A Comparative Study on Different Types of Sorting Algorithms (On the Basis of C and Java)

Rekha dwivedi
PG scholar
Department of Computer Science & Engineering
SVITS –Indore (M.P)
Email: rekhadwivedi40@yahoo.com

Dr. Dinesh C. Jain
Reader
Department of Computer Science & Engineering
SVITS –Indore (M.P)
Email: dineshwebsys@gmail.com

Abstract- Sorting is used for arranging the data in some sequence like increasing or decreasing order. I have discussed about various sorting algorithm with their comparison to each other in basis of time complexity and space complexity as well as C and Java. These papers also show running time of algorithm with the help of C language and Java. I have compared some types of sorting algorithm like insertion sort, selection sort, quick sort, and bubble sort by comparing time complexity and space complexity.

Keyword- Bubble sort, Insertion sort, Selection sort, Quick sort, Time complexity.

I. INTRODUCTION

A. Sorting

Let p be a list of m elements P1, P2, P3…….Pn in memory. Sorting P means arranging the content of P in either increasing or decreasing order i.e.,P1<P2<P3<P4………..<Pn.

There are m elements in the list, therefore there is m! ways to arrange them.

B. Sorting Algorithm

Sorting algorithm is an important task for arranging the elements in the list. Comparing the various types of sorting in this paper on the basis of C and Java.

II. TYPES OF SORTING ALGORITHM

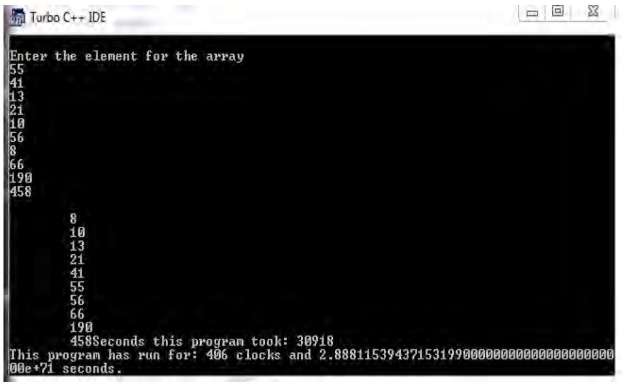
A. Insertion Sort

An insertion sort is one that sorts a set of value by inserting values into an existing sorted file. It is useful for smallest elements of array.

Therefore the total no. of comparisons are:
T(n)=1+2+3+….+(i-1)+……+(n-1)= n(n-1)/2

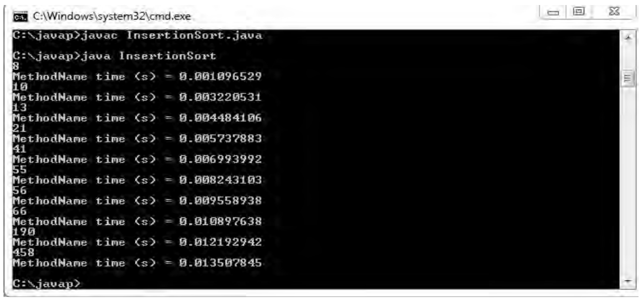
T(n)=O(n)

The execution time of Insertion Sort in C is more as compared to Java.



```
Enter the element for the array
55
41
13
21
10
56
8
66
190
458

        8
        10
        13
        21
        41
        55
        56
        66
        190
        458Seconds this program took: 30918
This program has run for: 406 clocks and 2.8881153943715319900000000000000000000000
00e+71 seconds.
```

Figure 1. Insertion Sort in C



```
C:\javap>javac InsertionSort.java

C:\javap>java InsertionSort
8
MethodName time (s) = 0.001096529
10
MethodName time (s) = 0.003220531
13
MethodName time (s) = 0.004484106
21
MethodName time (s) = 0.005737883
41
MethodName time (s) = 0.006993992
55
MethodName time (s) = 0.008243103
56
MethodName time (s) = 0.009558938
66
MethodName time (s) = 0.010897638
190
MethodName time (s) = 0.012192942
458
MethodName time (s) = 0.013507845

C:\javap>
```

Figure 2. Insertion Sort in Java

B. Bubble Sort

In Bubble sort, each element is compared with its adjacent element. If the first element is larger than the second one then the position of the element is interchanged, other it is not changed. Then next element is compared with its adjacent element and the same process is repeated for all the elements in the array.

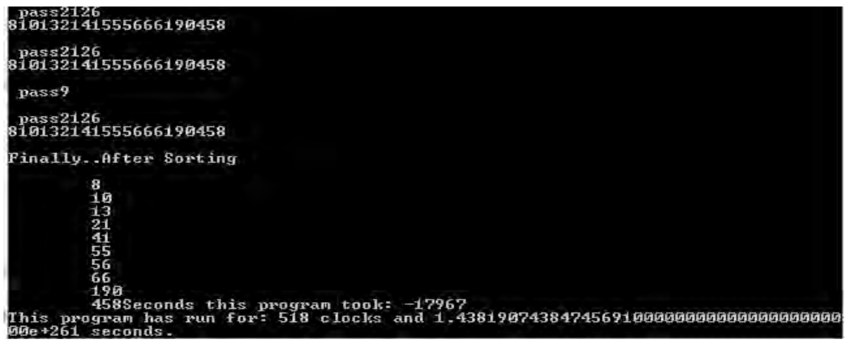
In bubble sort, the first pass requires (n-1) comparison to fix the highest element to its location , the second pass requires (n-2),……,ith pairs requires (n-i) and the last pass requires only one comparison to be fixed at its proper position.

Therefore the total no. of comparisons are:

T(n)= (n-1)+(n-2)+(n-3)+……..+(n-i)+3+2+1=n(n-1)/2

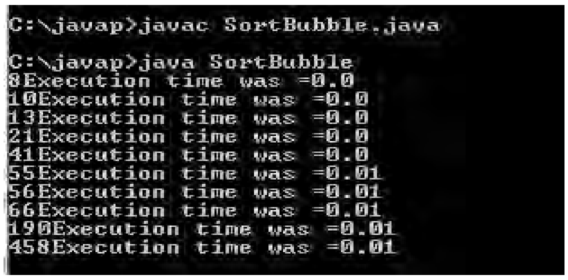
T(n)=O(n^2)

The execution time of Bubble Sort in C is more as compared to Java.



```
 pass2126
8101321415556661904 58

 pass2126
81013214155566619045 8

 pass9

 pass2126
810132141555666190458

Finally..After Sorting

        8
        10
        13
        21
        41
        55
        56
        66
        190
        458Seconds this program took: -17967
This program has run for: 518 clocks and 1.4381907438474569100000000000000000000000
00e+261 seconds.
```

Figure 3. Bubble Sort in C



```
C:\javap>javac SortBubble.java

C:\javap>java SortBubble
8Execution time was =0.0
10Execution time was =0.0
13Execution time was =0.0
21Execution time was =0.0
41Execution time was =0.0
55Execution time was =0.01
56Execution time was =0.01
66Execution time was =0.01
190Execution time was =0.01
458Execution time was =0.01
```

Figure 4. Bubble Sort in Java

C. Selection Sort

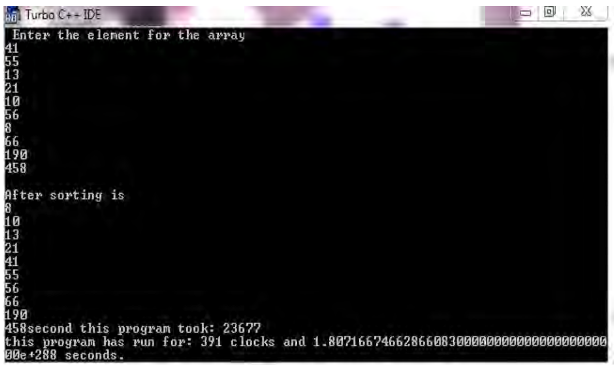
In selection sort, the first element of array is compared with minimum value of array and interchanged the position of element. Then element is compared with the next minimum value of array and the same process is repeated for all elements in the array.

In selection sort makes first pass in (n-1) comparisons, the second pass in (n-2) comparisons and so on.Total no. of comparison are:

T(n)=(n-1)+(n-2)+(n-3)+,……………,+(n-i)+3+2+1=n(n-1)/2

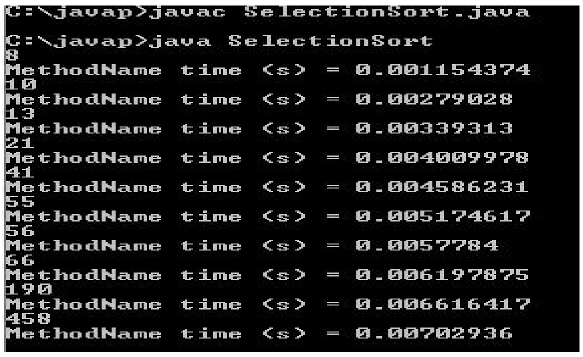
T(n)=O(n^2)

The execution time of Selection Sort in C is more as compared to Java.



```
 Enter the element for the array
41
55
13
21
10
56
8
66
190
458

After sorting is
8
10
13
21
41
55
56
66
190
458second this program took: 23677
this program has run for: 391 clocks and 1.8071667466286608300000000000000000000
00e+288 seconds.
```

Figure 5. Selection Sort in C



```
C:\javap>javac SelectionSort.java

C:\javap>java SelectionSort
8
MethodName time (s) = 0.001154374
10
MethodName time (s) = 0.00279028
13
MethodName time (s) = 0.00339313
21
MethodName time (s) = 0.004009978
41
MethodName time (s) = 0.004586231
55
MethodName time (s) = 0.005174617
56
MethodName time (s) = 0.0057784
66
MethodName time (s) = 0.006197875
190
MethodName time (s) = 0.006616417
458
MethodName time (s) = 0.00702936
```

Figure 6. Selection Sort in Java

D. Quick Sort

Quick sort works by partitioning methods for sorting the array. And each partition is in turn sorted recursively. In partition, one element of array is selected as a pivot value. This pivot value can be the first element of array. The array elements are grouped into two partition 1. One partition contains elements that are smaller than pivot value.2.Another partition contains elements that are larger than pivot value. Time required to partition the array is: O(n). The execution time of Quick Sort in C is more as compared to java.

III. COMPARISON OF SORTING ALGORITHM IN TABULAR FORM

Sort	*Time Complexity*	*Advantages & disadvantages*
Insertion Sort	O(n)	The advantage of insertion sort is its simplicity. It is also good performance for smallest array. The disadvantage of insertion sort is that it is not useful for large elements array.
Selection Sort	O(n^2)	The advantage of selection sort is that it performs well on small array. The disadvantage of selection is that it is poor efficiency for large elements array.
Bubble Sort	O(n^2)	The advantage of bubble sort is that it is easily implemented. In bubble sort, the elements are swapped without additional temporary storage, so space requirement is minimum. The disadvantage of bubble sort is same as a selection sort.
Quick Sort	O(n log n)	The advantage of Quick sort is that it is used for small elements of array as well as large elements of array. Disadvantage of Quick sort is that the worst case of quick sort is same as a bubble sort or selection sort.

IV. GRAPHICAL REPRESENTATION OF SORTING ALGORITHM

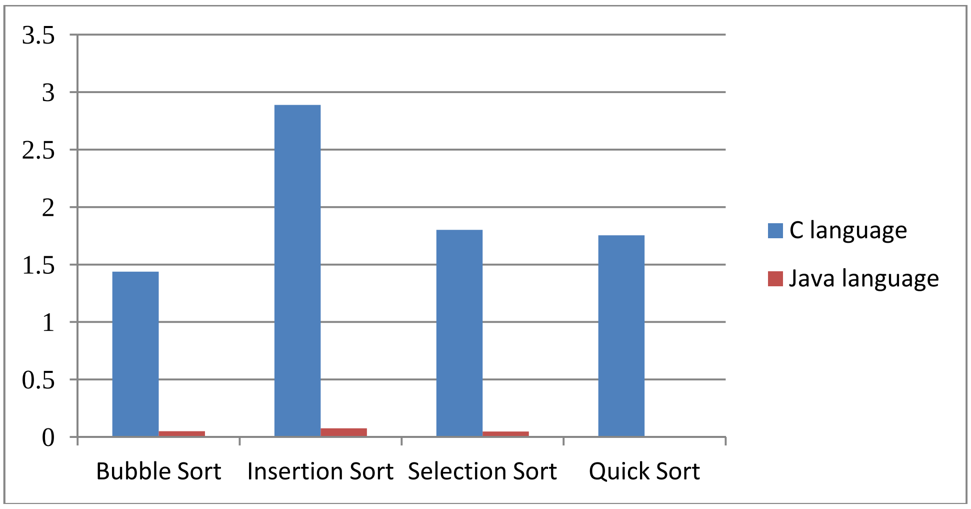


Figure 7. Run Time (Seconds) of Sorting In C and Java

V. CONCLUSION

In this study I have studied about various sorting algorithm and comparison on the basis of time complexity, execution time and C & Java languages. I used to the C and Java program for finding the execution time in second. I observe that when compare all the sorting algorithms to each other then find the execution time of quick sort algorithm is best to others and also observe that the execution time of all sorting algorithms in java is best then C language.

VI. FUTURE SCOPE

This paper could help to the researchers in evaluating the all types of Sorting Algorithms by which they could easily understand the pros and cons of Sorting algorithms and also to find the application of these Algorithms in different areas.

REFERENCES

[1] Ellis Horowwitz, Sartaj Sahini, Sanguthevar Rajasekaran , Fundamental of Computer Algorithms, ISBN 81-7515257-5 by 1998
[2] Demuth, H. Electronic Data Sorting. PhD thesis, Stanford University, 1956.
[3] BLELLOCH, G. E., LEISERSON, C. E., MAGGS, B. M., PLAXTON, C. G., SMITH, S. J., AND ZAGHA, M. 1991. A comparison of sorting algorithms for the connection machine CM-2. In Proc. Symposium on Parallel Algorithms and Architectures (Hilton Head, SC, July 1991).
[4] THEARLING, K. AND SMITH, S. 1992. An improved super-computing sorting benchmark. In Supercomputing 92 (1992), pp. 14 – 19. IEEE Press.
[5] Perl sort documentation (http:/ / perldoc. perl. org/ functions/ sort. html)
[6] Tim Peters's original description of timsort (http:/ / svn. python. org/ projects/ python/ trunk/ Objects/ listsort. txt)
[7] (http:/ / hg. openjdk. java. net/ jdk7/ tls/ jdk/ rev/ bfd7abda8f79)
[8] Merge sort in Java 1.3 (http:/ / java. sun. com/ j2se/ 1. 3/ docs/ api/ java/ util/ Arrays. html#sort(java. lang. Object[])), Sun.
[9] Java 1.3 live since 2000
[10] http://www.scribd.com/doc/45996720/Run-Time-Analysis-of-Insertion- Sort-and-Quick-Sort.
[11] Owen Astrachan, Bubble Sort: An Archaeological Algorithmic Analysis, SIGCSE,2003, http://www.cs.duke.edu/~old/papers/bubble.pdf.