Implementing and Analyzing an Efficient Version of Counting Sort (E-Counting Sort)

Keshav Bajpai
Krishna Institute of Engineering and Technology
Ghaziabad, Uttar Pradesh, India

Ashish Kots
Krishna Institute of Engineering and Technology
Ghaziabad, Uttar Pradesh, India

ABSTRACT
Counting sort is a simple, stable and efficient sorting algorithm with linear running time, which is a fundamental building block for many applications. Counting sort algorithm has been widely used in data processing systems, because of its high efficiency, fast speed and stable nature. Therefore, a thorough study of its time complexity is required. This paper presents a modified version of counting sort E-Counting Sort with some efficiency improvements. An analysis of the E-Counting sort algorithm in comparison with original counting sort algorithm clearly shows that E-Counting sort algorithm having a reduced time complexity approximately to half of the original one. The new version can be applied to many real world applications providing the required result as efficient and as effective as original counting sort without affecting its real nature and improve the efficiency of application program.

General Terms
Algorithms, Sorting

Keywords
Counting Sort, Algorithm, Efficiency, Running Time.

1. INTRODUCTION
Counting sort [1,2] is a linear time sorting algorithm used to sort items when they belong to a fixed and finite set. Counting sort [3] is an efficient algorithm that assumes that all elements to be sorted are of type integer in the range 1 to k, where k is some other integer. It assumes that each of the elements is an integer in the range 1 to k, for some integer k. When k = O(n), the Counting-sort runs in O(*n*) time. The basic idea of counting sort is to determine, for each input elements *x*, the number of elements less than *x*. This information can be used to place the elements directly into its correct position. For example, if there are 17 elements less than *x*, than x belongs to position 18 in the output.

The basic thoughts of counting sort are:

- The algorithm proceeds by defining an ordering relation between the items from which the set of sorted elements is to be derived (for a set of integers, this relation is trivial). Let the set to be sorted be called A.
- An auxiliary array with size equal to the number of items in the superset is defined, say B.
- For each element in A, say *e*, the algorithm stores the number of items in A smaller than or equal to *e* in B(*e*).

If the sorted set is to be stored in an array C, then for each *e* in A, taken in reverse order, C[B[*e*]] = *e*. After each such step, the value of B(*e*) is decremented.

```
COUNTING_SORT (A, B, k)

1.      for i ← 1 to k do
2.          c[i] ← 0
3.      for j ← 1 to n do
4.          c[A[j]] ← c[A[j]] + 1
5.      for i ← 2 to k do
6.          c[i] ← c[i] + c[i-1]
7.      for j ← n down to 1 do
8.          B[c[A[i]]] ← A[j]
9.          c[A[i]] ← c[A[j]] - 1
```

Fig.1 Counting Sort Algorithm

2. E-COUNTING SORT ALGORITHM

```
COUNTING_MODIFY_SORT (A, B, k)

1.      for i← 1 to k
2.          c[i] ←0;
3.      for i← 1 to n
4.          c[a[i]]=c[a[i]]+1;
5.      for i← 1 to k
6.          while(c[i]>0)
7.              a[j]=i;
8.              j=j+1;
9.              c[i]=c[i]-1
```

Fig.2 E-Counting Sort Algorithm

3. TIME COMPLEXITY ANALYSIS
a. All ANALYSIS OF ORIGINAL ALGORITHM[3]:

1. The loop of lines 1-2 takes Θ (*k*) time
2. The loop of lines 3-4 takes Θ (*n*) time
3. The loop of lines 6-7 takes Θ (*k*) time
4. The loop of lines 9-11 takes Θ (*n*) time

Therefore, the overall time of the counting sort is:

$$\Theta(k) + \Theta(n) + \Theta(k) + \Theta(n) = \Theta(\boldsymbol{k+n})$$

In practice, we usually use counting sort algorithm when have $k = O(n)$, in which running case time is $\boldsymbol{\Theta(n)}$.

b. ANALYSIS OF MODIFIED ALGORITHM:

1. The loop of lines 1-2 takes Θ (*k*) time
2. The loop of lines 3-4 takes Θ (*n*) time
3. The loop of line 5 takes Θ (*k*) time

Analysis of While loop of lines 6-9 ,if all the elements are distinct then lines 6-9 takes O(1) time and if all the elements are same then while loop of line 6 repeats Θ(n) times but then the value of k=1,so the line 5 takes O(1) times.

4. PERFORMANCE COMPARISION FOR BOTH THESE ALGORITHM

To analysis the performance of both algorithms following configuration is used:

Table 1 Machine Configuration used for Analysis

Device	LENOVO L412 ThinkPad
Processor	Intel(R) Core(TM) i3 CPU
Installed Memory (RAM)	2.00GB (1.86GB Usable)
Operating system:	Windows 7
System Type:	64-Bit OS

Both the algorithms are executed multiple times on the above configuration using C compiler. E-Counting sort have shown a consistent performance irrespective of the number of elements. E-Counting sort reduces the running time of the original counting sort algorithm almost to the half. The number of inputs and time taken by both the algorithms is shown in table below.

Table 2 Comparative Analysis of Counting sort and E-Counting Sort

NO. OF INPUTS(INTEGER)	**TIME TAKEN in (in ms)**	
	ORIGINAL (**Counting Sort**)	MODIFIED (**E-Counting Sort**)
1000	0.010000	0.00000
2000	0.109890	0.054945
3000	0.164835	0.087912
4000	0.219780	0.109890
5000	0.274725	0.153846
6000	0.296703	0.164835
7000	0.360725	0.197802
8000	0.382637	0.219780
9000	0.439560	0.241758
10000	0.472527	0.274725

A graphically representation of the running time of both the algorithms is shown in figure below. It is clear from the figure that in comparison to Courting sort E-Counting sort requires fewer time frames for the same number of elements.

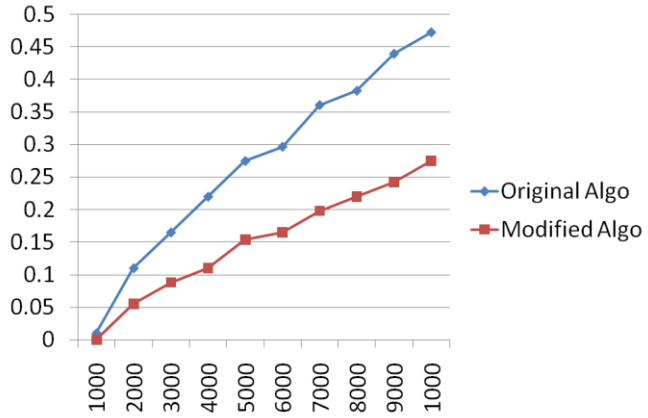

Fig.3 Number of elements vs. time taken by Counting Sort and E-Counting Sort

5. CONCLUSION

Counting sort is one of the sorting algorithms that is widely used for the GNU computing. The updated version will improve the effectiveness of GNU computing. The E-Counting sort takes less memory space and requires lesser instructions for execution. It can be ultize in the all the applications of courting sort.

6. REFERENCES

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Introduction to Algorithms*

[2] www.cse.iitk.ac.in/teaching/courses/CS210.html.

[3] M R Babu, M Khalid ," Performance Analysis of Counting Sort Algorithm using various Parallel Programming Models" International Journal of Computer Science and Information Technologies, Vol.2 (5), 2011