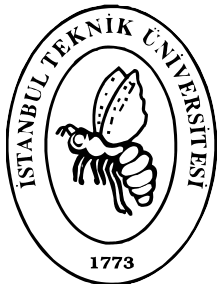
ISTANBUL TECHNICAL UNIVERSITY
COMPUTER ENGINEERING DEPARTMENT



Analysis of Algorithms 2

Homework 2
Report

Emre UYSAL
150160510

BINARY STRING MULTIPLICATION
WITH COMPARING
CLASSICAL METHOD AND KARATSUBA MULTIPLICATION
ALGORITHM

Explanation

Multiplication process for large numbers is an important problem in Computer Science. Given approach uses Divide and Conquer methodology. It divides problem into 4 sub problems and make recurrence with multiplying and adding them.

My Problem Formulation

In initial step after I got the binary strings from user, I checked lengths of the strings and made them equal by adding 0's to their heads.

In classical method, I created 4 function. Firstly, MakeMultiplication function runs over second string bit by bit and checks the current bit is 1 or 0. If current bit is 1 it calls MakeShifting function for shifting the first binary string according to bit position. After returning from shifting operation it sums the shifted numbers by using addBinary function. After these steps MakeMultiplication function returns a binary string as a result and BinaryStringToDecimal function gives decimal version of this string to user.

In Karatsuba Algorithm, firstly I implemented the given pattern for integers. But after that trying with 32 bit numbers I understand integer is not sufficient for large binary strings. So, I implemented the string version of it but for making operations on strings I implemented functions for; subtraction, addition, shifting, getting max length and converting decimal to binary. Subtraction function makes subtract operation between 2 binary strings with borrow. Shifting function makes shifting operation for given number. Getting max length function equals given strings lengths and returns the max length. Addition function add binary strings with carry. Conversion function converts the binary string to decimal number. All in all, I used all these function in multiply function to find solution. In multiply function I divided the problem into 4 sub problems as given in pseudocode. After that I recursively calculated multiplication operation given pattern from all recursive calls.

How Does My Algorithm Work ?

Classical Method

```
MakeMultiplication(string str1,string str2):
        string allsum = ""
        for 0->str2.length()
                int digit = str2[current]-'0'
                if digit = 1
                        returned = MakeShifting(str1,str2.size()-(current+1))
                        allsum = AddBinary(returned, allsum)
                else
                        continue
        return allsum
```

```
AddBinary(string a, string b)
        String result =""
        int forcarry = 0
        int i = a.size()-1
        int j = b.size()-1
        while(i >= 0 || j>=0 || s == 1)
                a[i]>=0 ?  a[i]-'0' : 0
                b[j]>=0 ?  b[j]-'0' : 0
                result = char(s%2+'0')+result
                s/=2
                j--;
                i--;
        return result

MakeShifting(string str,int stepnum)
        For 0->stepnum
                Str = str + '0'
        Return str
```

My time complexity on my pseudocode is O(n^2).

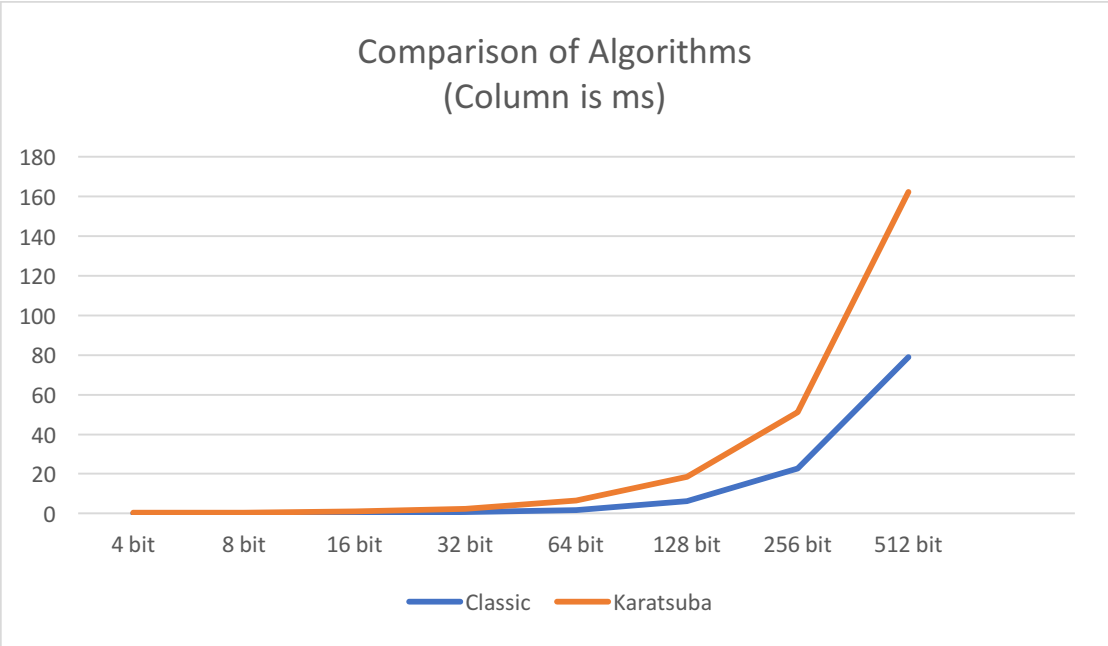
Karatsuba Algorithm

Pseudocode is given in homework document. For given pseudocode complexity is O(n^2). But in the recuded form I used:

2^(ceil(n/2))*XlYl + 2^(ceil(n/2))*[(Xl+Xr)(Yl+Yr)-XlYl-XrYr]

In this algorithm time complexity is O(n^1.59) because of recursive calls.

Analyzing and Explaining Results From the Graph



According to graph, I can say that Karatsuba Algorithm is slower than mine. According to implementations it must not be true, but I controlled the time results are growing according to algorithms complexities. The only thing that makes Karatsuba slower in my code is Karatsuba multiplication function returns string. You can say “What is that mean ? “. I returned string from function by making all operations in string type. This means, subtraction in string type, shifting in string type, adding in string type. So all in all, Karatsuba is a fast algorithm for string multiplications but not binary string multiplications. All the required functionalities for binary string multiplication makes Karatsuba slower.