BINARY TREE SORT IS MORE ROBUST THAN QUICK SORT IN AVERAGE CASE

Soubhik Chakraborty*[1] and Rajarshi Sarkar[2]

[1]Department of Applied Mathematics, BIT Mesra, Ranchi-835215, India
[2]Department of Computer Science & Engineering , BIT Mesra, Ranchi-835215, India

Email of the corresponding author: soubhikc@yahoo.co.in

ABSTRACT

Average case complexity, in order to be a useful and reliable measure, has to be robust. The probability distribution, generally uniform, over which expectation is taken should be realistic over the problem domain. But algorithm books do not certify that the uniform inputs are always realistic. Do the results hold even for non uniform inputs? In this context we observe that Binary Tree sort is more robust than the fast and popular Quick sort in the average case.

KEY WORDS

Binary Tree sort, robustness, expectation, probability distribution, average case complexity

1. Introduction

Average Case complexity is both interesting and intriguing part of algorithm analysis. It is interesting as some algorithms with bad worst case complexity performs better on the average like Quick sort and Binary Tree sort, the latter being investigated here. It is intriguing as for a complex code it is difficult to identify the pivotal operation or the pivotal region in the code for taking mathematical expectation for finding the average case complexity. Moreover, the probability distribution over which expectation is taken (which is generally uniform) may not be realistic over the domain of the problem. For comparison based sorting algorithms, the dominant operation turns out to be comparison. But algorithm books do not certify that the uniform inputs are always realistic. It is quite possible that the inputs come from non uniform distributions such as Binomial. Do the results derived assuming uniform inputs hold even for non uniform inputs? In other words, is the average case complexity having a robustness appeal? Quick sort unfortunately is not very robust (Sourabh and Chakraborty [1]). But our study shows that Binary Tree sort which has average and worst case complexity similar to those of Quick sort, namely O(nlogn) and O(n^2) respectively, is quite robust . Using computer experiments, a series of runs of a code for various inputs, we verify that the model $y = a+bx+error$ where $x = n\log_2 n$ and y is average (mean) sorting time, does hold remarkably well even for non uniform inputs like Binomial, Poisson and Standard Normal. For a comprehensive literature on sorting, we refer to

Knuth[2]. For a sound theoretical discussion on sorting with reference to the input probability distribution, Mahmoud[3] should be consulted. The next section gives the algorithm of Binary Tree sort. In section 3 we provide the statistical analysis. Section 4 gives a brief discussion on the results. Section 5 is reserved for conclusion.

2. Algorithm for Binary Tree Sort[4][5]:

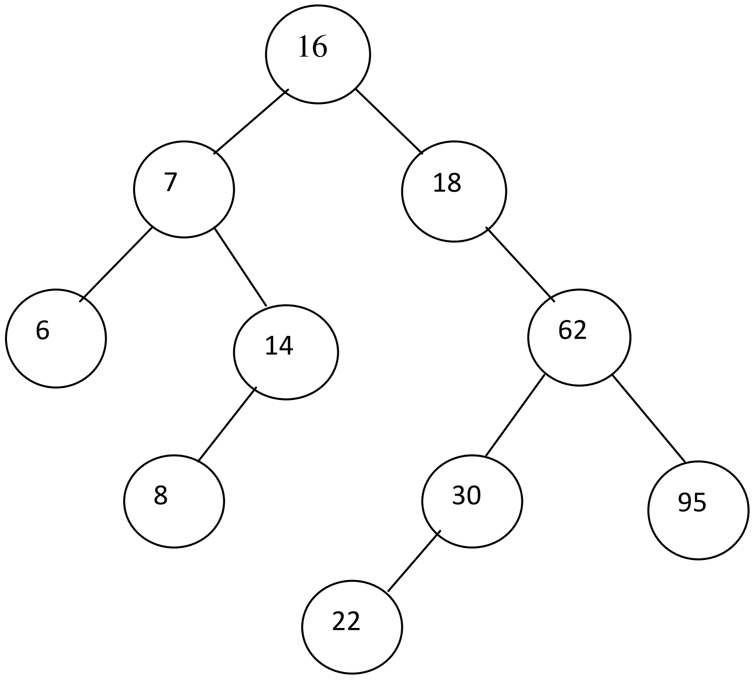

Fig.1.Binary Search Tree

a) In order to build the binary search tree, we begin with the 0th element 16 which we take as the root of the tree.
b) While inserting the 1st element, i.e. 7, 7 is compared with its root element 16 and being less than 16 it is made the left child of the root node 16.
c) While inserting the 2nd element of the list, i.e. 18, 18 is compared with the root element 16.But 18 is greater than 16, so it is made the right child of the root node 16.
d) Proceeding in a similar fashion, all the elements are appropriately placed in the binary search tree.
e) Now to get the elements in the sorted order, the tree is traversed in in-order. The in-order traversal of the binary search tree lists the elements in ascending order.

In-order traverse:

1) Traverse the left subtree of the root R in in-order.
2) Process the root R.
3) Traverse the right subtree of R in in-order.

3. Statistical Analysis

Our experimental work deals with certain empirical results in Binary tree sort algorithm. Here we make a statistical case study on the robustness of average complexity measures, which are derived assuming uniform distribution, for non-uniform inputs(both discrete and continuous).For simulating different variates, we consulted Kennedy and Gentle[6].

We have done a robustness study of Binary tree sort algorithm to verify whether the O(nlogn) complexity holds in the average case if the input comes from distributions other than uniform. We primarily include Monte Carlo Simulation (where inputs from different probability distributions will be simulated) and statistical modeling using simple linear regression with the size of input as the predictor and average sorting time as the response variable. We are taking 100 trials for calculating the average time of sorting different discrete and continuous distribution inputs of sample size n. The observation have been taken on fixed parameters for different distribution, but with varying sample size n in the range [5000, 50000]. The standard non uniform probability distributions investigated are Binomial, Poisson and Normal, the first two being discrete while the last one is continuous. We will of course be investigating discrete uniform and continuous uniform distributions also for the sake of completeness.

System Specification:

Processor : Intel(R) Celeron(R) CPU 2.13GHz
Hard Disk : 80GB
RAM : 256MB
Operating System : Windows XP

3.1 Average Case Complexity of Binary Tree Sort for Binomial Distribution Inputs:

The Binomial distribution Binomial (m, p) inputs are taken with parameters m and p, where m=100 and p=0.5 are fixed. The empirical results are shown in table 1 and fig. 2.

Table 1: Table of mean sorting time and standard deviation (both in seconds) of Binomial distribution inputs for Binary tree sort

N	n logn	Mean time	SD
5000	18494.85	0.0086	0.0078
10000	40000.00	0.0140	0.0139
15000	62641.37	0.0273	0.0146
20000	86020.60	0.0389	0.0234
25000	109948.50	0.0483	0.0279
30000	134313.64	0.0542	0.0326
35000	159042.38	0.0602	0.0385
40000	184082.40	0.0691	0.0402
45000	209394.56	0.0742	0.0475
50000	234948.50	0.0825	0.0502

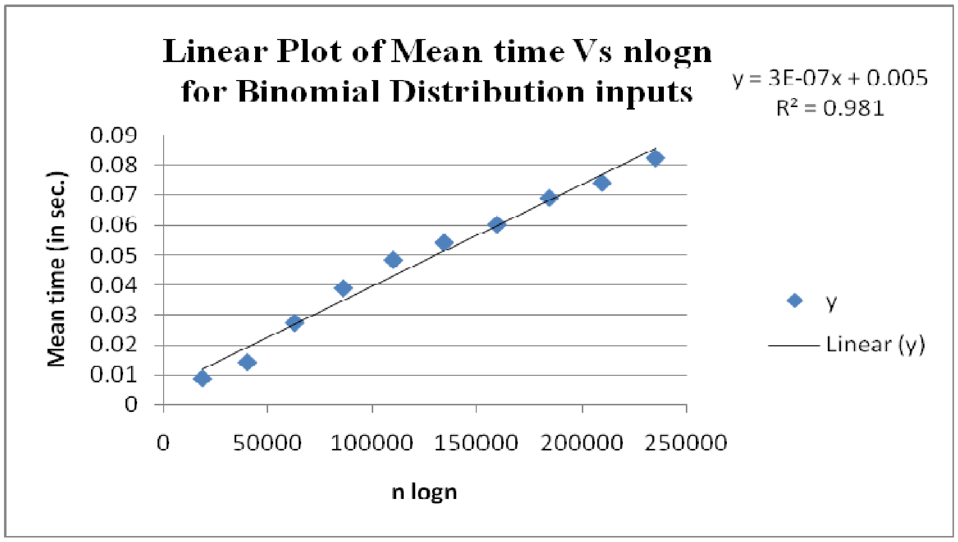

Fig. 2 Plot showing empirical O(nlogn) complexity for binomial inputs

Experimental results as shown in fig.2 are supporting O(n logn) complexity. We write $y_{avg}(n)=O_{emp}(nlogn)$.

3.2. Average Case Complexity for Poisson Distribution Inputs:

The Poisson distribution Poisson (μ) inputs are taken with parameter μ where μ=1 is fixed. The empirical results are shown in table 2 and fig. 3.

Table 2: Table of mean sorting time and standard deviation (both in sec.) of Poisson distribution inputs for Binary tree sort

n	n logn	Mean time	SD
5000	18494.85	0.0032	0.0064
10000	40000.00	0.0048	0.0073
15000	62641.37	0.0158	0.0121
20000	86020.60	0.0220	0.0159
25000	109948.50	0.0314	0.0139
30000	134313.64	0.0346	0.0251
35000	159042.38	0.0391	0.0199
40000	184082.40	0.0436	0.0250
45000	209394.56	0.0543	0.0291
50000	234948.50	0.0591	0.0302

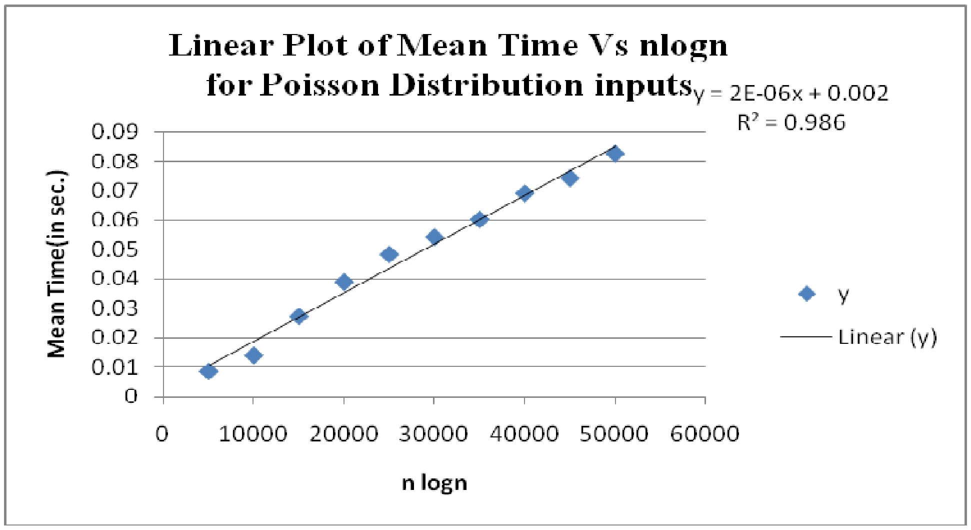

Fig.3 Plot showing empirical O(nlogn) complexity for Poisson inputs

Experimental results as shown in above fig. 3 are supporting O(n logn) complexity .We write y_{avg} (n)=O_{emp}(nlogn).

3.3. Average Case Complexity for Discrete Uniform Distribution Inputs:

The Discrete Uniform [1, 2, 3….., k] inputs are taken with parameter k where k=50 is fixed. The empirical results are shown in table 3 and fig.4.

Table 3: Table of mean sorting time and standard deviation (both in sec.) of Discrete Uniform distribution inputs for Binary tree sort

N	n logn	Mean time	SD
5000	18494.85	0.0031	0.0062
10000	40000.00	0.0110	0.0072
15000	62641.37	0.0124	0.0116
20000	86020.60	0.0190	0.0154
25000	109948.50	0.0217	0.0173
30000	134313.64	0.0297	0.0176
35000	159042.38	0.0296	0.0202
40000	184082.40	0.0390	0.0234
45000	209394.56	0.0564	0.0418
50000	234948.50	0.0602	0.0378

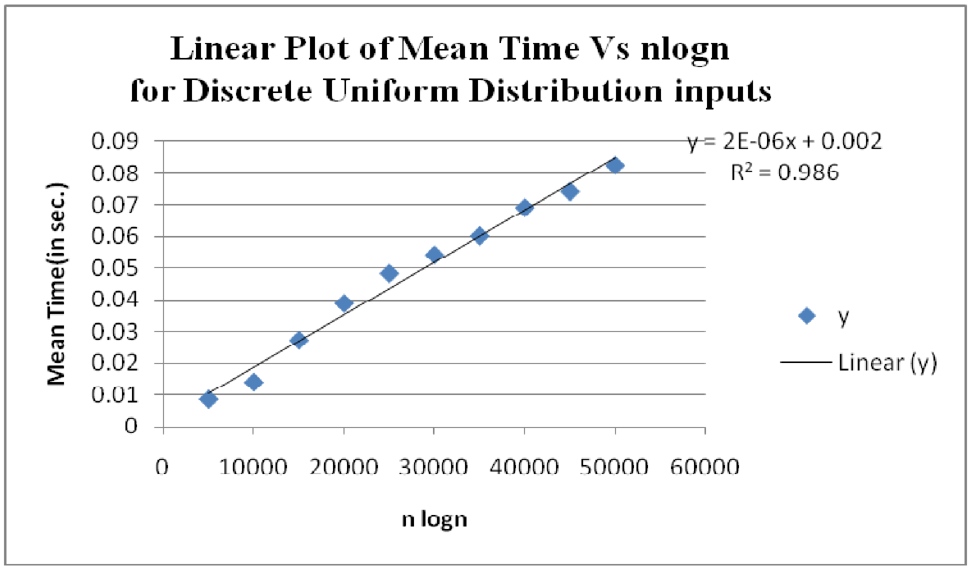

Fig. 4 Plot showing empirical O(nlogn) complexity for Discrete Uniform inputs

Experimental results as shown in above fig.4 are again supporting O(n log n) complexity. We write y_{avg} (n)=O_{emp}(nlogn).

3.4. Average Case Complexity for Continuous Uniform Distribution Inputs:

The Continuous Uniform distribution Continuous Uniform [0, θ] inputs are taken with parameter mean θ,where θ =1 is fixed. The empirical results are shown in table 4 and fig.5.

Table 4: Table of mean sorting time and standard deviation (both in sec.) of Continuous Uniform distribution inputs for Binary tree sort

N	n logn	Mean time	SD
5000	18494.85	0.0031	0.0062
10000	40000.00	0.0080	0.0080
15000	62641.37	0.0173	0.0109
20000	86020.60	0.0201	0.0099
25000	109948.50	0.0266	0.0172
30000	134313.64	0.0313	0.0157
35000	159042.38	0.0375	0.0224
40000	184082.40	0.0453	0.0256
45000	209394.56	0.0484	0.0284
50000	234948.50	0.0545	0.0296

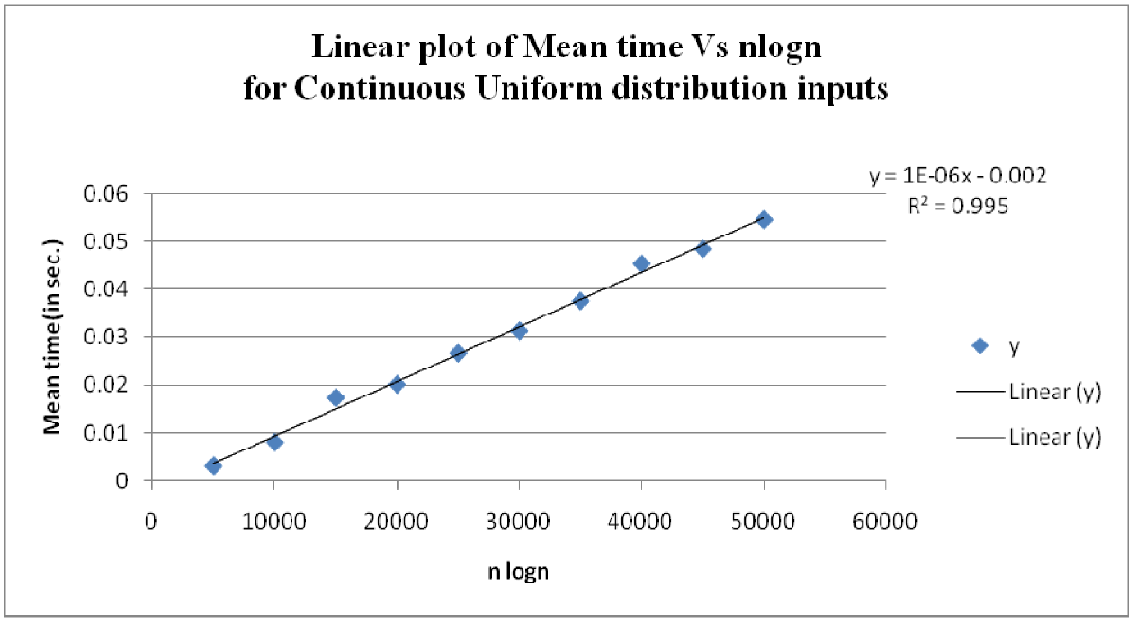

Fig. 5. Plot showing empirical O(nlogn) complexity for Continuous Uniform inputs

Experimental results as shown in above fig.5 are supporting O(n log n) complexity. We write y_{avg} (n)=O_{emp}(nlogn).

3.5. Average Case Complexity for Standard Normal Distribution Inputs:

For the Standard Normal distribution inputs, the empirical results are shown in table 5 and fig.6.

Table 5: Table of mean sorting time and standard deviation (both in sec.) of Standard Normal distribution inputs for Binary tree sort

N	n logn	Mean time	SD
5000	18494.85	0.0000	0.0000
10000	40000.00	0.0062	0.0076
15000	62641.37	0.0064	0.0078
20000	86020.60	0.0064	0.0078
25000	109948.50	0.0092	0.0075
30000	134313.64	0.0115	0.0067
35000	159042.38	0.0150	0.0000
40000	184082.40	0.0156	0.0004
45000	209394.56	0.0175	0.0065
50000	234948.50	0.0185	0.0056

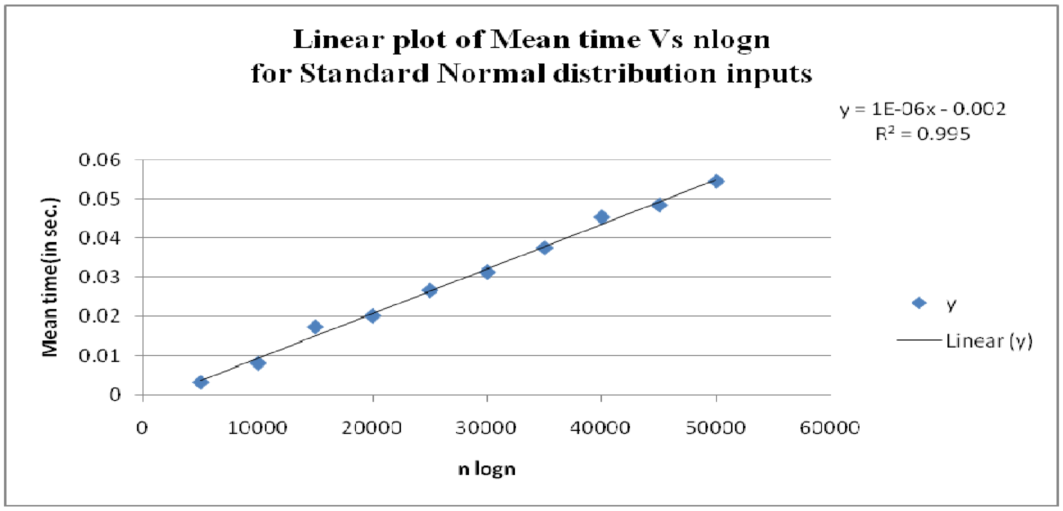

Fig.6 Plot showing empirical O(nlogn) complexity for Standard Normal inputs.

Experimental results as shown in above fig.6 are supporting O(n log n) complexity. We write y_{avg} (n)=O_{emp}(nlogn).

Remark: Box Muller transformation was used for simulating standard normal variates (Kennedy and Gentle [6]).

4. Discussion

There are a few points to be discussed. There is an important result which says average case complexity under the universal distribution equals worst case complexity [7]. So for those algorithms whose worst case complexity is different (higher or worse) than average case complexity, it is risky to say “this algorithm has a bad worst case but performs better on the average” without verifying the robustness. Since Binary tree sort is robust while Quick sort is not, we can use Binary tree sort in case there is a possibility that the sorting elements need not be uniform. The other important question is: why did we work on time which is system dependent? The answer to this question is that time of a computing operation is actually the weight (and not the count) of the operation and time of the entire code is a weighted sum of the computing operations, the weights being the corresponding times. A mathematical complexity bound (like big oh) is count based and operation specific. A bound that is weight based and consequently allows for mixing of different types of operations (like mixing comparisons with interchanges in sorting) is essentially a statistical bound. Empirical O, written as O with a subscript emp, is only an empirical estimate of the statistical bound obtained over a finite range by supplying numerical values to the weights. And these numerical values are obtained by running computer experiments. So design and analysis of computer experiments become crucial especially when the response is a complexity such as time. For a general literature on computer experiments we refer to the seminal paper by Sacks et. al. [8] and the book by Fang, Li and Sudjianto [9]. A recent book which gives a computer experiment outlook to algorithmic complexity is by Chakraborty and Sourabh [10]. This is the first published book on statistical complexity bounds.

5. Conclusion

The robustness of average case measure O(nlogn) for Binary tree sort has be verified by observing mean time. This may be crossed checked against the mean comparisons experimentally counted rather than working on time directly. Only for algorithms such as Heap sort where worst and average complexities are equal, it is safe to rely on an average complexity measure. In case average case complexity seems to be better than the same for worst case, the robustness must be confirmed first as otherwise it could be a dangerous practice to rely on it. The recent rejection of some proofs (see [11] in addition to [1] for example) is testimony to this fact. And this is precisely where Binary Tree sorts beats the fast and popular Quick sort.

References

[1] S. K. Sourabh and S. Chakraborty, How robust is quicksort average complexity? arXiv:0811.4376v1 [cs.DS]
[2] D. E. Knuth, The Art of Programming, Vol. 3: Sorting and Searching, Pearson Edu. reprint, 2000
[3] H. Mahmoud, Sorting: A Distribution Theory, John Wiley, 2000
[4] Y. Kanetkar, Data Structures through C, BPB publications, New Delhi, 2nd Ed.,2009
[5] S. Lipschutz (Adapted by G. A. V. Pai), Data Structures, Schaum's Outlines Series, Tata McGraw-Hill Publishing Company Ltd. 2006.
[6] W. Kennedy and J. Gentle, Statistical Computing, Marcel Dekker Inc., 1980
[7] .M. Li and P.M.B Vitanyi, Average case complexity under the universal distribution equals worst case complexity, Inf . Proc. Lett., 42,no.3,1992,145-149
[8] Sacks, J., W. Welch, T. Mitchell and H. Wynn, Design and Analysis of Computer Experiments, Statistical Science, vol. 4(4), 1989
[9] Fang, K. T., Li, R. and Sudjianto, A., Design and Modeling of Computer Experiments, Chapman and Hall, 2006
[10] Chakraborty, S. and Sourabh, S. K., A Computer Experiment Oriented Approach to Algorithmic Complexity, Lambert Academic Publishing, 2010
[11] S. Chakraborty, S. K. Sourabh, How Robust are Average Complexity Measures? A Statistical Case Study, Applied Math. and Compu., vol. 189(2), 2007, 1787-1797