International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 12 | Dec -2016 www.irjet.net p-ISSN: 2395-0072

A Comparative Study of Selection Sort and Insertion Sort Algorithms

Fahriye Gemci Furat

Research Assistant, Department of Computer Engineering, Iskenderun Technical University, Hatay, Turkey

---

Abstract - *The ordering of numbers in an integer array in computer science and mathematics is one of the research topics in the literature. For the purpose, there are a large variety of sorting algorithms like Selection sort, Insertion sort, Quick sort, Radix sort, Merge sort and Bubble sort. In this study, two sorting algorithms of these algorithms are investigated as Selection and Insertion sort. This study compares performance of Selection sort and Insertion sort algorithms that are used commonly in terms of running time.*

Key Words: Sorting Algorithm, Selection Sort, Insertion Sort, Time Complexity.

1. INTRODUCTION

The need for regular knowledge resulting from increased knowledge results in increasing development of data structures and algorithms. Sorting process in data structure is to make randomly distributed elements into elements in decreasing order or ascending order. Lots of sorting algorithms such as Selection sort, Insertion sort, Quick sort and Radix sort are developed in order to decrease complexity and increase performance of sorting.

In [1], the criterias that are given to compare performance of sorting algorithms are time efficiency, space efficiency, number of comparisons, number of data movements and stability of the sort technique. There is a lot of research in the literature such as [2, 3, 4, 5, 6, 10, 11, 12, 15]. In this study, time efficiency of these criterias is investigated to compare Selection sort and Insertion sort algorithms.

Performance of Selection sort and Shell sort is compared in terms of running time in [2]. Although shell sort shows better performance than selection sort, selection sort is more used because of its more simple structure. [2].

In [3], Enhanced Bubble sort and Enhanced Selection sort are explained. In addition, Selection sort, Enhanced Selection sort, Enhanced Bubble sort and Bubble sort are compared in terms of number of comparisons, swaps and time [3].

In [4], Grouping Comparison Sort algorithms (GCS) are introduced. Selection sort, Insertion sort, Merge sort, Quick sort, Bubble sort and GCS are compared in terms of time complexity. As a result of this comparison, Quick sort is the fastest and the selection sort the slowest for the large number of elements.

In [6], performance of Quick sort and Merge sort are compared in terms of time complexity. When number of elements is large, performance of Quick sort is better than Merge sort. In contrary, when number of elements is less, performance of Merge sort is better than Quick sort [6].

In [15], Quick sort for the large number of elements is the fastest algorithm, when compared to Quick sort, Selection sort, Insertion sort, Bubble sort, Shell sort and Cocktail sort.

In this study Selection sort and Insertion sort are compared in terms of running time. In chapter 2, Selection sort algorithm is explained. Work logic of this algorithm is explained with an example. Code of the algorithm is implemented using Java Programming Language. Running time for sorting using this algorithm is given. In chapter 3, Insertion sort algorithm is explained. Work logic of this algorithm is explained with an example. Code of the algorithm is implemented using Java Programming Language. Running time for sorting using this algorithm is given. In chapter 4, comparison of Selection sort and Insertion sort algorithm are given in terms of time complexity.

2. SELECTION SORT ALGORITHM

2.1 Selection Sort Algorithm

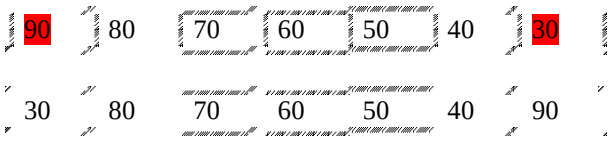
Selection sort is a simple comparison based sorting algorithm. Selection sort algorithm starts to finds the smallest element in the array. Then it exchanges this smallest element with the element in the first position. After this first step, this algorithm tries to select a smallest element in unsorted part of the array in each step of the sort. It exchanges this selected smallest element with the element in the unsorted part of this step of the sort. Until there are no unsorted elements in the array, this process continues. Selection sort algorithm spends most of its time to find the smallest element in the unsorted part of the array [1, 2, 3, 7, 8, 13, 15].

2.2 Work Logic of Selection Sort Algorithm on a Sample

Suppose that n is the number of elements in the array, the below integer array with 8 elements is given, so n number is 8 for this sample and sorting of this integer array in ascending order is wanted:

1.Step

In this step, the smallest element in the array is found and exchanges with element in the first index.

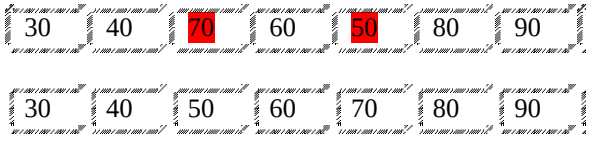


2.Step

In this step, the second smallest element in the array is found and exchanges with element in the second index. In addition, the second smallest element in the array is the smallest element of the unsorted part of array.

3.Step

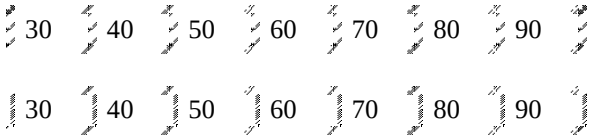
In this step, the third smallest element in the array is found and exchanges with element in the third index.



After the 3. step for above sample, in fact the integer array is sorted in ascending order. Although the array is sorted after the 3. step, the selection algorithm continues to search small numbers as 4. step, 5.step,..., n. step. In some studies like [3], this process finishes when the array is sorted. But it is not correct, because of not checked whether is sorted all integer numbers of array.

The selection sort algorithm continues to search small elements until the cycles are finished. So selection algorithm can improve to control whether the array is sorted. General Selection sort algorithm steps continue in the following:

4.Step



...

2.3 Implementation of Selection Sort Algorithm Using Java Programming Language

A code of selection sort algorithm with Java programming language is shown below:

```
public static int [] selectionsort(int [] A,int n)
        {
          int tmp;
          int min;

          for(int i=0; i< n-1; i++)
          {
            min=i;

                for(int j=i; j<n; j++)
                {
                  if (A[j]<A[min]){

                      min=j;
                                  }
                }

               tmp=A[i];
               A[i]=A[min];
               A[min]=tmp;

          }
          return A;
        }
```

2.4 Selection Sort Algorithm Running Time

Table -1: Running time to sort arrays using Selection sort

Number of elements	*Full sorted array*	*Semi sorted array*	*Unsorted array*
1,000	**0.003 s**	**0.002**	**0.001 s**
10,000	**0.103 s**	**0.098 s**	**0.1 s**
100,000	**10.233 s**	**10.455 s**	**9.621 s**

3. INSERTİON SORT ALGORİTHM

3.1 Insertion Sort Algorithm

Insertion sort is a simple comparison based sorting algorithm. Insertion sort algorithm starts to compare the first two elements in array. If the first element is bigger than the second element, they are exchanged with each other. This process is implemented for all neighbour indexed elements [7, 8, 14, 15].

3.2 Work Logic of Insertion Sort Algorithm on a Sample

Suppose that n is the number of elements in the array, the below integer array with 8 elements is given, so n number is 8 for this sample and sorting of this integer array in ascending order is wanted:

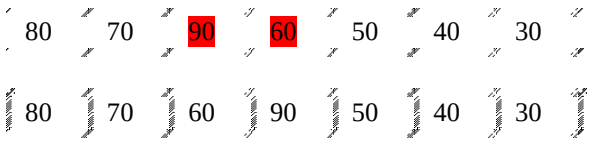
1.Step

In this step, the element in the first index in the array exchanges with element in the second index, if the element in the second index are smaller than the element in the first index.

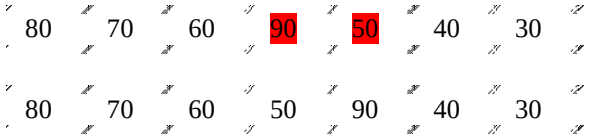
2.Step

In this step, the element in the second index in the array exchanges with element in the third index, if the element in the third index are smaller than the element in the second index.

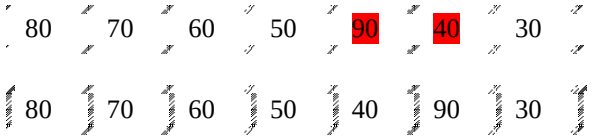
3.Step



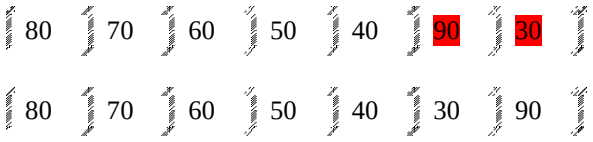
4.Step



5.Step



6.Step



3.3 Implementation of Insertion Sort Algorithm Using Java Programming Language

```
Public static int [] insertionSort(int [] A,int n)
{
            int tmp;

    for(int i=1; i<n; i++) {

       for(int j=i; j > 0 && A[j] < A[j-1]; j--) {

                    tmp=A[j];
                    A[j]=A[j-1];
                    A[j-1]=tmp;
                                                    }
                                }
          return A;
}
```

3.4 Insertion Sort Algorithm Running Time

Table -2: Running time to sort arrays using Insertion sort

Number of elements	*Full sorted array*	*Semi sorted array*	*Unsorted array*
1,000	**0 s**	**0.002 s**	**0.002 s**
10,000	**0 s**	**0.099 s**	**0.127 s**
100,000	**0 s**	**9.423 s**	**12.449 s**

4. COMPARISON OF SELECTION SORT AND INSERTION SORT ALGORITHM

In this study, nine different integer arrays are sorted using Selection sort and Insertion sort algorithms as three of the arrays with 1,000 elements, three of the arrays with 10,000 elements and three of the arrays with 100,000 elements. Three arrays having 1,000, 10,000 and 100,000 elements, that are called full sorted array consists of sequential numbers in ascending order from 1 to 1,000, from 1 to 10,000 and from 1 to 100,000, respectively. Three arrays having 1,000, 10,000 and 100,000 elements, that are called semi sorted array consists of sequential numbers in ascending order from 1 to 499 and in descending order from 500 to 1, from 1 to 4,999 and in descending order from 5,000 to 1, and from 1 to 49,999 and from 50,000 to 1, respectively. Remaining three arrays having 1,000, 10,000 and 100,000 elements that are called unsorted array consists of sequential numbers decreasing from 1,000 to 1, from 10,000 to 1 and from 100,000 to 1, respectively. In Table-3, results of these sorting processes are given in terms of running time.

Insertion sort algorithm sorts three full sorted arrays having 1,000 elements, 10,000 elements and 100,000 elements in 0 second, while Selection sort algorithm sorts full sorted arrays having 1,000 elements, 10,000 elements and 100,000 elements in 0.003 s, 0.103 s and 10.233 s, respectively. Therefore a full sorted array consists of sequential numbers in desired order, sorting process for these sorted arrays are expected to not waste time. On the contrary, as seen in Table-3, Selection sort spends seconds to sort full sorted arrays.

Selection sort for semi sorted arrays having 1,000 elements and 10,000 elements gives same performance with Insertion sort. On the other hand, when number of elements in the array increases to 100,000, Insertion sort outperforms Selection sort.

Selection sort for unsorted arrays works better more than Insertion sort in terms of time. Running time of Selection sort for unsorted arrays is getting shorter than Insertion sort.

Table -3: Running time to sort arrays using Selection sort and Insertion sort

	Selection Sort Running Time (Second)	*Insertion Sort Running Time (Second)*
Full sorted array with 1,000 elements	**0.003 s**	**0 s**
Semi sorted array with 1,000 elements	**0.002 s**	**0.002 s**
Unsorted array with 1,000 elements	**0.001 s**	**0.002 s**
Full sorted array with 10,000 elements	**0.103 s**	**0 s**
Semi sorted array with 10,000 elements	**0.098 s**	**0.099 s**
Unsorted array with 10,000 elements	**0.1 s**	**0.127 s**
Full sorted array with 100,000 elements	**10.233 s**	**0 s**
Semi sorted array with 100,000 elements	**10.455 s**	**9.423 s**
Unsorted array with 100,000 elements	**9.621 s**	**12.449 s**

5. CONCLUSIONS

To compare Selection sort with Insertion sort, these two algorithms have different advantages in different situations. Therefore, it can not be said that one of them is more successful than the other completely. Insertion sort algorithm for full sorted arrays ourperforms Selection sort algorithm in term of running time. On the contrary, Selection sort for unsorted arrays outperforms Insertion sort in term of running time.

REFERENCES

[1] Abdulla, Mirza. "An O (n^(4/3)) worst case time selection sort algorithm." IJCER 5.3 (2016): 36-41.

[2] Adhikari, Pooja., "Review On Sorting Algorithms A comparative study on two sorting algorithms." Mississippi State, Mississippi 4 (2007).

[3] Raza, Muhammad Ali, et al. "Comparison of Bubble Sort and Selection Sort with their Enhanced Versions.".

[4] Al-Kharabsheh, Khalid Suleiman, et al. "Review on Sorting Algorithms A Comparative Study." International Journal of Computer Science and Security (IJCSS) 7.3 (2013): 120-126.

[5] Aliyu, Ahmed M., and P. B. Zirra. "A Comparative Analysis of Sorting Algorithms on Integer and Character Arrays." The International Jornal of Engineering and Science (2013): 25-30.

[6] Grover, Deepti, and Sonal Beniwal. "Performance Analysis of Merge Sort and Performance Analysis of Merge Sort and Quick Sort: MQSORT.".

[7] Sharma, Sunny, et al. "VRF: A Novel Algorithm for optimized Sorting." (2016).

[8] Zafar, Sardar, and Abdul Wahab. "A new friends sort algorithm." Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on. IEEE, 2009.

[9] Alnihoud, Jehad, and Rami Mansi. "An Enhancement of Major Sorting Algorithms." Int. Arab J. Inf. Technol. 7.1 (2010): 55-62.

[10] ALI, WAQAS, et al. "COMPARISON OF DIFFERENT SORTING ALGORITHMS." International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE) 5.7 (2016): pp-63.

[11] Rajagopal, D., and K. Thilakavalli. "Different Sorting Algorithm's Comparison based Upon the Time Complexity." (2016).

[12] Jadoon, Sultanullah, Salman Faiz Solehria, and Mubashir Qayum. "Optimized selection sort algorithm is faster than insertion sort algorithm: a comparative study." International Journal of Electrical & Computer Sciences IJECS-IJENS11.02 (2011): 19-24.

[13] Abdulla, Mirza. "Selection Sort with Improved Asymptotic Time Bounds.". The International Journal of Engineering and Science (THE IJES), To Appear May-2016.

[14] Min, Wang. "Analysis on 2-element insertion sort algorithm." Computer Design and Applications (ICCDA), 2010 International Conference on. Vol. 1. IEEE, 2010.

[15] Yadav, Neelam, and Sangeeta Kumari. "SORTING ALGORITHMS." IRJET (2016).