Title: Multiplication of any number using Left Shift

Authors: Baldha Prashantkumar Mansukhbhai (pmbaldha@gmail.com)

Abstract: The new algorithm for multiplication. The multiplication algorithm is best for multiplication algorithm in some cases.

Content:

When multiplications of two numbers of having digit n occurred, the complexity of this algorithm is $O(n^{1.585})$ or like any other which are not optimal.

Now I am presenting algorithm for multiplication which have complexity is dynamic. This algorithm has two cases:

1) Best case
2) Average case
3) Worst case

This algorithm is based upon shift operation.

Multiplication using shift left operation:

When shifting left performed on signed number, the number is multiplied by $(2^{s),}$ where s indicates number of time shifting performed.

Let take one example:

Our number is 2, its binary form is 010. Now perform shift left operation on the binary formed number, and then we get 100 which is equivalent to 4. Now if we shifting operation performed two times, then we get 1000 which is equivalent to 8 in decimal.

Now suppose 2 is multiplier, and we shift it two times and get answer is 8,so multiplicand is 4 which is equal to 2^2, So multiplier is 2^s.

Table associate number of left shifting and correspond multiplicand:

Number of left shift operation performed on number	The answer is equal to that number has multiplier with below multiplicand
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768
16	65536
17	131072
18	262144
19	524288
20	1048576
21	2097152
22	4194304
23	8388608
24	16777216
25	33554432
26	67108864
27	134217728
28	268435456
29	536870912
30	1073741824
31	2147483648
32	4294967296
33	8589934592
34	17179869184
35	34359738368
36	68719476736

New Algorithm for multiplication:

Setup phase:

In Setup phase, We create table as denoted above. This is static table. So no require it update every time when operation performed.

In that table first column give how many time shift will occur for result. And second column indicates correspond multiplicand values.

Performance phase:

This phase is performed each time when we want to multiplication performed by this method.

First, we search multiplier location in table which created in setup phase using binary search.

Function binsearch(T[1..n],am)

If n==0 or am > t[n] then return n+1 and apply ordinary method for multiplication

Else return binrec(T[1..n],am)

Function binrec(T[1..j],am)

If i== j return i

K=(i+j)/2;

If am<=t[k] then return binrc(t[i..k],am)

Else return binrec(t[k+1..j,am)

The binary search time complexity is

After searching location of multiplier in table, we take above row multiplier for consideration. And take difference between actual multiplier and considedered multiplier from table.

D= mct – am

Here, D=difference

mct= multiplier considered from table

am= actual multiplier

Now shift left multiplicand mct times using loop

For i=1 to mct do

Temp=Shift_ left (multiplicand);

And then add D times multiplicand in Temp value

For j=1 to D do

Ans = Ans + Actual multiplier

Example computation using new algorithm:

Let take multiplicand is 0945 and multiplier is 1235., So binary search give location between 1024 and 2048. But we have to consider below range so we consider 1024 as multiplicand and lest shift 10 times multiplier.

Now we take difference between actual multiplier and a multiplier considered from table. This difference comes 211. So we add 211 times multiplicand to final shifted value.

In above example if we assume that one element operation take 1 ms then total time is 10+211=311ms for computing above example. If we calculate above example through traditional algorithm it takes time for same assumption is 0945*1235=1167075ms.

Note:

In same way we can reduce time complexity of division.