A Methodological Review for the Analysis of Divide & Conquer Based Sorting/ Searching Algorithms

Mr. Deepak Abhayankar[1] and Mrs. Maya Ingle[2]

**[1]School of Computer Science, Devi Ahilya University
Indore, M.P. 452017, India**

**[2]School of Computer Science, Devi Ahilya University ,
Indore, M.P. 452017, India**

Abstract

This paper develops a practical methodology for the analysis of sorting/searching algorithms. To achieve this objective an analytical study of Quicksort and searching problem was undertaken. This work explains that asymptotic analysis can be misleading if applied slovenly. The study provides a fresh insight into the working of Quicksort and Binary search. Also this presents an exact analysis of Quicksort. Our study finds that asymptotic analysis is a sort of approximation and may hide many useful facts. It was shown that infinite inefficient algorithms can easily be classified with a few efficient algorithms using asymptotic approach.

1. Introduction

There have been abundant computer applications which need sorting/searching as a key component. Since SQL operations use it as an internal database subroutine, all database applications gain advantage of an efficient sorting/searching algorithm. Also sorting/searching is a must for some rudimentary database operations like a creation of indices and binary searches. Sorting is functional in operations like finding closest pair, determining an element's uniqueness, finding k^{th} largest element, and identifying membership. Many practical applications in computational geometry need sorting. For instance sorting is used to find the convex hull in computational geometry [10]. Applications that need sorting/searching include supply chain management, bioinformatics and computer graphics. Since sorting/searching problem has a lot of importance in real world, hence it will be fruitful to evolve a practical framework or methodology for analysis of sorting algorithms.

This paper develops an intuitive framework or methodology for the analysis of sorting/searching algorithms. To achieve this objective an analytical study of Quicksort and searching problem was carried out. This effort explains that asymptotic analysis can be misleading if applied carelessly. This study provides a fresh insight into the working of Quicksort and Binary search. Also this study presents an exact analysis of Quicksort. Although there already exist a few average case analyses, majority of the attempts finish up as asymptotic analysis. Our study finds that asymptotic analysis is a sort of approximation and may hide many useful facts such as large constant factors which make any algorithm insane for practical purposes. It was shown that infinite inefficient algorithms can easily be classified with a few efficient algorithms using asymptotic approach.

2. Searching an Analytic Study

It is not difficult to design a set of binary search like divide and conquer searching algorithms which lead to following recurrence.

$T(n) = c + T(nk/k+1)$

$T(1) = d$

Master theorem suggests the solution of the recurrence relation is $T(n) = O(\log n)$.

For k = 1 we will have recurrence relation for binary search. For k = 2 one gets ternary search. One of the observations of this study is that for k > 1 we can produce a sequence of increasingly inefficient algorithms by incrementing the value of 1. But asymptotic analysis puts all the algorithms in the same set. In fact all algorithms can flaunt logarithmic time complexity. It is the constant factor that differs. The key conclusion is that constant factor matters and one cannot blindly trust the asymptotic order. The algorithm designer has to examine the situation thoughtfully. Too high a constant factor will render an algorithm useless with certainty. This endeavor finds that an exact analysis may provide better insight than what asymptotic analysis may offer.

3. Probabilistic Analysis of Quicksort

3.1 Review of Probabilistic Analysis

In probability theory, a probabilistic arrangement is defined by a sample space S and a probability measure p. The points of the sample space are the possible result of the experiment and are called elementary events. An event is a subset of the sample space. For instance, one event we may care about is the event that the first die comes up 1. Another is the event that the two dice sum to 7. The probability of an event is just the sum of the probabilities of the elementary events contained inside it [9].

A random variable is a function from elementary events to integers or reals. For instance, another way we can talk formally about these dice is to define the random variable Y1 representing the result of the first die, Y2 representing the result of the second die, and Y = Y1 + Y2 representing the sum of the two. We could then ask: what is the probability that Y = 6? [9].

One property of a random variable we often care about is its expectation. For a discrete random variable Y over sample space S, the expected value of Y is: E[Y] = Pr(e1) Y[e1] + Pr(e2) Y[e2] +..............Pr(en) X[en] for all e∈S. An important fact about expected values is Linearity of Expectation: for any two random variables U and V, E[U+V] = E[U] + E[V]. This fact is incredibly important for analysis of algorithms because it allows us to analyze a complicated random variable by writing it as a sum of simple random variables and then separately analyzing these simple RVs[9].

3.2 Probabilistic Analysis of Quicksort with Accurate Results

Theorem 1The expected number of comparisons made by randomized Quicksort on an array of size n is Hn(2n+2) – 4n, where Hn = (1+ (½) +(1/3) +............(1/n)).

Let us consider one of the random variables is Y_{ij}'s for i < j. Denote the i^th smallest element in the array by e_i and the jth smallest element by e_j. If the pivot we choose is between e_i and e_j then these two end up in different buckets and machine will never compare them to each other. If the pivot we choose is either e_i or ej then Computer does compare them. If the pivot is less than e_i or greater than e_j then both e_i and e_j end up in the same bucket and we have to pick another pivot. So, one can think of this like a dart game: we throw a dart at random into the array: if we hit e_i or e_j then Y_{ij} becomes 1, if we hit between e_i and e_j then Y_{ij} becomes 0, and otherwise we throw another dart. At each step, the probability that Y_{ij} = 1 conditioned on the event that the game ends in that step is exactly 2/(j − i + 1). Therefore, overall, the probability that Y_{ij} = 1 is 2/(j − i + 1).

$$Y = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} Y_{ij}$$

$$E[Y] = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} E[Y_{ij}]$$

$$E[Y] = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} \frac{2}{j-i+1}$$

Up to this stage we follow what the other researchers have already done [9], and from this point we move in the direction of exact value rather than a crude upper bound.

$$E[Y] = \left(\frac{2}{2}\right)(n-1) + \left(\frac{2}{3}\right)(n-2) + \left(\frac{2}{4}\right)(n-3) + \cdots + \left(\frac{2}{n-1}\right)(2) + \left(\frac{2}{n}\right)(1)$$

$$E[Y] = \left(\frac{2}{2}\right)(n-1) + \left(\frac{2}{3}\right)(n-2) + \cdots + \left(\frac{2}{n-1}\right)(n-(n-2)) + \left(\frac{2}{n}\right)(n-(n-1))$$

$$E[Y] = \left(\frac{2}{2}n - \frac{2}{2}\times 1\right) + \left(\frac{2}{3}n - \frac{2}{3}\times 2\right) + \cdots + \left(\frac{2}{n-1}n - \frac{2}{n-1}\times(n-2)\right) + \left(\frac{2}{n}n - \frac{2}{n}\times(n-1)\right)$$

$$E[Y] = 2n(H_n - 1) - 2(n - H_n)$$

$$E[Y] = H_n(2n+2) - 4n$$ ---------------- (Equation A)

Equation A is one of the central contributions of the paper. Equation A gives the exact value of the expected number of comparisons performed by Quicksort. If a researcher is inclined towards asymptotic approach s/he can easily have it. For the researchers, who are inclined towards asymptotic approach and approximate results, E[Y]=O(nlogn). Because H_n is approximately log n, E[Y] becomes O(log n).

3.3 Alternative Analysis

This section is basically a byproduct of the overall study. It is a bit crude but effective technique for asymptotic analysis. Quicksort partition may divide the array into two partitions. One of the partitions may be empty. If there are two partitions then either both are of same size or one of them will be larger than the other one. We are interested in upper bound on the average case time. Size of the Non smaller partition may vary from (n-1) to (n-1)/2. Average size of Non Smaller partition was found to be approximately (3n/4). Along the same line if we estimate average size of Non large partition we get approximately (n/4). This leads to following recurrence relation.

$T(n) = T(n/4) + T(3n/4) + (n-1)$.

Application of Recursion tree approach recommends that solution is $O(n \log n)$.

4. Results and Conclusion

Evidence of the analysis of a set of divide and conquer search algorithms suggests that asymptotic analysis can easily mislead. Exact analysis is a better option than asymptotic approach. Asymptotic analysis may play a side role but it cannot replace exact analysis. If exact mathematical analysis is not feasible then only approximations and asymptotic can play the key role. References preferred to provide only the asymptotic analysis; this study seems to be unique to go beyond asymptotic analysis and to provide an exact analysis of Quicksort. Moreover this study produces one more alternative asymptotic analysis.

References

[1] D. E. Knuth, The Art of Computer Programming, Vol. 3, Pearson Education, 1998.
[2] C. A. R. Hoare, "Quicksort," Computer Journal5 (1) , 1962, pp. 10-15.
[3] S. Baase and A. Gelder, Computer Algorithms: Introduction to Design and Analysis, Addison-Wesley, 2000.
[4] J. L. Bentley, "Programming Pearls: how to sort," Communications of the ACM, Vol. Issue 4, 1986, pp. 287-ff.
[5] R. Sedgewick, "Implementing quicksort Programs," Communications of the ACM, Vol. 21, Issue10, 1978, pp. 847-857.
[6]T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001.
[7] G. S. Brodal, R. Fagerberg and G. Moruz, "On the adaptiveness of Quicksort," Journal of Experimental AlgorithmsACM, Vol. 12, Article 3.2, 2008.
[8] N. Wirth, Algorithms and Data Structures, © N. Wirth 1985 (Oberon version: August 2004)
[9]http://www.cs.cmu.edu/afs/cs/academic/class/15451-s10/www/Probabilisticanalysis, Randomized Quicksort-1-July-2011
[10] J. Chhugani, W. Macy, A. Baransi, A.D. Nguyen, M. Hagog, S. Kumar, V. W. Lee, Y. K. Chen, P. Dubey, “Efficient Implementation of Sorting on Multi-Core SIMD CPU Architecture , “ Journal Proceedings of the VLDB Endowment, Volume 1, Issue 2, August 2008.