

Unidade IV:

Ordenação Interna - Counting Sort

Prof. Max do Val Machado



PUC Minas

Instituto de Ciências Exatas e Informática
Curso de Ciência da Computação

Ideia Básica

- Considera-se três *arrays*: entrada, contagem e saída
- Para cada elemento do *array* de entrada, determina-se no *array* de contagem a quantidade de elementos menores ou igual a ele
- Usa-se essa informação para inserir cada elemento em sua posição no *array* de saída
- Por exemplo, se um elemento x for maior que outros 17 elementos, x será inserido na 18ª posição

Ideia Básica

- Suponha que o *array* de entrada tenha n elementos cujos valores estão entre 0 e k
- Logo, o *array* de saída terá n elementos e o de contagem terá k elementos onde o valor inicial de cada posição será zero

Funcionamento Básico

- 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

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

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	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

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	2	4	7	7

Array de saída

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

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 $O(n)$
- 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 $O(n)$
- Para cada elemento do *array* de entrada, incrementá-lo no de contagem $O(n)$
- 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 $O(n)$
- Para cada elemento do *array* de entrada, incrementá-lo no de contagem $O(n)$
- 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 $O(n)$
- 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 $O(n)$
- Para cada elemento do *array* de entrada, incrementá-lo no de contagem $O(n)$
- 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 $O(n)$
- Sabendo o número de elementos menores ou iguais a i , preencher o *array* de saída $O(n)$

Análise das Operações com Elementos do *Array*

- Análise do complexidade para operações com elementos do array:

$$O(n) + O(n) + O(n) + O(n) = O(n)$$

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

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

```
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]--], 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--);  
}
```

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