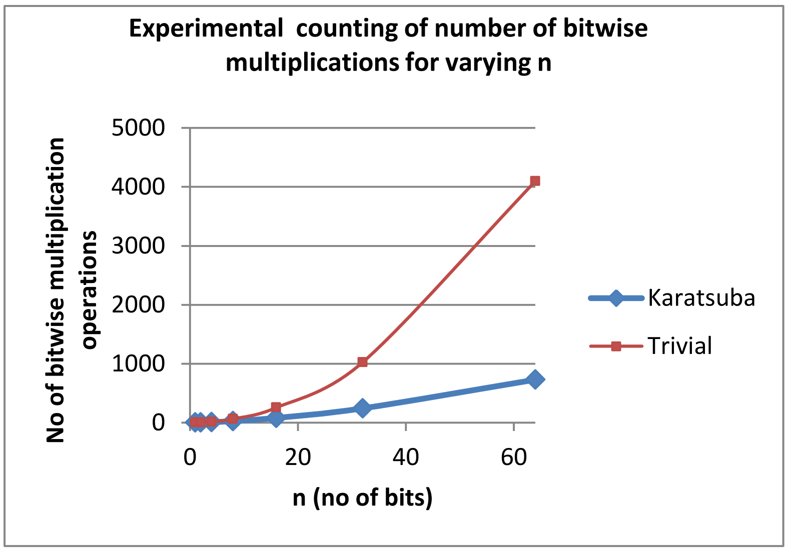
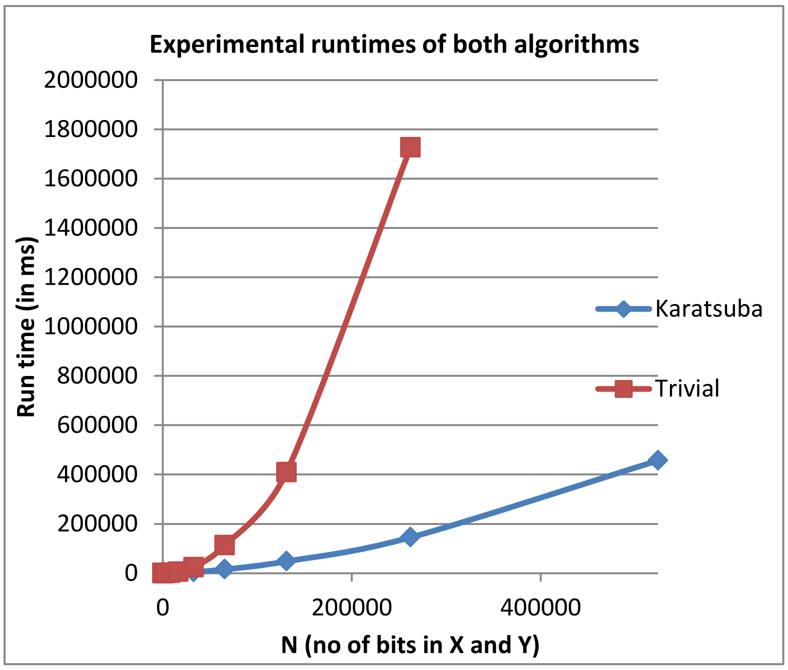
Q1) Firstly two kinds of experimental results are plotted – 1) the actual runtimes of the two algorithms' implementations and 2) a modelling of the number of bitwise multiplication operations performed in each case. It can be shown why considering only the bitwise multiplication operations are enough to model the time complexity of the algorithm.



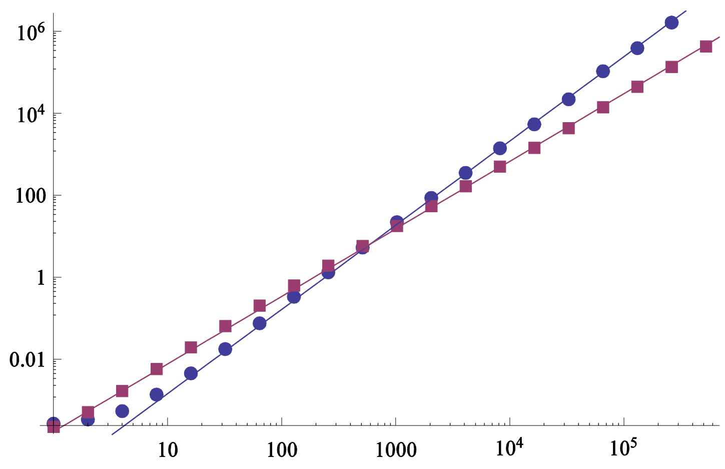


From these graphs, it there is clear indication that the Karatsuba algorithm is asymptotically faster than the Trivial algorithm. The disadvantage of plotting the runtimes versus n directly is that it is difficult to capture the entire range of runtimes on y-axis for n ranging from 1 to 524288. The log – log scale is better suited for this, in addition to also being easier to do an experimental fitting, since the fit is expected to be linear on that scale.The comparison of runtimes for some small n are shown:

N (# of bits)	Karatsuba (ms)	Trivial (ms)
1	0.000243	0.000288
2	0.000555	0.000364
4	0.001807	0.000581
8	0.006267	0.001468
16	0.021078	0.004842
32	0.069657	0.01885
64	0.219732	0.080302
128	0.67981	0.357015
256	2.057613	1.40647
512	6.25	5.649718
1024	19.230769	23.488372
2048	58.823529	91.818182
4096	178.333333	373.333333

There exists a threshold n (n = 1024) beyond which , i.e. for values of n greater than 1024, the Karatsuba algorithm is faster than the Trivial algorithm . However, for n from 1 upto 1024, the Trivial algorithm is faster.

Now, we study the log-log plots of runtimes, T_Karat and T_Trivial vs n, and the data is fitted to the model log (T) = a*Log(n) + Log(b), where a and b are constants corresponding to T = b*n^a



The parameters a and b in $T(n) = b*n^a$ are determined by fitting:

Parameter ->	a (exponent) from Expt.		Theoretical expected value of exponent	b (proportional to hidden constant in O notn)	
Algorithm ↓	Value	Std devn		Value (in ms)	Std devn (in ms)
1. Karatsuba	1.647	0.003	≈ 1.584	0.000171	7×10^{-6}
2. Trivial Algo.	2.05	0.01	2	0.000012	2×10^{-6}

The exponents are seen to be very close to the theoretical value within about 4 % for Karatsuba and 2.5 % for the trivial algorithm. Note that the constant b is greater for the Karatsuba case as compared to the trivial algorithm – for this reason only we see a threshold value of n below which the Trivial algorithm is faster than the Karatsuba algorithm. So one imporant observation is that *there there exists a critical value of n below which the trivial algorithm is faster. The exact critical value will depend on the actual implemention of the algorithm, since different implementations of the same algorithm (say Karatsuba) , will have different values of the constant b (the hidden constant in the big O notation). Furthermore, the critical n value will depend also on the platform on which the program is run since different types of operations may take slightly different times to execute on different operating systems.*

From the data points along with the fitting, it is easy to determine the maximum number of bits (n) for which either of the algorithm implementations will finish in a specified time :

	Karatsuba Algorithm	Trivial Algorithm
1 min	152 983	51 097
2 min	232 989	71 603
4 min	354 836	100 338
10 min	618 777	156 737

The Karatsuba algorithm is performing better than the trivial algorithm. Also note that though it seems that in the recursive algorithm, digit-multiplication steps were reduced at the cost of increased number of additions. However, it is easily shown that the total number of digit operations is also of $O(n^{\log_2(3)})$. For the same reason, just counting bit multiplications will model the actual time complexity correctly.