Improved Double Selection Sort using Algorithm

Nirupma Pathak*[1] & Shubham Tiwari[2]

1. * Assistant Professor, Computer Science and Engineering, S.R. Institute of Management & Technology, Lucknow, (U.P.), India
e-mail : nirupmapathak@gmail.com
2. B.Tech Student, Computer Science and Engineering, S.R. Institute of Management & Technology, Lucknow, (U.P.), India.
e-mail : shubham.tiwari540@outlook.com

Publication Info

Article history :

Received : 11th Nov. 2017

Accepted : 05th Dec. 2017

DOI : 10.18090/samriddhi.v9i02.10866

Keywords :

Selection sorting technique, Sorting Algorithm, NlogN sort.

***Corresponding author :**

Nirupma Pathak

e-mail : nirupmapathak@gmail.com

Abstract

In this paper, we present the work regarding the selection sorting technique for double ended selection sort. This sorting algorithm is both theoretical and programmatically analysis show that the introduce advance selection sort algorithm which enhances the performance of selection sort. It is much faster than the selection sort because of its selection of minimum and maximum elements simultaneously. Advance selection sort algorithm possibility of enhancing execution speed up to 30%. Code for this algorithm is written in C programming Language. So easy to understand the concept of this sorting algorithm by everyone because C is the popular language. Results and discussion show a higher level of performance for the sorting algorithm. It can theoretically prove that the algorithm can reduce steps with the selection short and will improve N2 sorts toward NlogN sort.

1. INTRODUCTION

The phenomenon of selection sorting has been known for a long time and has been used for the database management system. Selection shorting is the most important method used to arrange data according to user requirement. In object-oriented software, represent the sorting by programming [1]. The practical work in sorting has so far, generally focused on ascending or descending order sequence. The important issue of sorting data has successfully addressed in this paper. The automated tool, such as Turbo C, has raised awareness of the importance of shoring of data. Many sorting algorithms are available in the literature. Sorting is a data structure operation, which is used for making searching and arranging of data item or record. The here arrangement of sorting involves either into ascending or descending order. Everything in this world has some advantage and disadvantage, some sorting algorithms are problem specific means they work well on some specific problem, not all the problem [2]. It saves time and helps searching data quickly. Sorting algorithm performance varies on which type of data being sorted, not easier to say that which one algorithm is better than another is. Here, the performance of the different algorithm is according to the data being sorted [3]. Some common sorting algorithms are the exchange or bubble sort, the selection sort, the insertion sort and the quicksort [4]. The Selection sort is a good one that uses for finding the smallest element in the array and put in the proper place. Swap it with the value in the first position [5]. Repeat until the array is sorted (starting at the second position and advancing each time). It is very intuitive and simple to program, offer quite good performance for particular strength being the small number of exchanges needed [6]. The selection sort always goes through a set number of comparisons, for a given no of data items. However, sometimes

question raise in front of us, is there any way through this sorting can be more effective and how to convert that algorithm into code [7, 8]. Then demonstrate a modification of this algorithm, and finally to assign the coding modification as a programming. This work introduces one new simple modification of sorting algorithm. The proposed algorithm of sorting algorithm is very simple and easy to implement in Turbo C tool, a well known and easy to operate software for the structural programmed implement. The theoretical and practical are successfully utilized for the short algorithm to evaluate the caused for a particular task and close agreement is revealed.

The organization of the paper is as follows: Section 1 covers the introduction to shoringalgorithm. The proposed advance selection sort algorithm is presented in Section 2. The proposed pseudo code of selection short is presented in Section 3. Finally, the paper is concluded with the summarization of all the contents in Section 4.

2. OUR PROPOSED ADVANCE SELECTION SORT ALGORITHM

In this section, one new advance selection sort algorithm has been introduced followed by the generalized pseudo code of proposed algorithm.

Advance selection sort algorithm is an internal sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into three parts, the sorted part at the left and right end and the unsorted part between left and right end. It selects the minimum and maximum element requires scanning all n elements (this takes n-1 comparison) and then swapping it into the first and last position respectively. Finding the next minimum and maximum element requires scanning the remaining (n-2) elements and so on. Practical analysis ofshort algorithm has been carried out Turbo C which works on the structural programming language. The results from the presented idea are in good agreement with the targeted results. Thus the developed advance selection sort algorithm is validated.

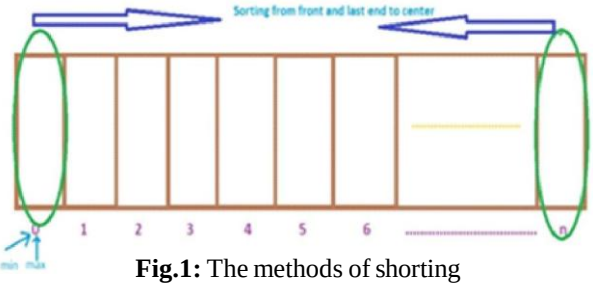

Fig.1: The methods of shorting

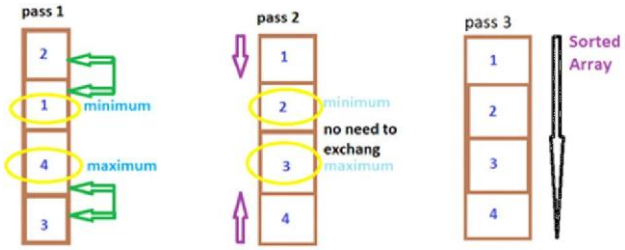

Fig.2: Shorting based on array

At this point, we have developed an advance short algorithm for the Database management system (DBMS) [10, 11, 12]. Combining 12 steps the short algorithm has been implemented. Algorithm 1 demonstrate the design procedure of the proposed idea of the sort algorithm. Line 1 of the algorithm assign array length. Line 6 checking the minimum array location. Line 8 of the algorithm check the maximum array location. Finally, Line 10 and 11, there is exchange position of array location [13].

Algorithm 1. Algorithm for the proposed idea of advance selection short.

1. n ← length[Array]
2. **for** i ← 0 to n-1
3. max ← i
4. min ← i
5. **for** j ← i+1 to n
6. **if** (Array[j] < min)
7. then min ← j
8. **if** (Array[j] > max)
9. then max ← j
10. Exchange (Array[i], Array[min])
11. Exchange (Array[n-1], Array[max])
12. n ← n-1

S-JPSET : Vol. 9, Issue 2, ISSN : 2229-7111 (Print) & ISSN : 2454-5767 (Online)

3. PROPOSED PSEUDO CODE OF ADVANCE SELECTION SORT ALGORITHM

In this section, we have presented the pseudo code of advance selection sort algorithm. To present the advance selection sort, we present the shorting methods in Section 2.

Algorithm 2 presents a formal representation of the proposed idea of advance selection sort algorithm based on C programming language. In algorithm 2, these outputs are taken from line 26. The time complexity of Algorithm 2 depends on the loops, data storage array, and temporary storage. Thus the complexity is O(n) where n is the size of the algorithm. This is the optimum number of size of shorting algorithm for the proposed idea as proved in the Algorithm 2. The details pseudo code of advance selection sort algorithm is presented below.

```
*  Pseudo code for advance selection short
1. #include<stdio.h>
2. #include<conio.h>
3. void main()
   {
4.  int a[100],n, i, j, f, min, max, temp;
5.   printf ("\n Enter the Number of Elements: ");
6.    scanf ("%d",&n);
7.    f=n;
   printf ("\n Enter %d Elements: ",n);
8.  for (i=0;i<n-1;i++)
  {
9.     Scanf ("%d",&a[i]);
  }
10.  For (i=0;i<n-1;i++)
  {
11.    min = i;
12.     max = i;
13.    for (j=i+1;j<n;j++)
       {
14.       If (a[min]>a[j])
15.       min=j;
16.      if (a[max]<a[j])
         max = j;
         }
17.   Temp = a[i];
18.  a[i] = a[min];
19.    a[min] = temp;
20.   temp = a[n-1];
21.   a[n-1] = a[max];
22.  a[max] = temp;
23.  n = n- 1;
  }
24.   Printf ("\n The Sorted array in ascending or
25.  For (i=0;i<f;i++)
{
26.      Printf ("%d ",a[i]);
         }
27.   Getch ();
         }
```

4. CONCLUSION AND FUTURE WORK

This paper presented the double selection-shorting algorithm. In the customized software development, shorting of array provide less time and low-cost development. The programmability of sorting algorithm is based on the algorithms. This saves the routine time of array shorting. In double selection, sort makes up the large number of sorts that is easy to understand and faster than selection sort. In double selection sort, the number of comparisons is reduced and double selection sort similar to selection sort. As a result, we have designed minimal steps based shorting methodology to make the algorithm faster and less time to compute. An optimal step is maintained to reduce the runtime and hence, enhancing the performance. Finally, this study reveals that it is an optimal algorithm and it applied to any system where the shorting is required. A possible future work of double shorting is the object-oriented based algorithm with fewer steps.

REFERENCES

[1] Yijie Han, "Deterministic sorting in O (n log n) time and linear space." Proceedings of the third-fourth annual ACM symposium on Theory of computing. ACM, 2002.

[2] R. B. Patel, and M. M. S. Rauthan. "Expert Data Structures with C++." (2000).

[3] Gray J Bronson, "Program development and design using C++". Brooks/Cole Publishing Co., 2000.

[4] D P Chavey, "Double sorting: testing their sorting skills." Proceedings of the 41st ACM technical symposium on Computer science education. ACM, 2010.

[5] T Tiwari, S. Singh, R Srivastava, and N. Kumar, (2009, August). A bi-partitioned insertion algorithm for sorting. 2nd IEEE International Conference on In Computer Science and Information Technology ICCSIT, pp. 139-143, 2009.

[6] Ming Zhou, and Hongfa Wang. "An efficient selection sorting algorithm for two-dimensional arrays.", Fourth International Conference on Genetic and Evolutionary Computing (ICGEC), 2010.

[7] Oyelami Olufemi Moses, "Improving the performance of bubble sort using a modified diminishing increment sorting." Scientific Research and Essays 4.8, pp. 740-744, 2009.

[8] Mikkel Thorup,. "Randomized sorting in O (nloglogn) time and linear space using addition, shift, and bit-wise boolean operations." Journal of Algorithms, 42.2, pp. 205-230, 2002.

[9] Arne Andersson et al. "Sorting in linear time." Proceedings of the twenty-seventh annual ACM symposium on Theory of computing. ACM, 1995.

[10] Paul Beame, and Faith E. Fich. "Optimal bounds for the predecessor problem and related problems." Journal of Computer and System Sciences 65.1, pp. 38-72, 2002.

[11] Genshiro Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models." Journal of computational and graphical statistics 5.1, pp. 1-25, 1996.

[12] Mikkel Thorup, "Undirected single-source shortest paths with positive integer weights in linear time." Journal of the ACM (JACM), 46.3, pp. 362-394, 1999.

[13] Nalluri, S. K., & Parasaram, V. K. B. (2015). Automating Software Builds with Jenkins: Design Patterns and Failure Handling. International *Journal of Technology, Management and Humanities*, 1(01), 16-33.

