Multiplication Algorithms

Submitted by

Navneet Tewatia (Sec.-02)

Roll no.-202151097

A multiplication algorithm is algorithm to multiply two numbers. Depending on the size of the numbers, different algorithms are used. Efficient multiplication algorithms have existed since the advent of the decimal system.

Some of them are:

➢ *Booth's Multiplication Algorithm*

Booth's algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation.

Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed. Booth's algorithm is of interest in the study of computer architecture.

Here's the implementation of the algorithm.

Example:

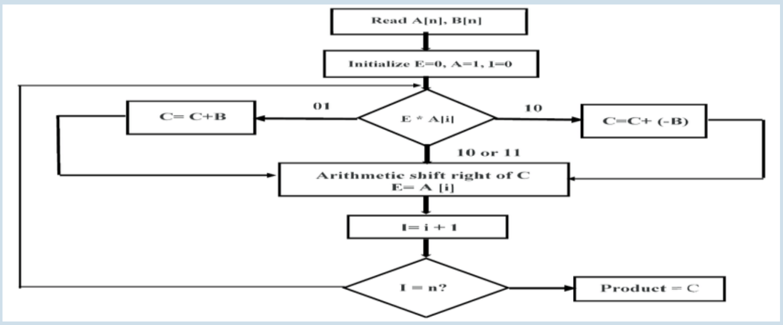
```
Input :  0110, 0010
Output :  qn      q[n+1]                 AC      QR      sc(step
                                                              count)
                        initial          0000   0010          4
          0       0     rightShift       0000   0001          3
          1       0     A = A - BR       1010
                        rightShift       1101   0000          2
          0       1     A = A + BR       0011
                        rightShift       0001   1000          1
          0       0     rightShift       0000   1100          0
Result=1100
```

Algorithm:

Put multiplicand in BR and multiplier in QR and then the algorithm works as per the following conditions:

- *If Q_n and Q_{n+1} are same i.e. 00 or 11 perform arithmetic shift by 1 bit.*
- *If Q_nQ_{n+1} = 10 do A = A + BR and perform arithmetic shift by 1 bit.*
- *If Q_nQ_{n+1} = 01 do A = A – BR and perform arithmetic shift by 1 bit.*

Flowchart:



➢ *Karatsuba Multiplication Algorithm*

*The **Karatsuba algorithm** is a fast multiplication algorithm that uses a Divide and conquer approach to multiply two numbers. It was discovered by **Anatoly Karatsuba** in 1960 and published in 1962.*

This happens to be the first algorithm to demonstrate that multiplication can be performed at a lower complexity than O(N^2) which is by following the classical multiplication technique. Using this algorithm, multiplication of two n-digit numbers is reduced from O(N^2) to O(N^log3) that is O(N^1.585).

Algorithm:

1. *Divide the numbers x and y into two equal half. Let's indicate the two half of x as x left and x right. Similarly for y as y left and y right.*
2. *Multiply x left and y left value.*
3. *Find the sum of x left and x right, y left and y right, and multiply the two sum.*
4. *Multiply the x right and y right values.*
5. *Subtract the value from step3 – step2 – step5.*
6. *Finally, add the value of step2 + step4 + step5 with appropriate padding of 0.*

Application:

The Karatsuba algorithm is very efficient in tasks that involve integer multiplication. It can also be useful for polynomial multiplications.

➢ *Toom-Cook Algorithm*

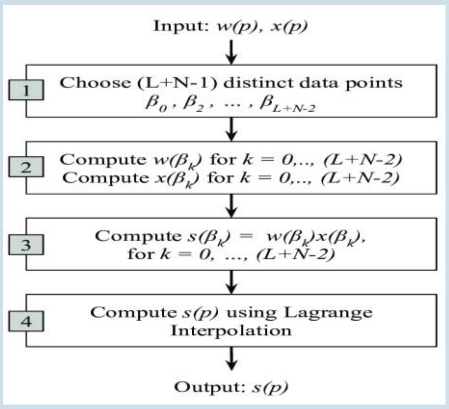
It is an algorithm for multiplying two n digit numbers in O(N^1.46) time.

The idea is based on divide and conquer technique. Given two large integers, a and b, 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 again, and so on.

<u>Applications:</u> There are various areas where this algorithm application is done, which involves multiplication of large numbers.

1. *This method is used in McEliece Cryptosystems to overcome the drawbacks in terms of size of the encrypted key and transmission rate.*
2. *Big number arithmetic.*
3. *In cryptographic algorithms especially for reducing the complexity in encoding and decoding of the keys like ElGamal, RSA, Elliptical Curvecryptosystems and Diffie Hellman key exchange protocol.*
4. *Calculation of mathematical constants like Pi, e etc.*

<u>Flowchart:</u>

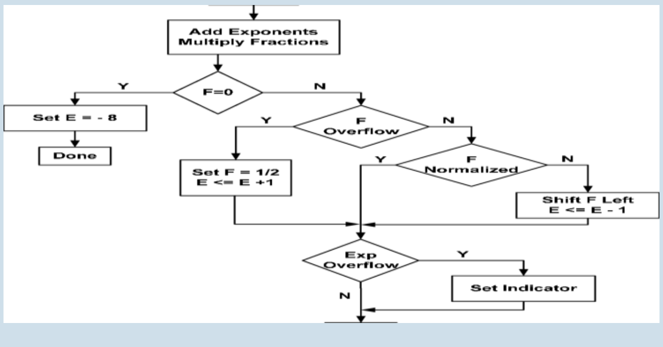

Input: $w(p)$, $x(p)$
1
Choose (L+N-1) distinct data points $\beta_0, \beta_2, \ldots, \beta_{L+N-2}$
2
Compute $w(\beta_k)$ for $k = 0,\ldots, (L+N-2)$
Compute $x(\beta_k)$ for $k = 0,\ldots, (L+N-2)$
3
Compute $s(\beta_k) = w(\beta_k)x(\beta_k)$, for $k = 0, \ldots, (L+N-2)$
4
Compute $s(p)$ using Lagrange Interpolation
Output: $s(p)$

➢ *<u>Multiplying Floating Points Numbers</u>*

<u>Algorithm:</u>

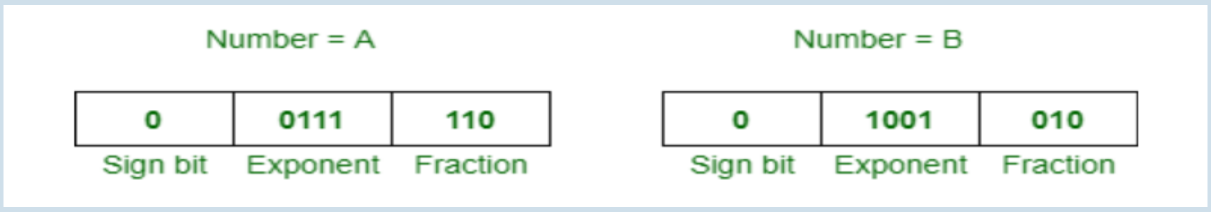
1. *Convert these numbers in scientific notation, so that we can explicitly represent hidden 1.*
2. *Let 'a' be the exponent of x and 'b' be the exponent of y.*
3. *Assume resulting exponent c = a + b. It can be adjusted after the next step.*
4. *Multiply mantissa of x to mantissa of y. call this result m.*
5. *If m does not have a single 1 left of radix point, then adjust radix point so it does, and adjust exponent c to compensate.*
6. *Add sign bits, mod 2, to get sign of resulting multiplication.*
7. *Convert back to one byte floating point representation, truncating bits if needed.*

<u>Flowchart:</u>

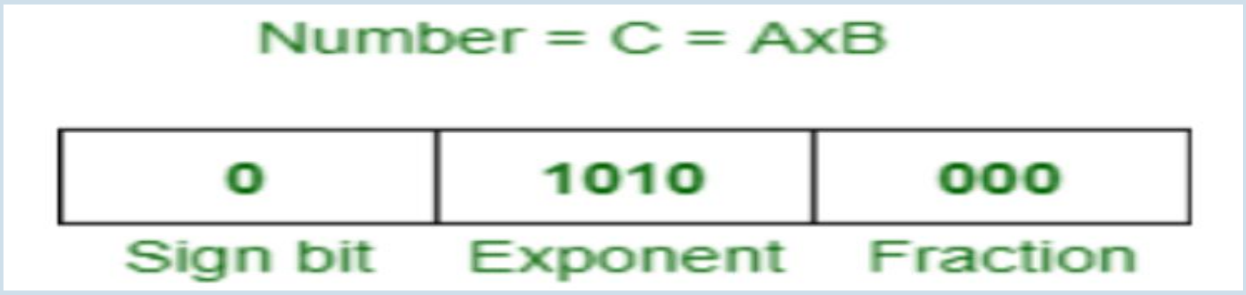


<u>Example:</u>

Suppose you want to multiply following two numbers:



Therefore, resultant number is,



References:

- *Booth's Multiplication Algorithm - GeeksforGeeks*
- *karatsuba algorithm for big integer multiplication » DrukLearn (drukinfotech.com)*
- *Karatsuba Algorithm (for fast integer multiplication) (opengenus.org)*
- *Toom Cook method for multiplication (opengenus.org)*
- *Multiplying Floating Point Numbers - GeeksforGeeks*
- *Multiplication Algorithm & Division Algorithm Notes - Computer Science Engineering (CSE) (edurev.in)*
- *Flowchart for floating-point multiplication | Download Scientific Diagram (researchgate.net)*
- *flowchart of the karatsuba algorithm - Bing*

Thank You