DAA Assignment No. : 3
To multiply two complex numbers.

Lovely Digra
IIT2017090

Abhishek Vishwakarma
IIT2017091

Anuj Raj
IIT2017092

***Abstract*—Our required aim is to design and analyze an optimal algorithm to perform multiplication of two complex numbers without using multiplication,division or modulo operator. A 1-D lookup table is created to get the result by the multiplication of two single digit numbers in order to get product of real-real, real-imaginary or imaginary-imaginary parts. While designing our problem we calculated the time complexity for the algorithm and plotted graphs of no. of digits of nos. vs time elapsed.**

I. INTRODUCTION

To design and analyse algorithm for multiplication of two nos. without using multiplication, division or modulo operator and use this application to multiply two complex nos.

With the advancement of technology certain basic computations is taken for granted by the programmers.One such example is arithmetic multiplication of two nos. Suppose the system hardware does not have architecture required for multiplication using '$*$' operator.

Now the question arises how to perform single digit multiplication without using an asterisk?

For this, A 1-D look-up table is created to get the result by the multiplication of two single digit numbers.

Now what is a look-up table?

A Look-Up table is an array that replaces run time computation with a simpler array indexing operation. It can be 2D,3D, single input single output, multi input multi output.

II. ALGORITHM DESIGN AND EXPLANATION

For the problem, we followed the following approach to design an algorithm:

Approach 1: 2-D array

For multiplication of two complex nos. of the form (a+ib and c+id), we separate real and img part of the result (ac-bd + i(ad+bc)) and perform multiplication of simply two floating point nos.

For multiplication of two numbers without using arithmetic "*" operation, take a digit from the multiplier and check its product with every digit of the multiplicand using the look-Up table.

Since for multiplying two nos. using the look-Up table main problem lies in accessing digits of a number separately.

The key idea is to use string operations and string-Streams for accessing and manipulating digits separately.

Firstly, store the product of all single digit numbers in 1-D look-Up Table.Product of a and b can be found in the look-up table in location "9*(a-1)+(b-1)". For calculating 9*(a-1) repeated addition comes handy. Use the look-Up table for the product of num1[i] with num2[j].

Store the intermediate result of product of num1 with each num2[j] in a 2-D Matrix, like the way a primary school students does .

Finally add the intermediate results in column fashion to get the final result.

Approach 2: Optimization

Instead of storing intermediate values in 2-D matrix concatenate the result into a string and parse it to get the intermediate result.

Now addition becomes very simple . Just add it to the variable storing the final result.

Repeat the process for every digit of num2 to get the final result. The pseudo-code for this algorithm is:-

Algorithm 1 PseudoCode for Approach 2:

```
procedure mul(int a)
    for all i = 1 to a do
        //a times repeated addn of c where is 9
    return c
procedure lookup(int a, int b)
    return look-Up[mul(a − 1)+(b − 1)]
a ← Real part of 1st num
b ← Img part of 1st num
c ← Real part of 2nd num
d ← Img part of 2nd num
num1String ← parsed a to string
num2String ← parsed c to string
reverse(num1Str)
reverse(num2Str)
for all i = 1 to no of digits in num1 do
    carry ← 0
    for all j = 1 to no of digits in num2 do
        product ← lookup(num1Str[i], num2Str[j])
        product ← product + carry
        result ← unit_place_of_product
        carry ← tens_place_of_product
        intermediate_result ← concatenate(result)
        procedure reverse(intermediate_result)
        finalResult ← finalResult + intermediateRes
    print(finalResult)
```

III. EXPERIMENTAL STUDY

For the experimental analysis of this algorithm, we will take same values of real and imaginary part for both complex numbers for easy calculations and find product of real-real,real-imaginary or imaginary-imaginary parts. A .dat file is generated containing duration for each corresponding num value in a tabluated manner(see Fig 1).

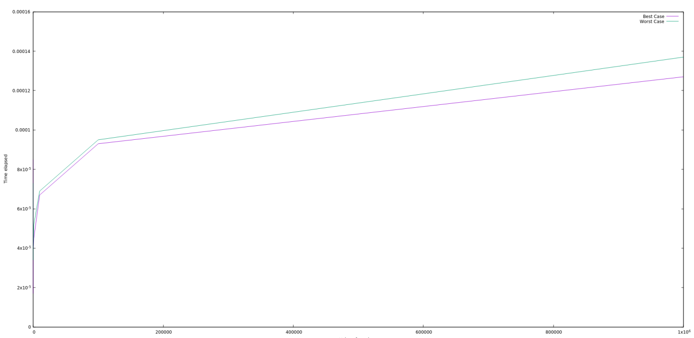


Fig. 1. Graphical Explanation

IV. NUMERICAL ANALYSIS

Data with error analysis is attached(see Fig. 3).Floating point operations has rounding errors and its datatype has fixed precision while double datatype has double times precision than that of float.Therefore, error in double is much much less than that of float (see Fig 2) .

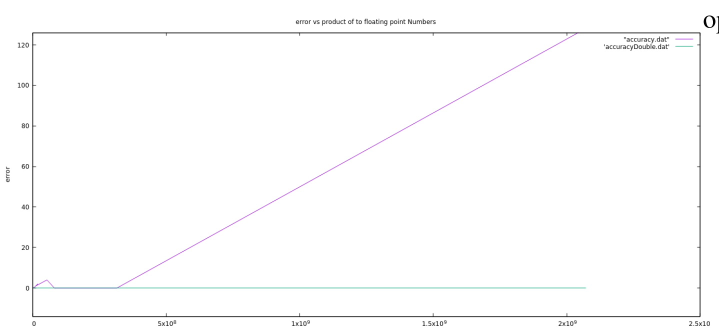
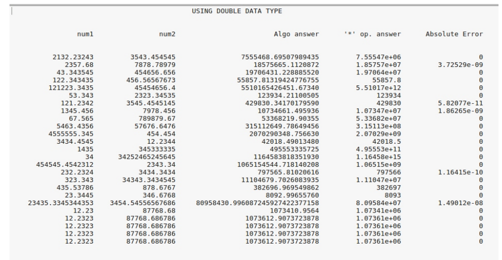


Fig. 2. Error analysis graph



USING DOUBLE DATA TYPE

num1	num2	Algo answer	'*' op. answer	Absolute Error
2132.23243	3543.454545	7555468.69507989435	7.55547e+06	0
2357.68	7878.78979	18575665.1120872	1.85757e+07	3.72529e-09
43.343545	454656.656	19706431.228885520	1.97064e+07	0
122.343435	456.56567673	55857.81319424776755	55857.8	0
121223.3435	45454656.4	5510165426451.67340	5.51017e+12	0
53.343	2323.34535	123934.21100505	123934	0
121.2342	3545.4545145	429830.34170179590	429830	5.82077e-11
1345.456	7978.456	10734661.495936	1.07347e+07	1.86265e-09
67.565	789879.67	53368219.90355	5.33682e+07	0
5463.4356	57676.6476	315112649.78649456	3.15113e+08	0
4555555.345	454.454	2070290348.756630	2.07029e+09	0
3434.4545	12.2344	42018.49013480	42018.5	0
1435	345333335	495553335725	4.95553e+11	0
34	34252465245645	1164583818351930	1.16458e+15	0
454545.4542312	2343.34	1065154544.718140208	1.06515e+09	0
232.2324	3434.3434	797565.81020616	797566	1.16415e-10
323.343	34343.3434545	11104679.7026083935	1.11047e+07	0
435.53786	878.6767	382696.969549862	382697	0
23.3445	346.6768	8092.99655760	8093	0
23435.3345344353	3454.54556567686	80958430.996087245927422377158	8.09584e+07	1.49012e-08
12.23	87768.68	1073410.9564	1.07341e+06	0
12.2323	87768.686786	1073612.9073723878	1.07361e+06	0
12.2323	87768.686786	1073612.9073723878	1.07361e+06	0
12.2323	87768.686786	1073612.9073723878	1.07361e+06	0
12.2323	87768.686786	1073612.9073723878	1.07361e+06	0

Fig. 3. Error Analysis Data

V. TIME COMPLEXITY AND DISCUSSION

The process of accessing product from the look-Up table at max 9 iterations in the repeated addition loop of mul(9) for function and 0 iterations for mul(0). This happens for every single digit pair of the multiplicant and the multiplier. Thus, the complexity depends in the number of digits of multiplier , number of digits in the multiplicant and value of the single digit of each pair of the two nos. For simplicity take same value for both the nos. That is num1 = num2 = n.

So worst case and best case are defined for total digits in the given no.

Worst Case : each single digit pair of integers is 9.
growth rate : $O(9 * d * d)$ (see fig.1, green line)
Best Case : each single digit pair contains atleast one 0.
growth rate : $\omega(1 * d * d)$ (see fig.1 voilet line)
where d is the total digits in the given integer.

Nature of the graph: The graph Time vs Value of Number will be logarithmic of base 10 (See Fig.1) .

This is because total digits in a number is equal to $[log_1 0(num)] + 1$

VI. CONCLUSION

We designed an algorithm to perform multiplication of two complex nos. with the help of a 1-D lookup table. We also analyzed the algorithm and plotted the graph for Value of nos vs time elapsed.

One of the possible application of the project can be in designing Compiler for systems (say, embedded systems) with native processors without the architecture for multiplication operations.