Divide and Conquer | Set 4 (Karatsuba algorithm for fast multiplication)

Given two binary strings that represent value of two integers, find the product of two strings. For example, if the first bit string is "1100" and second bit string is "1010", output should be 120.

For simplicity, let the length of two strings be same and be n.

A **Naive Approach** is to follow the process we study in school. One by one take all bits of second number and multiply it with all bits of first number. Finally add all multiplications. This algorithm takes O(n^2) time.

```
    X = 101001 = 41
    Y = 101010 = 42
 -------------
      1010010
     101001
 + 101001
 -------------
   11010111010 = 1722
```

Using **Divide and Conquer**, we can multiply two integers in less time complexity. We divide the given numbers in two halves. Let the given numbers be X and Y.

For simplicity let us assume that n is even

```
X =  Xl*2^(n/2) + Xr    [Xl and Xr contain leftmost and rightmost n/2 bits of X]
Y =  Yl*2^(n/2) + Yr    [Yl and Yr contain leftmost and rightmost n/2 bits of Y]
```

The product XY can be written as following.

```
XY = (Xl*2^(n/2) + Xr)(Yl*2^(n/2) + Yr)
   = 2^n XlYl + 2^(n/2)(XlYr + XrYl) + XrYr
```

If we take a look at the above formula, there are four multiplications of size n/2, so we basically divided the problem of size n into for sub-problems of size n/2. But that doesn't help because solution of recurrence T(n) = 4T(n/2) + O(n) is O(n^2). The tricky part of this algorithm is to change the middle two terms to some other form so that only one extra multiplication would be sufficient. The following is tricky expression for middle two terms.

```
XlYr + XrYl = (Xl + Xr)(Yl + Yr) - XlYl- XrYr
```

So the final value of XY becomes

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

With above trick, the recurrence becomes T(n) = 3T(n/2) + O(n) and solution of this recurrence is $O(n^{1.59})$.

What if the lengths of input strings are different and are not even? To handle the different length case, we append 0's in the beginning. To handle odd length, we put *floor(n/2)* bits in left half and *ceil(n/2)* bits in right half. So the expression for XY changes to following.

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

The above algorithm is called Karatsuba algorithm and it can be used for any base.

Following is C++ implementation of above algorithm.

```
// C++ implementation of Karatsuba algorithm for bit string multiplication.
#include<iostream>
#include<stdio.h>

using namespace std;
```

```
// FOLLOWING TWO FUNCTIONS ARE COPIED FROM http://goo.gl/q0OhZ
// Helper method: given two unequal sized bit strings, converts them to
// same length by adding leading 0s in the smaller string. Returns the
// the new length
int makeEqualLength(string &str1, string &str2)
{
    int len1 = str1.size();
    int len2 = str2.size();
    if (len1 < len2)
    {
        for (int i = 0 ; i < len2 - len1 ; i++)
            str1 = '0' + str1;
        return len2;
    }
    else if (len1 > len2)
    {
        for (int i = 0 ; i < len1 - len2 ; i++)
            str2 = '0' + str2;
    }
    return len1; // If len1 >= len2
}

// The main function that adds two bit sequences and returns the addition
string addBitStrings( string first, string second )
{
    string result;  // To store the sum bits

    // make the lengths same before adding
    int length = makeEqualLength(first, second);
    int carry = 0;  // Initialize carry

    // Add all bits one by one
    for (int i = length-1 ; i >= 0 ; i--)
    {
        int firstBit = first.at(i) - '0';
        int secondBit = second.at(i) - '0';

        // boolean expression for sum of 3 bits
        int sum = (firstBit ^ secondBit ^ carry)+'0';

        result = (char)sum + result;

        // boolean expression for 3-bit addition
        carry = (firstBit&secondBit) | (secondBit&carry) | (firstBit&carry);
    }

    // if overflow, then add a leading 1
    if (carry)  result = '1' + result;

    return result;
}
```

```
// A utility function to multiply single bits of strings a and b
int multiplyiSingleBit(string a, string b)
{  return (a[0] - '0')*(b[0] - '0');  }

// The main function that multiplies two bit strings X and Y and returns
// result as long integer
long int multiply(string X, string Y)
{
    // Find the maximum of lengths of x and Y and make length
    // of smaller string same as that of larger string
    int n = makeEqualLength(X, Y);

    // Base cases
    if (n == 0) return 0;
    if (n == 1) return multiplyiSingleBit(X, Y);

    int fh = n/2;   // First half of string, floor(n/2)
    int sh = (n-fh); // Second half of string, ceil(n/2)

    // Find the first half and second half of first string.
    // Refer http://goo.gl/lLmgn for substr method
    string Xl = X.substr(0, fh);
    string Xr = X.substr(fh, sh);

    // Find the first half and second half of second string
    string Yl = Y.substr(0, fh);
    string Yr = Y.substr(fh, sh);

    // Recursively calculate the three products of inputs of size n/2
    long int P1 = multiply(Xl, Yl);
    long int P2 = multiply(Xr, Yr);
    long int P3 = multiply(addBitStrings(Xl, Xr), addBitStrings(Yl, Yr));

    // Combine the three products to get the final result.
    return P1*(1<<(2*sh)) + (P3 - P1 - P2)*(1<<sh) + P2;
}

// Driver program to test aboev functions
int main()
{
    printf ("%ld\n", multiply("1100", "1010"));
    printf ("%ld\n", multiply("110", "1010"));
    printf ("%ld\n", multiply("11", "1010"));
    printf ("%ld\n", multiply("1", "1010"));
    printf ("%ld\n", multiply("0", "1010"));
    printf ("%ld\n", multiply("111", "111"));
    printf ("%ld\n", multiply("11", "11"));
}
```

Output:

```
120
60
```

```
30
10
0
49
9
```

Time Complexity: Time complexity of the above solution is $O(n^{1.59})$.

Time complexity of multiplication can be further improved using another Divide and Conquer algorithm, fast Fourier transform. We will soon be discussing fast Fourier transform as a separate post.

Exercise
The above program returns a long int value and will not work for big strings. Extend the above program to return a string instead of a long int value.

References:

Wikipedia page for Karatsuba algorithm
Algorithms 1st Edition by Sanjoy Dasgupta, Christos Papadimitriou and Umesh Vazirani
http://courses.csail.mit.edu/6.006/spring11/exams/notes3-karatsuba
http://www.cc.gatech.edu/~ninamf/Algos11/lectures/lect0131.pdf

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Related Topics:

- Subtract two numbers without using arithmetic operators
- Calculate square of a number without using *, / and pow()
- Check if binary representation of a number is palindrome
- Swap two nibbles in a byte
- How to turn off a particular bit in a number?
- Check if a number is multiple of 9 using bitwise operators
- How to swap two numbers without using a temporary variable?
- Find position of the only set bit

Tags: Divide and Conquer

Writing code in comment? Please use **ideone.com** and share the link here.

prashant saxena · 3 months ago

This is a solution for large Numbers. Try out any big number.

This is a solution for large Numbers. Try out any big number.

```
#include <string>
#include <math.h>
#include <cassert>
#include <sstream>

int adjustLength(std::string& s1, std::string& s2)

{

int len1 = s1.size();

int len2 = s2.size();

std::string s_adjusted;

std::string s_orig;

if (len1 != len2)
```

see more

^ | v • Reply • Share ›

prashant saxena ➦ prashant saxena • 3 months ago

Ok, the format sucks.

^ | v • Reply • Share ›

Diptesh Patel • 5 months ago

When n is odd can't we just add one more bit ahead of strings, that does not changes the values and reduces the difficulty.
Your approach of floor and n-floor is also simpler. but just curious to know will my solution work?

^ | v • Reply • Share ›

Arjun K • 8 months ago

Check this for analysis of algorithm which one is best. http://www.msccomputerscience....

^ | v • Reply • Share ›

n@733d • 9 months ago

PLEASE explain how we got this X = Xl*2n/2 + Xr

^ | v • Reply • Share ›

rohit ➦ n@733d • 4 months ago

shift left by n/2 bits the first half bits of X (i.e. Xl) and then add other half.. so 1101 (11 = xl , 01 = xr) (2^2xl = 1100 + 0001 (xr) = x(1101)

^ | v • Reply • Share ›

Reply • Share ›

blaze • a year ago

how did the recurrence expression comes
t(n)=4t(n/2)+o(n)

Reply • Share ›

RK → blaze • a year ago

The problem of size(n) is reduced to 4 problems of size(n/2) and it would take extra O(n) time to multiply with 2^n, 2^(n/2)

Reply • Share ›

Amit • a year ago

For normal high school multiplication, the complexity is O(n^2) right?

1 • Reply • Share ›

RK → Amit • a year ago

yes

Reply • Share ›

Spanky • a year ago

Mistake : O(n^159) -> O(n^158) ~= log2(3)

Reply • Share ›

anvesh • a year ago

can any one write code for complex numbers multiplication using FFT

Reply • Share ›

frenzydude • a year ago

This can be done even faster in O(nlogn) using FFT.

Reply • Share ›

sudhanshu • a year ago

can any1 pls tell me wht are floor n ceil values ??

Reply • Share ›

Himanshu Dagar → sudhanshu • a year ago

floor means greatest integer less than or equal to that no
and ceil is least integer greater than or equal to that no
e.g floor(2.6)=2;
ceil(2.6)=3;
defined in math.h

Reply • Share ›

Sanjeet • a year ago

/******Sanjeet from NIT allahabad ********/ you can simply adjust the value of i & j to get the multiplication of two strings .

```
#include<stdio.h>
#include<math.h>
int convertbin(char a[]);
int main()
{
char string1[4]="0111";
char string2[4]="0111";
int DecimalNum1=0;
int DecimalNum2=0;
int Product=0;
DecimalNum1=convertbin(string1);
DecimalNum2=convertbin(string2);
Product=DecimalNum1*DecimalNum2;
printf("Product of two string =%d\n",Product);
return 0;
}
```

see more

• Reply • Share ›

Shabaz Ahmed Sanjeet • a year ago

The point here is to do the multiplication in less than O(n^2)..!

1 • Reply • Share ›

Raunak Lakhwani • a year ago

```
public class Main {

static String first = "1", second = "00110";

/**

* @param args

*/

public static void main(String[] args) {

//int n = makeEqualString(first, second);

System.out.println(multiplication(first, second));

}
```

public static int makeEqualString(StringBuffer a, StringBuffer b) {

if (a.length() > b.length()) {

see more

^ | v • Reply • Share ›

Trilok • 2 years ago

Typo mistake ..
XY = 2n XlYl + 2ceil(n/2) * [(Xl + Xr)(Yl + Yr) - XlYl - XrYr] + XrYr

^ | v • Reply • Share ›

komal • 2 years ago

```
#include
#include
#include

using namespace std;

int pow(int x,int n)
{
if(x==0) return 0;
if(n==0) return 1;
if(n==1) return (x);
return (x*pow(x,n-1));
}
void print(string s)
{
for(int i=0;i<=s.size();i++)
cout<<s[i]<<" ";
cout<<endl;
```

see more

^ | v • Reply • Share ›

akshat gupta → komal • 2 years ago

1.you are entering a number in binary representation(serves no purpose here),
2.converting it to decimal,
3.printing the multiplication...

PROBLEM:
question asks you to find an algorithm for "fast multiplication of 2 numbers"..

^ | v • Reply • Share ›

akshat gupta • 2 years ago

```
int karatsuba(int n1,int n2)
{
int l1,l2,l;
int i,mask1=1;
int a,b,c,d;
int res1,res2,res3;

l1=dig(n1);//returns number of digits
l2=dig(n2);

if(n1==0 || n2==0)
return(0);

l=min(l1,l2);

if(l>1)
{
for(i=1;i<=l/2;i++)
mask1=mask1*10;
```

see more

• Reply • Share ›

akshat gupta • 2 years ago

C IMPLEMENTATION:

```
int karatsuba(int n1,int n2)
{
int l1,l2,l;
int i,mask1=1;
int a,b,c,d;
int res1,res2,res3;
l1=dig(n1);//returns number of digits
l2=dig(n2);
if(n1==0 || n2==0)
return(0);

l=min(l1,l2);

if(l>1)
{
for(i=1;i<=l/2;i++)
mask1=mask1*10;
```

see more

• Reply • Share ›

Sandeep Jain • 2 years ago

Thanks for pointing this out. We have corrected it.

• Reply • Share ›

Anwit Roy • 2 years ago

There is a typo: it should be XlYr + XrYl = (Xl + Xr)(Yl + Yr) - XlYl- XrYr.

1 • Reply • Share ›

Popular Posts

- Follow @GeeksforGeeks

- ## Recent Comments

  - Nikhil kumar

    public class...

    Print missing elements that lie in range 0 – 99 · 5 minutes ago

  - Ashish Aggarwal

    Try Data Structures and Algorithms Made Easy -...

    Algorithms · 27 minutes ago

  - Vlad

    Thanks. Very interesting lectures.

    Expected Number of Trials until Success · 1 hour ago

  - cfh

    My implementation which prints the index of the...

    Longest Even Length Substring such that Sum of First and Second Half is same · 1 hour ago

  - Gaurav pruthi

    forgot to see that part ;)

    Bloomberg Interview | Set 1 (Phone Interview) · 2 hours ago

  - saeid aslami

    thanks

    Greedy Algorithms | Set 7 (Dijkstra's shortest path algorithm) · 2 hours ago