The International Journal Of Engineering And Science (IJES)
||Volume||2 ||Issue|| 7 ||Pages|| 25-30||2013||
ISSN(e): 2319 – 1813 ISSN(p): 2319 – 1805

A Comparative Analysis of Sorting Algorithms on Integer and Character Arrays

Ahmed M. Aliyu, Dr. P. B. Zirra
Computer Science Department, Adamawa State University, Mubi

----------***ABSTRACT***----------
Most Practical applications in computer programming will require the output to be arranged in a sequential order. A lot of sorting algorithms has been developed to enhance the performance in terms of computational complexity, memory and other factors. This paper is an attempt to compare the performance of two sorting algorithm: Selection and Quick Sort, with the aim of comparing their speed when sorting an integer and string arrays. Analysis of these two sorting algorithm were also carried out. The finding shows that selection sort performs better than Quick sort. The study also indicates that integer array have faster CPU time than string arrays although both have the upper bound running time $O(n^2)$.

KEYWORDS: *Algorithm, Time Complexity, Quick Sort, Selection Sort,*

Date of Submission: 06 July. 2013, Date of Publication: 20.July 2013

I. INTRODUCTION

Sorting is the rearrangement of items in a list into their correct lexicographic order, while algorithm is a step by step why of achieving solution for a desired result. A number of sorting algorithms have been developed like Quick sort, heap sort, merge sort; selection sort all of which are comparison based sort .There are other classes of sorting algorithms which are non-comparison based sort. The paper gives a brief introduction about sorting algorithms as put forward by [8] where he discussed about classes of sorting algorithms and their running times. He mainly analysed the performance between two sorting algorithms .Selection sort and Quick sort . Selection sort is the simple sorting method with a very simple sorting algorithm. However the running time of this algorithm is not that optimal as compared to performance of best sorting algorithm. [1] Analyses the running time of quick sort where he discovered that the running time increases as the array size increases. Quick sort is another comparison based sorting algorithm which is better than selection sort and works in a gap sequence [6]. The running time and the performance analysis. [9] shows that although its performance is better than the selection sort, there are other various factor that needs to be keeps in mind. The advantages of selection sort contain within it a lot of disadvantage and so are Quick sort However; there are equally important and is best for a particular purpose. This paper compare the performance of two sorting algorithm with the aim to compare their speed when sorting an integer and string arrays.

STATEMENT OF THE PROBLEM

Sorting is a problem that mostly arises in computer programming. Many sorting algorithms have been developed while some existing ones have been improved upon; all to make them run faster. According to [1] efficiency of an algorithm can be measured in terms of execution (time complexity) and the amount of memory required. While time complexity is the amount of time required to execute an algorithm it is very important for a programmer to note that there are certain factors that does not affect time complexity, which includes among them are, the programming language chosen to develop the algorithm and the quality of the compiler. Therefore most programmers are faced with the challenges in chosen which algorithm would be best used in developing their software as some are faster than others in execution time while some may be slower when developing a string dependent database or integer dependent database, This paper therefore would analyze these two situations on two commonly used algorithms, Selection sort and Quick sort and their behavior on both string and integer arrays.

AIM AND OBJECTIVES OF THE STUDY

The aim of this paper is to compare the time taken to sort integer array and string of same size for two selected sorting algorithm in line with other specific objectives, which include:

(i) Determine whether sorting a string is faster or sorting an integer array.
(ii) Develop software to perform the analysis.

SCOPE OF THE STUDY

This paper had concentrate on sorting items in an array in memory using comparison sorting involving commonly used sorting algorithm namely, selection sort and Quick Sort.
Inbuilt CPU system time was used while the data was generated randomly for a fixed array size in the range of 1000 to 3000.
C++ programming plat form was used for the developing the software to run these algorithms.

RELATED WORK

[8] In his book titled Sorting and searching algorithm: A cook book state that Quicksort executes in $O(nlgn)$ on average, and $O(n^2)$ in the worst-case. However, with proper precautions, worst-case behavior is very unlikely. Quicksort is a non-stable sort. It is not an in-place sort as stack space is required.[3] In their work "parallel Quicksort algorithm Part 1 - Run time analysis" stated that Quicksort being such a popular sorting algorithm, there have been a lot of different attempts to create an efficient parallelization of it. The obvious way is to take advantage of its inherent parallelism by just assigning a new processor to each new subsequence. This means, however, that there will be very little parallelization in the beginning, when the sequences are few and large. Another approach by [5] has been to divide each sequence to be sorted into blocks that can then be dynamically assigned to available processors . However, this method requires extensive use of atomic FAA2 which makes it too expensive to use on graphics processors.[2] In his work "Pre_x Sums and Their Applications" suggested using pre_x sums to implement Quicksort and recently [7] used this method to make an implementation for CUDA . The implementation was done as a demonstration of their segmented scan primitive, but it performed quite poorly and was an order of magnitude slower than their radix-sort implementation in the same paper.[7] have presented a radix-sort and a Quicksort implementation. Recently, while [8] presented a hybrid sorting algorithm a modified version of [7] which splits the data with a bucket sort and then uses merge sort on the resulting blocks.

II. METHODOLOGY

SELECTION SORT

Selection sort is one of the simplest sorting algorithms. It is actually an improvement in performance of bubble sort. This algorithm is called selection sort because it works by selecting a minimum element in each step of the sort. This algorithm, iterate through a list of n unsorted items, has a worst-case, average-case, and best-case run- time of $\Theta(n^2)$, assuming that comparisons can be done in constant time. The Selection sort spends most of its time trying to find the minimum element in the "unsorted" part of the array. The brief algorithm for selection sort is given below.

Name: algorithm 1

SELECTION_SORT (A)		Cost of Each Step
1:	for i ← 0 to n-1 do	O(n)
2:	min ← i;	O(1)
3:	for i ← 0 to n-1 do	$O(n^2)$
4:	If A[j] < A[min] then	O(1)
5:	min←j	O(1)
6:	swap(A[i] ,A[min])	O(1)

Total run time Cost== $\Theta(n^2)$ asymptotically
It works as follows:

1. Find the smallest element using a linear scan.
2. Swap the element with the element in first position.
3. Find the second smallest element in the remaining array.
4. Then swap to the second position.
5. Continue the steps until a sorted list is obtained.

PERFORMANCE ANALYSIS OF SELECTION SORT

For each value of element until n it needs to compare and check whether there is any value smaller than it and needs (n-1) comparisons. Thus it can be easily seen that run time complexity of selection sort is $\Theta (n^2)$ from line No.3.But since it just requires the Θ (n) swaps means linear order writes to the memory which is optimal to any sorting algorithm. Table 1 and 2. Further clarifies the above statement. The running time is obtained by calculating how many times the line 4 of the above algorithm got executed on the specified array size.

Table 1. Running time of selection sort on integer

Number of Array Elements	Running Time (sec)
1000	0.364
1500	0.565
2000	0.763
2500	0.965
3000	1.064

Table 2. Running time of selection sort on String Array

Number of Array Elements	Running Time (sec)
1000	0.384
1500	0.762
2000	0.963
2500	1.156
3000	1.361

From the above table it is clear that this naive approach is very good for small number of elements. But it requires the comparison of every element to every other element which will lead to a great deal of work for large data sets. The worst case occurs if the array is already sorted in descending order. Nonetheless, the time require by selection sort algorithm is not very sensitive to the original order of the array to be sorted: the test "if A[j] <A[min]" is executed exactly the same number of times in every case.

QUICK SORT

Quick sort is a comparison sort developed by Tony Hoare. Also, like merge sort, it is a divide and conquer algorithm, and just like merge sort, it uses recursion to sort the lists. It uses a pivot chosen by the programmer, and passes through the sorting list and on a certain condition, it sorts the data set. Quick sort algorithm can be depicted as follows:

Name: algorithm 2

```
QUICKSORT (A)
1:      step←m;
2:      while step > 0
3:      for (i←0 to n with increment 1)
4:      do   temp←0;
5:      do       j←i;
6:      for (k←j+step to n with increment step)
7:      do       temp←A[k];
8:      do       j←k-step;
9:      while (j>=0 && A[j]>temp)
10:     do       A[j+step]=A[j];
11:     do       j←j-step;
12:     do       Array[j]+step←temp;
13:     do       step←step/2;
```

Run Time of Quick Sort

Best case is $O(nlogn)$, Average case is $O(nlogn)$ and the worst case is $O(n^2)$.

The main focus of the Quick sort lies in line 6. Line 6-12 is algorithm for insertion sort. But insertion sort is performed between the elements between some specified gaps. Since the sorted order of elements get increased with the completion of each step these lines gives a constant value less than the value of n. Thus in the best case of this is almost O(n).But there may be the worst case condition for large number of inputs where computational complexity where the gap insertion sort can give almost n steps resulting into running time of $\Theta(n^2)$.

PERFORMANCE ANALYSIS OF QUICK SORT

The following table gives the running time of Quick sort for both integer and string array .It is obtained by running the above algorithm in the different input elements. Running time is obtained by counting the number of times line 10 and 11 got executed.

Table 3. Running time of Quick sort on integer

Number of Array Elements	Running Time (sec)
1000	0.836
1500	0.998
2000	1.032
2500	1.221
3000	1.398

Table 4. Running time of Quick sort on string

Number of Array Elements	Running Time (sec)
1000	1.002
1500	1.124
2000	1.232
2500	1.431
3000	1.634

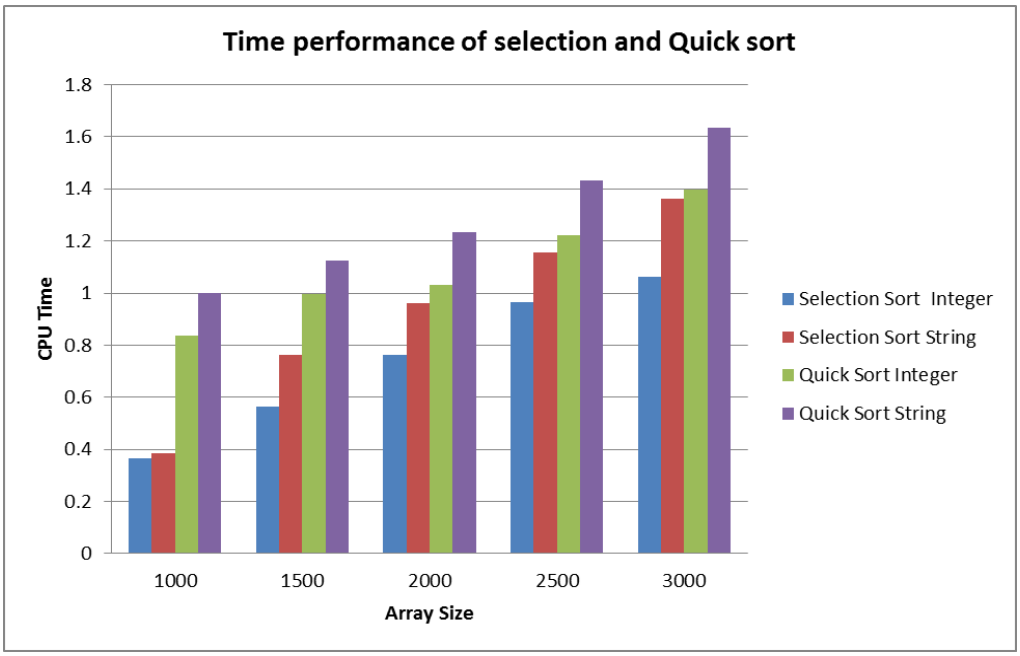
Thus from the above table it is clear that it is not exactly $\Theta(n^2)$. The increment in the performance of Quick sort is because it tries to implement the best part of insertion sort. After each gap step the array elements will get partly sorted which makes the performance increase for the Quick sort.

PERFORMANCE COMPARISION

Selection sort and Quick sort are both comparisons based sorting algorithms which uses two different perspectives. One operates by selecting smallest element of the unsorted part of the list each time. Another operates by performing insertion sort in a gap sequence. The generalized performance of the two algorithms on both integer and string arrays is summarized in the table below:

Table 5 summary of run time on of integers and String Arrays for the two sorting techniques

	Selection Sort		Quick Sort	
	Integer	String	Integer	String
1000	0.364	0.384	0.836	1.002
1500	0.565	0.762	0.998	1.124
2000	0.763	0.963	1.032	1.232
2500	0.965	1.156	1.221	1.431
3000	1.064	1.361	1.398	1.634



III. DISCUSSION OF RESULT

From the above analysis also it is clear that performance of selection sort is better than Quick sort. Selection Sort is fairly simple algorithm and can be easily implemented. The algorithm of Quick sort is rather complex and it will give very poor running time in case of large sets of data. But the performance of both algorithms is worse than the lower bound run time of comparison sort O (nlogn).These algorithms can be implemented for small amount of data but not for large amount of data.

VI. CONCLUSION

This paper discusses two comparison based sorting algorithms. It analyses the performance of these algorithms for the same number of elements .It then concludes that selection sort shows better performance than Quick sort but being the simple structure selection sort is more preferred and widely used. It is clear that both sorting techniques are not that popular for the large arrays because as the arrays size increases both timing is slower for integer and string, but generally the study indicate that integer array have faster CPU time than string arrays. Both have the upper bound running time O (n^2).

REFERENCES

[1] Ananth G. ,Anshul G.,George K. & Vipin K.(2007): ,*Introduction to Parallel Computing*, 2nd Ed.,Addison-Wesley
[2] Guy, E. Blelloch. (1993): Pre_x Sums and Their Applications. In John H. Reif, editor, Synthesis of Parallel Algorithms. Morgan Kaufmann
[3] Macllory, H., Norton, A., & John, T. R. (1996): Parallel Quicksort Using Fetch-And Add. IEEE Trans. Comput., 133-138.
[4] Pooja A.,(2012) Review On Sorting Algorithms :A comparative study on two sorting algorithms IBM Dublin Research Lab, Mulhuddart, Dublin 15, Ireland
[5] Philippas, T. & Yi Z. (2003): A Simple, Fast Parallel Implementation of Quicksort and its Performance Evaluation on SUN Enterprise 10000. Eleventh Euromicro Conference on Parallel, Distributed and Network-Based Processing,372-374,
[6] Robert L(2002): *Data Structures and Algorithms in Java*,2nd Ed.24-28.
[7] Sengupta *et al*.,(2007): *Algorithms in Java, Parts 1-4, 3rd ed.* Addison-Wesley Professional Publisher, New York.
[8] Sntorn *et al*. (2007) :Adapting Radix Sort to the Memory Hierarchy", *Journal of Experimental Algorithmic* (JEA), pp.7-9
[9] Thomas H. Cormen,C. Leiserson E.,Ronald L. Rivest & Clifford S. (2004): Introduction toAlgorithms,2nd Ed., Prentice-Hall New Delhi
[10] Wikipedia,(2007) :*Sorting Algorithm*, Retrieved from http://en.wikipedia.org/wiki/Sorting_algorithm: 24-05-2013
[11] Wikipedia,(2007) :*Selection Sort*, Retrieved from http://en.wikipedia.org/wiki/Selection_sort 26-05-2013

BIOGRAPHY

Ahmed M. Aliyu is a Postgraduate Student from Adamawa State University, Mubi Nigeria and a Lecturer with Federal Polytechnic, Mubi. He is a Holder of B.Tech Degree in Computer Science from Abubakar Tafawa Balewa University Bauchi, Nigeria and has Postgraduate Diploma from Bayero University Kano. He is currently an M.Sc student at Adamawa University. His main area of research is Algorithm Analysis. He is a Registered Member of Nigeria Computer Society .He is happily married with six children.

Dr. P. B. Zirra is a lecturer with Adamawa state University, Mubi Nigeria. He obtained his Doctorate Degree in Computer Science from Modibbo Adamawa University Yola in 2012, M.sc in Computer science from A.T.B.U Bauchi in 2006, MBA (Finance) from University of Maiduguri in 2000 and had his B.Tech. Computer 1994 from ATBU. His Area of interest include Computer Network and Security, he is happily married with two Children.