Ordenação em tempo linear

CLRS 8.2–8.3

Ordenação por contagem

Recebe vetores $A[1\,..\,n]$ e $B[1\,..\,n]$ e devolve no vetor $B[1\,..\,n]$ os elementos de $A[1\,..\,n]$ em ordem crescente.

Cada $A[i]$ está em $\{0, \ldots, k\}$.

Entra:

	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										

Ordenação por contagem

Recebe vetores $A[1\,..\,n]$ e $B[1\,..\,n]$ e devolve no vetor $B[1\,..\,n]$ os elementos de $A[1\,..\,n]$ em ordem crescente.

Cada $A[i]$ está em $\{0, \ldots, k\}$.

Entra:

	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										

Sai:

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

Ordenação por contagem

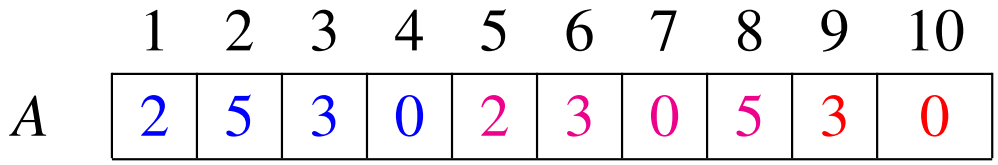
Cada $A[i]$ está em $\{0, \ldots, 5\}$.

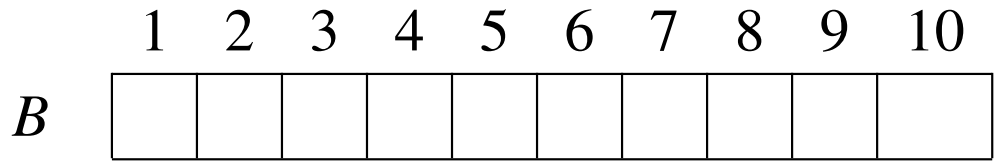
	1	2	3	4	5	6	7	8	9	10
A	2	5	3	0	2	3	0	5	3	0

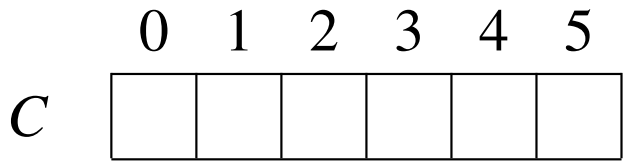
	1	2	3	4	5	6	7	8	9	10
B										

Ordenação por contagem

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

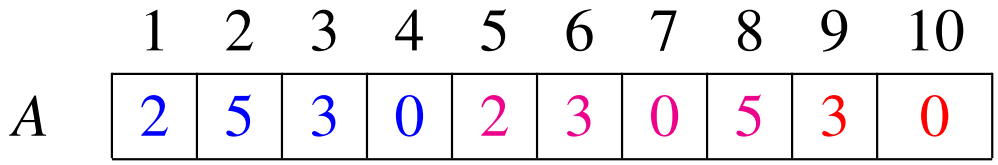


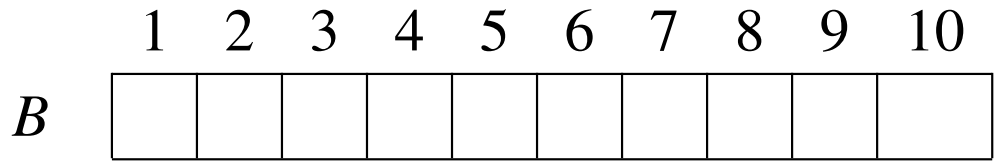


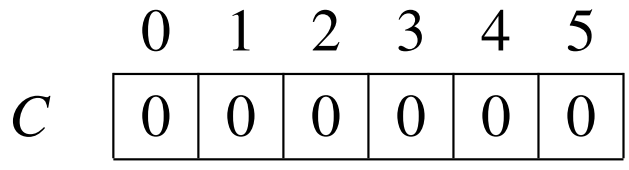


Ordenação por contagem

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

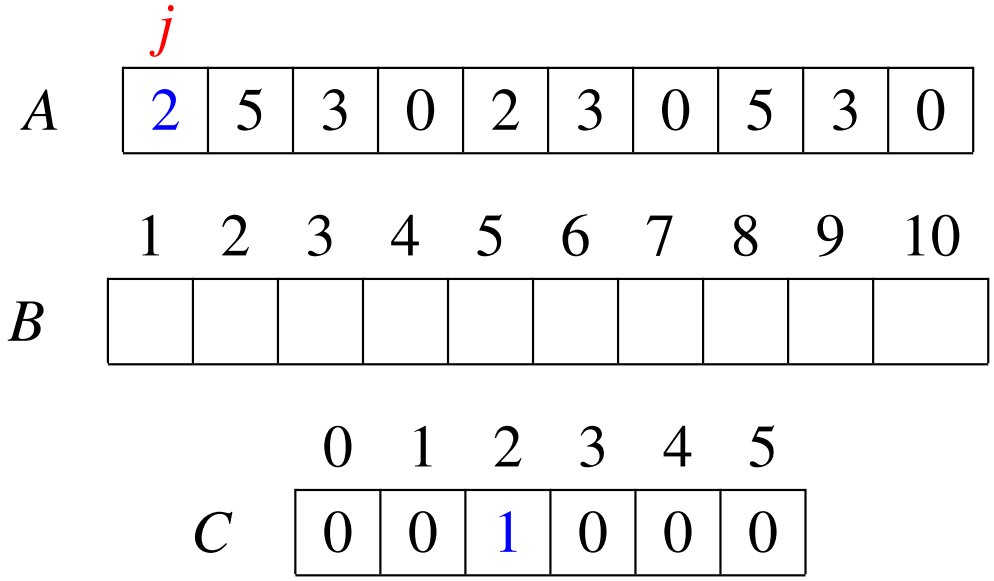






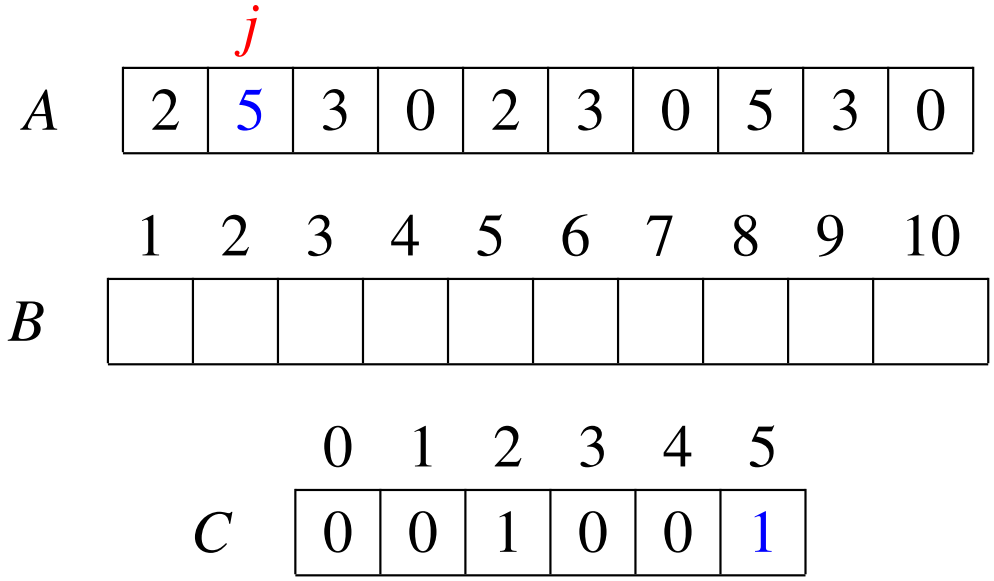
Ordenação por contagem

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



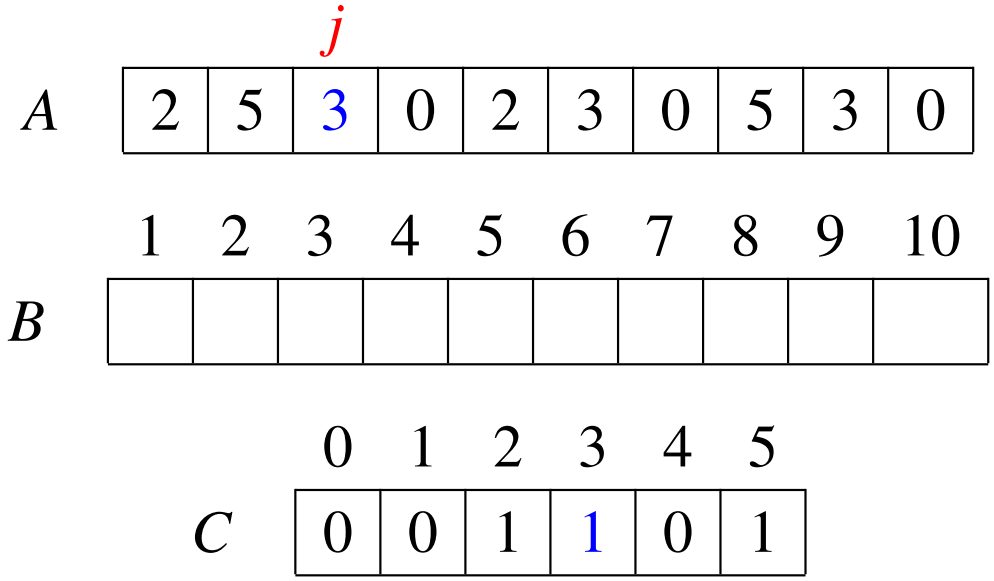
Ordenação por contagem

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



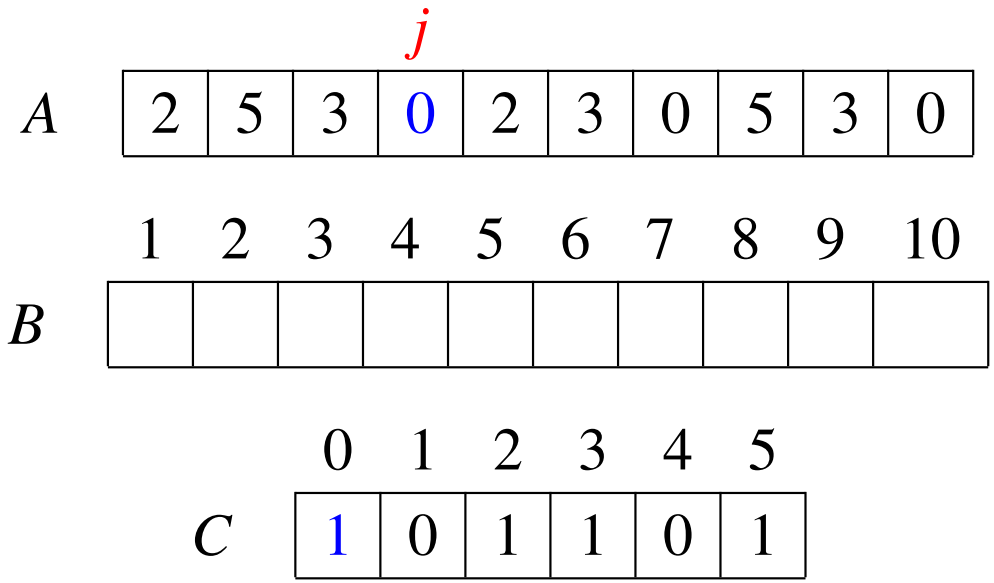
Ordenação por contagem

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



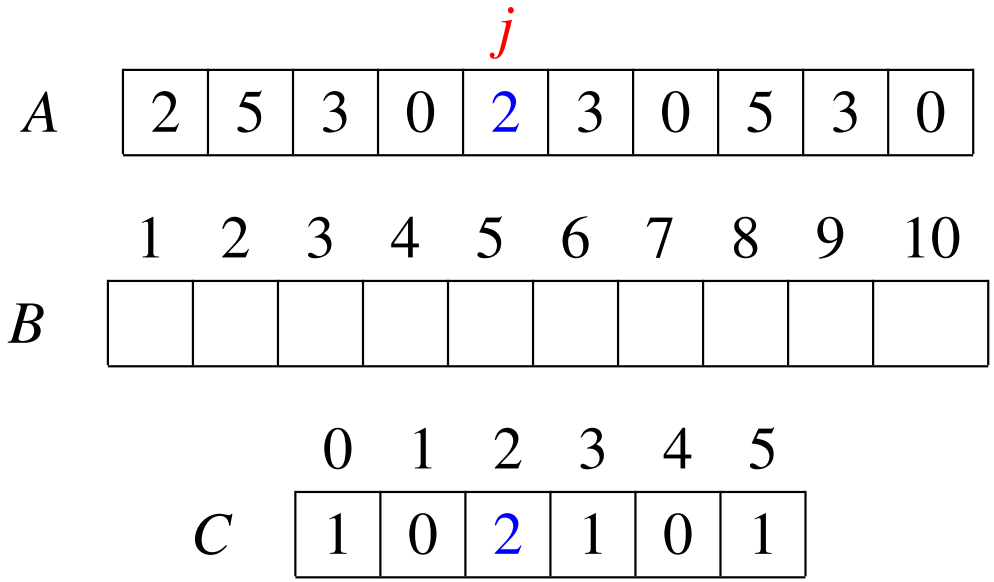
Ordenação por contagem

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



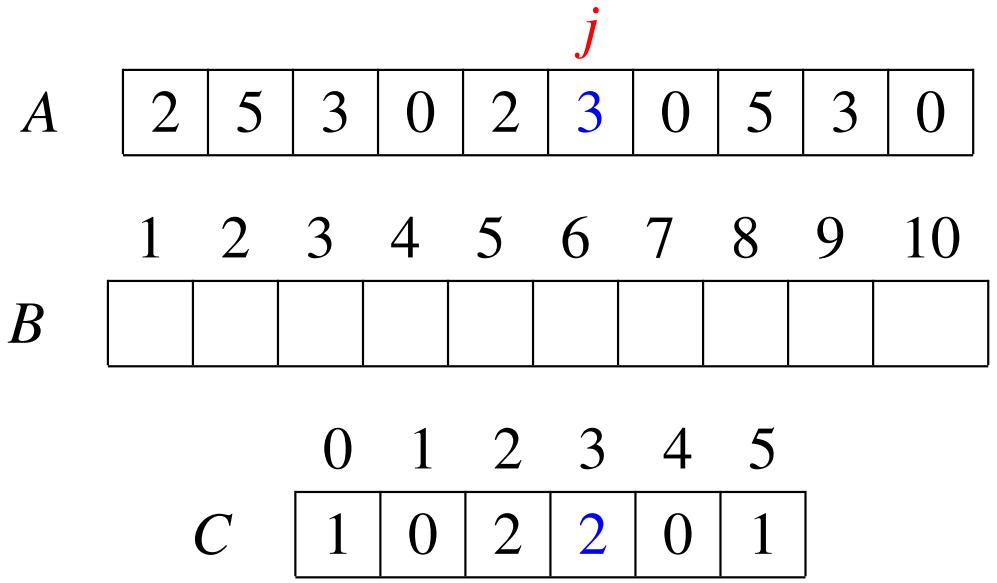
Ordenação por contagem

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



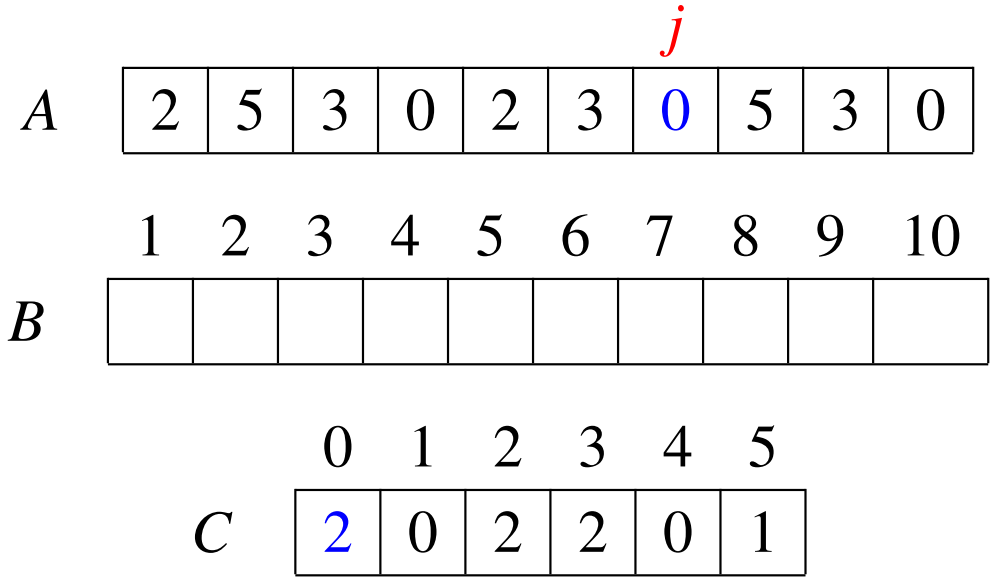
Ordenação por contagem

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



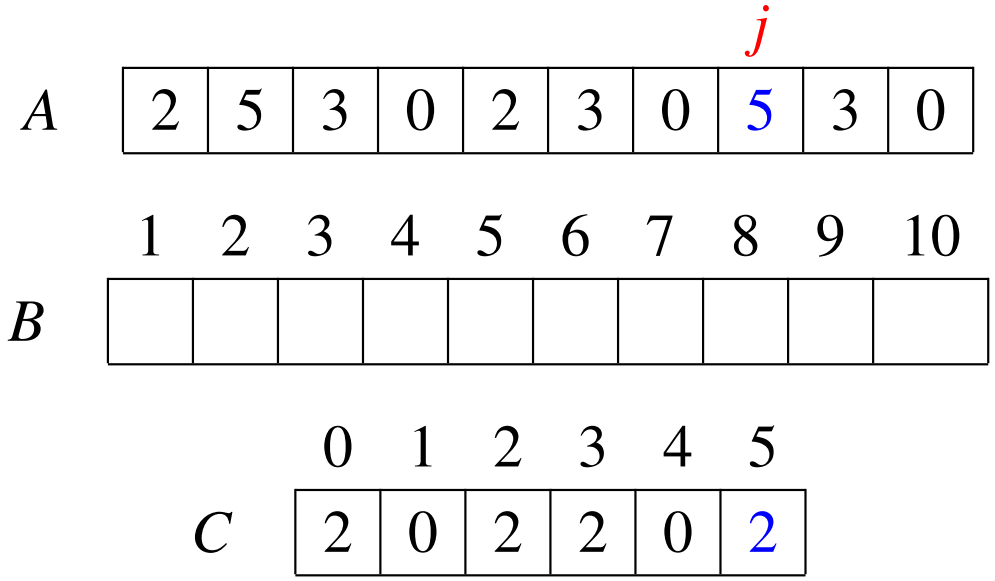
Ordenação por contagem

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



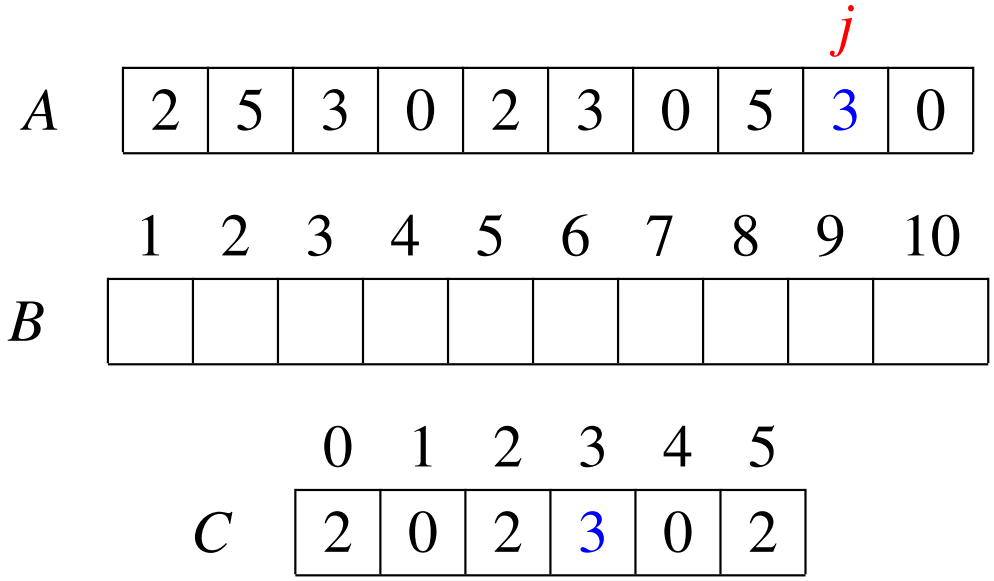
Ordenação por contagem

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



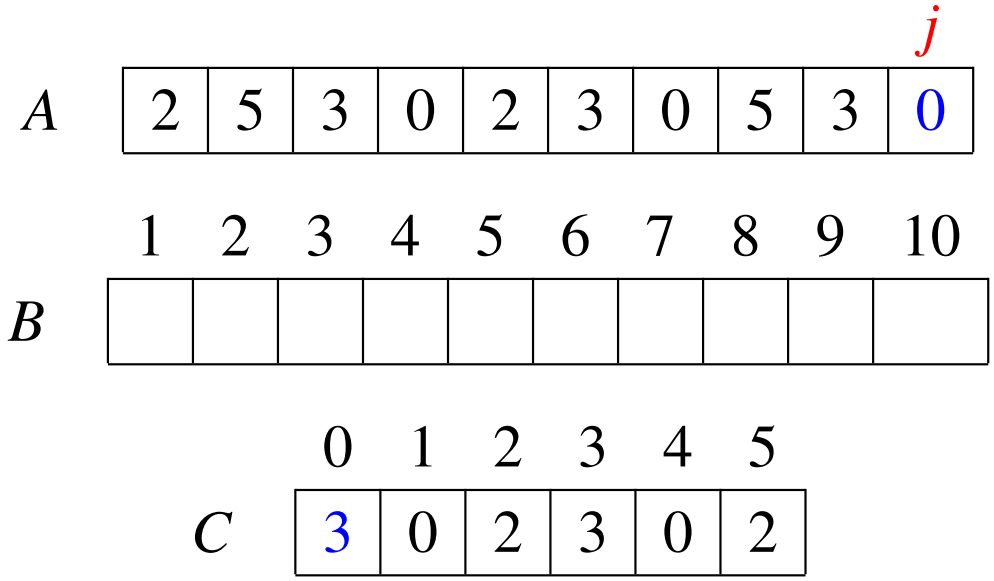
Ordenação por contagem

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



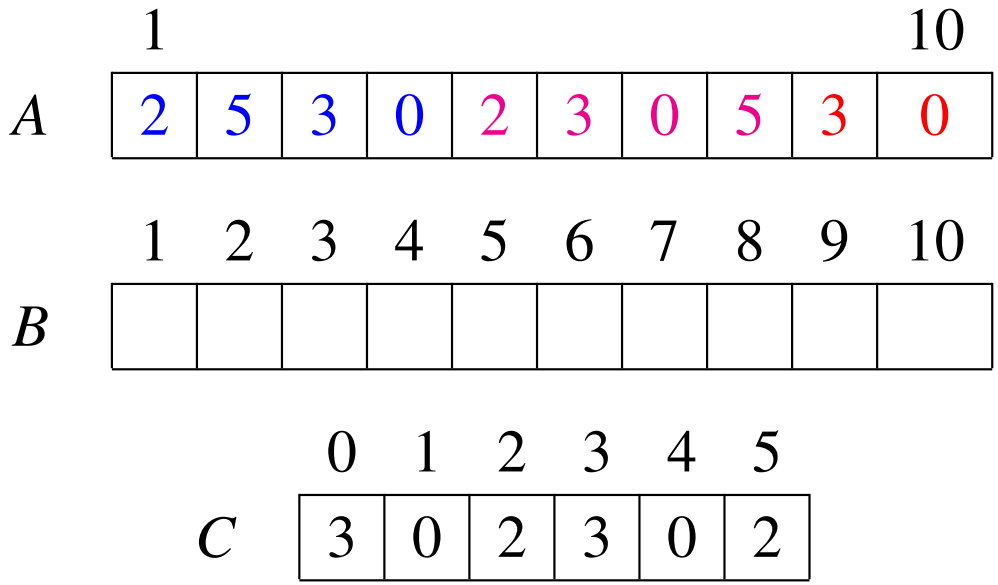
Ordenação por contagem

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



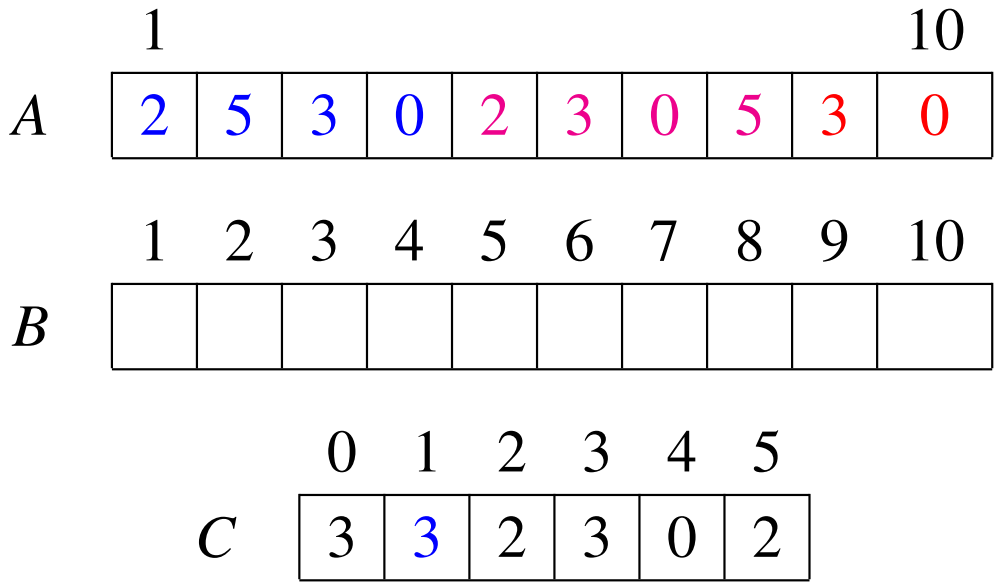
Ordenação por contagem

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



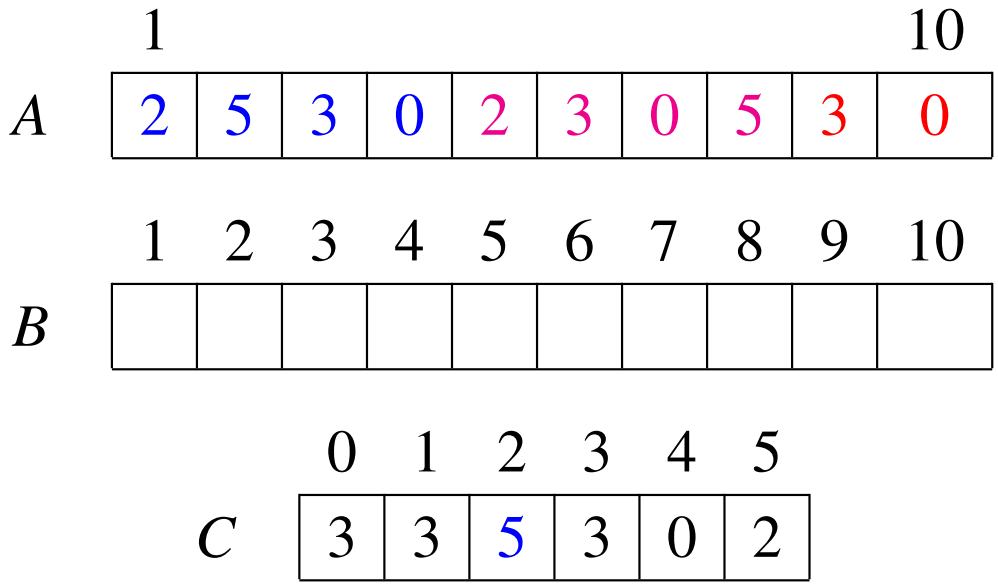
Ordenação por contagem

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



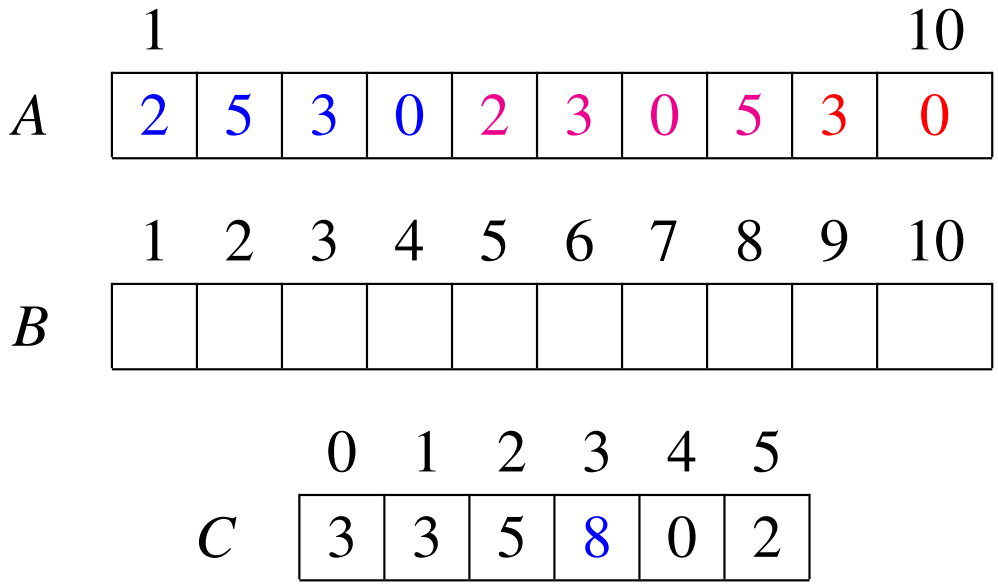
Ordenação por contagem

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



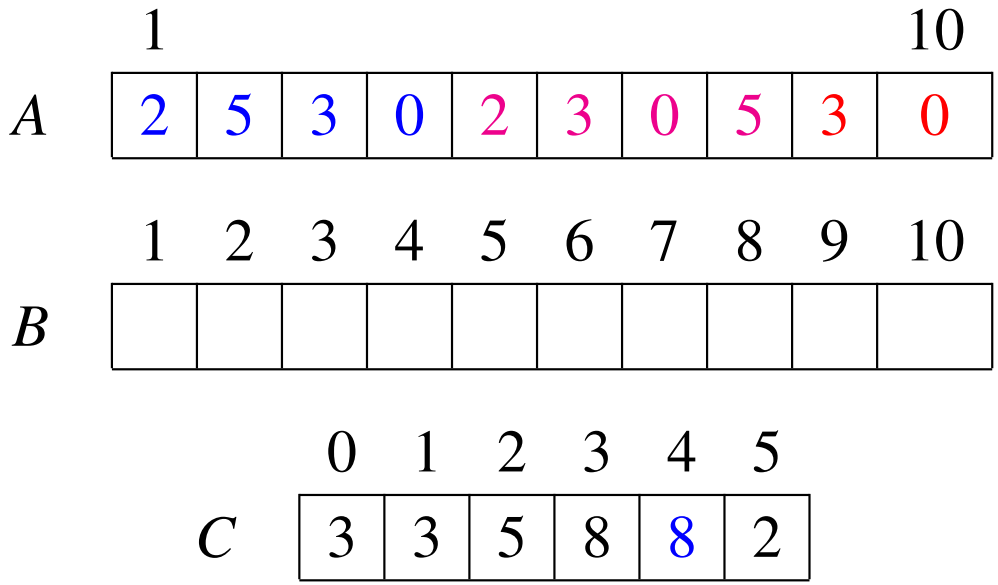
Ordenação por contagem

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



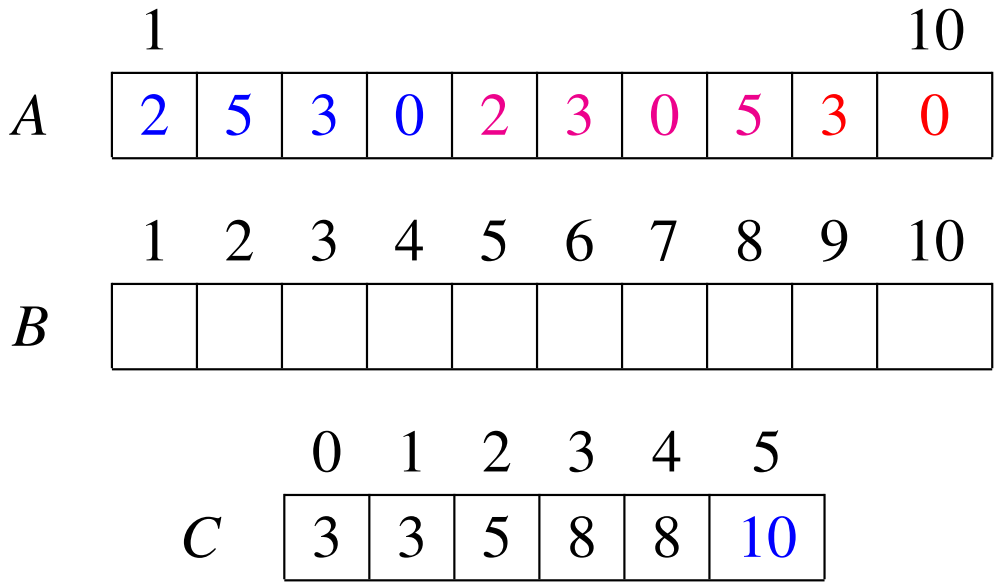
Ordenação por contagem

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



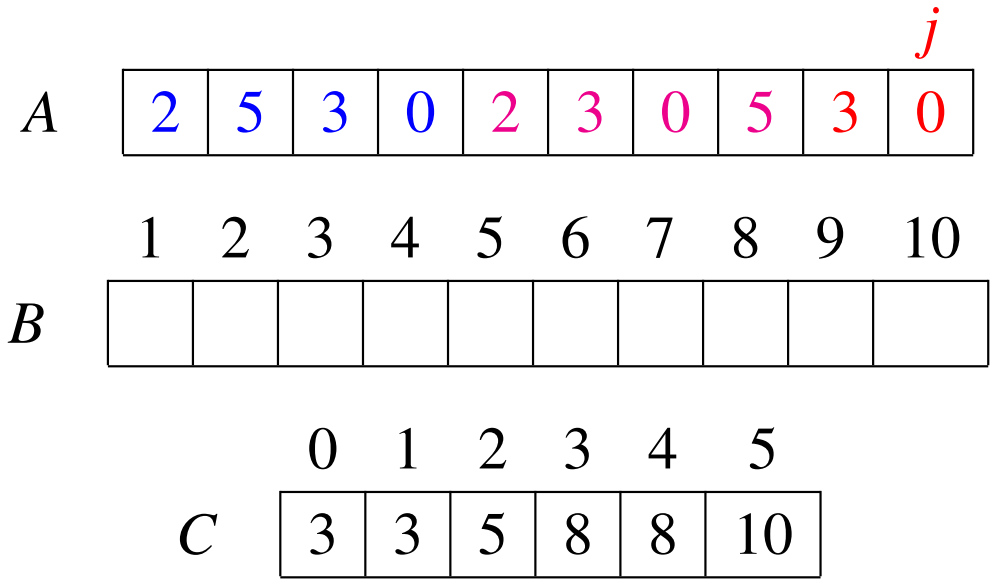
Ordenação por contagem

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



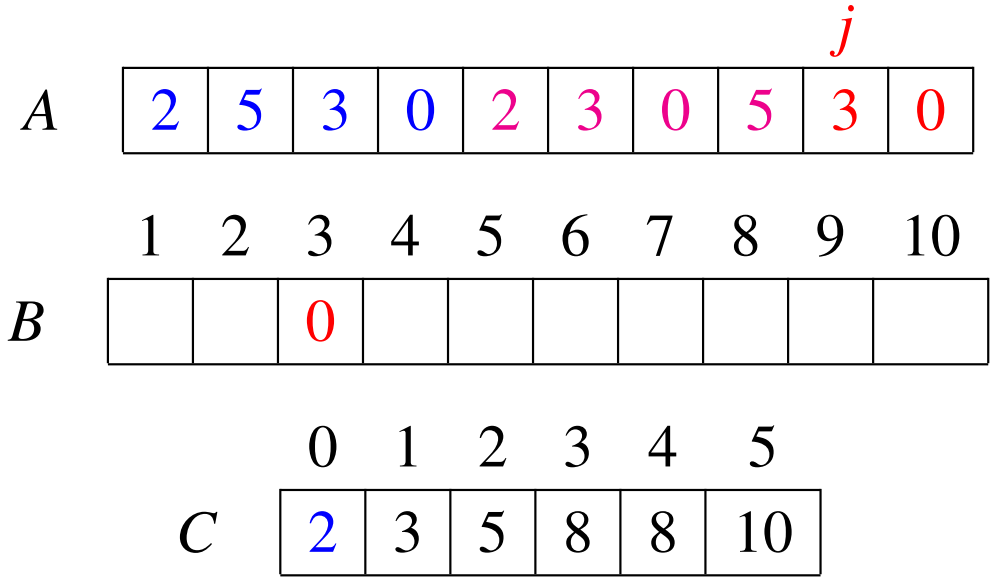
Ordenação por contagem

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



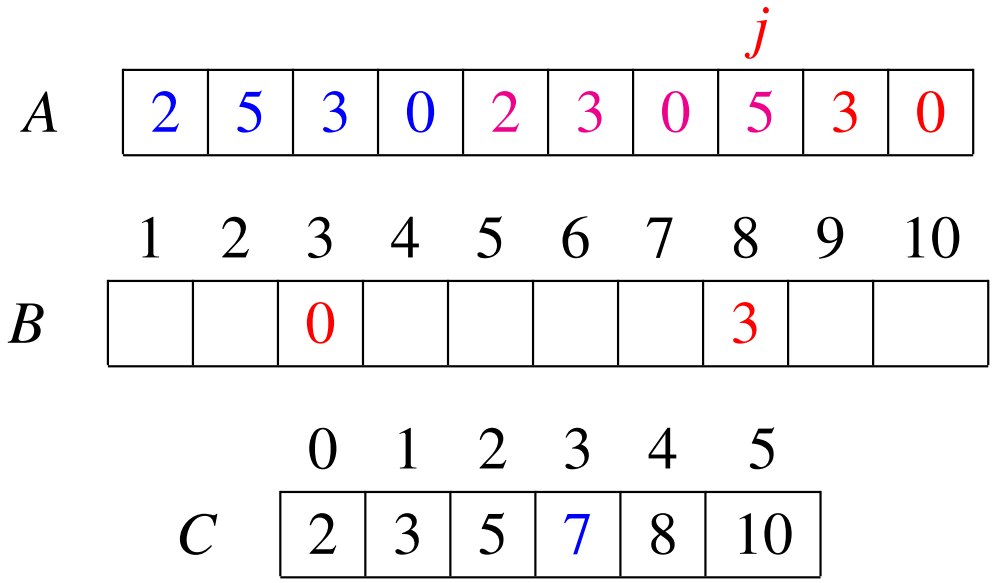
Ordenação por contagem

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



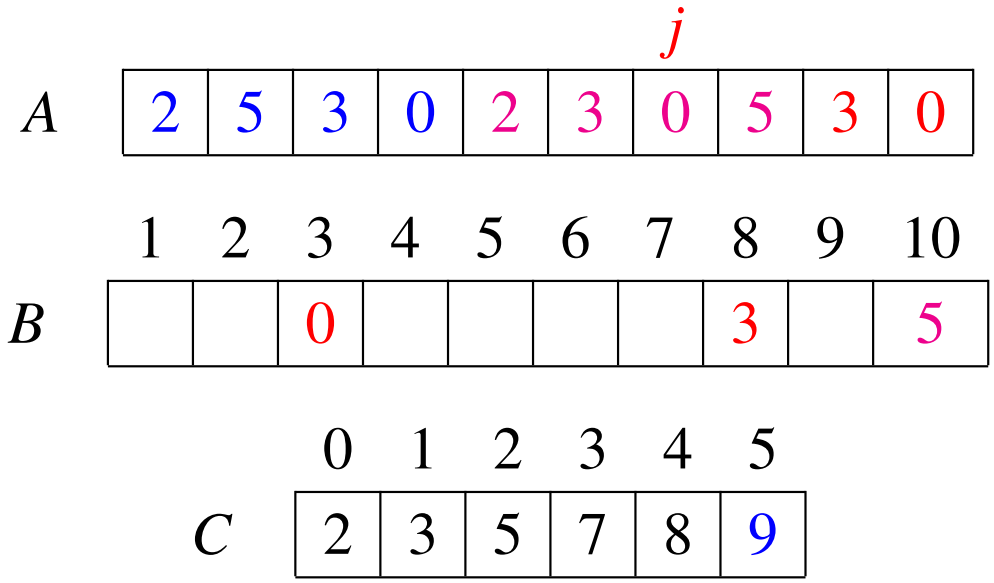
Ordenação por contagem

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



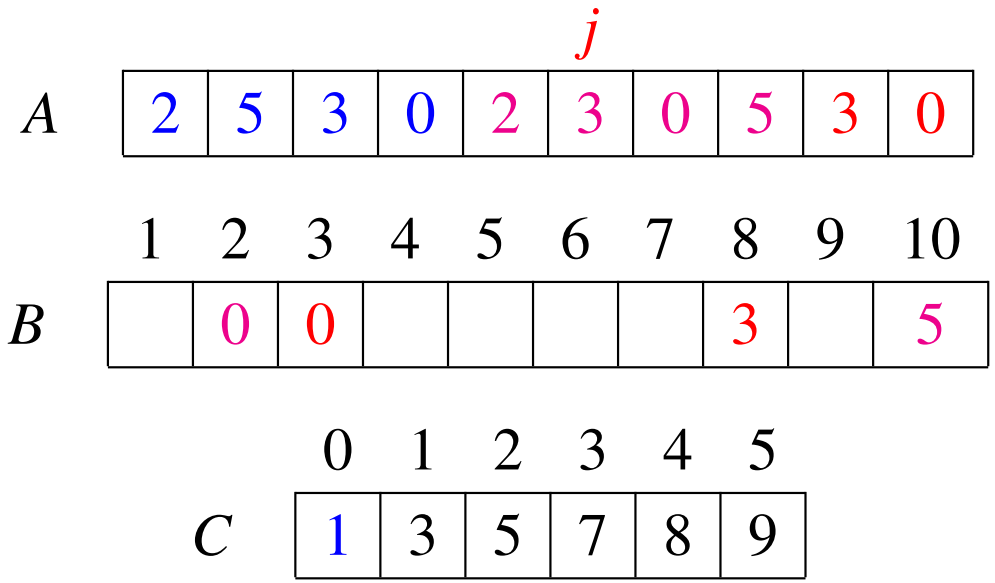
Ordenação por contagem

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



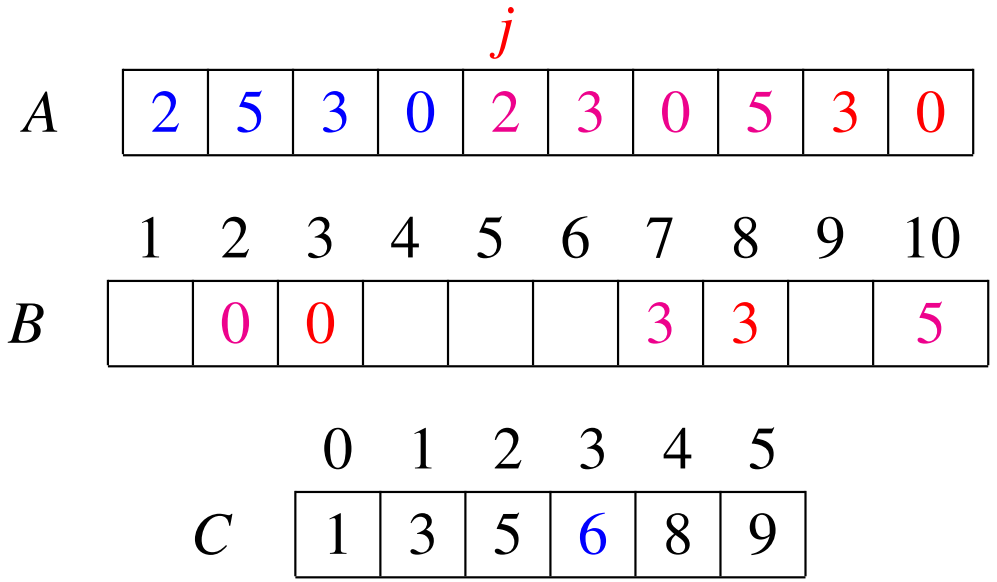
Ordenação por contagem

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



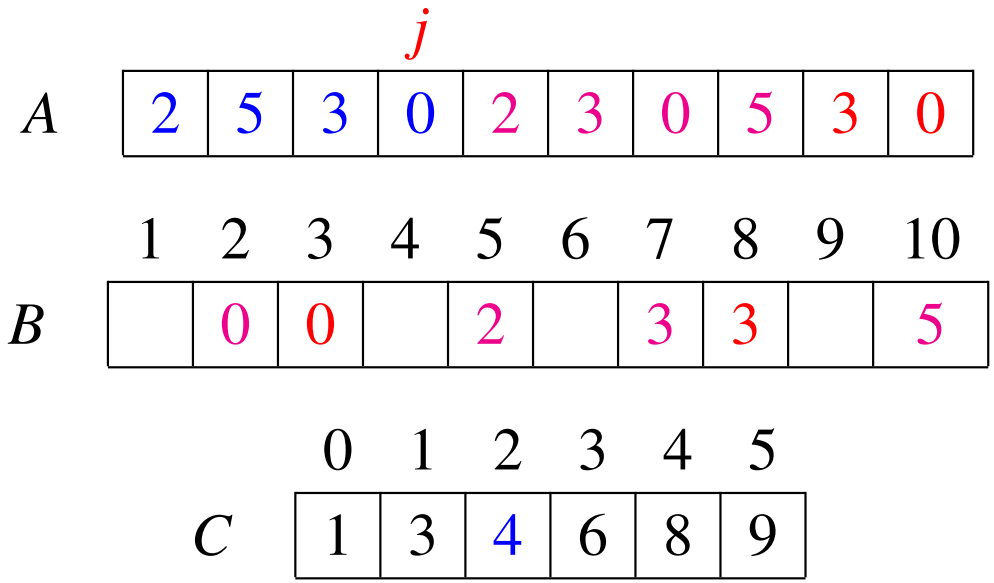
Ordenação por contagem

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



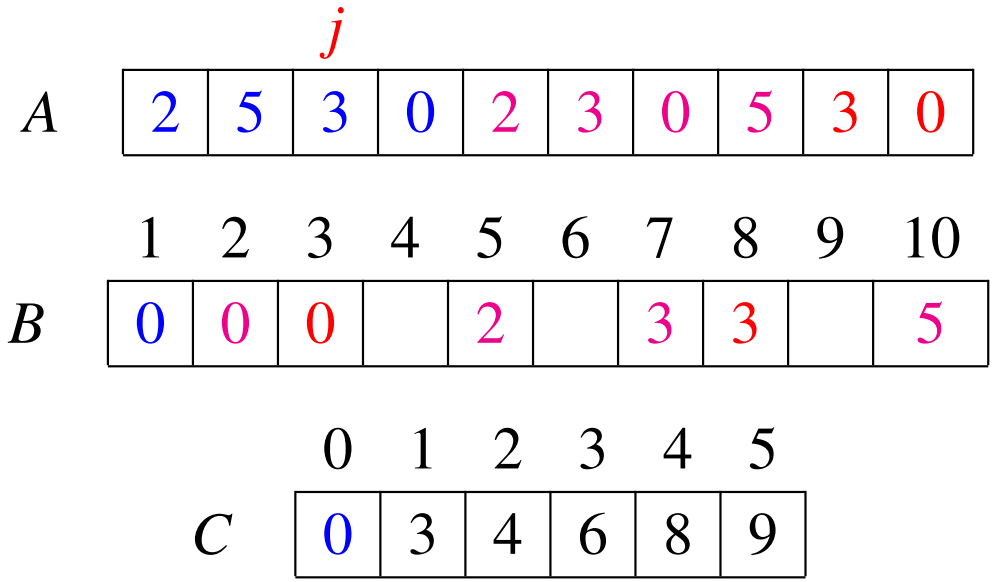
Ordenação por contagem

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



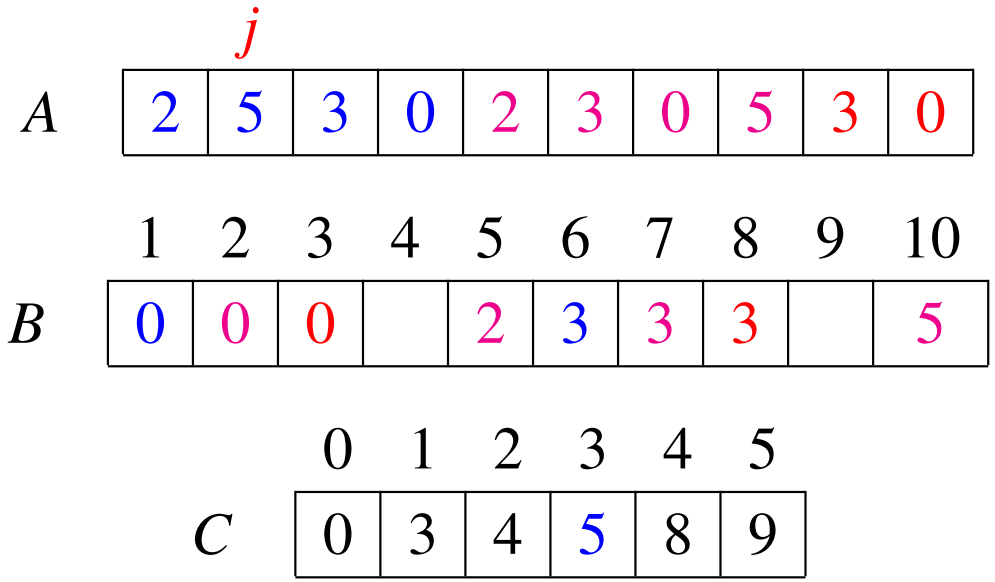
Ordenação por contagem

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



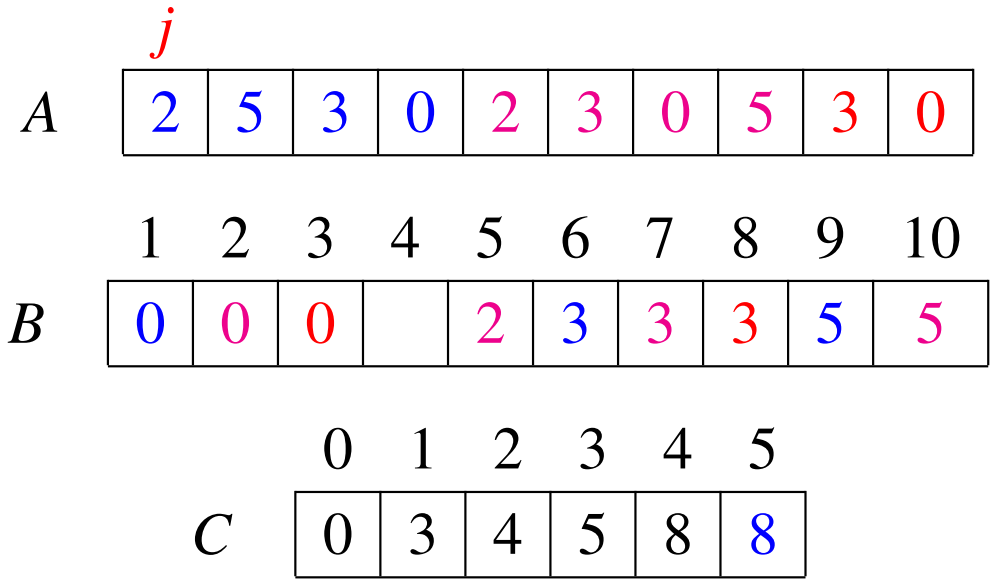
Ordenação por contagem

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



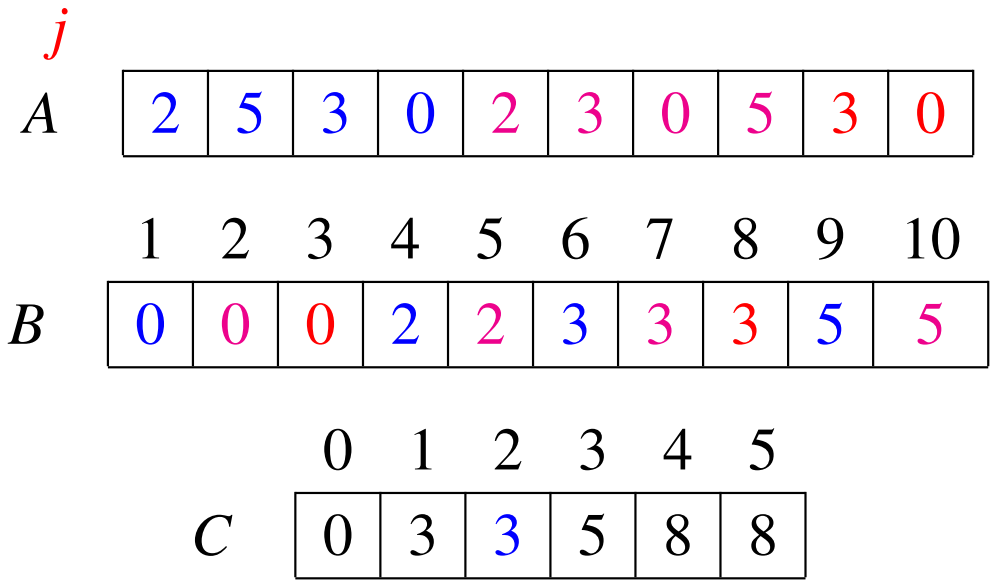
Ordenação por contagem

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



Ordenação por contagem

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



Ordenação por contagem

COUNTING-SORT (A, B, n, k)

1 **para** $i \leftarrow 0$ **até** k **faça**
2 $\quad C[i] \leftarrow 0$

3 **para** $j \leftarrow 1$ **até** n **faça**
4 $\quad C[A[j]] \leftarrow C[A[j]] + 1$
$\triangleright$ $C[i]$ é o número de js tais que $A[j] = i$

5 **para** $i \leftarrow 1$ **até** k **faça**
6 $\quad C[i] \leftarrow C[i] + C[i-1]$
$\triangleright$ $C[i]$ é o número de js tais que $A[j] \leq i$

7 **para** $j \leftarrow n$ **decrescendo até** 1 **faça**
8 $\quad B[C[A[j]]] \leftarrow A[j]$
9 $\quad C[A[j]] \leftarrow C[A[j]] - 1$

Obs: são feitas 0 comparações entre elementos do vetor.

Consumo de tempo

linha	consumo na linha
$1-2$	$\Theta(k)$
$3-4$	$\Theta(n)$
$5-6$	$\Theta(k)$
$7-9$	$\Theta(n)$

Consumo total: $\Theta(n+k)$

Conclusões

O consumo de tempo do COUNTING-SORT é $\Theta(n+k)$.

- se $k \leq n$ então consumo é $\Theta(n)$
- se $k \leq 10n$ então consumo é $\Theta(n)$
- se $k = \mathrm{O}(n)$ então consumo é $\Theta(n)$
- se $k \geq n^2$ então consumo é $\Theta(k)$
- se $k = \Omega(n)$ então consumo é $\Theta(k)$

Estabilidade

A propósito: COUNTING-SORT é estável:

na saída, chaves com mesmo valor estão na mesma ordem que apareciam na entrada.

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

Ordenação digital (=radix sort)

Exemplo:

329
457
657
839
436
720
355

Ordenação digital (=radix sort)

Exemplo:

329	720
457	355
657	436
839	457
436	657
720	329
355	839

Ordenação digital (=radix sort)

Exemplo:

329	720	720
457	355	329
657	436	436
839	457	839
436	657	355
720	329	457
355	839	657

Ordenação digital (=radix sort)

Exemplo:

329	720	720	329
457	355	329	355
657	436	436	436
839	457	839	457
436	657	355	657
720	329	457	720
355	839	657	839

Ordenação digital (=radix sort)

Exemplo:

329	720	720	329
457	355	329	355
657	436	436	436
839	457	839	457
436	657	355	657
720	329	457	720
355	839	657	839

Cada $A[j]$ têm d dígitos decimais:
$A[j] = a_d\, 10^{d-1} + \cdots + a_2\, 10^1 + a_1\, 10^0$
Exemplo com $d = 3$: $\quad 3 \cdot 10^2 + 2 \cdot 10 + 9$

Ordenação digital

RADIX-SORT (A, n, d)
1 para $i \leftarrow 1$ até d faça
2 $\quad \triangleright$ 1 até d e não o contrário!
3 $\quad$ ordene $A[1\,..\,n]$ pelo dígito i

Linha 3:

- faz ordenação $A[j_1\,..\,j_n]$ de $A[1\,..\,n]$ tal que

$$A[j_1]_i \leq \cdots \leq A[j_n]_i;$$

- ordenação deve ser estável; e
- use COUNTING-SORT.

Exemplos

- dígitos decimais: $\Theta(dn)$
- dígitos em $0\,..\,k$: $\Theta(d\,(n+k))$.

Exemplo com $d = 5$ e $k = 127$:

$$a_5 128^4 + a_4 128^3 + a_3 128^2 + a_2 128 + a_1$$

sendo $0 \leq a_i \leq 127$

Conclusão

Dados n números com b bits e um inteiro $r \leq b$, RADIX-SORT ordena esses números em tempo

$$\Theta\left(\frac{b}{r}(n + 2^r)\right).$$

Prova: Considere cada chave com $d = \lceil b/r \rceil$ dígitos com r bits cada.

Conclusão

Dados n números com b bits e um inteiro $r \leq b$,
RADIX-SORT ordena esses números em tempo

$$\Theta\left(\frac{b}{r}(n + 2^r)\right).$$

Prova: Considere cada chave com $d = \lceil b/r \rceil$ dígitos com r bits cada.
Use COUNTING-SORT com $k = 2^r - 1$.

Conclusão

Dados n números com b bits e um inteiro $r \leq b$,
RADIX-SORT ordena esses números em tempo

$$\Theta\left(\frac{b}{r}(n+2^r)\right).$$

Prova: Considere cada chave com $d = \lceil b/r \rceil$ dígitos com r bits cada.
Use COUNTING-SORT com $k = 2^r - 1$.
Cada passada do COUNTING-SORT: $\Theta(n+k) = \Theta(n+2^r)$.

Conclusão

Dados n números com b bits e um inteiro $r \leq b$,
RADIX-SORT ordena esses números em tempo

$$\Theta\left(\frac{b}{r}(n+2^r)\right).$$

Prova: Considere cada chave com $d = \lceil b/r \rceil$ dígitos com r bits cada.
Use COUNTING-SORT com $k = 2^r - 1$.
Cada passada do COUNTING-SORT: $\Theta(n+k) = \Theta(n+2^r)$.
Tempo total:

$$\Theta(d(n+2^r)) = \Theta\left(\frac{b}{r}(n+2^r)\right).$$

Bucket sort

Bucket sort: algoritmo de ordenação em tempo esperado linear.

Descrito em CRLS 8.4.

Exercícios

Exercício 1.A

O seguinte algoritmo promete rearranjar o vetor $A[1\,..\,n]$ em ordem crescente supondo que cada $A[i]$ está em $\{0, \ldots, k\}$. O algoritmo está correto?

C-SORT (A, n, k)
  para $i \leftarrow 0$ **até** k **faça**
    $C[i] \leftarrow 0$
  para $j \leftarrow 1$ **até** n **faça**
    $C[A[j]] \leftarrow C[A[j]] + 1$
  $j \leftarrow 1$
  para $i \leftarrow 0$ **até** k **faça**
    enquanto $C[i] > 0$ **faça**
      $A[j] \leftarrow i$
      $j \leftarrow j + 1$
      $C[i] \leftarrow C[i] - 1$

Qual o consumo de tempo do algoritmo?

Mais exercícios

Exercício 1.B

O seguinte algoritmo promete rearranjar o vetor $A[1\,..\,n]$ em ordem crescente supondo que cada $A[j]$ está em $\{1, \ldots, k\}$. O algoritmo está correto? Estime, em notação O, o consumo de tempo do algoritmo.

```
VITO-SORT (A, n, k)
1    i ← 1
2    para a ← 1 até k − 1 faça
3        para j ← i até n faça
4            se A[j] = a
5                então A[j] ↔ A[i]    ▷ troca
6                     i ← i + 1
```

Exercício 1.C

Suponha que os components do vetor $A[1\,..\,n]$ estão todos em $\{0, 1\}$. Prove que $n - 1$ comparações são suficientes para rearranjar o vetor em ordem crescente.

Exercício 1.D

Qual a principal invariante do algoritmo RADIX-SORT?