Analysis of Fast multiplication algorithm

Indiana University, Bloomington

Fall-2011

A project report submitted to Indiana University

By

Jayesh K. Kawli

Under supervision of Prof.Paul Purdom

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington

Table of contents

1. Abstract

This report includes the details of implementation and analysis of Karatsuba algorithm. This algorithm is used for multiplying two arbitrary bit numbers.

Karatsuba algorithm was invented by Antolii Karatsuba in 1960 and published in 1962 before invention of this algorithm the multiplication of two numbers would take much longer time due to large number of intermediate multiplications involved in calculation

Study of this algorithm is reasonably important as compared to grade school or classic algorithm which performs large number of multiplications karatsuba algorithm reduces total number of multiplications to be done and saves reasonable amount of time. But this saving comes at the cost of additional additions and subtractions that to be done to perform multiplications of numbers. But since cost of addition and subtraction operations is much less than multiplication operation, it ends up saving reasonable amount of time at the end especially for numbers having large number of bits.

2. System specification

- **Software specification**

- Operating system: Linux
- Compiler: GNU compiler
- GMP library which used for arbitrary precision arithmetic, operating on signed integers, rational numbers, and floating point numbers generation

- **Hardware Specification**
- Processor - Intel Core I5 processor
- Memory – 4GB

3. Karatsuba Algorithm

3.1 Introduction

Karatsuba algorithm used for Multiplication of large integers is Divide and Conquer algorithm. The Karatsuba algorithm provides a example of how the "Divide and conquer" technique can achieve an asymptotic speedup over an ancient algorithm.

The grade school method of multiplying two n-digit integers requires O(m*n) digit operations. Karatsuba is a simple recursive algorithm solves the problem in $O(n^{\alpha})$ digit operations, where

$\alpha = \log_2 3 \approx 1.58$

This is a considerable improvement of the asymptotic order of magnitude of the number of digit operations.

3.2 The basic steps in Multiplication

We compute the product of two large numbers U and V using three multiplications of smaller numbers, each with about half as many digits as U or V, plus some additions, subtractions and digit shifts.

Let U and V be represented as m and n digit numbers in some base B. For any positive integers a and b we can split the two given numbers as follows

Let **$U=U2*2^{a}+U1$**
$V=V2*2^{b}+V1$

1. $T1=U1+2^{c}U2$
2. $T2=V1+2^{d}V2$
3. $W3=T1*T2$
4. $W2=U1*V1$
5. $W4=U2*V2$
6. $W3=W3-W2-W4$
7. $C=W4/2^{n}$ and $W4=W4 \bmod 2^{n}$ (where n=minimum(a,b))
8. $W3=W3+C$, $C=W3/2^{n}$,$W3=W3 \bmod 2^{n}$
9. $W2=W2+C$,$W1=W2/2^{n}$,$w2=w2 \bmod 2^{n}$

3.3 Analysis of size of result in successive steps

Step Number	Variable	Size
1	T1	max(a,m-a+c)
2	T2	max(b,n-b+d)
3	W3	max(a,m-a+c)+max(b,n-b+d)
4	W2	a+b
5	W4	m-a+n-b
6	W3	max(a+b,m-a+n-b,m+b-a+c,a+n-b+d,m+n-a-b+c+d)
7	C	m-a+n-b-min(a,b)
8	W3	max(max(a+b,m-a+n-b,m+b-a+c,a+n-b+d,m+n-a-b+c+d), m-a+n-b-min(a,b))
8	C	max(a+b-n,m-a-b,m+b-a+c-n,a-b+d,m-a-b+c+d)
9	W2	max(a+b, max(a+b-n,m-a-b,m+b-a+c-n,a-b+d,m-a-b+c+d))
9	W1	max(a+b-n, max(a+b-2n,m-a-b-n,m+b-a+c-2n,a-b+d-n,m-a-b+c+d-n))

Table 4.1

4. Time analysis

4.1 Analysis for time taken by each step in algorithm

We can divide total time taken as follows with assumption that time is spent is considered only for multiplications, additions and subtractions. The time spent on other operations is assumed to be negligible.

Let T(U,V) be the time total time taken to multiply U digit number with V digit number

Step number	Operation type	Operation	Total time taken
1	Addition	A(U1,U2)	A(a,m-a+c)
2	Addition	A(V1,V2)	A(b,n-b+d)
3	Multiplication	M(T1,T2)	M(max(a,m-a+c), max(b,n-b+d))
4	Multiplication	M(U1,V1)	M(a,b)
5	Multiplication	M(U2,V2)	M(m-a,n-b)
6	Addition	A(W3,W2)	A(max(a,m-a+c)+max(b,n-b+d),a+b)
6	Addition	A(W3,W4)	A(max(max(a,m-a+c)+max(b,n-b+d),a+b),m+n-a-b))
8	Addition	A(W3,C)	A(max(a+b,m-a+n-b,m+b-a+c,a+n-b+d,m+n-a-b+c+d), m-a+n-b-min(a,b))
9	Addition	A(W2,C)	A(a+b, max(a+b-n,m-a-b,m+b-a+c-n,a-b+d,m-a-b+c+d)

Thus total time taken is given by following equation:

T(U,V)= A(a,m-a+c)+A(b,n-b+d) + M(max(a,m-a+c), max(b,n-b+d)) + M(a,b) + M(m-a,n-b) + A(max(a,m-a+c) + max(b,n-b+d),a+b) + A(max(max(a,m-a+c) + max(b,n-b+d),a+b),m+n-a-b)) + A(max(a+b,m-a+n-b,m+b-a+c,a+n-b+d,m+n-a-b+c+d), m-a+n-b-min(a,b)) + A(a+b, max(a+b-n,m-a-b,m+b-a+c-n,a-b+d,m-a-b+c+d)

5. Finding optimum division point

5.1. To find values of parameters from given equation and to find optimum division point for both the numbers

For two numbers U and V each of length m and n.

Let U be divided into two parts U2 and U1 each of length m-a and a respectively

Also V be divided as V2 and V1 each of length n-b and b

$U=U2*2^{a}+U1$

$V=V2*2^{b}+V1$

Given

$T1=U1+2^{c}*U2$

$T2=V1+2^{d}*V2$

$UV=U2*V2*2^{a}*2^{b}+V1*U2*2^{a}+U1*V2*2^{b}+U1*V1$

But given the value of W=UV

Thus putting value of T1 and T2

$U2*V2*2^{a}*2^{b}+V1*U2*2^{a}+U1*V2*2^{b}+U1*V1 = 2^{e}*U1*V1+(2^{f}(U1+2^{c}*U2)*(V1+2^{d}*V2)-2^{g}*U1*V1-2^{h}*U2*V2)+2^{f}*U2*V2$

Thus expanding the terms on both sides,

$U2*V2*2^{a+b}+V1*U2*2^{a}+U1*V2*2^{b}+U1*V1 = U2*V2(2^{i}-2^{h}+2^{f+d+c}+V1*U2*2^{f+c}+V2*U1*2^{f+d}+V1*U1*(2^{e}+2^{f}-2^{g})$

Thus comparing RHS and LHS with same coefficients

$2^{a+b} = 2^{i}-2^{h}+2^{f+d+c}$

$2^{a}=2^{f+c}$

$2^{b}=2^{f+d}$

$2^{0}=2^{e}+2^{f}-2^{g}$

This gives,

$a=f+c$

$b=f+d$

And for last equation there are three cases,

$2^0+2^g=2^e+2^f$

Since LHS contains one odd and one even term, the same organization should be present in RHS

Thus suppose $g>0$ then $e=0$ and $g=f$

Thus,

$e=0$

$g=f$

From above equations it gives,

$a-c = b-d$

And

$f=g=a-c;$

From equation (1) we get,

$2^{f+c+d}(2^f-1) = 2^h(2^{i-h}-1)$

Thus $h=f+c+d$

And $f=i-h$

Solving above equations,

$i=a-b$

$h=b+c$

$d=c-a+b$

We want to multiply U and V each of digits m and n respectively

We can make table for maximum digits for each number,

Variable	Number of digits
U1	a
V1	b
U2	m-a
V2	n-b
T1	Max(m-a+c , a)
T2	Max(n-b+d , b)

Table 4.1

Let T(m,n) be the total time to multiply m digit number with n digit number

T (m,n) = T (a,b) + T (m-a,n-b) + T (max(m-a+c,a),max(n-b+d,b))

Let assume max(m-a+c,a) = m-a+c

and

max(n-b+d,b) = n-b+d

Thus,

T(m,n)=a*b+(m-a)(n-b)+(m-a+c)(n-b+d)

We know from above equations that d=b+c-a

Thus,

T(m,n)=ab+(m-a)(n-b)+(m-a+c)(n+c-a)

Differentiating above equation with a,b and c respectively and setting derivative to zero,

b-n+b-n-c+a-m+a-c=0

thus

2a-2c+2b=2n+m

And

a-m+a=0

Which gives

a=m/2

Which is optimum value of a for best performance of karatsuba algorithm

And finally,

m-a+c+n+c-a=0

Thus

c=-n/2

Putting values of c and a in first equation,

We get ,

b=n/2

This is optimum value for b

d=b+c-a

d=-m/2

With this value we can write two numbers U and V as

$U=U2*2^{m/2}+U1$

$V=V2*2^{n/2}+V1$

$T1=U1+2^{-m/2}\ U2$

$T2=V1+2^{-n/2}\ V2$

Thus two optimize performance, of karatsuba algorithm a and b should be m/2 and n/2 respectively

That is we need to divide number into exact two half equal parts

8. Time complexity of karatsuba

Consider two numbers U and V

For simplicity let us assume for two numbers U and V each having the same length n.

So from above analysis there will be three multiplications each of length n/2

Further two additions each of length n/2

two additions each of length n

And two additions each of length n+1

Thus,

$T(n)=3T(n/2)+(n/2)+(n/2)+n+n+(n+1)+(n+1)$

$T(n)=3T(n/2)+5*n+2$

Using recurrence equation,

$T(n)=3*(3*T(n/4)+5*(n/2)+b)+5*n+2$

Repeating above procedure i times, we get,

$T(n)=3^{i}T(n/2^{i})+5*n\ \Sigma\ [1+(3/2)+(3/2)^{2}\ldots\ldots\ldots+3/2^{i-1}]+ 2/2\ \ \Sigma\ (1+3+3^{2}+\ldots\ldots.3^{i-1})$

Let assume for simplicity n be the power of 2

Hence, $n=2^{i}$

Thus

$T(n)=3^{i}T(2^{i}/2^{i})+5*n\ \Sigma\ [1+(3/2)+(3/2)^{2}\ldots\ldots\ldots+(3/2)^{i-1}]+\ \Sigma\ (1+3+3^{2}+\ldots\ldots.3^{i-1})$

As T(1)=1

And

$T(n)=3^{i}+5*2^{i}(((3/2)^{i}-1)/(3/2)-1)+(3^{i}-1)/(3-1)$

$T(n)=3^{i}+5*2*2^{i}[(3/2)^{i}-1]+(1/2)(3^{i}-1)$

Ignoring time required for addition and subtractions

$T(n)=3^{\log_{2} n}$

Which is equivalent to

$$\mathbf{T(n)=n^{\log_2 3}}$$

$$\mathbf{T(n)=n^{1.58}}$$

Substituting $i = \log_2 n$ above equation becomes,

$$\mathbf{T(n)=3^{\log_2 n}+a*2^{\log_2 n+1}[3/2^{\log_2 n}-1]+(1/2)(3^{\log_2 n}-1)}$$

6.2. Recursion tree – Graphical Representation

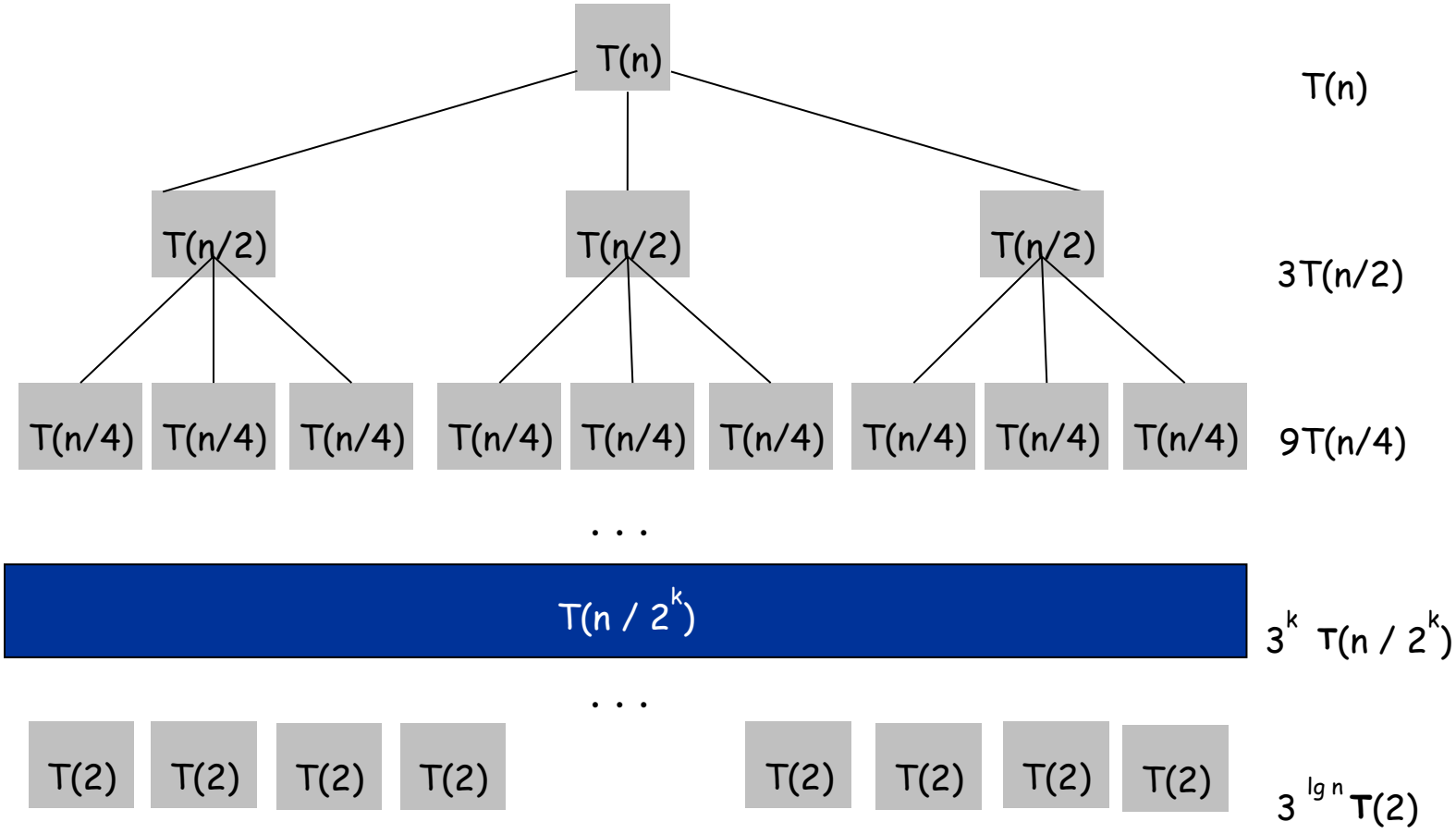


Fig. 5.2

7 Calculation table

7.1 Classic Algorithm

Serial Number	Number of bits in first number	Number of bits in second number	GMP time elapsed	Code time elapsed	Code/GMP ratio
1	1	1	0.000087	0.000081	0.925438
2	0	0	0.000093	0.000083	0.887626
3	0	1	0.000082	0.000081	0.985843
4	1	0	0.000081	0.000082	1.005069
5	32	32	0.000097	0.000079	0.808414
6	64	64	0.000110	0.000087	0.794902
7	480	480	0.000174	0.000282	1.621439
8	4064	4064	0.003179	0.032582	10.248913
9	32736	32736	0.289407	3.889953	13.441124
10	32	64	0.000122	0.000090	0.738082
11	32	480	0.000122	0.000099	0.815791
12	32	4064	0.000156	0.000237	1.513720
13	32	32736	0.000394	0.001332	3.379173
14	480	4064	0.000744	0.002065	2.774470
15	480	32736	0.008469	0.063993	7.556145
16	4064	32736	0.062961	0.330425	5.248117

Table 6.1

7.2 Karatsuba algorithm

Serial Number	Number of bits in first number	Number of bits in second number	GMP time elapsed	Code time elapsed	Code/GMP ratio
1	1	1	0.000080	0.000080	0.992339
2	0	0	0.000089	0.000084	0.942897
3	0	1	0.000080	0.000081	1.008383
4	1	0	0.000085	0.000088	1.032678
5	32	32	0.000104	0.000091	0.869911
6	64	64	0.000112	0.000090	0.805973
7	480	480	0.000191	0.000334	1.752878
8	4064	4064	0.017113	0.048241	2.818932

9	32736	32736	0.272995	2.622098	9.604930
10	32	64	0.000103	0.000086	0.831377
11	32	480	0.000135	0.000104	0.766085
12	32	4064	0.000151	0.000265	1.757824
13	32	32736	0.000391	0.001590	4.066722
14	480	4064	0.002190	0.002190	3.002604
15	480	32736	0.023197	0.043409	1.871310
16	4064	32736	0.076937	0.283104	3.679700

Table 6.2

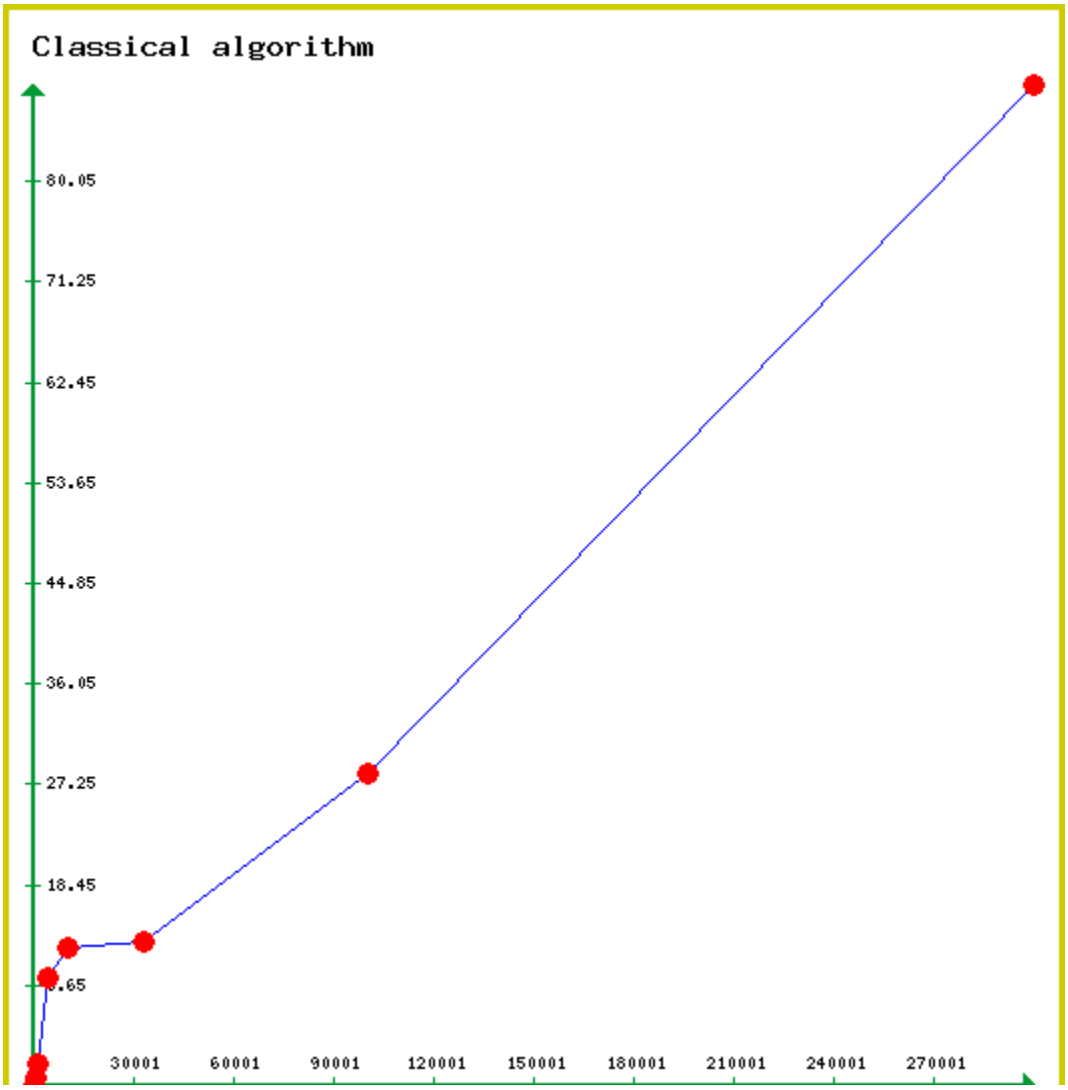
For higher bits

Number of bits for input number	Grade school algorithm	Karatsuba Algorithm
1	0.932102	0.903130
10	0.855444	0.869737
100	0.833784	0.918447
1000	2.824470	2.774192
10000	12.352405	3.528403
32736	22.99,15.71	12.061783
100000	26.930229	22.477842
300000	88.528202	70.173251

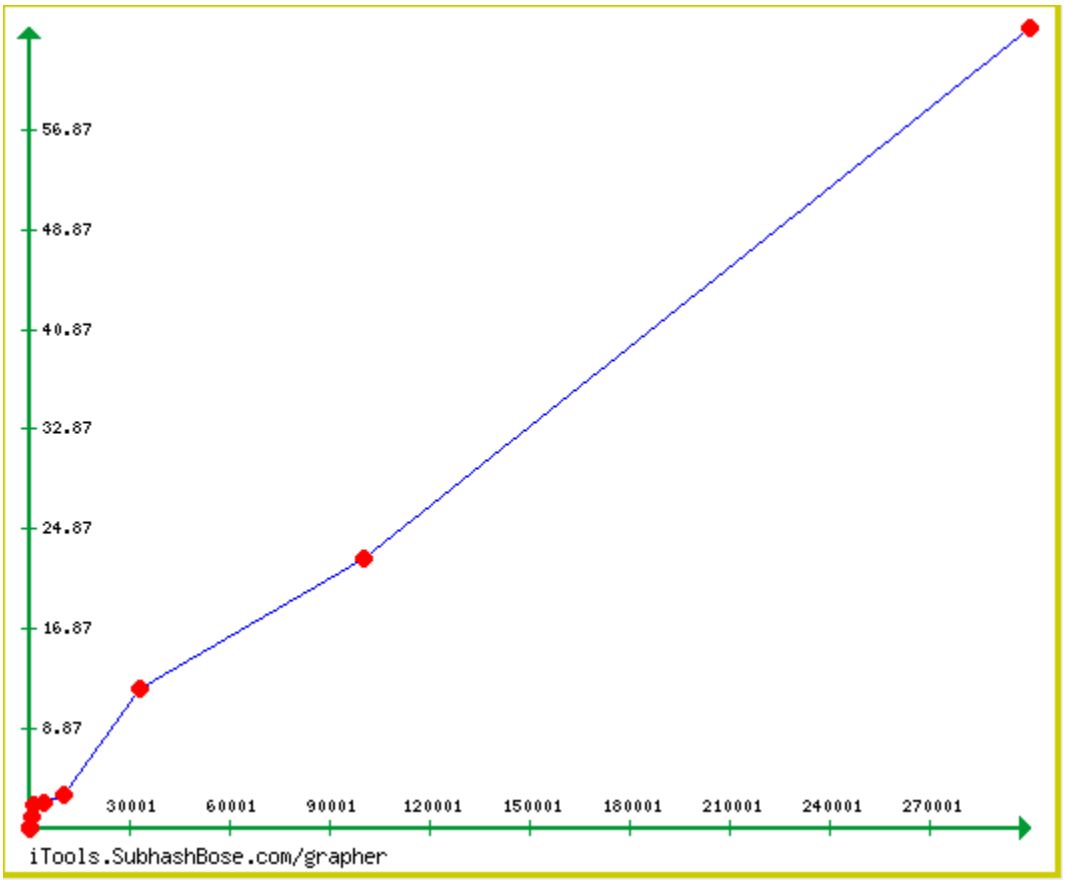
Note: values of Code/GMP ratio obtained for higher bits calculated for Karatsuba algorithm are calculated using threshold =2200 as for lower threshold values segmentation error was encountered. For all other values of number of bits below 1000 threshold value is 26

8 Graphs Analysis

Classical algorithm



Karatsuba Algorithm



9 To find threshold value for algorithm

From the given algorithm we can make following observations for multiplications of two numbers having digits m and n respectively.

A. For classical algorithm total number of multiplications are m*n in addition to n addition overhead
B. In karatsuba algorithm considering worst case scenario there are two multiplications each with numbers of length m/2 and n/2 and other multiplication for numbers of length m/2+1 and n/2+1
C. We allocate the memory space of size m/2+n/2+2 for array w3 in algorithm let cost of this operation be the size of memory thus allocated
D. There are two additions involving numbers each of length n/2 + m/2 + 2
E. And in the last phase of shifting numbers we perform two additions each involving digits of length n/2+1

Calculations:

Threshold condition can be given as follows

Grade school(T(m,n)) < Karatsuba(T(m,n))

m*n + n < m*n/4+m*n/4+m/2+n/2+1+2*n+m+8

m+8n+n < 3*m*n/4+3*m/2+5*n/2+9

m*n/4-3*n/2 < 3*m/2 + 9

10 Conclusion

Thus from the above experiment it is found that Karatsuba algorithm gives best result for numbers having large number of digits. However its performance decreases for small numbers and it becomes inefficient.

On the other hand grade school multiplication algorithm gives good results for small sized number where as performance decreases as length of number increases.

To deal with tradeoff associated with both algorithms we use threshold value set for number of bits in given numbers. If the numbers of bits in each number are above threshold then Karatsuba algorithm is utilized otherwise as number of bits fall below the threshold value, classical algorithm is used for multiplication. This operation gives the best result as we utilize the best capacity of both algorithms.

11 Future scope

There are other algorithms for fast multiplication developed in recent years. Following is one of the fastest algorithm known as Toom-3 algorithm

Toom–Cook, sometimes known as Toom-3, is a multiplication algorithm, for multiplying two large integers.

Given two large integers, sa and sb, Toom–Cook splits up a and b into k smaller parts each of length L, and performs operations on the parts. As k grows, one may combine many of the multiplication sub-operations, thus reducing the overall complexity of the algorithm. The multiplication sub-operations can then be computed recursively using Toom–Cook multiplication algorithm again. Toom-3 is special case of the Toom–Cook algorithm, where k = 3.

Toom-3 reduces multiplications and runs in $O(n^{\log(5)/\log(3)})$, about $O(n^{1.465})$ which is faster than karatsuba algorithm which runs in $O(n^{1.58})$. Thus is evident that Karatsuba algorithm is a special case of Toom–Cook, where the number is split into two smaller ones.

Thus in the future Toom-3 algorithm can be used to give better performance over karatsuba algorithm for the numbers of long length.

12.References

Web Resources

http://www.cs.indiana.edu/classes/b503/project.html

http://www.valgrind.org/

http://www.cplusplus.com/reference/

www.wikipedia.org

http://gmplib.org/

http://www.cprogramming.com/debugging

http://alexott.net/en/writings/prog-checking

http://www.cs.indiana.edu/classes/b503/resources/

Text Book resources

Analysis of Algorithms by Paul purdom and Cynthia Brown

Project understanding in collaboration with Prithvi Raj