Algorithms Research 2012, 1(1): 1-4
DOI: 10.5923/j.algorithms.20120101.01

K Sort Revisited for Negative Binomial Inputs

Kiran Kumar Sundararajan[1], Mita Pal[2], Soubhik Chakraborty[2,*], Bijeeta Pal[3], N.C. Mahanti[2]

[1]Barclays Bank PLC, United Arab Emirates, Dubai
[2]Department of Applied Mathematics, Birla Institute of Technology, Mesra, Ranchi, 835215, India
[3]Department of Computer Engineering, Institute of Technology, Banaras Hindu University, Varanasi , India

Abstract Parameterized complexity is a branch of computational complexity theory in computer science that focuses on classifying computational problems according to their inherent difficulty with respect to multiple parameters of the input. In this context, the present paper examines, through computer experiments, the behavior of a new version of Quick sort, called K-sort, when the sorting elements follow a Negative Binomial distribution. 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. If the response of the computer experiment is the complexity of the underlying algorithm then it is deterministic for a fixed input but may be taken as stochastic for fixed input size and randomly varying input elements as in sorting. Even otherwise we can advocate stochastic modeling imagining the response as stochastic to achieve cheap and efficient prediction.

Keywords K-Sort, Parameterized Complexity, Negative Binomial, Factorial Experiments, Robustness

1. Introduction

Parameterized complexity is a branch of computational complexity theory in computer science that focuses on classifying computational problems according to their inherent difficulty with respect to *multiple* parameters of the input. The complexity of a problem is then measured as a function in those parameters. This allows to classify NP-hard problems on a finer scale than in the classical setting, where the complexity of a problem is only measured by the number of bits in the input. The first systematic work on parameterized complexity was done by Downey & Fellows[1]. The present paper examines the behavior of K-sort[2] (a new version of Quick sort that removes interchanges; an older version[3] used an auxiliary array which has been removed now) for negative binomial distribution inputs and is in continuation of our earlier work on this new algorithm for binomial distribution inputs[4] with the acknowledgement that here the focus will be on how the parameters of negative binomial distribution affect the average sorting time. This is a work in parameterized complexity whereby use is made of factorial experiments[*] when the n observations to be sorted come independently from negative binomial distribution NB (k, p).Here k is the desired number of successes (trials, that can result in either success or failure, continue independently until k successes are obtained; when k=1, we get geometric distribution) and p is the probability of successes in a trial that is fixed for all trials. To investigate the individual effect of number of sorting elements (n), negative binomial distribution parameters k and p as well as their interaction effects, a 3-cube factorial experiment is conducted with three levels of each of the three factors n, k and p. Although K-sort has been described in[2] we brief it again in section 1.2 for the sake of completeness. The experimental results are obtained through computer experiments.[**]

[*]Remark 1: A factorial experiment allows the effect of several factors and even interactions between them to be determined with the same number of trials as are necessary to determine any one of the effects by itself with the same degree of accuracy[5]. The term "factorial" may not have been used in print before 1935, when Prof. R. A. Fisher used it in his book The Design of Experiments[6]. Frank Yates made significant contributions, particularly in the analysis of designs, by the Yates Analysis[7,8].

[**]Remark 2: A computer experiment is a series of runs of a code for various inputs. Further literature on computer experiments can be found in[9]. A deterministic computer experiment is one which produces identical results if the code is re-run for identical inputs. If the response of the computer experiment is the complexity of the underlying algorithm then it is deterministic for a fixed input but may be taken as stochastic for fixed input size and randomly varying input elements as in sorting. Even otherwise we can advocate stochastic modeling *imagining the response as stochastic* to achieve cheap and efficient prediction. A recent book that gives a computer experiment oriented approach to algorithmic complexity, including parameterized complexity,

* Corresponding author:
soubhikc@yahoo.co.in (Soubhik Chakraborty)
Published online at http://journal.sapub.org/ algorithms

is[10].

1.2. K-sort[2]

The steps of K-sort 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

2. Empirical Results

Our first study examines the behavior of K sort for varying p with fixed n and k.

Table - 1 gives the average run time y (average taken over 100 readings) for different values of the argument p for fixed n=100000 and k=100.

Table 1. Average sorting time Paverage sorting time in sec.

0.1	0.1205
0.2	0.2345
0.3	0.3565
0.4	0.5082
0.5	0.6814
0.6	0.911
0.7	1.231
0.8	1.7219
0.9	2.8236

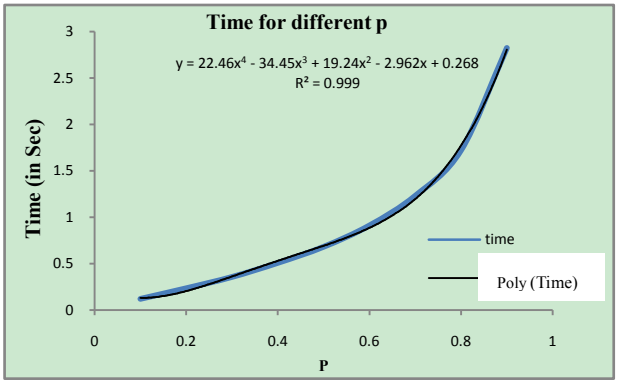

Figure 1. Average sorting time versus p: 4th degree polynomial fit.

From Fig. 1, based on the experimental results given in table1, we find that a fourth degree polynomial is the adequate fit to represent the average sorting time in terms of p for negative binomial distribution input for fixed n and k. We next study how time varies with varying k for n and p fixed. Table 2 and fig. 2 based on table 2 give a summary of the results.

Table 2. Average sorting time for varying k with fixed n=100000 , p=0.5.

K	Time
100	0.6814
500	0.319
1000	0.2346
1500	0.2004
2000	0.1752

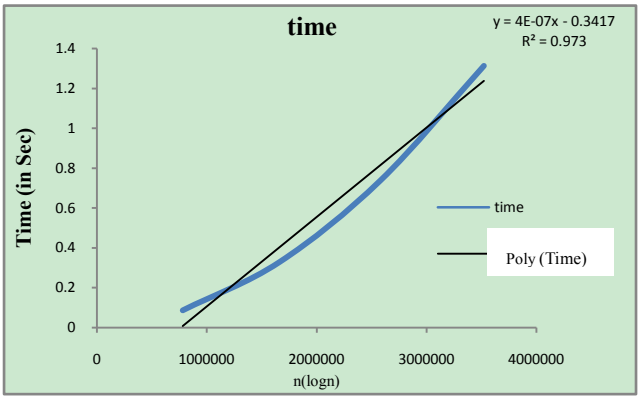

Figure 2. Average time contradicting O(nlogn) complexity.

We will next examine the robustness of K sort for negative binomial inputs. Our results are summarized in tables 3 and 4, fig. 3 and 4. Surprisingly, the results are supporting the worst case $O(n^2)$ complexity even for the average case. See the discussion in section 3.

Table 3. Table showing nlogn versus time.

X=n($\log_2$n)	Time
780482	0.086
1660964	0.3281
2579190	0.7372
3521928	1.3128

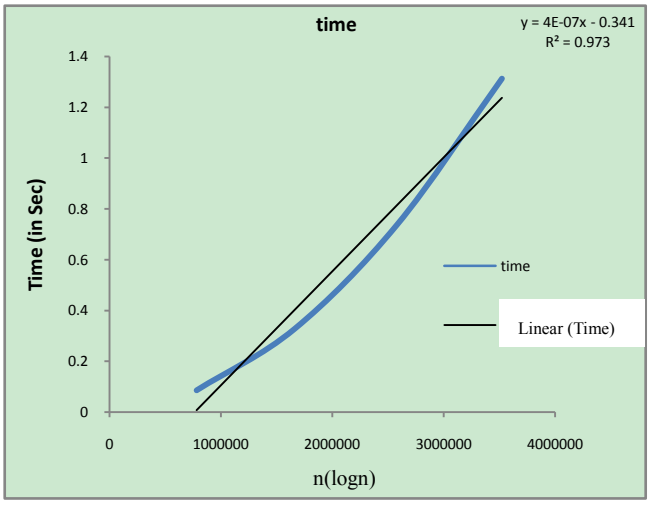

Figure 3. Average time contradicting O(nlogn) complexity.

Table 4. Table showing n versus time.

X= n	time
50000	0.086
100000	0.3281
150000	0.7372
200000	1.3128

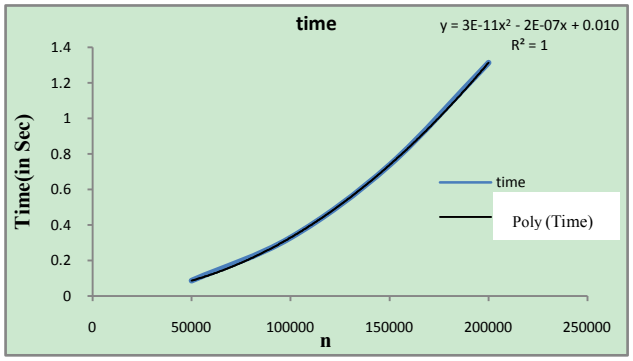

Figure 4. Average complexity supporting the worst case $O(n^2)$ complexity!

Table 5. Data for 3^3 factorial experiment for k-sort.

n = 100000

m	p=0.2	p=0.5	p=0.8
100	0.1093	0.3281	0.8232
1000	0.0598	0.1512	0.3718
1500	0.0358	0.1093	0.265

n=150000

m	p=0.2	p=0.5	p=0.8
100	0.2372	0.7372	1.8592
1000	0.114	0.3376	0.8312
1500	0.0907	0.2502	0.5924

n = 200000

m	p=0.2	p=0.5	p=0.8
100	0.4252	1.3128	3.2952
1000	0.2008	0.594	1.4735
1500	0.1748	0.4596	0.6704

K-sort times in seconds for negative binomial (k , p) distribution input for various n (100000, 150000, 200000) , K (100 , 1000, 1500) and p (0.2, 0.5, 0.8).

Table 6 gives the results using MINITAB statistical package version 15

Table 6. Results of 33 factorial experiment on K-sort for negative binomial inputs.

General Linear Model: y versus n, k, p

Factor	Type	Levels	Values
n	fixed	3	1, 2, 3
k	fixed	3	1, 2, 3
p	fixed	3	1, 2, 3

Analysis of Variance for y, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
n	2	7.6505	7.6505	3.8252	281942.68	0.000
k	2	7.0575	7.0575	3.5287	260089.68	0.000
p	2	14.4923	14.4923	7.2462	534085.81	0.000
n*k	4	1.8387	1.8387	0.4597	33881.48	0.000
n*p	4	3.7694	3.7694	0.9424	69457.80	0.000
k*p	4	3.8380	3.8380	0.9595	70721.00	0.000
n*k*p	8	1.0142	1.0142	0.1268	9343.78	0.000
Error	54	0.0007	0.0007	0.0000		
Total	80	39.6614				

S = 0.00368340 R-Sq = 100.00% R-Sq(adj) = 100.00%

3. Discussion

K- sort is highly affected by the main effects n, k and p. When we consider the interaction effects, interestingly we find that all interactions are significant in K- sort. Strikingly, even the three factor interaction n*k*p cannot be neglected. It is observed that y increases for increasing p in fig. 1. When the success probability p increases, it is common sense that we would be requiring comparatively lesser trials to get the same number of successes k. This is likely to increase the number of tied observations in the negative binomial variate values. A moment's reflection on the construction of the algorithm convinces us that more computations are required for the condition "if(key<=a[p])" (step 3) than for the condition "if(key>a[p])". *In other words, the case of equality resulting in ties is attached with the less than type (<) operator where more computations are taking place.* This increases the average sorting time. In contrast, for Binomial inputs, it was observed that y decreases for increasing p up to 0.5 and then increases for increasing p. This is because as p gets away from 0.5, either the lower values of the variate ($p<0.5$) or the higher values of the variate ($p>0.5$) are more likely resulting in more ties. With reference to k, sorting time decreases with increasing k because increasing k reduces the ties!

Although this is a study on parameterized complexity, we examined whether the average case O(nolgn) complexity is robust over negative binomial inputs. Surprisingly, it is not and the results are supporting the worst case $O(n^2)$ complexity even for the average case. It is important to mention here that the quick sort O(nlogn) complexity has been recently challenged for non-uniform inputs[11]. K sort is only a variation of Quick sort.

Remark: Average case complexity under the universal distribution equals worst case complexity[12].

4. Conclusions and Suggestions for Future Work

Our second case study on three-cube factorial experiments conducted on K-sort again reveal that the negative binomial parameters alongwith the input size singularly and interactively have significant effect on the average sorting time which increases with increasing p but decreases with increasing k. The algorithm supports the worst case complexity even in the average case for negative binomial inputs. Thus what holds for uniform inputs may not in general hold for non uniform inputs placing the robustness of average complexity as a challenging research problem. Some recent works in this direction have shown that the ideal bound for average case complexity is a statistical bound that is weight based and takes all operations collectively rather than mathematical bounds that are count based and operation specific. The problem with the count based mathematical bounds is that in average case we need to know the pivotal operation before applying mathematical expectation on it

and, moreover, the probability distribution over which expectation is taken has to be realistic over the problem domain. We refer the reader to[13,14]. See also[15] for a general overview on parameterized complexity.

REFERENCES

[1] Downey, Rod G.; Fellows, MichaelR.(1999).Parameterized Complexity, Springer

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

[3] Sundararajan , K. K., Chakraborty, S., A New Sorting Algorithm, Applied Mathematics and Computation, Vol. 188(1), 2007, p. 1037-1041

[4] Sundararajan, K. K., Pal, M., Chakraborty, S., Pal, B., and Mahanti, N.C., An Empirical Study on K-Sort for Binomial Inputs, International Journal of Mathematical Archive, Vol. 2(8), 2011, 1274-1278

[5] Fisher, R. A., The Arrangements of Field Experiments, Journal of the of the Ministry Agriculture of Great Britain, Vol. 33, (1926), p. 503-513

[6] Fisher, R. A. The Design of Experiments, Macmillan Pub. Co., 9th ed., (1971)

[7] http://www.iasri.res.in/ebook/EB_SMAR/e-book_pdf%20files/Manual%20III/5-Factorial-Expts.pdf

[8] Box, G. E., Hunter,W.G., Hunter, J.S., Statistics for Experimenters: Design, Innovation, and Discovery, 2nd Edition, Wiley, (2005)

[9] Fang, K. T., Li, R., 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] Sourabh, S.K. and Chakraborty, S.: How robust is quicksort average complexity? arXiv:0811.4376v1 [cs.DS]

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

[13] S. K. Sourabh and S. Chakraborty, Empirical Study on the Robustness of Average Complexity & Parameterized Complexity Measure for Heapsort Algorithm, International Journal of Computational Cognition, Vol. 7, No. 4, 2009, 1-11

[14] Sourabh and S. Chakraborty, Empirical Study on the Average Time Complexity of ShellSort Algorithm: Historical perspective, Robusness and Parameterized Complexity, International Journal of Mathematical Modeling, Simulation and Applications, Vol. 3(2), 2010

[15] http://en.wikipedia.org/wiki/Parameterized_complexity