Сортировки за линейное время

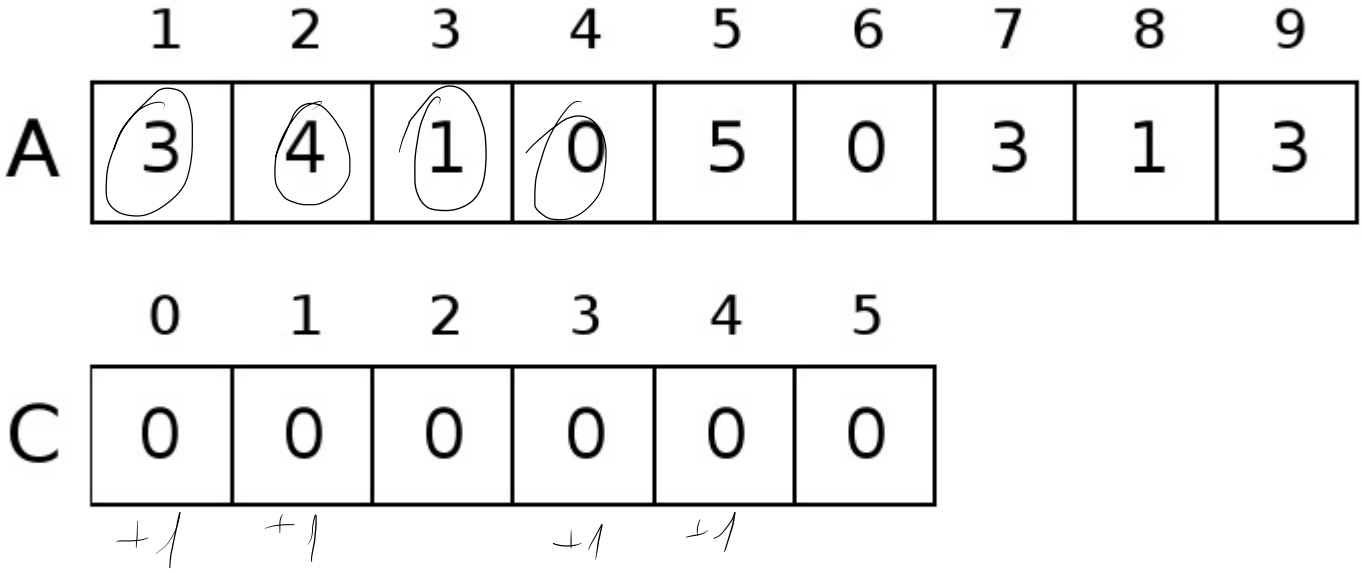
Сортировки за линейное время

Сортировка подсчетом = Counting Sort

Вход: n целых чисел из промежутка от 0 до k

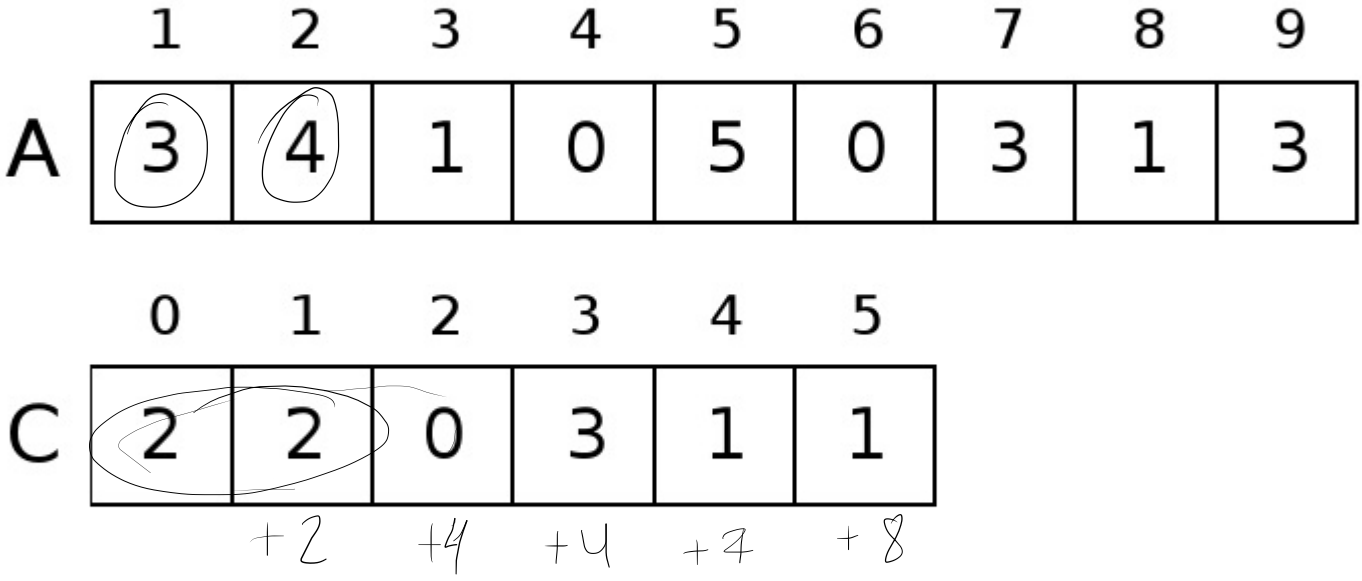
Основная идея: для каждого элемента определить число элементов, которые его меньше

Сортировка подсчетом



1 Пусть $C[0\,..\,k]$ — новый массив
2 **for** $i = 0$ **to** k
3 $\quad C[i] = 0$

Сортировка подсчетом

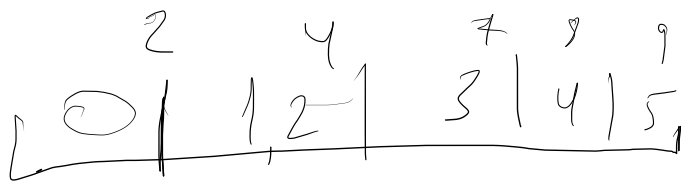


4 **for** $j = 1$ **to** $A.length$
5 $\quad C[A[j]] = C[A[j]] + 1$
6 // Сейчас $C[i]$ содержит количество элементов, равных i.

Сортировка подсчетом

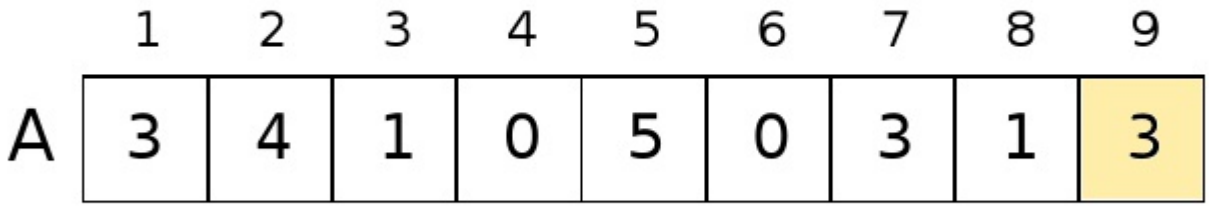
	1	2	3	4	5	6	7	8	9
A	3	4	1	0	5	0	3	1	3

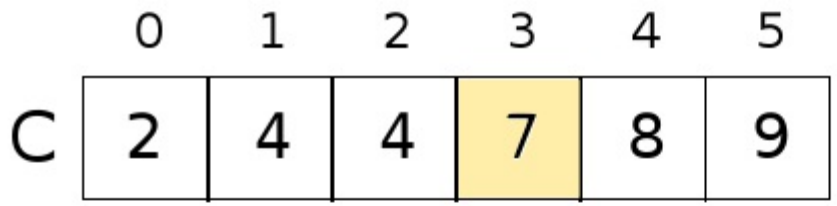
	0	1	2	3	4	5
C	2	4	4	7	8	9



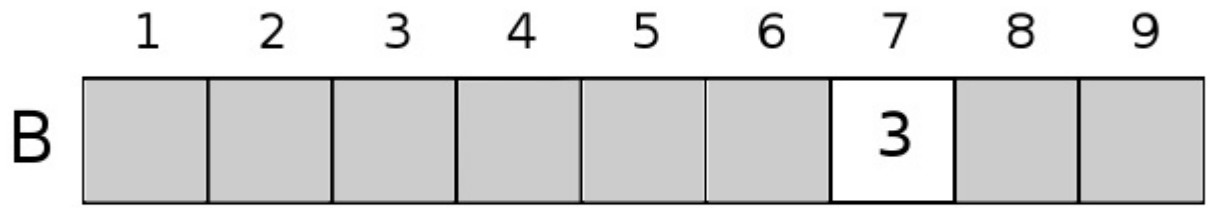
7 **for** $i = 1$ **to** k
8 $\quad C[i] = C[i] + C[i-1]$
9 // Сейчас $C[i]$ содержит количество элементов, не превышающих i.

Сортировка подсчетом

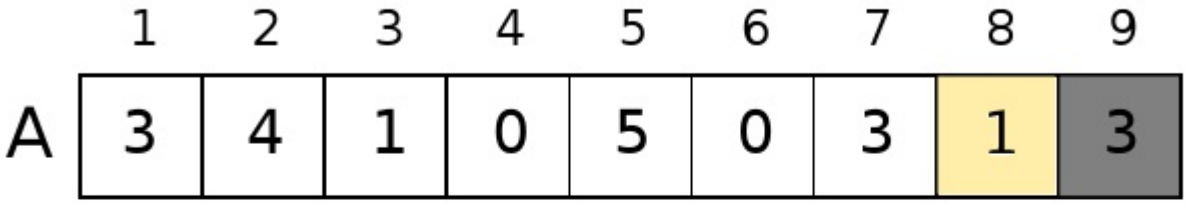


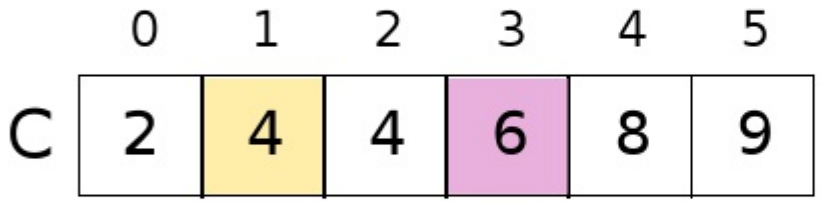



```
10  for j = A.length downto 1
11      B[C[A[j]]] = A[j]
12      C[A[j]] = C[A[j]] − 1
```

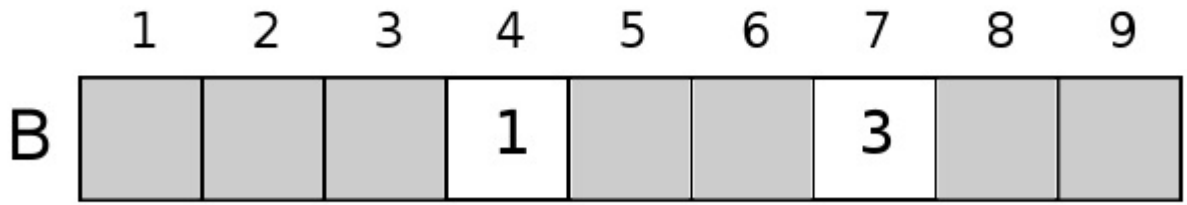


Сортировка подсчетом

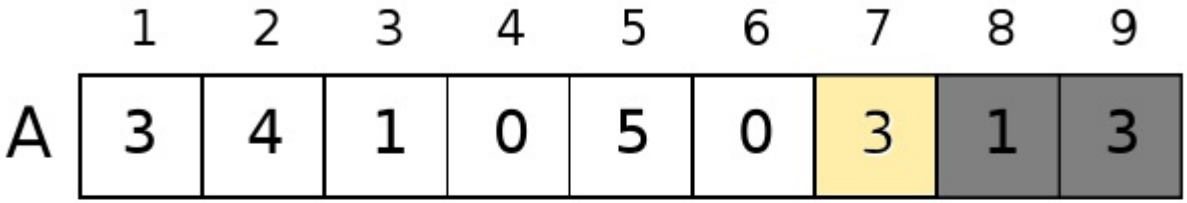


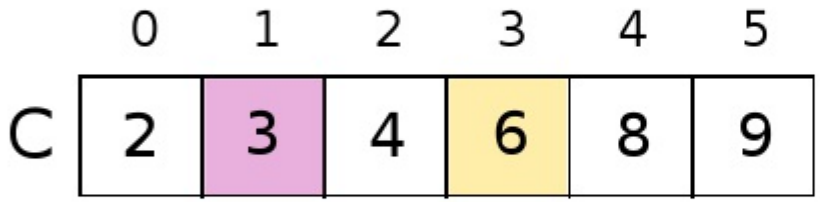



```
10  for j = A.length downto 1
11      B[C[A[j]]] = A[j]
12      C[A[j]] = C[A[j]] − 1
```

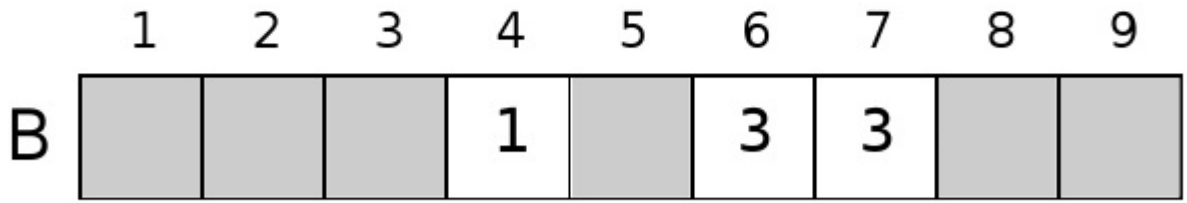


Сортировка подсчетом

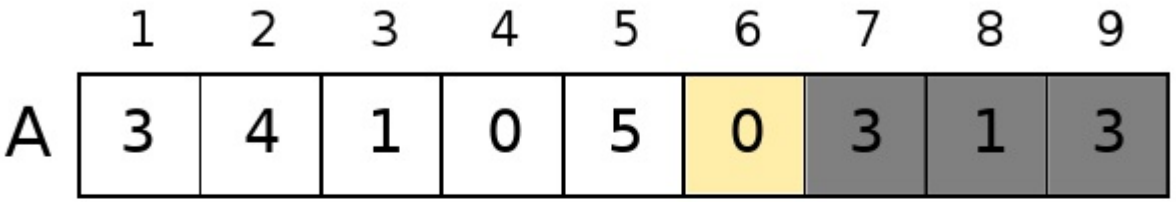


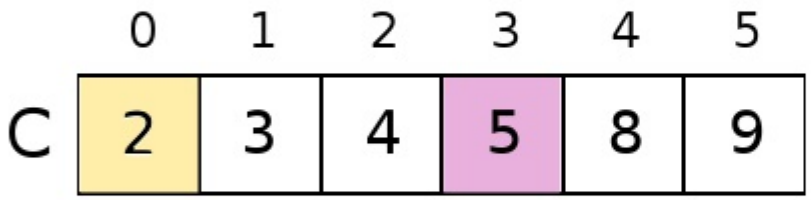



```
10  for j = A.length downto 1
11      B[C[A[j]]] = A[j]
12      C[A[j]] = C[A[j]] − 1
```

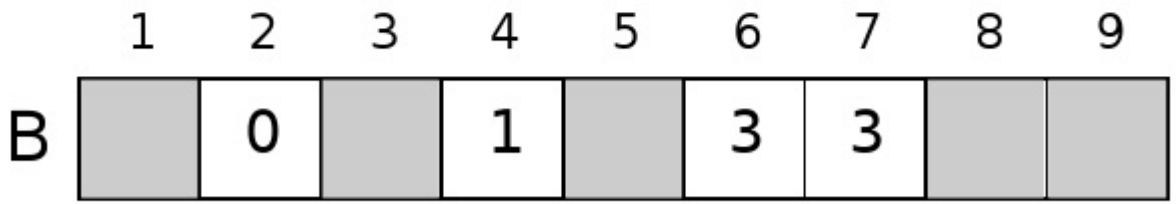


Сортировка подсчетом

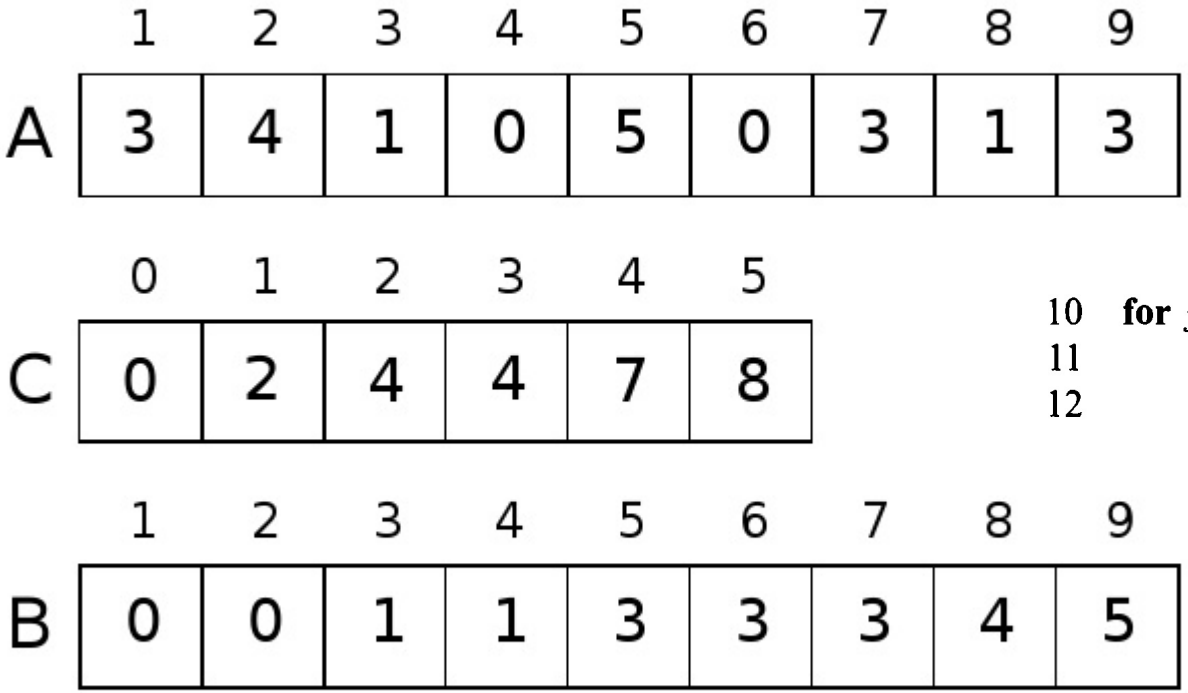





```
10  for j = A.length downto 1
11      B[C[A[j]]] = A[j]
12      C[A[j]] = C[A[j]] − 1
```



Сортировка подсчетом




```
10  for j = A.length downto 1
11      B[C[A[j]]] = A[j]
12      C[A[j]] = C[A[j]] − 1
```

COUNTING-SORT(A, B, k)

```
1  Пусть C[0..k] — новый массив
2  for i = 0 to k
3      C[i] = 0
4  for j = 1 to A.length
5      C[A[j]] = C[A[j]] + 1
6  // Сейчас C[i] содержит количество элементов, равных i.
7  for i = 1 to k
8      C[i] = C[i] + C[i − 1]
9  // Сейчас C[i] содержит количество элементов, не превышающих i.
10 for j = A.length downto 1
11     B[C[A[j]]] = A[j]
12     C[A[j]] = C[A[j]] − 1
```

$\Theta(k)$

$\Theta(n)$

$\Theta(n+k)$

$\Theta(n)$

$\Theta(n)$

Устойчив! - элементы с одинаковым значением находятся в выходном массиве в том же относительном порядке, что и во входном

Сортировки за линейное время

Поразрядная сортировка = Radix Sort

Вход: строки одинаковой длины

Основная идея: сортировать от младшего разряда к старшему с помощью устойчивой сортировки

Цифровая сортировка

```
RADIX-SORT(A, d)
1  for i = 1 to d
2      Выполнить устойчивую сортировку массива A по цифре i
```

BAR
BOX
BIG
ROW
EAR
RUG
COW
TAR
TAN

Цифровая сортировка

RADIX-SORT(A, d)

1 **for** $i = 1$ **to** d

2 Выполнить устойчивую сортировку массива A по цифре i

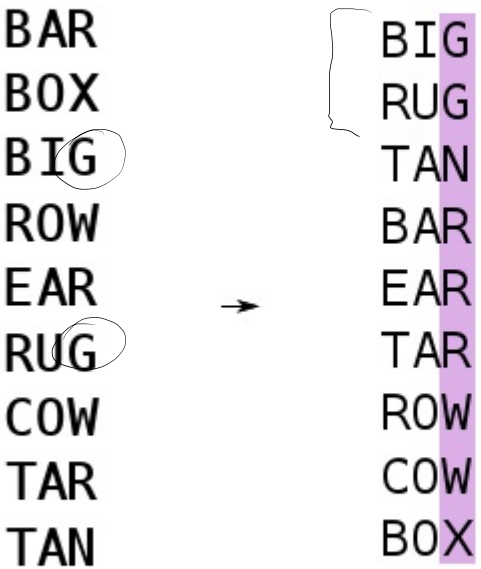
BAR
BOX
BIG
ROW
EAR
RUG
COW
TAR
TAN

Цифровая сортировка

RADIX-SORT(A, d)

1 **for** $i = 1$ **to** d

2 Выполнить устойчивую сортировку массива A по цифре i

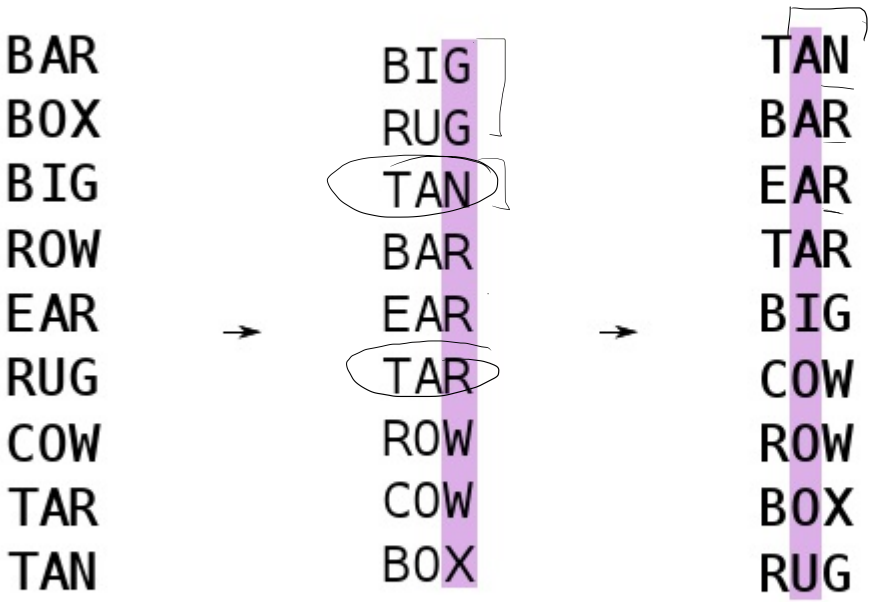


Цифровая сортировка

RADIX-SORT(A, d)

1 **for** $i = 1$ **to** d

2 Выполнить устойчивую сортировку массива A по цифре i



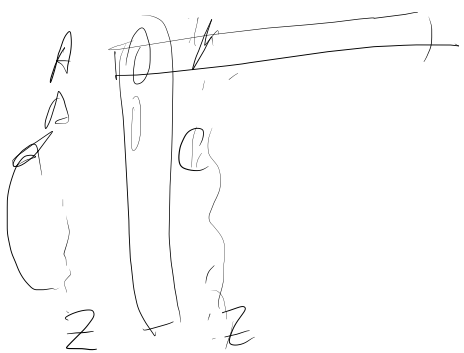
Цифровая сортировка

RADIX-SORT(A, d)

1 **for** $i = 1$ **to** d

2 Выполнить устойчивую сортировку массива A по цифре i

BAR	BIG	TAN	BAR	
BOX	RUG	BAR	BIG	
BIG	TAN	EAR	BOX	
ROW	BAR	TAR	COW	
EAR	EAR	BIG	EAR	
RUG	TAR	COW	ROW	
COW	ROW	ROW	RUG	
TAR	COW	BOX	TAN	
TAN	BOX	RUG	TAR	



Название	Время	Доп. память	Ограничения	Устойч.
Выбором, вставкой, пузырьком				
Слиянием				
Быстрая				
Подсчетом				
Поразрядная				
Карманная				
Кучей (итеративная)				
Кучей (рекурсивная)				