Count Sort Algorithm

→ Suppose, we are given an array:

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

US ARRAY:

1) The algorithm first asks you to find the largest element from all the elements in the array and store it in some integer variable max. Then create a count array of size

MAX+1. This array would count the no. of occurances of some number in the given array. We will have to initialize all count array elements with 0 for that to work.

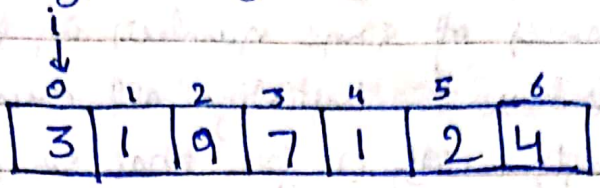
2) After initializing the count array, traverse through the given array, and increment the value of that element in the count array by 1. By defining the size of the count array as the maximum element in the array, you ensure that each element in the array has its own corresponding index in the count array. After we traverse through the whole array, we'll have the count of each element in the array.

3) Now traverse through the count array, ~~and~~ and look for the non-zero elements. The moment you find an index with some value other than zero, fill in the sorted array the index of the non-zero element until it becomes zero by decrementing it by 1 every time you fill it in the resultant array. Only then move further. This way, you create a sorted array.

→ First of all, find the maximum element in the array. Here, it is 9. So, we'll create a count array of size 10 and

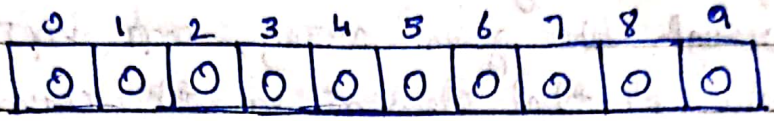
Page No. _____
Date

→ initialize every element by 0.



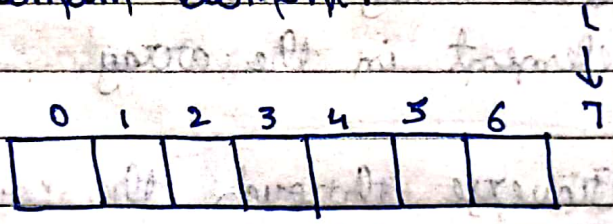
MAX = 9

US

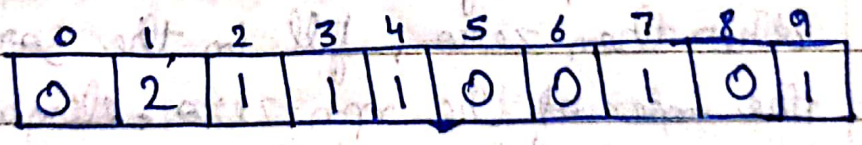


Count Array

→ Now, let is iterate through the given array and count the no. of occurances of each number less than equal to the maximum element.

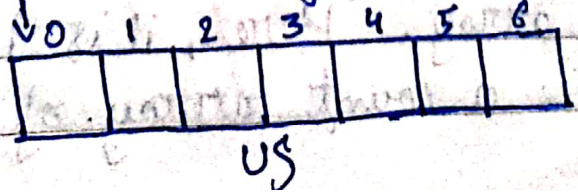


US

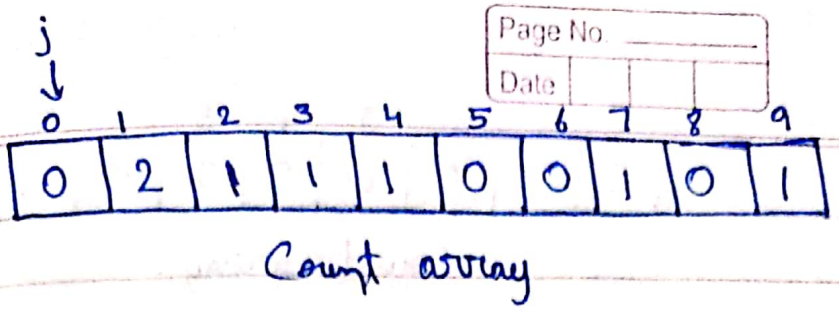


Count Array

→ We would iterate through the count array and fill the unsorted array with the index we encounter having a non-zero number of occurances.

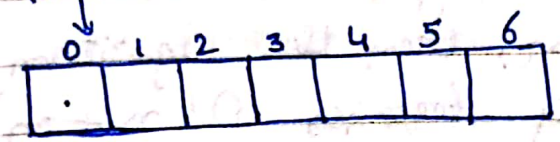


US

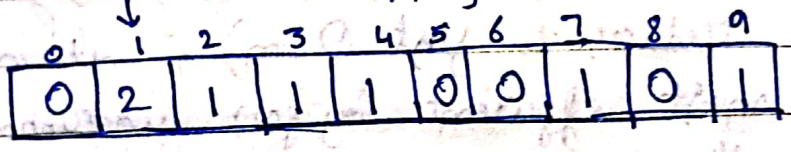


Count array

→ Now, since there are zero number of zeros in the given array, we move further to index 1. i

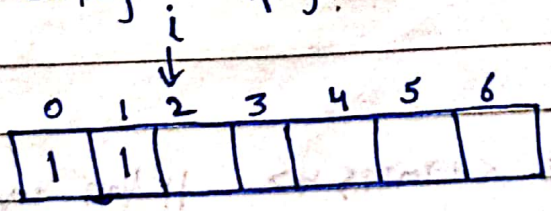


j Sorted Array

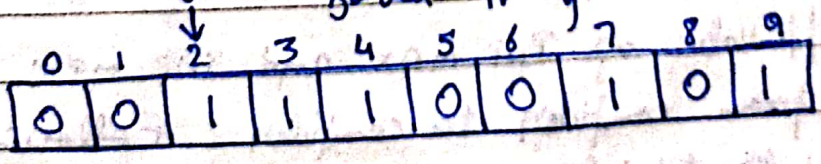


Count Array.

→ And since there were two ones in the array we were given, we push two ones in the sorted array and move our iterator to the next ~~emty~~ empty index.

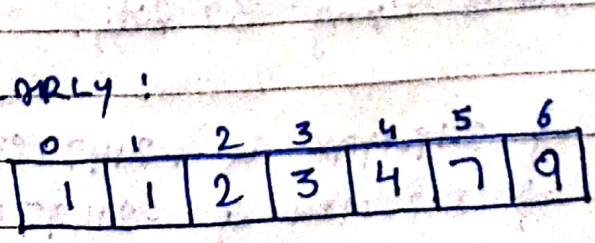


j Sorted Array



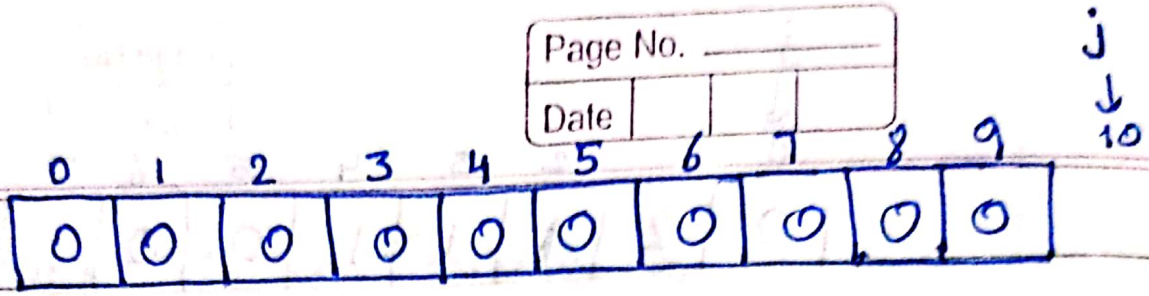
Count Array

→ Similarly:



Sorted Array.

Page No.
Date



Count array.

→ Time complexity of Count Sort:

→ If we suppose the maximum element to be m, then the algorithm's time complexity becomes O(n+m), and for an array of some huge size, this reduces to just O(n), which is the most efficient by far algorithm. However there is a negative point as well. The algorithm uses extra space for the count array. And this linear complexity is reachable only at the cost of the space the count array takes.