ISSN-2394-5125 VOL 5, ISSUE 03, 2018

Positional Approach for Alphabetic Sort Algorithm

Dr. Sunil Mishra (Cse) [1]*, Dr. Yogesh Bhomia (Ec)[2], Mr. Ravin Kumar(Cs)[3], Mr. Nawneet Panday (Cs) [4]

[1]*,[2,3,4] Accurate Institute Of Management & Technology, Greater Noida

***Corresponding Author:** Dr. Sunil Mishra
*Accurate Institute Of Management & Technology, Greater Noida

Abstract-
In This paper, the authors present a positional algorithmic approach for alphabetic sort. Results are achieved in linear time. Within this approach two embedded algorithms,Binary Search and Counting Sort are executed in parallel to achieve the goal. In this approach a Pre-Processor or Priority Queue is used, which minimizes time complexity. The algorithm is linear in speed. Time Complexity of this newly proposed algorithm is Θ (n). The interesting feature of this algorithm is that the order of alphabets is not change and the approach is too much simpler

Keywords- Algorithm, Priority Queue, Sort, Search, Complexity, Analysis.

I. INTRODUCTION

Algorithm is any well-defined computational procedurethat takes some value, or set of values, as input and produces some value, or set of values, as output. Analgorithm is thus a sequence of computational steps thattransform the input into the output[1][2]. Algorithm is a toolfor solving a well-specified computational problem. Thealgorithm describes a specific computational procedure forachieving the input/output relationship [1][2][3][9].

An algorithm is an orderly systematic procedure to solve a problem. The term algorithm is derived from the title Khowrizmi of ninth-century Persian mathematician Abu Musa al-Khowrizmi, who is credited with systematic study and development of important algebraic procedures. An algorithm is a sequence of unambiguous instructions for solving a problem in a finite amount of time.

A large variety of problems in computer science, mathematics and other disciplines depend on the use ofalgorithms for their solutions. The broad categories of applications types are:

i. Searching Algorithms (Linear and non-linear)
ii. Sorting Algorithms (Elementary and Advanced)
iii. Strings Processing (Pattern matching, Parsing,Compression, Cryptography)
iv. Optimization Algorithms(Shortest routes,minimum cost)
v. Geometric Algorithms(Triangulation, ConvexHull)
vi. Image Processing (Compression, Matching,Conversion)
vii. Data Mining Algorithms(Clustering, Cleansing,Rules mining)
viii. Mathematical Algorithms (Random numbergenerator, matrix operations, FFT, etc)

II. BINARY SEARCH ALGORITHM

Binary Search is an algorithm for locating the position of an element in a sorted list by checking the middle, eliminating half of the list from consideration, and then performing the search on the remaining half. If the middle element is equal to the sought value, then the position has been found; otherwise, the upper half or lower half is chosen for search based on whether the element is greater than or less than themiddle element. The method reduces the number of elements needed to be checked by a factor of two each time,and finds the target value, if it exists in logarithmic time. A binary search is a divide and conquer search algorithm. [1][3][9][4][5].

Pseudo Code for Binary Search AlgorithmBINARY-SEARCH

```
i.    start = 1; end = n ;
ii.   while ( start < end )
iii.  {
iv.   middle = ( start + end ) / 2 ;
v.    if s > a middle  then start = middle + 1 ;
vi.   else end = middle – 1 ;
vii.  }
viii. if ( s == a start ) location = start ;
ix.   else location = 0;
```

Time & Space Complexities of Binary Search Algorithm:[3][4][5]

Worst case performance = O (log n)Best case performance = O (1)
Average case performance = O (log n)Worst case space complexity = O (1)

III.COUNTING SORT ALGORITHM

Counting Sort (sometimes referred to as Ultra Sort or Math Sort) is a sorting algorithm, which takes advantage of knowing the range of the numbers in the array to be sorted (array A). It uses this range to create an array C of this length. Each index i in array C is then used to count how many elements in A have the value i; then counts stored in C can then be used to put the elements in A into their right position in the resulting sorted array. The algorithm was created by Harold H. Seward [1][3][6].

Pseudo Code for Counting Sort Algorithm:COUNTING -SORT (A)

```
i.    n=length[A]
ii.   for j← 0 to n do
iii.  C[j]← 0
iv.   for j ← 1 to n do
v.    k← A[j]
vi.   C[k]← C[k] +1
vii.  for j ← 1 to n do
viii. C[j]← C[j] + C[j-1]
ix.   for j ← n downto 1 do
x.    i ← A[j]
xi.   k← C[i]
xii.  B[k]← A[j]
xiii. C[i] ← C[i]-1
xiv.  return B
```

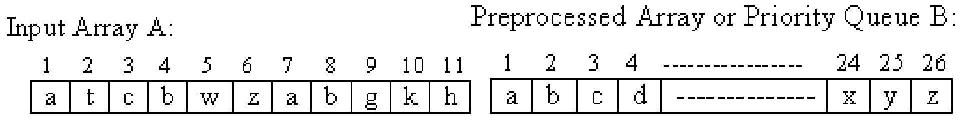
Time & Space Complexities of Counting Sort: [5][6][1]Worst case performance = O (n + k) Best case performance= O (n + k) Average case performance = O (n + k) Worst case space complexity = Θ (n + k)

IV. PROPOSED ALPHABETIC SORTALGORITHM

The proposed algorithm uses two arrays named as array A and array B. Array A is an input array while as Array B is a Pre-Processed array or Priority Queue. In the whole process two well-known algorithms, Binary Search and Counting Sort are used in parallel. The Binary Search algorithm is used for searching the corresponding positions while as Counting Sort algorithm is used for sorting purposes.
Steps involved in the proposed alphabetic sort algorithm.

i. Comparison is made between array A and pre- processed array B for searching the corresponding positions of alphabets in Priority Queue B. For this purpose Binary searchalgorithm is used.
ii. Substituting the alphabets in array A with corresponding positional digits.
iii. For sorting array A, Counting Sort algorithm is used.
iv. Again fetching corresponding elements of the positional digits held in Array A from Priority Queue B. So Binary Search algorithm is used.
v. Replacing the digits in the array A by the fetched elements.

V. GRAPHICAL DEPICTION OF THEPROPOSED ALGORITHM

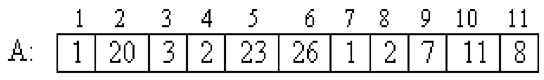

Input Array A:

1	2	3	4	5	6	7	8	9	10	11
a	t	c	b	w	z	a	b	g	k	h

Preprocessed Array or Priority Queue B:

1	2	3	4	----------------	24	25	26
a	b	c	d	--------------	x	y	z

Take alphabet from array A, fetch its corresponding location/position from Priority Queue B and then replace alphabet in array A by this fetched positional digit.


A:

1	2	3	4	5	6	7	8	9	10	11
1	20	3	2	23	26	1	2	7	11	8

Now sort this array A by using sorting algorithm. The sortedarray will be as,

	1	2	3	4	5	6	7	8	9	10	11
A:	1	1	2	2	3	7	8	11	20	23	26

Now again fetch corresponding elements of these new positional digits held in Array A from Priority Queue B. Replace the digits in this updated array A by the fetched elements. Now the array A will be as,

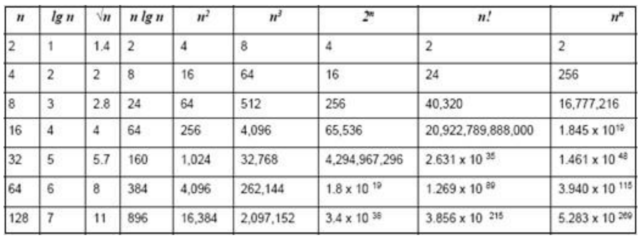
	1	2	3	4	5	6	7	8	9	10	11
A:	a	a	b	b	c	g	h	k	t	w	z

Sorted alphabets list.

Here a pre-processor is also introduced, due to which time complexity is minimized.

VI.ANALYSIS OF THE PROPOSED ALGORITHM

For the whole process twice Binary Search algorithm, oncesorting algorithm and twice substitution is used.Time Complexity of the proposed algorithm is,T(n) = Θ (lg n) + Θ (lg n) + Θ (n) = Θ (n)(Ignoring constant terms & values)



n	$lg\ n$	$\sqrt{n}$	$n\ lg\ n$	n^2	n^3	2^n	$n!$	n^n
2	1	1.4	2	4	8	4	2	2
4	2	2	8	16	64	16	24	256
8	3	2.8	24	64	512	256	40,320	16,777,216
16	4	4	64	256	4,096	65,536	20,922,789,888,000	1.845×10^{19}
32	5	5.7	160	1,024	32,768	4,294,967,296	2.631×10^{35}	1.461×10^{48}
64	6	8	384	4,096	262,144	1.8×10^{19}	1.269×10^{89}	3.940×10^{115}
128	7	11	896	16,384	2,097,152	3.4×10^{38}	3.856×10^{215}	5.283×10^{269}

The logarithm (lg n) has lower growth rate and exponentialfunction nn has the highest growth rate.The symbolic representation of the relationship of functionsgrowth rates is,

$$lg\ n < \sqrt{n} < n < n\ lgn < n^2 < n^3 < 2^n < n! < n^n$$

Therefore, Time Complexity of the proposed algorithm is Θ(n), Linear Time Complexity.

VII. CONCLUSION

The linear Time Complexity T(n) = Θ (n) is achieved by this algorithm. The flavor of this newly proposed algorithm is its speed and simplicity. The interesting feature here is that the order of alphabets is not changed.

VIII. REFERENCES

1. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford, Introductionto Algorithms (3rd ed.), MIT Press.
2. A.Levitin, Introduction to Design and Analysis of Algorithms, Pearson Edition, 2006.
3. R.E.Neapolitan, K.Naimipour, Fundamentalsof Algorithms, Heath and Company.
4. http://mathworld.wolfram.com/BinarySearch.html , 2009.
5. http://en.wikipedia.org/wiki/ Binary_search_algorithm , 2009.
6. Anthony Ralston, Edwin D. Reilly, David Hemmendinger, ed (2003). "Ultrasort". Encyclopedia of Computer Science (4th ed.). Wiley. pp. 1660-1661.
7. http://www.shannarasite.org/, 2009
8. D.F. Stubbs and N.W. Webre, Data Structures with Abstract Data Type and Ada, PWS- KENT Publishing Company, pp.301-341, 1993.
9. S.Basse, V.A.Gelder, Computer Algorithms,Pearson Edition Inc.