PES UNIVERSITY

Text Book: Introduction to the Design and Analysis of Algorithms
Author: Anany Levitin 2 nd Edition

Unit-4

2. Distribution Counting Sort

1. **Distribution Counting Sorting**
   - Suppose the elements of the list to be sorted belong to a finite set (aka domain).
   - Count the frequency of each element of the set in the list to be sorted.
   - Scan the set in order of sorting and print each element of the set according to its frequency, which will be the required sorted list.

Consider sorting the array

13 11 12 13 12 12

whose values are known to come from the set {11, 12, 13} and should not be overwritten in the process of sorting. The frequency and distribution arrays are as follows:

Array values **11 12 13**
Frequencies **1 3 2**
Distribution values **1 4 6**

Note that the distribution values indicate the proper positions for the last occurrences of their elements in the final sorted array. If we index array positions from 0 to $n - 1$, the distribution values must be reduced by 1 to get corresponding element positions.

It is more convenient to process the input array right to left. For the example, the last element is 12, and, since its distribution value is 4, place this 12 in position 4 − 1= 3 of the array *S* that will hold the sorted list. Then decrease the 12's distribution value by 1 and proceed to the next (from the right) element in the given array. The entire processing of this example is depicted in Figure below,

	D[0..2]		
A [5] = 12	1	**4**	6
A [4] = 12	1	**3**	6
A [3] = 13	1	2	**6**
A [2] = 12	1	**2**	5
A [1] = 11	**1**	1	5
A [0] = 13	0	1	**5**

S[0..5]					
			12		
		12			
					13
	12				
11					
				13	

ALGORITHM *DistributionCountingSort*$(A[0..n-1], l, u)$

```
//Sorts an array of integers from a limited range by distribution counting
//Input: An array A[0..n − 1] of integers between l and u (l ≤ u)
//Output: Array S[0..n − 1] of A's elements sorted in nondecreasing order
for j ← 0 to u − l do D[j] ← 0                          //initialize frequencies
for i ← 0 to n − 1 do D[A[i] − l] ← D[A[i] − l] + 1     //compute frequencies
for j ← 1 to u − l do D[j] ← D[j − 1] + D[j]            //reuse for distribution
for i ← n − 1 downto 0 do
    j ← A[i] − l
    S[D[j] − 1] ← A[i]
    D[j] ← D[j] − 1
return S
```

Assuming that the range of array values is fixed, this is obviously a linear algorithm because it makes just two consecutive passes through its input array *A*. This is a better time-efficiency class than that of the most efficient sorting algorithms like mergesort, quicksort, and heapsort. It is important to remember, however, that this efficiency is obtained by exploiting the specific nature of inputs for which sorting by distribution counting works, in addition to trading space for time.