

Empirical Performance of Internal Sorting Algorithm

Abstract

Internal Sorting Algorithms are used when the list of records are small enough to be maintained entirely in primary/physical memory for the duration of the sort, while External Sorting Algorithms are used when the list of records are large enough to be maintained in physical memory hence a need for external/secondary storage for the duration of the sort. Almost all operations carried out by computing devices involve sorting and searching which employs Internal Sorting Algorithms for better performance. Over the years, researchers have given more attention to External Sorting Algorithm in respect to sorting large data set; meanwhile, little attention has been given to Internal Sorting Algorithm which is also of great importance in modern computing. In this study, an empirical analysis of Six Internal Sorting Algorithms (bubble, insertion, quick shaker, shell and selection) was performed using sample comprising of 100 to 50,000 lists of integer values. Using C++ time function, it was observed that insertion sort has the best performance on small sample say between 100 to 400. But when the sample size increases to 500, Shaker sort has better performance. Furthermore, when the sample grows above 500, shell sort outperforms all the internal sorting algorithms considered in the study. Meanwhile, selection sort has displayed the worst performance on data samples of size 100 to 30,000. As the samples size grows to further to 50,000 and above, the performance of shaker sort and bubble sort depreciates even below that of selection sort. And when the sample size increases further from 1000 and above then shell sort should be the best choice for sorting

Keyword: *Internal Sorting, empirical analysis, complexity analysis, Space complexity, Time complexity*

1. Introduction

Sorting algorithm is an algorithm that rearranges data elements of a list in certain user defined order. Sorting can be classified in terms of the data-size and the type of memory required for a sorting computation. When the list of data-size is small enough to be accommodated at a time in the primary memory we refer to such sorting algorithm as internal sorting. On the order hand, when the list of data-size is large to an extent that will require secondary memory other than primary memory are refer as external sorting [8].

Internal sorting algorithms (ISAs) are very prevalent in practical and real world situations. This is because most practical problems in computing require a specific arrangement of outputs. Most often you need to sort data in order to normalize it, and make querying efficient. There are many sorting algorithms and some programmers tend to develop some with the intent of getting faster algorithms [1]. The efficiency of an algorithm is estimated in terms of execution time and the amount of memory it requires. This execution time of an algorithm which is also known as time complexity is the amount of time required for the algorithm to be executed. It is worthy to note that programming language used in the implementation of an algorithm and quality of compiler does not affect time complexity of algorithms [1]. This does not rule out the challenge faced by programmers in which ISAs to use in a particular situation because some are faster than others under different situations. It therefore becomes necessary to evaluate algorithms to know their behavior on different instances. This enables programmers know the precise algorithm to apply to a particular situation at a particular time. Several authors have based their study on performance of external sorting algorithm using large data set. It is also important to know which ISA work on small

data samples like telephone number, salary of workers, age of students, matriculation number of students etc.

This study therefore, evaluates the performance of ISAs using empirical analysis not the Asymptotic Complexity Analysis since all the algorithms are internally sorted algorithms [10]. Though, a little discussion was done on asymptotic complexity analysis to alternate the limitations of empirical analysis which is system dependent, [18] more complexity analysis techniques are available [19, 17, 21].

2. Review of Related Literature

Over the years, Empirical analysis has become a crucial part of the study of algorithms [17]. Five features were considered in the process of analyzing an algorithm empirically, these features include; Correctness of the algorithm, work done, and space used, simplicity or clarity and optimality [28]. Some of these features are discussed in details in [30, 31, 32, 33]. Empirical analysis entails; (1) Understanding the theoretical analysis. (2) Decide on what features are to be measured: in this study we measured Time and space complexity, stability and adaptability. (3) What is the most appropriate hardware to run the measurement on: we used the processor speed, internal memory size and operating system type. (4) Which is the most appropriate implementation language: we chose C++ to implement ISAs. (5) Which is the most appropriate data structure to use. (6) Implement the algorithms for comparison, (7) implement some form of timing device. (8) Create the input dataset necessary to produce the measurement we need. (9) Measure the performance of the algorithm on the different input dataset created to meet the aim of the analysis. (10) interpret the result and relate to the theoretical analysis [16, 29].

It is worth knowing that, at the course of performing empirical analysis of ISA, the execution time of implementing a given algorithm depends on the CPU processing capacity, compilation rate of the Compiler, programming language used in the implementation, algorithm construction and implementation, memory access time of input/output and whether the operating system is multitasking of single tasking [16].

3. Overall analysis of Time and Space Complexity with Stability and Adaptability of ISA

Time complexity analysis is an effective way of comparing algorithms through their time complexity, which depends upon the number of comparisons made when an algorithm runs at magnitude of $O(f(n))$. Where n is the number of items in the list to be sorted [18, 22, 23, 25].

Table 1: Time/Space Complexity and Stability/Adaptability of ISAs [18, 26, 27]

		Time Complexity			Auxiliary Space Complexity	Stability	Adaptability
		Best Case	Average Case	Worst Case			
ISA	Insertion Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$	Yes	Yes
	Quick Sort	$\Omega(n \log n)$	$\Theta(n \log n)$	$O(n^2)$	$O(n)$	No	Yes
	Selection Sort	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$	No	No
	Bubble Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$	Yes	Yes
	Shaker Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$	Yes	
	Shell Sort	$\Omega(n \log n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$	No	Yes

Table 1 shows that Quick sort has better performance in its best case ($\Omega(n \log n)$) and average case ($\Theta(n \log n)$) runtime compared to the rest of the ISAs. Shell sort is next with best case of $\Omega(n \log n)$. It is therefore obvious that selection sort will not be a preferred choice over other sorting algorithms with its best case ($\Omega(n^2)$), average case ($\Theta(n^2)$) and worst case ($O(n^2)$) respectively. Therefore, empirical analysis would be handy to provide a better comparability outcome.

4 Empirical analysis

Empirical analyses are based on executing the algorithms on a computer system rather than using asymptotic notations. To achieve that, the system used for the experiment runs on windows 8.1, with Intel Pentium (R), CPU speed of 2.80GHz, 4GB RAM and 64bits OS system type.

The data samples selected were inserted into an array of size from 100 to 50,000 elements. In each implementation, a “CLOCKS_PER_SEC” function in C++ library was used to estimate the execution time of ISA. Each of the algorithms was run for 100 trials and an average was calculated by dividing the sum of execution time of trials by the number of trials.

The time it takes to sort each sorting algorithm in respect to the data was recorded and represented as shown in table 2.

5. Experimental Result and Interpretation

Table 2 presents the outcome of the test conducted on each of the ISA with various data categories.

Table 2: Time comparison (sec) for ISA on sizes (100 to 50,000) of data samples

		Data Sample Size							
		100	500	1,000	1,500	5,000	10,000	30,000	50,000
ISAs	Insertion	0.015	0.046	0.094	0.140	0.451	0.938	3.317	6.25
	Quick	0.016	0.047	0.141	0.156	0.516	0.937	2.796	5.35
	Selection	0.047	0.187	0.312	0.468	1.453	2.829	8.484	14.172
	Bubble	0.016	0.047	0.094	0.140	0.547	1.328	6.734	15.985
	Shaker	0.016	0.048	0.097	0.156	0.546	1.252	5.94	13.532
	Shell	0.016	0.047	0.093	0.141	0.453	0.906	2.743	4.552

Taking a critical look at table 2, each data sample column has its lowest and highest execution time value in bold and italics. The lowest values represent best execution time and the fastest algorithm in respect to that particular data sample size, while the highest values represent the worst execution time and the slowest algorithm in respect to that particular data sample size. The result on table 2 shows that Selection sort is the slowest ISA since it has the highest value all through the data samples. But at 50,000 data sample, bubble sort became slower than selection sort since it has the largest value. This means bubble sort performance depreciates with increase in data sample size. On the other hand, Shell sort has the least execution time all through the data samples making it the fastest ISA both with small and large data samples. The visual representation of this result is shown on figure 3.

Figure 1: Execution time performance between ISAs over 50,000 data samples

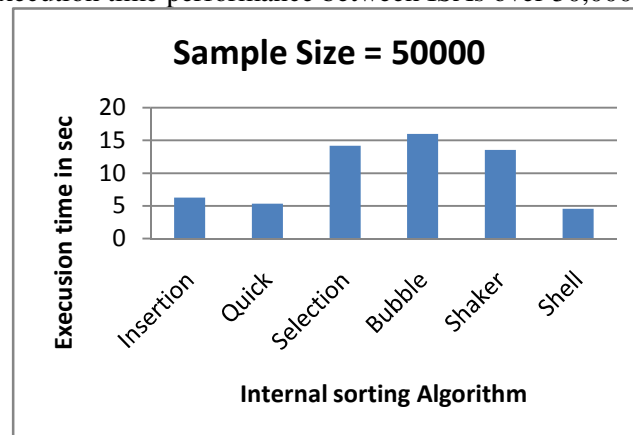


Figure 1 validates the fact that the performance of bubble sort, followed by shaker sort compared to selection sort depreciates as data sample size increases from 50,000 and above. However, shell sort remains the best choice of ISA when working with large sample size.

Figure 2: Comparison of ISA across data sample sizes

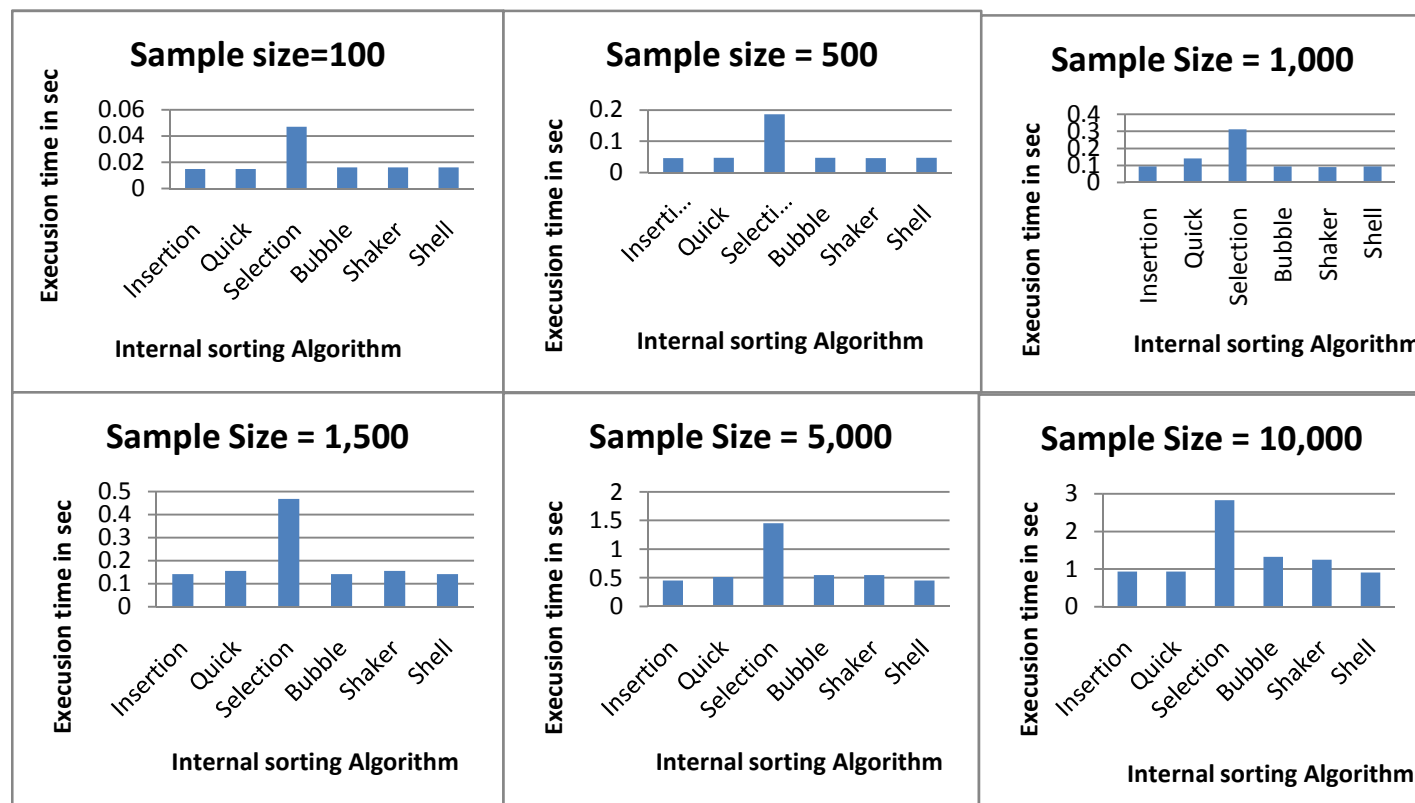
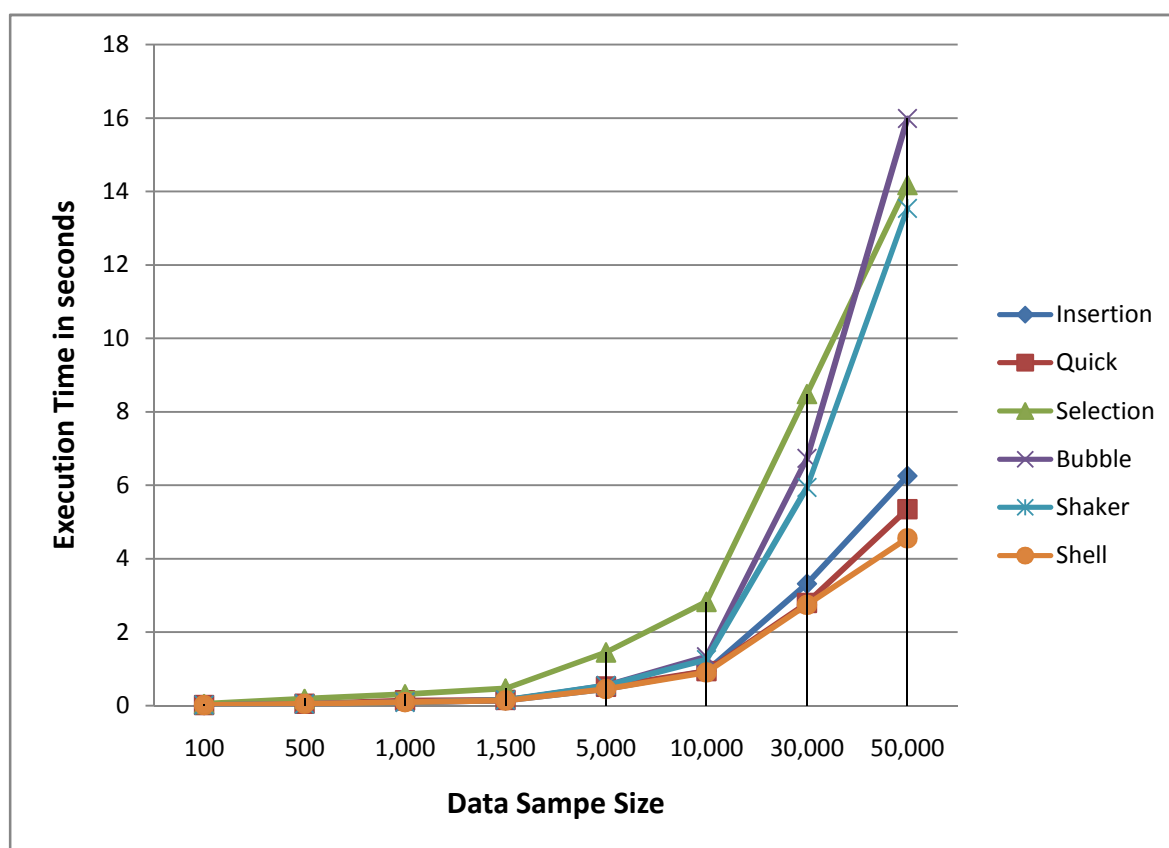


Figure 2 shows the comparison of each data sample size used in the study. Considering chart of sample size 10,000, the performance of quick sort improves compared to bubble sort. While considering chart of sample size 1000, bubble and shaker sort shows better performance compared to quick sort. However selection sort displayed a poor performance compared to all data sizes considered in the study. Overall, shell sort has the best performance (with least sorting time) across the data samples used in the study.

113 Figure 3: General overview of relationship between data size and execution time in sec of ISAs



115 The relationship between the time taken for sorting and the amount of element is nonlinear as seen in
 116 figure 3. Comparing the ISA, a clear variation of execution time can be seen.

117 The performance of the entire data sample used in the study (100, 500, 1000, 1500, 10,000, 30,000, and
 118 50,000) for each of the ISA is shown in figure 3. The study observes that insertion sort has the best
 119 performance for small data. Therefore, Insertion sort is much faster in computing less random numbers of
 120 small size than any of the ISA considered in the study. However shell sort outperforms insertion sort as
 121 the data sample increases.

122 5. Further Study

123 This study was carried out with a single computing device. In the future, researchers could use different
 124 computing resources with varying computing speed to compare the effect of processor speed on these
 125 data samples. Also, only integers were used as data sample; it is the interest of the researchers to know
 126 what will happen to character arrays in respect to internal sorting in the future.

127 6. Conclusion

128 In this paper, we have tested six ISAs on random data sample of sizes from 100 to 50,000. We apply the
 129 six ISA on each of the data samples sizes and compare their performance in each case. And also we have
 130 found out best and worst case with respect to execution time. All the ISA implemented in C++ language.
 131 The programs is designed on CodeBlocks 10.5 IDE with C++ 5.02 compiler and executed on Intel
 132 Pentium (R) processor, and the programs running at 2.80GHz clock speed.
 133 Insertion sort should be considered ahead of every other ISAs when data sample is small (say less than
 134 100 data sample). But when the data sample increases from 100 to 500, shaker sort should be considered

ahead of insertion sort. And when the sample size increases further from 1000 and above then shell sort should be the best choice for sorting.

7 References

- [1] Ahmed, M. A., Zirra ,P. B. A. Comparative Analysis of Sorting Algorithms on Integer and Character Arrays. The International Journal of Engineering and Science (IJES), Volume 2, Issue 7, Pages 25-30. 2013
- [2] Tim, B. Sorting Algorithms as Special Cases of a Priority Queue Sort, SIGCSE'11, March 9–12, Dallas, Texas, USA. Copyright 2011 ACM 978-1-4503-0500. 2011.
- [3] Ghiath, A. “A New Priority-Sort Based Optimization Algorithm Integrated Process Planning and Scheduling”, International Journal of Modeling and Optimization, Vol. 3, No. 2, April. 2013
- [4] Jehad H, “A Comparative Study between Various Sorting Algorithms”, IJCSNS International Journal of Computer Science and Network Security, VOL.1 5 No.3, March 2015.
- [5] Donald K.E. The Art of Computer Programming: Volume 3: Sorting and Searching (2nd Edition) 2nd Edition.1998
- [6] Vinther, K. Engineering Cache-Oblivious Sorting Algorithms, Available on <https://users-cs.au.dk/gerth/advising/thesis/kristoffer-vinther.pdf>, Retrieve 04/05/2016. 2003.
- [7] Kelvin, W. The Development of an Aid for Teaching Sorting Algorithms to Students. Available on <http://www.cs.bath.ac.uk/~mdv/courses/CM30082/projects.bho/2003-4/KevinWhiteDissertation.pdf>, Retrieved 04/05/2016. 2004.
- [8] Munro J.I., Paterson M. S Selection and Sorting with limited Storage: volume 12: Theoretical Computer Science. page (315 - 323). 1980.
- [9] Sorting Algorithms, Page 9.Available on <http://www.saylor.org/site/wp-content/uploads/2011/06/Sorting-Algorithm.pdf> Retrieved 10/05/2016
- [10] Clifford,A.S. Theory of Algorithms. Available on <http://courses.cs.vt.edu/cs5114/spring2010/notes.pdf>, Retrieved on 15/05/2016. 2010
- [11] Access on Available on <http://web.cs.wpi.edu/~mebalazs/cs507/slides07/slides.html>
- [12] Thomas Niemann. Sorting and Searching Algorithms: A Cookbook
- [13] Knuth, Donald. E. The Art of Computer Programming, Volume 3, Sorting and Searching. Addison-Wesley, Reading, Massachusetts,1998
- [15] Mahfooz Alam, Ayush Chugh. Sorting Algorithm: An Empirical AnalysisInternational Journal of Engineering Science and Innovative Technology (IJESIT) Volume 3, Issue 2, March 2014
- [16] Brunskill, D,. Turner, J. Understanding Algorithms and Data Structures. M CGraw-Hill, Maidenhead, England, 1996. 1997 Reprint
- [17] Sedgewick ,R., Flajolet, P. An Introduction to the Analysis of Algorithms. Addison-Wesley, Reading,MA, 1996.

- 171 [18] Juliana Pena Oampo. An Empirical Comparison of the runtime of five sorting algorithms ;Colegio
172 Colombo Britanico, Columbia,2008
- 173 [19] Purdom, P. W., Jr., Brown, C. The Analysis of Algorithms. Holt, Rinehart and Winston, New
174 York, 1985.
- 175 [20] Daniel H. Greene, Donald E. Knuth (Boston: Birkhäuser, 1990). Mathematics for the Analysis of
176 Algorithms, Third Edition available on <http://www-cs-faculty.stanford.edu/~uno/gk.html>,
- 177 [21] Niraj Kumar Singh, Soubhik Chakraborty. Smart Sort: Design and Analysis of a Fast, Efficient
178 and Robust Comparison Based Internal Sort Algorithm; Cornell University Library
179 arXiv:1204.5083, 2012
- 180 [22] Jagdeep , S., Alka, S., Garima, S. An Experimental Study of performance Comparison for various
181 parallel sorting algorithms; International journal of Computer Science and Information
182 Technology, vol 7 (2) 738-740, ISSN:0975-9646, 2016
- 183 [23] Miraj Gul, Noorul Amin, Sulaiman, M. An Analytical Comparison of Different Sorting
184 Algorithms in Data Structure; International Journal of Advanced Research in Computer Science
185 and Software Engineering, Vol (5), ISSN: 2277128X, 2015
- 186 [24] Mahfooz A ., Ayush C. Sorting Algorithm: An Empirical Analysis; International Journal of
187 Engineering Science and Innovative Technology (IJESIT) Volume 3, Issue 2, March 2014
- 188 [25] Robert L. Data Structures and Algorithms in Java (2nd Edition) 2nd EditionISBN-13: 075-
189 2063324530 ISBN-10: 0672324539 ,2002
- 190 [26] Neetu F,. Satya Prakash G. Analysis and Testing of Sorting Algorithms on a Standard Dataset;
191 2015
- 192 [27] Al-kharabshah Khalid S., Alturani I.M., Alturani, A. M. I, and Zanoon N. I. Review on sorting
193 algorithms A Comparative Study. International Journal of Computer Science and Security (IJCSS)
194 vol 7.3 page 120. 2013
- 195 [28] Baase S. Computer Algorithms: Introduction to Analysis and Design. Anderson Wesley, Reading,
196 Massachusetts second edition, 1988
- 197 [29] Ian Sanders. Empirical Analysis of algorithms is easy, (or is it?), ACM SIGCSE Bulletin, 2002
198 accessed on 21/08/2016, 2002
- 199 [30] Brassard,G., Bratley,P.Fundamentals of Algorithmic. Prentice Hall, Englewood Cliffs, NJ, 1996
- 200 [31] Cormen, T. H., Leieron, C. E., Rivest, R. L. Introduction to Algorithms. The MIT Press,
201 Cambridge, Massachusetts, 1990. Sixteenth Printing, 1996
- 202 [32] Neapolitan, R. Naimipour, K. Foundations of Algorithms. Jones and Bartlett, Sudbury, MA, 1997
- 203 [33] Manber, U.. Introduction to Algorithms: A Creative Approach. Addison Wesley, Reading,
204 Massachusetts, 1988