③ **COUNT SORT** → Kab lagaty hai : Jab humare pe elements bahut saare ho! Pr array ke ander jo data hai vo bahut vary na kra ho!

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
9	6	3	5	3	4	3	9	6	4	6	5	8	9	9

- Humare pe ek array hai ⑮ size ka!
- Aur uss array me elements saare [3 – 9] ke beech hi hai! Range jyada nahi hai!

KESE KRENGY → Toh chaliye Shuru krty hai! ☺

- Sabse phele given array ko scan krke uska minimum aur maximum identify krlety hai

  ∴ minimum = 3
  maximum = 9

- Aab hum ek frequency array banayengy of size equal to range.

  ∴ Range = Max – Min + 1
  Range = 9 – 3 + 1 = ⑦ → ∴ Array of size 7 → [3, 4, 5, 6, 7, 8, 9]

0	1	2	3	4	5	6
for [3]	for [4]	for [5]	for [6]	for [7]	for [8]	for [9]

farr [frequency Array]

- Initially (farr) me har element ki frequency ZERO hai,
  hum main array ko scan krengy aur jo jo element ayega uski frequency increase krdengy!

0 [3]	1 [4]	2 [5]	3 [6]	4 [7]	5 [8]	6 [9]
~~0~~	~~0~~	~~0~~	~~0~~	0	~~0~~	~~0~~
~~1~~	~~1~~	~~1~~	~~1~~		1	~~1~~
~~2~~	2	2	~~2~~			~~2~~
3			3			~~3~~
						4

kese Frequency increase kri ?
Humne yeh formula use kiya!

$$\boxed{\text{Element in main array we encountered} - \text{minimum value} = \text{Index of (farr) where we have to increase Frequency}}$$

9 – 3 = 6th index of farr
6 – 3 = 3rd ______
3 – 3 = 0th ______
⋮ ⋮ ⋮
9 – 3 = 6th ______

→ In Indexes pe increase hogi!

* farr

3	2	2	3	0	1	4
0	1	2	3	4	5	6

3 →	3	baar aya
4 →	2	______
5 →	2	______
6 →	3	______
7 →	0	______
8 →	1	______
9 →	4	______

→ siddha siddha likh sakty thay par nahi kra
↓
Kyu?

KYUNKI Agar essa kiya toh counter sort UNSTABLE Hoga! Aur counter sort ek STABLE sort algorithm hai

* Hum kuch alag krengy Humare pass ek main array, ek farr hai aur ab hum ek ans array banayengy!

ans | | | | | | | | ...

* Aab Hum farr par loop lagayengy aur PREFIX-SUM ARRAY Bana dengy!

farr

3	2	2	3	0	1	4

↓ converted into prefix-sum array

3	5	7	10	10	11	15
↓	↓	↓	↓	↓	↓	↓
1	2	3	4	5	6	7

prefix sum = prev element + curr. element

STABLE SORT

10 (a) 20 (b) 20 (c) 30 (d) 10 (e)
↓
10 (a) 10 (e) 20 (b) 20 (c) 30 (d)

This sorting is stable becoz the order of balls is maintained. smaller value first with maintaining the order of Alphabets

1 → 3 position tak 3 ayega
2 → 5 position tak 4 ayega
3 → 7 ______ 5 ______
4 → 10 ______ 6 ______
5 → 7 ayega hi nahi
6 → 11 position tak 8 ayega
7 → 15 ______ 9 ______

$\underbrace{3_1\ 3_2\ 3_3}\ \underbrace{4_4\ 4_5}\ \underbrace{5_6\ 5_7}\ \underbrace{6_8\ 6_9\ 6_{10}}\ \underbrace{8_{11}}\ \underbrace{9_{12}\ 9_{13}\ 9_{14}\ 9_{15}}$

* Hum Pre-fix Sum array me se 1 minus krdengy!

[3]	[4]	[5]	[6]	[7]	[8]	[9]
2	4	6	9	9	10	14

Farr (FINAL)

9	6	3	5	3	4	3	9	6	4	6	5	8	9	9
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

$\xrightarrow{3}_{0-2}\ \xrightarrow{4}_{3-4}\ \xrightarrow{5}_{5-6}\ \xrightarrow{6}_{7-9}\ \xrightarrow{8}_{10}\ \xrightarrow{9}_{11-14}$

* Aab Hum main array me reverse order me loop chalaengy! Aur jo bhi element encounter hoga uski position farr me dekhengy aur ans array me farr wala index pe uss element ko rakh dengy aur uss element ke index ko -1 krdengy farr me!

main array

9	6	3	5	3	4	3	9	6	4	6	5	8	9	9
0	1	2	3	04	05	06	07	08	9	10	11	12	13	14

ans array

3	3	3	4	4	5	5	6	6	6	8	9	9	9	9
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

farr

[3]	[4]	[5]	[6]	[7]	[8]	[9]
~~2~~	~~4~~	~~6~~	~~9~~	9	~~10~~	~~14~~
~~1~~	~~3~~	~~7~~	~~8~~		9	~~13~~
~~0~~	2	~~6~~	~~7~~			~~12~~
-1		5	6			~~11~~
						10

prefix-sum array ke ander elements ki last position thi!

```
public static void countSort (int[] arr, int min, int max){
  int range = max - min + 1;
  int[] ans = new int[arr.length];
   // make frequency arr
   int[] farr = new int[range];
   for( int i=0; i< arr.length; i++){
       farr[arr[i] - min] ++;
   }
   // convert it into prefix sum array
   for( int i = 1; i< farr.length; i++){
        farr[i] += farr[i-1];
   }
   // stable Sorting (filling ans array)
   for (int i = arr.length -1; i >=0; i--){
      int pos = farr[arr[i]-min]-1;
      ans[pos] = arr[i];
      farr[arr[i]- min]--;
   }
   //filling original array with the help of ans array
   for(int i=0; i<arr.length; i++){
        arr[i] = ans[i];
   }
}
public static void print (int[] arr){
   for(int i=0; i<arr.length; i++){
        System.out.println(arr[i]);
   }
}
```

```
public static void main (String [] args) {
  Scanner s = new Scanner (System.in);
  int n = s.nextInt ();
  int [] arr = new int [n];
  int max = Integer.MIN_VALUE;
  int min = Integer.MAX_VALUE;
  for (int i = 0; i < n; i++) {
      arr [i] = s.nextInt();
      max = Math.max (max, arr[i]);
      min = Math.min (min, arr [i]);
  }
  countSort (arr, min, max);
  print (arr);
}
```

<u>Time Complexity</u>

we travelled arr[] and farr[] 2 times

$(n+k) + (n+k) = 2(n+k)$

$\therefore \boxed{O(n+k)}$

<u>Space Complexity</u>

$\boxed{O(n+k)}$