International Journal on Future Revolution in Computer Science & Communication Engineering
Volume: 4 Issue: 1
ISSN: 2454-4248
117 – 123

Parameterized Complexity of Quick Sort, Heap Sort and K-sort Algorithms with Quasi Binomial Input

Priyadarshani
Research Scholar, Department of Statistics, Patna University, Patna, India
priyadarshini.bca.mmc@gmail.com

Soubhik Chakraborty*
*corresponding author
Professor, Department of Mathematics, B.I.T Mesra, Ranchi, India
soubhikc@yahoo.co.in

Anchala Kumari
Professor, Department of Statistics, Patna University, Patna, India
anchalastat@gmail.com

Abstract: Parameterized complexity is one of the important concepts that has emerged in the recent past in the field of computer science. It has been observed that for certain algorithms such as sorting, the parameters of the input distribution characterizing the sorting elements are very crucial in explaining the complexity of an algorithm (see Anchala, Chakraborty (2007,2008), Chakraborty&Sourabh,(2007)).The present paper investigates the parameterized complexity of three sorting algorithms namely, Quick sort, Heap sort and K-sort (with the same average case complexity O(Nlog2N)), for the quasi binomial input parameters and compares the results with those for binomial input parameters.

AMS Mathematics Subject classification code 62P99

Key words *: Parameterized complexity , Empirical-O , Computer Experiment, quasi binomial input*

*****

I. Introduction

Since long one of the problemsof great concern for the researchers while examining the efficiency of sorting algorithms, has been the problem of parameterized complexity. The multiple parameters of the distribution from which the array elements are being generated have direct effect on the sorting time of the algorithms as such, apart from the input size, the parameter of input distributions which must also be taken into account for precise evaluation of computational and time complexity of analgorithm. A comprehensive literature is found to exist on the parameterized complexity. To name a few are the works by Anchala and Chakarborty(2007,2008,2009,2015), Sudararajan and Chakarborty (2007), Chakarborty and Saurabh(2007).Their approach is based onEmperical –O analysis , an estimate of the statistical bound over a finite range obtained by supplying numerical values to the weights which emerge from computer experiment. A computer experiment is a series of runs of a code for various inputs and is called deterministic if it gives identical outputs if the code is rerun for the same input. In this paper attempt has been made to examine the parameterized complexity of quick sort, heap sort and k sort, having the same average case complexity O(Nlog2N).

Quasi binomial distribution has the probability of success varying from trial to trial where as in Binomial distribution it is fixed. There are situations demanding varying probability of success. For example in a jury trial, the probability p_i that the ith juror takes a correct decision may vary and in national TB testing the probability that an animal will be found to be a reactor could depend on the farming region (Boland(2007)).

We perform the empirical analysis of the results obtained by applying the specified algorithms over the input data generated from the quasi binomial and binomial distribution.The codes are written in Dev C++ and 3^2 factorial experimentswere performed using Minitab Statistical Package version 16.

The array size varied from 1 lac -10 lac which may be considered large enough for practical data set.

The response(CPU time to run the code) the mean time in seconds for different algorithms is given in the tables1-3 and the relative performance plots for binomial and quasi binomial distributions under different algorithms are presented in figures1-3. Average case analysis is performed directly on program run time to estimate the weight based statistical bound over a finite range by running computer experiments. This estimate is called empirical-O.Time of an operation is taken as weight. Weighing permits collective consideration of all operations into a conceptual bound called as statistical bound in order to distinguish it from the count based mathematical bounds that are operation specific.(Chakraborty and Sourabh, 2010).

II. Relative performance analysis of different Algorithms

The Binomial distribution has two parameters m and p, m being the number of independent Bernoulli trials and p the probability of success in each trial. The mean time under different algorithms by varying N and fixing m at 1000 is given in tables to follow.

 ISSN: 2454-4248
117 – 123

Table1: Data for quick sort (mean time in sec)

N	P=.2	P =0.5	P=.8	P rand
100000	.35367	.28720	.3489	.28470
200000	1.40367	1.1166	1.3803	1.1128
300000	3.10000	2.4636	3.0833	2.4880
400000	5.48833	4.3850	5.5697	4.38760
500000	8.58033	6.8992	8.5589	6.83380
600000	12.3750	9.8526	12.3707	9.8960
700000	16.7627	13.5338	16.8107	13.6270
800000	21.9366	17.68875	21.8577	17.4786
900000	27.7303	22.0784	27.8573	22.2047
1000000	34.2713	27.2628	34.2670	27.05678

Figure-1: Relative performance of p=0.2, 0.5, 0.8 and p random for quick sort

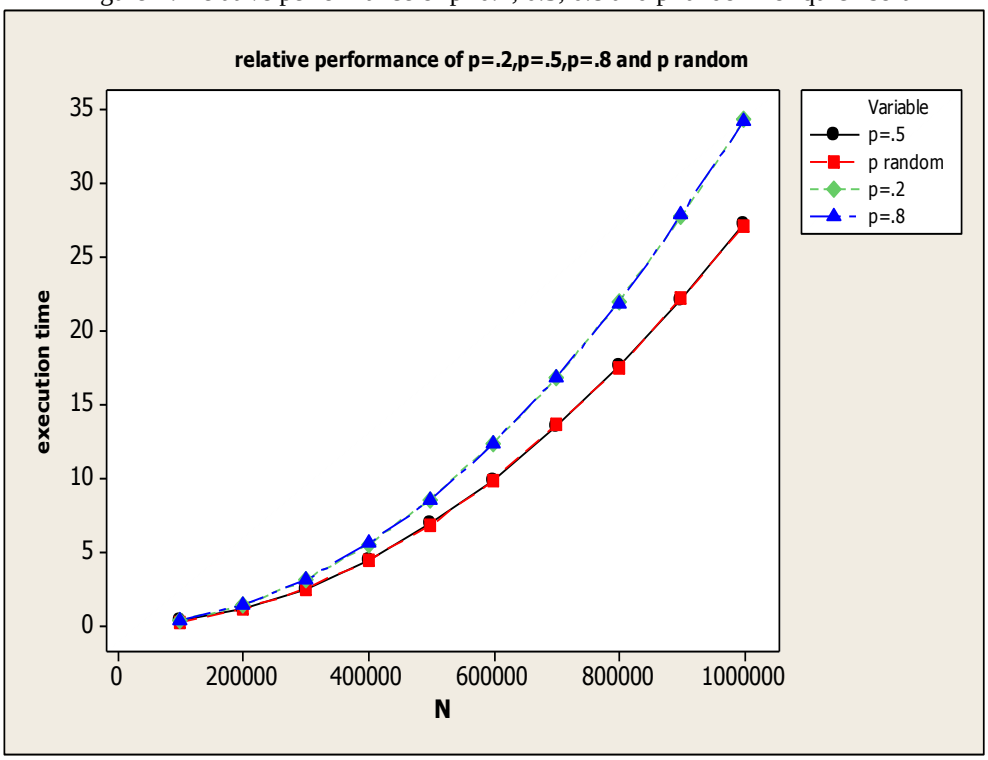


Table -2: Data for Ksort (mean time in sec)

N	P=.2	P=.5	P=.8	Prand
100000	.7810	.6096	.7503	.6106
200000	3.0063	4.0198	2.98967	2.4062
300000	6.2300	5.3716	6.7190	5.3592
400000	12.0166	9.9072	11.9397	9.5980
500000	18.8156	15.0238	18.7949	14.978
600000	26.682	22.8832	26.7099	21.5746
700000	36.7319	30.5168	36.7606	29.4714
800000	48.158	41.0869	47.9140	38.3089
900000	60.8359	49.8287	60.591	48.6076

 ISSN: 2454-4248
117 – 123

1000000	75.0027	62.4471	75.0099	60.008

Figure 2: Relative performance of p=0.2, 0.5, 0.8 and p random for K sort

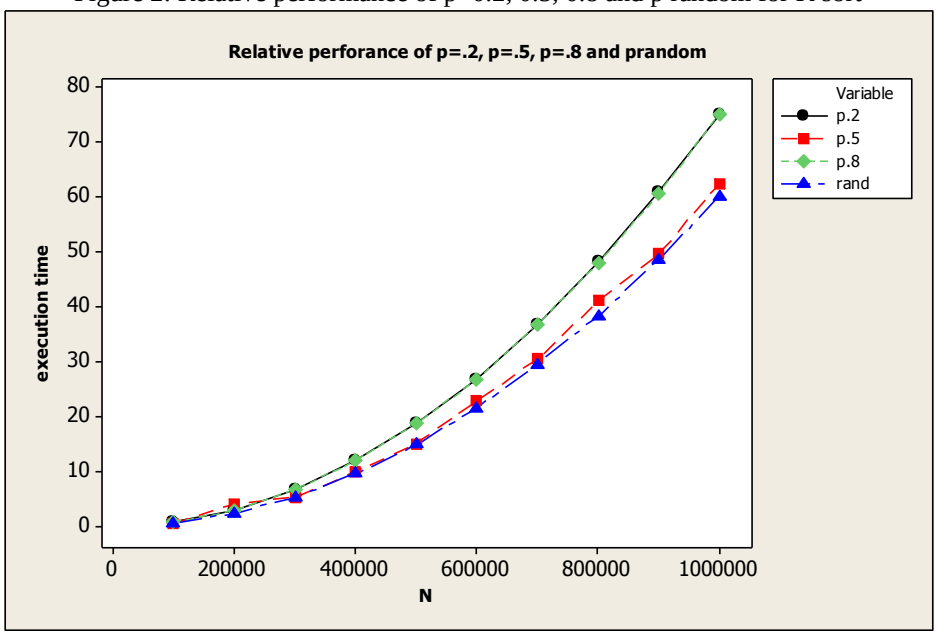
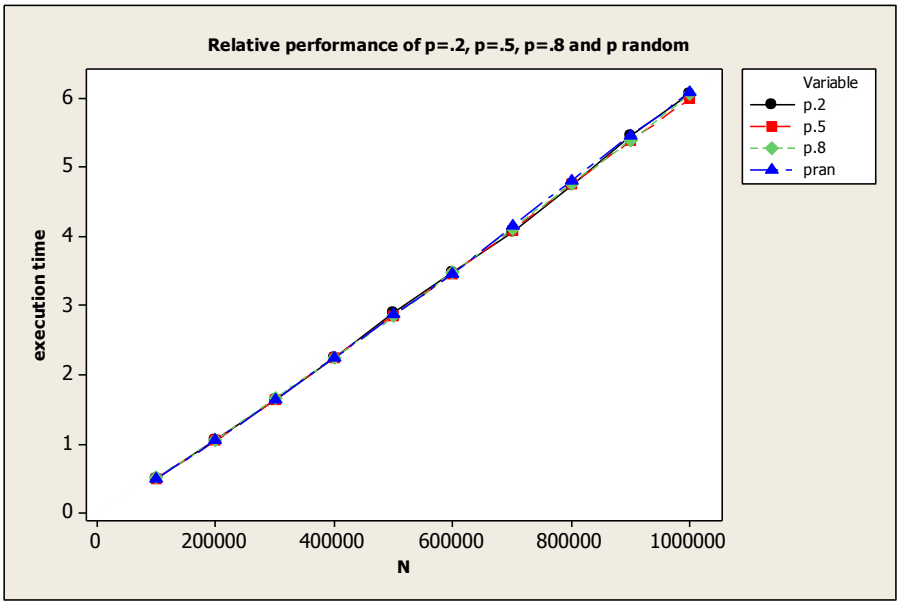


Table -3 : Data for Heap sort (mean time in sec)

N	P=.2	P=.5	P=.8	Prand
100000	.5010	.4998	.50533	.4906
200000	1.0520	1.0450	1.0547	1.0470
300000	1.6317	1.6344	1.6560	1.6348
400000	2.2410	2.2400	2.2383	2.226
500000	2.8863	2.8534	2.8387	2.8656
600000	3.4647	3.4602	3.4687	3.4620
700000	4.0593	4.0792	4.1057	4.1374
800000	4.7250	4.7480	4.7510	4.7970
900000	5.4497	5.3766	5.3873	5.4543
1000000	6.0490	5.9798	6.0469	6.0749

Figure-3: Relative performance of p=0.2, 0.5, 0.8 and p random for heap sort

 ISSN: 2454-4248
117 – 123



The following points can be revealed from the above analysis.

1.As far as the relative performance of quick sort and k-sort algorithms is observed, both reveal almost the same pattern. For N<300000, Quick sort as well as K- sort both have the same complexity measure irrespective of different values of p or whether it is fixed or varies from trial to trial, but for N> 300000, execution time at central part of the distribution, i.e., for p=.5 coincides with execution time when p is random in case of Quick sort algorithm, however a little higher complexity level is observed for p=.5 as compared to p random in case of k sort algorithm.

2.For homogeneous case (probability of success same at each trial) we get an upper bound for tail probabilities and lower bound for the more central part of the distribution for sorting an array of same size.

3. Heap sort reveals a different pattern as compared to the other two sorting algorithms. It shows the same complexity level irrespective of the fact that p is varying from trial to trial or it is same at each trial. Even for homogeneous case (probability of success same at each trial) complexity level remains same at the tail probabilities as well as at the central part of the distribution (p=.5) for varying N.

Table-4: Data for different algorithms for varying p

N	QS	HS	KS
100000	0.284700	0.4906	0.61060
200000	1.112800	1.0470	2.40620
300000	2.38800	1.6348	5.35920
400000	4.38760	2.24260	9.59580
500000	6.83380	2.86560	14.9780
600000	9.89680	3.4620	21.57498
700000	13.6270	4.13740	29.42139
800000	17.47860	4.7970	38.30899
900000	22.20473	5.45433	48.69759
1000000	27.05678	6.07499	60.00879

 ISSN: 2454-4248
117 – 123

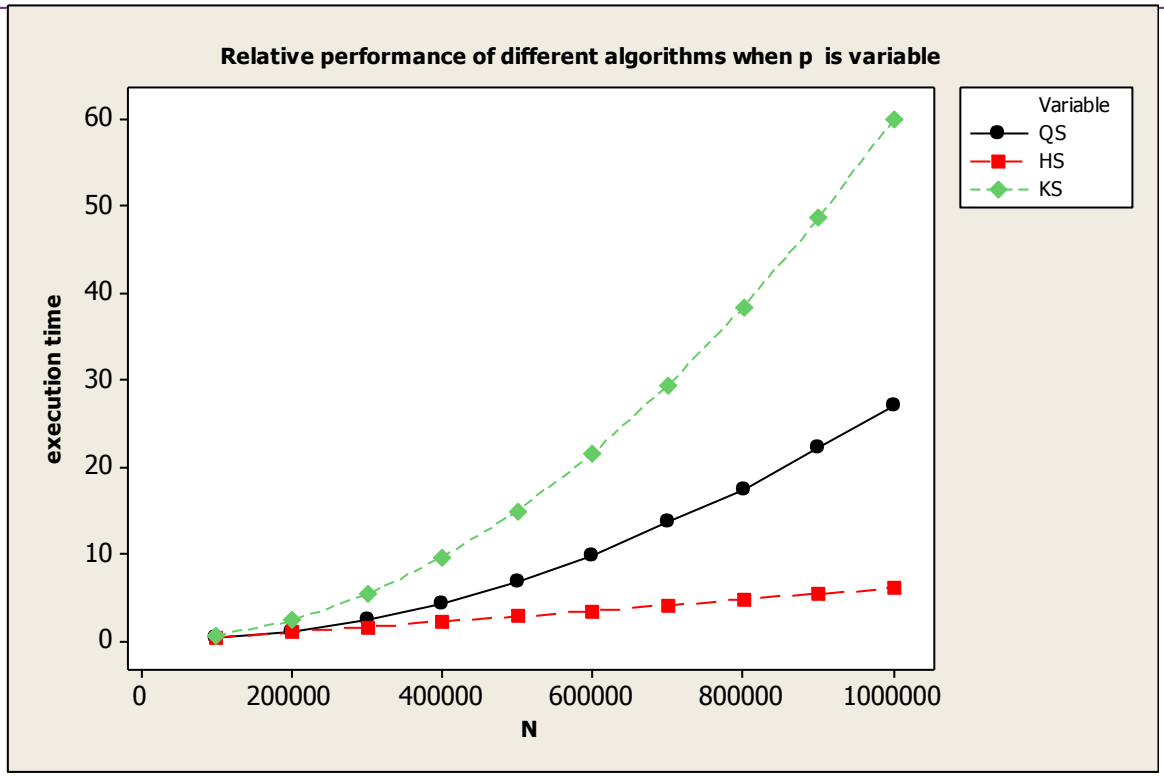


Figure-4: Relative performance of different algorithms when p is varying

Some interesting results are seemed to emerge from the above table. For N>100000, heapsort outperforms the quicksort. The reason for illperformance of quick sort may be due to the increase in number of ties as N increases. As far as the K-sort is considered,it consumes more time as compared to other two algorithms for sorting array of same size. However quicksort and K-sort both confirms to $O_{emp}(N^2)$, whereas heap sort has $O(N\log_2 N)$ complexity. Similar results are obtained when probability of success for each trial is 0.5 (see Anchala Singh And Chakarborty(2015))

III. Parameterized complexity

Parametrized complexity is one of the criterion for selection of an algorithm among the several algorithms. The parameter of input distribution has direct effect on the complexity of an algorithm.

Table 5. Parameterized complexity of heap, k and quick sort

Probability of Success (p=0.5)							
		Heap Sort		K-Sort		Quick Sort	
Sources	D.f.	F	p	F	p	F	P
N	2	18356.13	0.000	12719.788	0.000	775.04	0.000
M	2	17.5	0.000	14345.99	0.000	766.74	0.000
NM	4	1.75	0.273	1508.06	0.000	110.54	0.000
Probability of success (p random)							
		Heap Sort		K-Sort		Quick Sort	
Sources	D.f.	F	p	F	p	F	P
N	2	32601.46	0.000	39907.0	0.000	285803.62	0.000
M	2	0.65	0.531	41190.83	0.000	285803.62	0.000
NM	4	0.56	0.695	5230.11	0.000	36195.52	0.000

The F and p values revealed that in case of Heap Sort,m has significant and independent effect (mn interaction being non significant) on sorting complexity, when the probability of success is same from trial to trial. On the other hand number of trials (m) has no effect on the complexity as probability of success varies from trial to trial as shown by the small value of F (.65 for m and .56 for mn). If we compare the significant points of Quick and K sort algorithms, it is found that the number of trials (m) shows highly significant singular and interactive effect on sorting efficiency of the algorithms irrespective of the Input distributions,i.e., whether the Input distribution is Binomial or Quasibinomial , m and mn both have significant effect on sorting time. This implies that proper selection of input

 ISSN: 2454-4248
117 – 123

parametersmay have rewarding effect on reducing the complexity of an algorithm.

Now we shall try to find for which value of m the sorting time is minimum. We try to find the optimal value of m for all the algorithms under (i) p homogeneous and (ii) p non homogeneous except for p varying in heap sort case as (m in this case shows non significant effect on execution time).

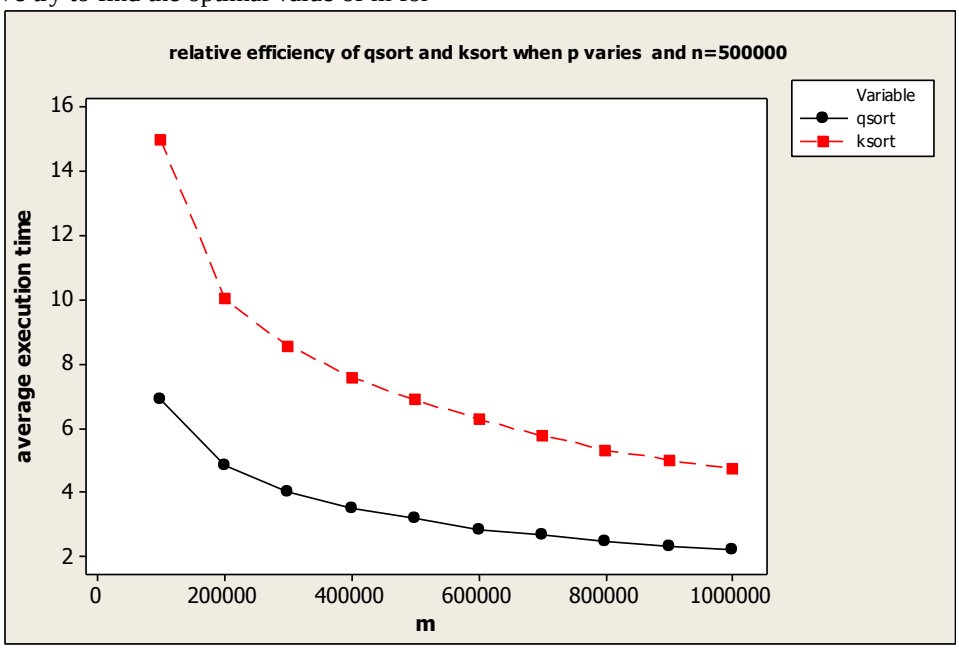


Fig 5: Relative efficiency of qsort and ksort when p varies and n = 500000

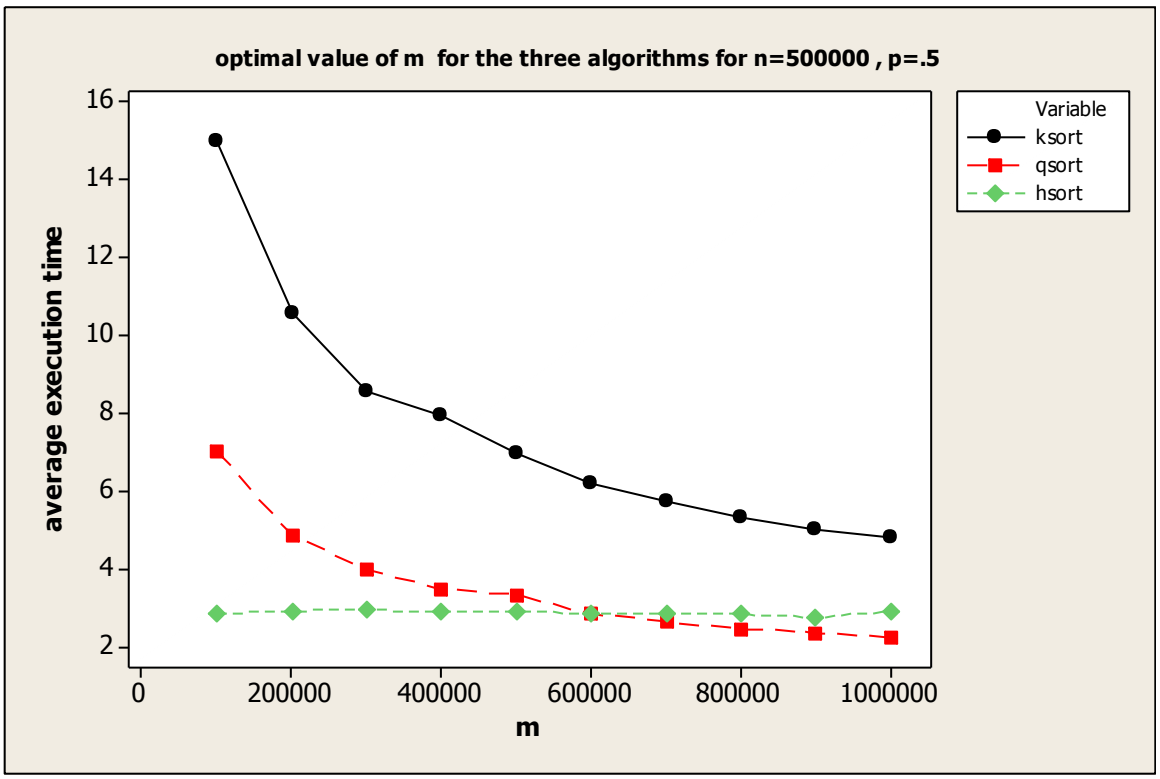


Figure – 6: optimal value of m for the three algorithms for n = 500000, p = 0.5

From the above two tables it is observed that irrespective of the fact that which algorithm is being used or from which distribution the array elements have been generated, the execution time is inversely proportional to the number of trials (m). Execution time decreases as the value of m is increased.Thus a high value of m is preferableso as to minimise the execution time .Secondly it is further observed that for a fixed value of m ,K-sort consumes more time than the quick sort(whether p varies or it does not vary from trial to trial (figure-5&6). We have seen earlier also that quick sort gives better performance than K-sort (Figure-4) irrespective of the fact that array of any size is

 ISSN: 2454-4248
117 – 123

being sorted. As for as the Quick Sort and Heap sort are considered, for m<60,000, Heap sort is more efficient than Quick sort(Figure-5) and after m>60000 Quick sort gives better performance than Heap sort.

IV. Conclusion

1. As far as the empirical-O analysis is considered, the average case complexity of Quick sort is same whether p is fixed at .5 or p varies from trial to trial. In case of K–sort, the average performance for sorting an array of same size generated from the two distributions is almost same but as far as the preference is considered, K-sort when applied to array generated from quasi binomial gives better result. Average complexity level in case of Heap sort remains same whether array is generated from quasi binomial or binomial distribution with tail probabilities or keeping the probability of success and failure as equal.

2. As far as the relative performance of the three algorithms is considered while sorting an array of fixed size generated from quasi binomial distribution, heap sort is most preferable.
3. The different behaviour of algorithms to input parameters can be summarized by parametric complexity analysis. The complexity analysis exhibits that number of trials plays an important role in explaining the complexity of sorting algorithms. As for as the heap sort is considered, m has no effect on the complexity for varying p, while it has significant but independent effect on complexity for fixed p. Quick sort and K-sort algorithms both are highly affected by the value of m . A high value of m has diminishing effect on the complexity.
4. Lastly it may be inferred, that for sorting an array generated from binomial distribution irrespective of its size, heap sort gives better performance for m(no.of trails)<60,000, and for m>62000, quicksort outperforms the heap sort and for values of m between 60,000 to 62,000 either of the two algorithms can be used for sorting purpose.When the array of a fixed size is generated from Quasi binomial, heap sort may be preferred with moderate value of m. As mentioned earlier, theoretically , all the three algorithms exhibit to same average complexity but heap sort for m <60,000 with binomial inputs and for moderate value of m with quasi binomial inputs for sorting an array of any size supports to worst case complexity $O(N\log_2 N)<O(N^2)$, worst case complexity of Quick and K sorts.

Acknowledgements and Declaration of conflict of interest: The authors hereby declare that this research did not receive any financial grant. They further declare that they do not have any conflict of interest.

References

[1] Anchala Kumari, S. Chakraborty, Software Complexity ; A Statistical Complexity,A Statistical Case Study Through Insertion Sort, journal of Applied Math. And Computation vol.190(1),2007, p. 40-50

[2] Anchala Kumari, S. Chakraborty, A Simulation study on Quicksort Parameterized complexity using response surface design; International journal of Mathematical Modeling , Simulation and Applications.vol.1,no.,pp, 448-458,2008.

[3] Anchala Kumari, S. Chakraborty, Parameterized Complexity; A Statistical Approach CombiningFactorial Experimentswith Principal Component Analysis: International journal of Ccomputer Science Engineering,2013, vol. 2, No. 5, 166-176

[4] Anchala Kumari, Niraj Kumar Singh ansd Soubhik Chakraborty; A Statistical Comparative Study of Some Sorting Algorithms, International journal in Foundations of Computer Science and Technology, Vol.5, No. 4, July 2015, 21-29

[5] P.J.Boland, The Probability Distribution of the number of Successes in Independent Trials, Communications in Statistics –Theory and Methods,36:132-131, 2007.

[6] S.Chakraborty, S.K.Sourabh,On Why an Algorithm Time Complexity Measure can be System Invariant rather than System Independent,Applied Math and Computation, Vol.190(1) 2007,p.195-204.

[7] S.Chakraborty K.K. Sundararajan , B.K. .Das and S.K.Sourabh On How Statistics Provides a Reliable and Valid Measure for an Algorithm's Complexity. InterStat,Dec2004#2http://InterStat,statournals.net/

[8] . K. K. Sundararajan, M. Pal, S. Chakraborty and N. C. Mahanti, K-Sort: A New Sorting Algorithm that beats Heap Sort for n <= 70 lakhs!, International on Recent Trends in Engineering and Technology (ACEEE), Vol. 8, No. 1, Jan 2013, 64-67

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