length; i++) {
        if (C[i] == true) {
            B[j] = i + 1;
            j += 1;
        }
    } ...
```



# Counting Sort

- **Exemplo:**

Assim, para  $A=[7,2,1,4]$ , temos:

$C = [\text{true}, \text{true}, \text{false}, \text{true}, \text{false}, \text{false}, \text{true}]$

$B = [1, 2, 4, 7]$ , representando a sequência de valores de  $A$ , porém ordenada.

**Note** que **B** foi preenchido com os valores de  $i+1$  em que  $C[i] == \text{true}$ . Ou seja,  $B[0]=1$  , pois  $C[0] == \text{true}$ .  $B[1]=2$ , pois  $C[1] == \text{true}$ .  $B[2]=4$ , pois  $C[3] == \text{true}$ . Por fim,  $B[3]=7$  , pois  $C[6] == \text{true}$ .



# Counting Sort: E se houver repetição no array?

- O fato de não haver repetição nos permitiu criar um array C de booleanos e registrar a presença ou não de um elemento.
- Daí surge ordenação por contagem (Counting Sort). A ideia geral é registrar a frequência dos elementos ao invés da simples presença.
- Isso faz com que o array C passe a ser um array de inteiros, não de booleanos.

# Counting Sort: E se houver repetição no array?

- O algoritmo do Counting Sort é baseado na ideia que vimos, **mas possui algumas modificações substanciais para permitir elementos repetidos e para manter a estabilidade**

# Counting Sort: Como funciona?

1. Crie um **vetor auxiliar H** de tamanho igual à maior chave presente no vetor original A.

# Counting Sort: Como funciona?

1. Crie um vetor auxiliar H de tamanho igual à maior chave presente no vetor original A.
2. Conte o **número de ocorrências de cada elemento em A e armazene essa contagem no vetor auxiliar H**. Nesse vetor auxiliar, a primeira posição dá a frequência de elementos de valor 0 em A, a segunda posição dá a frequência de elementos de valor 1 em A e assim por diante.

# Counting Sort: Como funciona?

1. Crie um vetor auxiliar H de tamanho igual à maior chave presente no vetor original A.
2. Conte o número de ocorrências de cada elemento em A e armazene essa contagem no vetor auxiliar H. Nesse vetor auxiliar, a primeira posição dá a frequência de elementos de valor 0 em A, a segunda posição dá a frequência de elementos de valor 1 em A e assim por diante.
3. **Percorra o vetor auxiliar H para criar o histograma cumulativo de cada elemento de A.** Para tanto, **acumule o valor da frequência do elemento  $i + 1$  com a frequência do elemento  $i$ .**

# Counting Sort: Como funciona?

1. Crie um vetor auxiliar H de tamanho igual à maior chave presente no vetor original A.
2. Conte o número de ocorrências de cada elemento no vetor A e registre a frequência de cada elemento no vetor H.
3. Percorra o vetor auxiliar H e calcule a soma cumulativa de suas frequências. Para tanto, acumule o valor da frequência do elemento  $i$  com a frequência do elemento  $i + 1$ .

Calcular a soma cumulativa de C. Esse passo registra, para cada elemento  $x$  da entrada, o número de elementos menores ou iguais a  $x$ ;

# Counting Sort: Como funciona?

1. Crie um vetor auxiliar  $H$  de tamanho igual à maior chave presente no vetor original  $A$ .
2. Conte o número de ocorrências de cada elemento em  $A$  e armazene essa contagem no vetor auxiliar  $H$ . Nesse vetor auxiliar, a primeira posição dá a frequência de elementos de valor 0 em  $A$ , a segunda posição dá a frequência de elementos de valor 1 em  $A$  e assim por diante.
3. Percorra o vetor auxiliar  $H$  para criar o histograma cumulativo de cada elemento de  $A$ . Para tanto, acumule o valor da frequência do elemento  $i + 1$  com a frequência do elemento  $i$ .
4. **Use o vetor  $H$  para construir um vetor de saída ordenado.** Para tanto, se  $x$  é um elemento no vetor  $A$  que possui uma frequência cumulativa  $H(x)$ , então  $x$  deve ser colocado no vetor de saída entre a posição  $H(x)$  e  $H(x + 1)$ .

# Counting Sort

- Exemplo:

$A = [1, 9, 1, 3, 4, 7, 6, 7]$



# Counting Sort

- Exemplo:

$A = [1, 9, 1, 3, 4, 7, 6, 7]$

**Passo 1: Contagem de frequência.**

```
...  
int[] C = new int[k];  
// frequência  
for (int i = 0; i < A.length; i++) {  
    C[A[i] - 1] += 1;  
} ...
```

# Counting Sort

- Exemplo:

Para  $A=[1,9,1,3,4,7,6,7]$  e  $k=9$  , temos:

- $C = [2, 0, 1, 1, 0, 1, 2, 0, 1]$ , isto é, no array a ser ordenado há dois elementos de valor 1, nenhum elemento de valor 2, um elemento de valor 3 e assim por diante.

# Counting Sort

- Exemplo:

Para  $A=[1,9,1,3,4,7,6,7]$  e  $k=9$ , temos:

1, 2, 3, 4, 5, 6, 7, 8, 9

- $C = [2, 0, 1, 1, 0, 1, 2, 0, 1]$ , isto é, no array a ser ordenado há dois elementos de valor 1, nenhum elemento de valor 2, um elemento de valor 3 e assim por diante.

# Counting Sort

- Exemplo:

Para  $A=[1,9,1,3,4,7,6,7]$  e  $k=9$  , temos:

**Passo 2: Soma cumulativa em C.**

```
...  
    // cumulativa  
    for (int i = 1; i < C.length; i++) {  
        C[i] += C[i-1];  
    } ...
```

# Counting Sort

- Exemplo:

Para  $C=[2,0,1,1,0,1,2,0,1]$ , após a execução do cálculo da cumulativa, temos  $C=[2,2,3,4,4,5,7,7,8]$ , isto é, no array a ser ordenado há:

- 2 elementos menores ou igual a 1
- 2 elementos menores ou iguais a 2
- 3 elementos menores ou iguais a 3
- 4 elementos menores ou iguais a 4
- 4 elementos menores ou iguais a 5
- 5 elementos menores ou iguais a 6
- 7 elementos menores ou iguais a 7
- 7 elementos menores ou iguais a 8
- 8 elementos menores ou iguais a 9.

# Counting Sort

- Antes de avançar para o ultimo passo (passo 3), vamos rever os passos graficamente.

# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

Vetor a ser ordenado

	1	2	3	4	5	6	7	8	9	10
<i>A</i>	2	5	3	0	2	3	0	5	3	0

	1	2	3	4	5	6	7	8	9	10
<i>B</i>										

	0	1	2	3	4	5
<i>C</i>						

# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

	1	2	3	4	5	6	7	8	9	10
A	2	5	3	0	2	3	0	5	3	0
	1	2	3	4	5	6	7	8	9	10
B										
						</				

Saída: Vetor ordenado



# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

	1	2	3	4	5	6	7	8	9	10
<i>A</i>	2	5	3	0	2	3	0	5	3	0

	1	2	3	4	5	6	7	8	9	10
<i>B</i>										

	0	1	2	3	4	5
<i>C</i>						

Vetor temporário

# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

	1	2	3	4	5	6	7	8	9	10
<i>A</i>	2	5	3	0	2	3	0	5	3	0

	1	2	3	4	5	6	7	8
<i>B</i>								

	0	1	2	3	4	5
<i>C</i>	0	0	0	0	0	0

Inicialização do vetor  
temporário (registra as  
frequências)

# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

	1	2	3	4	5	6	7	8	9	10
$A$	2	5	3	0	2	3	0	5	3	0

	1	2	3	4	5	6	7	8
$B$								

	0	1	2	3	4	5
$C$	0	0	0	0	0	0

Tamanho  $k = 5$

# Counting Sort

Cada A

Tamanho  $k = 5$ ,  
Isso porque 5 é o maior  
elemento em A

	1	2	3	4	5	6	7	8	9	10
<i>A</i>	2	5	3	0	2	3	0	5	3	0

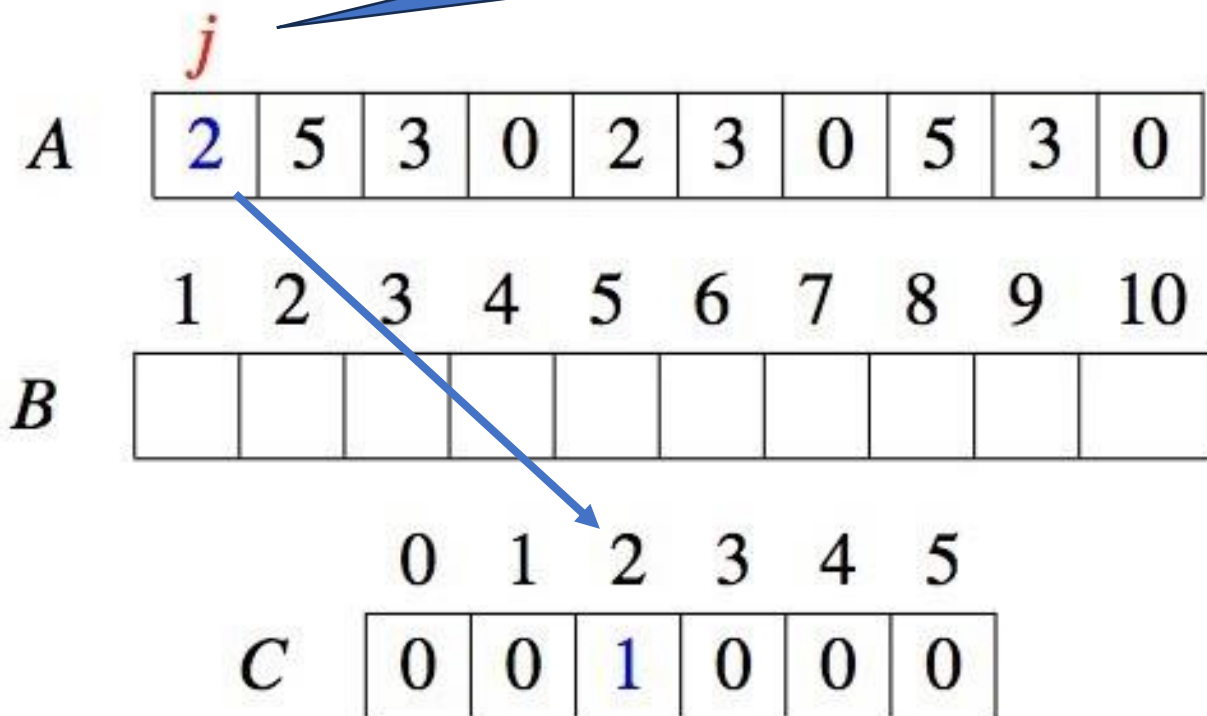
	1	2	3	4	5	6	7	8	9	10
<i>B</i>										

	0	1	2	3	4	5
<i>C</i>	0	0	0	0	0	0

# Counting Sort

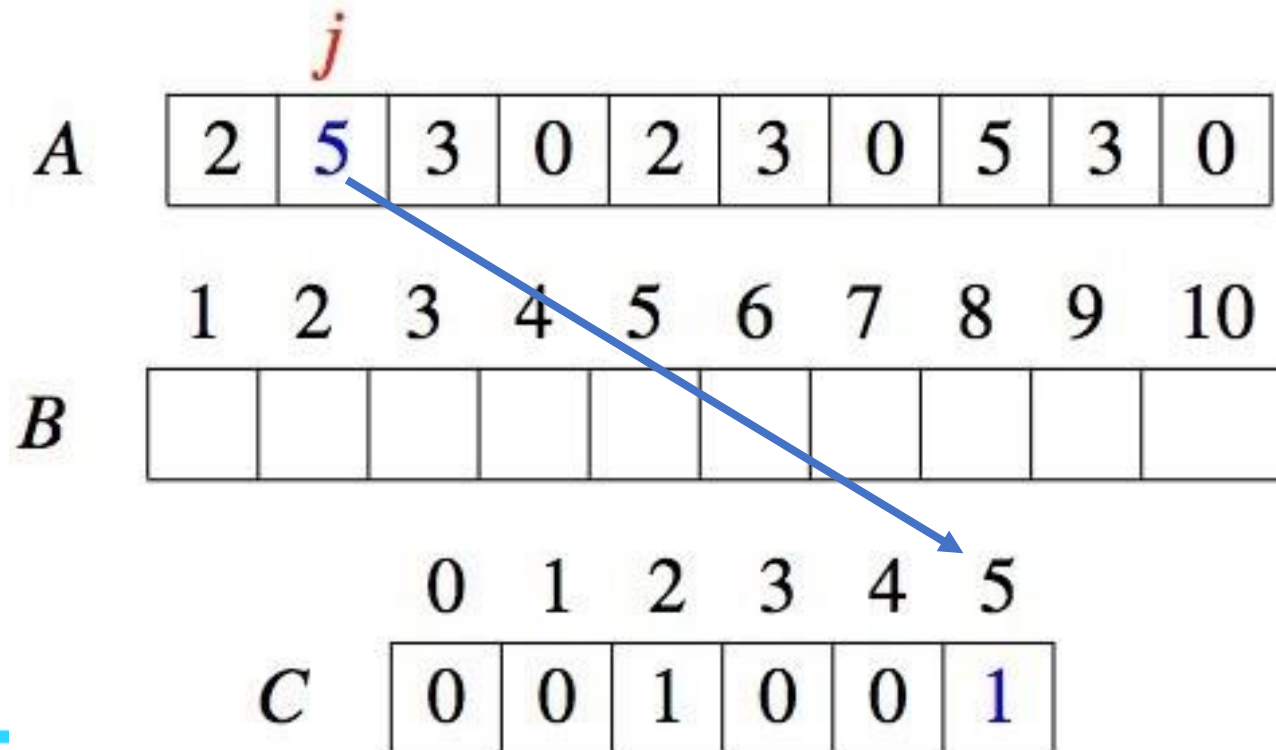
Cada  $A_j$

Inicia a varredura para registrar as frequências



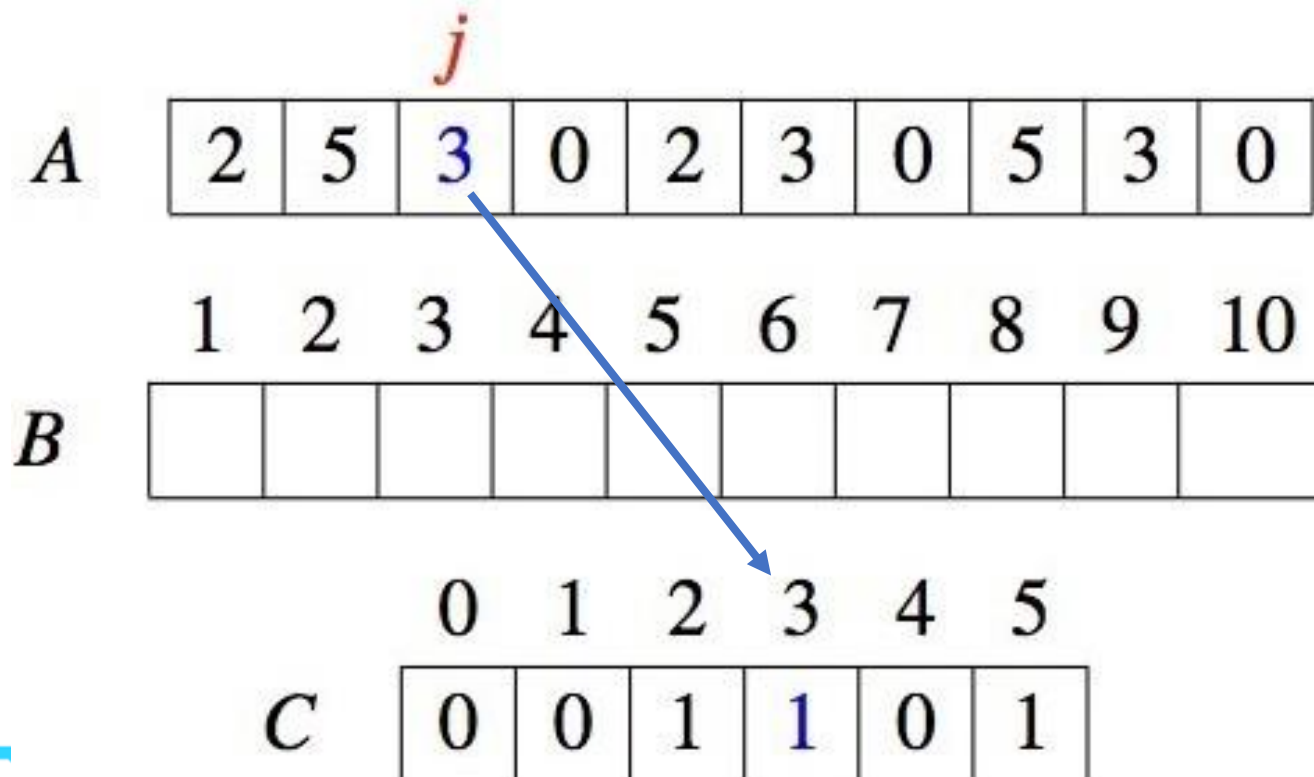
# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .



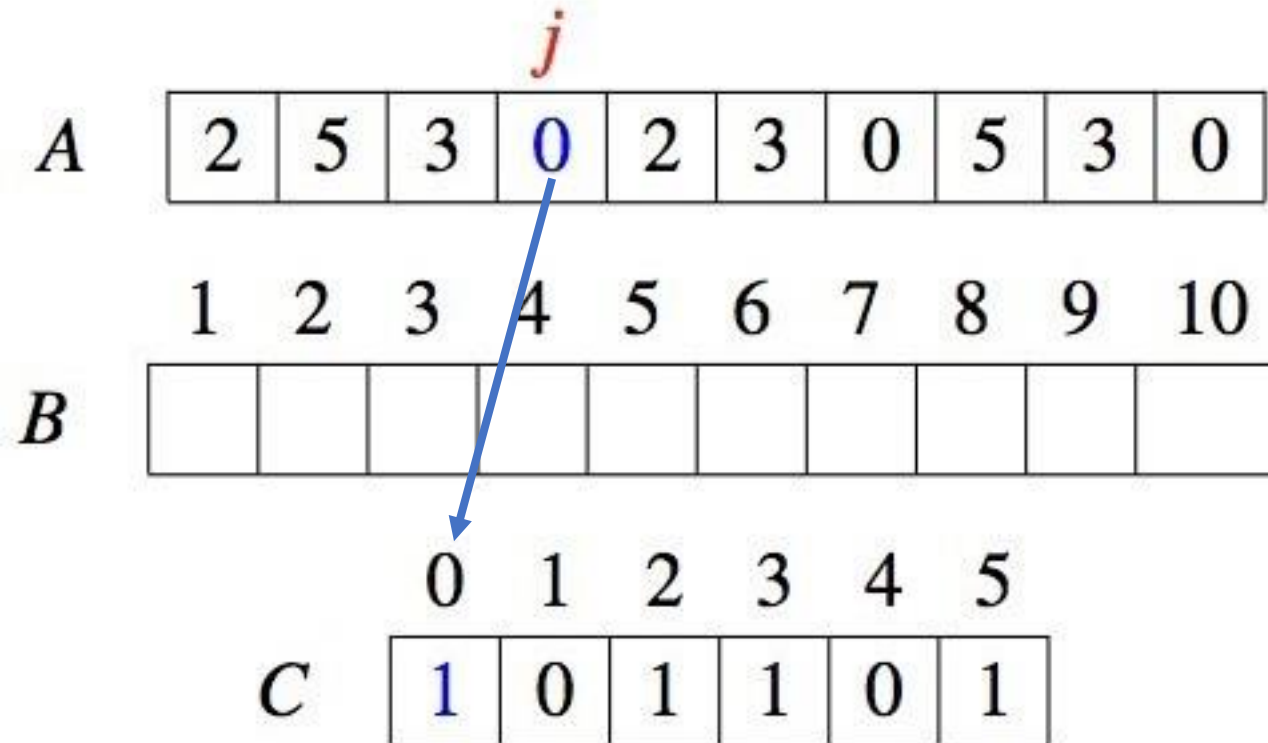
# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .



# Counting Sort

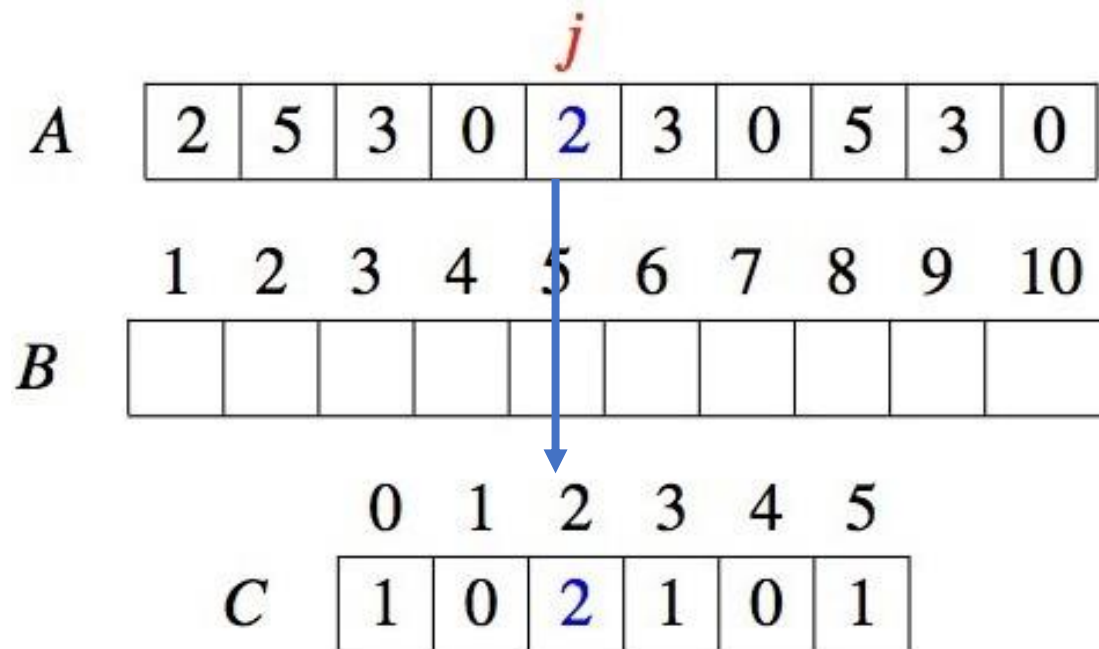
Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .





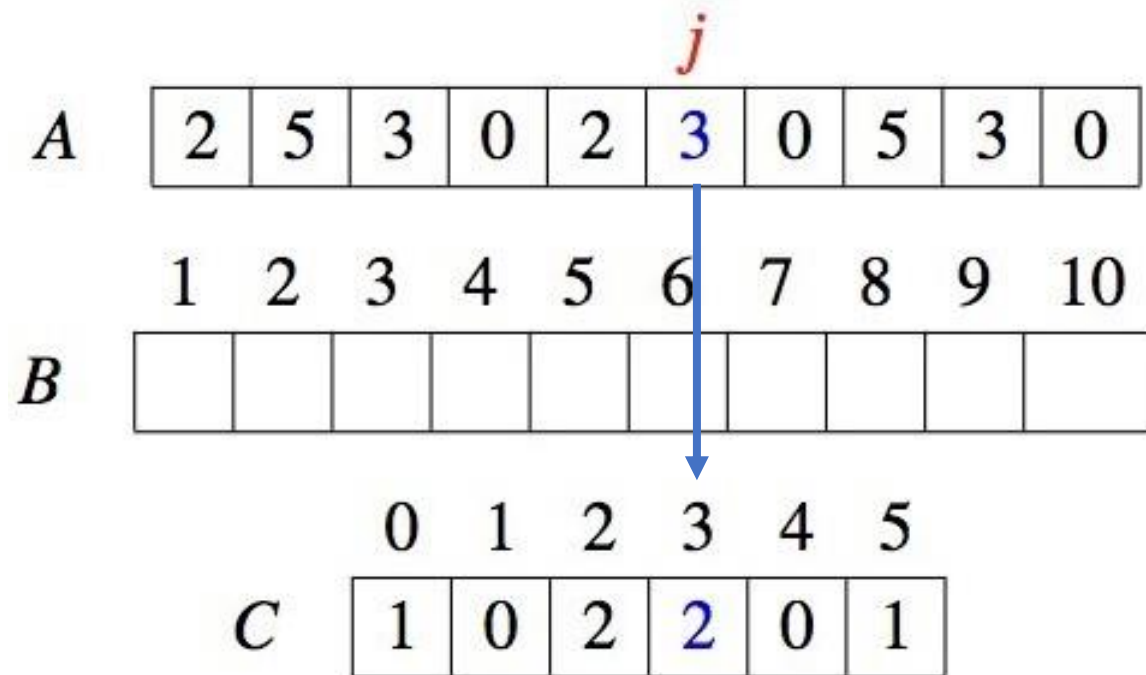
# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .



# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .



# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

$A$										
	2	5	3	0	2	3	$j$ 0	5	3	0
$B$	1	2	3	4	5	6	7	8	9	10
$C$	0	1	2	3	4	5				
	2	0	2	2	0	1				

# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

A										
	2	5	3	0	2	3	0	5	3	0
B	1	2	3	4	5	6	7	8	9	10

# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

$A$

2	5	3	0	2	3	0	5	3	0
---	---	---	---	---	---	---	---	---	---

$B$

--	--	--	--	--	--	--	--	--	--

$C$

2	0	2	3	0	2

# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

$A$	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div><math>j</math></div></div>									
	2	5	3	0	2	3	0	5	3	0
$B$	1 2 3 4 5 6 7 8 9 10									
$C$	0 1 2 3 4 5									
	3	0	2	3	0	2				

# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

	1								10	
$A$	2	5	3	0	2	3	0	5	3	0

	1	2	3	4	5	6	7	8	9	10
$B$										

	0	1	2	3	4	5
$C$	3	0	2	3	0	2

# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

	1								10	
A	2	5	3	0	2	3	0	5	3	0

	1	2	3	4	5	6
$B$						

Inicia o processo de  
somas (passo 3)

	0	1	2	3	4	5
$C$	3	0	2	3	0	2



# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

	1								10	
A	2	5	3	0	2	3	0	5	3	0

	1	2	3	4	5	6
$B$						

Inicia o processo de  
somas (passo 3)

	0	1	2	3	4	5
$C$	3	0	2	3	0	2



# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

	1								10	
$A$	2	5	3	0	2	3	0	5	3	0

	1	2	3	4	5	6	7	8	9	10
$B$										

	0	1	2	3	4	5
$C$	3	3	2	3	0	2



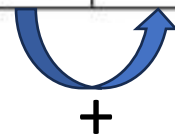
# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

	1								10	
A	2	5	3	0	2	3	0	5	3	0

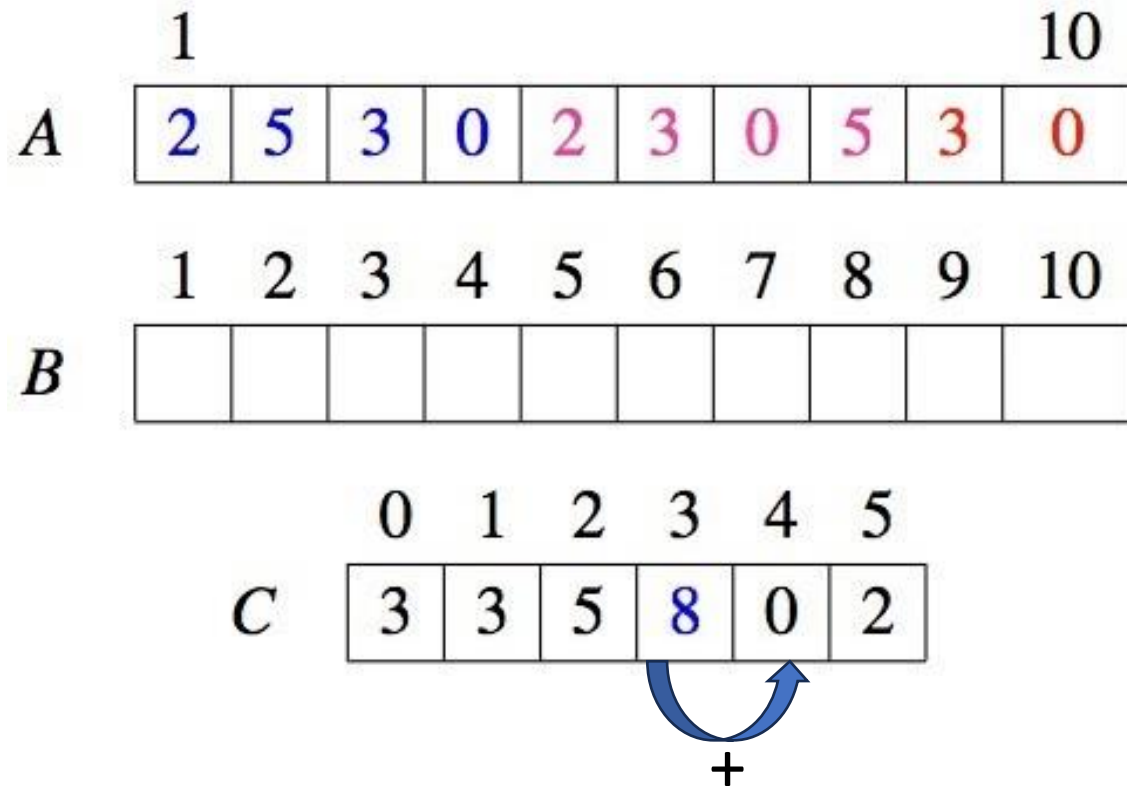
	1	2	3	4	5	6	7	8	9	10
$B$										

	0	1	2	3	4	5
$C$	3	3	5	3	0	2



# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .



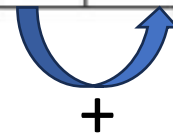
# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

	1								10	
A	2	5	3	0	2	3	0	5	3	0

	1	2	3	4	5	6	7	8	9	10
<i>B</i>										

	0	1	2	3	4	5
<i>C</i>	3	3	5	8	8	2



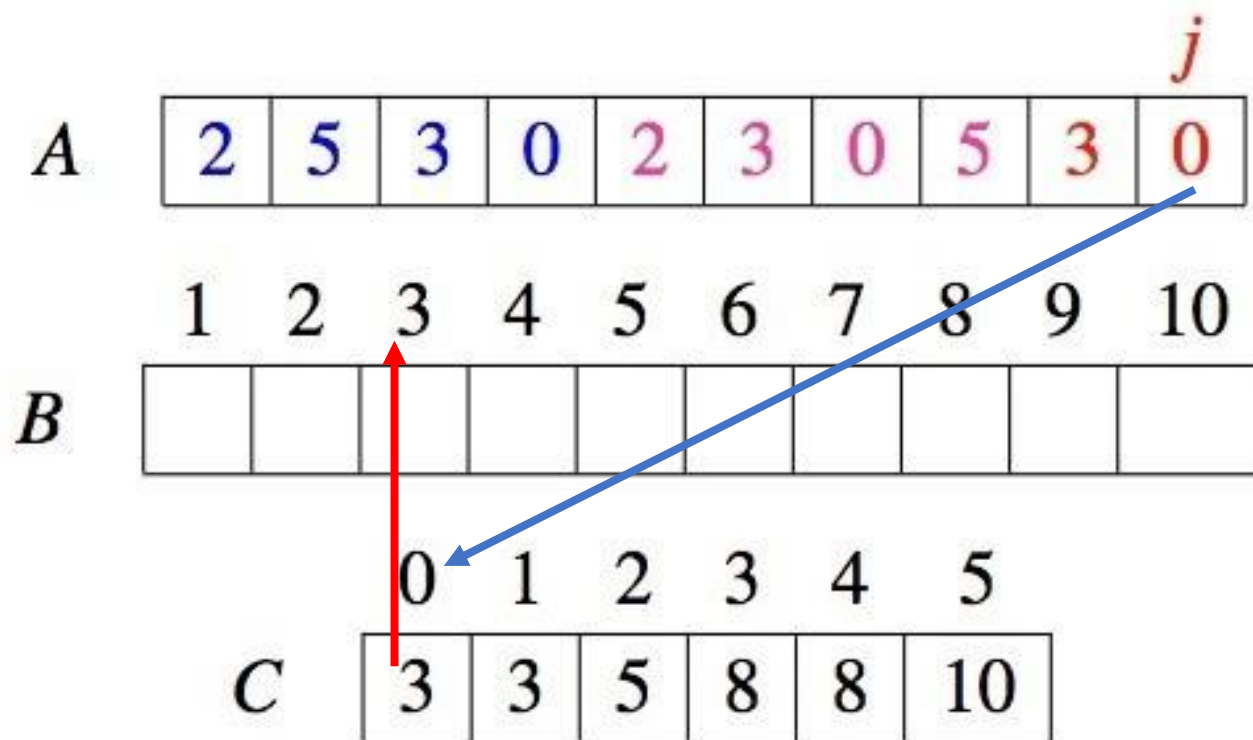
# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

	1									10
A	2	5	3	0	2	3	0	5	3	0
	1	2	3	4	5	6	7	8	9	10
B										
		0	1	2	3	4	5			
C		3	3	5	8	8	10			

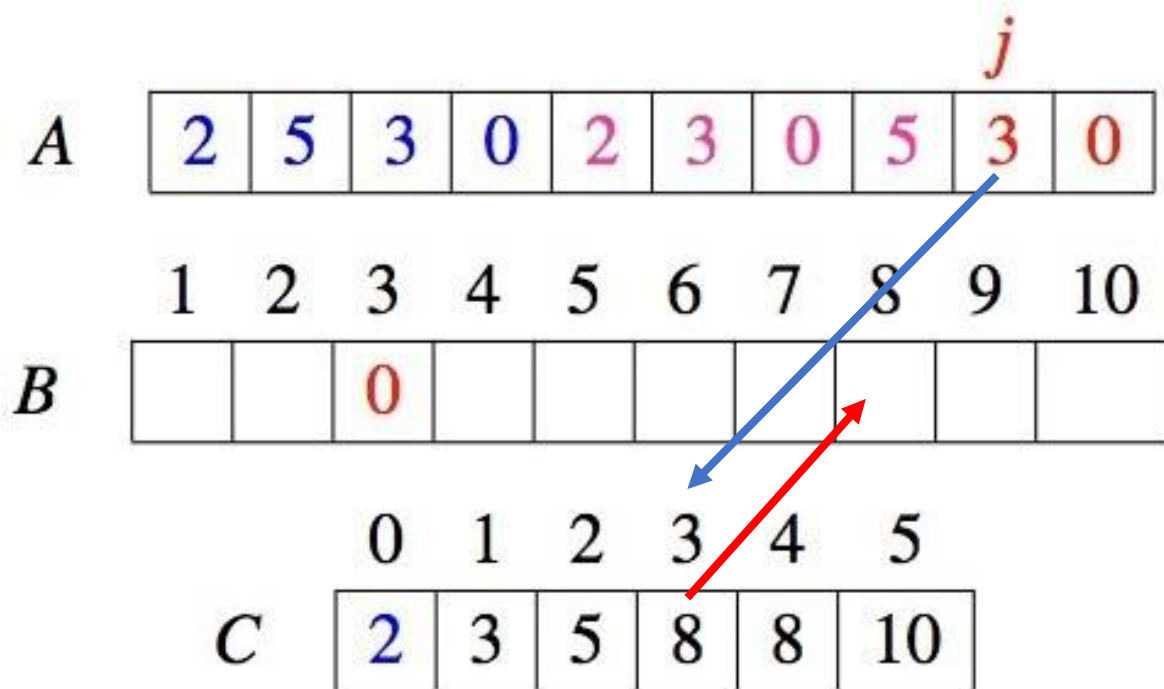
# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .



# Counting Sort

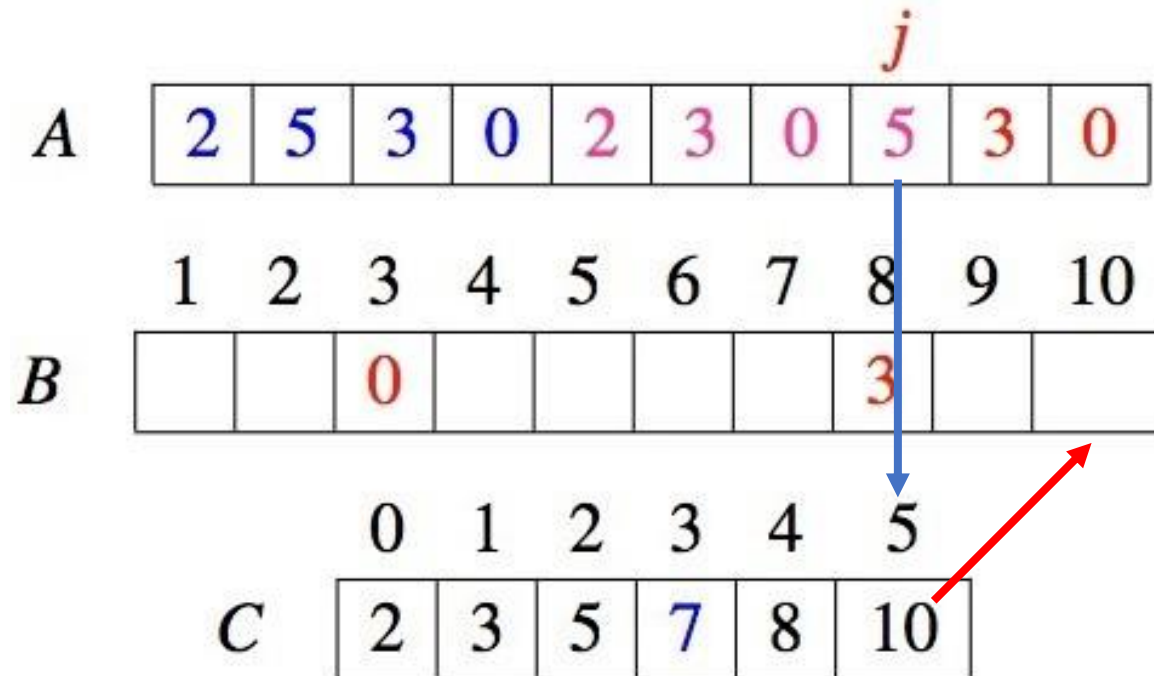
Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .





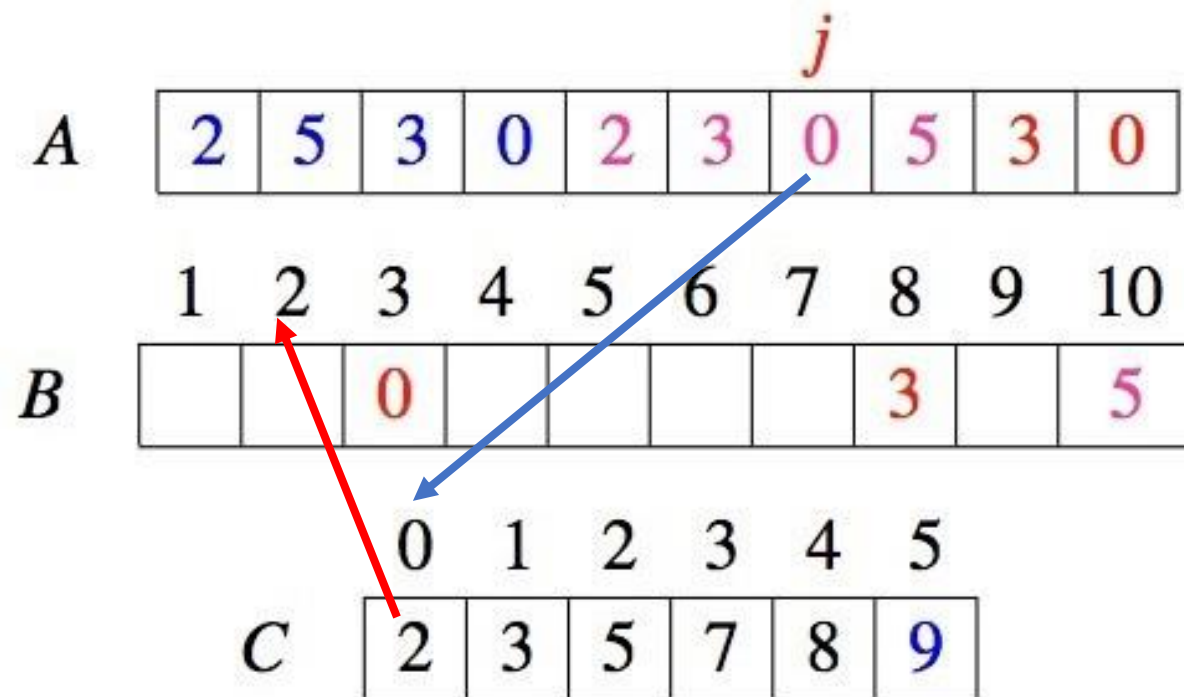
# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .



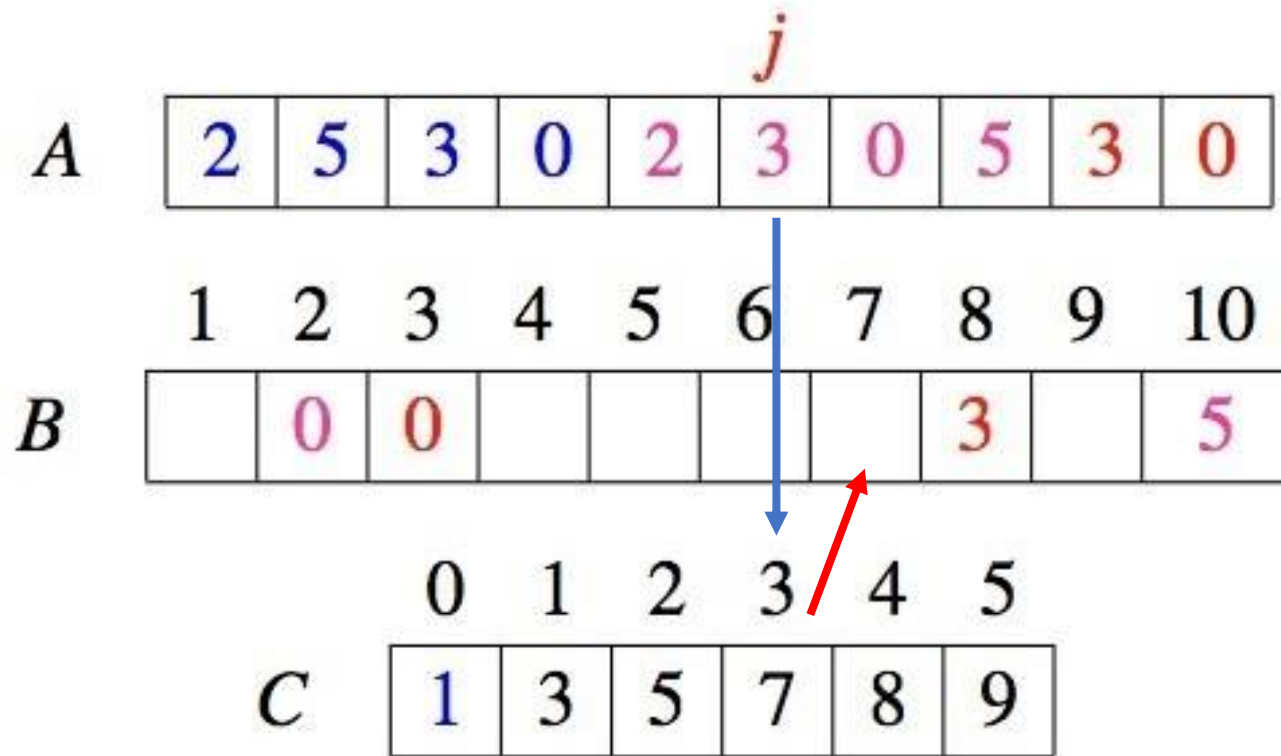
# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .



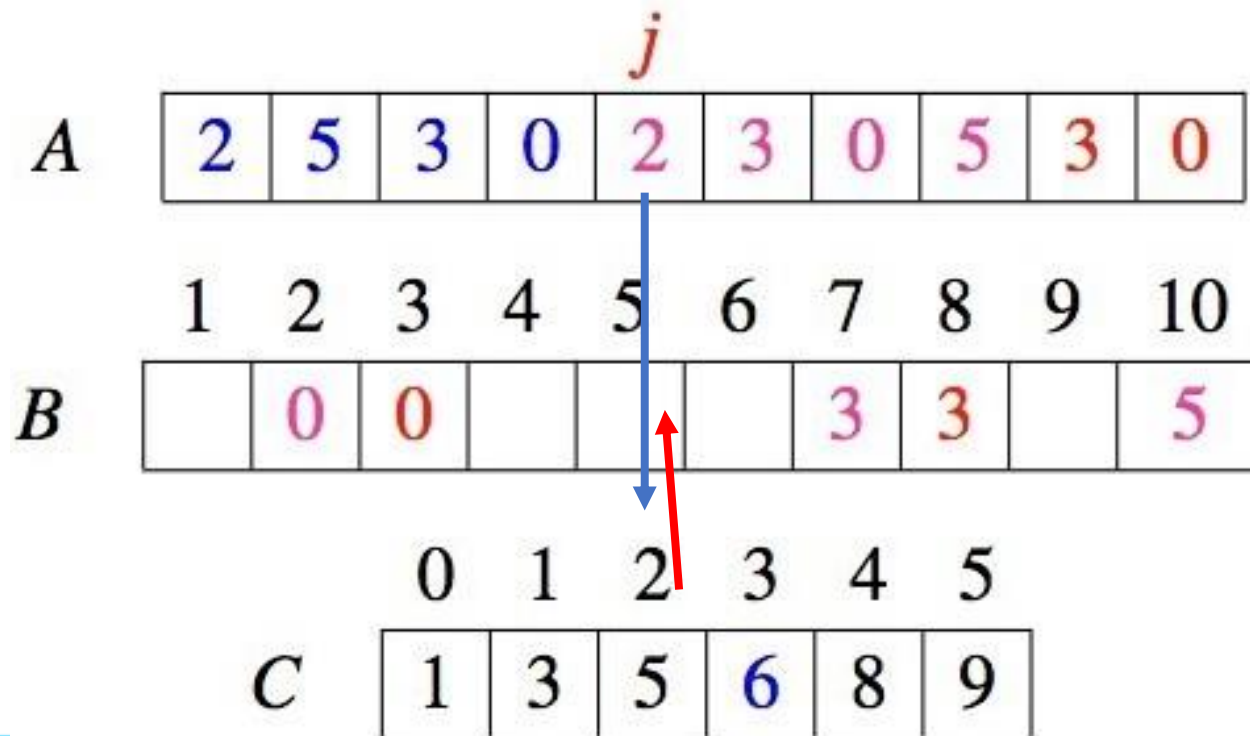
# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .



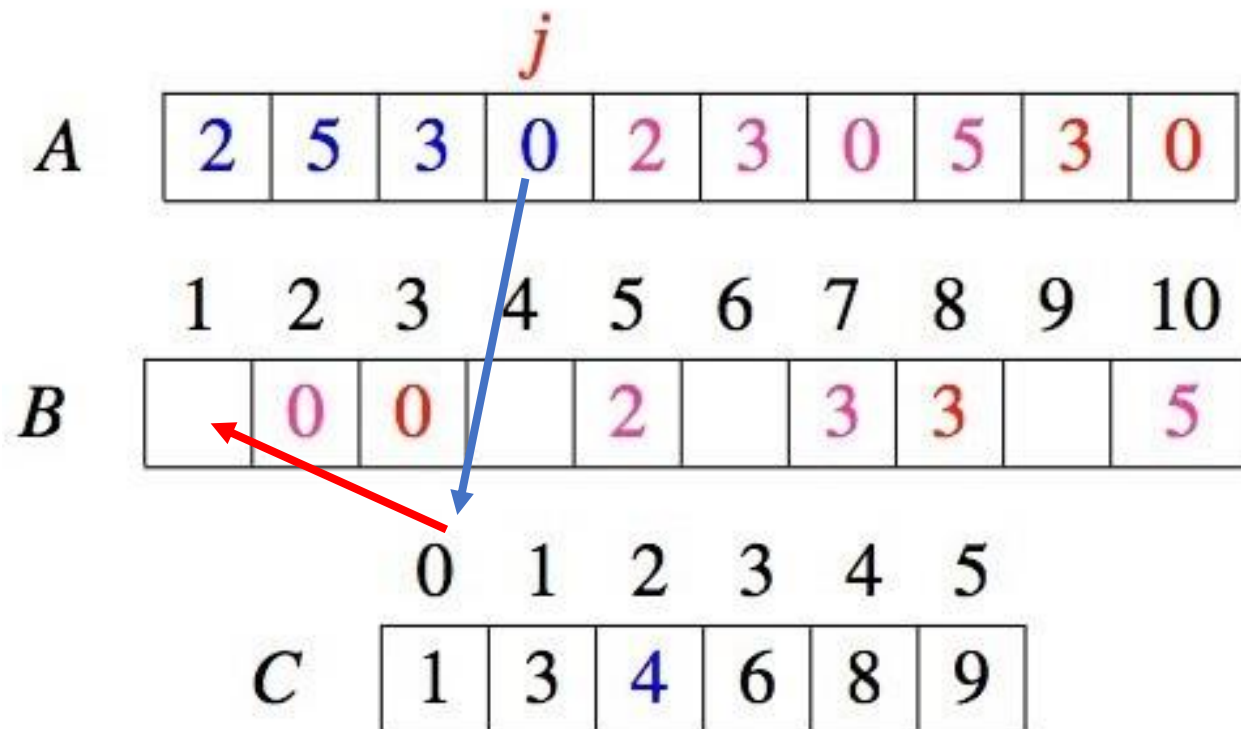
# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .



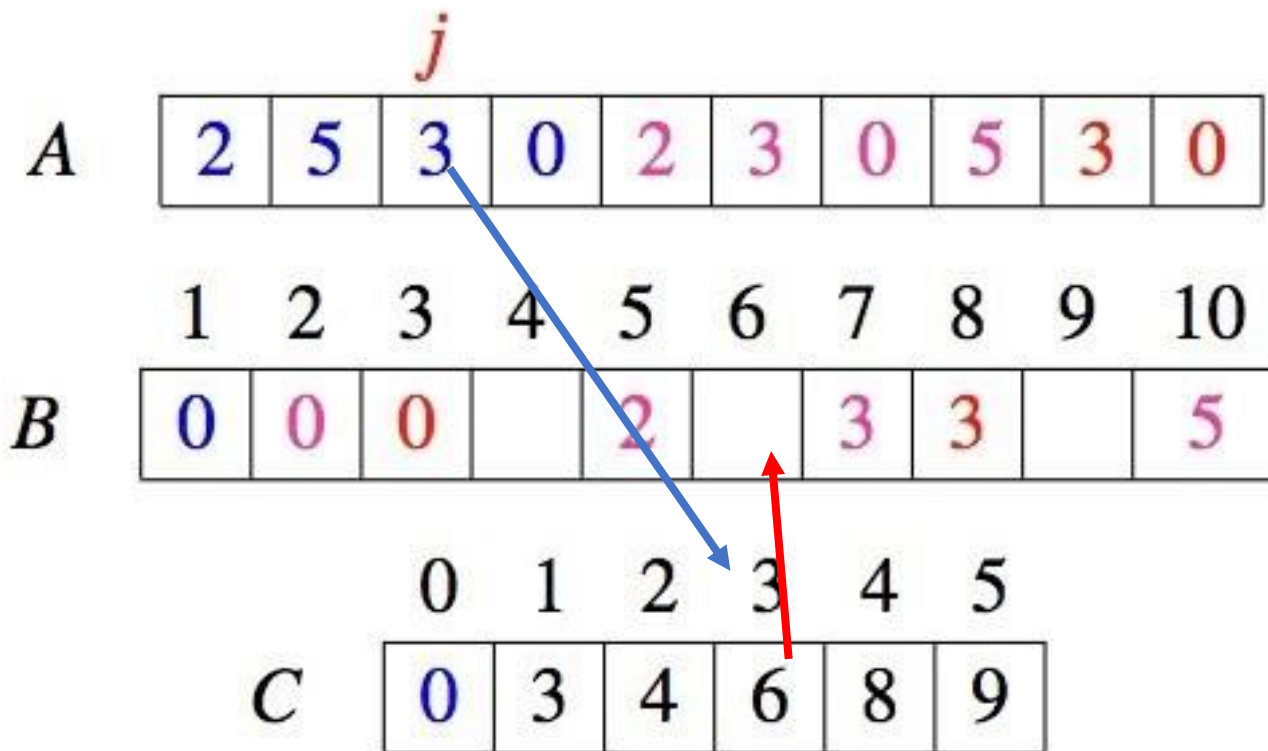
# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .



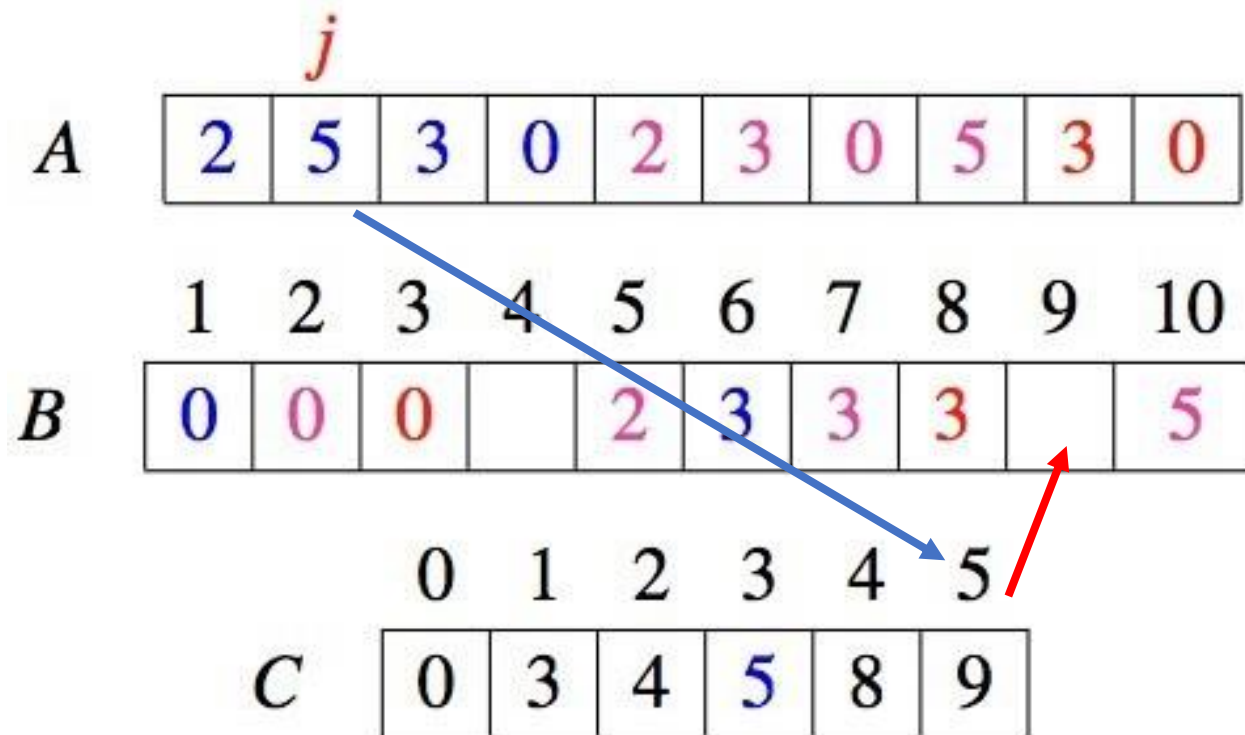
# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .



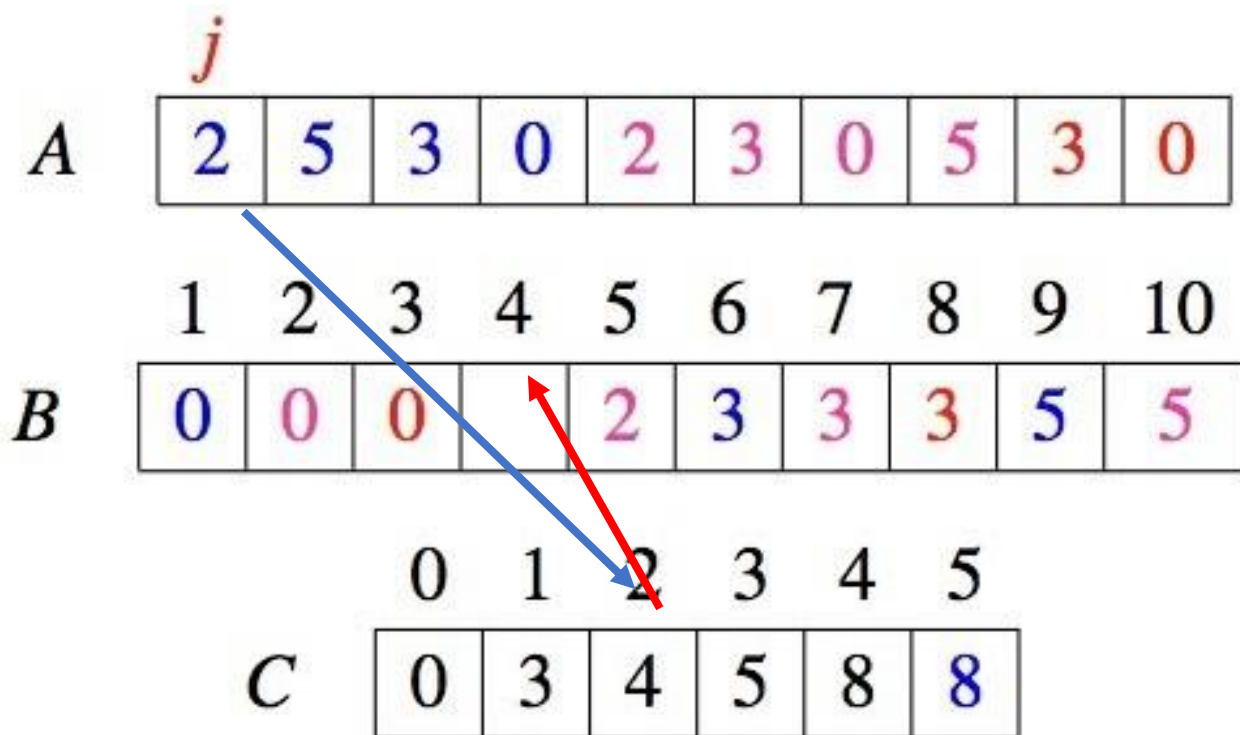
# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .



# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .





# Counting Sort

Cada  $A[i]$  está em  $\{0, \dots, 5\}$ .

$j$										
$A$	2	5	3	0	2	3	0	5	3	0
	1	2	3	4	5	6	7	8	9	10
$B$	0	0	0	2	2	3	3	3	5	5

# Counting Sort

- Exemplo:

Passo 3: Iterar do fim ao início de A registrando em B os elementos.

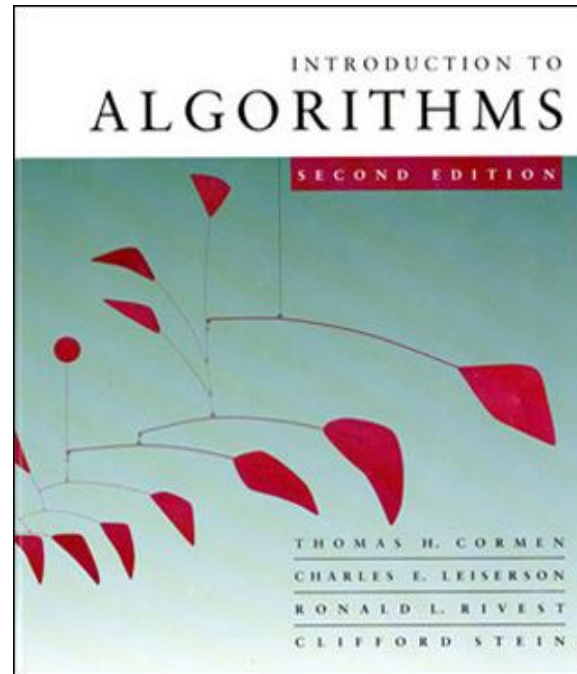
```
...  
int[] B = new int[A.length];  
for (int i = A.length - 1; i >= 0; i--) {  
    B[C[A[i] - 1] - 1] = A[i];  
    C[A[i] - 1] -= 1;  
}  
...
```

# Bibliografia

- [http://desenvolvendosoftware.com.br/algoritmos/ordenacao/counting-sort.html#:~:text=O%20Counting%20Sort%20\(Ordena%C3%A7%C3%A3o%20por,um%20arranjo%20de%20sa%C3%ADda%20ordenado.](http://desenvolvendosoftware.com.br/algoritmos/ordenacao/counting-sort.html#:~:text=O%20Counting%20Sort%20(Ordena%C3%A7%C3%A3o%20por,um%20arranjo%20de%20sa%C3%ADda%20ordenado.)
- <https://joaoarthurbm.github.io/eda/posts/ordenacao-linear/>

# Leitura importante

livro “Algorithms” de Cormen et al.



# Classificação e Pesquisa de Dados

Cristiano Santos

*cristiano.santos@amf.edu.br*