

# Counting Sort 1



## Comparison Sorting

Quicksort usually has a running time of  $n \times \log(n)$ , but is there an algorithm that can sort even faster? In general, this is not possible. Most sorting algorithms are *comparison sorts*, i.e. they sort a list just by comparing the elements to one another. A comparison sort algorithm cannot beat  $n \times \log(n)$  (worst-case) running time, since  $n \times \log(n)$  represents the minimum number of comparisons needed to know where to place each element. For more details, you can see [these notes](#) (PDF).

## Alternative Sorting

Another sorting method, the *counting sort*, does not require comparison. Instead, you create an integer array whose index range covers the entire range of values in your array to sort. Each time a value occurs in the original array, you increment the counter at that index. At the end, run through your counting array, printing the value of each non-zero valued index that number of times.

For example, consider an array  $arr = [1, 1, 3, 2, 1]$ . All of the values are in the range  $[0 \dots 3]$ , so create an array of zeroes,  $result = [0, 0, 0, 0]$ . The results of each iteration follow:

```
i arr[i] result
0 1 [0, 1, 0, 0]
1 1 [0, 2, 0, 0]
2 3 [0, 2, 0, 1]
3 2 [0, 2, 1, 1]
4 1 [0, 3, 1, 1]
```

Now we can print the list of occurrences, **0 3 1 1** or determine the sorted array:  $sorted = [1, 1, 1, 2, 3]$ .

## Challenge

Given a list of integers, count and output the number of times each value appears as a list of space-separated integers.

## Function Description

Complete the `countingSort` function in the editor below. It should return an array of integers where each value is the number of occurrences of the element's index value in the original array.

`countingSort` has the following parameter(s):

- *arr*: an array of integers

## Input Format

The first line contains an integer  $n$ , the number of items in *arr*.  
Each of the next  $n$  lines contains an integer  $arr[i]$  where  $0 \leq i < n$ .

## Constraints

$$100 \leq n \leq 10^6$$
$$0 \leq arr[i] < 100$$

## Output Format

Output the number of times every number from 0 through 99 appears in *arr* as a list of space-separated integers on one line.

## Sample Input

```
100
63 25 73 1 98 73 56 84 86 57 16 83 8 25 81 56 9 53 98 67 99 12 83 89 80 91 39 86 76 85 74 39 25 90 59 10 94 32 44 3 89 30 27
79 46 96 27 32 18 21 92 69 81 40 40 34 68 78 24 87 42 69 23 41 78 22 6 90 99 89 50 30 20 1 43 3 70 95 33 46 44 9 69 48 33 60
```

```
65 16 82 67 61 32 21 79 75 75 13 87 70 33
```

### Sample Output

```
0 2 0 2 0 0 1 0 1 2 1 0 1 1 0 0 2 0 1 0 1 2 1 1 1 3 0 2 0 0 2 0 3 3 1 0 0 0 0 2 2 1 1 1 2 0 2 0 1 0 1 0 0 1 0 0 2 1 0 1 1 1 0 1 0 1 0 2 1 3 2
0 0 2 1 2 1 0 2 2 1 2 1 2 1 1 2 2 0 3 2 1 1 0 1 1 1 0 2 2
```

### Explanation

Each of the resulting values *result*[*i*] represents the number of times *i* appeared in *arr*.