**Day 36**

**Counting Sort**

[*Counting sort*](http://en.wikipedia.org/wiki/Counting_sort)*is a sorting technique based on keys between a specific range. It works by counting the number of objects having distinct key values (a kind of hashing). Then do some arithmetic operations to calculate the position of each object in the output sequence.*

**Characteristics of counting sort:**

* Counting sort makes assumptions about the data, for example, it assumes that values are going to be in the range of 0 to 10 or 10 – 99, etc, Some other assumption counting sort makes is input data will be all real numbers.
* Like other algorithms this sorting algorithm is not a comparison-based algorithm, it hashes the value in a temporary count array and uses them for sorting.
* It uses a temporary array making it a non-In[Place algorithm](https://www.geeksforgeeks.org/in-place-algorithm/).

Recommended Problem

Counting Sort

[Sorting](https://practice.geeksforgeeks.org/explore?page=1&category%5b%5d=Sorting&sortBy=submissions)

[Algorithms](https://practice.geeksforgeeks.org/explore?page=1&category%5b%5d=Algorithms&sortBy=submissions)

[Cisco](https://practice.geeksforgeeks.org/explore?page=1&company%5b%5d=Cisco&sortBy=submissions)

[Goldman Sachs](https://practice.geeksforgeeks.org/explore?page=1&company%5b%5d=Goldman%20Sachs&sortBy=submissions)

+1 more

[Solve Problem](https://practice.geeksforgeeks.org/problems/counting-sort/1?utm_source=gfg&utm_medium=article&utm_campaign=bottom_sticky_on_article" \o "Permalink to Counting Sort)

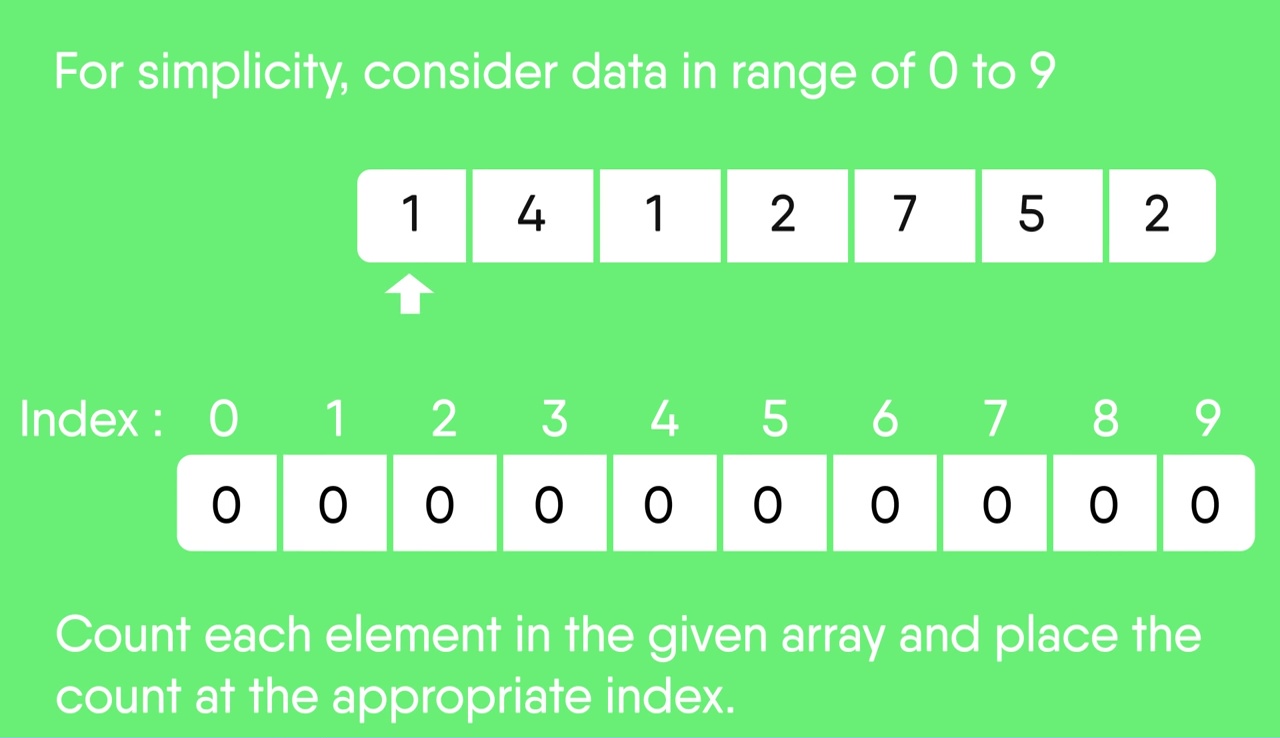
Submission count: 30K

**Example:**

* For simplicity, consider the data in the range of 0 to 9.
* **Input data:** {1, 4, 1, 2, 7, 5, 2}
* Take a count array to store the count of each unique object.

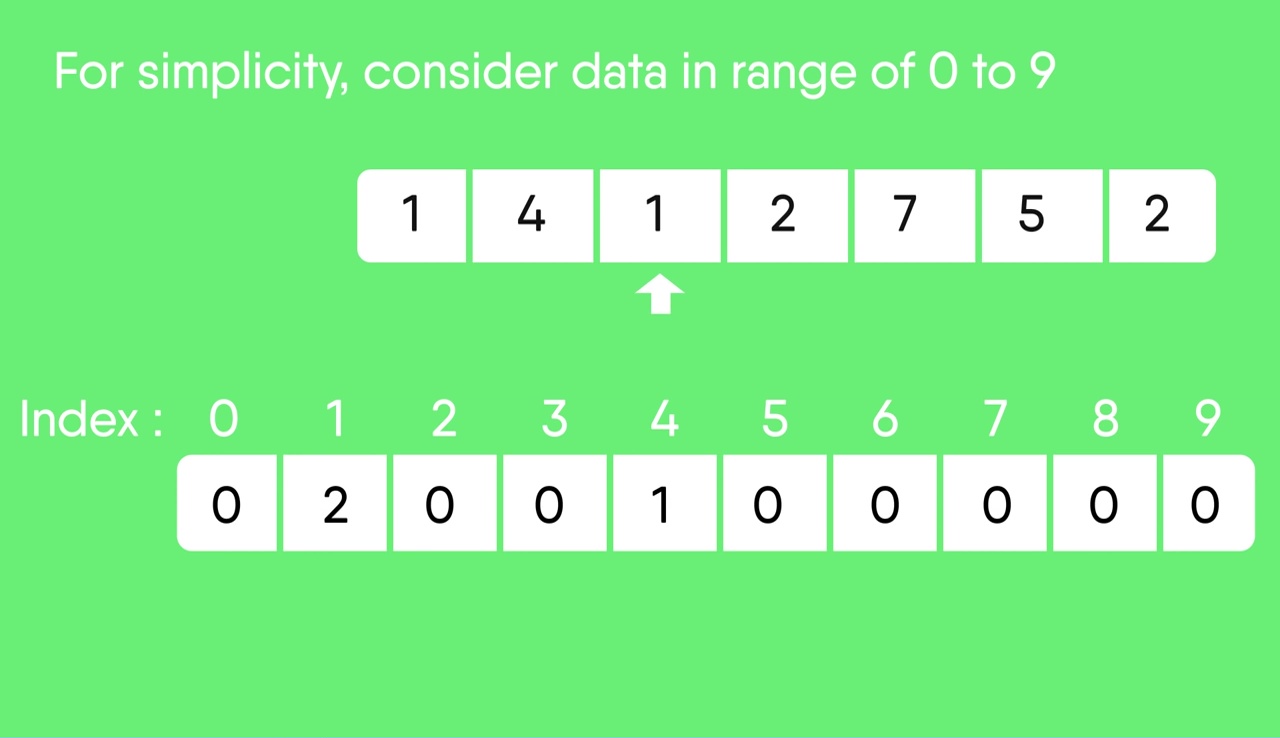
Follow the below illustration for better understanding of the counting sort algorithm

**Illustration:**

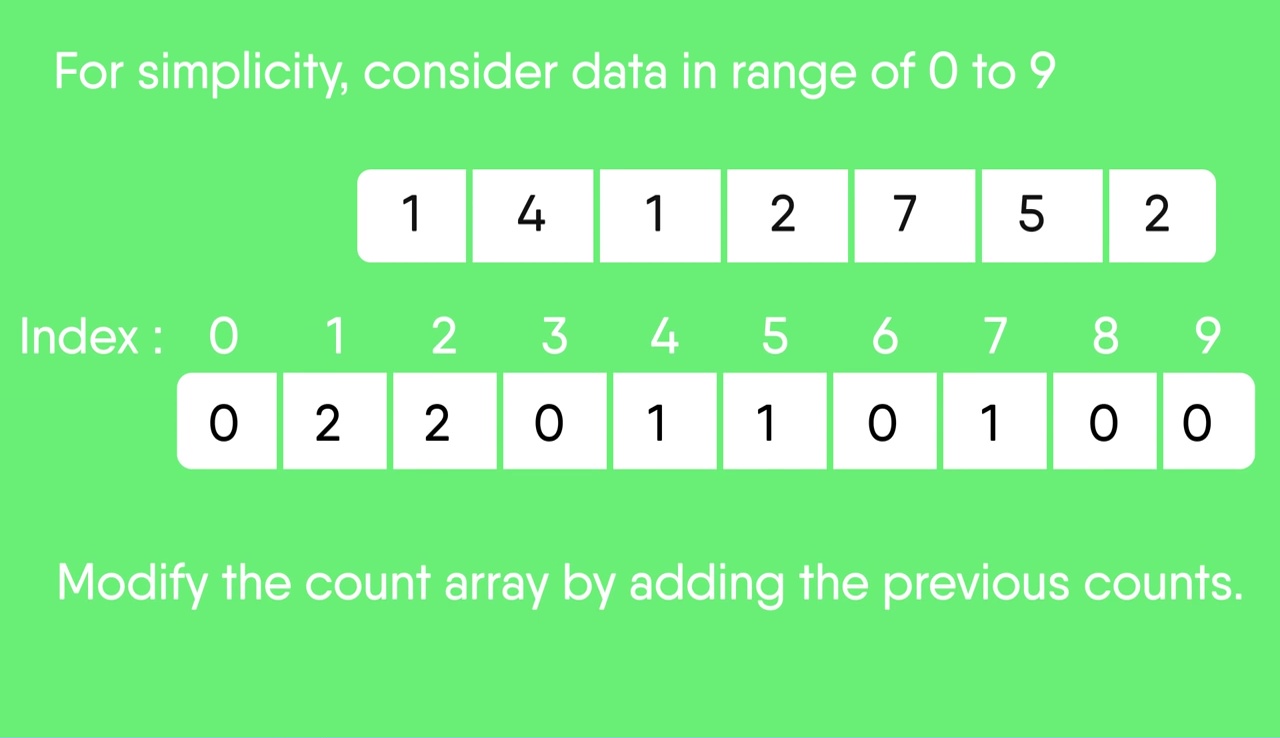


Advertisement

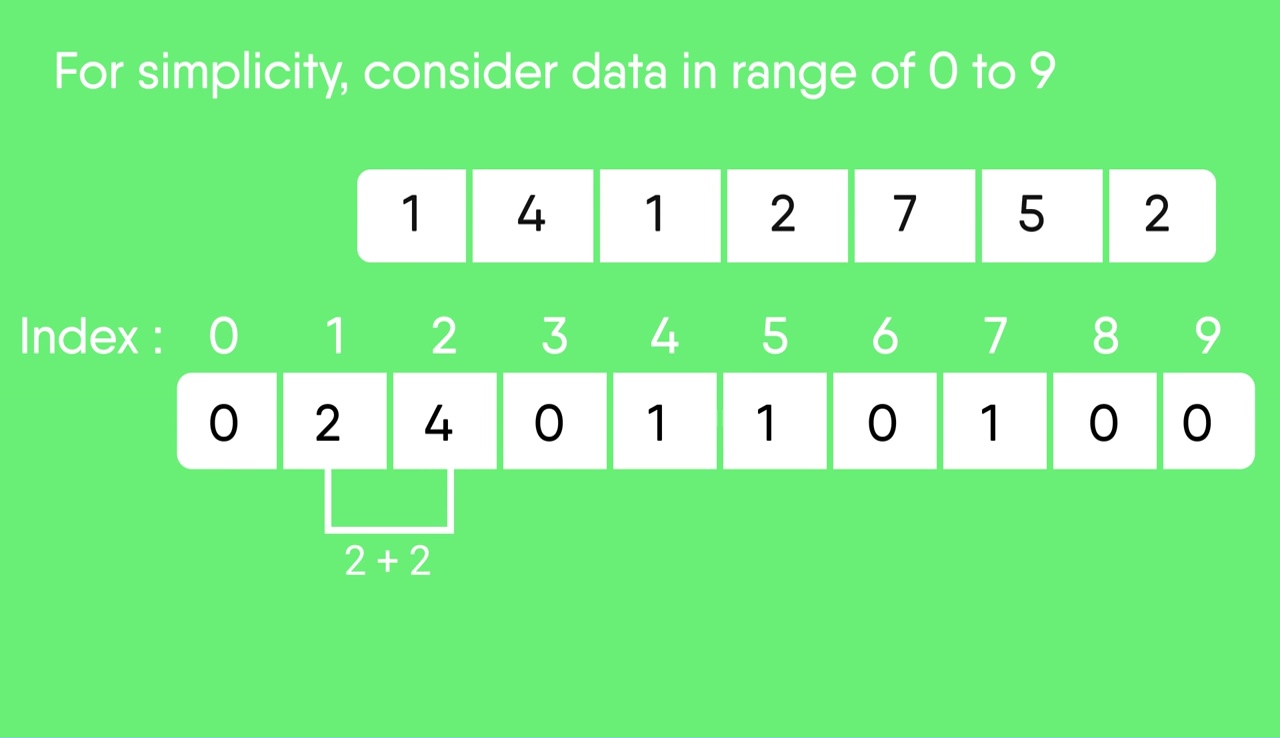
* Now, store the count of each unique element in the count array
* If any element repeats itself, simply increase its count.

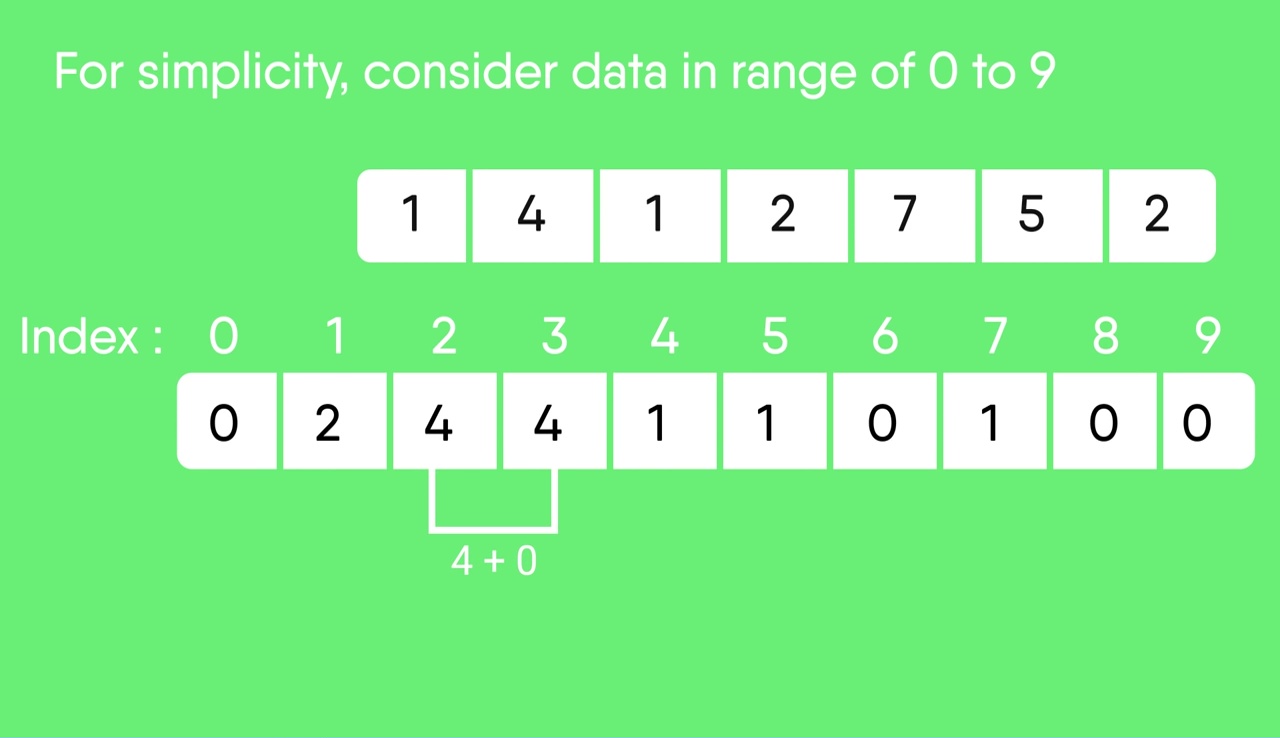


* Here, the count of each unique element in the count array is as shown below:
  + **Index:**  0  1  2  3  4  5  6  7  8  9
  + **Count:** 0  2  2  0   1  1  0  1  0  0

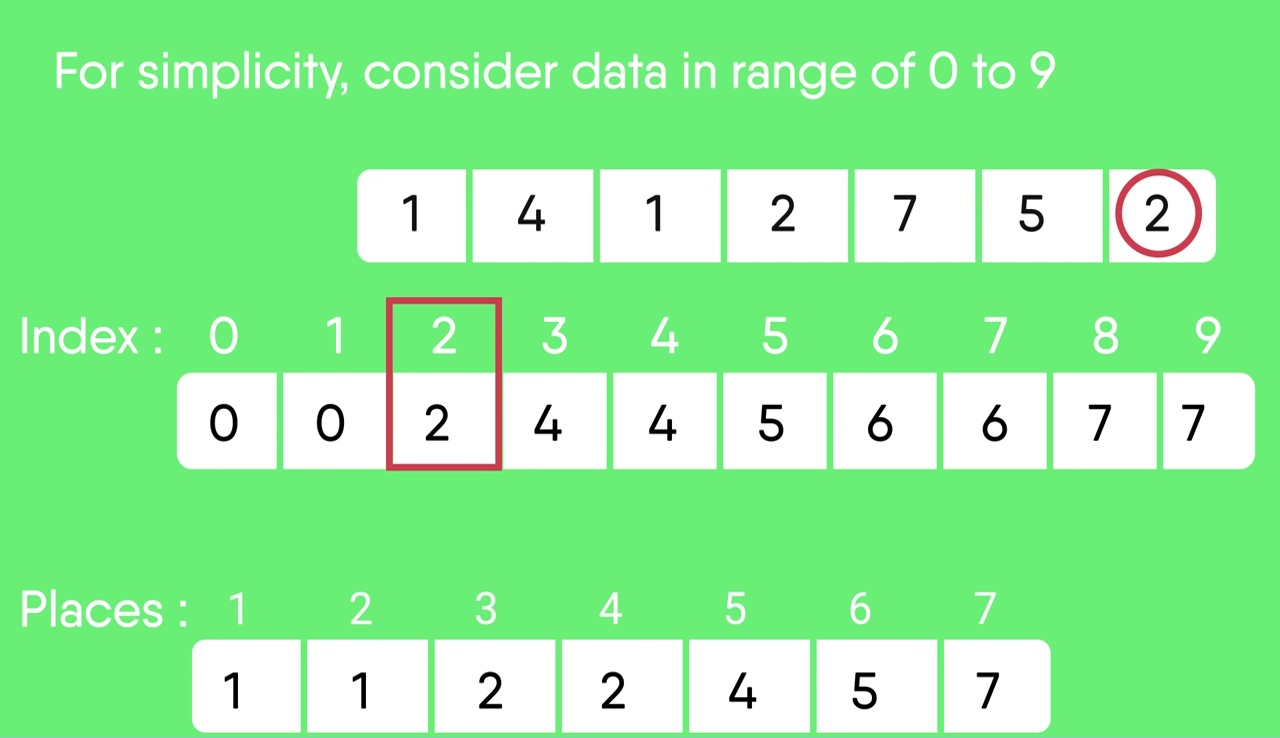


* Modify the count array such that each element at each index stores the sum of previous counts.
  + **Index:**   0  1  2  3  4  5  6  7  8  9
  + **Count:**  0  2  4  4  5  6  6  7  7  7
* The modified count array indicates the position of each object in the output sequence.
* Find the index of each element of the original array in the count array. This gives the cumulative count.

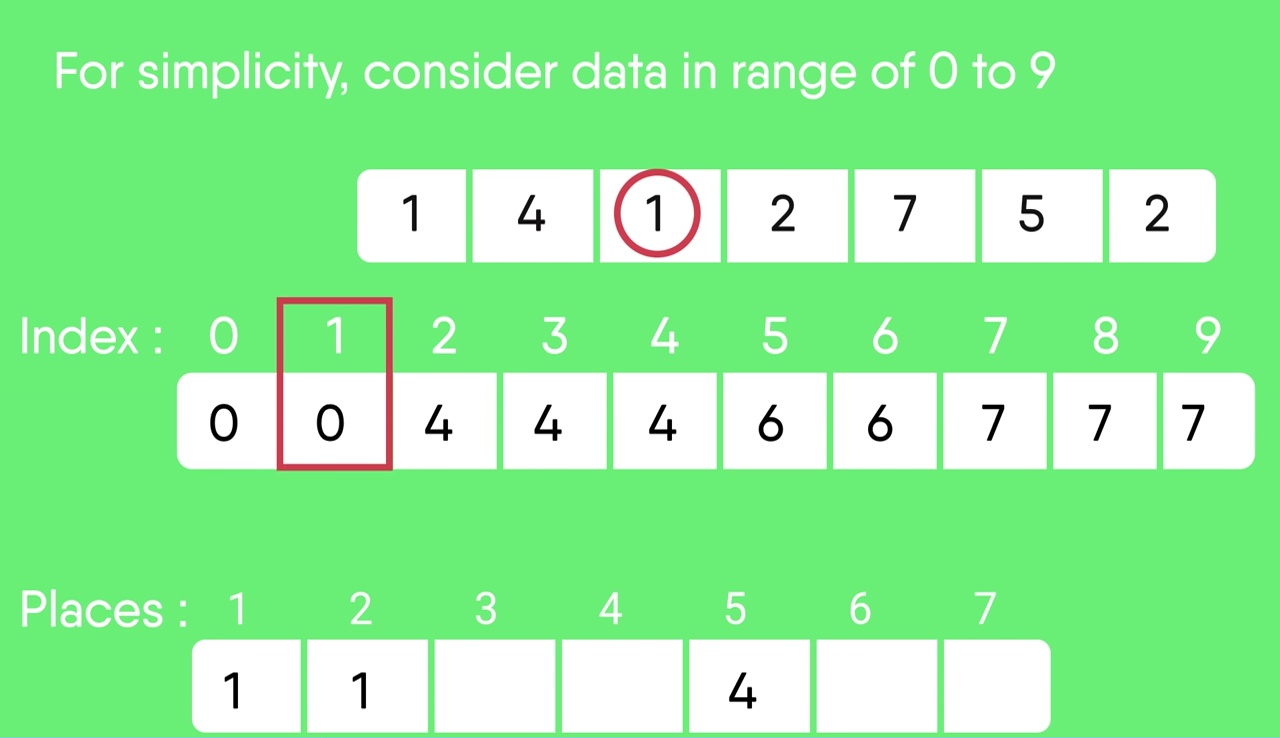




* Rotate the array clockwise for one time.
  + **Index:**0 1 2 3 4 5 6 7 8 9
  + **Count:** 0 0 2 4 4 5 6 6 7 7



* Output each object from the input sequence followed by increasing its count by 1.
* Process the input data: {1, 4, 1, 2, 7, 5, 2}. The position of 1 is 0.
* Put data 1 at index 0 in output. Increase count by 1 to place next data 1 at an index 1 greater than this index.



* After placing each element in its correct position, decrease its count by one.