ANALYSIS OF FAST MULTIPLICATION ALGORITHM

As part of course work for B503

Under Supervision of Prof. Paul Purdom

By

Ramakant Khandel
0003080002
Fall 2013
MS in CS

TABLE OF CONTENTS

1) Abstract:

The details of implementation and analysis of karatsuba algorithm and traditional algorithm is included in this report. The Karatsuba algorithm uses divide and conquer approach to divide numbers into words and once the threshold is reached traditional algorithm is used to multiply the numbers.

2) Introduction:

The traditional algorithm for multiplying two n bit numbers requires time O (n^2). The time complexity for traditional algorithm is small for small numbers but increases quadratically for large numbers.

The Karatsuba algorithm uses divide and conquer approach for multiplication of numbers. The Karatsuba algorithm requires lesser number of multiplications to be done and saves reasonable amount of time. The Karatsuba algorithm has an overhead of addition and subtraction compared to traditional algorithm but this operation requires less time compared to multiplication.

This algorithm uses the best of traditional which has a good performance for small numbers and Karatsuba which has a good performance for large numbers. TheKaratsuba algorithm divides the number recursively till a threshold is reached and once it is reached it computes value using traditional algorithm.

3) Traditional Algorithm:

In traditional algorithm we compute the result by multiplying each digit of one number with the digit of another number to compute the result.

3-a) Table for Time Taken by Algorithm 1.8 and GMP:

Sr. No	Number of bits for number 1	Number of bits for number 2	Time taken By code(seconds)	Time taken by GMP (seconds)	Ratio(code/GMP)
1	1	1	0.000066	0.000068	0.969274
2	0	0	0.000065	0.000065	0.999245
3	0	1	0.000065	0.000066	0.992827
4	1	0	0.000066	0.000068	0.977304
5	32	32	0.000065	0.000078	0.838097
6	64	64	0.000070	0.000086	0.809424
7	480	480	0.000273	0.000151	1.811764
8	4064	4064	0.013404	0.001901	7.051948
9	32736	32736	0.855298	0.040265	21.241532
10	32	64	0.000068	0.000084	0.805037
11	32	480	0.000080	0.000092	0.865307
12	32	4064	0.000183	0.000141	1.291577
13	32	32736	0.001014	0.000333	3.044902
14	480	4064	0.001650	0.000460	3.583326
15	480	32736	0.012683	0.002754	4.604934
16	4064	32736	0.106044	0.013581	7.808346
17	100000	100000	8.019797	0.197172	40.674016

*Fig 1:*Traditional Time Stamp

3-b) Graphical Representation for number of input size vs. time taken by code(ms) For algorithm 1.8

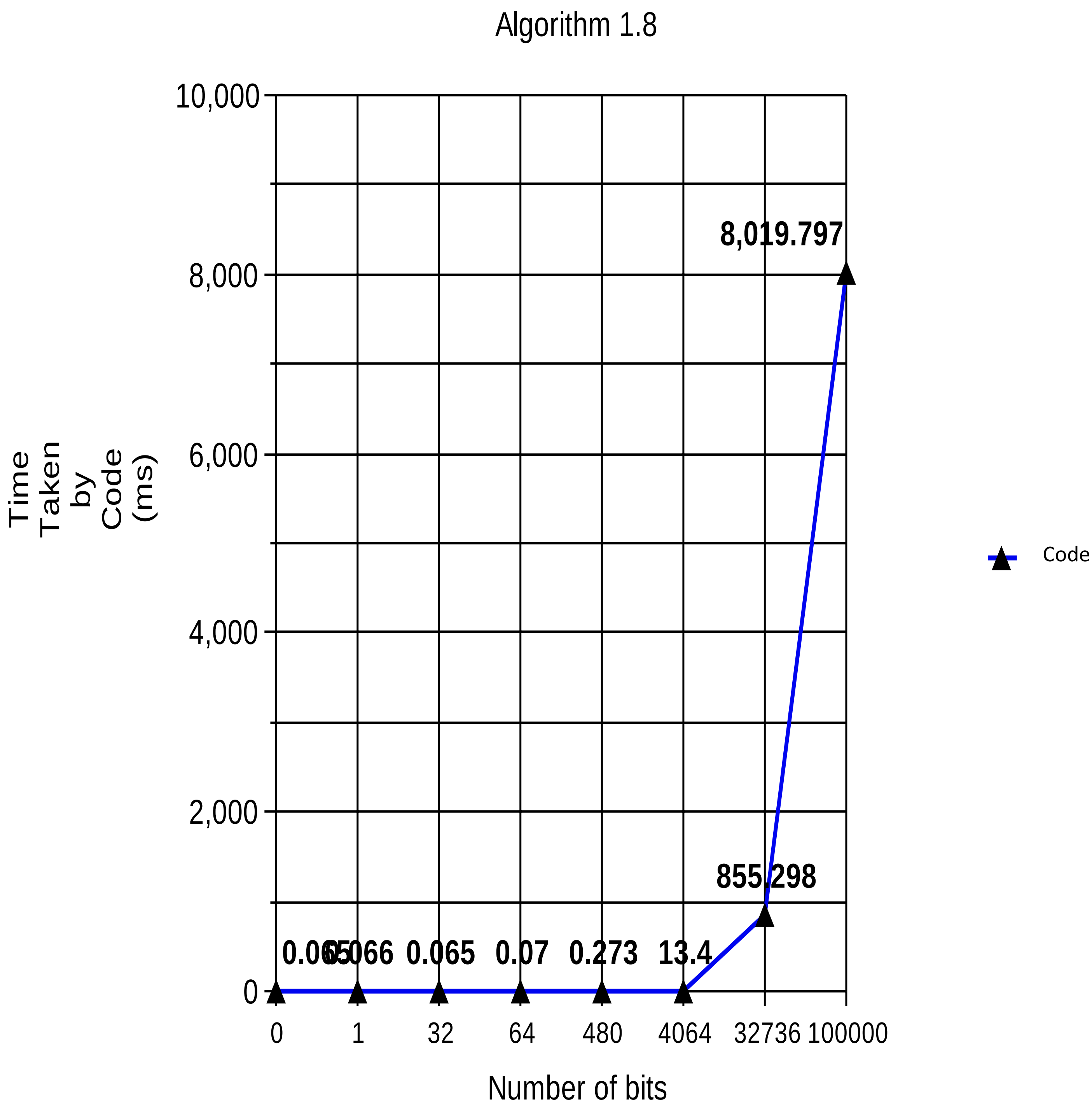

Fig 2: Time stamp (code) for 1.8

3-c) Graphical Representation for number of input size vs. time taken by GMP(ms)
For algorithm 1.8

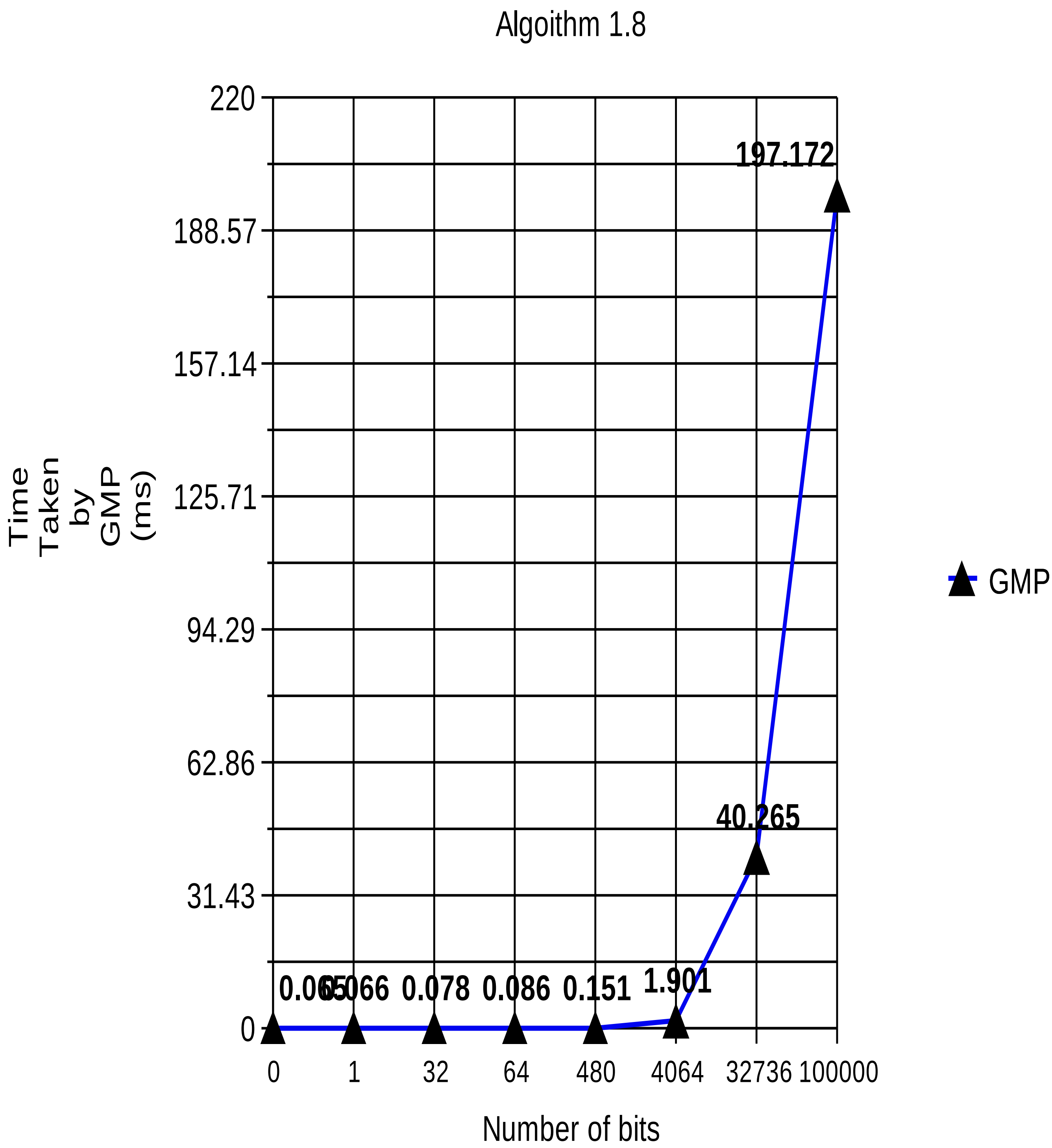

Fig 3: Time Taken GMP

4) Karatsuba Algorithm:

Karatsuba algorithm is specially used for multiplication of large numbers. This algorithm uses the divide and conquer technique. This algorithm multiplies two 2n bit numbers and breaks it into multiplication of two pairs of n-bits numbers. As compared to the traditional algorithms, this algorithm has overheads for additions and subtractions but as multiplication is faster, the overall time complexity is better than the traditional algorithm.

4-a)Complexity:

As per Knuth [1998], the complexity of this algorithm O ($n^{\log_2 3}$) which is approximately equal to O ($n^{1.58}$).

4-b)Algorithm:

Input: The 2n-bit number $U = U_1 2^n + U_2$ where $0 \le U_1 < 2^n$ and $0 \le U_2 < 2^n$, and the 2n bit number $V = V_1 2^n + V_2$ where $0 \le V_1 < 2^n$ and $0 \le V_2 < 2^n$.

Output: The product UV represented by $UV = W_1 2^{3n} + W_2 2^{2n} + W_3 2^n + W_4$.

Step 1: Set $T_1 \leftarrow U_1 + U_2$

Step 2: Set $T_2 \leftarrow V_1 + V_2$

Step 3: Set $W_3 \leftarrow T_1 T_2$

Step 4: Set $W_2 \leftarrow U_1 V_1$

Step 5: Set $W_4 \leftarrow U_2 V_2$

Step 6: Set $W_3 \leftarrow W_3 - W_2 - W_4$

Step 7: Set $C \leftarrow \lfloor W_4/2^n \rfloor$ and $W_4 \leftarrow W_4 \bmod 2^n$

Step 8: Set $W_3 \leftarrow W_3 + C$, $C \leftarrow \lfloor W_3/2^n \rfloor$ and $W_3 \leftarrow W_3 \bmod 2^n$

Step 9: Set $W_2 \leftarrow W_2 + C$, $W_1 \leftarrow \lfloor W_2/2^n \rfloor$ and $W_2 \leftarrow W_2 \bmod 2^n$

4-c) Table for Time Taken by Algorithm 5.2 and GMP:

Sr. No.	Number of bits for number 1	Number of bits for number 2	Time Taken by code(seconds)	Time Taken by GMP(seconds)	Ratio(code/GMP)
1	1	1	0.000066	0.000069	0.964327
2	0	0	0.000067	0.000068	0.976960
3	0	1	0.000067	0.000065	1.031547
4	1	0	0.000068	0.000066	1.018863
5	32	32	0.000069	0.000076	0.909828
6	64	64	0.000071	0.000087	0.816963
7	480	480	0.000270	0.000144	1.875710
8	4064	4064	0.015077	0.001967	7.665276
9	32736	32736	0.501655	0.041532	12.078897
10	32	64	0.000071	0.000084	0.839901
11	32	480	0.000083	0.000090	0.919003
12	32	4064	0.000187	0.000133	1.406400
13	32	32736	0.001012	0.000335	3.017106
14	480	4064	0.001636	0.000456	3.588635
15	480	32736	0.012651	0.002730	4.633762
16	4064	32736	0.472650	0.014455	32.697460
17	100000	100000	2.897670	0.201601	14.373273

Fig 4: Time Ratio for code and GMP for algorithm 5.2

4-d) Graphical Representation for number of input size vs time taken taken by code(ms) For algorithm 5.2

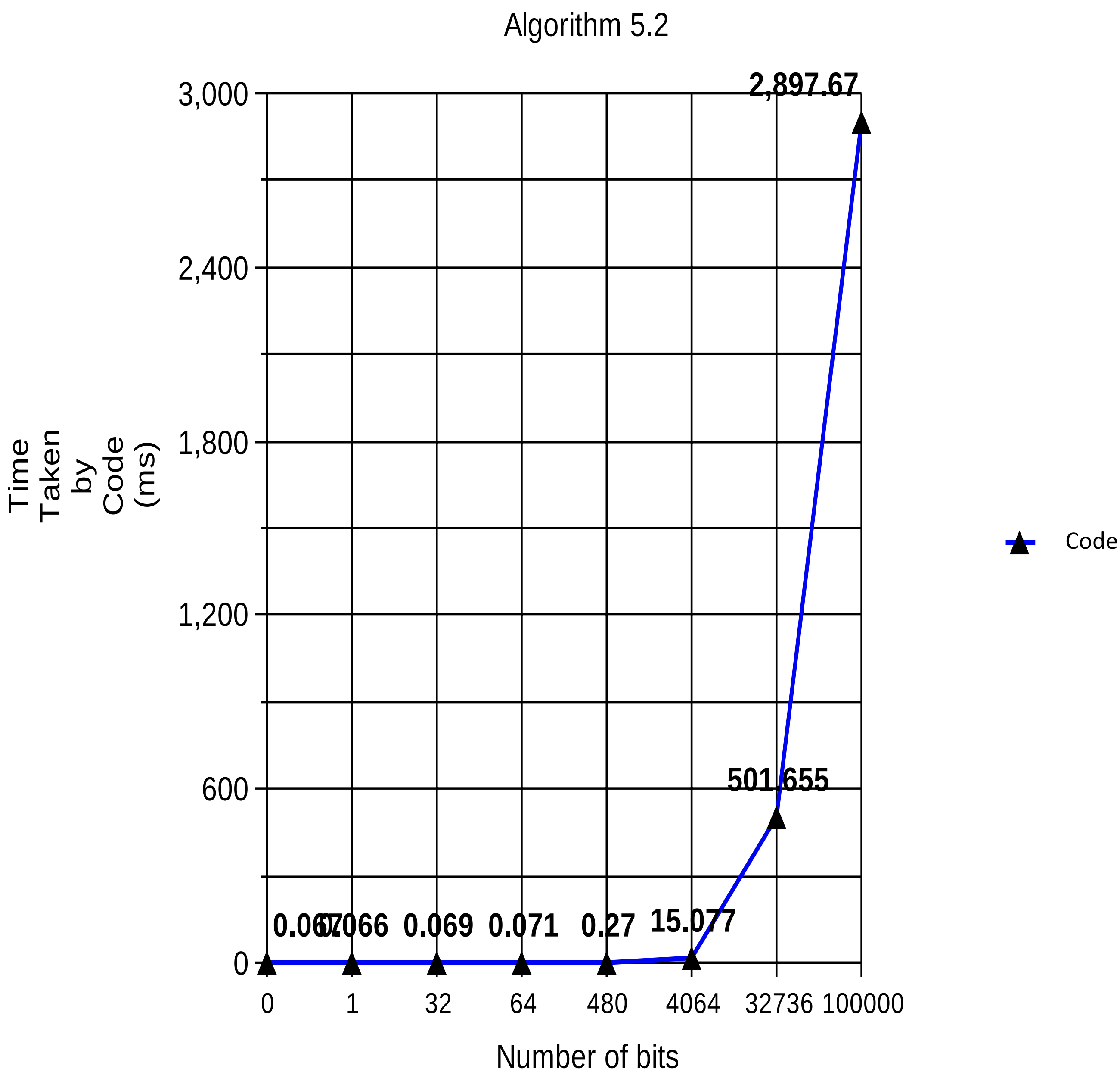

Fig 5: Time Taken(code) for 5.2

4-e) Graphical Representation for number of input size vs time taken taken by GMP(ms) For algorithm 5.2

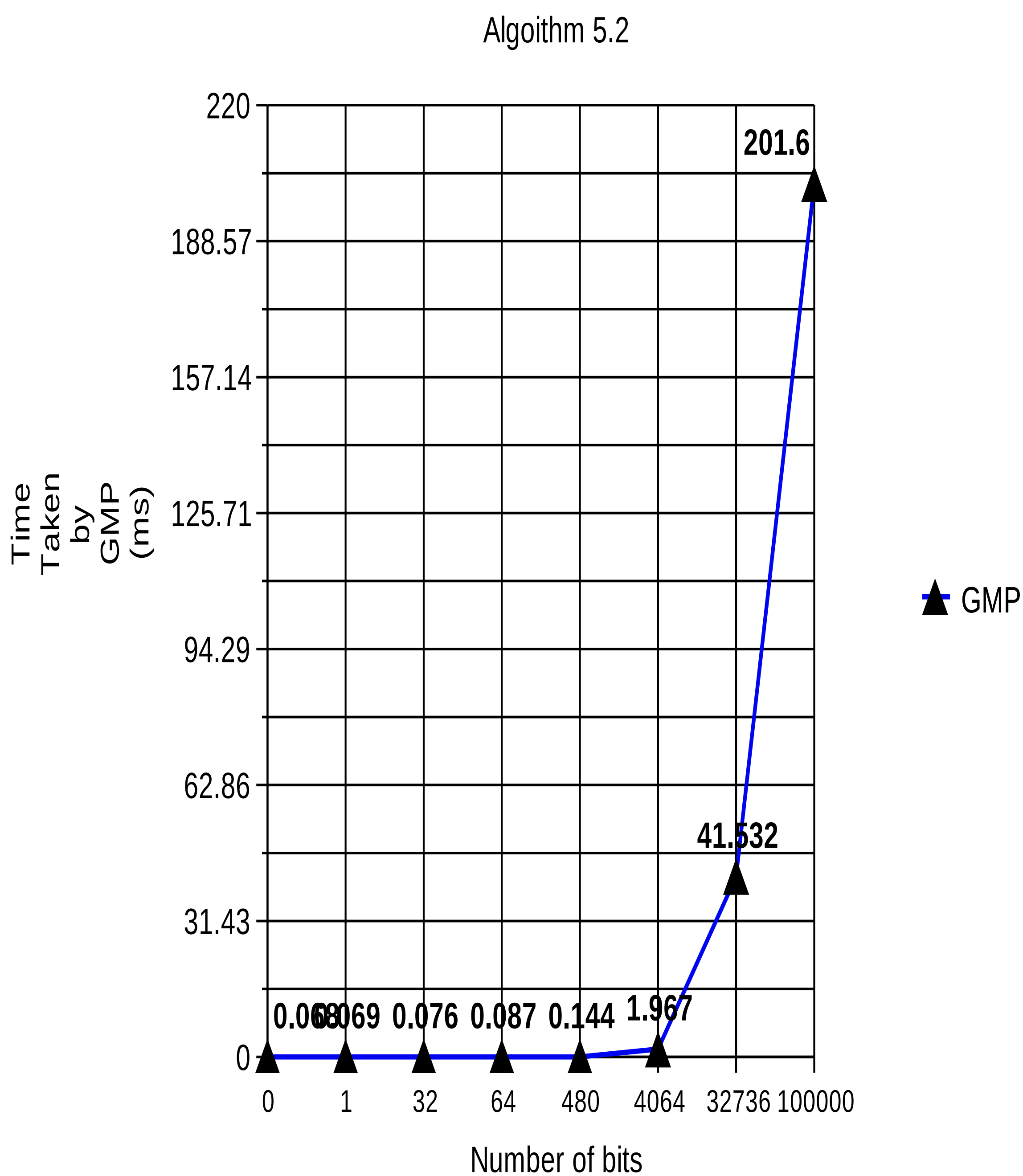

Fig 6: Time taken GMP

4-f) Implementation

Karatsuba algorithm is implemented in scaffold32.c and the base for the same is 2^{32}.

1) Check for word size for any number(27 ideally my derivation is 35)
If word size <=27
Traditional Multiplication
Else
Recursive Multiplication

2) Calculation for numberofwords (division of words for optimum results)
a) For equal size words.
Numberofwords=sa/2 or sb/2;
b) For unequal size words
b-1) If sa<sb
sa=sb(by appending zero to the first number)
numberofwords=sa/2 or sb/2
b-2) ifsb>sa
sb=sa(by appending zero to the second number)
numberofwords=sa/2 or sb/2

3) num1_suma=add (cint_a,cint_mida)

4) num2_sumb=add(cint_b,cint_midb)

5) cint_w3=recursive_multiply(num1_suma, num2_sumb)

6) cint_w2= recursive_multiply(cint_a, cint_b)

7) cint_w4= recursive_multiply(cint_mida, cint_midb)

8) cint_w3=subtract(cint_w3, cint_w2)

9) cint_w3=subtract(cint_w3, cint_w4)

10) div(cint_w4, numberofwords)

Remainder-remw4

Quotient-quotientw4

11) div(cint_w3, numberofwords)

Add(cint_w3,quotientw4)

Remainder-remw3

Quotient-quotientw3

12) div(cint_w2, numberofwords)

Add(cint_w2,quotientw3)

Remainder-remw2

Quotient-quotientw2

13)quotientw1=quotientw2

4-g)Time Complexity

Multiplication: 3T(n/2)

Addition: 2(n/2)+2n

Subtraction: 2n

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

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

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

.

$= 3(3(3(3(\ldots (3T(n/2^k) + 5(n/2^k) + 5(n/2^3) + 5(n/2^2) + 5(n/2)))))) + 5n$

$= 3^k (T(n/2^k) + 5n(1 + (3/2)^1 + (3/2)^2 + (3/2)^3 + \ldots + (3/2)^k))$

$= 3^{\lg n} + 10n((3/2)^k - 1)$

$= n^{\lg 3} + 10n*((3/2)^{\lg n} - 1)$

$= n^{\lg 3} + 10n^{\log 3} - 10n$

$\approx 11n^{1.58} - 10n$

Considering the above recurrence equation and according to the book [1] time complexity for traditional algorithm is $(n^2 + n)$.

For the traditional algorithm to be more efficient than Karatsuba algorithm,

$(n2 + n) < 3T (n/2) + 5n$

$< 3((n/2+1)^2 + (n/2 + 1)) +5n$

$< 3(n^2/4 + n + 1 + n/2 + 1) + 5n$

$< 3(n^2/4 + 3n/2 +2) + 5n$

$< 3n^2/4 + 9n/2 + 6 + 5n$

$< 3n^2/4 + 19n/2 + 6$

$n^2-34n-24<0$

Finding the roots of the equation we get $n \approx 34.69 =35$

Hence threshold is 35.

This is different from the threshold mentioned in the book.

5) Software used for implementation:

Operating System: Unix
Programming Language: C
Compiler: GNU
Library: GMP library which used for arbitrary precision arithmetic, operating on signed integers, rational numbers, and floating point numbers generation.

6) Conclusion:

The Karatsuba algorithm gives best result when numbers are large and traditional algorithm when numbers are small. However performance for traditional algorithm decreases for large numbers whereas, for Karatsuba decreases for small numbers.

The current algorithm uses best of the above two algorithms. If the number of bits is above the threshold Karatsuba algorithm is used and if it is less than the threshold traditional algorithm is used.

7) Future Scope:

The ratio of time taken by code to time taken by GMP for the current implementation can be improved further. The current implementation appends zero to the smaller of the two numbers so that the two numbers have same number of words. This logic can be changed and the ratio can be improved. Further code can be optimized by improving the logic for addition, multiplication, subtraction and division.

8) References:

Web Resources:

http://www.cs.indiana.edu/classes/b503/project.html
http://www.cplusplus.com/reference/
http://www.wikipedia.org
http://gmplib.org/
http://www.cprogramming.com/debugging

9) Text Book resources

Analysis of Algorithms by Prof. Paul Purdom and Prof. Cynthia Brown.

Project Understanding in collaboration with Ratish Dalvi.