ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 3, Issue 2, March 2014

Sorting Algorithm: An Empirical Analysis

Mahfooz Alam, Ayush Chugh

Abstract*— The 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 "Sorting Algorithm: An Empirical Analysis" consists of rigorous complexity analysis by various sorting algorithms, in which comparison and real swapping of all the variables are calculated. User run time of CPU is also captured for each sorting algorithm. It is an attempt to compare the performance of various sorting algorithm, with the aim of comparing their speed when sorting an integer inputs. The empirical data obtained by using the program reveals that Quick sort algorithm is fastest and Bubble sort is slowest.

***Index Terms*—Sorting, Comparison of sorting Algorithm, and empirical Analysis of sorting Algorithm.**

I. INTRODUCTION

A Sorting is the reorganization of items in a list into their correct lexicographic order, while algorithm is a step by step way of achieving solution for a desired result. A number of sorting algorithms have been developed like Quick sort, heap sort, shell sort, selection sort, etc. all of which are comparison based sort. There are other classes of sorting algorithms which are non-comparison based sort. This paper gives a brief introduction about sorting algorithms as put forward by [5] where we discussed about classes of sorting algorithms and their running times. We mainly analyzed the performance among various sorting algorithms.

Further, the data is often taken to be in an array, which allows random access, rather than a list, which only allows sequential access, though often algorithms can be applied with suitable modification to either type of data. Since the dawn of computing, the sorting problem has attracted a great deal of research, perhaps due to the complexity of solving it efficiently despite its simple, familiar statement. This paper investigates the characteristic of the sorting algorithms with reference to number of comparisons made and number of swaps made for the specific number of elements. Sorting algorithms are used by many applications to arrange the elements in increasing/decreasing order or any other permutation. Sorting algorithms, like Quick Sort, Shell Sort, Heap Sort, Insertion Sort, Bubble Sort etc. have different complexities depending on the number of elements to sort [1]. The purpose of this investigation is to determine the number of comparisons, number of swap operations and after that plotting line graph for the same to extract values for polynomial equation. The values a, b and c got is then used for drawing parabola graph. Through this paper, a conclusion can be brought on what algorithm to use for a large number of elements. For larger arrays, the best choice is Quicksort, which uses recursion method to sort the elements, which leads to faster results. Program for each sorting algorithm in which a counter is used to get the number of comparisons, number of swap/assignment operations is used. The data is stored in a file, from where it is used for calculation purpose in an excel file. Least square method and Matrix inversion method is used to get the value of constants a, b and c for each polynomial equation of sorting algorithms. After calculating the values, Graph is drawn for each sorting algorithm for the polynomial equation [2] i.e. $Y = A.X\text{^}2 + B.X + C$ or $Y = A.X.\lg X + B.X + C$. A graph for each sorting algorithm for the above two equation is drawn to show the variation and finally the variation in A, B and C is shown in the table.

Aims and Objectives

- How long the various algorithms would take?
- Count number of times two values compared
- Count number of times two values swapped

Algorithmic Performance

There are two aspects of algorithmic performance[1]:

- Time
  - Instructions take time.
  - How fast does the algorithm perform?
  - What affects its runtime?
- Space

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 3, Issue 2, March 2014

- Data structures take space
- What kind of data structures can be used?
- How does choice of data structure affect the runtime?

We will focus on time:

– How to estimate the time required for an algorithm

Mathematical vs Empirical Analysis [1]

Mathematical Analysis	Empirical Analysis
The algorithm is analyzed with the help of mathematical deviations and there is no need of specific input.	The algorithm is analyzed by taking some sample of input and no mathematical deviation is involved.
The principal weakness of these types of analysis is its limited applicability.	The principal strength of Empirical analysis is it is applicable to any algorithm.
The principal strength of Mathematical analysis is it is independent of any input or the computer on which algorithm is running.	The principal weakness of Empirical analysis is that it depends upon the sample input taken and the computer on which the algorithm is running.

General Plan for Empirical Analysis of Algorithms [3]

1. Understand the purpose of experiment of given algorithm.
2. Decide the efficiency matrix M. Also decide the measurement. For example operation's count vs. time.
3. Decide on characteristic of input.
4. Create a program for implementing the algorithm. This program is ready experiment.
5. Generate a sample of input.
6. Run the algorithm for some set of input sample. Record the result obtained.
7. Analyze the resultant data.

Analytical Comparison

A limitation of the empirical comparison is that it is system-dependent. A more effective way of comparing algorithms is through their time complexity upper bound to guarantee that an algorithm will run at most a certain time with order of magnitude $O(f(n))$ where n is the number of items in the list to be sorted[7]. This type of comparison is called asymptotic analysis. The time complexities of the algorithms studied are shown in below table:

Algorithm	Time Complexity		
	Best Case	Average Case	Worst Case
Bubble Sort	$O(n)$	$O(n\text{^}2)$	$O(n\text{^}2)$
Insertion Sort	$O(n)$	$O(n\text{^}2)$	$O(n\text{^}2)$
Selection Sort	$O(n.\lg(n))$	$O(n\text{^}2)$	$O(n\text{^}2)$
Quick Sort	$O(n.\lg(n))$	$O(n.\lg(n))$	$O(n\text{^}2)$
Shell Sort	$O(n.\lg(n))$	$O(n.\lg(n))$	$O(n\text{^}2)$
Heap Sort	$O(n.\lg(n))$	$O(n.\lg(n))$	$O(n.\lg(n))$

Table1: Time Complexity of Sorting Algorithms

Although all algorithms have a worst-case runtime of $O(n\text{^}2)$, only Quicksort & Shell Sort has a best and average runtime of $O(n.\lg(n))$ [4]. This means that Quicksort & Shell Sort, on average, will always be faster than Bubble, Insertion and Selection sort, if the list is sufficiently large.

Empirical comparison

a. Tests made

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 3, Issue 2, March 2014

The tests were made using the C++. Each algorithm was run on the lists of length of 1, 3, 5, 7, 10, and 15 lakhs. The number of comparisons and number of Assignment/Swap operations was recorded by using a counter for number of comparisons and number of Assignment/Swap operations [6]. The code was run on Windows 7, with an Intel Core i5 processor and 3GB of RAM. The raw results were recorded by the reading and writing in the file. These raw results were tabulated, Calculated, and graphed using C++ and MS-Excel.

b. Results

i. Total Results

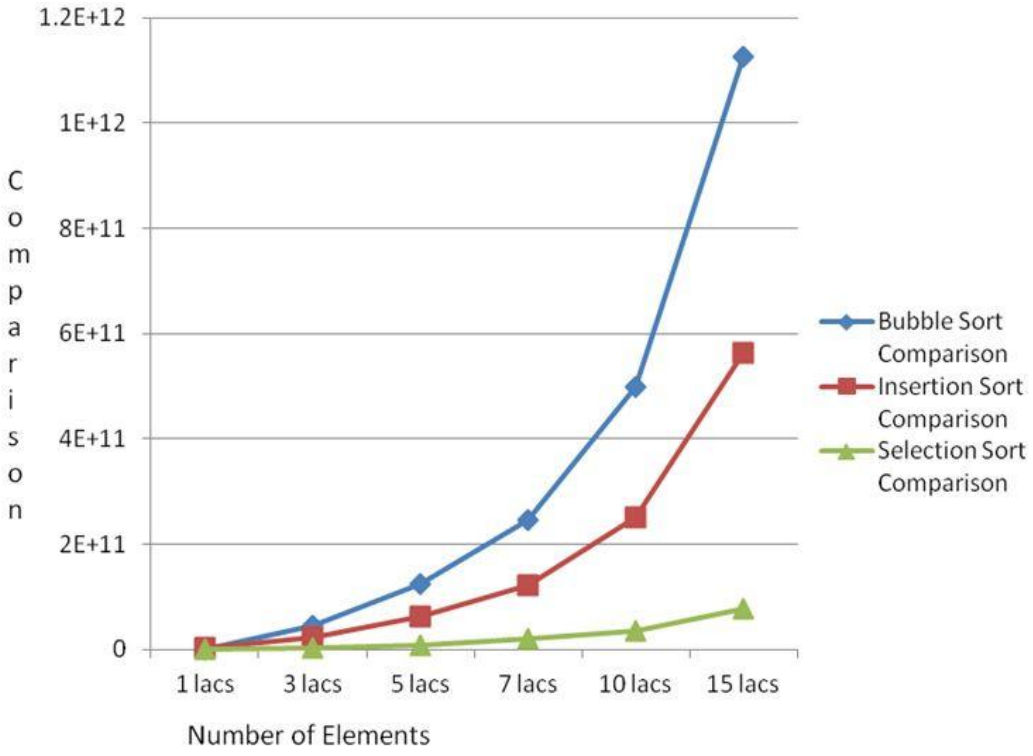
The total results for all runs for each algorithm and each list length are shown on Table and Graph: Table for number of comparisons of sorting algorithm on given number of elements:

Algo.	Number of elements					
	1 lacs	3 lacs	5 lacs	7 lacs	10 lacs	15 lacs
Bubble Sort	499950001	44999850001	124999750001	244999650001	499999500001	1124999250001
Insertion Sort	2503921057	22500033726	62489124089	122464377656	249931402775	562246099741
Selection Sort	717121978	2040124110	8248512308	21247437765	35983140378	76314509973
Quick Sort	2083013	7154514	12439847	18266103	25881883	41478603
Shell Sort	1868928	6075712	10475712	15052412	21951424	33902848
Heap Sort	5173930	16938139	29321482	42113995	61645978	95209435

Table 2: Number of comparisons for sorting algorithms

Graph drawn from values obtained from Table 2:

Graph for Comparison of $N\text{^}2$ Sorting Algorithm



Graph for Comparison of $N.\lg(N)$ Sorting Algorithm

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 3, Issue 2, March 2014

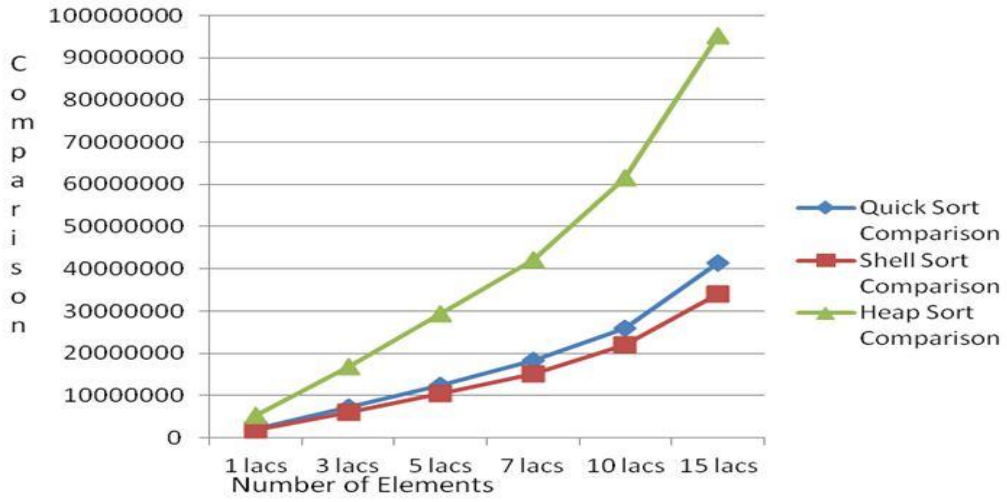


Fig 1: Total number of comparisons for sorting algorithms

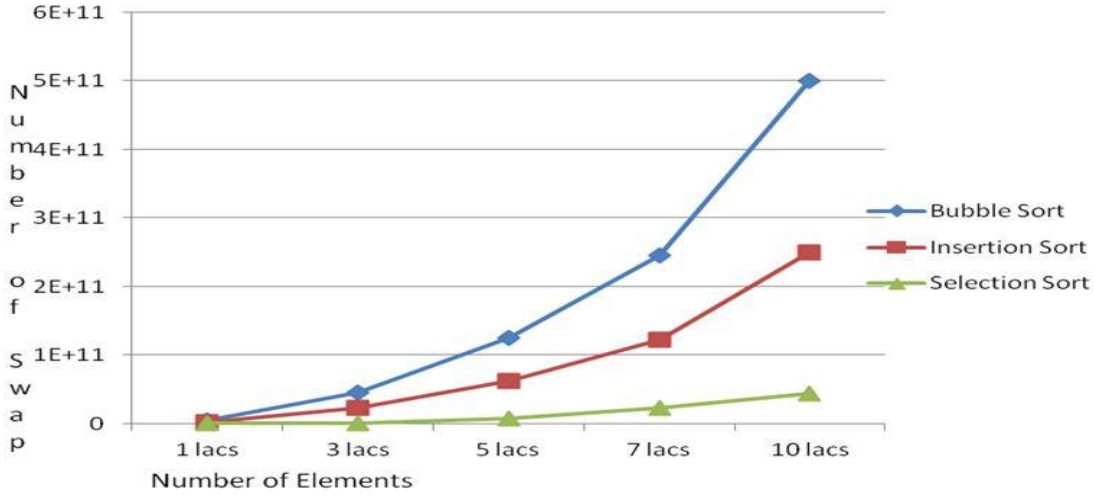
Table for number of Swap/Assignment of sorting algorithm on given number of elements:

Algo.	Number of elements					
	1 lacs	3 lacs	5 lacs	7 lacs	10 lacs	15 lacs
Bubble Sort	4999950001	44999850001	124999750001	244999650001	499999500001	1124999250001
Insertion Sort	2503921057	22500033726	62489124089	122464377656	249931402775	562246099741
Selection Sort	740393104	750103362	8258923407	23246435765	44693141271	67224709954
Quick Sort	1026729	3609352	6335691	8665471	12252113	22301526
Shell Sort	1668928	5475712	9475712	13651424	19951424	30902848
Heap Sort	1674642	5496045	9523826	13687997	20048658	30986477

Table 3: Number of Swap/Assignment operations for sorting algorithms

Graph from tabular data is:

Graph for number of Swap of $N\text{^}2$ Sorting Algorithm



Graph for number of Swap of $N.\lg(N)$ Sorting Algorithm

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 3, Issue 2, March 2014

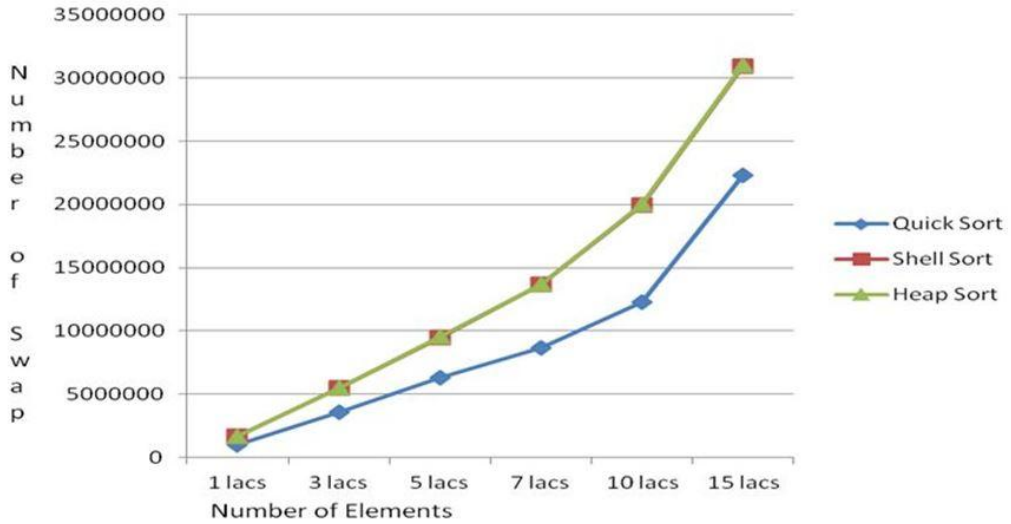


Fig. 2: Total number of Swap/Assignment for sorting algorithms

Table for time taken (in mili Seconds) for sorting algorithm on given number of elements:

Algo.	Number of elements					
	1 lacs	3 lacs	5 lacs	7 lacs	10 lacs	15 lacs
Bubble Sort	303	2414	3619	5534	6899	7462
Insertion Sort	42	432	689	1172	1363	1689
Selection Sort	22	583	814	1261	1429	1932
Quick Sort	9	36	148	203	256	321
Shell Sort	13	18	22	26	44	55
Heap Sort	448	3281	5375	6411	7357	8121

Table4: Total time taken for sorting algorithms

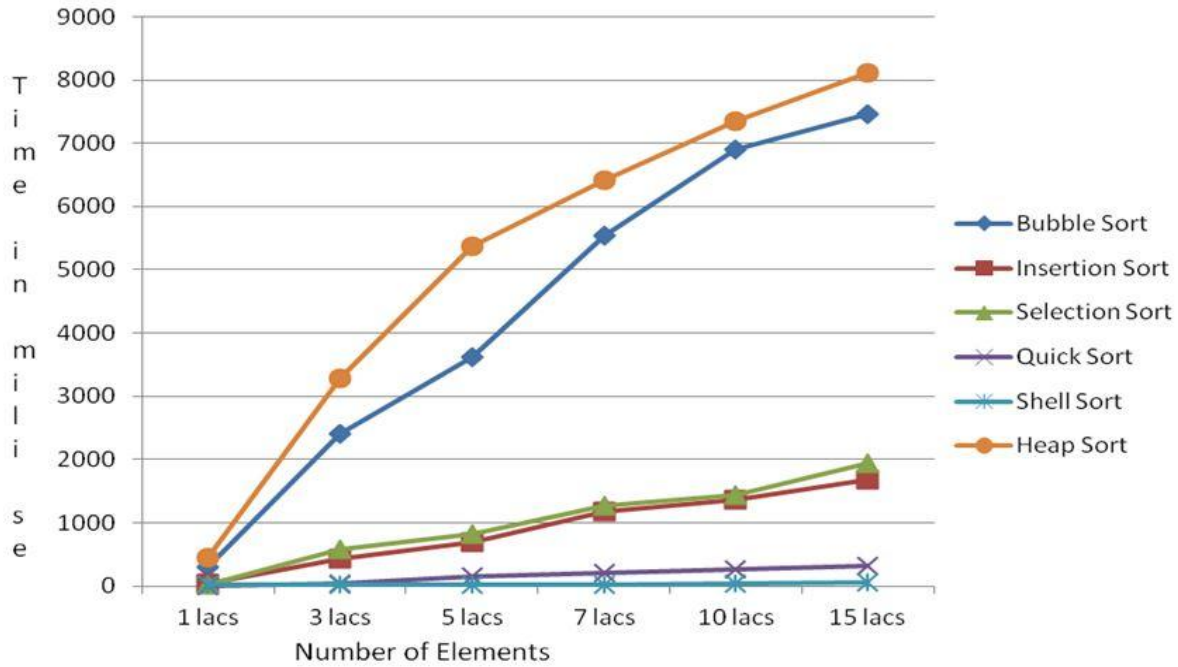


Fig. 3: Total time taken for sorting algorithms

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 3, Issue 2, March 2014

Calculate Results

The calculated results (least square fitting) for the above results of Bubble and Quick sorting algorithm are shown on Table and Graph. Here X is number of elements and Y is number of comparisons.

1. Bubble Sort

X (No. of elements)	**Y** (No. of Comparison)	X^2	X^3	X^4	$Y.X$	$Y.X$^2
100000	4.9999E+9	1.0E+10	1E+15	1E+20	4.99995E+14	4.99995E+19
300000	4.4998E+10	9E+10	2.7E+16	8.1E+21	1.35E+16	4.04999E+21
500000	1.25E+11	2.5E+11	1.25E+17	6.25E+22	6.24999E+16	3.12499E+22
700000	2.45E+11	4.9E+11	3.43E+17	2.401E+23	1.715E+17	1.2005E+23
1000000	5E+11	1E+12	1E+18	1E+24	5E+17	5E+23
1500000	1.125E+12	2.25E+12	3.375E+18	5.0625E+24	1.6875E+18	2.53125E+24
∑X=**4100000**	∑Y=2.045E+12	$\sum X^2$=4.09E+12	$\sum X^3$=4.871E+18	$\sum X^4$=6.373E+24	∑YX=2.4355E+18	$\sum YX^2$=3.18665E+24

Table 5 : Bubble Sort calculation using Least square fitting method

$$6a1 + 4100000a2 + 4090000000000a3 = 2044997950006$$
$$4100000a1 + 4090000000000a2 + 4871000000000000000a3 = 2435497955004100000$$
$$4090000000000a1 + 4871000000000000000a2 + 6.3733E + 24a3 = 3.18664756450409E + 24$$

Calculating the values of a1, a2, a3 by using Matrix Inversion Method:

$$a1 = 0.98144531250000, a2 = -0.50000003352761, a3 = 0.49999999999999$$
$$y = 0.98144531250000 - 0.50000003352761x + 0.49999999999999x2$$

Calculated values from the above values are:

X	**Y**
10	45.9814449772238999999999999
500	124750.981428546000000
1000	499500.981411775000000
1500	1124250.981395000000000
2000	1999000.981378220000000

Graph from above tabular data is:

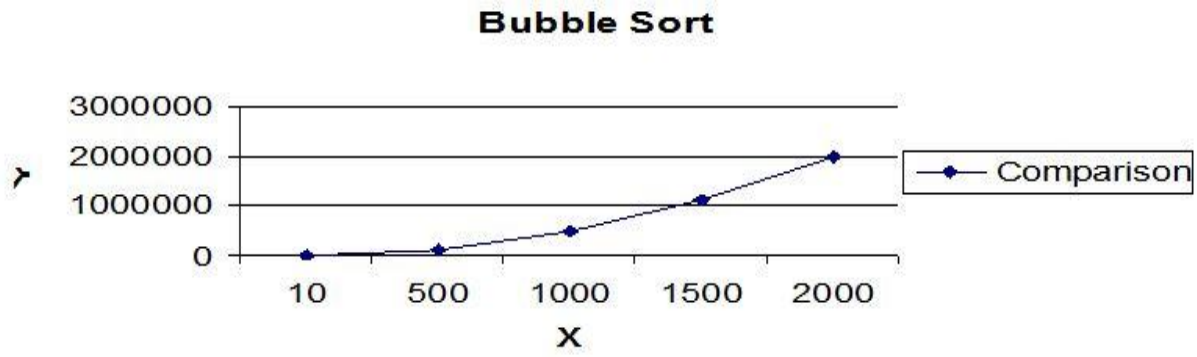


Fig. 4: Bubble Sort Graph for X-Y values.

2. Quick Sort

Number of Elements	100000	300000	500000	700000	1000000	1500000
Number of Comparison	2083013	7154514	12439847	18266103	25881883	41478603

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 3, Issue 2, March 2014

$500000\log 500000a + 500000b + c = 12439847$

$9465784.28466208A + 500000B + C = 12439847$

$19931568.5693241A + 1000000B + C = 25881883$

$30774796.605068A + 1500000B + C = 41478603$

Solving the above equations using Matrix Inversion Method we get the values of A, B and C as:

$A = 5.7086227916717434$

$B = -12.6063574003357$

$C = 235321.896$

$Y = 5.7086227916717434X.\lg X - 92.606357400272373X + 4706433.79167138$

X	Y
10	235385.468770355
500	254609.857557621
1000	279606.441907242
1500	311257.710256986
2000	335308.233398483

Table 10: Quick Sort calculation using simple matrix inversion method

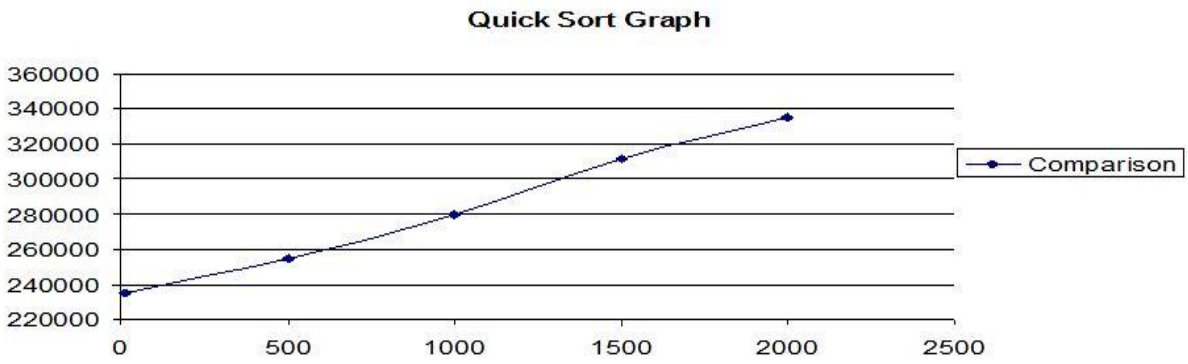


Fig. 9: Quick Sort Graph for X-Y values.

SUMMARY

Based on the above calculations the values calculated can be tabulated as follows:

Sorting Algorithm	A	B	C
Shell Sort	1.2603520355992	-2.909018983	260352.0356
Heap Sort	3.2825049996173	-3.500006479	279490.9996
Quick Sort	5.70862279167174	-12.60635740033	235321.896
Insertion Sort	-9.2744141	95.404689602554	0.24983064989564
Bubble Sort	0.98144531250000	-0.50000003352761	0.49999999999999

Table 11: Summary of calculations of Sorting Algorithms.

Graph showing Polynomial Equations Comparison for $N.\lg(N)$ algorithms:

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 3, Issue 2, March 2014

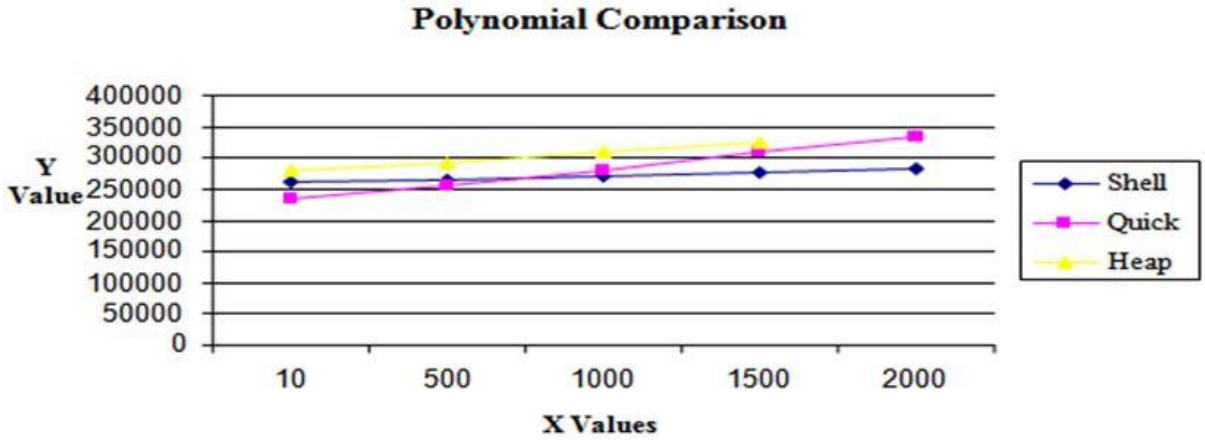


Fig. 10: Polynomial Equation Comparison for $N.\lg(N)$ algorithm.

Graph showing Polynomial Equations Comparison for $N.\lg(N)$ algorithms:

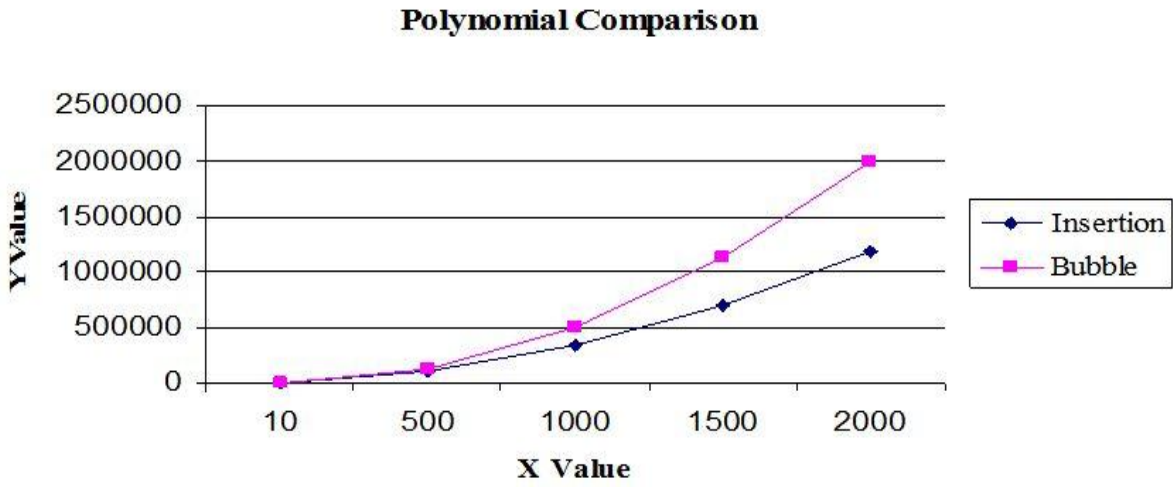


Fig. 11: Polynomial Equation Comparison for N^2 algorithm.

Based on the calculated value of A, B and C the values of X and Y can be tabulated as follows:

Table for calculated no. of Comparisons for sorting algorithm on given number of elements:

Algo.	Number of elements				
	10	500	1000	1500	2000
Bubble Sort	45.98144497722	124750.9814285	499500.9814118	1124250.981395	1999000.981378
Insertion Sort	969.7555469151	110150.7328611	345226.0650841	705216.7222549	1190122.704374
Quick Sort	235385.4687704	254609.8575576	279606.4419072	311257.7102570	335308.2333985
Shell Sort	260364.81340	264547.54835	270003.41313	275934.98991	282175.49473
Heap Sort	279565.04199	292456.11223	308703.72986	326190.30849	474409.0364815

Table12: No. of Calculated Comparisons using simple matrix inversion method

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 3, Issue 2, March 2014

II. CONCLUSION

The empirical data obtained by using the program reveals the speed of each algorithm, from fastest to slowest for very large list and ranks as follows:

1. Quicksort
2. Shell Sort
3. Heap sort
4. Insertion sort
5. Selection sort
6. Bubble sort

There is a large difference in the time taken to sort very large lists between the fastest three and the slowest three. This is due to the efficiency of Quick Sort, Shell Sort and Heap sort have over the others when the list to sort is sufficiently large.

ACKNOWLEDGMENT

I wish to convey my sincere thanks and gratitude to **All the teachers,** and our internal thesis report guide for all the help they extended to me while completing the thesis report. Their expert guidance, timely suggestions and inspiration brought life in this paper. I am grateful to them for their cooperation and remarkable suggestion in preparing this thesis report.

REFERENCES

[1] Donald E. Knuth et al. "The Art of Computer Programming," Sorting and Searching Edition 2, Vol.3.

[2] Cormen et al. "Introduction to Algorithms," Edition 3, 31 Jul, 2009.

[3] Ahmed M. Aliyu, Dr. P. B. Zirra, "A Comparative Analysis of Sorting Algorithms on Integer and Character Arrays," The International Journal Of Engineering And Science (IJES)., ISSN(e): 2319 – 1813 ISSN(p): 2319 – 1805.

[4] John Harkins, Tarek El-Ghazawi, Esam El-Araby, Miaoqing Huang, "Performance and Analysis of Sorting Algorithms on the SRC 6 Reconfigurable Computer," in The George Washington University, 2 Nov. 2005.

[5] Wikipedia,(2007) :Sorting Algorithm, Retrieved from http://en.wikipedia.org/wiki/Sorting_algorithm: 24-05-2013 .

[6] Wikipedia,(2007) :Selection Sort, Retrieved from http://en.wikipedia.org/wiki/Selection_sort 26-05-2013.

[7] Robert L(2002): Data Structures and Algorithms, 2nd Ed.24-28.

AUTHOR BIOGRAPHY

Mr. Mahfooz Alam
M.Tech-CSE
University School of Information and Communication Technology (USICT)
Guru Gobind Singh Indraprastha University

Mr. Ayush Chugh
B.Tech-IT
University School of Information and Communication Technology (USICT)
Guru Gobind Singh Indraprastha University