ON THE ROBUSTNESS OF K-SORT AND ITS COMPARISON TO QUICK SORT IN AVERAGE CASE

Mita Pal and Soubhik Chakraborty

Department of Applied mathematics, B.I.T, Mesra, Ranchi-835215, India

ABSTRACT: The present paper examines the robustness of the average case O (nlogn) complexity on K-sort, a new version of quick sort. In our first study we reconfirm this through computer experiments. A computer experiment is a series of runs of a code for various inputs. A deterministic computer experiment is one which produces identical results if the code is re-run for identical inputs. Our second study reveals that K-sort is the better choice for discrete uniform distribution U(1, 2, ..., k) inputs whereas quick sort is found better for continuous uniform distribution U(0,1) inputs. Interestingly, increasing k which decreases the ties is good for quick sort but bad for K-sort.
KEYWORDS: Cauchy distribution, Computer experiment, K-Sort, Robustness, average complexity

1.INTRODUCTION

In our previous work [S+11a], we introduced K-sort, a modified version of quick sort that removes the interchanges, and found it to be having a more robust average case O(nlogn) complexity for negative binomial inputs. In this paper, we reconfirm this through computer experiments for inputs from Cauchy distribution for which expectation theoretically does not exist. The reconfirmation is crucial as some recent studies show that average case complexity derived for uniform inputs need not necessarily hold for non uniform inputs (see for example [CS07a] and [SC08]) or that expectation is applied to an operation that appears to be dominant but is not so (e.g. comparisons, and not multiplication, is dominant in Schoor's matrix multiplication algorithm [CS07b]). When in doubt as to which operation or which region in the code is dominant (this is difficult to predict for a complex code), one suggestion could be replacing the count based mathematical bound (which is also unfortunately operation specific) by a weight based statistical bound that a permits *collective consideration of all operations* and then estimate it by directly working on time, regarding the time consumed by an operation as its weight. In other words, the credibility of this statistical bound estimate (called empirical O) depends on the design and analysis of our *computer experiment*. A computer experiment is a series of runs of a code for various inputs. A deterministic computer experiment is one which produces identical results if the code is re-run for identical inputs[S+89]. A recent book on computer experiments with complexity as the response variable is [CS10]. For a comprehensive account on sorting and searching algorithms, see [Knu00]. For sorting with special emphasis on the input probability distribution, see [Mah00].

In the present work, a comparison is made of average time between two algorithms, quick sort and K-sort, on the discrete uniform distribution U(1, 2, ..., k) with probability 1/k for each variates' value and continuous uniform distribution U(0,1) inputs. The experimental results exhibit that the K-sort has an edge over quick sort for discrete uniform distribution inputs. However, for continuous uniform distribution inputs it is the quick sort which performs better.

2. ALGORITHM

The steps of K-sort [S+11a], [S+11b] are given below:-

Step-1: Initialize the first element of the array as the key element and i as left, j as (right+1), k = p where p is (left+1).
Step-2: Repeat step-3 till the condition (j-i) ≥ 2 is satisfied.
Step-3: Compare a[p] and key element. If key ≤ a[p] then
Step-3.1: if (p is not equal to j and j is not equal to (right + 1))
then set a[j] = a[p]
else if (j equals (right + 1)) then
set temp = a[p] and flag = 1
decrease j by 1 and assign p = j
else (if the comparison of step-3 is not satisfied i.e. if key > a[p])
Step-3.2: assign a[i] = a[p] , increase i and k by 1 and set p = k
Step-4: set a[i] = key
if (flag = = 1) then
assign a[i+1] = temp
Step-5: if (left < i - 1) then Split the array into sub array from start to i-th element and repeat steps 1-4

with the sub array
Step-6: if (left > i + 1) then
Split the array into sub array from i-th element to end element and repeat steps 1-4 with the sub array

3. EMPIRICAL RESULTS

First of all we would focus on reconfirming the robustness of average complexity of K-sort. Next we examine the behavior of K-sort when inputs come from discrete uniform distribution U(1, 2, …, k) and continuous uniform distribution (0,1) and compare the same with quick sort.

3.1. Reconfirming the robustness of average complexity of K-sort

Theorem 1: If U_1 and U_2 are two independent uniform U [0, 1] variates then Z_1 and Z_2 defined below are two independent Standard Normal variates:

$$Z_1 = (-2\ln U_1)^{1/2} \cos(2\pi U_2)$$
$$Z_2 = (-2\ln U_1)^{1/2} \sin(2\pi U_2)$$

This result is called Box Muller transformation.

Theorem 2: If Z_1 and Z_2 are two independent standard Normal variates then Z_1 / Z_2 is a standard Cauchy variate. For more details, we refer to [KG80]. We all know that for Cauchy distribution, expectation does not exist theoretically [GK02]. Hence it is not possible to know the average case complexity theoretically for inputs from this distribution. Working directly on time, using computer experiments, we have obtained an empirical O (nlogn) complexity in average sorting time for K-sort for Cauchy distribution inputs which we simulated using theorems 1 and 2 given above. This result goes a long way in reconfirming that K-sort's average complexity is robust unlike quick sort [SC08]. It is of interest to note that K-sort beats Heap Sort for n ≤ 70 lakhs [S+11a] and it is easier to program K-sort as compared to heap sort. As a final comment, we strongly insist that for those algorithms which have bad worst case but better average case, the robustness of the average case must be tested because average case under universal distribution equals worst case [MV92].
Table 1 and Fig 1 based on table 1 summarize our results.

Table 1: Average time for K-sort for Cauchy distribution inputs (average taken over 500 trials)

n	nlogn	Average time (Sec)
3000	10431.36	0.006
6000	22668.91	0.016
9000	35588.18	0.047
12000	48950.17	0.069
15000	62641.37	0.109
18000	76594.91	0.145
21000	90766.61	0.188
24000	105125.1	0.235
27000	119646.8	0.297
30000	134313.6	0.355

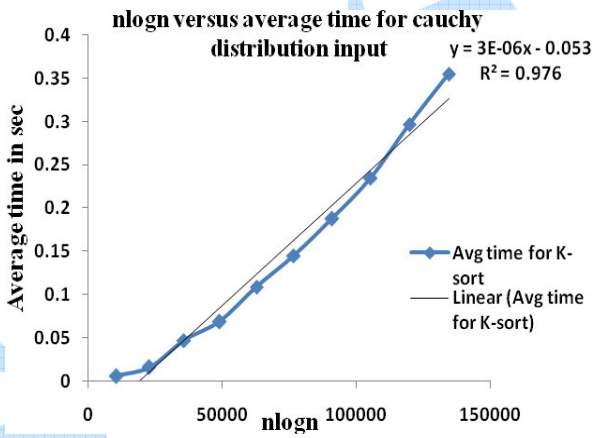

Fig 1: Average complexity supporting O(nlogn) complexity

3.2. Comparison between quick sort and K-sort for discrete uniform distribution U(1, 2, …, k) and continuous uniform distribution U(0,1)

A computer experiment is a series of runs of a code for various inputs. By running computer experiments on Borland International Turbo 'C++' ver 5.02, we could compare the average sorting time in seconds (average taken over 500 readings) for different values of n for both K-sort and quick sort. Using Monte Carlo simulation [KG80], the array of size n was filled with independent discrete uniform distribution U(1, 2, …, k) in phase one of the second study and also continuous uniform U[0,1] variates in phase two and the elements are copied to another array. One array is sorted by K-sort while the other is sorted by quick sort. Table 2, Table 3, Table 4 and Fig. 2, Fig 3, Fig 4 give the empirical results respectively.

Table 2: Average time for quick sort and K-sort on discrete uniform distribution for fixed k=1000 and various value of n

n	Average time for quick sort (Sec)	Average time for K-sort (Sec)
50000	0.058	0.0155
100000	0.184	0.025
500000	3.955	0.422
1000000	15.364	1.574

Table 3: Average time for quick sort and K-sort on discrete uniform distribution U(1, 2, ..., k) for fixed n=100000 and various value of k

n	Average time for quick sort (Sec)	Average time for K-sort (Sec)
5000	0.068	0.016
10000	0.053	0.025
20000	0.047	0.0306
30000	0.047	0.031

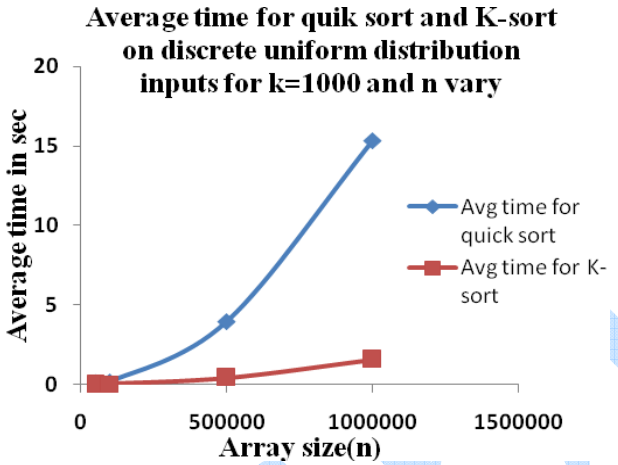

Fig 2: Comparison of average time of quick sort and K-sort on discrete uniform distribution for k=1000 and varying n

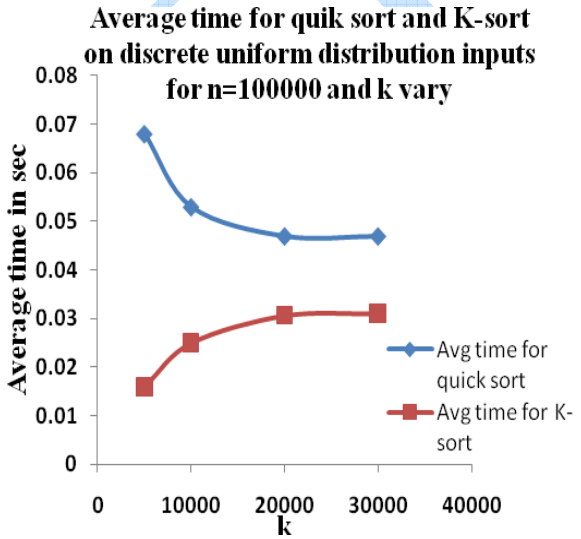

Fig 3: Comparison of average time of quick sort and K-sort on discrete uniform distribution for n=100000 and varying k

Table 4: Average time for quick sort and K-sort on continuous uniform distribution for varying n

n	Average time for quick sort (Sec)	Average time for K-sort (Sec)
100000	0.0154	0.154
500000	0.075	0.081
1000000	0.156	0.187
2500000	0.519	0.656
5000000	1.515	1.916
7500000	2.963	3.812
10000000	4.87	6.348

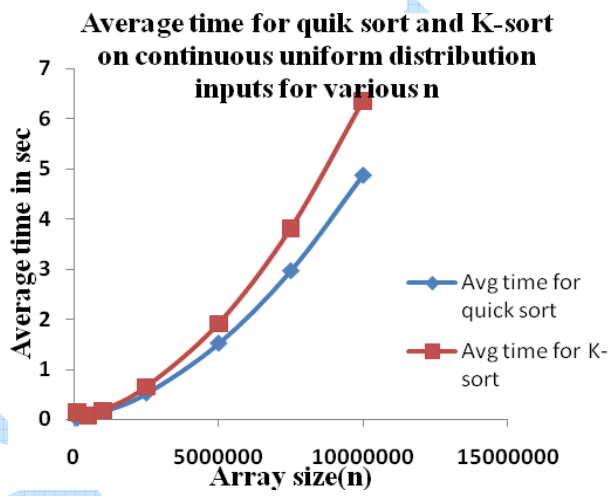

Fig 4: Comparison of average time of quick sort and K-sort on continuous uniform distribution for varying n

3. DISCUSSION

From Fig 1, it is clear that K-sort supports O (nlogn) average complexity for Cauchy distribution inputs like quick sort. Although K-sort is faster than heap sort up to n ≤ 70 lakhs and both algorithms have average $O(n\log_2 n)$ complexity [S+11a], it is not easy to calculate the average time for higher value of n where data comes from Cauchy distribution because two independent standard normal variates are divided in the case of Cauchy distribution (theorem 2). The program failed to run and division by zero error is shown by the computer frequently.

From a moment's reflection from Table 2 and Fig 2, it is clear that K-sort is faster than quick sort for discrete uniform distribution inputs for various value of n and fixed k Table 3 and Fig 3 show that the average sorting time for quick sort is decreased for discrete uniform distribution inputs when k increases (possibly because increasing k amounts to a less ties). But the average sorting time for K-sort is increased for discrete uniform distribution inputs when k increases. This is because K sort is so designed that more ties means more computations [S+11b]. Table 4 and Fig 4 suggest that K-sort is

inferior to quick sort for continuous uniform distribution inputs, although the robustness of K-sort remained unchallenged.

4. CONCLUSION AND FUTURE WORK

The empirical results confirm the average case O (nlogn) complexity for K-sort. It is important to mention here that the quick sort O(nlogn) complexity has been recently challenged for non-uniform inputs [SC08]. K sort is only a variation of Quick sort which removes the interchanges. So it was important to verify the robustness, given that it is also faster than heap sort for input size up to 70 lakhs. In the bargain, we have exposed the acknowledged problem of the probability distribution over which expectation is taken not being realistic over the problem domain as in the case of quick sort. The experimental results exhibit that K- sort has an edge over quick sort for discrete uniform distribution inputs although inferior to quick sort for continuous uniform distribution inputs. Future work involves studies on parameterized complexity on K-sort.

REFERENCES

[CS07a] **S. Chakraborty, S. K. Sourabh** - *How robust are average complexity measures? A statistical case study*. Applied Mathematics and Computation 189 (2): 1787-1797 (2007)

[CS07b] **S. Chakraborty, S. K. Sourabh** - *On why an algorithmic time complexity measure can be system invariant rather than system independent*. Applied Mathematics and Computation 190(1): 195-204 (2007)

[CS10] **S. Chakraborty, S. K. Sourabh** - *A Computer Experiment Oriented Approach to Algorithmic Complexity*, Lambert Academic Publishing, 2010

[GK02] **S. C. Gupta, V. K. Kapoor** - *Fundamentals of Mathematical Statistics*, Sultan Chand and Sons, 11th revised edition, June 2002.

[Knu00] **D. E. Knuth** - *The Art of Computer Programming, sorting and searching, vol. 3*, Addison Wesely (Pearson Education Reprint), 2000.

[KG80] **W. Kennedy, J. Gentle** - *Statistical Computing*, Marcel Dekker Inc.,1980.

[Mah00] **H. Mahmoud** - *Sorting: A Distribution Theory*, John Wiley and Sons, 2000.

[MV92] **M. Li, P. M. B. Vitanyi** - *Average case complexity under the universal distribution equals worst case complexity*, Inf. Proc. Lett., 42, no. 3, 145-149, 1992.

[SC08] **S. K. Sourabh, S.Chakraborty** - *How robust is quick sort average complexity?* arXiv:0811.4376v1 [cs.DS]

[S+11a] **K. K. Sundararajan, M. Pal, S. Chakraborty, N. C. Mahanti** - *K-sort: A new sorting algorithm that beats Heap sort for n ≤ 70 lakhs!*, arXiv:1107.3622v1 [cs.DS]

[S+11b] **K. K. Sundararajan, M. Pal, S. Chakraborty, B. Pal, N. C. Mahanti** - *K-Sort Revisited for Negative Binomial Inputs*, Algorithms Research; 1(1): 1-4, 2011.

[S+89] **J. Sacks, W. Weltch, T. Mitchel, H. Wynn** - *Design and Analysis of Computer Experiments*, Statistical Science Vol.4 (4), (1989)