

Unidade V:


Ordenação Interna - Radix Sort



PUC Minas

Instituto de Ciências Exatas e Informática
Departamento de Ciência da Computação

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

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

Introdução

- 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

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

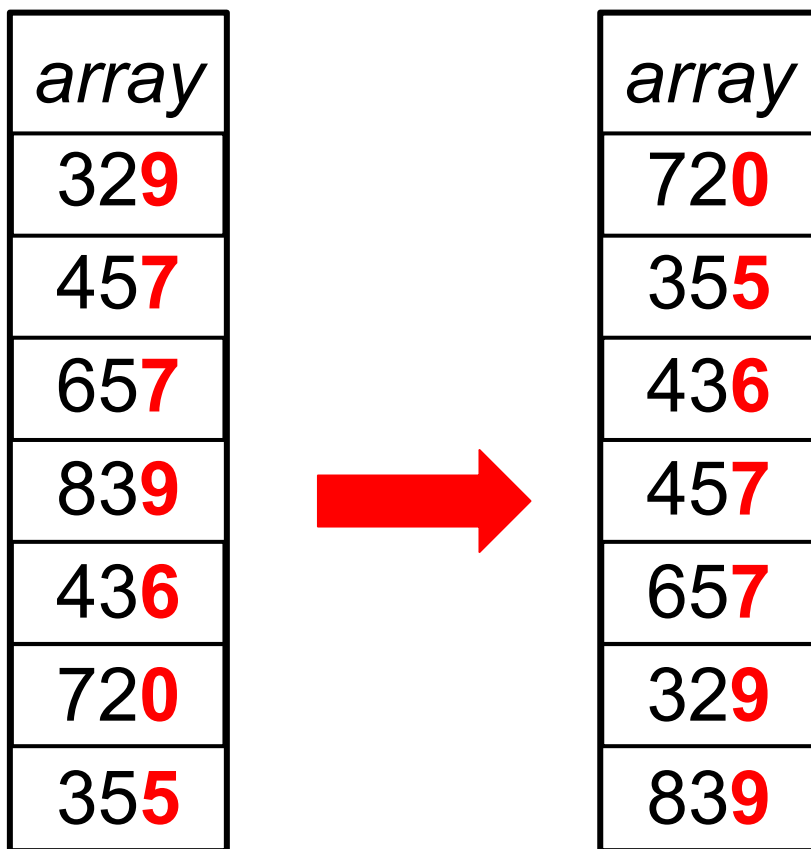
Ideia Básica

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

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

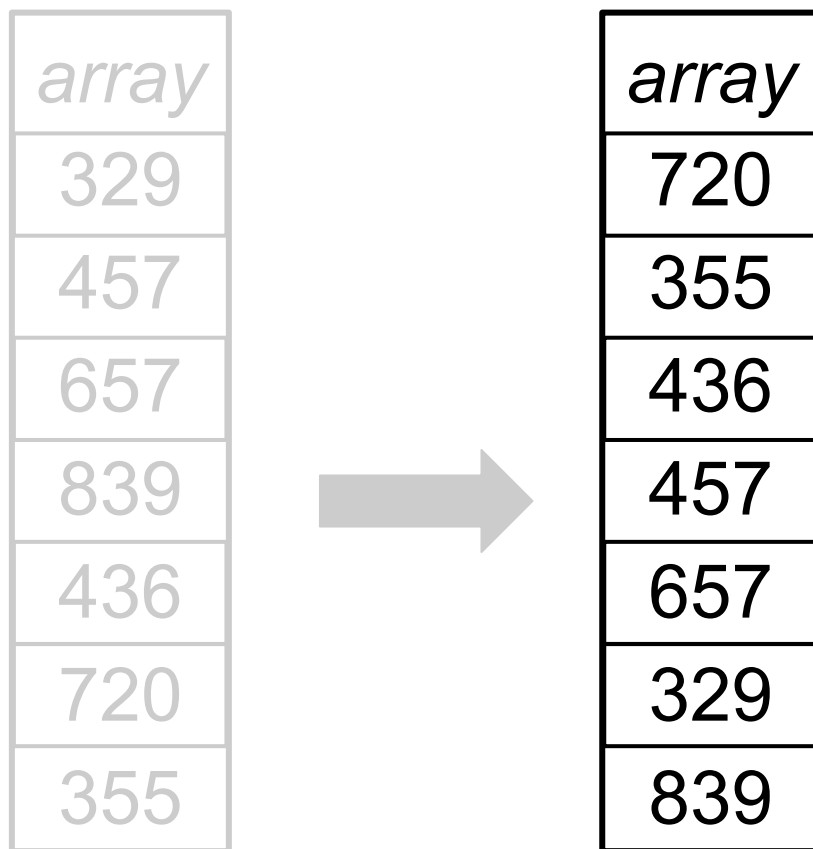
Ideia Básica

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



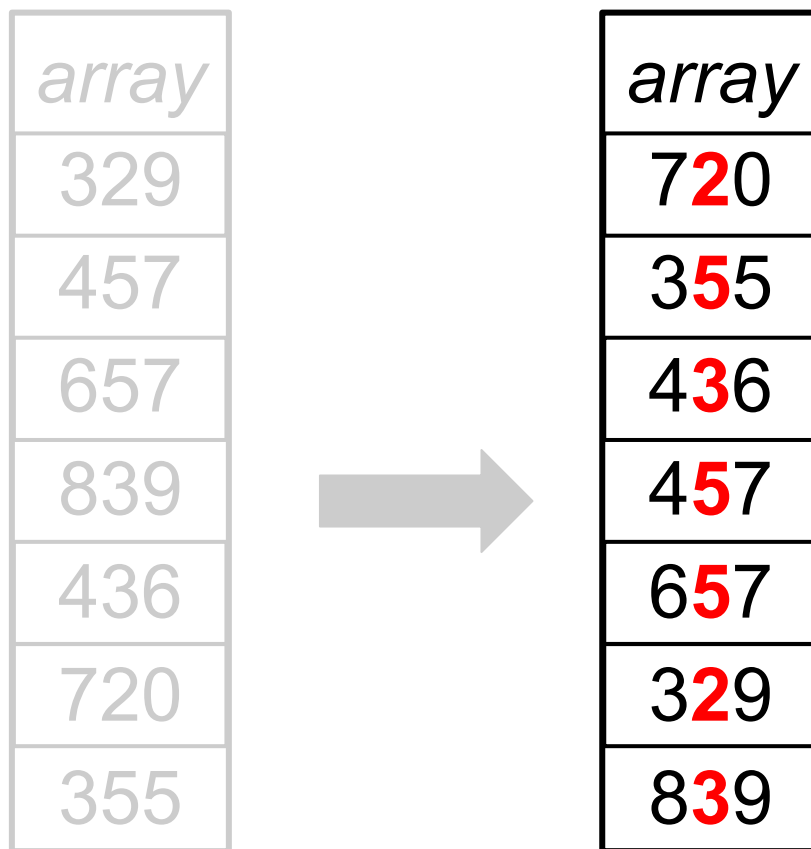
Ideia Básica

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



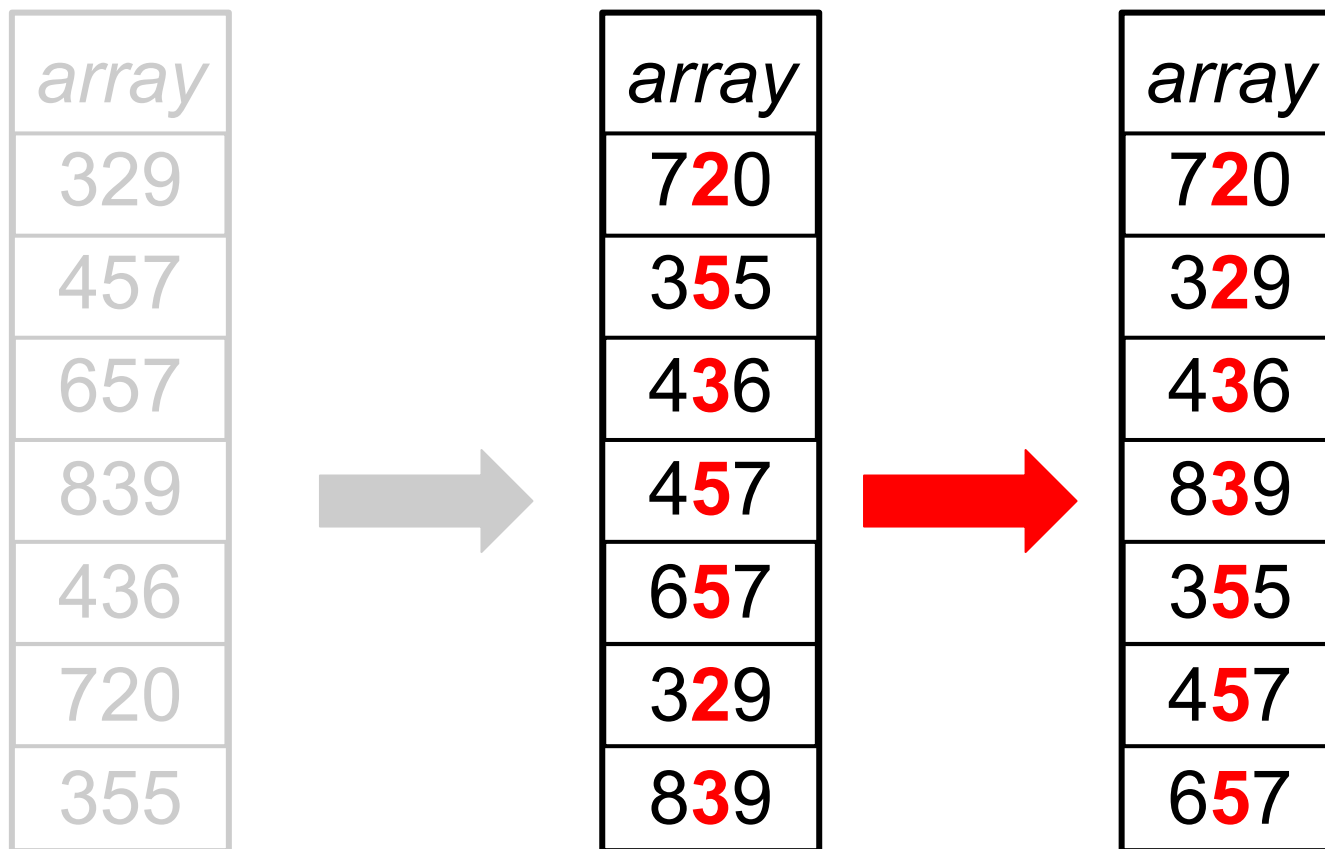
Ideia Básica

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



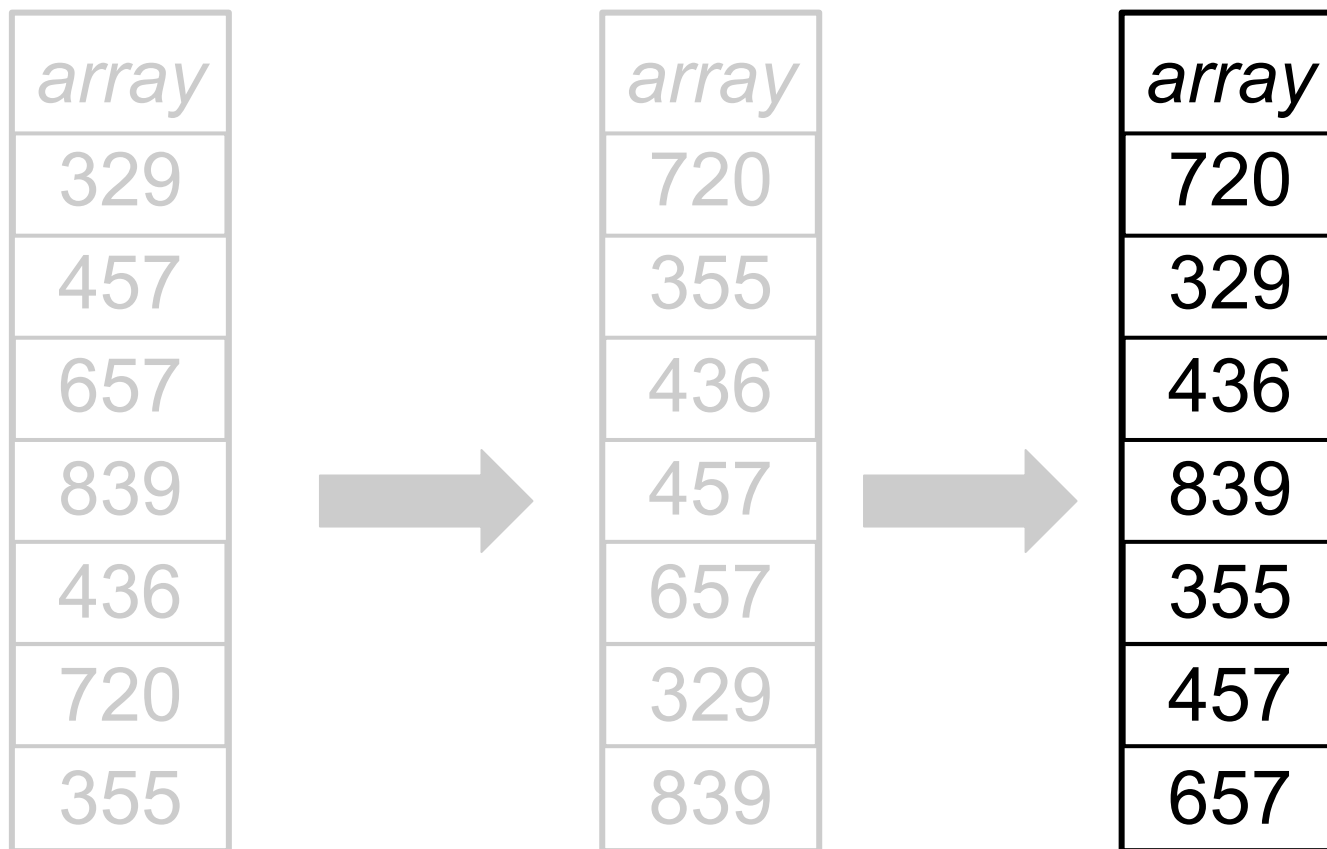
Ideia Básica

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



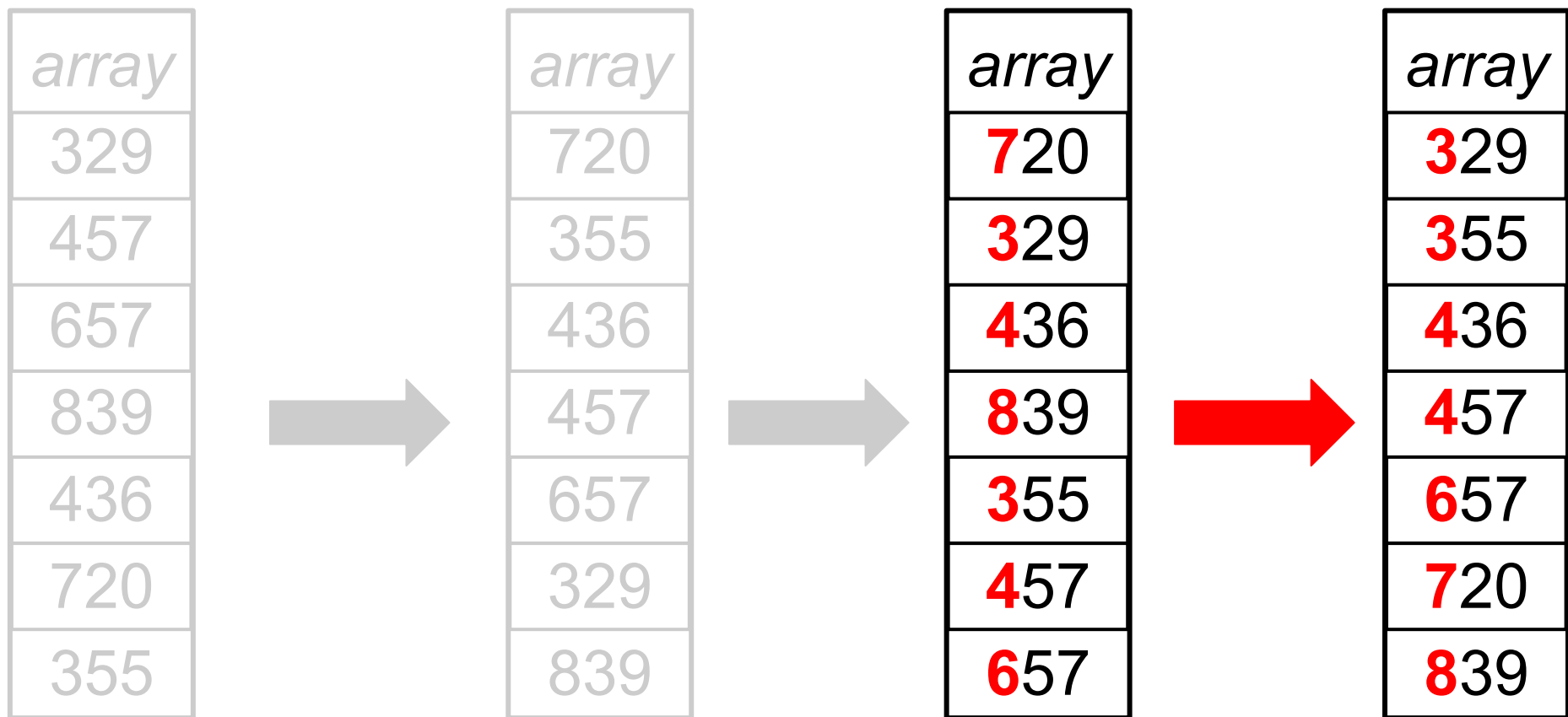
Ideia Básica


- 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



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

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

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

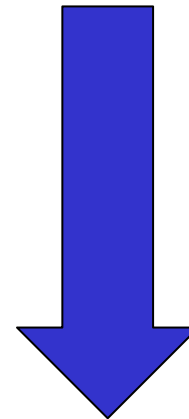
- Introdução
- Funcionamento básico
- Algoritmo em C *like*
- Análise dos número de movimentações e comparações
- **Conclusão** 

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 (2)

- Seja o *array* de entrada abaixo, quais serão os valores contidos no *array* de contagem antes e depois de copiarmos os elementos da entrada para a saída?

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

Exercício Resolvido (2)

- Seja o *array* de entrada abaixo, quais serão os valores contidos no *array* de contagem antes e depois de copiarmos os elementos da entrada para a saída?

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

Antes de copiarmos, supondo a posição zero, teremos:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18

Depois, teremos:

0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17

Exercício Resolvido (2)

- Seja o *array* de entrada abaixo, quais serão os valores contidos no *array* de contagem antes e depois de copiarmos os elementos da entrada para a saída?

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

Antes de copiarmos, supondo a posição zero, teremos:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18

Depois, teremos:

0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17

Exercício Resolvido (3)

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

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).

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

Algoritmo em C *like*

```
void countingsort() {  
    //Array para contar o numero de ocorrencias de cada elemento  
    int[] count = new int[getMaior() + 1];  
    int[] ordenado = new int[n];  
  
    //Inicializar cada posicao do array de contagem  
    for (int i = 0; i < count.length; count[i] = 0, i++);  
  
    //Agora, o count[i] contem o numero de elemento iguais a i  
    for (int i = 0; i < n; count[array[i]]++, i++);  
  
    //Agora, o count[i] contem o numero de elemento menores ou iguais a i  
    for (int i = 1; i < count.length; count[i] += count[i-1], i++);  
  
    //Ordenando  
    for (int i = n-1; i >= 0; ordenado[count[array[i]]-1] = array[i], count[array[i]]--, i--);  
}
```


Algoritmo em C *like*

```
void countingsort() {  
    //Array para contar o numero de ocorrencias de cada elemento  
    int[] count = new int[getMaior() + 1];  
    int[] ordenado = new int[n];  
}
```

Algoritmo em C *like*

```
void countingsort() {  
    //Array para contar o numero de ocorrencias de cada elemento  
    int[] count = new int[getMaior() + 1];  
    int[] ordenado = new int[n];  
  
    //Inicializar cada posicao do array de contagem  
    for (int i = 0; i < count.length; count[i] = 0, i++);  
  
    //Agora, o count[i] contem o numero de elemento iguais a i  
    for (int i = 0; i < n; count[array[i]]++, i++);  
  
    //Agora, o count[i] contem o numero de elemento menores ou iguais a i  
    for (int i = 1; i < count.length; count[i] += count[i-1], i++);  
  
    //Ordenando  
    for (int i = n-1; i >= 0; ordenado[count[array[i]]-1] = array[i], count[array[i]]--, i--);  
}
```

Algoritmo em C *like*

```
void countingsort() {
```

```
//Inicializar cada posicao do array de contagem
```

```
for (int i = 0; i < count.length; count[i] = 0, i++);
```

```
//Agora, o count[i] contem o numero de elemento iguais a i
```

```
for (int i = 0; i < n; count[array[i]]++, i++);
```

```
//Agora, o count[i] contem o numero de elemento menores ou iguais a i
```

```
for (int i = 1; i < count.length; count[i] += count[i-1], 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

Algoritmo em C *like*

```
void countingsort() {  
    //Array para contar o numero de ocorrencias de cada elemento  
    int[] count = new int[getMaior() + 1];  
    int[] ordenado = new int[n];  
  
    //Inicializar cada posicao do array de contagem  
    for (int i = 0; i < count.length; count[i] = 0, i++);  
  
    //Agora, o count[i] contem o numero de elemento iguais a i  
    for (int i = 0; i < n; count[array[i]]++, i++);  
  
    //Agora, o count[i] contem o numero de elemento menores ou iguais a i  
    for (int i = 1; i < count.length; count[i] += count[i- 1], i++);  
  
    //Ordenando  
    for (int i = n-1; i >= 0; ordenado[count[array[i]]-1] = array[i], count[array[i]]--, i--);  
}
```

Algoritmo em C *like*

```
void countingsort() {
```

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

```
//Ordenando
```

```
for (int i = n-1; i >= 0; ordenado[count[array[i]]-1] = array[i], count[array[i]]--, i--);
```

```
}
```

Algoritmo em C *like*

```
void countingsort() {
```

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

//Ordenando

```
for (int i = n-1; i >= 0; ordenado[count[array[i]]-1] = array[i], count[array[i]]--, i--);
```

```
}
```

7

Algoritmo em C *like*

```
void countingsort() {
```

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

//Ordenando

```
for (int i = n-1; i >= 0; ordenado[count[array[7]]-1] = array[i], count[array[i]]--, i--);
```

}

3

Algoritmo em C *like*

```
void countingsort() {
```

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

//Ordenando

```
for (int i = n-1; i >= 0; ordenado[count[3]-1] = array[i], count[array[i]]--, i--);
```

}

7

Algoritmo em C *like*

```
void countingsort() {
```

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

//Ordenando

```
for (int i = n-1; i >= 0; ordenado[7 - 1] = array[7], count[array[i]--, i--);
```

}

6

3

Algoritmo em C *like*

```
void countingsort() {
```

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	

//Ordenando

```
for (int i = n-1; i >= 0; ordenado[7 - 1] = array[7], count[array[i]--];
```

```
}
```

6

3

Algoritmo em C *like*

```
void countingsort() {
```

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	

```
//Ordenando
```

```
for (int i = n-1; i >= 0; ordenado[7 - 1] = array[7], count[array[i]--], i--);
```

```
}
```

Algoritmo em C *like*

```
void countingsort() {  
    //Array para contar o numero de ocorrencias de cada elemento  
    int[] count = new int[getMaior() + 1];  
    int[] ordenado = new int[n];  
  
    //Inicializar cada posicao do array de contagem  
    for (int i = 0; i < count.length; count[i] = 0, i++);  
  
    //Agora, o count[i] contem o numero de elemento iguais a i  
    for (int i = 0; i < n; count[array[i]]++, i++);  
  
    //Agora, o count[i] contem o numero de elemento menores ou iguais a i  
    for (int i = 1; i < count.length; count[i] += count[i-1], i++);  
  
    //Ordenando  
    for (int i = n-1; i >= 0; ordenado[count[array[i]]-1] = array[i], count[array[i]]--, i--);  
}
```

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

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:

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

Exercício

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