eTHNUS™

Explore | Expand | Enrich

<Codemithra />™

TEST TIME ON EUCLID'S ALGORITHM

URL:https://forms.gle/2Qag3HhHTrpx5fna8

QR CODE:

KARATSUBA ALGORITHM

Karatsuba Algorithm

Introduction

The Karatsuba algorithm is a fast multiplication algorithm. It was discovered by Anatoly Karatsuba in 1960 and published in 1962.

A fast multiplication algorithm uses a divide and conquer approach to multiply two numbers.

The naive algorithm for multiplying 2 numbers has a running time of $O(n^2)$, where karatsuba algorithm has a runtime of

$$\Theta\left(n^{\log_2 3}\right) \approx \Theta\left(n^{1.585}\right)$$

Karatsuba Algorithm

Introduction

Being able to multiply numbers quickly is very important. Computer scientists often consider multiplication to be a constant time ~ O(1) operation which is reasonable for small numbers.

Whereas for larger numbers, the actual running times need to be factored in, which is $O(n^2)$

Karatsuba Algorithm

Naive Method

The naive method is to follow the elementary school multiplication method, i.e. to multiply each digit of the second number with every digit of the first number and then add all the multiplication results.

This algorithm takes $O(n^2)$ time.

Naive Method

Karatsuba Algorithm

```
public static int multiplication(int X, int Y) {
    // Convert numbers into string
    String x = Integer.toString(X);
    String y = Integer.toString(Y);
    int result = 0;
    // Looping over y
    for (int i = 0; i < y.length(); i++) {
      int carry = 0; // intermediate carry
      String inter_res = ""; // intermediate
result
      // Looping over x.
      for (int j = x.length() - 1; j >= 0; j--) {
        // intermediate multiplication of each
digit and addition of carry.
        int num =
Character.getNumericValue(y.charAt(i)) *
Character.getNumericValue(x.charAt(j)) + carry;
 if (num > 9 && j > 0) {
          inter_res = Integer.toString(num % 10)
+ inter_res;
```

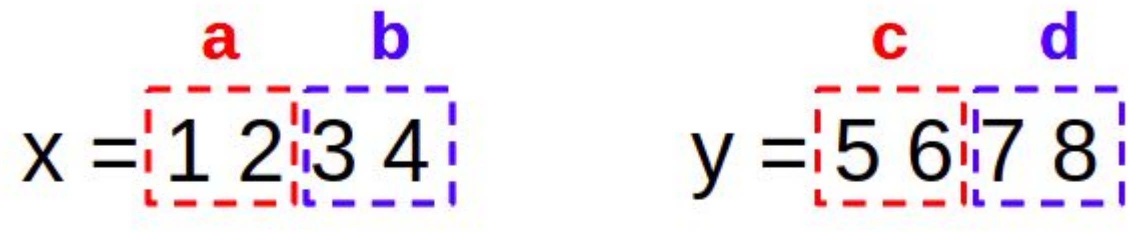
```
 carry = num / 10;
        }
// else the digit is append to the
intermediate result
        // And assign carry as zero
        else {
          inter_res =
Integer.toString(num) + inter_res;
          carry = 0;
        }
      }
      // Adding the intermediate
results
      result *= 10;
      result +=
Integer.parseInt(inter_res);
    }
     System.out.print("result: ");
    return result;
  }}
```

Karatsuba Algorithm

Karatsuba Algorithm

Let’s consider two 4-digit numbers x and y where x=1234 and y=5678.

First of all, we should divide the n-digit numbers into n/2-digit numbers as shown below.



a and c represent the first n/2 digits of x and y. Similarly, b and d represent the last n/2 digits of x and y

Karatsuba Algorithm

The Karatsuba algorithm involves 4 main steps

Step 1: Compute a.c = 12 x 56 = 672

Step 2: Compute b.d = 34 x 78 = 2652

Step 3: Compute (a+b)(c+d) = 46 x 134 = 6164

Step 4: Compute (3)-(2)-(1)=6164-2652-672 = 2840

Finally, multiply the output of step 1 by 10^n, the output of step 4 by $10^{n/2}$, and add them both with the output of step 2.

6720000 + 284000 + 2652 = 7006652

Karatsuba Algorithm

To multiply X = 1234 and Y = 2345 using the Karatsuba algorithm

	a	b			c	d
X =	12	34		Y =	23	45

Steps:

Step 1: Compute a.c = 12 x 23 = 276

Step 2: Compute b.d = 34 x 45 = 1530

Step 3: Compute (a+b)(c+d) = 46 x 68 = 3128

Step 4: Compute (3) - (2) - (1) = 3128 - 1530 - 276 = 1322

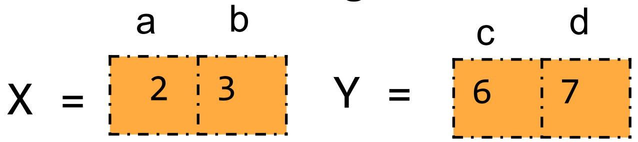
Finally, multiply the output of step 1 by 10000 (10^4), the output of step 4 by 100 (10^2), and add them both with the output of step 2.

Result: 2760000 + 132200 + 1530 = 2893730

The product of 1234 and 2345 using the Karatsuba algorithm is 2897730.

Karatsuba Algorithm

compute the product of 23 and 67 using the Karatsuba algorithm.



Step 1: Compute a.c = 2 x 6 = 12

Step 2: Compute b.d = 3 x 7 = 21

Step 3: Compute (a+b)(c+d) = (2+3)(6+7) = 5 x 13 = 65

Step 4: Compute (3) - (2) - (1) = 65 - 21 - 12 = 32

Finally, multiply the output of Step 1 by 10^n (where n is the number of digits), the output of Step 4 by 10^n/2, and add them both with the output of Step 2.

1200 + 320 + 21 = 1541

product of 23 and 67 using the Karatsuba algorithm is 1541.

Karatsuba Algorithm

Time Complexity

Assuming that we replace two of the multiplications with only one makes the program faster.

Karatsuba improves the multiplication process by replacing the initial complexity from quadratic to

$$\Theta(n^{\log_2 3}) \approx \Theta(n^{1.585})$$

Where n is the number of digits of the numbers multiplying.

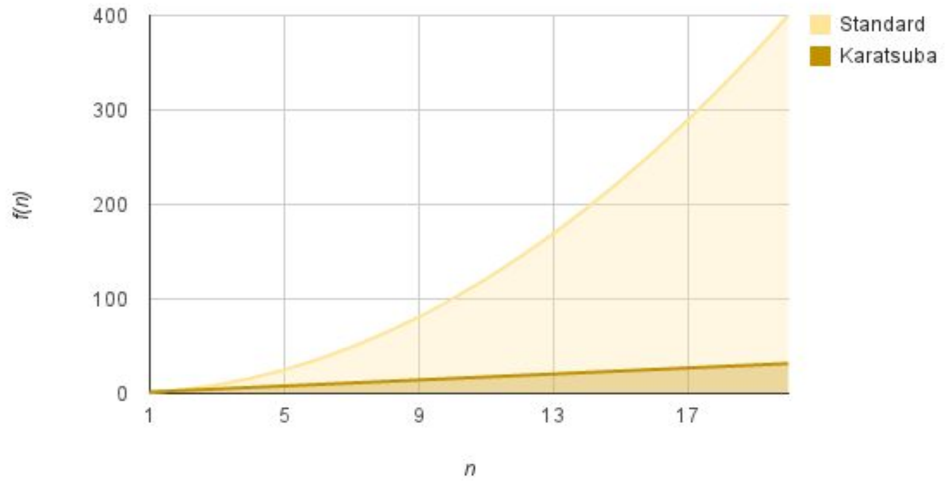
The Time Complexity of the algorithm can be represented as follows

$$T(n) = 3 * T(n/2) + O(n)$$

Karatsuba Algorithm

Time Complexity

$$T(n) = 3 * T(n/2) + O(n)$$



Karatsuba Algorithm

Implementation

Algorithm

1. Compute starting set (a*c)
2. Compute set after starting set may it be ending set (b*d)
3. Compute starting set with ending sets (a+b)(c+d)
4. Subtract values of Step 3 from Step 2 from Step 1
5. Add all the values with the following modifications:-
    1. Pad up 10^n to the number obtained from Step 1
    2. Step 2 value unchanged
    3. Pad up $10^{n/2}$ to the value obtained from Step 4.

Karatsuba Algorithm

Program

<u>Sample IO</u>

```
Karatsuba Multiplication Algorithm Test

Enter the first number: 1234
Enter the second number: 2345
Product: 2893730
```

Karatsuba Algorithm

```
import java.math.BigInteger;
import java.util.Scanner;
public class KaratsubaAlgorithm {
    public static BigInteger karatsuba(BigInteger
x, BigInteger y) {
        int n = Math.max(x.bitLength(),
y.bitLength());
        // Base case: if either x or y is small,
use standard multiplication
        if (n <= 2000) {
            return x.multiply(y);
        }
        // Split the numbers into two halves
        int half = (n + 32) / 64 * 32;  // round
up to the nearest multiple of 64 bits
        BigInteger mask =
BigInteger.ONE.shiftLeft(half).subtract(BigInteger
.ONE);
        BigInteger xLow = x.and(mask);
        BigInteger yLow = y.and(mask);
        BigInteger xHigh = x.shiftRight(half);
        BigInteger yHigh = y.shiftRight(half);
```

```
BigInteger z0 = karatsuba(xLow, yLow);
        BigInteger z1 = karatsuba(xLow.add(xHigh),
yLow.add(yHigh));
        BigInteger z2 = karatsuba(xHigh, yHigh);
        // Combine the results
        BigInteger result = z2.shiftLeft(2 *
half).add(z1.subtract(z2).subtract(z0).shiftLeft(h
alf)).add(z0);
        return result;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the first number:
");
        BigInteger x = scanner.nextBigInteger();
        System.out.print("Enter the second number:
");
        BigInteger y = scanner.nextBigInteger();
        BigInteger product = karatsuba(x, y);
        System.out.println("Product: " + product);
    }}
```

Interview questions

What is the Karatsuba algorithm?

The Karatsuba algorithm is a fast multiplication algorithm that allows multiplying large integers in a more efficient manner than the traditional multiplication algorithm. It reduces the number of recursive multiplications required by breaking down the numbers into smaller parts.

Interview questions

How does the Karatsuba algorithm work?

The Karatsuba algorithm works by splitting the input numbers into two halves and recursively calculating three products. These three products are combined using some arithmetic operations to obtain the final product.

Interview questions

What are the advantages of the Karatsuba algorithm over traditional multiplication?

The Karatsuba algorithm has a lower time complexity than traditional multiplication algorithms for large numbers. It reduces the number of multiplications required and, therefore, improves the overall efficiency of the multiplication operation.

Interview questions

Can you explain the recursive steps involved in the Karatsuba algorithm?

In the Karatsuba algorithm, the input numbers are split into two halves. Three recursive multiplications are performed: one for the lower halves, one for the sum of the halves, and one for the upper halves. These products are combined using arithmetic operations to obtain the final result.

/ethnuscodemithra Ethnus Codemithra /ethnus /code_mithra

<Codemithra />™

https://learn.codemithra.com

Explore | Expand | Enrich

codemithra@ethnus.com +91 7815 095 095 +91 9019 921 340