Algorithms and Data Structures 2

9 – Beyond comparison sorts

Dr Michele Sevegnani

School of Computing Science
University of Glasgow

michele.sevegnani@glasgow.ac.uk

Outline

- **COUNTING-SORT**
- **RADIX-SORT**

COUNTING-SORT

- **Efficient sorting algorithm for integers in the range 0 to k**
  - Invented by H. H. Seward in 1954

- **Sort integers without comparisons in linear time O(n+k)**
- **Requires O(n+k) memory**
- **Stable**

- **Often used as a subroutine for RADIX-SORT an O(d(n+k)) algorithm to sort integers according to their digits**
  - Input elements have d digits

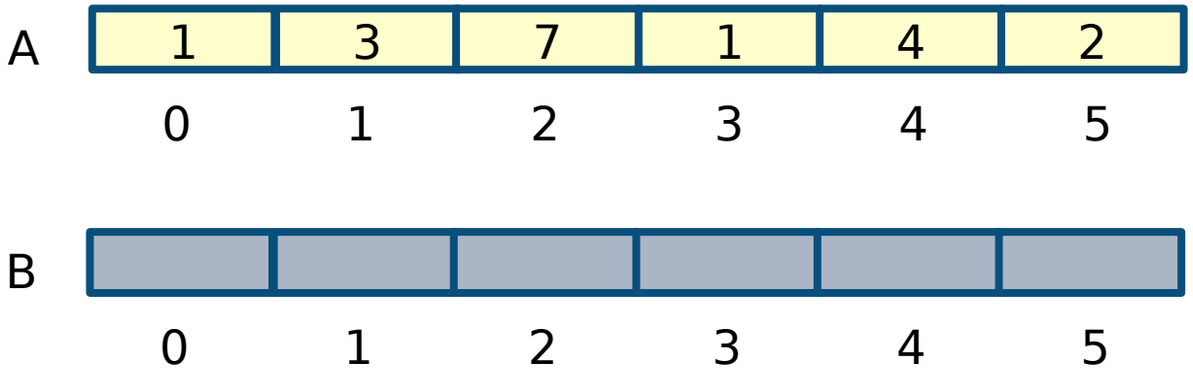
COUNTING-SORT

- **Input**
  - Array A of size n of integers in the range 0 to k
  - Array B of size n
- **Output**
  - Array B contains the elements of A in sorted order
- **Three steps**
  1. Count occurrences
  2. Modify counts to have a running sum. The modified count array indicates the position of each object in the output sequence
  3. Use running sum to copy elements to B

```
COUNTING-SORT(A,B,k)
  new C[0..k] // all zeros
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

Example execution of COUNTING-SORT(A,B,7)

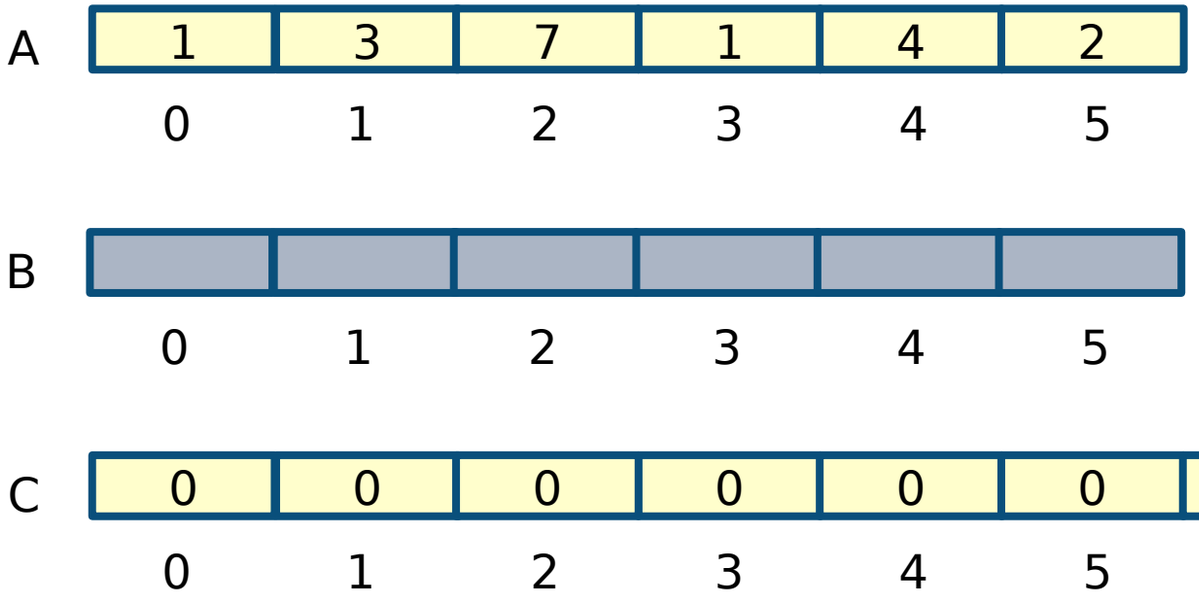
- **A = [1,3,7,1,4,2] and B has length n = 6**




```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

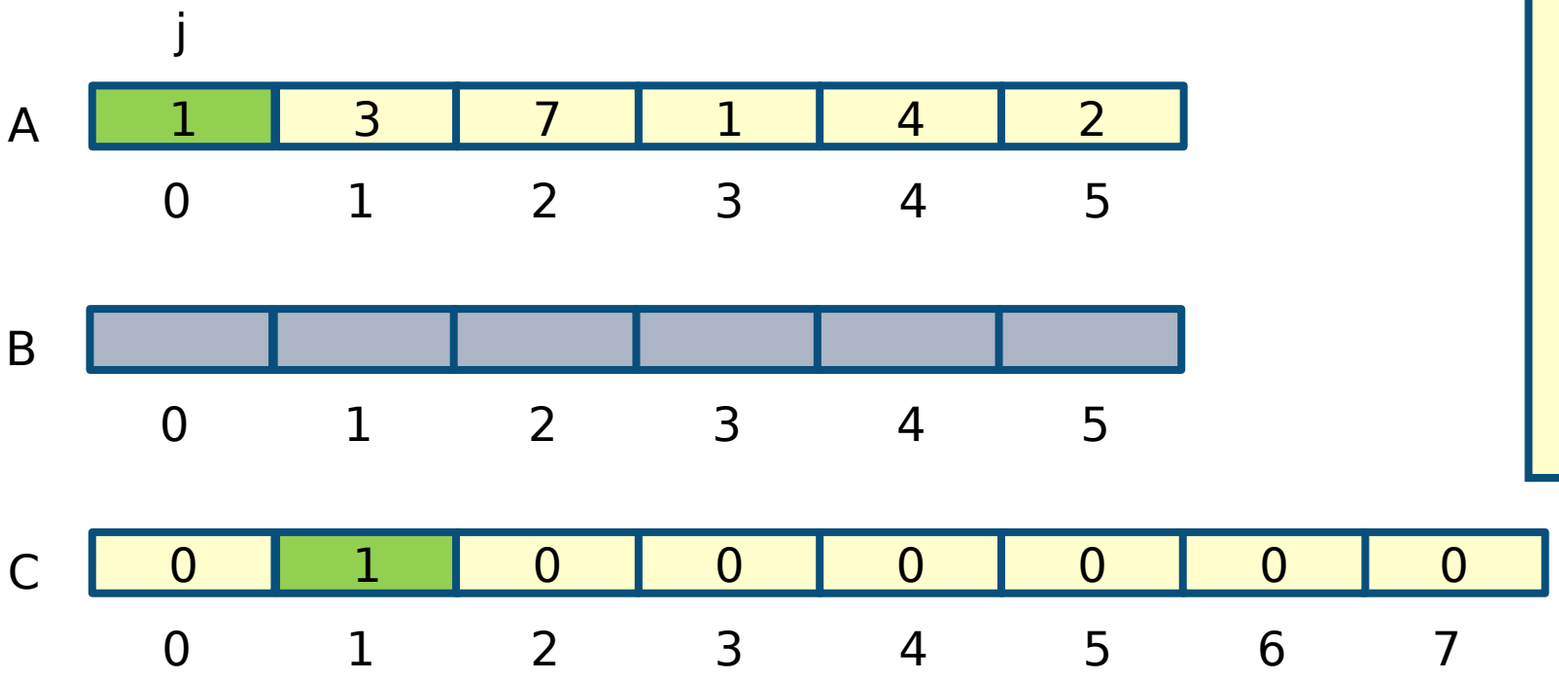



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - Allocate and initialise C

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

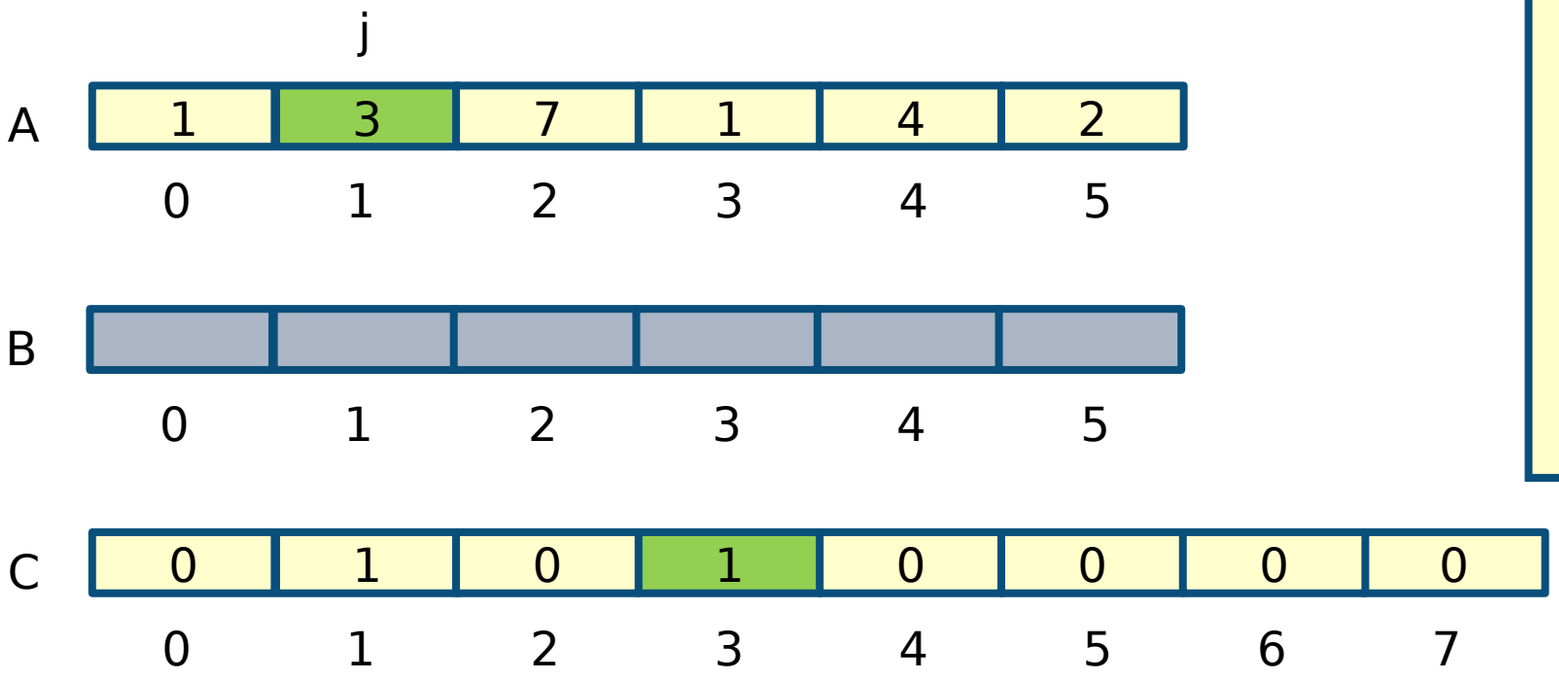



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - Record one occurrence of 1 in C[1]

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

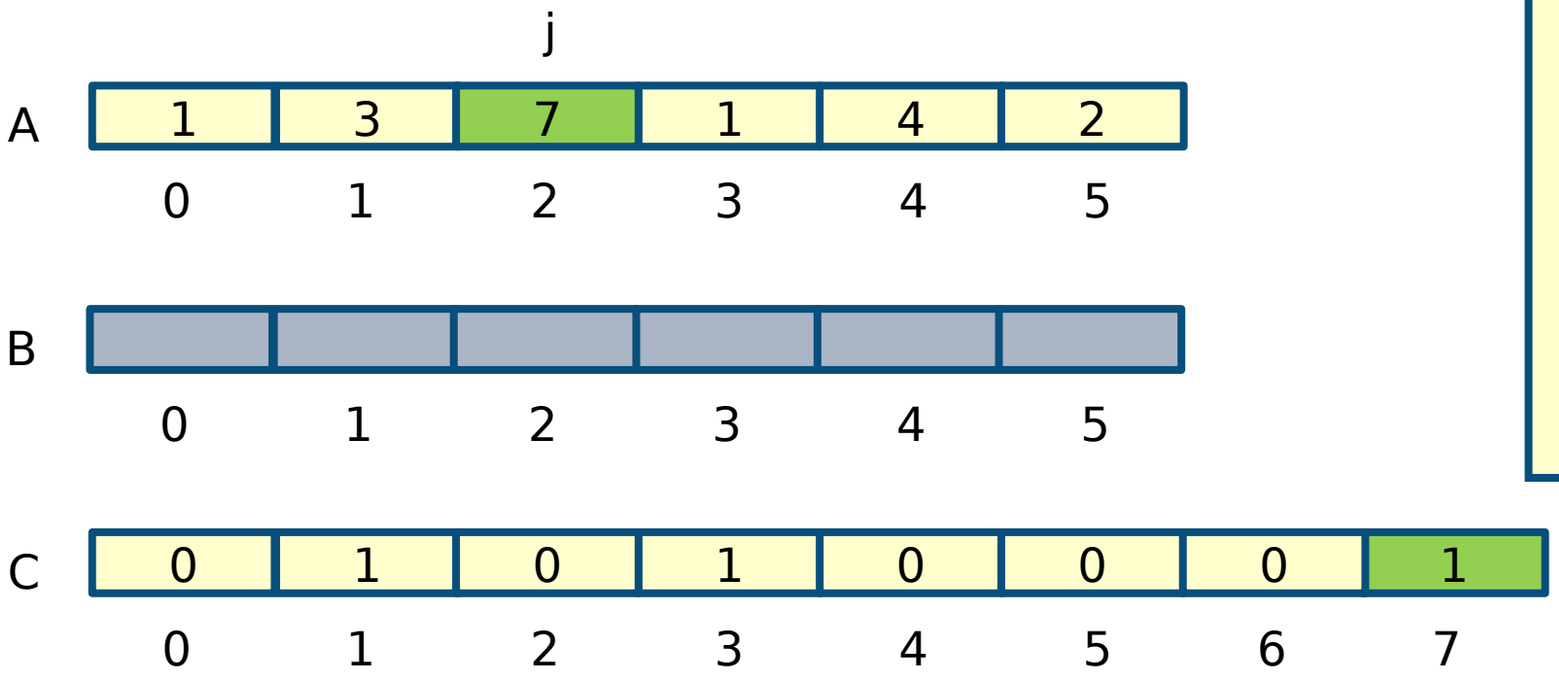



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - Record one occurrence of 3 in C[3]

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

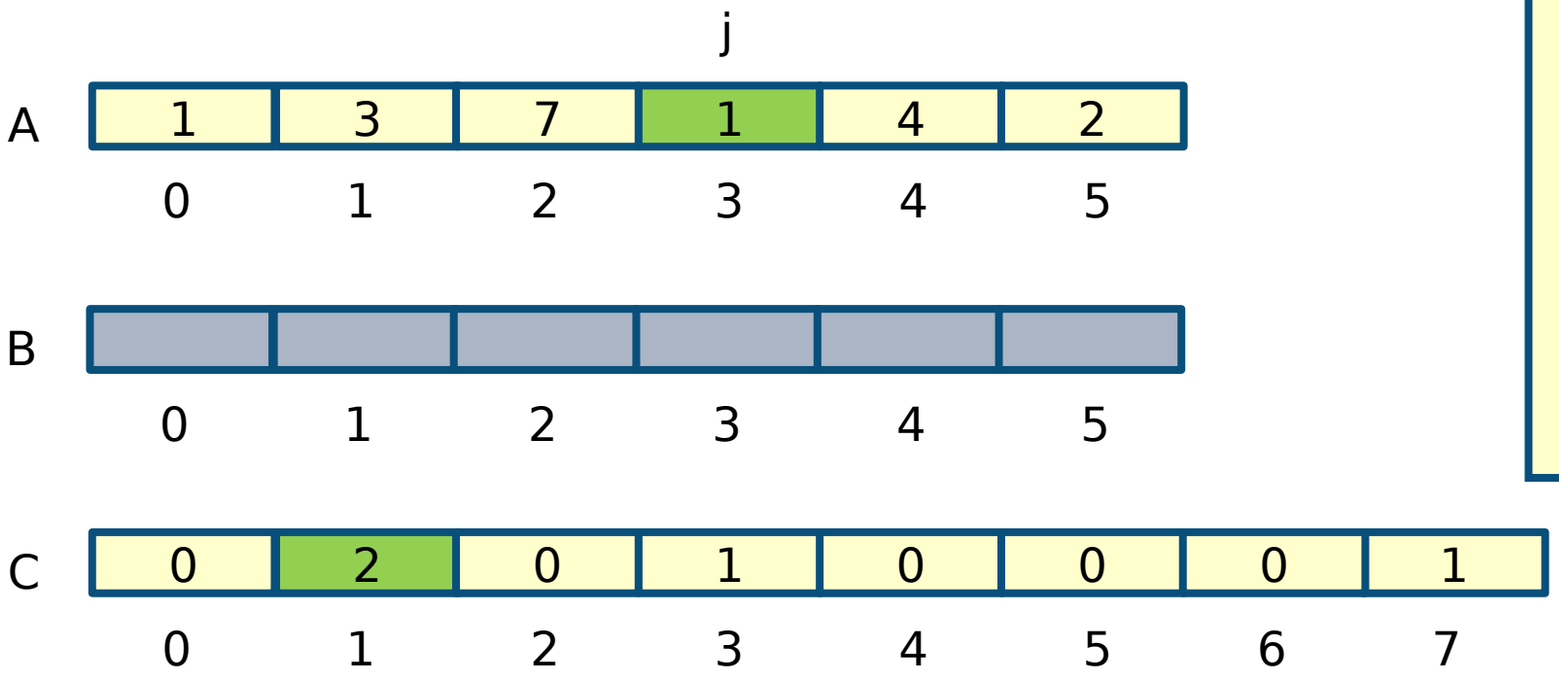



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - Record one occurrence of 7 in C[7]

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

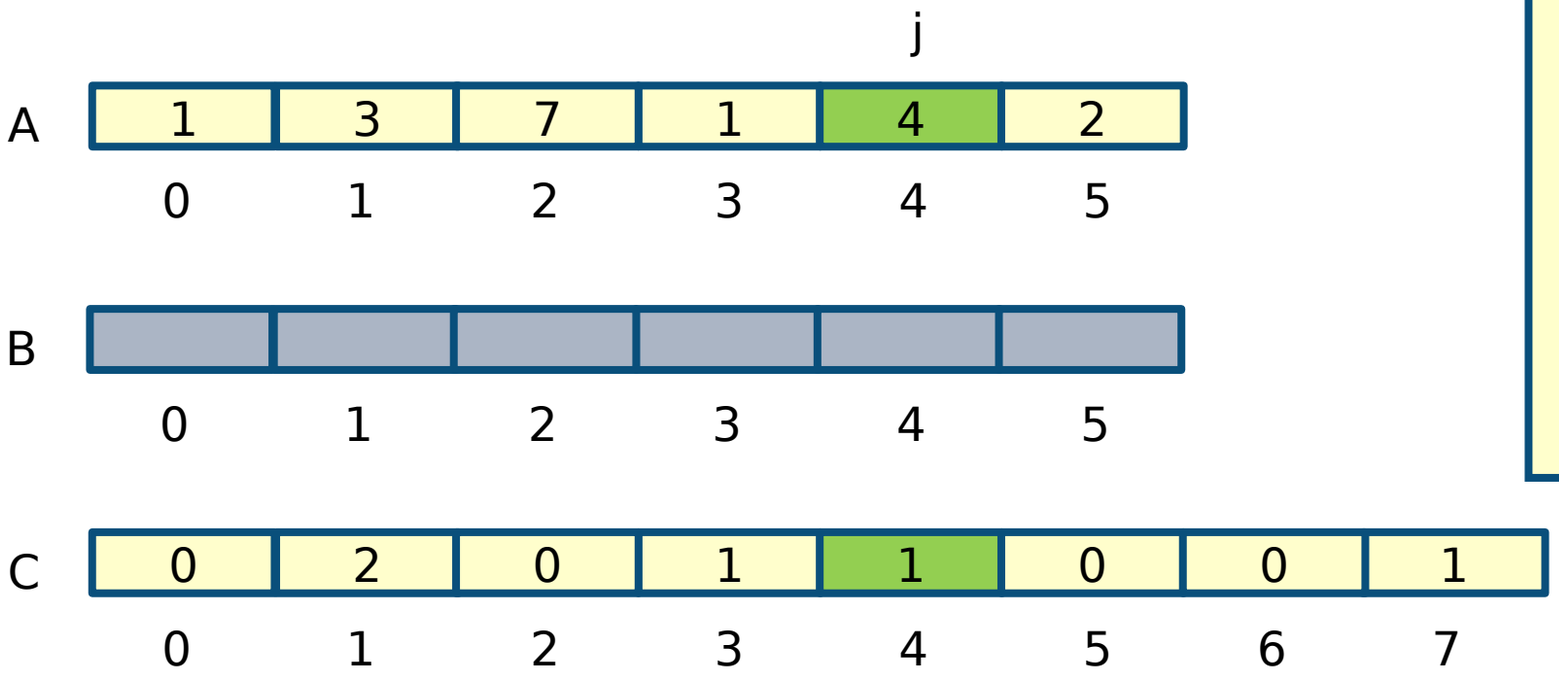



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - Record another occurrence of 1 in C[1]

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

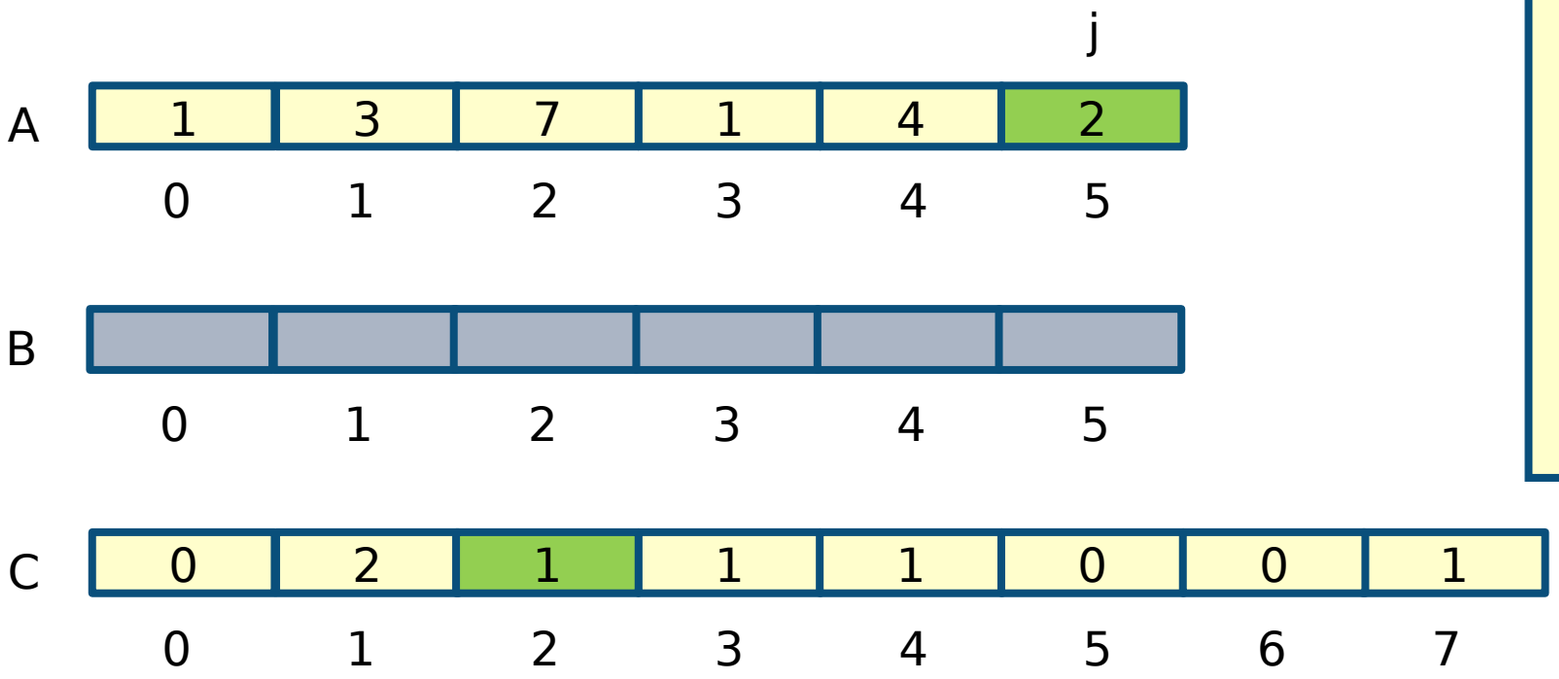



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - Record one occurrence of 4 in C[4]

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

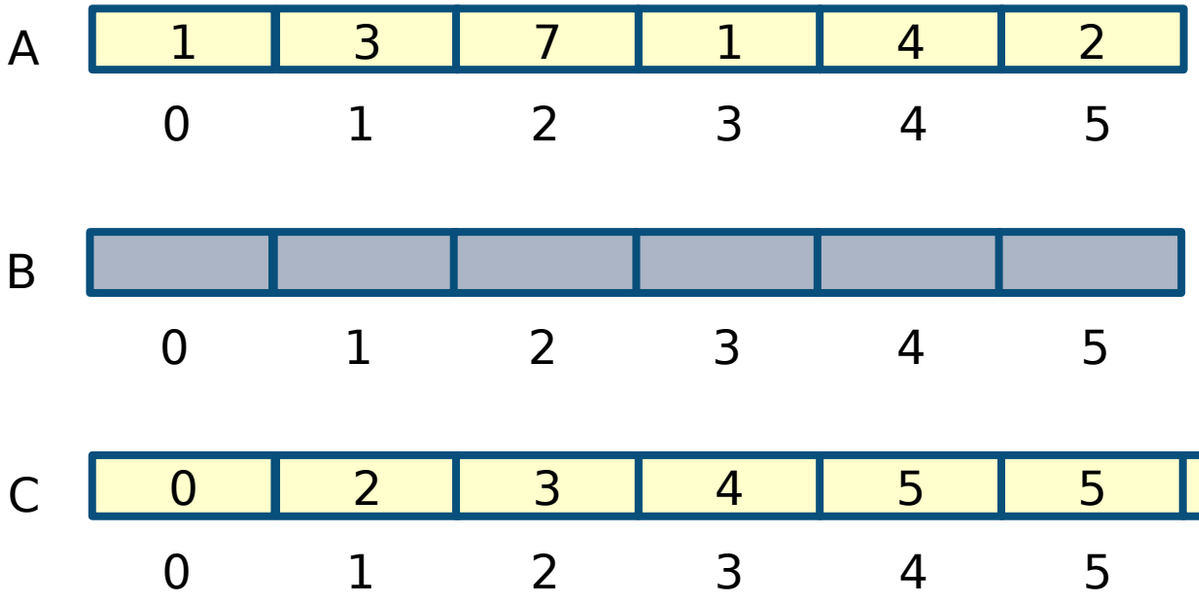



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - Record one occurrence of 2 in C[2]

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

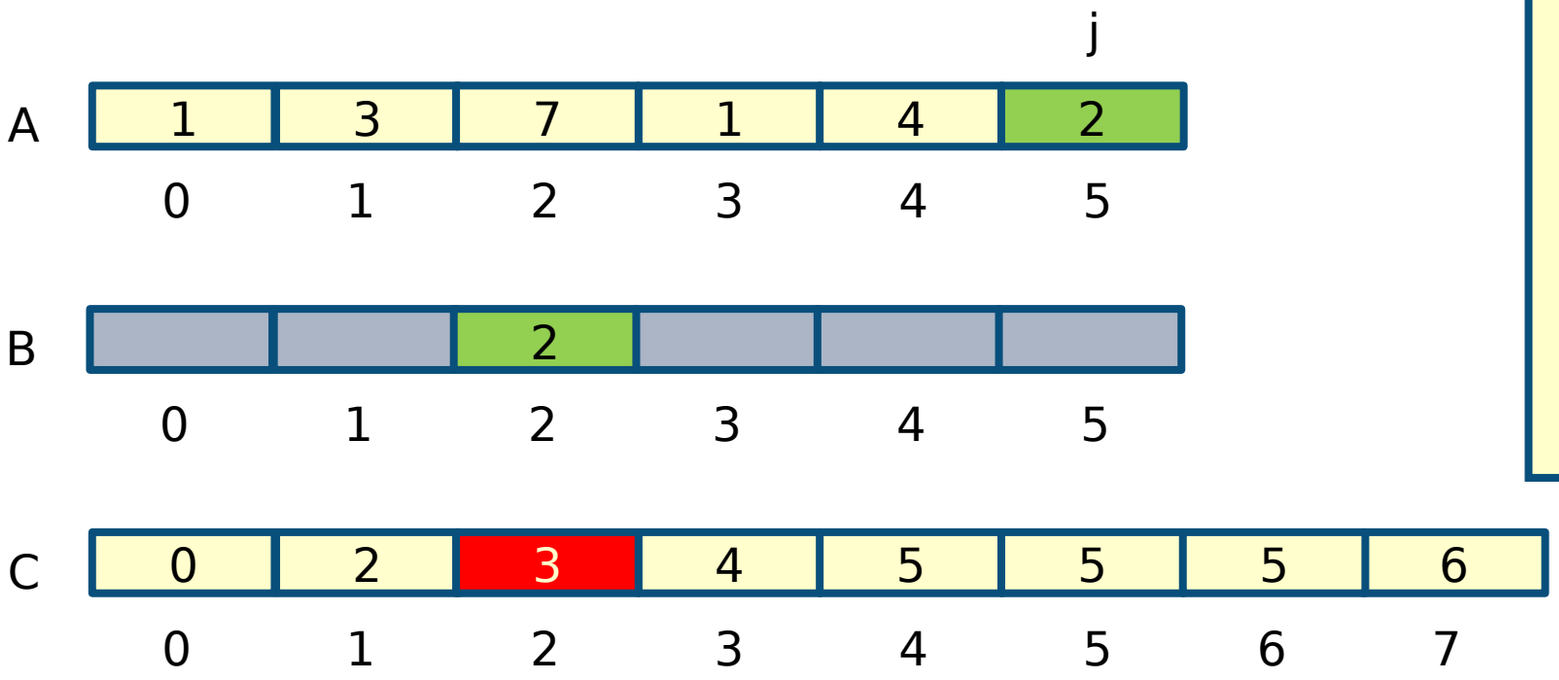



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - The second for loop computes the running sum on the elements of C

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

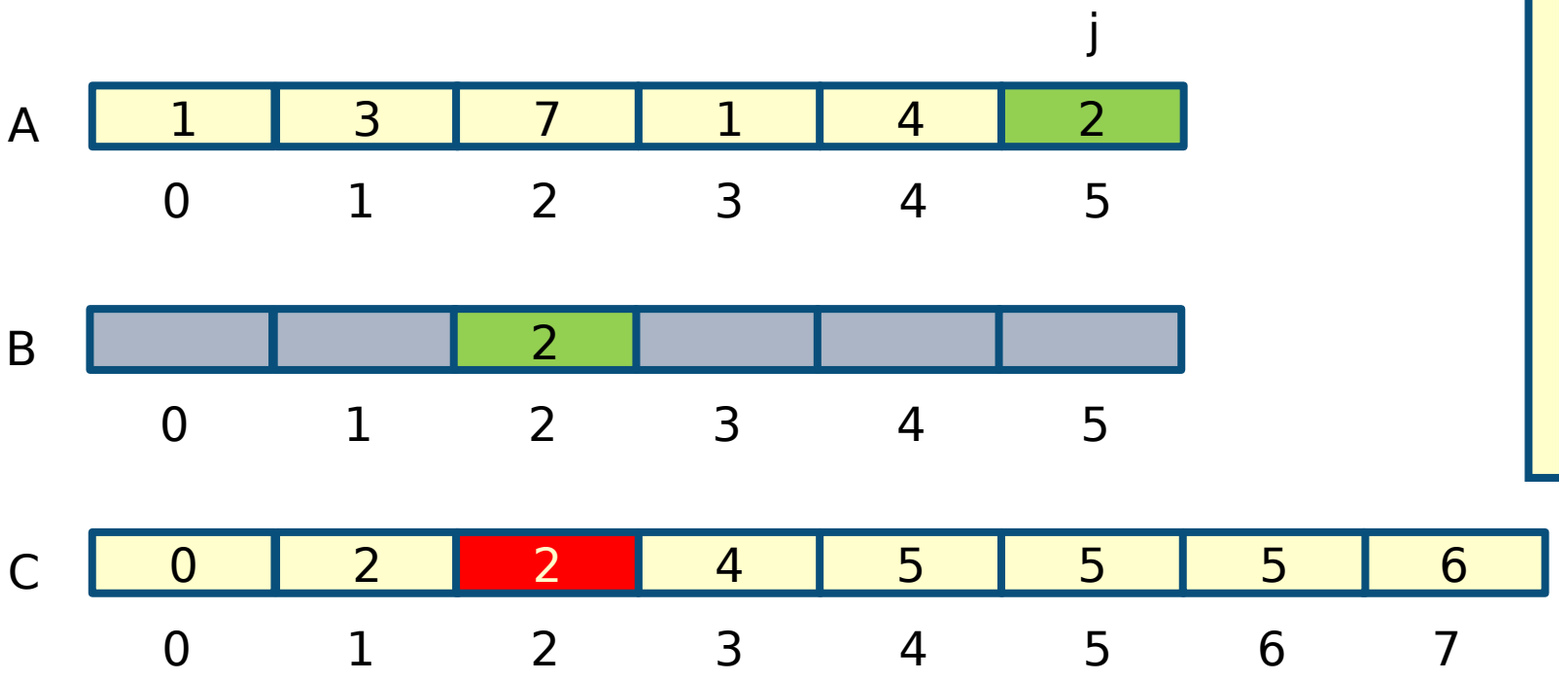



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - 2 is put in its final position C[2] – 1 = 2 in B

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

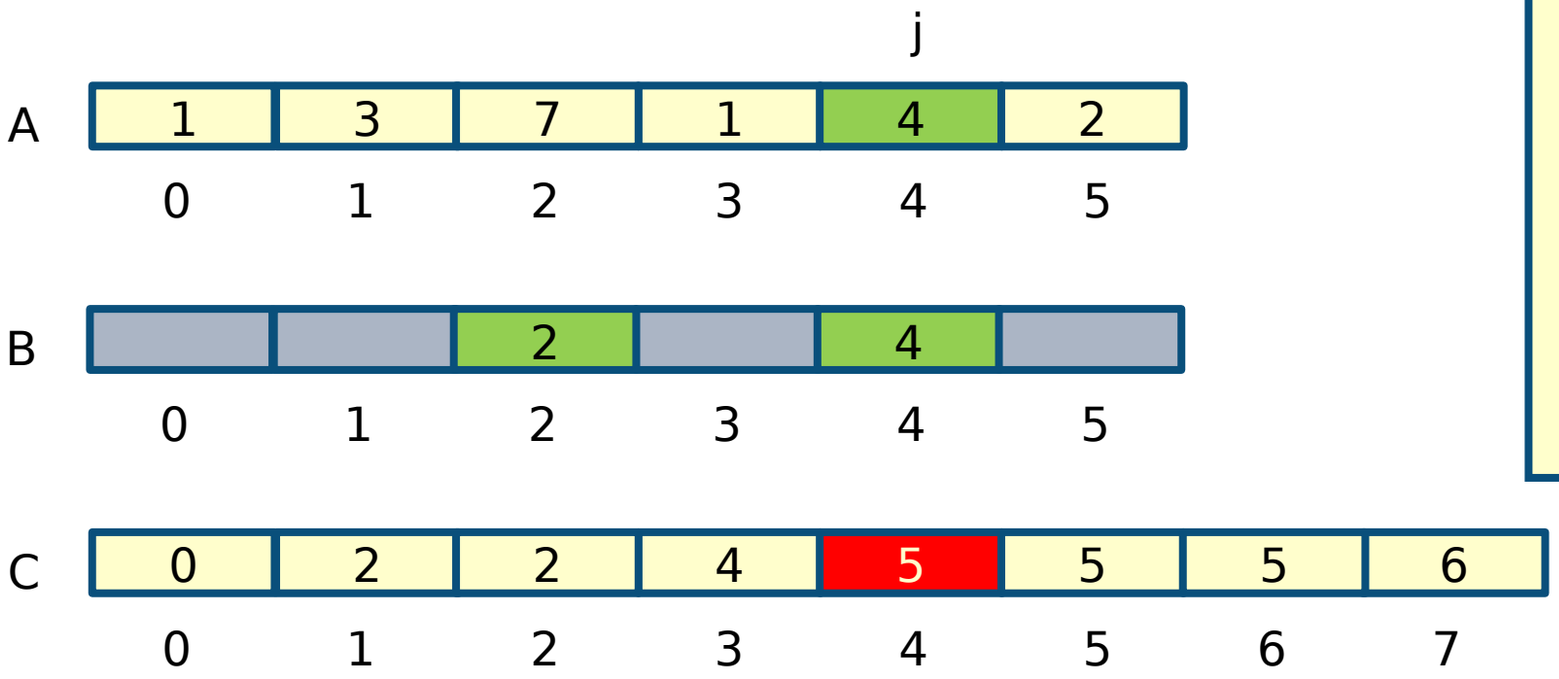



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - The corresponding count is decreased

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

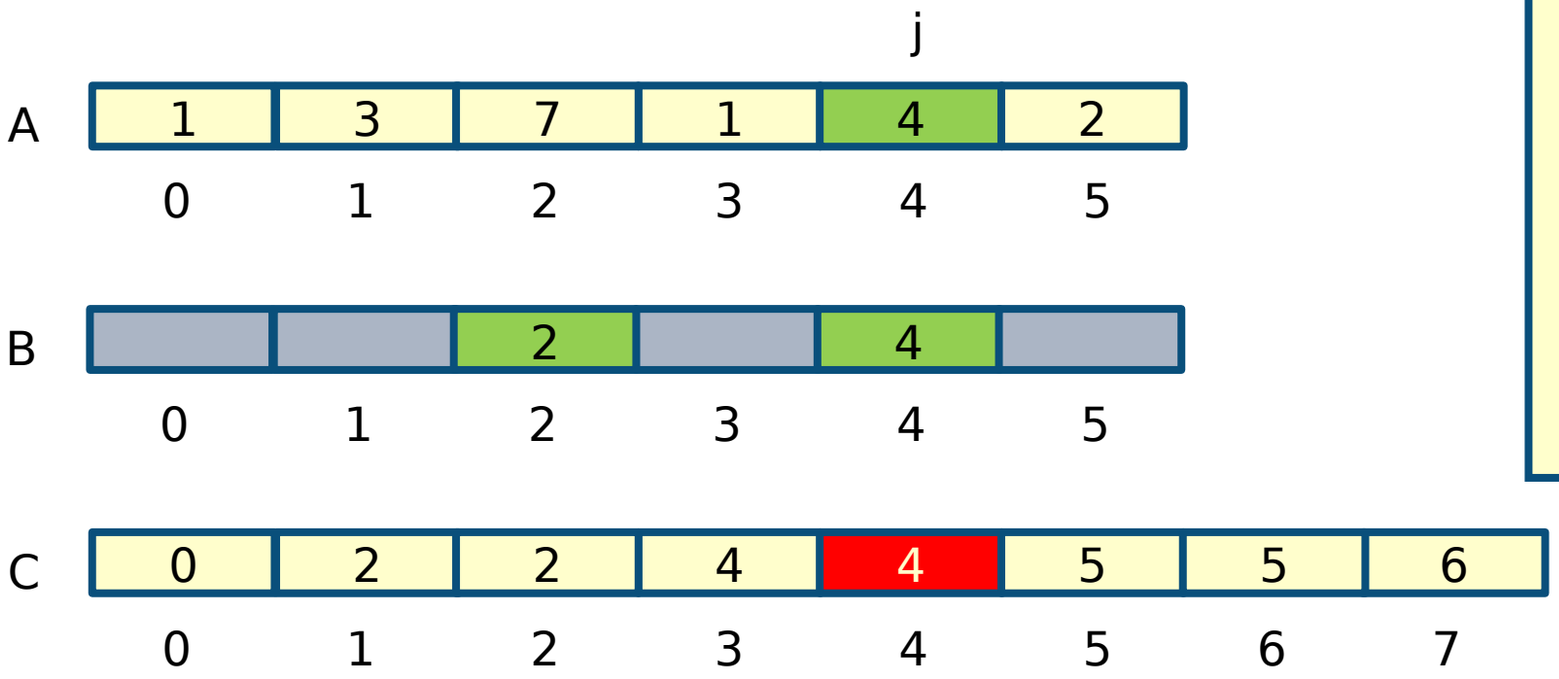



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - 4 is put in its final position C[4] – 1 = 4 in B

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

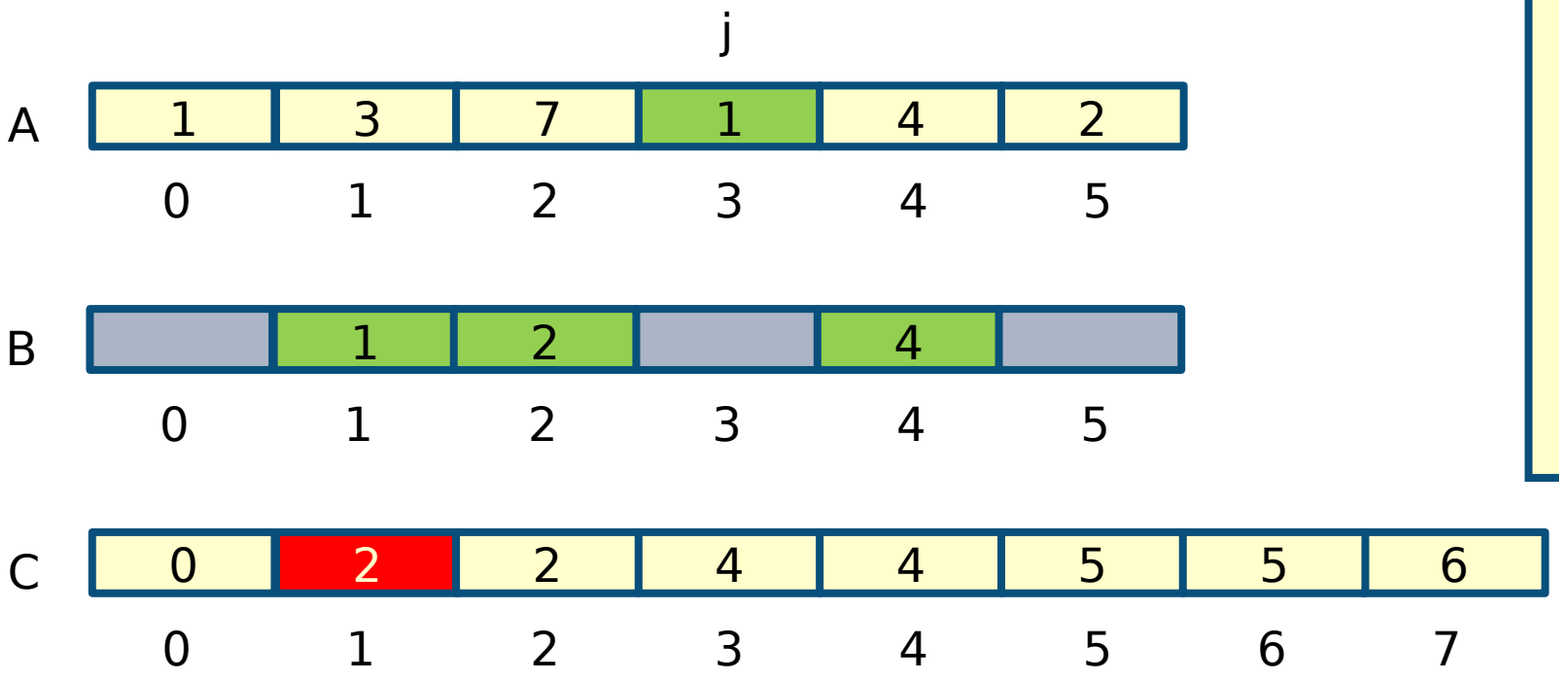



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - The corresponding count is decreased

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

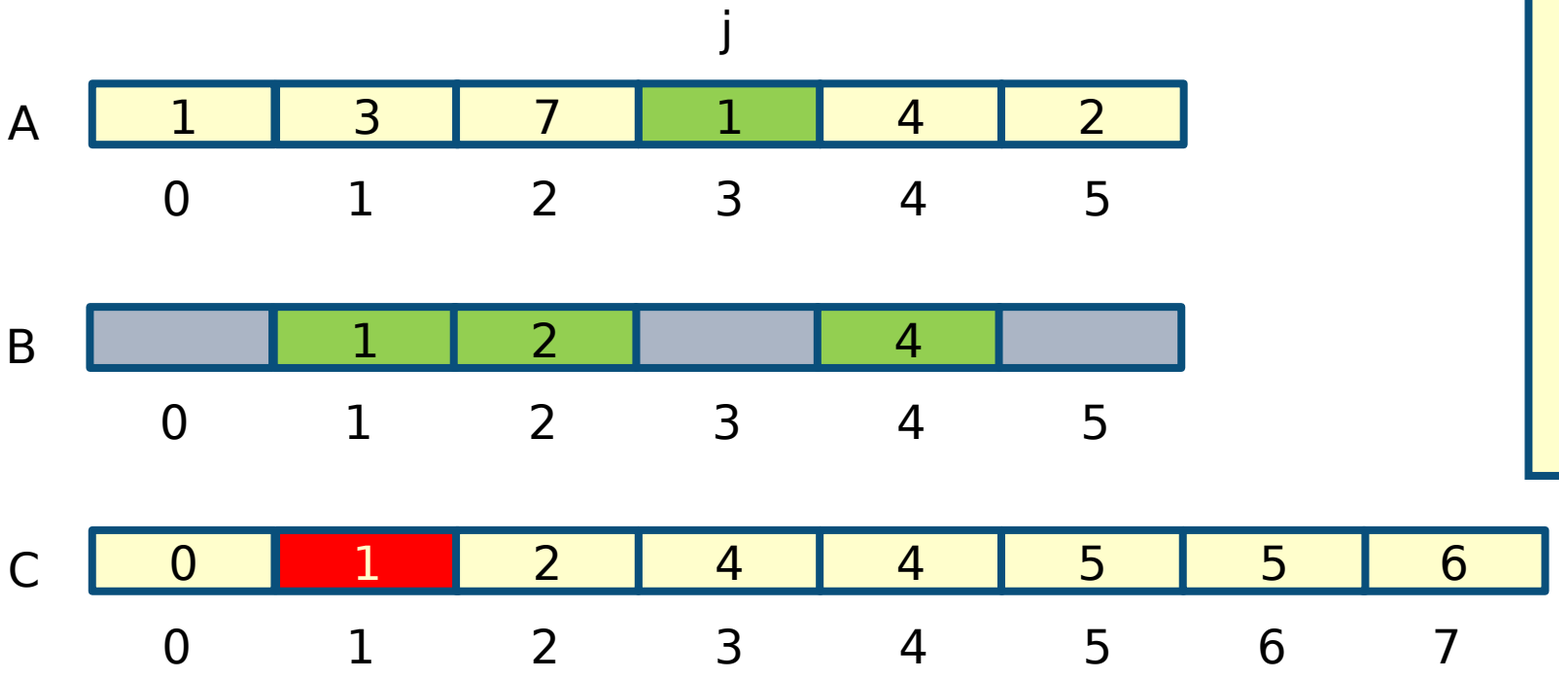



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - 1 is put in its final position C[1] – 1 = 1 in B

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

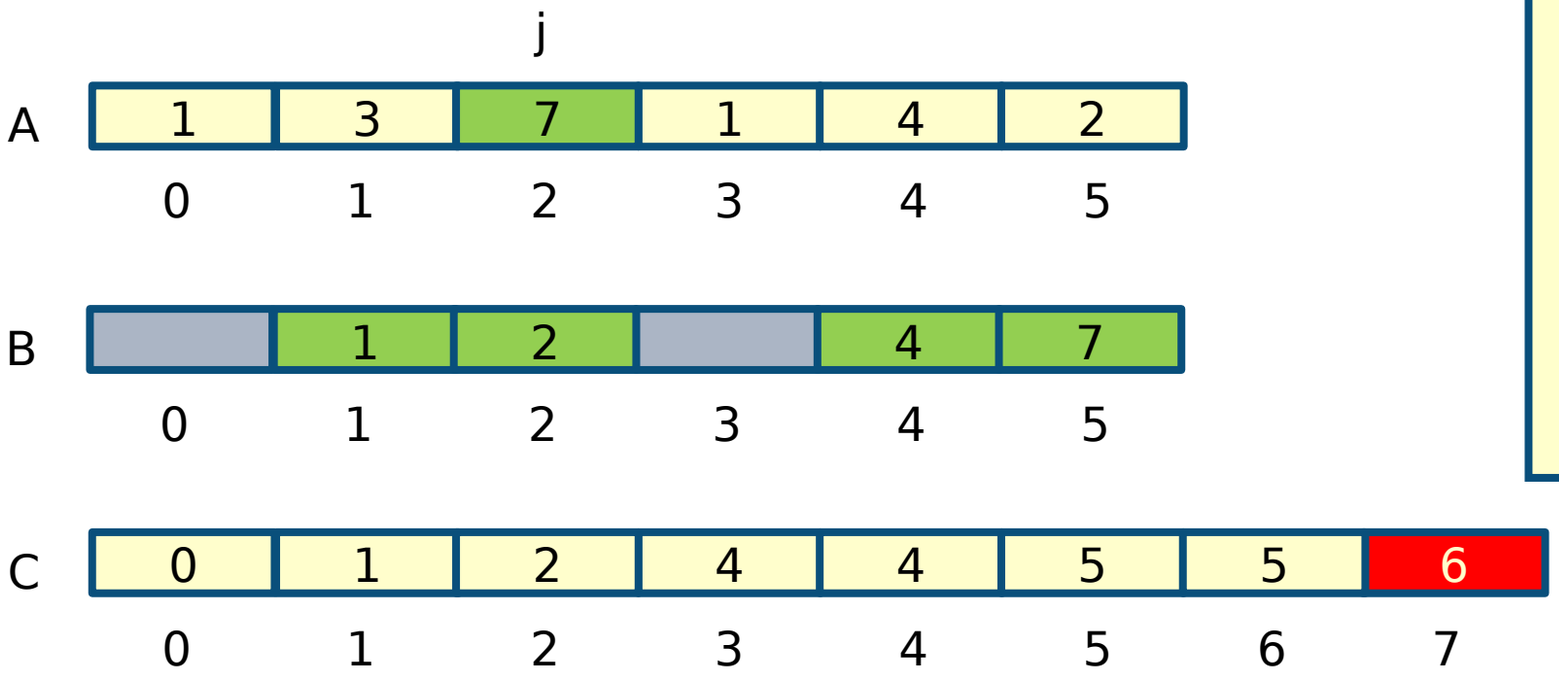



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - The corresponding count is decreased

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

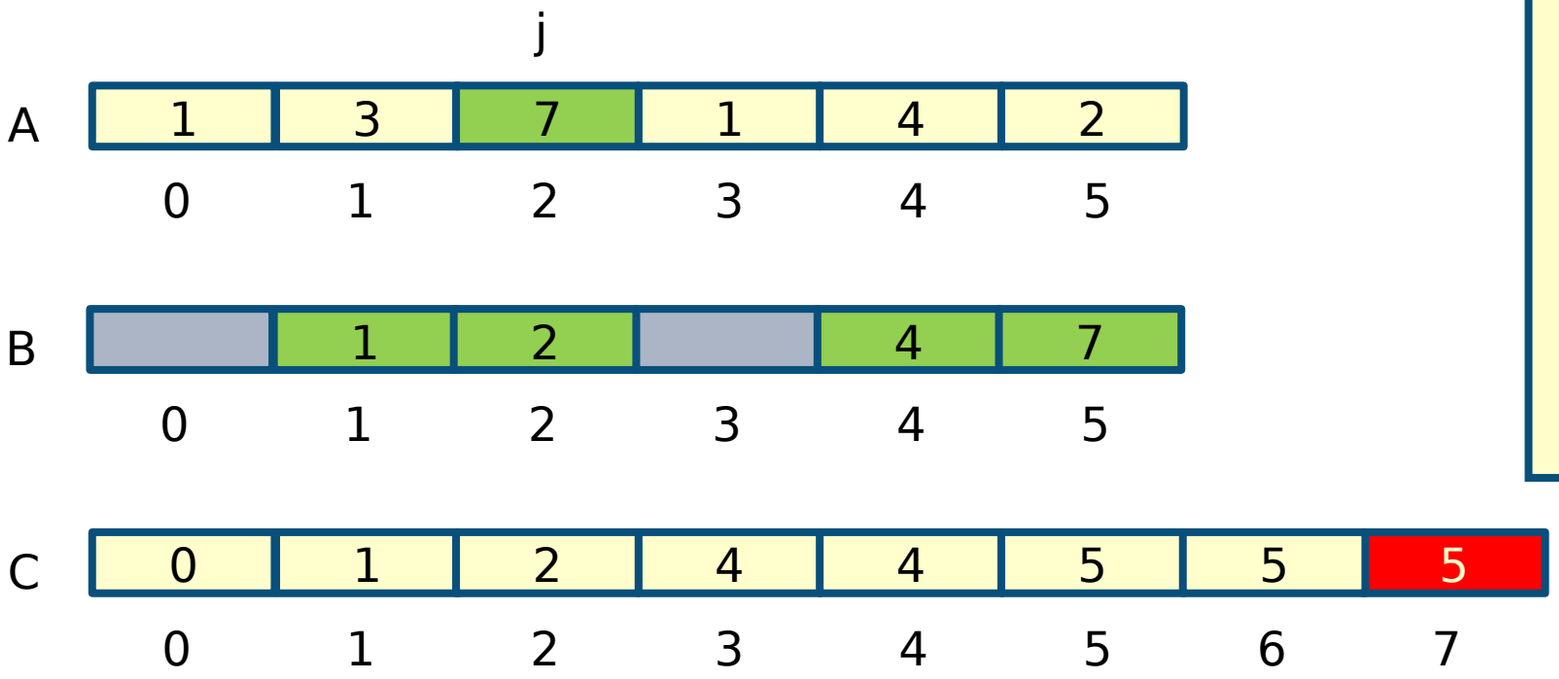



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - 7 is put in its final position C[7] – 1 = 5 in B

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

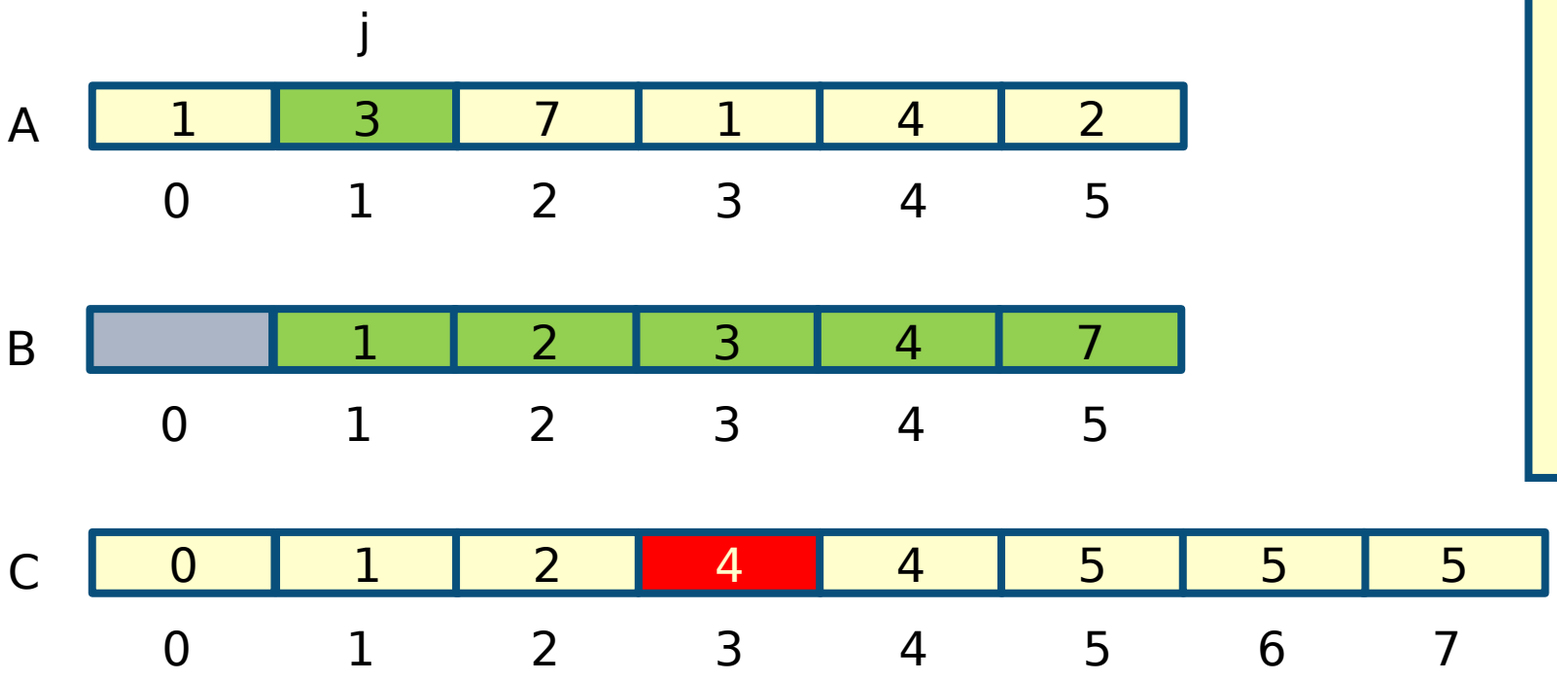



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - The corresponding count is decreased

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

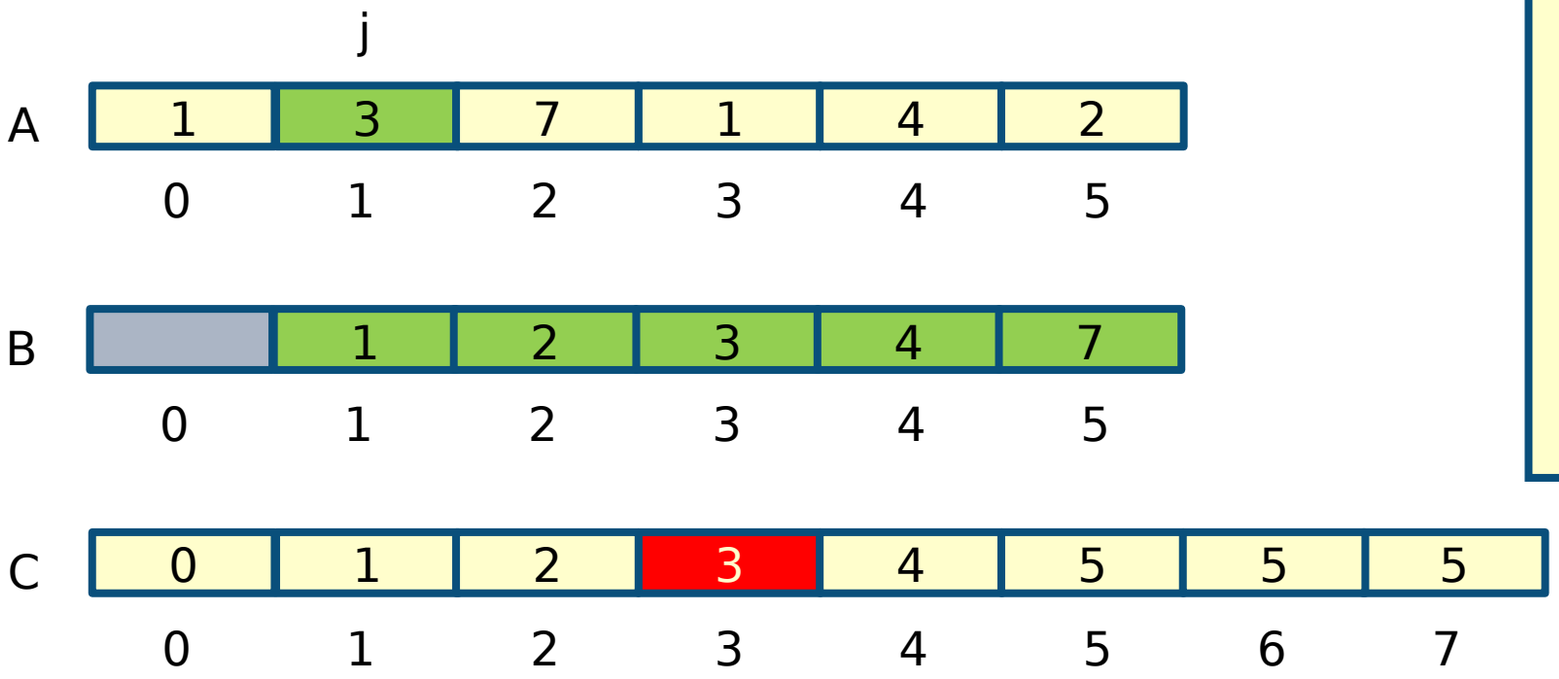



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - 3 is put in its final position C[3] – 1 = 3 in B

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

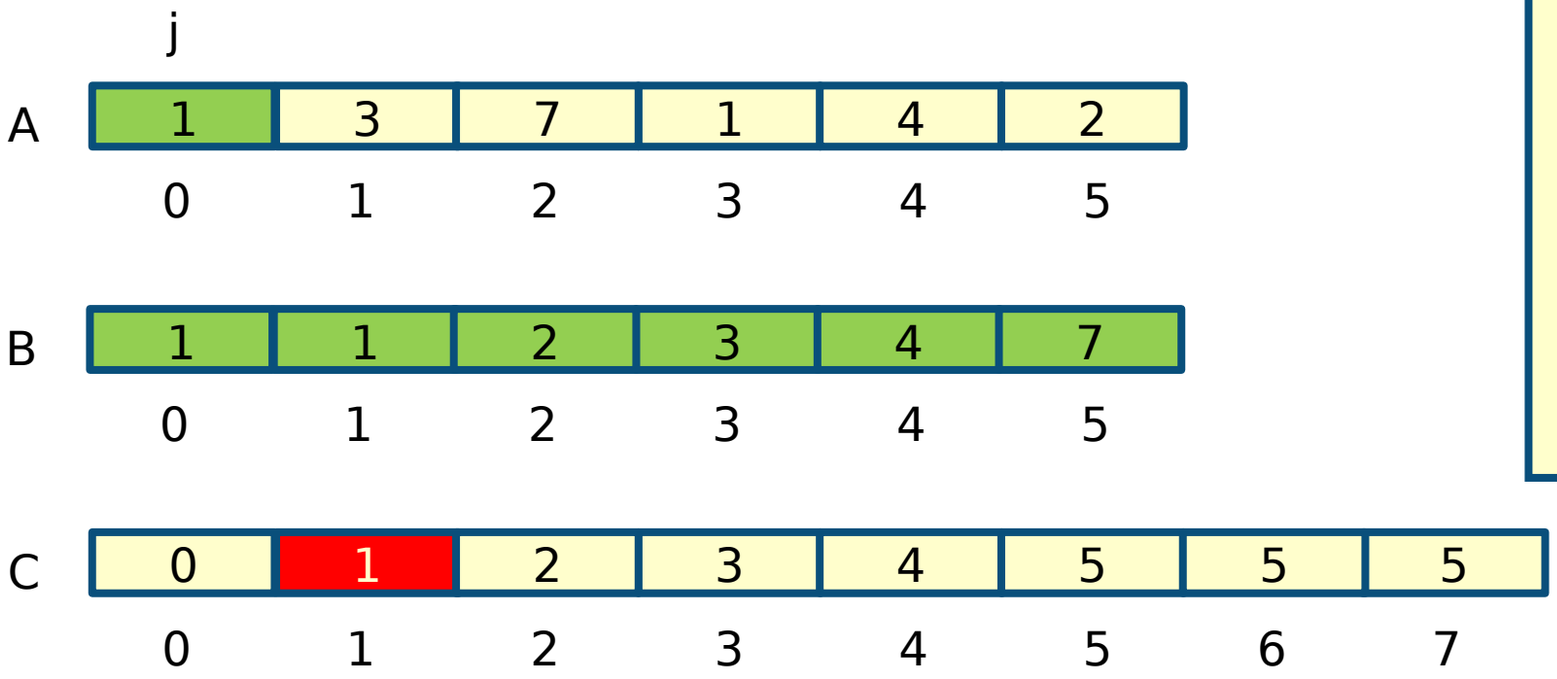



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - The corresponding count is decreased

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**

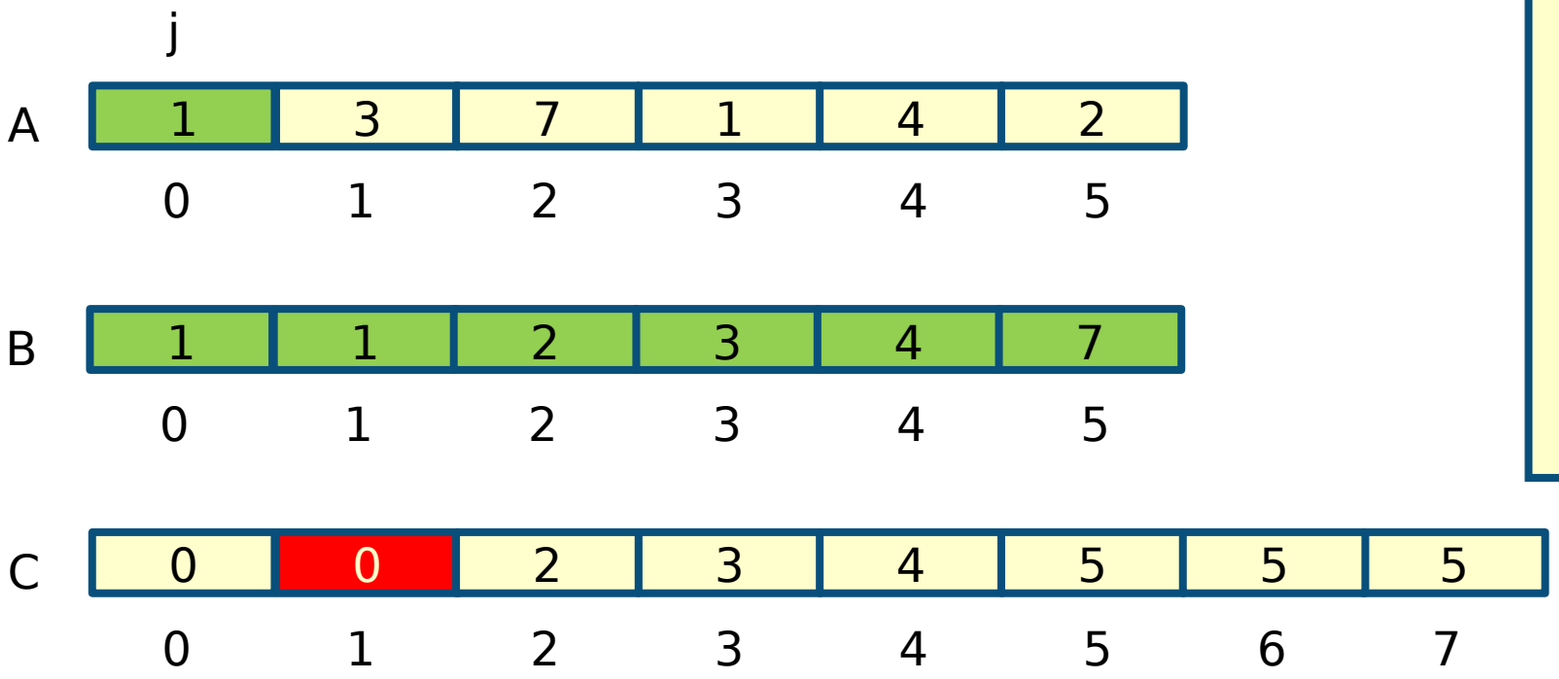



```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

- 1 is put in its final position C[1] – 1 = 0 in B
- The position is correct as we decreased C[1] when we processed the first occurrence of 1

Example execution of COUNTING-SORT(A,B,7)

- **A = [1,3,7,1,4,2] and B has length n = 6**




```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]] - 1] := A[j]
    C[A[j]] := C[A[j]] - 1
```

  - The corresponding count is decreased
  - Termination

Running time

- **Analysing each block of code we have that**
  - Allocation and initialisation of C is O(k)
  - First for loop is O(n)
  - Second for loop is O(k)
  - Third for loop is O(n)
- **T(n) = O(n + k)**

- **In practice COUNTING-SORT is used when k = O(n)**
  - Therefore, T(n) = O(n)

```
COUNTING-SORT(A,B,k)
  new C[0..k]
  for j = 0 to n - 1
    C[A[j]] := C[A[j]] + 1
  for i = 1 to k
    C[i] := C[i] + C[i-1]
  for j = n-1 downto 0
    B[C[A[j]]] := A[j]
    C[A[j]] := C[A[j]] - 1
```

RADIX-SORT

- **Assume that each element in the array A[0..n-1] has d digits**
  - Digit 1 is the lowest-order digit
  - Digit d is the highest-order digit

```
RADIX-SORT(A,d)
  for i = 1 to d
    stable sort A on digit i
```

- **Running time when COUNTING-SORT is used as subroutine**
  - d iterations
  - Complexity of COUNTING-SORT is O(n + k)
- **T(n) = O(d(n + k))**
  - When d is a constant and k = O(n), T(n) = O(n)

Example execution of RADIX-SORT(A,3)

- **Input array A of 6 3-digit integers**
- **k = 9**

```
RADIX-SORT(A,d)
  for i = 1 to d
    stable sort A on digit d
```

A

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

Example execution of RADIX-SORT(A,3)

- **Input array A of 6 3-digit integers**
- **k = 9**

```
RADIX-SORT(A,d)
  for i = 1 to d
    stable sort A on digit d
```

A

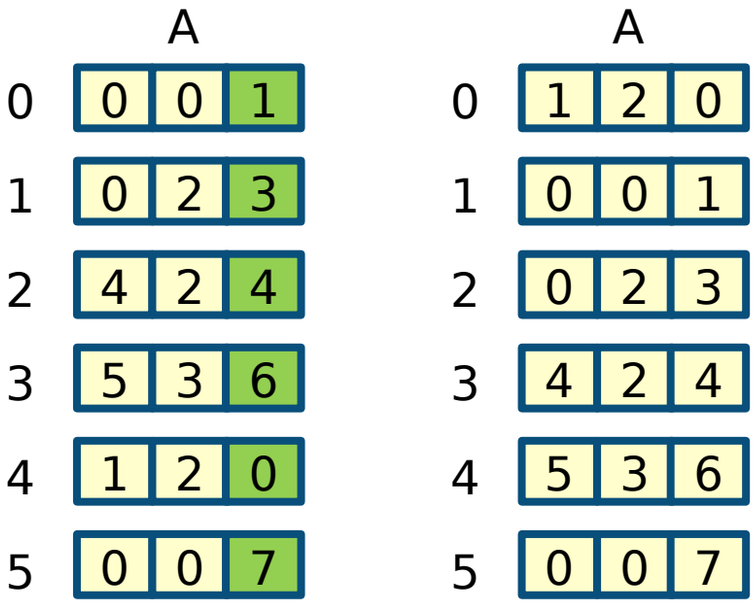
0	0	0	1
1	0	2	3
2	4	2	4
3	5	3	6
4	1	2	0
5	0	0	7

- Sort on digit i = 1

Example execution of RADIX-SORT(A,3)

- **Input array A of 6 3-digit integers**
- **k = 9**

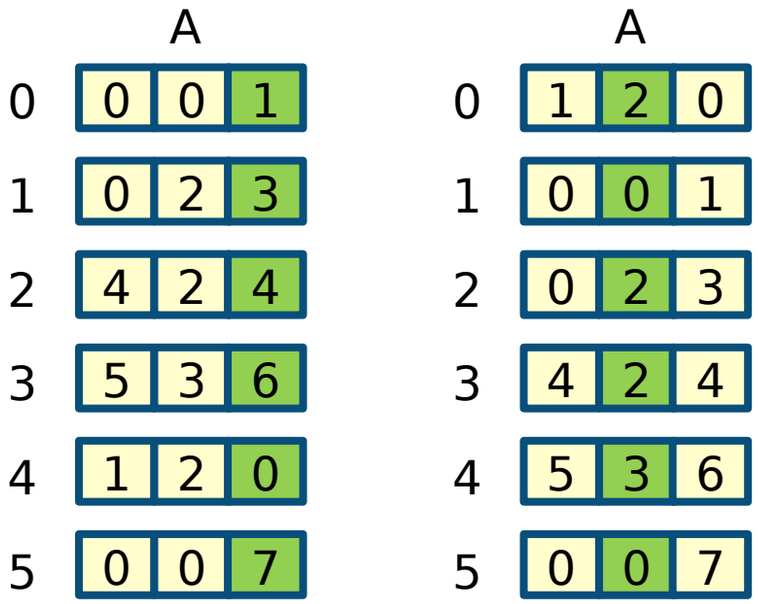
```
RADIX-SORT(A,d)
  for i = 1 to d
    stable sort A on digit d
```



Example execution of RADIX-SORT(A,3)

- **Input array A of 6 3-digit integers**
- **k = 9**

```
RADIX-SORT(A,d)
  for i = 1 to d
    stable sort A on digit d
```

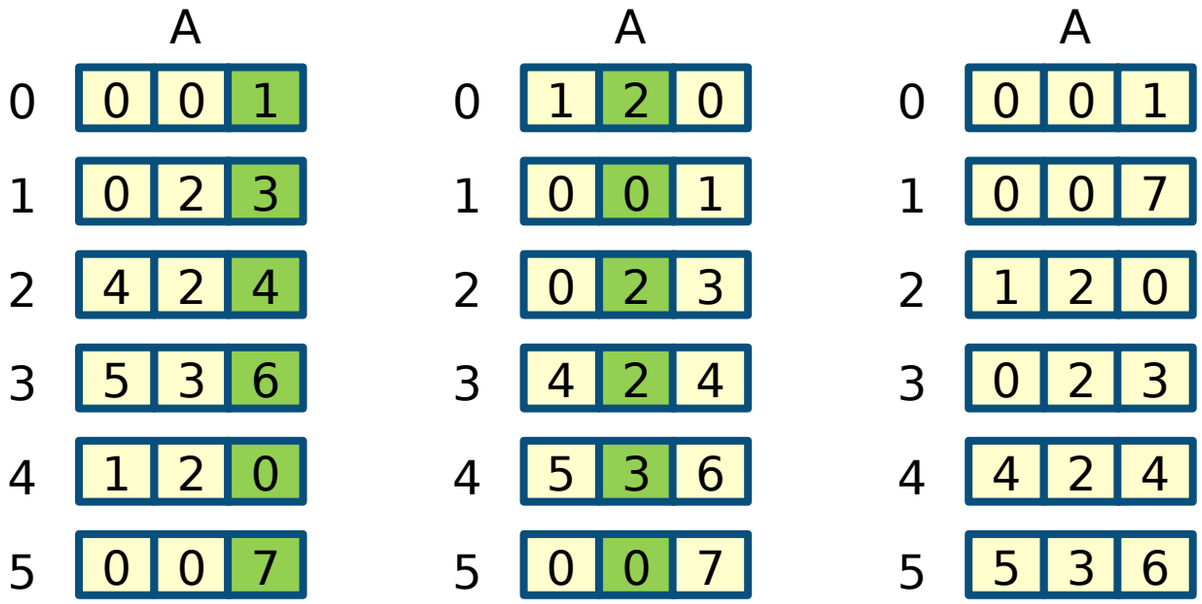


- Sort on digit i = 2

Example execution of RADIX-SORT(A,3)

- **Input array A of 6 3-digit integers**
- **k = 9**

```
RADIX-SORT(A,d)
  for i = 1 to d
    stable sort A on digit d
```

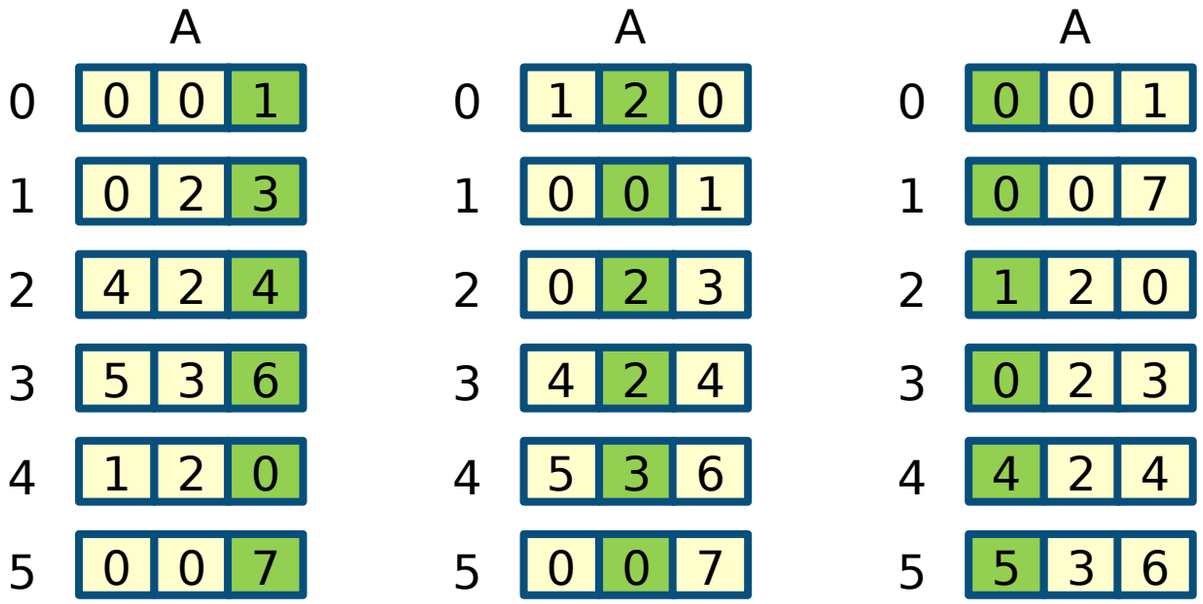


- Stable sorting (see relative order of 001 and 007)

Example execution of RADIX-SORT(A,3)

- **Input array A of 6 3-digit integers**
- **k = 9**

```
RADIX-SORT(A,d)
  for i = 1 to d
    stable sort A on digit d
```



- Sort on digit i = 3

Example execution of RADIX-SORT(A,3)

- **Input array A of 6 3-digit integers**
- **k = 9**

```
RADIX-SORT(A,d)
  for i = 1 to d
    stable sort A on digit d
```

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

- Termination

Summary

- **COUNTING-SORT**
  - Running time
- **RADIX-SORT**