Binary splitting method

Xavier Gourdon and Pascal Sebah
January 11, 2001[1]

Most classical series formulaes to compute constants (like the exponential series for e, arctan formulaes for π, Chudnovsky or Ramanujan formulaes for π, ...) have a time cost of $O(n^2)$ to compute n digits of the series using a classical approach. Using FFT based multiplication, the binary splitting approach (foreshadowed in [2], [3] ; see also [1][p. 329]) usually permits to evaluate such series in time $O(n\ \log(n)^3)$ (or $O(n\ \log(n)^2)$ for e), which is much better.

1 Introduction

1.1 The example of the factorial

Suppose you would like to compute the digits of the factorial $n!$. A basic approach would consist in computing successively the following values

$$x_1 = 1, \quad x_2 = 2x_1, \quad x_3 = 3x_2, \quad \ldots, \quad x_n = nx_{n-1}.$$

At iteration number k, the value x_k contains $O(k\log(k))$ digits, thus the computation of $x_{k+1} = kx_k$ has cost $O(k\log(k))$. Finally, the total cost with this basic approach is $O(2\log(2) + \cdots + n\ \log(n)) = O(n^2\log(n))$.

A better approach is the *binary splitting* : it just consists in recursively cutting the product of m consecutive integers in half. It leads to better results when products on large integers are performed with a fast method.

More precisely, the computation of $p(a, b)$, where

$$p(a, b) \equiv (a + 1)(a + 2)\cdots(b - 1)b = \frac{b!}{a!},$$

is done by performing the product

$$p(a, b) = p\left(a, \frac{a+b}{2}\right) p\left(\frac{a+b}{2}, b\right),$$

where the two terms in the product are computed recursively in the same way. This permits to compute $n! = p(0, n)$.

We now concentrate on the estimation of the timing cost $C(a, b)$ of this process. We denote by $M(d)$ the cost of multiplying two integers of size d. The terms $p(a,(a + b)/2)$ and $p((a + b)/2, b)$ have size $O((b - a)\log(b))$, thus

$$C(a, b) = O(M((b-a)\log(b)))+C(a,(a+b)/2)+C((a+b)/2, b) \leq O(M((b-a)\log(b)))+2C((a+b)/2, b).$$

[1]This pages are from *//xavier.gourdon.free.fr/Constants/constants.html*

An easy induction entails

$$C(a,b) = O(\log(b-a)M((b-a)\log(b)))$$

(the superlinearity of $M(d)$ should be used : $2M(d/2) \leq M(d)$).

Finally, we have proved that computing $n!$ can be done in time $C(0,n) = O(\log(n)M(n\,\log(n))$. This is always better than the classical approach when products on large integers are done with a fast method. When FFT is used for large multiplication (see *FFT based multiplication of large numbers*), it leads to the bound

$$C(0,n) = O(n\,\log(n)^3).$$

1.2 Generalization

The example of the factorial can be generalized for many series. Instead of presenting a very general and complicated approach, the next sections are dedicated to illustrate the techniques on classical series.

2 Binary splitting on the exponential series

We want to compute the first n decimal digits of the series

$$e = \sum_{k\geq 0} \frac{1}{k!}. \qquad (*)$$

We should first stop the summation as soon as $k! > 10^n$, which occurs for $k = K \simeq d/\log(d)$. We make use of the notations

$$Q(a,b) = (a+1)(a+2)\cdots b,$$

$$P(a,b) = b(b-1)\cdots(a+2) + b(b-1)\cdots(a+3) + \cdots + (b-1)b + b + 1.$$

$P(a,b)$ and $Q(a,b)$ are integers which satisfy

$$\frac{P(a,b)}{Q(a,b)} = \frac{1}{a+1} + \frac{1}{(a+1)(a+2)} + \cdots + \frac{1}{(a+1)(a+2)\cdots b}.$$

In particular, $1 + P(0,K)/Q(0,K)$ are the first K terms of the series (*).

To compute $P(a,b)$ and $Q(a,b)$ by the binary splitting method, we proceed as follows. We set m to the half of $a+b$,

$$m = \left[\frac{a+b}{2}\right] \quad (integerpart).$$

Then we have

$$P(a,b) = P(a,m)Q(m,b) + P(m,b), \qquad Q(a,b) = Q(a,m)Q(m,b).$$

These operations are performed recursively.

The cost of the process to compute $P(0,K)$ and $Q(0,K)$ is easily proved to be $O(\log(K)M(K\log(K)))$. A final division of $P(0,K)$ by $Q(0,K)$ is then required to compute the first K terms of the series (its cost is $O(M(n))$, see *Inverse and n-th roots of large numbers*). Since $K \simeq n/\log(n)$, we have finally obtained a process to compute e in time $O(\log(n)M(n))$. An FFT based multiplication satisfies $M(n) = O(n\,\log(n))$ and leads to a total cost of $O(n\,\log(n)^2)$ to compute n digits of e.

An easy code illustrating the binary splitting method to compute e can be found in *Easy programs for constants computation.*

Practical implementation

The computation of the exponential series thanks to the binary splitting approach is quite easy. In the practice, one should stop the splitting recursion when $b-a$ becomes small (the threshold should be an optimized value) to use a classical approach. An slight improvement can also be obtained by using a classical multiplication when $P(a,b)$ and $Q(a,b)$ have a small number of digits.

Thanks to the binary splitting and using the FFT code of *PiFast*, the first million digits of e was computed in less than 10 seconds (pentium II 350). The binary splitting method to compute e is better than any other approaches (much better than the AGM based approach, see *The constant* e).

It must be pointed out that binary splitting needs more memory than the classical method (this is nearly always the case : fast methods are memory consuming). Netherveless, memory is quite easy to manage with this process.

3 Binary splitting on the arctan series

We now present the binary splitting method to compute digits of the arctan series

$$\arctan\left(\frac{1}{q}\right) = \frac{1}{q} - \frac{1}{3q^3} + \frac{1}{5q^5} + \cdots = \sum_{k\geq 0} \frac{(-1)^k}{(2k+1)q^{2k+1}}.$$

The technique is slightly more complicated than with the exponential series. We need to introduce three integers

$$R(a,b) = (2a+3)(2a+5)\cdots(2b+1),$$

$$Q(a,b) = (2a+3)(2a+5)\cdots(2b+1)q^{2(b-a)},$$

$$P(a,b) = (-1)^{a+1}\frac{R(a,b)}{2a+3}q^{2(b-a-1)} + (-1)^{a+2}\frac{R(a,b)}{2a+5}q^{2(b-a-2)} + \cdots + (-1)^{b-1}\frac{R(a,b)}{2b-1}q^2 + (-1)^b\frac{R(a,b)}{2b+1}.$$

Notice that

$$\frac{P(a,b)}{Q(a,b)} = \sum_{k=a+1}^{b} (-1)^k \frac{1}{(2k+1)q^{2(k-a)}}.$$

With the particular case $a = 0$, $b = K$, the expression

$$\frac{1}{q}\left(1 + \frac{P(0,K)}{Q(0,K)}\right)$$

is equal to the first K terms of the arctan series.

The splitting process consists in recursively performing the operations (m denotes the half of $a + b$)

$$P(a,b) = Q(m,b)P(a,m) + R(a,m)P(m,b), \quad Q(a,b) = Q(a,m)Q(m,b), \quad R(a,b) = R(a,m)R(m,b).$$

Now, to compute n decimal digits of $\arctan(1/q)$, one must stop the summation at $k = K$ as soon as $10^n < q^{2K+1}$, which gives $K \simeq n \log(10)/2$. The total cost of the process to compute the first n digits of the series is

$$O(\log(K)M(K\log(K))) = O(\log(n)M(n\ \log(n))).$$

With FFT based multiplication, this leads to the bound $O(n\ \log(n)^3)$.

Notice that the process is more complicated and expensive than for the exponential series for several reasons :

- An auxilliary large integer $R(a,b)$ is needed.
- More products should be performed at each step.
- The large integers $P(0,K)$ and $Q(0,K)$ used to compute the first K terms of the series have size $O(n\ \log(n))$ which is much bigger than n.

The last point is important. Nethertheless, the problem can be overpassed by working with a maximal precision of n decimal digits during the process (this requires a treatment for large integers with limited precision). This solution reduces the memory to $O(n)$ instead of $O(n\ \log(n))$ but only reduces the cost of $O(n\ \log(n)^3)$ by a constant factor (but appreciable). It must be pointed out that this crucial remark for practical implementations does not often appear on the litterature concerning binary splitting.

The technique to compute arctanh series is very similar and can be efficiently used to compute digits of $\log(2)$ (see *The logarithmic constant :log(2)*).

4 Other series and constants

Binary splitting generalizes to all hypergeometric series which involves only integer coefficients (see [1][p. 335]). A succint and recursive general approach is also presented in [4] (note: the authors were unfortunately unaware of the original work of E. A. Karatsuba on FEE, just presented below). The general algorithm also includes the evaluation of such series for non rational parameters.

A method of called FEE (Fast E-function Evaluation) has been pioneered by Ekatherine A. Karatsuba, and while reminiscent of splitting, FEE has a unique and general power of its own. (for details and examples of FEE see

Borwein, Bradley, Crandall *"Computational Strategies for the Riemann Zeta Function"*). The FEE method of E. A. Karatsuba can also be used to compute other constants which do not write in a geometrically convergent hypergeometric series : examples include $\zeta(n)$ for integer values of n [6], Hurwitz Zeta function and Dirichlet L-Series [7].

References

[1] J. M. and P. B. Borwein *Pi and the AGM* John Wiley and sons, (1987)

[2] R. P. Brent, The Complexity of Multiple-Precision Arithmetic, *Complexity of Computational Problem Solving*, R. S. Andressen and R. P. Brent, Eds (Univ. of Queensland Press, Brisbane, 1976)

[3] R. P. Brent, Fast multiple-precision evaluation of elementary functions, *Journal of the ACM*, 23 (1976) 242-251.

[4] B. Haible and T. Papanikolaou, *Fast multiprecision evaluation of series of rational numbers*, report TI-97-7.binsplit, TH Darmstadt. Available *here*.

[5] E. A. Karatsuba, Fast evaluation of transcendental functions, *Problems of Information Transmission* 27 (1991) 339-360.

[6] E. A. Karatsuba, Fast Calculation of the Riemann Zeta function $\zeta(s)$ for Integer Values of the Argument s, *Problems of Information Transmission* 31 (1995) 353-362.

[7] E. A. Karatsuba, Fast evaluation of the Hurwitz Zeta function and Dirichlet L-series, *Problems of Information Transmission* 34 (1998) 342-353.

[8] E. A. Karatsuba, Fast evaluation of hypergeometric function by FEE, Computational Method and Function Theory 1997, Proceeding of the third CMFT Conference (N. Papamichael, St. Ruscheweyh, E.B. Saff, eds) *in Approximation and Decomposition. World Scientific* (1999), 303-314.