ДАА – практикум

(Дизайн и анализ на алгоритми - практикум)

Изготвил: Иван Камбуров
ivankamburov96@gmail.com

Сложности на алгоритми

Сложности на алгоритми

- *Big-O нотация.*
  - $O\big(g(n)\big) = \{f(n) | \exists c > 0\ \exists n_0 > 0\colon\ \forall n \geq n_0\colon\ 0 \leq f(n) \leq c.g(n)\}$
  - *Кратка табличка*
- *Сложност по време.*
  - *Кратка табличка + примери*
- *Сложност по памет.*
- *Cheat sheet*

Задача:

При дадена редица $A_1, A_2, \ldots, A_N$
$(1 \leq A_i \leq 100\,000\,000)$,
сортирана в нарастващ ред,
колко двойки има сред тях,
чиято сума е равна на X?

Задача: При дадена редица
$A_1, A_2, \ldots, A_N$ $(1 \le A_i \le 100\,000\,000)$,
сортирана в нарастващ ред,
колко двойки има сред тях, чиято сума е равна на X?

Задача: При дадена редица
$A_1, A_2, \ldots, A_N$ $(1 \le A_i \le 100\,000\,000)$,
сортирана в нарастващ ред,
колко двойки има сред тях, чиято сума е равна на *X*?

Задача: При дадена редица
$A_1, A_2, \ldots, A_N$ $(1 \le A_i \le 100\,000\,000)$,
сортирана в нарастващ ред,
колко двойки има сред тях, чиято сума е равна на X?

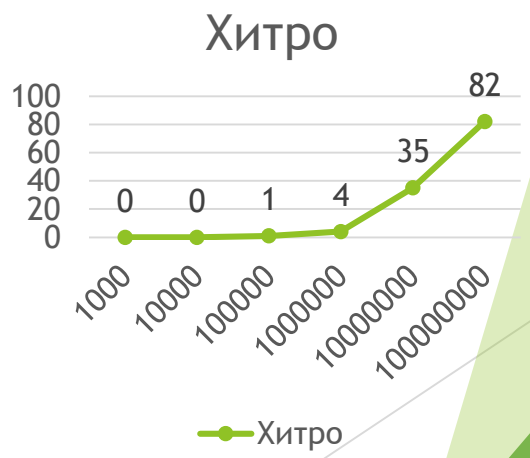
Хитро решение

Задача: При дадена редица
$A_1, A_2, \ldots, A_N$ $(1 \le A_i \le 100\,000\,000)$,
сортирана в нарастващ ред,
колко двойки има сред тях, чиято сума е равна на X?

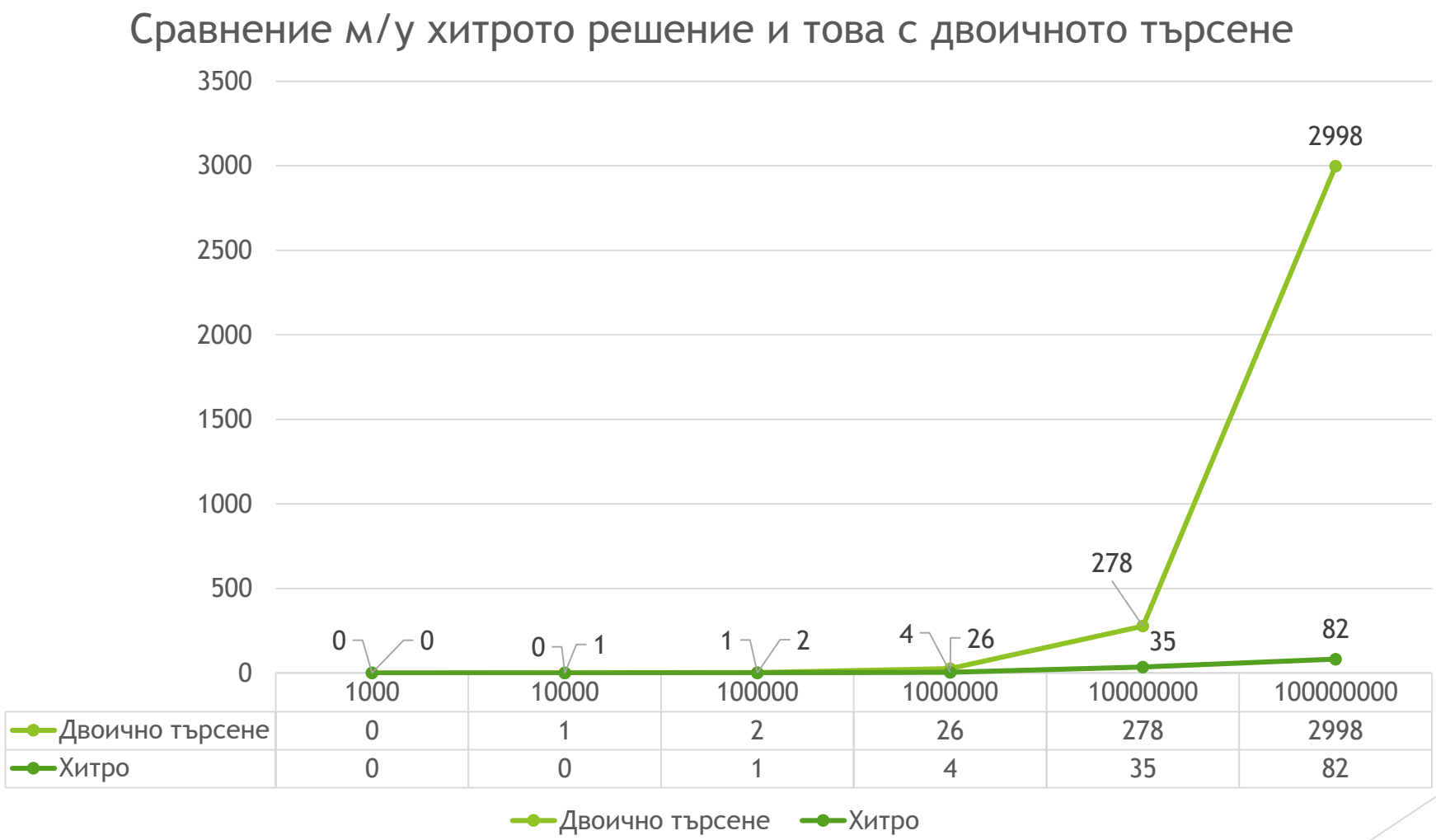
Наивно решение

Двоично търсене

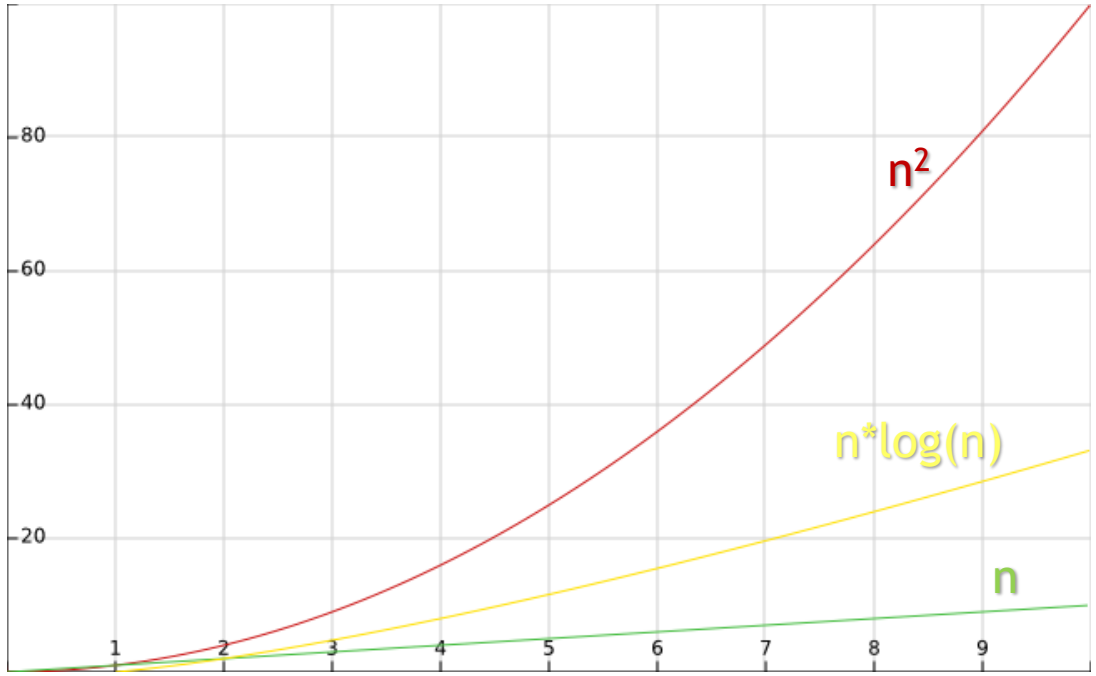
Хитро решение



Малко сравнение



По-общо сравнение



	1000	10000	100000	1000000	10000000	100000000
Наивно	0.000s	0.024s	2.376s	237.700s	6.6h	27d*
Двоично	0.000s	0.001s	0.002s	0.026s	0.278s	2.998s
Хитро	0.000s	0.000s	0.001s	0.004s	0.035s	0.082s

Сложности на алгоритми

- *Big-O нотация.*
  - $O\big(g(n)\big) = \{f(n) | \exists c > 0\ \exists n_0 > 0\colon\ \forall n \geq n_0\colon\ 0 \leq f(n) \leq c.g(n)\}$
  - *Кратка табличка*
- *Сложност по време.*
  - *Кратка табличка + примери*
- *Сложност по памет.*
- *Cheat sheet*

Алгоритми за сортиране

Бавни сортировки - $O(n^2)$

Бавни сортировки:

Бавни сортировки:

- Метод на мехурчето (Bubble Sort);

Бавни сортировки:

- Метод на мехурчето (Bubble Sort);
- Метод на пряката селекция (Selection Sort);

Бавни сортировки:

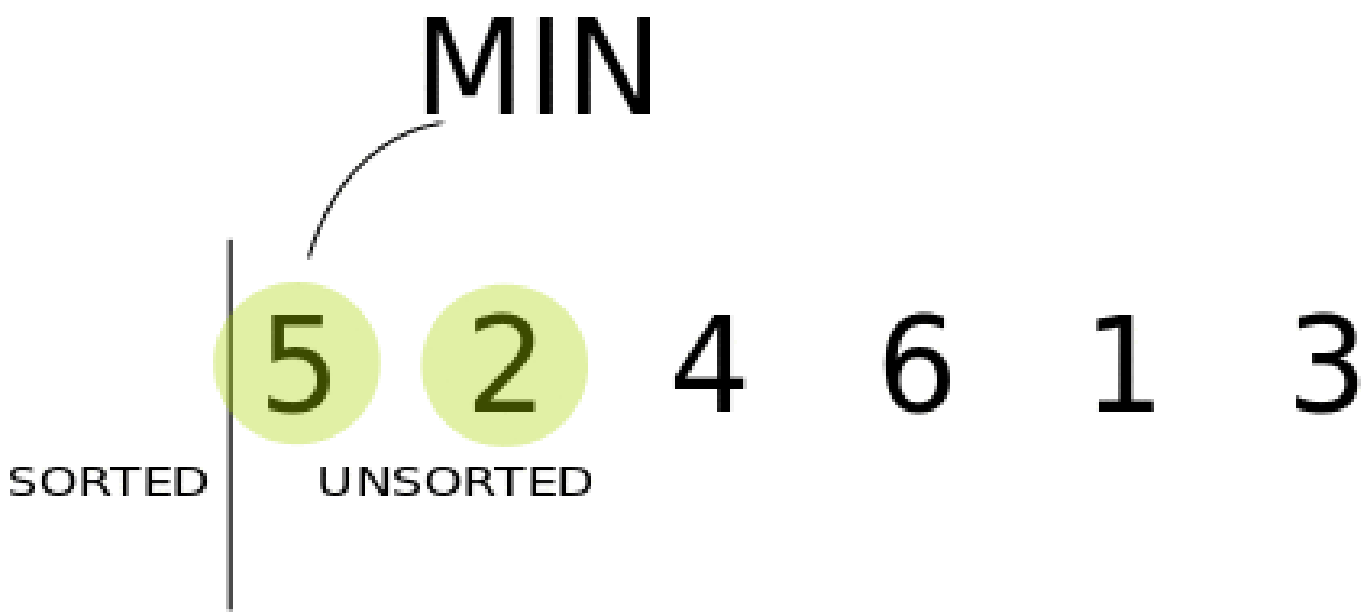
- Метод на мехурчето (Bubble Sort);
- Метод на пряката селекция (Selection Sort);
- Сортиране чрез вмъкване (Insertion Sort);

Бавни сортировки:

- Метод на мехурчето (Bubble Sort);
- Метод на пряката селекция (Selection Sort);
- Сортиране чрез вмъкване (Insertion Sort);
- Gnome Sort (Stupid Sort);

Метод на мехурчето (Bubble Sort, Sinking Sort)

Метод на пряката селекция (Selection Sort)



Сортиране чрез вмъкване (Insertion Sort)

Gnome Sort (Stupid Sort)

```
gnomeSort(a[]):
    pos := 0
    while pos < length(a):
        if (pos == 0 or a[pos] >= a[pos-1]):
            pos := pos + 1
        else:
            swap a[pos] and a[pos-1]
            pos := pos - 1
```

Counting sort

(метод за сортиране, който не се базира на сравнения)

Counting Sort

Input:

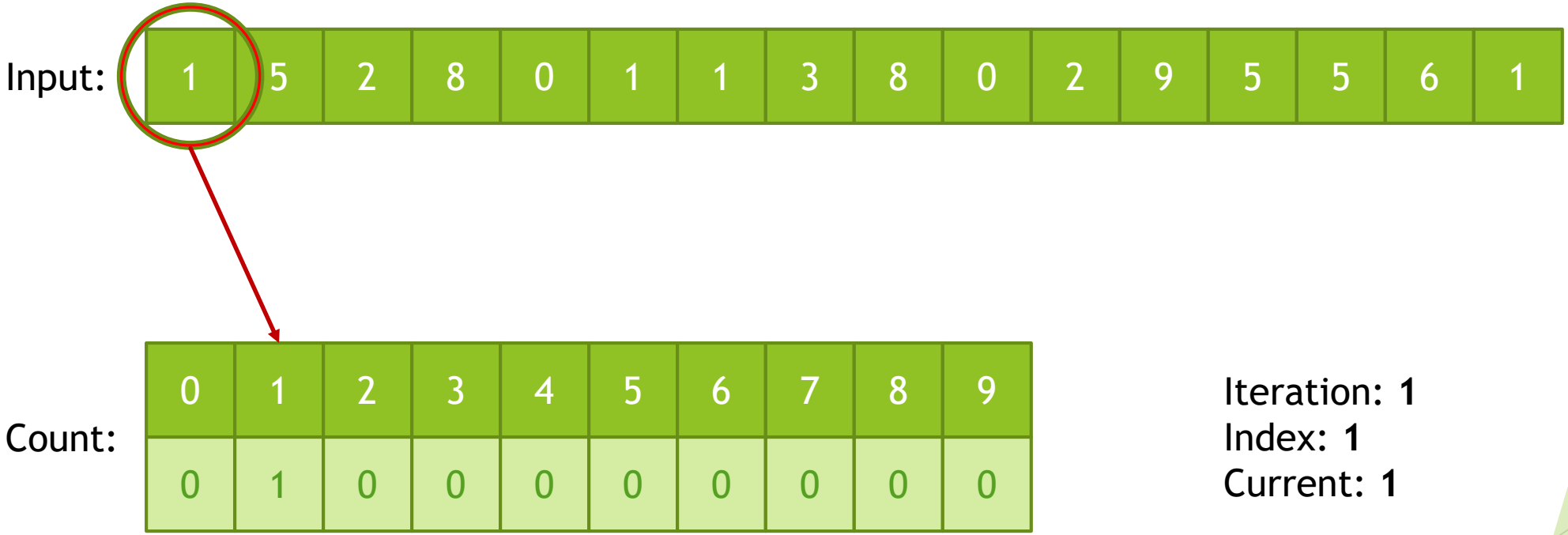
1	5	2	8	0	1	1	3	8	0	2	9	5	5	6	1

Count:

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

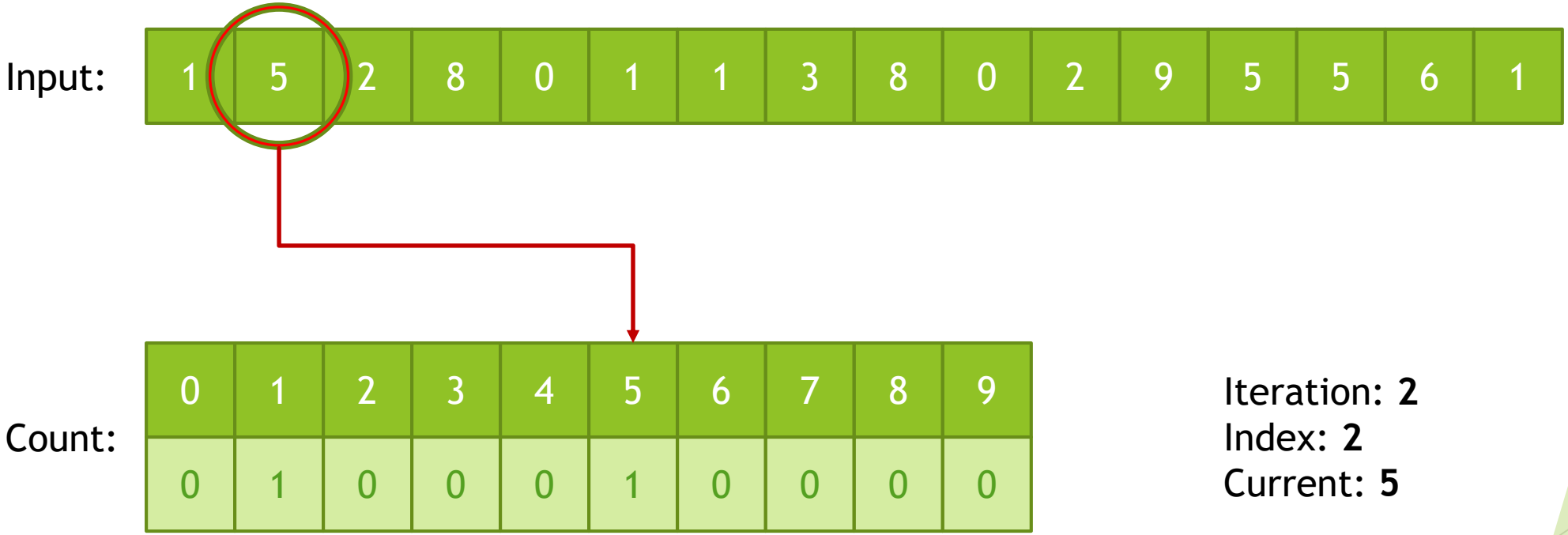
Iteration: **0**

Counting Sort



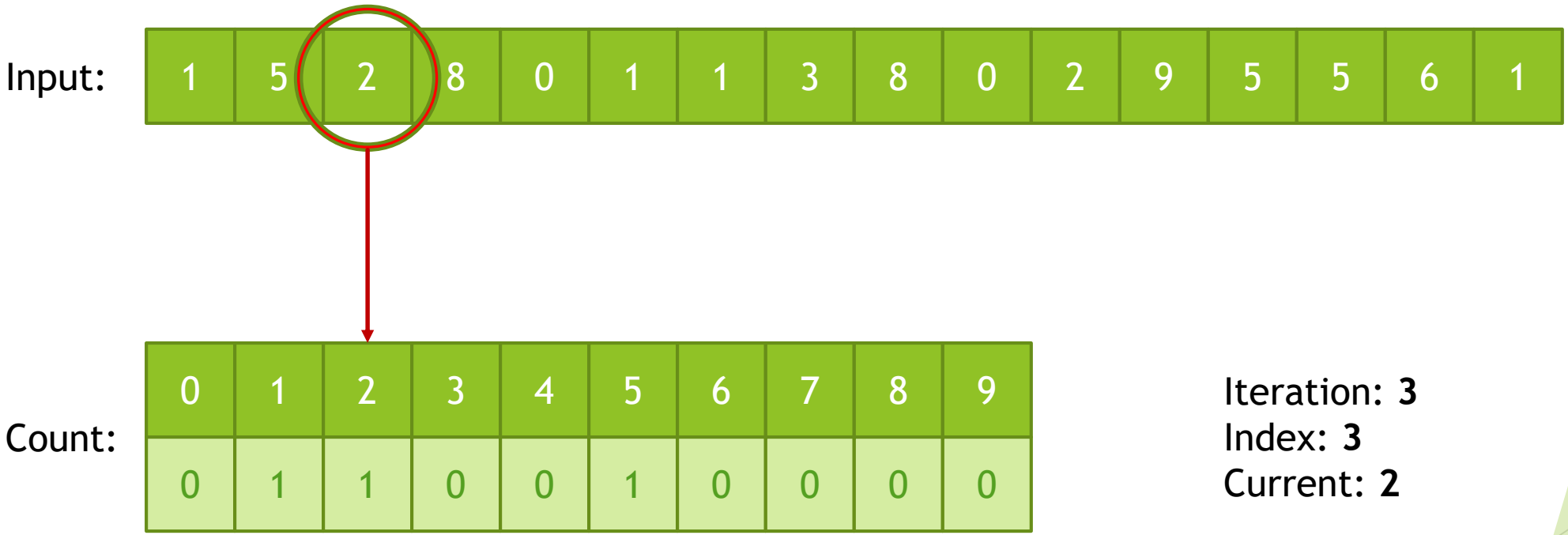
Iteration: **1**
Index: **1**
Current: **1**

Counting Sort



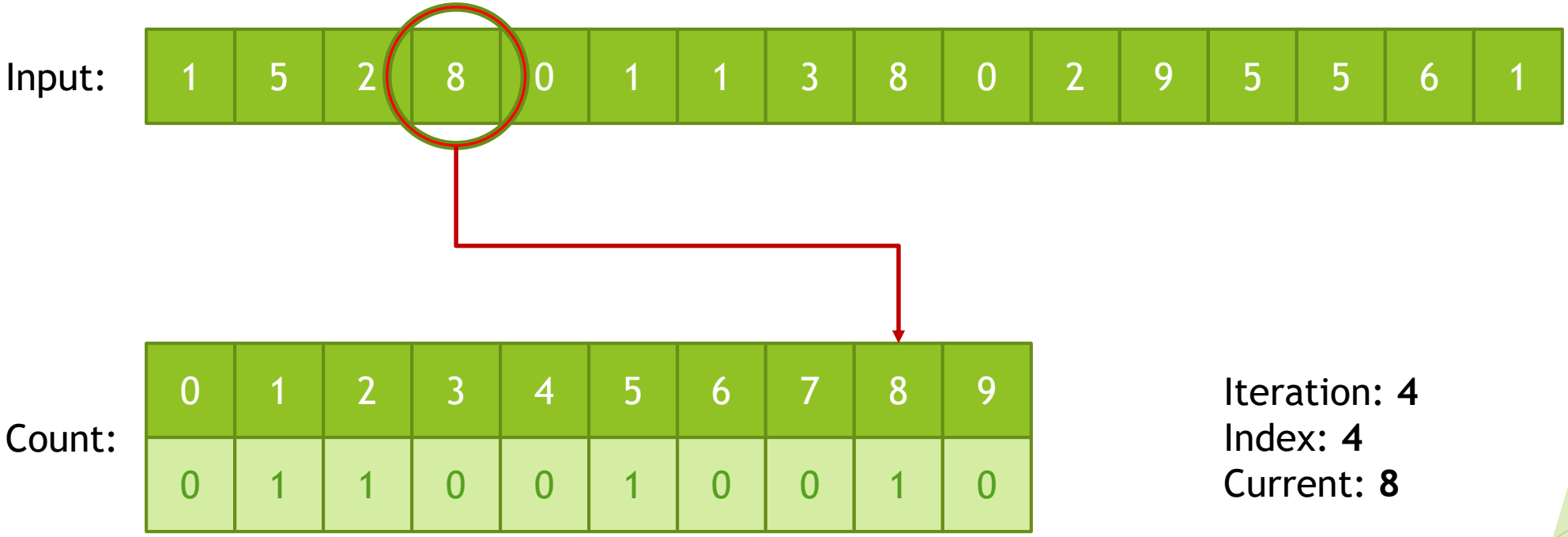
Iteration: **2**
Index: **2**
Current: **5**

Counting Sort



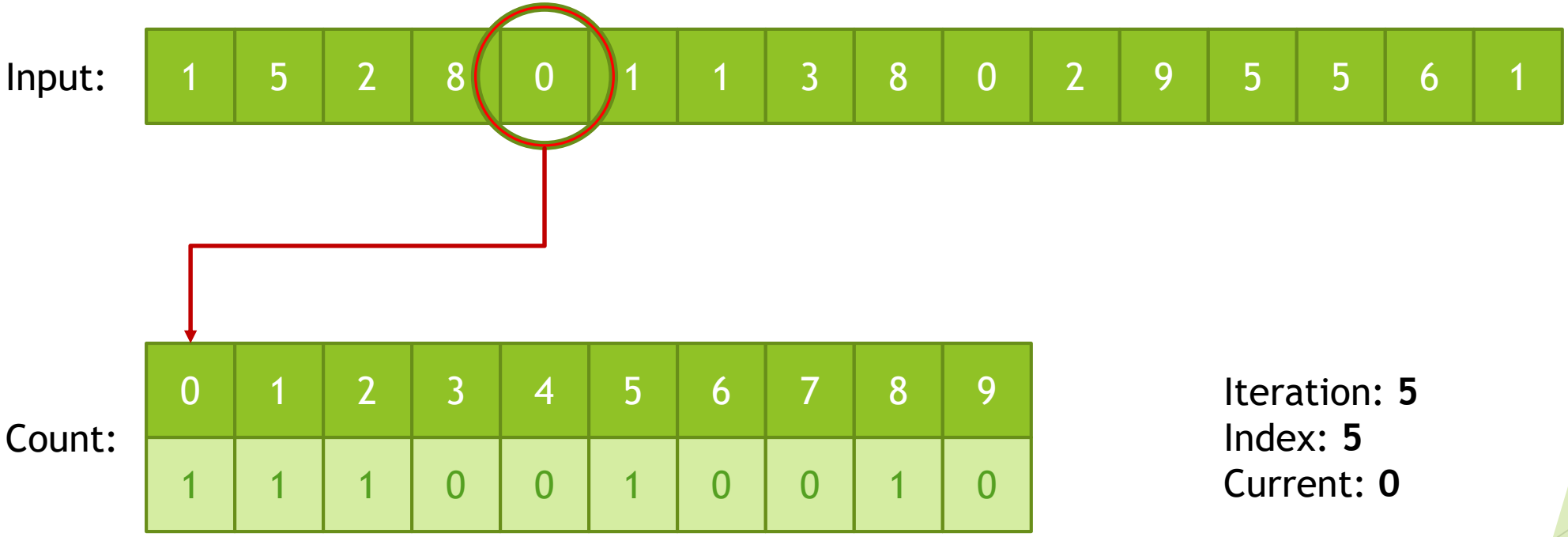
Iteration: **3**
Index: **3**
Current: **2**

Counting Sort



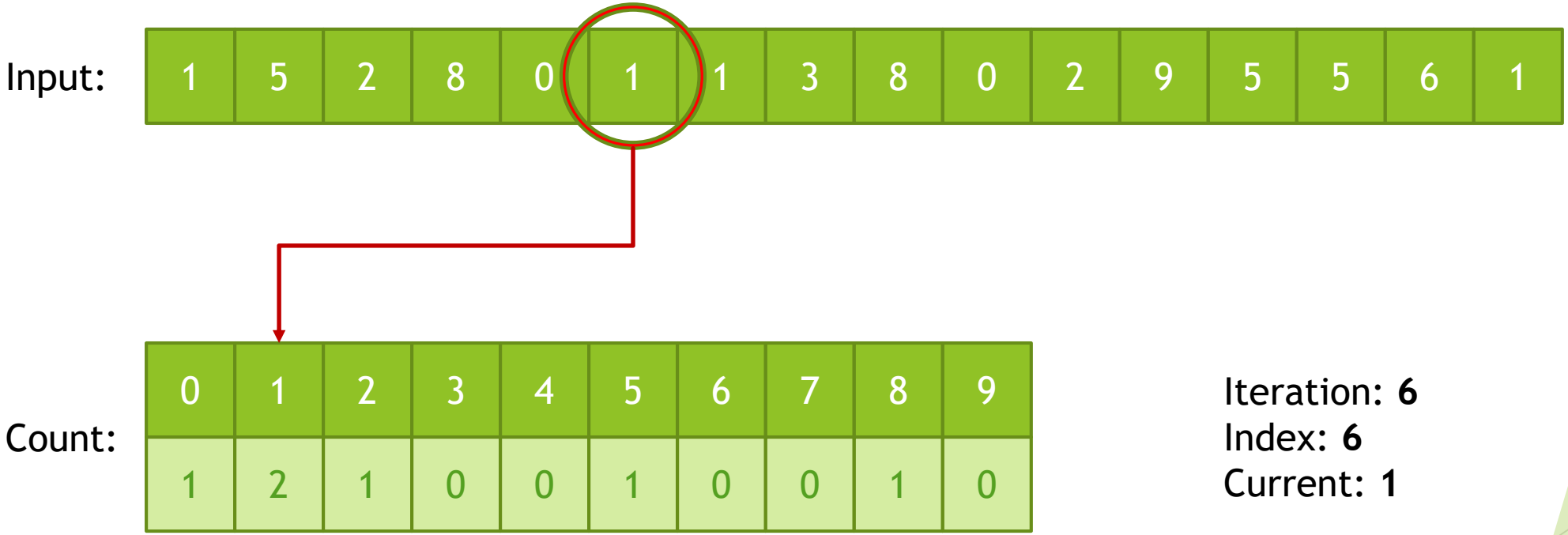
Iteration: **4**
Index: **4**
Current: **8**

Counting Sort



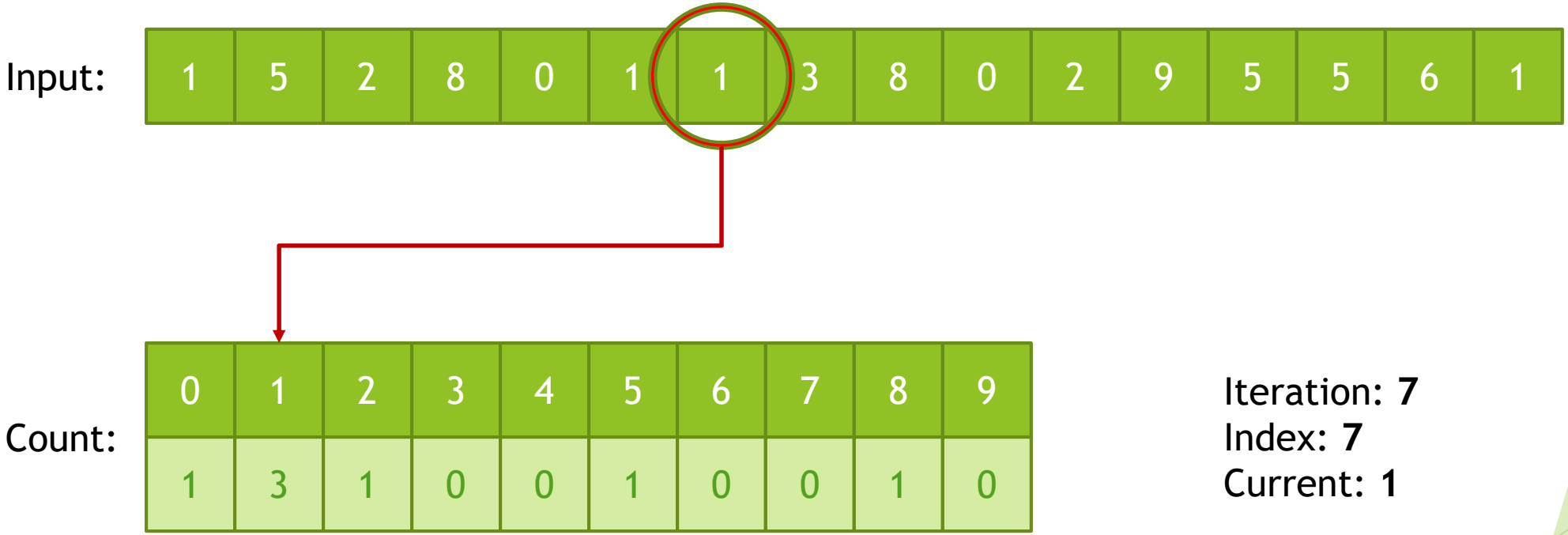
Iteration: **5**
Index: **5**
Current: **0**

Counting Sort



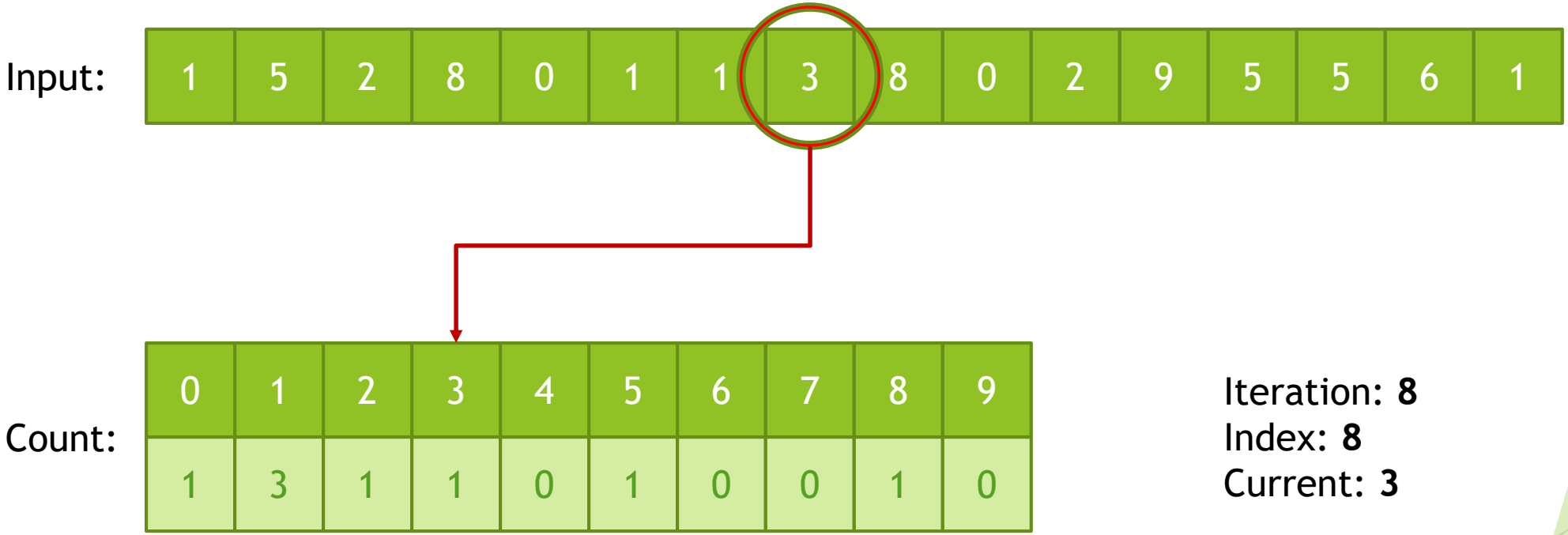
Iteration: **6**
Index: **6**
Current: **1**

Counting Sort



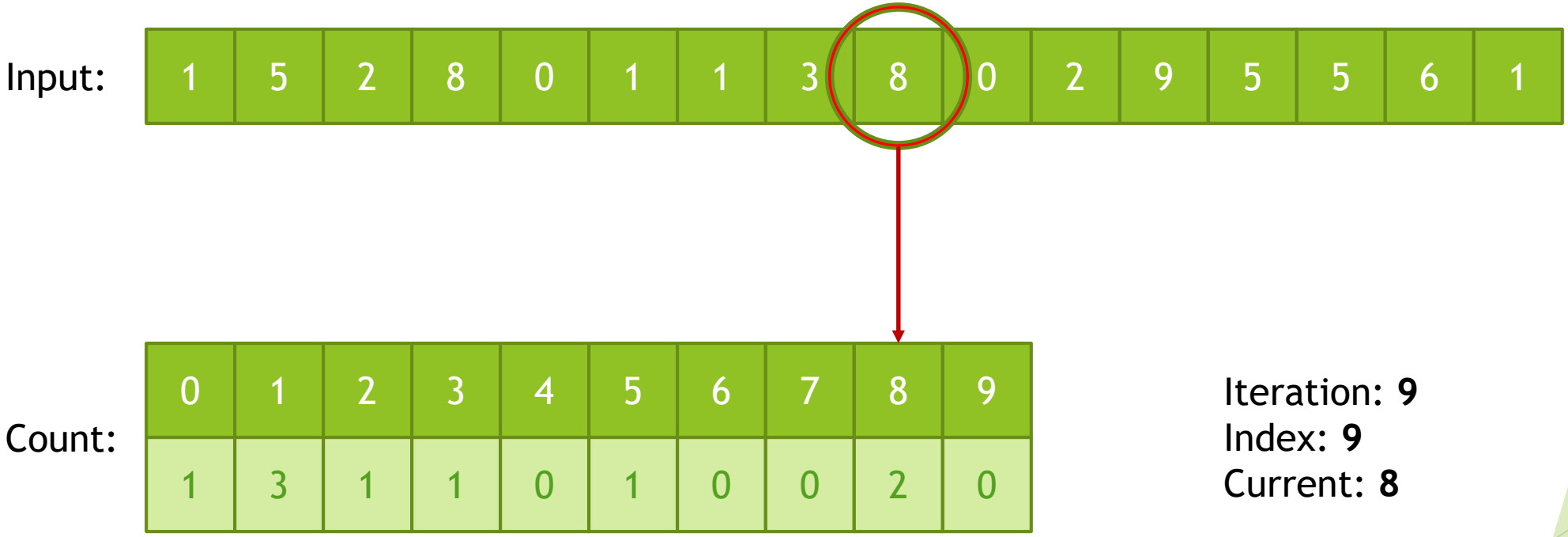
Iteration: **7**
Index: **7**
Current: **1**

Counting Sort



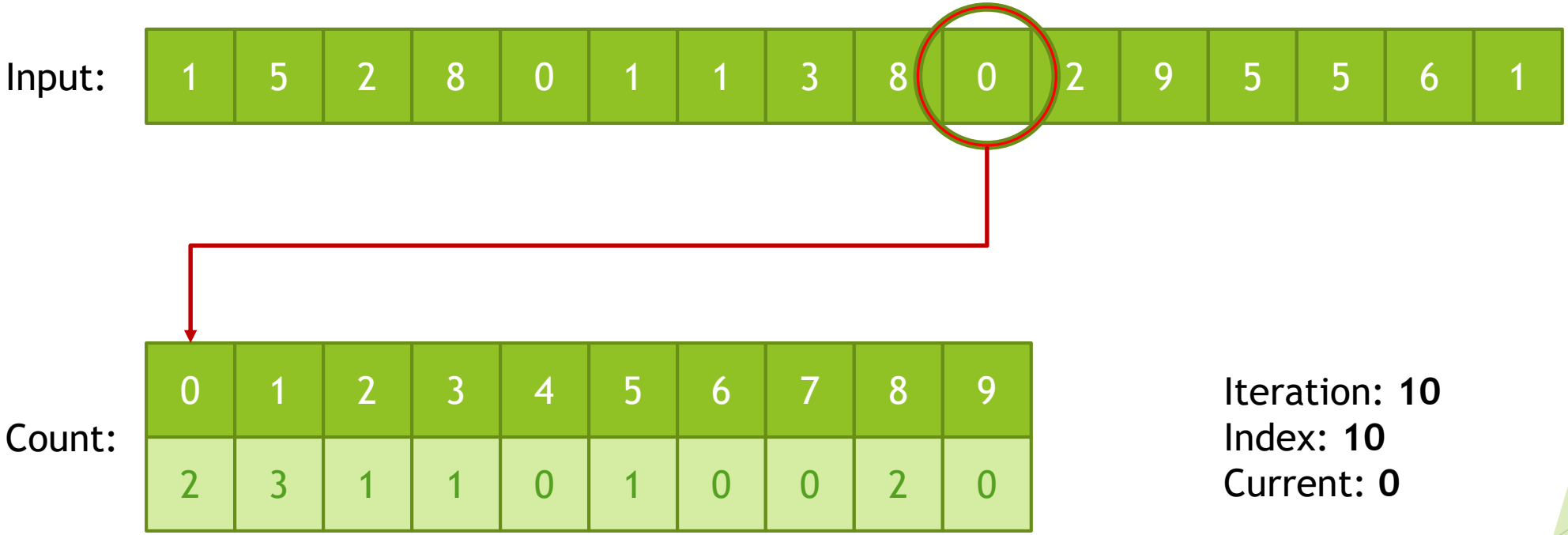
Iteration: **8**
Index: **8**
Current: **3**

Counting Sort



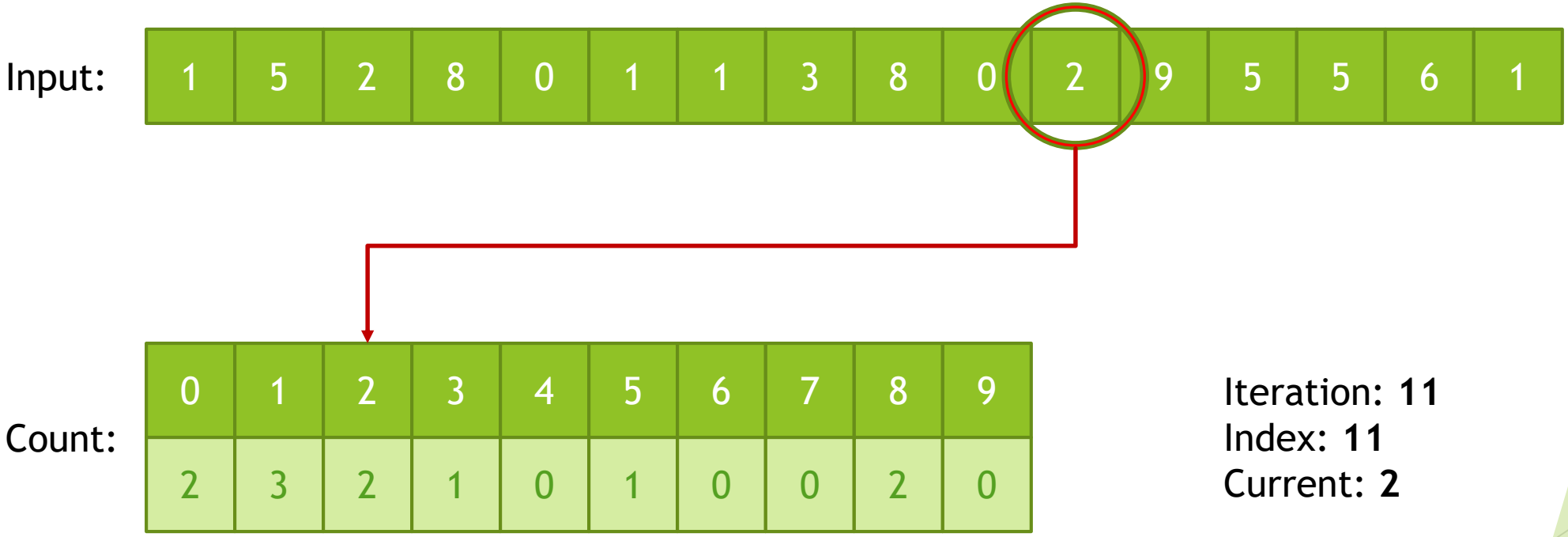
Iteration: **9**
Index: **9**
Current: **8**

Counting Sort



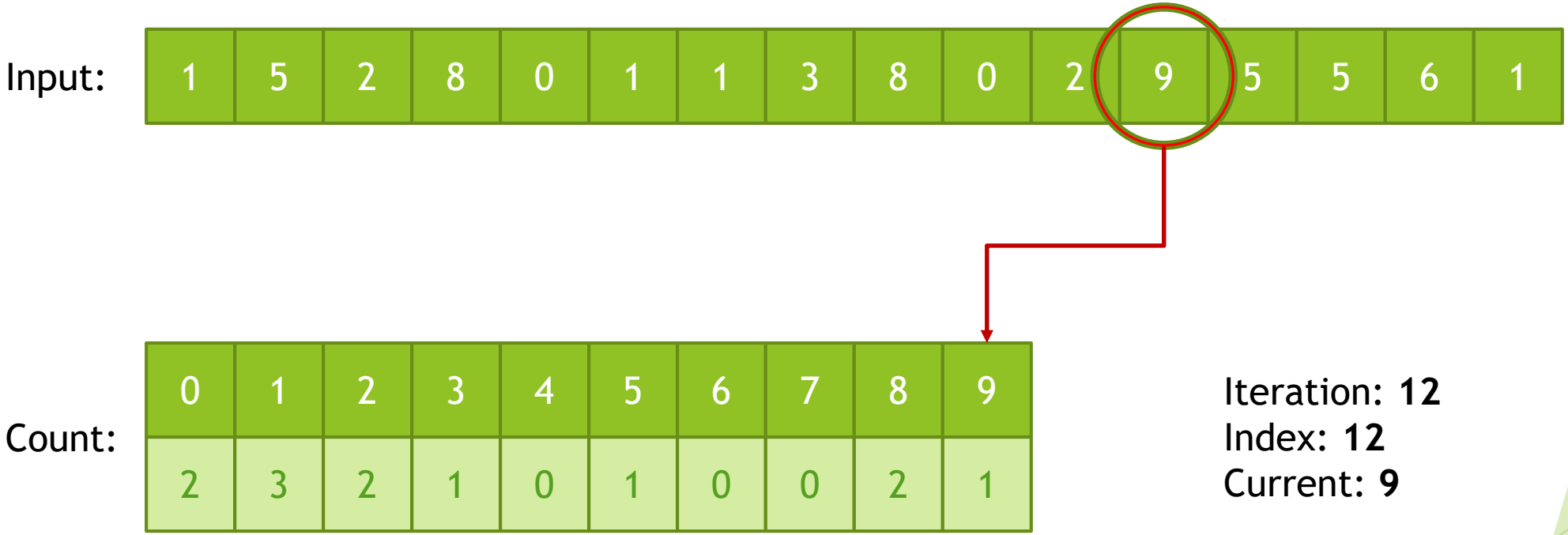
Iteration: **10**
Index: **10**
Current: **0**

Counting Sort



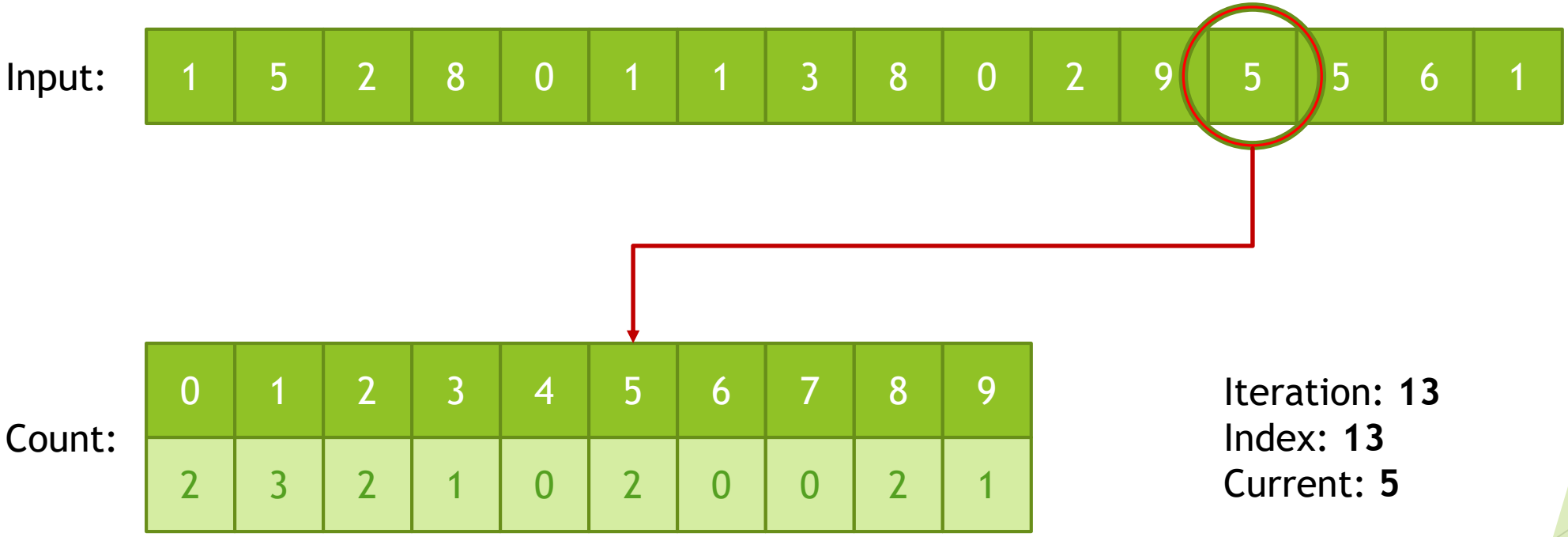
Iteration: **11**
Index: **11**
Current: **2**

Counting Sort

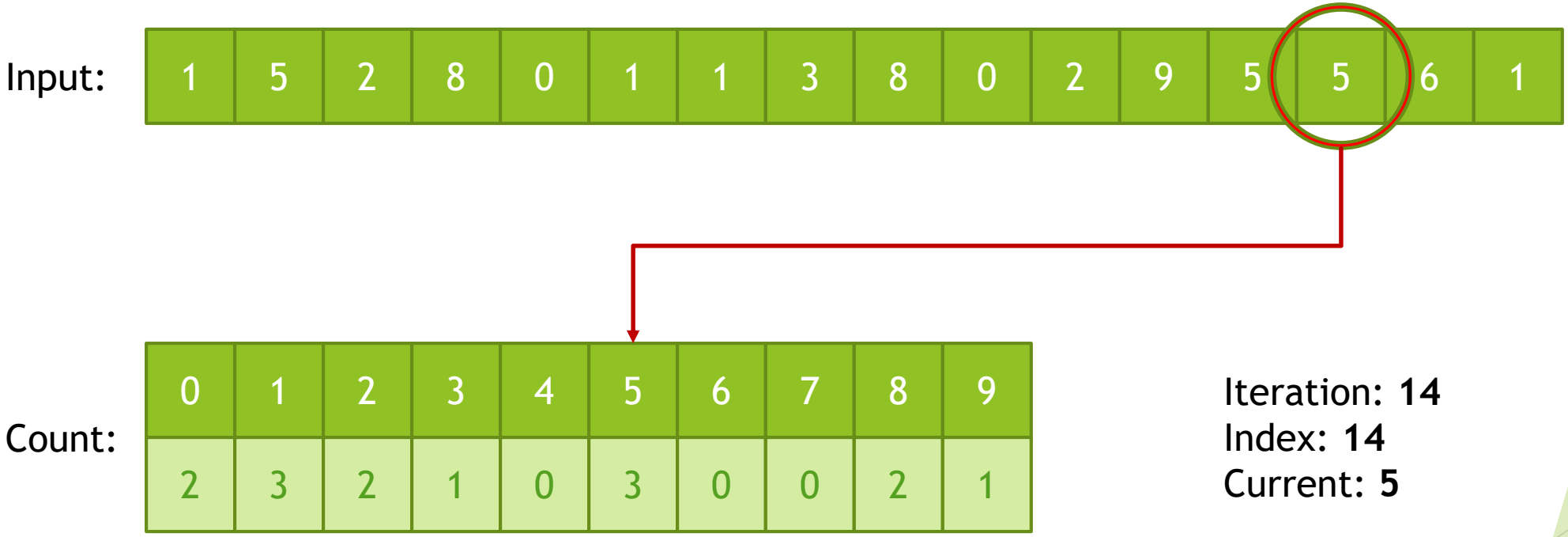


Iteration: **12**
Index: **12**
Current: **9**

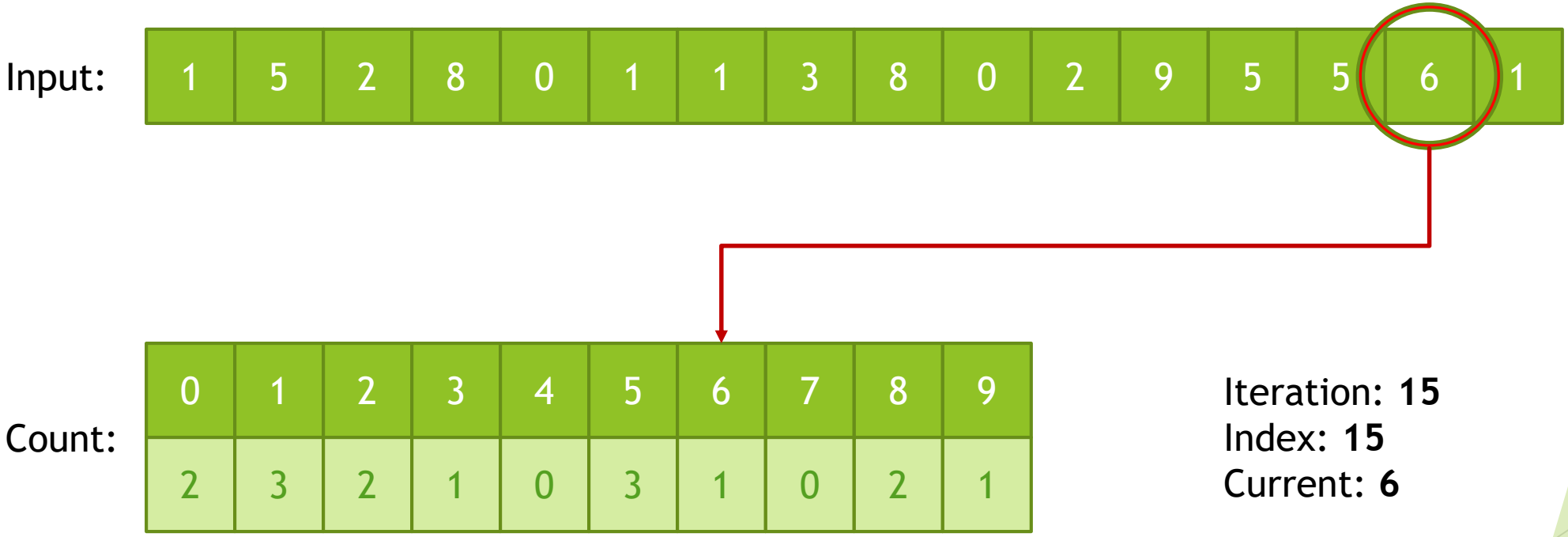
Counting Sort



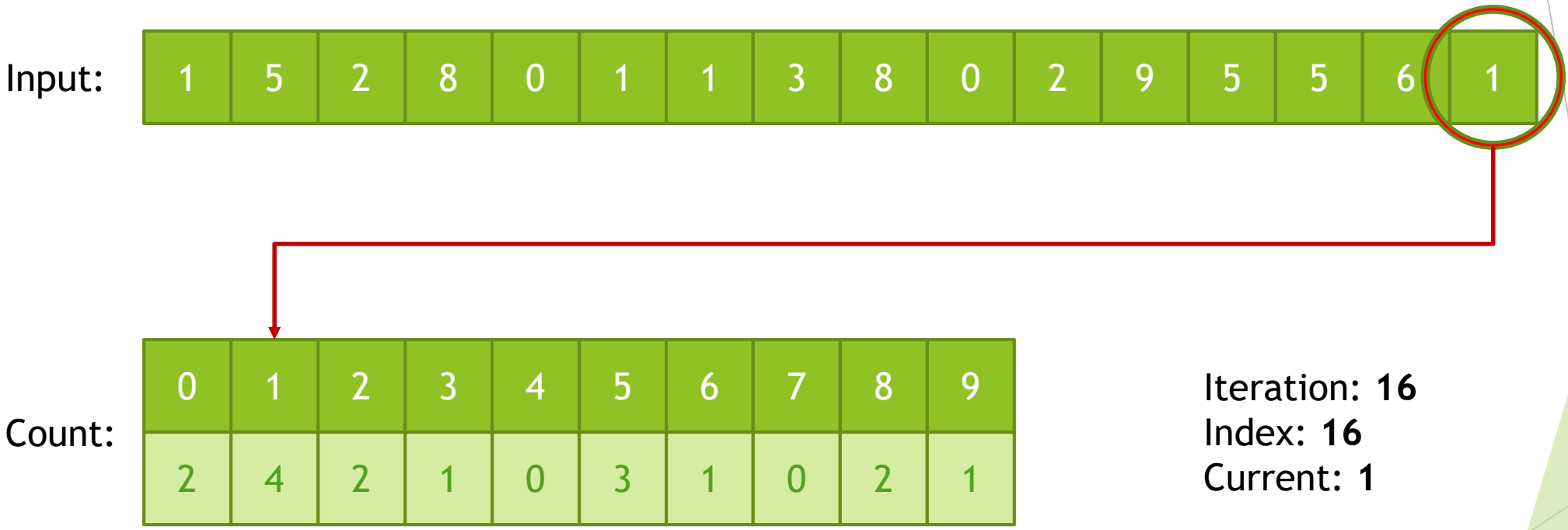
Counting Sort



Counting Sort



Counting Sort



Counting Sort

Input:

1	5	2	8	0	1	1	3	8	0	2	9	5	5	6	1

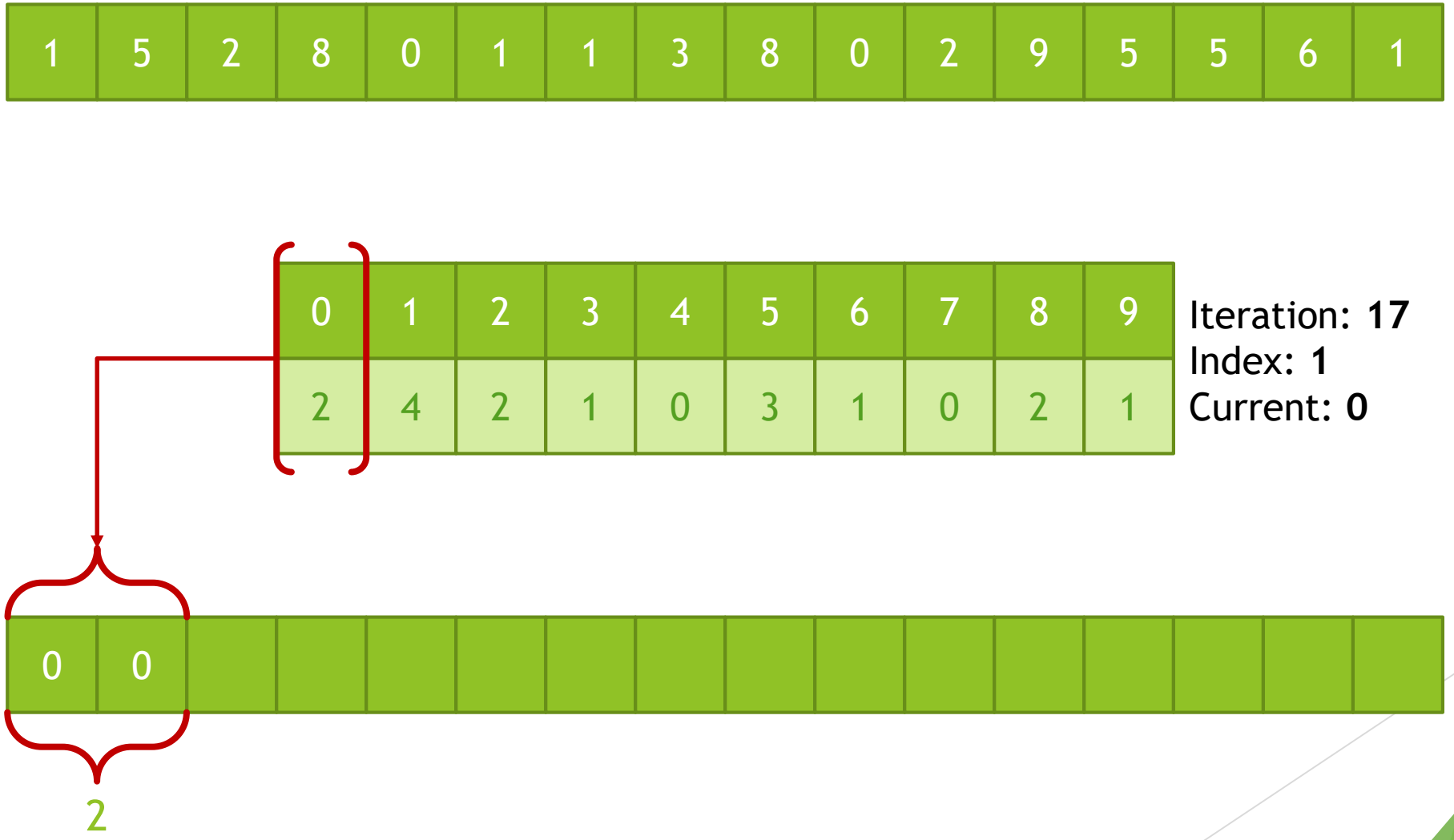
Count:

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

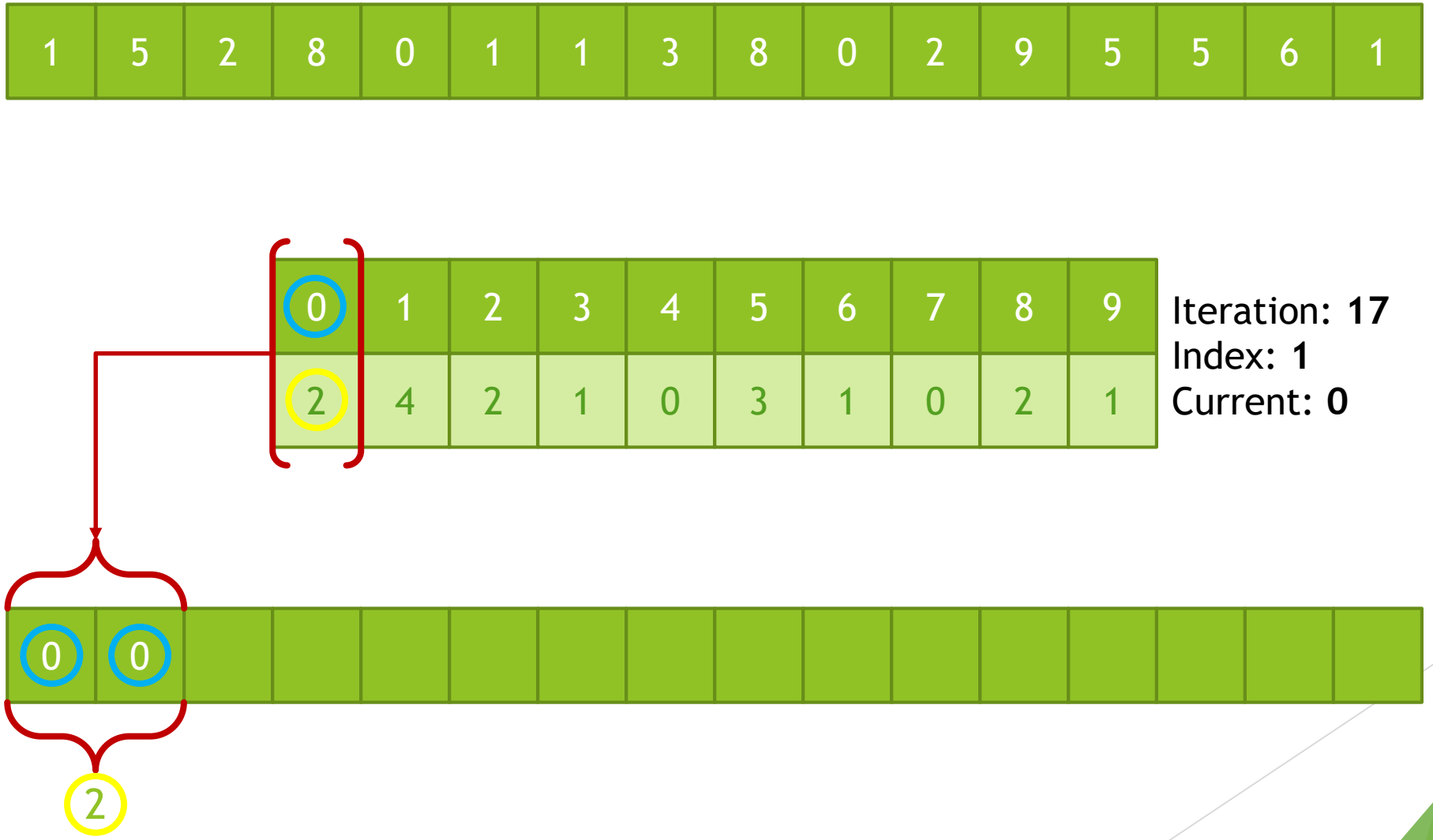
Iteration: **16**

Output:

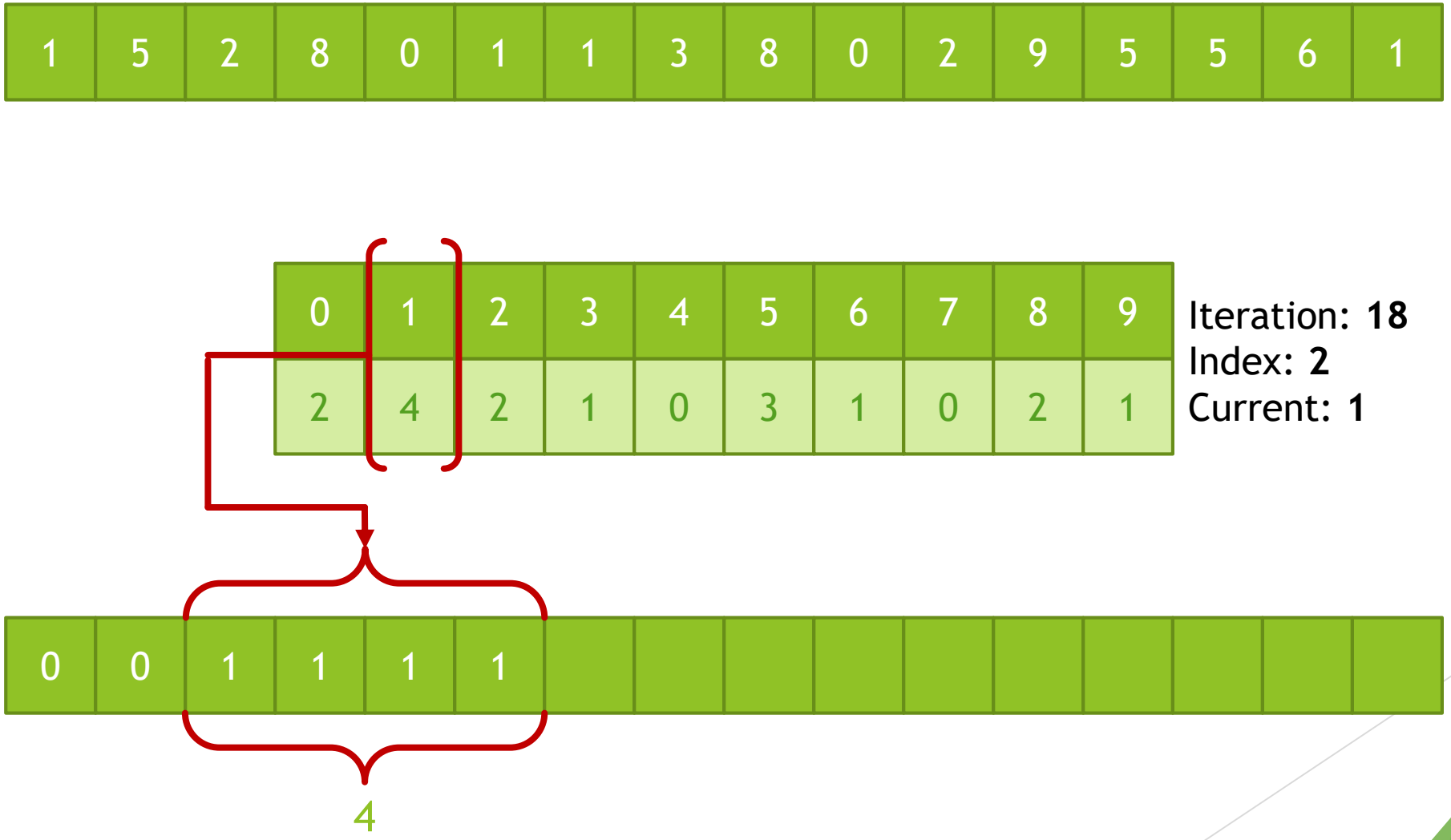
Counting Sort



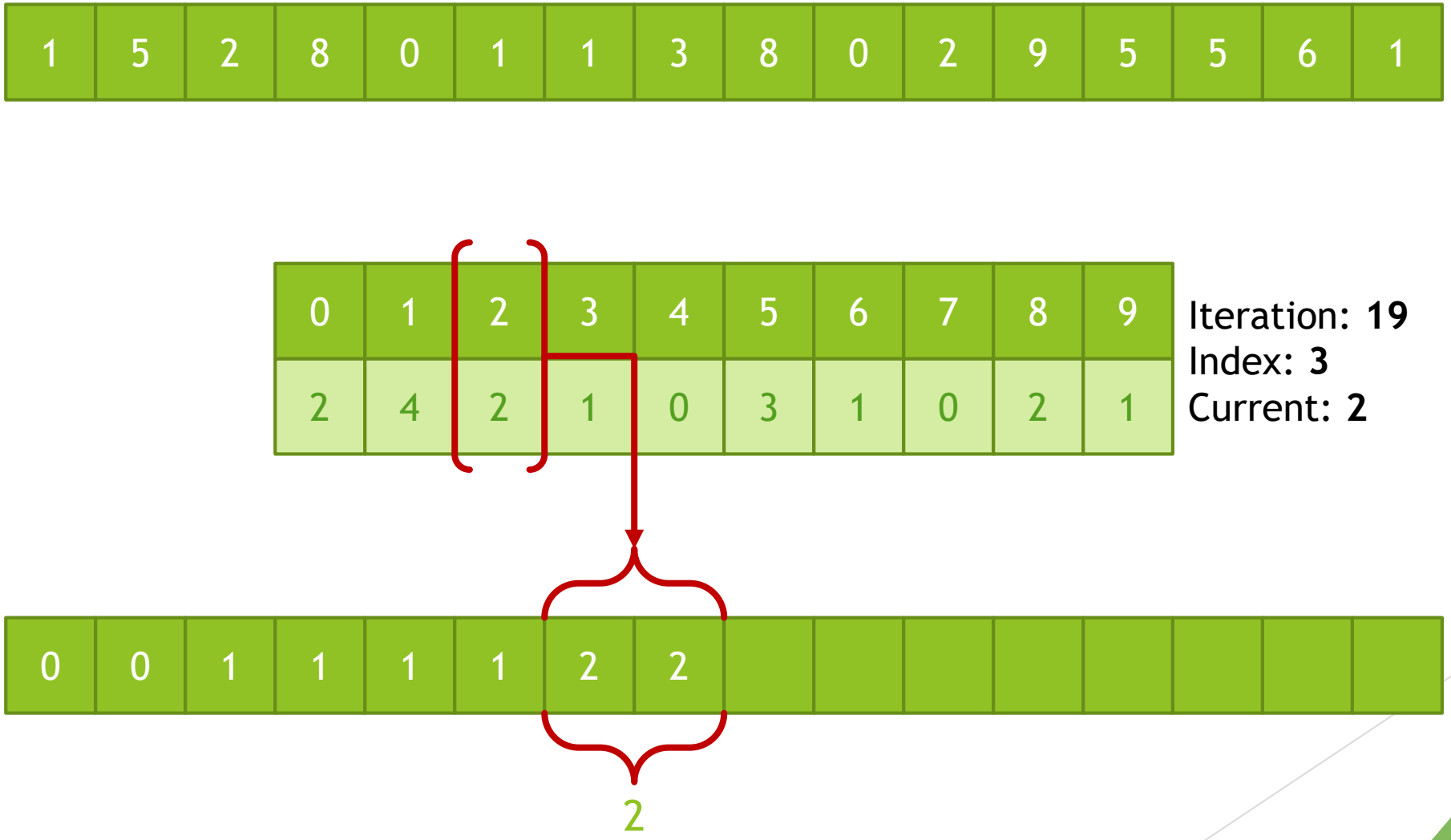
Counting Sort



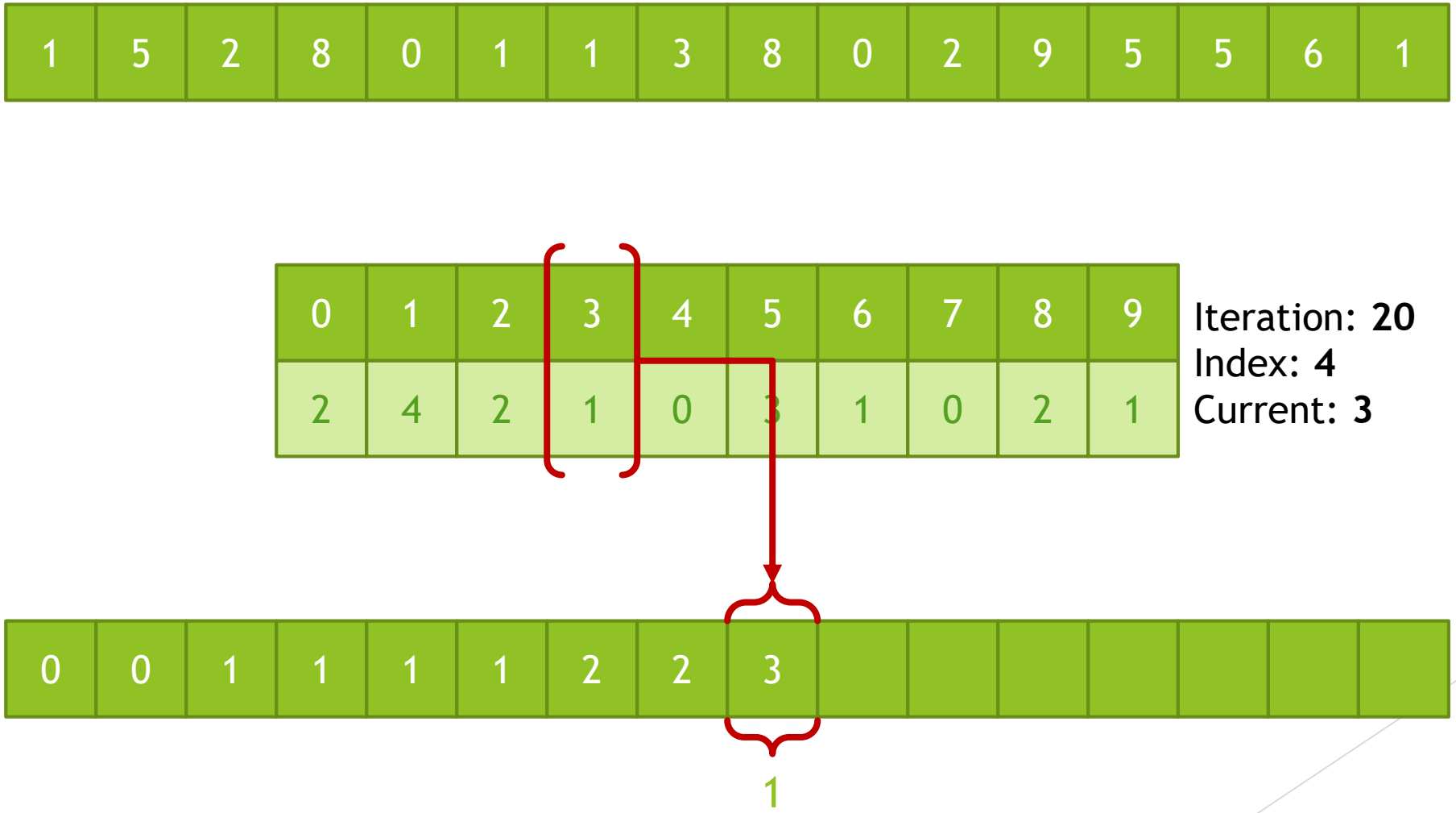
Counting Sort



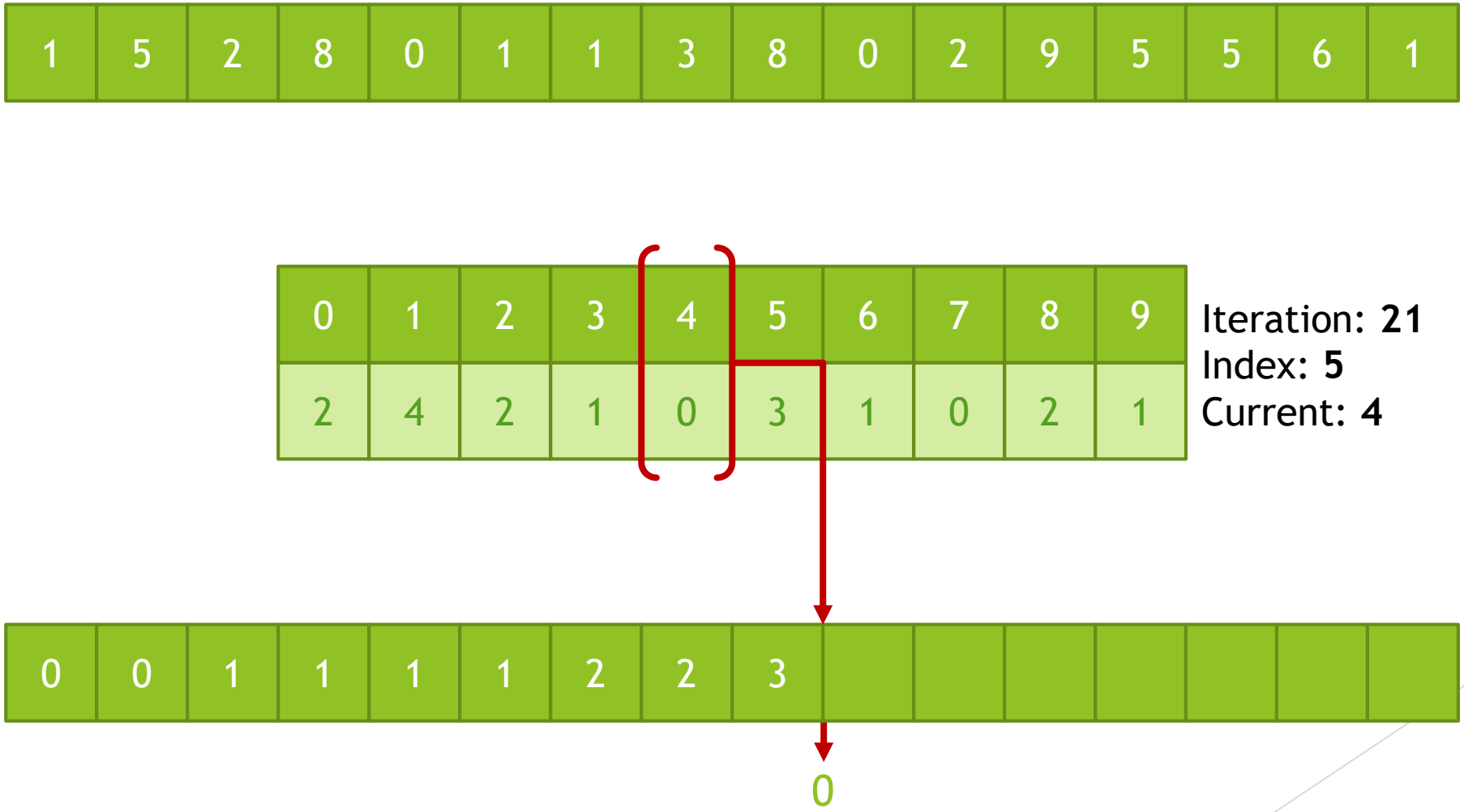
Counting Sort



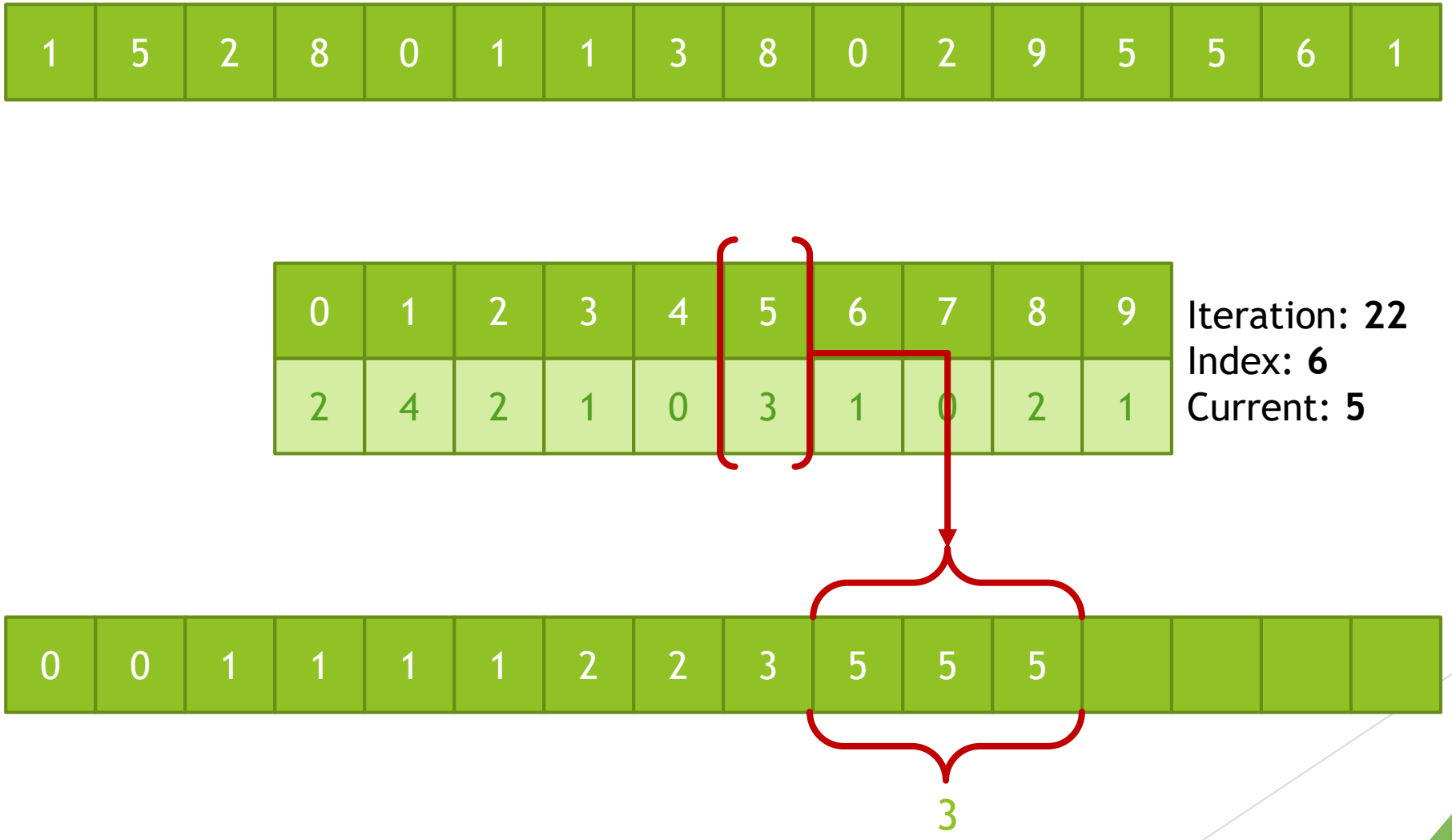
Counting Sort



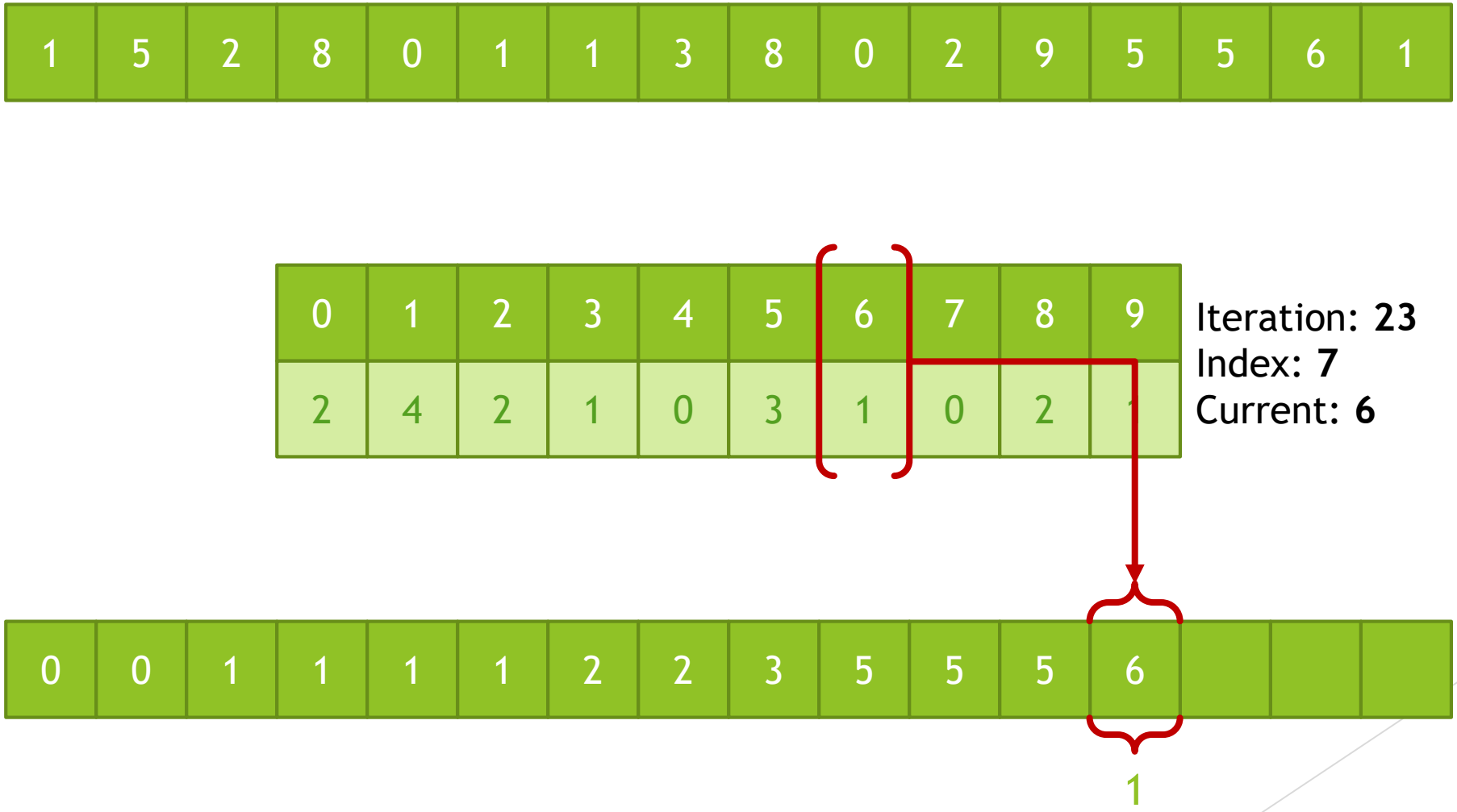
Counting Sort



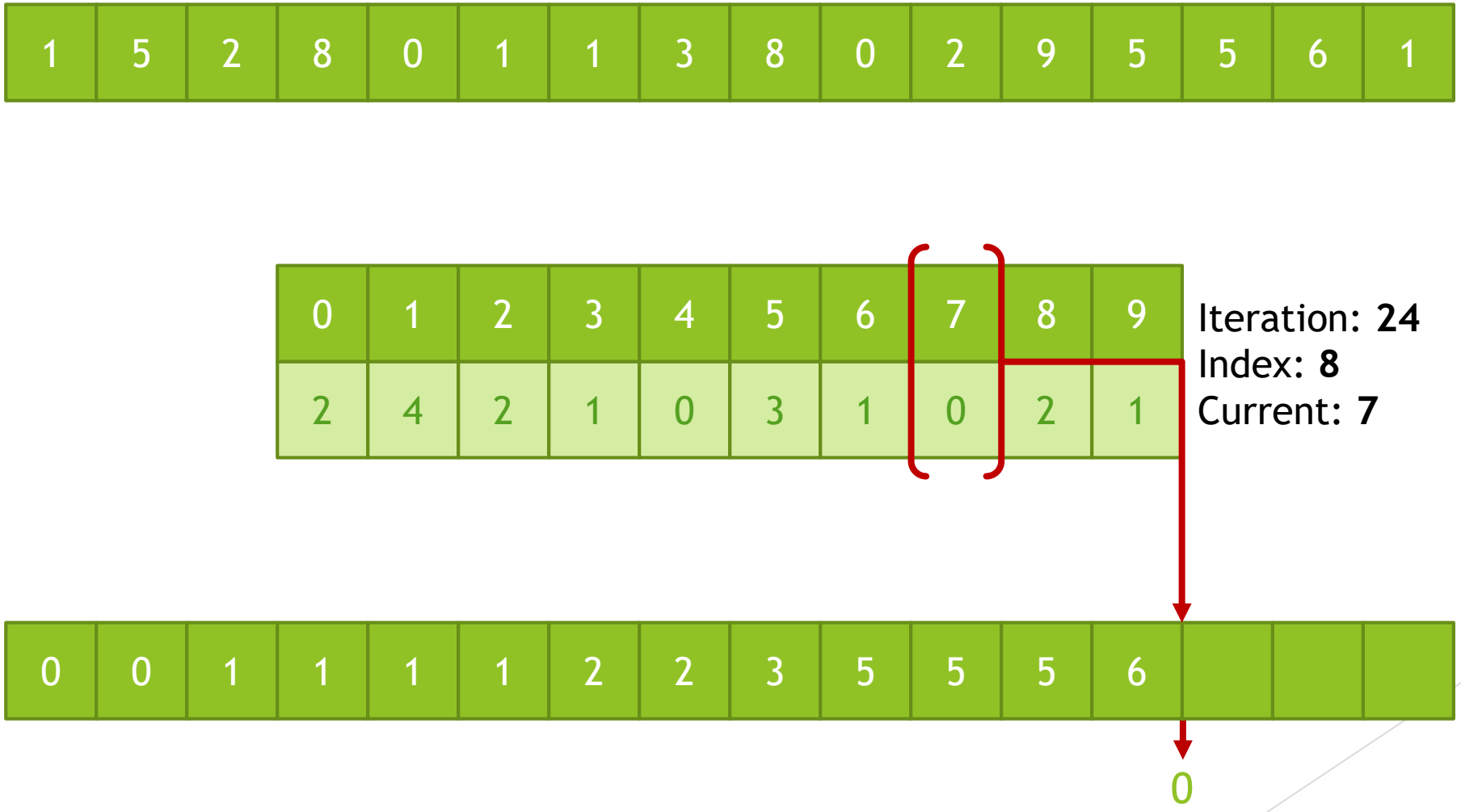
Counting Sort



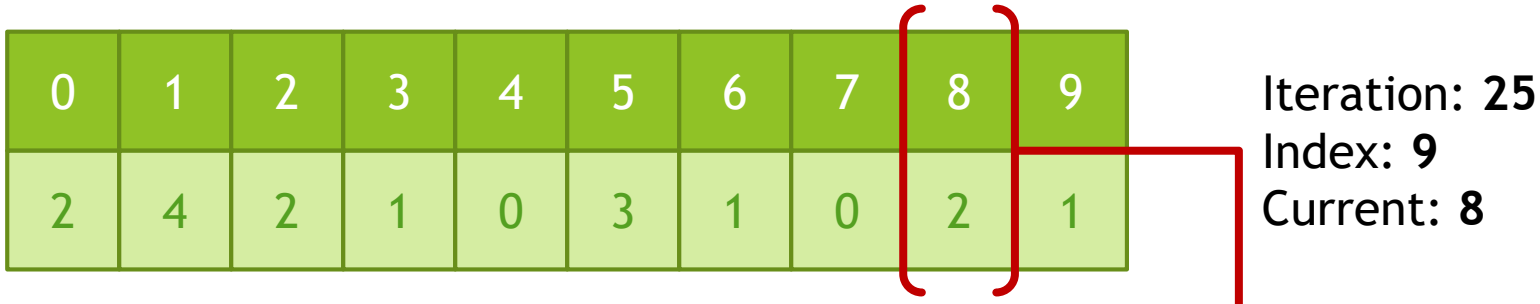
Counting Sort



Counting Sort



Counting Sort



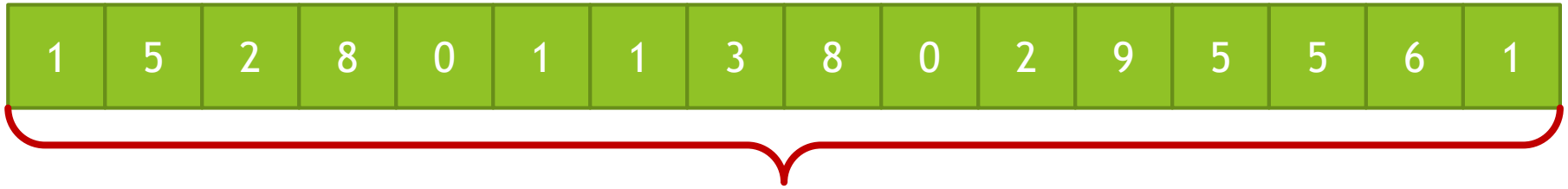
Counting Sort

Iteration: **26**
Index: **10**
Current: **9**

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

1

Counting Sort



Iteration: **26**
n + k = 16 + 10 = 26
O(n+k)