KARATSUBA ALGORITHM

-Vignesh Reddy Lingala

From the early stages of the Mathematics we have been using traditional Arithmetic operations like Addition, Subtraction, Multiplication ,Division etc. But things were being changing since early 1900’s because of computer revolution. Because of this the way of operations has been changed and many modern techniques were imposed. Multiplication was one of them.

In the year 1960 Anatoly Karatsuba discovered the fastest multiplication by which we can reduce some operations and save time.

Drawback of Traditional Multiplication

Let us consider two a 4 digit number’s

X=1234.

Y=9876.

Now multiply it in a traditional way:

1. 2 3 4

\* 9 8 7 6

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_

(6\*1) (6\*2) (6\*3) (6\*4)

X X X X 0

Y Y Y Y 0 0

Z Z Z Z 0 0 0

* From the above example we can say that Every digit of “ Y ”

Is multiplied with every digit of “ X ” .By counting the number of multiplications then total multiplications

are termed as ‘ 16 ’.

* We can clearly say that for every n\*n digit multiplications occurs then there are ‘ n\*n ’ i.e. [n^2].
* In the computer basis the time complexity is ‘ O(n^2) ’.

ANATOLY KARATSUBA

* He stated that the number of multiplications required for a multiplication can be reduced by following rules to multiply .
* Which paved a way to finding an algorithm famously known as

“ KARATSUBA ALGORITHM ” .

* Karatsuba algorithm still it is the fastest multiplication algorithm exists.
* Which uses a Recursive algorithm to solve it.
* In this Divide and Conquer method is used to solve the recursion & which makes the algorithm faster.

Basic terms to be known to solve Karatsuba Algorithm

Atomic element: After continuous division of an element from a group if it stands alone is called an atomic element.

Atomic Addition: The addition which was taken over on the atomic elements is known as Atomic Addition (α) read as ‘alpha’.

Atomic Multiplication: The multiplication which were done on the atomic elements is known as Atomic Multiplication ( µ).read as ‘mu’.

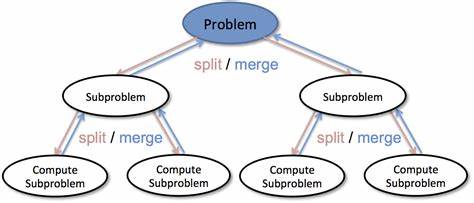
Divide & conquer --------what ? and why?...........

* **What is divide and conquer?**

Divide and conquer is the naive approach of solving problems.

In which the given problem is divided in to sub problems and this division goes on till the elements in the problem reaches atomic level and from the atomic level all the problems are solved and hence merging of all the sub-problem solutions

We will get the solution to the prescribed problem.



* This process is mainly based on the recursion i.e. dividing the problems and calling itself to solve the sub problems .

3\_BASIC STEPS TO UNDERATAND RECURRSION:

|  |
| --- |
| 1. Step 1: Find the Base Case. There are two base cases for this problem. 2. Step 2: Find the Recursive Relation. Recursive relation divides the problem into subproblems. 3. Step 3: Combine the results from the Recursive function calls. |

KARATSUBA\_ALGORITHM

Consider 2 numbers 1234 ,9876;

Which can be written as

X=12\*100+34;

Y=98\*100+76; {NUMBER INTRGRATION}

N=4;(number of digits in the given digit);

X= 10^(N/2)\*12+34;-

Y=10^(N/2)\*98+76;

Now Consider:

a =12; SIMILARLY c = 98;

b=34; d = 76;

Convert in to equations:

X= a \* 10^(N/2) +b; ----eqn\_1

Y= c \* 10^(N/2) + d; -----eqn\_2

Multiply eqn\_1 and eqn\_2:

X \* Y = (a \* 10^(N/2) +b) \* (c \* 10^(N/2) + d);

= (a\*10^(N/2)) \* (b\*10^(N/2)) + (a\*10^(N/2)) \* (d) + (b) \*( c \* 10^(N/2)) +(bd);

* Equation can be divided in to 3 parts by picking common elements;

Part 1= (a \* b (10^(N));

Part 2 = 10^(N/2) (ad + bc) ;

Part 3 = bd;

Consider part 1;

a \* b (10 ^(N))

* When n digit number multiplied by an n digit number then we get an n digit out put.
* a is an n/2 digit number similarly b is an n/2 digit number then the out put is of n digit .
* We can reduce this stage by further dividing the numbers then the numbers get divided in to “n/4”.
* Further this way we can achieve the all digits in to atomic reduction and perform the multiplication.
* Thus Atomic Multiplication is achieved then
* All the subproblems of the problems are solved and merged .

Here it computes µ (n/2) multiplications.

Consider part 2;

Part 2 = 10^(N/2) (ad + bc) ;

* In this part we have both additions and the multiplications then we have to reduce the equations then both atomic addition and the atomic multiplications are possible here.
* From the reference of part 1 we can say that it computes two 2-digit multiplications then it generates 2\*n atomic multiplications then we may get equation as

“ µ (n/2) + µ (n/2) ”

* Considering equations then we may get some additions

>> Conditions for atomic additions :

For an n digit number the total atomic additions are said to be α(n/2) the here 2 n/2 digits were multiplied and added then the number of atomic additions is represented as

(1)α(n)

Consider part 3;

Part 3 = bd;

* IT is similar to part a where two n/2-digit numbers were multiplied the outcomes are as follows

* Here it computes µ (n) multiplications

Note :

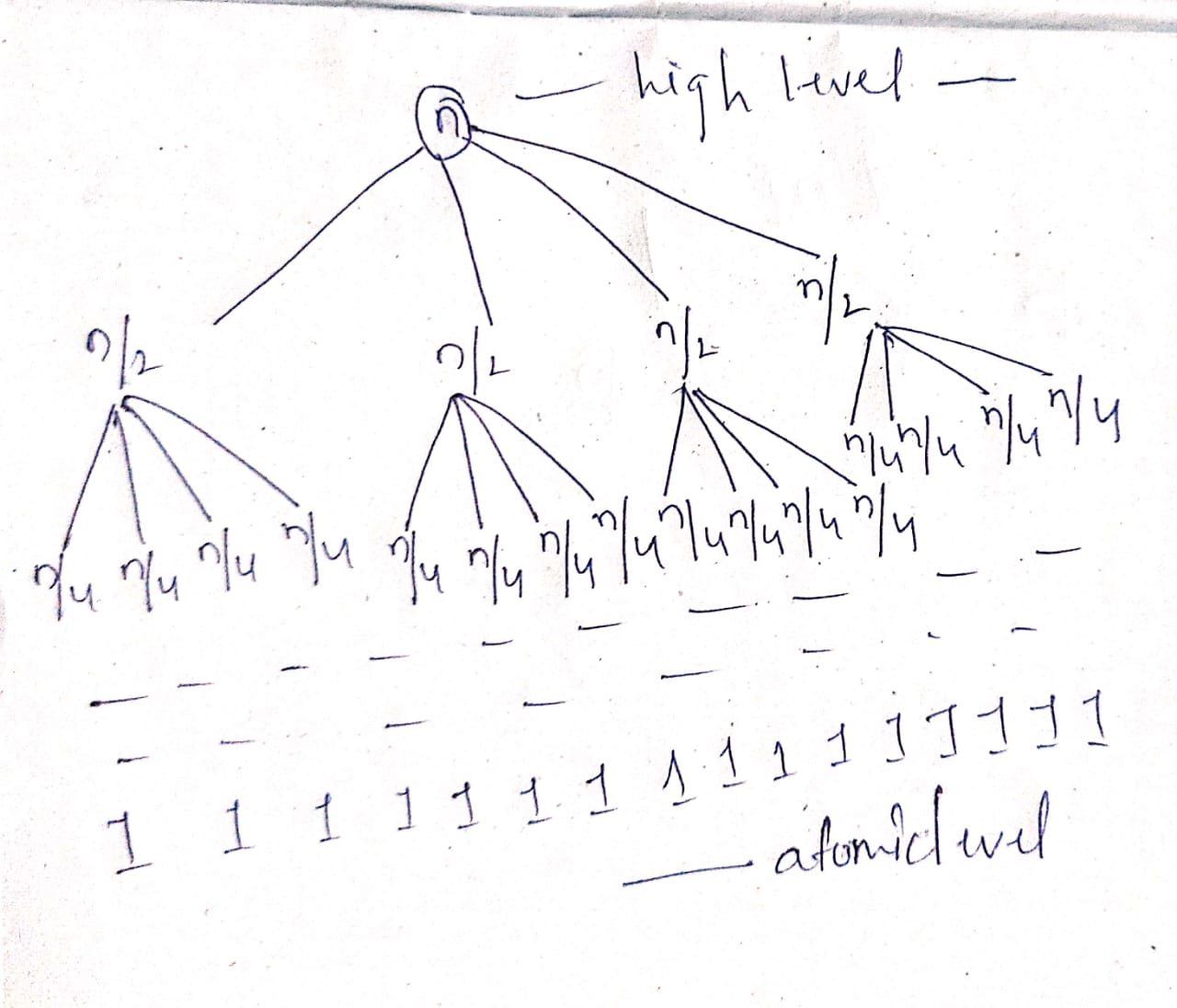
To reduce the additions between part 1 and part 2 we have to consider them as the both strings and we have to concat them then we have a chance to reduce the occurance of the

(1)α(n) by which we can reduce some computations policies .

NOW Consider the Recursion tree below:

Recurrence relation for the above computation process:

M(n)= 4m (n/2) + 2α(n).



* From the Above tree we can analyse the number of splits occurring at each level.
* And number of atomic additions and atomic Multiplications occurring at each level.

TOTAL NUMBER OF ATOMIC ADDITIONS

We can generate equations like:

At higher level :

1\* 2α(n)

N tends to be halved for each split then number of nodes obtained is 4

@1st split:

4\* 2α(n/2)

@2nd split :

16\* 2α(n/4)

For “ i” number of splits we may get equation as:

“(4^(i)\* 2α(n/2^(i))”

TOTAL NO. OF ATOMIC MULTIPLICATIONS

The splitting of the numbers determines the “log(x)” functions so the number of atomic multiplications carried out through out the process is determined by the following equation:

“ MUL= µ \* 4 (log (n))”

By solving the equations of multiplications and the addition we may get the following equation:

Final equation :

µ \* n^2 +2 αn(n-1)

From the above equation we can derive the time complexity an O(n^2) which is equal to the traditional way of doing multiplication .We have to reduce more.

From the analysis of the above equations we can say that the part-2 is taking more computations of both additions and multiplications. We can reduce it by some mathematical reductions

Part 2 = 10^(N/2) (ad + bc) ;

-> This can be reduced in to:

(a+b) \* (c+d) – ac - bd

Calculate the number of additions and multiplications

Multiplications:

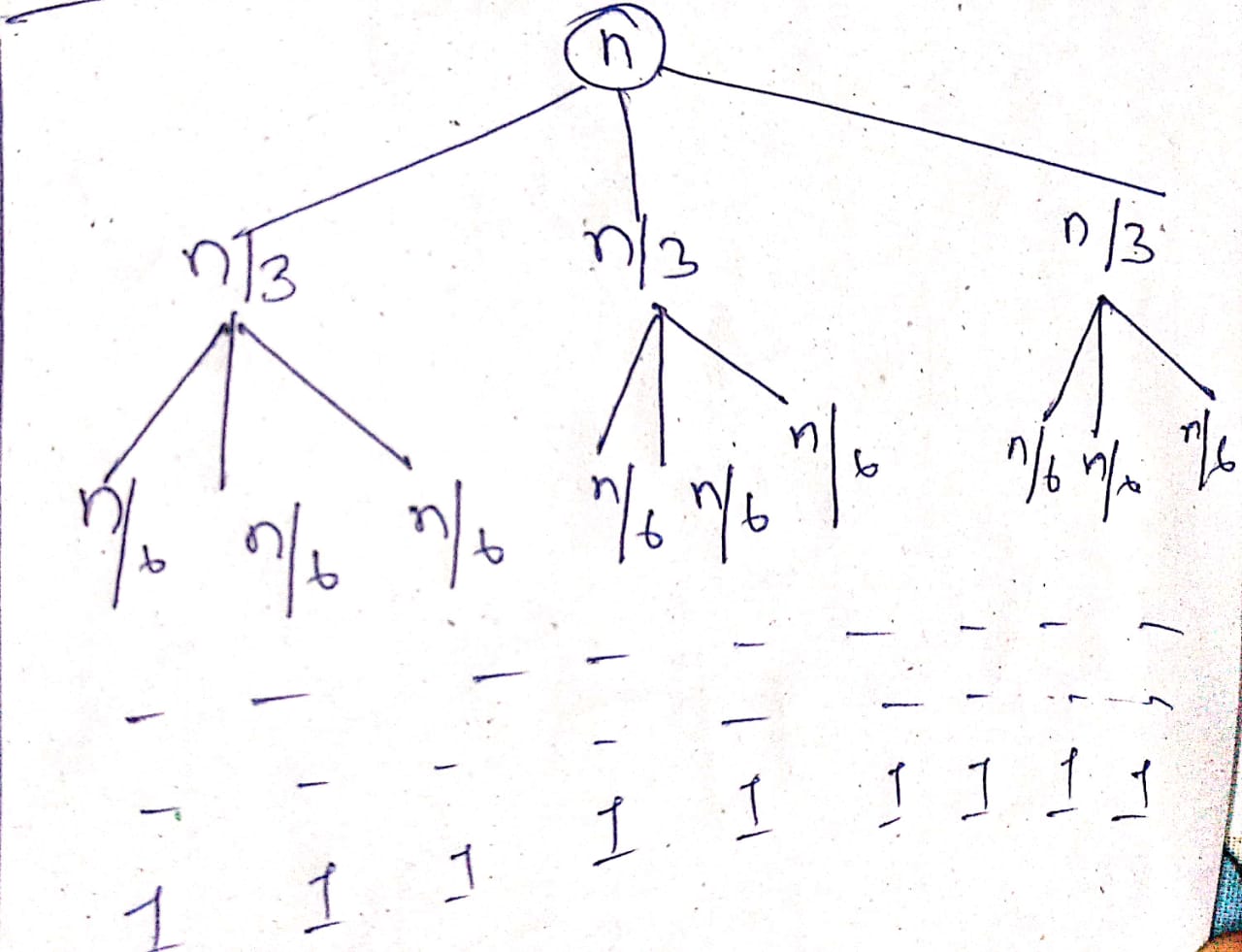
ac --1

bd --2

(a+b) (c+d) --3

In the above equations both 1 and 2 were calculated in the part 1 and part 3 respectively then compared to the previous calculations of part-2 we had reduced the one multiplication in every node and this may lead to Drastic effect on the algorithm .and 2 Additions were added.

Recursion Tree for the new calculations :



Recurrence relation for the above computation process:

“M(n)= 3m (n/2) + 4α(n)”

Total number of Atomic Additions:

“(3^(i)\* 4α(n/2^(i))”

Total number of Atomic Multiplications:

“ MUL= µ \* 3 (log (n))”

By solving the equations of multiplications and the addition we may get the following equation:

Final equation :

µ \* n^(log(3)) +8 αn(n\*log(3)\*-n/n)

By considering the equation we can say that Time Complexity has reduced i.e.

O(n^2) 🡪O(n ^ log(3))

Where log (3) =1.58.

Code demonstration for the Karatsuba Algorithm:

Implementing Karatsuba using JAVA:

import java.io.\*;

class HelloWorld {

static long karatsuba(long X, long Y) {

// Base Case

if (X < 10 && Y < 10)

return X \* Y;

// determine the size of X and Y

int size = Math.max(get\_size(X), get\_size(Y));

if(size < 10)

return X \* Y;

// Finding max lenght and gettting output

size = (size/2) + (size%2);

long multiplier = (long)Math.pow(10, size);

long b = X/multiplier;

long a = X - (b \* multiplier);

long d = Y / multiplier;

long c = Y - (d \* size);

long u = karatsuba(a, c);

long z = karatsuba(a + b, c + d);

long v = karatsuba(b, d);

return u + ((z - u - v) \* multiplier) + (v \* (long)(Math.pow(10, 2 \* size)));

}

static int get\_size(long value) {

int count = 0;

while (value > 0) {

count++;

value /= 10;

}

return count;

}

public static void main(String args[]) {

// two numbers X,Y

long start;

long end;

long x = 172864652;

long y = 74622755;

System.out.print("The final product is ");

start=System.currentTimeMillis();

long product = karatsuba(x, y);

System.out.println(product);

end=System.currentTimeMillis();

System.out.println("Time taken By Karatsuba Algorithm:"+(end-start)/10000);

}

}

Output:

The final product is 12899636574356260

Time taken By Karatsuba Algorithm:0.00098sec

Conclusion:

The number of multiplication and the total process time are used as analysis parameters. According to the study results, the bit length raises along with the number of multiplication owing to the processing of Karatsuba algorithm. The more the bit length increases, the more the total process time rises.

*Performance of Karatsuba on various bit Length:*

