



All Domains &gt; Algorithms &gt; Sorting &gt; Counting Sort 1

Badge Progress [\(Details\)](#)

Points: 1440.13 Rank: 4843

# Counting Sort 1

by [idlecool](#)

Problem

Submissions

Leaderboard

Discussions

## 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

However, for certain types of input, it is more efficient to use a non-comparison sorting algorithm. This will make it possible to sort lists even in linear time. These challenges will cover *Counting Sort*, a fast way to sort lists where the elements have a small number of possible values, such as integers within a certain range. We will start with an easy task - counting.

## Challenge

Given a list of integers, can you count and output the number of times each value appears?

*Hint:* There is no need to sort the data, you just need to count it.

## Input Format

There will be two lines of input:

- $n$  - the size of the list
- $ar$  -  $n$  space-separated numbers that make up the list

## Output Format

Output the number of times every number from 0 to 99 (inclusive) appears on the list.

## Constraints

$$100 \leq n \leq 10^6$$

$$0 \leq x < 100, x \in ar$$

## 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

The output states that **0** appears 0 times, **1** appears 2 times, **2** appears 0 times, and so on in the given input array.

Copyright © 2016 HackerRank.  
All Rights Reserved

Submissions: 11867

Max Score: 30

Difficulty: Easy

[More](#)

Current Buffer (saved locally, editable)  

Java 8



```
1 import java.io.*;
2 import java.util.*;
3
4 public class Solution {
5
6     public static void main(String[] args) {
7         /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named
8         Solution. */
9     }
10 }
```

Line: 1 Col: 1

 [Upload Code as File](#)

☐ Test against custom input

Run Code

Submit Code

Join us on IRC at [#hackerrank](#) on freenode for hugs or bugs.

[Contest Calendar](#) | [Blog](#) | [Scoring](#) | [Environment](#) | [FAQ](#) | [About Us](#) | [Support](#) | [Careers](#) | [Privacy Policy](#) | [Request a Feature](#)