Sorting Algorithms: Review Paper

Rashid Nizam¹, Asad Mansoori²

¹Department of Physics, ²Department of Computer Science, IFTM University, Moradabad, India

Abstract: The computer system is performed any operation in sequential order. All computer operation perform with help of sorting algorithms. The system's operation are solved some problems. Sorting algorithm can merge space and arrange sequence order. The computational complexity, ordering and other properties are performs under sorting algorithm. This paper is discus two type sorting algorithm: Selection sort and Shell sort

Keywords: Selection sorts, sequential order, merge space and Shell sort

I. INTRODUCTION

Sorting algorithm is form of the rearrangement any operation performs correct lexicographic order [1]. The sorting algorithm has been developed which are some type methods used like as include heap sort, merge sort, quick sort, selection sort etc [1, 2]. The sorting methods are compare complexity, run time and perfumes one sorting method to any other sorting method. According to this paper two sorting methods are compare selection sort [3] and shell sort [4].

The selection sort is a very simple method [3] because this method running time is not optimal to compare any other sorting algorithm. The running time analysis is fixed in sorting method. The better sorting algorithm compared other shell sort selection sort and work in gap sequence.

Selection_Sort (A)
for i = 0 to n-1;
    min←i;
    for j=i+1 to n;
        if {A[j] <A[min]
            min←j;
        }else
            Swap (A[i], A[min])

Cost of each step in solving the problem
O (n)
O (n^2)
O (1)
O (1)
O (1)

Total run time Cost==Θ (n^2)

II. SORTING

Selection and shell sort comparison running time and performance analysis decided better sorting algorithm. Selection sort has advantage as well as disadvantage and so shell sort. All sorting algorithm are equally important and particular purpose.

III. SELECTION SORT

Selection sort is the simplest algorithm. Selection sort is the improve form of bubble sort in perform. The selection sorts minimize each step to given element as well as each element right position in the given list unsorted items. In sorting the number of items, an algorithm uses a worst-case, average-case and best-case run time of Θ (n2) and comparing in the constant time. The selection sort spends the maximum time and minimize element in “unsorted” by array. The selection sort algorithm in define as:

Accordingly sorting any element string first of all choose the left element in the string and search the small element campers to choose first element values (n-1). After comparing inter change location again next largest element select right side in string and the search minimum value element search left side in string.

Selection sort has advantage as well as disadvantage and so shell sort. All sorting algorithm are equally important and particular purpose.
**Example:**

<table>
<thead>
<tr>
<th>Step</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>64 25 12 22 11 ←search</td>
</tr>
<tr>
<td>2</td>
<td>11 25 12 22 64</td>
</tr>
<tr>
<td>3</td>
<td>11 25 12 22 ← 64</td>
</tr>
<tr>
<td>4</td>
<td>11 22 12 25 64</td>
</tr>
</tbody>
</table>

Time Complexity for Selection Sort is the same for all cases - worse case, best case or average case $O(n^2)$

The Running time formula is $N(N-1)/2$.

<table>
<thead>
<tr>
<th>Number of Array Element</th>
<th>Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>28</td>
</tr>
<tr>
<td>16</td>
<td>120</td>
</tr>
<tr>
<td>32</td>
<td>496</td>
</tr>
<tr>
<td>64</td>
<td>2016</td>
</tr>
<tr>
<td>128</td>
<td>8128</td>
</tr>
</tbody>
</table>

**Table 1.**

The statement is clarifies. The above algorithm many time executed the running time depend on array size.

Form the table it become more clear that the smallest number is very good but the comparison of large data (large number) will lead to a huge time.

The array is sorted in descending order if is the worst case. Nonetheless, the original order of the array is the set selection sort algorithm in not time saving. The array to be sorted the test "if $A[j] < A[\text{min}]$"is exactly the same number of time in every case.

V. SHELL SORT

The Shell sort was developed by DONALD SHELL in 1959. Shell Sort is comparison other sort and the case run time is $\Theta(n^2)$. This is an insertion sort with working in the gap sequence. Since the input almost sorted is perfect by insertion sort because any one array of element value is one potion move at a time, the shell sort fully uses the advantages and removing the disadvantage of insertion sort in a given time.

Shell sort having gap of several position according insertion sort on elements separated. Its expected position forward “Large Steps” take element in this sort.

*For Example:*

Let us assume that we have a array in 15 number to be sorted.

Input data $\rightarrow$ 83 15 62 12 53 7 16 85 47 25 10 11 69 3 9

Let us first do the sorting between elements which are in gap 5 to each other.

A1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 a15

Input data $\rightarrow$ 83 15 62 12 53 7 16 85 47 25 10 11 69 3 9

5 sorting

1. 83 53 47 69
2. 15 7 25 3
3. 62 16 10 9
4. 12 82 11

The result of this step: 3 7 9 10 11 12 15 16 25 47 53 62 69 83 85

Similarly same step follow in second step and the gap is less than first sorting gape. After that in final step gap is 1. This is a local insertion sort science all element solve a particular point it will not use more time thus the insertion sort safely condition in the worst case.

This method is inefficient in large number of data and complex but work faster in finding the quantity data sets.

After the first step the data is some part in sequence sort and next steps will increase. The algorithm of shell sort in given below.

SHELLSORT (A)

1. Step $\leftarrow m$
2. While step $> 0$
3. For (i$\leftarrow 0$ to n with increment 1)
4. Temp$\leftarrow 0$
5. j$\leftarrow i$
6. for (k$\leftarrow j+$step to n with increment step)
7. Temp$\leftarrow A[k]$
8. j$\leftarrow k-step$
9. While (j$\leftarrow 0$ & & A[j]$\geq$temp)
11. j$\leftarrow j-step$
12. Array[j]+$step$← temp$
13. Step$← step/2$
The algorithm performs insertion sort between elements and some specific gaps. The sorted order completion each step of element increase gap and constant value less than the value of n the best case $O(n)$ and the n step resulting running time $\Theta(n^2)$.

### VI. PERFORMANCE ANALYSIS OF SHELL SORT

The following given table represents running time in shell sort. According to this algorithm, the running time is different depending on input element:

<table>
<thead>
<tr>
<th>Number of array Element</th>
<th>Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>16</td>
<td>38</td>
</tr>
<tr>
<td>32</td>
<td>125</td>
</tr>
<tr>
<td>64</td>
<td>386</td>
</tr>
<tr>
<td>128</td>
<td>1682</td>
</tr>
</tbody>
</table>

In this table, not exactly $\Theta(n^2)$. The shell sort is increment performance in implementing the best part of insertion sort and the gap step array is increase partition in shell sort.

### VII. GAP SEQUENCE

The gap sequence is an important part of shell algorithm. The array element increment sequence work in last element is 1. This algorithm performs a gap by insertion sort for all numbers of elements in array.

### VIII. PERFORMANCE COMPARISISON

Selection sort and shell sort are comparing in the same input array data. According to selection sort, select small elements and increase to large elements but shell sort is perform array element perform insertion sort by gap sequence. The running time is compared in Table 1 and Table 2 in a graph as follows:

![Chart Title](chart.png)

### IX. CONCLUSION

This paper discusses two papers based upon sorting algorithms and the performance with the same number in algorithms.

After all, selection sort and shell sort analyses and it is found that the shell sort is best when compared with the selection sort because it is structure cannot be used for large array. The selection sort and shell sort upper bound running time are equal $O(n^2)$.
REFERENCES


Corresponding author:

Email: rashid.nizam@gmail.com