Hit Sort: A New Generic Sorting Algorithm

Hitesh Nagpal

Department Of Computer Engineering/ Pillai's Hoc College of Engineering & Technology/
Mumbai University, A-9, Shanti Apt, Nr. Kalan Samaj, Panvel-410206, Maharashtra, India
hitesh.nagpal94@gmail.com

ABSTRACT

This paper gives a new alternate sorting technique, "Hit Sort" which has exclusively been developed to sort the number of elements. In computer science and mathematics, a sorting algorithm is an algorithm that puts elements of a list in a certain order, not necessarily in increasing order, it may be in decreasing order as well. Efficient sorting is important to optimizing the use of other algorithms that require sorted lists to work efficiently. This algorithm works on the basis of counting the elements smaller than the index element after 1st pass the index element hits the position by count of the number smaller than that. A maximum of n-1 passes are executed to sort n elements offering the complexity of O(n^2). Under our numerical observations, discussions and conclusions, it has been revealed that the "Hit Sort" technique is an efficient and stable sorting technique for sorting versatile data.

Keywords — Sorting, Algorithm, Comparison.

1. INTRODUCTION

Hit Sort is the simple and spontaneous sorting algorithm that starts at the beginning of the data set. The task of sorting is standard algorithmic benchmark. . There are many sorting techniques available like "Selection Sort", "Bubble Sort", "Merge Sort", "Quick Sort", "Heap Sort", "Bin Sort", "Radix Sort"[4]. Several researchers confined their attention in this connection and explored number of different sorting algorithms. However, these sorting algorithms believe on keys being of a special type i.e. a type with a limited set of values[1]. All programs use a capability of programming languages and machines that is much more powerful than a simple comparison of values, namely the ability to find in one step a location in an array, given the index of that location [2].
"Hit Sort" is exclusively developed for sorting the number of elements in a new way. This algorithm differs from all the other existing sorting techniques available. It works on the method of counting. In the initial pass in the algorithm involves scanning elements in the left-to-right direction pair-wise all elements are compared for counting the elements smaller than the index element after that index element hits the (index + count)th elements are swapped. The index position remains first till the count for its position element becomes zero. Once the count becomes zero then the index is incremented by 1.The following passes involves same procedure till the index becomes equal to n and all the elements are sorted in order with the last element in the sequence. In developing a new sorting algorithm, for time and space –efficiency, it is imperative to give more effort to reduce the number of the design iterations. A maximum of n-1 passes are executed to complete the process of sorting n elements. The time to execute the sorting a set of elements using this approach evidently increases with the number of elements. It changes elements positions to hit directly to their destination. It is highly advantageous if changing positions of elements is costly in a system. Here, a temporary array of double length of the list to be sorted is used. So the space complexity is 2n or O (n).
The time complexity of algorithm is O (n^2). The build environment of the algorithm is built using the C language. The following will represent the algorithm as a way to sort an array or integers and run random trails of length. The research will provide the number of swaps and the runtime of the algorithm.

2. ALGORITHM

There are some obvious improvements to the foregoing method. Considering index element as i. First since all the elements in position greater than or equal than the ith element-after first iteration, it should not be considered in succeeding iteration. Thus on the first pass n-1 comparisons are made, on the another passes as the value of the i increases the comparisons reduces as (n-1), (n-2), (n-3)….1. Therefore the process speeds up as it proceeds through successive passes.

The hit sorting algorithm is essentially divided into two parts. First, we count the no. of the values greater than the ith element. Second step, we sort the array according to the count of the particular array values.

2.1. Counting the No. of Elements Greater than the ith Element.

```
1.  Count(i, a[],n)
2.  {
3.  Initialize j=cn=0
4.  for j=i+1 to n do
5.  {
6.  If a[i] < a[j] then
7.  increment cn
8.  }
9.  return cn
10. }
```

In the second Algorithm we sort the elements according to the count, if the count is 0 then i is incremented and if i > 0 then the ith element is hit on the element sitting at the count position of the array and both are swapped.

2.2. Sorting Array

```
1.  Hitsort(n,a[])
2.  {
3.  i=0
4.  for j=0 to 2*n do
5.  {
6.  cn=Count(i,a,n)
7.  if cn=0 then
8.  increment i
9.  if i=n then
10. break
11. else
12. swap ith and (i + count)th element
13. }
14. }
```

3. STABILITY OF ALGORITHM

Stable sorting algorithms maintain the relative order of records with equal keys. A key is that portion of the record which is the basis for the sort; it may or may not include all of the record [4]. If all keys are different then this distinction is not necessary. But if there are equal keys, then a sorting algorithm is stable. , i.e a sorting algorithm is stable if whenever there are two records M and N with the same key and with M appearing before N in the original list, M will appear before N in the sorted list[5]. Hit Sort being data dependent is Stable.

4. TIME COMPLEXITY

The complexity of the proposed sorting technique, “Hit Sort” algorithm if is used to sort n elements is observed as O(n2).Time Complexity remains same for all cases best, average and worst.

5. FLOWCHART

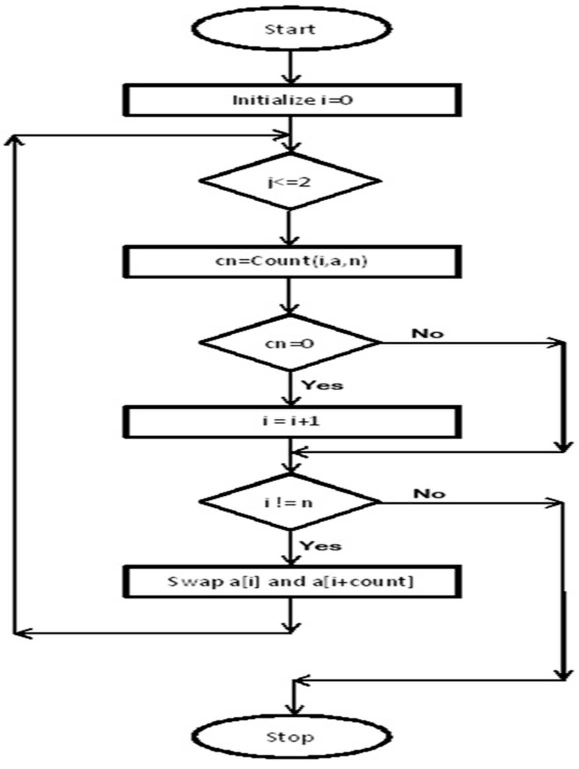

Fig-1: Flowchart Based on the given Algorithm.

6. APPLICATION OF PROPOSED SORTING ALGORITHM

Suppose five elements are taken, which are as follows:

5 8 2 12 4

For these above five elements, position of the each element in the array after each pass will be as follows:

```
Index(i)   0    1    2    3    4
i=0        5    8    2    12   4
     count=2
i=0        2    8    5    12   4
     count=0 then i++
i=1        2    8    5    12   4
     count=2
i=1        2    12   5    8    4
      count=3
i=1        2    4    5    8   12
     count=0 then i++
i=2        2    4    5    8    12
     count=0 then i++
i=3        2    4    5    8    12
     count=0 then i++
i=4        2    4    5    8    12
      i=n then stop
```

Hence the Final Sorted Array.

7. ANALYSIS BASED ON THE OBSERVED RESULT

No. of Input(n)	Execution Time	No. of Input(n)	Execution Time
0	0.001	10	0.04
1	0.001	20	0.01
2	0.002	40	0.02
3	0.002	80	0.037
4	0.003	100	0.048
5	0.003	150	0.054

Table-1: Data on number of inputs and respective execution time.

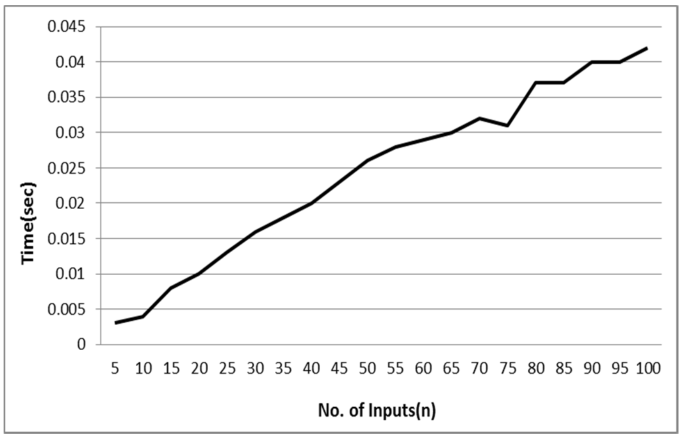

Fig-2: Inputs Vs Execution Time.

8. EXISTING TECHNIQUES

8.1. Bubble Sort

It is a Simple method of sorting data that is used in computer science. It compares the first two elements, if the first element is greater than the second, it swaps them. This process is repeated for each pair of adjacent elements of the end of the data set. It then starts again with the first two elements, repeating until no swaps have occurred on the last pass [3].

8.2. Insertion Sort

It is a simple sorting algorithm that is relatively efficient for small lists and mostly-sorted lists, and often is used as part of more sophisticated algorithms. It works by taking elements from the list one by one and inserting them in their correct position into a new sorted list. In arrays, the new list and the remaining elements can share the array's space, but insertion is expensive, requiring shifting all following elements over by one [8].

8.3. Selection Sort

It is also known as comparison sort. It is inefficient on large lists, and generally performs worse than the similar insertion sort. Selection sort is mainly known for its simplicity, and also has good performances over more complicated algorithms [8].

8.4. Quick Sort

It depends on the divide and conquer strategy. Array is partitioned with the help of the pivot element. All the smaller and greater elements are moved before and after the pivot respectively and then recursively sort the lesser and greater sub lists. It has complexity of O(nlogn)[7].

8.5. Merge Sort

It is a comparison based algorithms and in most implementations it is stable. It uses the approach of the divide and conquers [8].

8.6. Cocktail Sort

It is also known as bidirectional bubble sort. It is a stable algorithm and a comparison based sort. The algorithm sorts in both directions each pass through the list [8].

9. COMPARISON

Name	Average Case	Worst Case	Stable
Bubble Sort	$O(n^2)$	$O(n^2)$	Yes
Insertion Sort	$O(n^2)$	$O(n^2)$	Yes
Merge Sort	O(n logn)	O(n logn)	Yes
Quick Sort	O(n logn)	$O(n^2)$	No
Cocktail Sort	-	$O(n^2)$	Yes
Selection Sort	$O(n^2)$	$O(n^2)$	No
Hit Sort	**$O(n^2)$**	**$O(n^2)$**	**Yes**

Table-2: Time Complexity [6][8]

10. CONCLUSION

In this paper, a sorting technique, "Hit Sort" has been proposed for a versatile data. Graphical representation of number of input(s) vs. execution time has been displayed in fig.a.The developed sorting technique, "Hit Sort" is sufficiently stable. From this point of view the suggested sorting technique is quite efficient with an overhead of adding a reasonably largest element in the case of even number of elements[4] and it also as it reduces the number of comparisons as well as number of swaps in comparison to the other algorithms[5].The Complexity of the algorithm does not changes with the input cases, complexity remains constant $O(n^2)$ for the worst case as well as best case the only advantage of best case, it requires less comparisons as compared to other cases. The algorithm is stable for all cases.

REFERENCES

[1] Md. Khairullah, "Enhancing Worst Sorting Algorithms" ,International Journal of Advanced Science and Technology,Vol. 56, July 2013.

[2] Jehad alnihoud and Rami Mansi, "An Enhacement of Major Sorting Algorithms", The International Arab Journal of Information and Technology, Vol. 7, Jan 2010.

[3] V.Mansotra and Kr.Sourabh, "Implementing Bubble Sort Using a New Approach", March 10, 2011.

[4] Dr.V.N.Maurya, Dr. R.K.Bathla, Diwinder Kaur Arora, Er. Avadesh Kumar Maurya and Ram Asrey Gautam, "An Alternate Efficient Sorting Algorithm Applicable for Classification of Versatile Data", International Journal of Mathematical Modeling and Applied Computing, Vol. 1, April 2013.

[5] Surmeet Kaur, Tarundeep Singh Sodhi and Parveen Kumar, "Freezing Sort", International Journal of Applied Information Systems ,Vol. 2-No.4, May 2012.

[6] Sonal Beniwal and Deepti Grover, "Comparison Of Various Sorting Algorithms: A review", International Journal of Emerging Research in Management &Technology, Vol. 2-5, May 2013.

[7] Pankaj Sareen, "Comparison of Sorting Algorithms (On the Basis of Average Case)", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3-3, March 2013.

[8] Jariya Phongsai, "Research Paper on Sorting Algorithms", Prof. Lawrence Muller, Oct 26, 2009.