Karatsuba Algorithm

高子腾

151220030

GZT@outlook.com

2017 年 3 月 29 日

大整数乘法

算法主要有普通长乘法，分治算法和 FFT 算法

长乘法需要 $O(n^2)$，比较慢

FFT 算法复杂，较少使用

分治算法: Karatsuba Algorithm, $O(n^{lg_2 3}) = O(n^{1.58})$, 涉及字符操作，实际耗时多

(不过最流行的还是长乘法)

Figure: Anatoly Karatsuba

Karatsuba Algorithm

基于分治的算法

朴素的分治法

$$2341 \times 1234$$

朴素的分治法

23*100+41

$\times$

12*100+34

$O(n^2)$?

T(n)=4T(n/2)+O(n)

Optimal? No!

Karatsuba Algorithm

$$a = a_1 * 10^d + a_2 \tag{1}$$

$$b = b_1 * 10^d + b_2 \tag{2}$$

$$ab = a_1 b_1 * 10^{2d} + (a_1 b_2 + a_2 b_1) * 10^d + a_2 b_2 \tag{3}$$

$$\ldots \tag{4}$$

$$a_1 b_2 + a_2 b_1 = (a_1 + a_2)(b_1 + b_2) - a_1 b_1 - a_2 b_2 \tag{5}$$

Note that multiplications here are recursive procedures!

Karatsuba Algorithm

$PROCEDURE\ MUL(a, b):$

$split\ a, b\ into\ (a_1, a_2, b_1, b_2)\ by\ \ divider\ D\ //O(n)\ process$

$as\ string$

$u \leftarrow MUL(a_1, b_1)\ //T(n/2)\ why?$

$v \leftarrow MUL(a_2, b_2)\ //T(n/2)\ why?$

$w \leftarrow MUL(a_1 + a_2, b_1 + b_2)\ //T(n/2)\ (why?) + O(n)$

$return\ uD^2 + (w - u - v)D + v\ //O(n)$

How to pick D?

Karatsuba Algorithm

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

According to Master Theorem,

$$T(n) = O(n^{lg_2 3})$$

```python
import math
def pow2lg(x):
    return int(2**(math.floor(math.log(x-0.01)/math.log(2))))
def fl2(x):
    return 10**(pow2lg(len(str(x))))
def karatsuba(x,y):
    if x < 10 and y < 10:
        return x*y
    else:
        if x > y:
            divider = fl2(x)
        else:
            divider = fl2(y)
    a1 = x // divider
    a2 = x % divider
    b1 = y // divider
    b2 = y % divider
    u = karatsuba(a1,b1)
    v = karatsuba(a2,b2)
    w = karatsuba(a1+a2, b1+b2)
    return (u*pow(divider,2) + v + ((w-v-u)*divider))
print karatsuba(666, 999)==666*999
```

The End