Polynomial Arithmetic

Lecture 1 **(out of 2)**

Apr. 12, 2005

Plan:

1. Multiplication of large integers

    1.1 Karatsuba Algorithm

    1.2 Toom-Cook Algorithm

2. Polynomial Evaluation

■ Arithmetic with large integers

What is a large integer?

$$\mathbf{2^{10}, \quad 2^{16}, \quad 2^{32}, \quad 2^{64}}$$

How do we represent a large integer?

by means of an array or a linked list

How do we measure complexity of arithmetic operation?

by the input size - the number of digits

Our discussion is based on integers in base 10.

What is the time complexity of adding (subtracting) two large integers?

linear in the input size.

So adding two *n*-digits integers takes $O(n)$

What is the time complexity of multiplication a large integer *x* by a base, $x * 10^p$?

■ Multiplication of large integers

The brute force approach ("grammar school" method)

```
    1 2 3
      4 5
   ------
    6 1 5
  4 9 2
---------
  5 5 3 5
```

We say that multiplication of two n-digits integers has time complexity at worst $O(n^2)$.

We develop an algorithm that has better asymptotic complexity. The idea is based on divide-and-conquer technique.

Consider the above integers and split each of them in two parts

```
123 = 12 * 10 + 3

 45 =  4 * 10 + 5
```

and then multiply them:

$$123*45 = (12*10 + 3)(4*10 + 5) = 12 * 4 * 10^2 + (12 * 5 + 4 * 3) * 10 + 3 * 5$$

In general, the integer which has n digits can be represented as

$$\text{num} = x * 10^m + y$$

where

$$m = \text{floor}\left(\frac{n}{2}\right)$$
$$x = \text{ceiling}\left(\frac{n}{2}\right)$$
$$y = \text{floor}\left(\frac{n}{2}\right)$$

Example,

$$154517766 = 15451 * 10^4 + 7766$$

Consider two *n*-digits numbers

$$\text{num}_1 = x_1 * 10^p + x_0$$

$$\text{num}_2 = y_1 * 10^p + y_0$$

Their product is

$$\text{num}_1 * \text{num}_2 = x_1 * y_1 * 10^{2p} + (x_1 * y_0 + x_0 * y_1) * 10^p + x_0 * y_0$$

Just looking at this general formula you can say that just instead of one multiplication we have 4.

Where is the advantage?

numbers x_1, x_0 and y_1, y_0 have twice less digits.

The worst-case complexity

Let $T(n)$ denote the number of digit multiplications needed to multiply two *n*-digits numbers.

The recurrence (since the algorithm does 4 multiplications on each step)

$$T(n) = 4\,T(\tfrac{n}{2}), \quad \text{T(c)} = 1$$

Note, we ignore multiplications by a base!!!

Its solution is given by (can you prove this?)

$$T(n) = 4^{\log n} = n^2$$

The algorithm is still quadratic!

■ The Karatsuba Algorithm

1962, Anatolii Karatsuba, Russia.

$$\text{num}_1 * \text{num}_2 = x_1 * y_1 * 10^{2p} + (x_1 * y_0 + x_0 * y_1) * 10^p + x_0 * y_0$$

The goal is to decrease the number of multiplications from 4 to 3.

We can do this by observing that

$$(x_1 + x_0) * (y_1 + y_0) = x_1 * y_1 + x_0 * y_0 + (x_1 * y_0 + x_0 * y_1)$$

It follows that

$$\text{num}_1 * \text{num}_2 = x_1 * y_1 * 10^{2p} + \big((x_1 + x_0) * (y_1 + y_0) - x_1 * y_1 - x_0 * y_0\big) * 10^p + x_0 * y_0$$

and it is only 3 multiplications (see it ?).

The total number of multiplications is given by (we ignore multiplications by a base)

$$T(n) = 3\,T(\tfrac{n}{2}), \qquad \text{T(c)} = 1$$

Its solution is

$$T(n) = 3^{\log n} = n^{\log 3} = n^{1.58...}$$

■ Toom-Cook 3-Way Multiplication

1963, A. L. Toom, Russia.

1966, Cook, Harvard, Ph.D Thesis

The key idea of the algorithm is to divide a large integer into 3 parts (rather than 2) of size approximately $n/3$ and then multiply those parts.

For example,

$$\mathbf{154517766 = 154 * 10^6 + 517 * 10^3 + 766}$$

In general case,

$$\text{num}_1 = x_2\, 10^{2\,p} + x_1 * 10^p + x_0$$

$$\text{num}_2 = y_2\, 10^{2\,p} + y_1 * 10^p + y_0$$

Multiplying them out, we obtain

$$\text{num}_1 * \text{num}_2 =$$

$$x_2 * y_2 * 10^{4\,p} + (x_2 * y_1 + y_2 * x_1) * 10^{3\,p} + (x_2 * y_0 + x_1 * y_1 + y_2 * x_0) * 10^{2\,p} +$$

$$(x_1 * y_0 + y_1 * x_0) * 10^p + x_0 * y_0$$

Here is the equation of for the total number of multiplications

$$T(n) = 9\,T\left(\frac{n}{3}\right), \qquad T(c) = 1$$

and the solution

$$T(\,n) = 9^{\,\log_3 n} = n^2$$

Let us reduce the number of multiplications by one

$$T(n) = 8\,T\left(\frac{n}{3}\right)$$

$$T(n) = 8^{\log_3 n} = n^{\log_3 8} = n^{1.89\ldots}$$

No advantage. This does not improve the previous algorithm, that runs at $O(n^{1.58\ldots})$
How many multiplication should we eliminate?

Let us consider that equation in a general form, where parameter $p > 0$ is arbitrary

$$T(n) = p\,T\left(\frac{n}{3}\right)$$

$$T(n) = p^{\log_3 n} = n^{\log_3 p}$$

Therefore, the new algoritnm will be faster than $O(n^{1.58})$ if we reduce the number of multiplications to five

$$T(n) = 5^{\log_3 n} = n^{\log_3 5} = n^{1.47\ldots}$$

This is an improvement over Karatsuba.

Is it possible to reduce a number of multiplications to 5?

$$x_0 * y_0 = Z_0$$

$$12\,(x_1 * y_0 + x_0 * y_1) = 8\,Z_1 - Z_2 - 8\,Z_3 + Z_4$$

$$24\,(x_2 * y_0 + x_1 * y_1 + x_0 * y_2) = -30\,Z_0 + 16\,Z_1 - Z_2 + 16\,Z_3 - Z_4$$

$$12\,(x_2 * y_1 + x_1 * y_2) = -2\,Z_1 + Z_2 + 2\,Z_3 - Z_4$$

$$24\,x_2 * y_2 = 6\,Z_0 - 4\,Z_1 + Z_2 - 4\,Z_3 + Z_4$$

where

$$Z_0 = x_0\,y_0$$

$$Z_1 = (x_0 + x_1 + x_2)\,(y_0 + y_1 + y_2)$$

$$Z_2 = (x_0 + 2\,x_1 + 4\,x_2)\,(y_0 + 2\,y_1 + 4\,y_2)$$

$$Z_3 = (x_0 - x_1 + x_2)\,(y_0 - y_1 + y_2)$$

$$Z_4 = (x_0 - 2\,x_1 + 4\,x_2)\,(y_0 - 2\,y_1 + 4\,y_2)$$

■ Further Generalization

It is possible to develop a **faster** algorithm by increasing the number of splits.

Let us consider a 4-way splitting

$$154517766 = 154 * 10^6 + 51 * 10^4 + 77 * 10^2 + 66$$

How many multiplications should we have on each step so this algorithm will outperform the 3-way splitting?

$$T(n) = p\,T\left(\frac{n}{4}\right)$$

$$T(n) = p^{\log_4 n} = n^{\log_4 p}$$

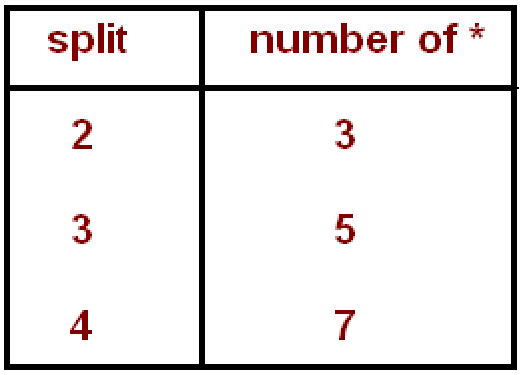
We find parameter p from

$$\log_4 p < \log_3 5$$

which yields

$$p = 7$$

The following table demonstrates a relationship between splits and the number of multiplications:



split	number of *
2	3
3	5
4	7

Intuitively we see that the $\boldsymbol{k}$-way split requires $\mathbf{2\,k - 1}$ multiplications.

This means that instead of k^2 multiplications we do only $2\,k - 1$.

The recurrence equation for the total number of multiplication is given by

$$T(n) = (2\,k - 1)\,T\left(\frac{n}{k}\right) + O(n)$$

and its solution is

$$T(n) = (2\,k - 1)^{\log_k n} = n^{\log_k (2\,k-1)}$$

Here is the sequence of the *k*-way splits when *k* runs from 2 to 10:

$$n^{1.58},\ n^{1.46},\ n^{1.40},\ n^{1.36},\ n^{1.33},\ n^{1.31},\ n^{1.30},\ n^{1.28},\ n^{1.27} \ldots$$

We can prove that asymptotically multiplication of two *n*-digits numbers requires $O(n^{1+\epsilon})$ multiplications, where $\epsilon \to 0$.

Note, we will NEVER get a linear performance (prove this!)

Is it always possible to find such $2\,k - 1$ multiplications?

Consider two polynomials of $k - 1$ degree

$$\text{polyn}_1 = a_{k-1}\,x^{k-1} + a_{k-2} * x^{k-2} + \ldots + a_1 * x + a_0$$

$$\text{polyn}_2 = b_{k-1}\,x^{k-1} + b_{k-2} * x^{k-2} + \ldots + b_1 * x + b_0$$

when we multiply them we get a polynomial of $2\,k - 2$ degree

$$\text{polyn}_1 * \text{polyn}_2 = a_{k-1}\,b_{k-1} * x^{2\,k-2} + \ldots + (a_1\,b_0 + b_1\,a_0) * x + a_0\,b_0$$

The above polynomial has exactly $2\,k - 1$ coefficients, therefore it's uniquely defined by $2\,k - 1$ values.

■ Polynomial Evaluation

Let us consider how to evaluate a polynomial at a given point.

For example,

$$p(x) = x^4 + 2\,x^3 - 5\,x^2 + x + 1$$

$$p(4) = ?$$

Direct approach takes many multiplications... Moreover, it requires recomputations.

Exercise.

One of sub-problems is the *exponentiation problem*.

Develop a divide-and-conquer algorithm for computing x^m.

A better idea that avoids recomputations is known as a Horner rule

$$p(x) = x * (x * (x * (x + 2) - 5) + 1) + 1$$

Therefore, any polynomial of order n requires $n - 1$ multiplications.

The Inverse Problem

Given n points (x_k, y_k), find a unique polynomial of degree $n - 1$.

This problem is known as a polynomial interpolation.

Example, find a polynomial that goes through (1, 2) and (3, 4)

$$\frac{x-1}{1-3} = \frac{y-2}{2-4}$$

$$p(x) = y = x + 1$$

The general solution is given by Lagrange's interpolation formula

$$p(x) = \sum_{j=1}^{n} y_j \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}$$

where the set of points (x_k, y_k) is given.

Observe that Lagrange's formula does not give immediately a standard form polynomial representation, but rather a sum of products of polynomials.

Exercise.

Develop a quadratic algorithm that converts Lagrange's formula to a coefficient representation.

Polynomial Multiplication

Two polynomials of degree n can be multiplied using $O(n^{1.58})$ multiplications.

However, there is a faster algorithm that can do this in $O(n * \log n * \log\log n)$.

The approach is based on the fact that a polynomial of degree n is uniquely specified by its value on $n+1$ points.

High level idea

1. compute two polynomials in $n+1$ points

2. multiply those values

3. restore a new polynomial