УДК [004:504]:[004.5:519.687.1]

О.Л. Шпилинская, К.Р. Сафин

Оптимизация работы программ, использующих процедуру возведения в степень по модулю

Национальный аэрокосмический университет им. Н. Е. Жуковского «Харьковский авиационный институт»

При решении задач, использующих процедуру возведения в степень по модулю, часто возникает необходимость оптимизации программного кода с целью ускорения вычислительных операций с большими числами. В работе рассмотрены алгоритмы, позволяющие увеличить скорость выполнения процедуры модульного умножения. На языке программирования Си разработаны программные коды, реализующие рассмотренные алгоритмы.

Ключевые слова: модулярное возведение в степень, быстрые алгоритмы, натуральные числа, оптимизация, программа

Введение

Оптимизация алгоритмов возведения в степень по модулю позволяет значительно ускорить процесс выполнения кода. Это особенно актуально в тех случаях, когда и основание и показатель степени являются большими числами, например в криптосистемах [1]. Применение специальных алгоритмов в этом случае не только позволяет ускорить работу программы, но и избежать переполнения.

1. Анализ текущего представления проблемы

Стандартные процедуры возведения в степень и нахождения остатка от деления безусловно имеются в каждом языке программирования. Однако их применение к модулярному возведению в степень часто приводят к переполнениям в случаях когда основание и показатель степени достаточно большие числа. Кроме того, применение стандартных процедур не всегда позволяют добиться желаемой скорости работы программ с многоразрядными числами. Решение этих проблем можно добиться путем уменьшения количества умножений и минимизации вычислений при операции возведения в степень по модулю [2].

2.Оптимизация алгоритма работы программ при использовании модулярной арифметики

Пусть требуется вычислить $a^m(mod\ p)$ для некоторых натуральных чисел a, m, p. В случае, когда число p является простым и $(a, p) = 1$ малая теорема Ферма позволяет снижать показатели степени. Действительно, согласно утверждению теоремы в рамках сделанных предположений $a^p \equiv a(mod\ p)$. Простым следствием этой теоремы является сравнение $a^{p-1} \equiv 1(mod\ p)$. Представляя показатель степени $m = k(p-1) + r$ заключаем, что

$$a^m = a^{k(p-1)+r} \equiv a^r(mod\ p). \quad (1)$$

Таким образом, применение малой теоремы Ферма позволяет снижать показатели степени до остатка от деления по модулю $(p-1)$.

Рассмотрим один из алгоритмов быстрого возведения в степень, который сводит процедуру к последовательным возведениям в квадрат и умножениям. Согласно этому методу вычисление $a^m(mod\ p)$ основано на двоичном представлении показателя степени, что делает возможным производить модулярное возведение в степень в среднем в полтора раза быстрее [3]. Пусть показатель степени в двоичной системе счисления имеет представление $m = (m_k m_{k-1} \dots m_0)_2$ то есть

$$m = \sum_{i=0}^{n} m_i 2^i. \quad (2)$$

Используя представление (2) преобразуем вычисляемую величину к виду

$$\mathrm{a^m} = \mathrm{a^{\sum_{i=0}^{n} m_i 2^i}} = \prod_{\mathrm{i=0}}^{\mathrm{n}} \left(\mathrm{a^{2^i}}\right)^{\mathrm{m_i}} (\mathrm{mod\ p}). \quad (3)$$

Алгоритм, позволяющий на основе формулы (3) выполнять возведение в степень по модулю состоит в следующем. Положим $A_0 = a$. Тогда в результате рекурсии согласно формуле

$$A_i = \begin{cases} {A_{i-1}}^2\ (mod\ p), \text{если } m_i = 0, \\ {A_{i-1}}^2 \cdot a\ (mod\ p), \text{если } m_i = 1. \end{cases}$$

Реализация программного кода, использующего приведенный алгоритм представлен на рис. 1.

```
typedef long long int lli;

lli pow1(lli a, lli p, lli modulo)
{
    lli aToTwoToI = (a % modulo);
    lli ans = 1;
    while (p > 0)
    {
        if (p % 2 != 0)
        {
            ans = ans * aToTwoToI;
            ans = ans % modulo;
        }
        aToTwoToI = aToTwoToI * aToTwoToI;
        aToTwoToI = aToTwoToI % modulo;
        p = p / 2;
    }
    return ans;
}
```

Рис.1. Программный код процедуры быстрого возведения в степень при помощи двоичного представления показателя степени на С.

В том случае, когда показатель степени не является простым числом и его можно факторизовать, то есть представить в виде произведения попарно взамнопростых множителей эффективный инструментом для выполнения модулярного умножения дает Китайская теорема об остатках [4]. Согласно этой теореме система сравнений

$$\begin{cases} x \equiv r_1\ (mod\ p_1) \\ \dots\dots \\ x \equiv r_n\ (mod\ p_n) \end{cases},$$

в которой неотрицательные целые числа r_i меньше p_i, имеет единственное решение тогда и только тогда, когда модули попарно взаимнопросты.

Пусть пусть модуль p является произведением попарно взаимно простых чисел $p = p_1 p_2 \dots p_n$. Обозначим $a^m \equiv r_i (mod\ p_i)$. Согласно теореме об остатках для нахождения остатка от деления $a^m (mod\ p)$ достаточно решить систему сравнений

$$\begin{cases} a^m \equiv r_1\ (mod\ p_1) \\ \cdots\cdots \\ a^m \equiv r_n\ (mod\ p_n) \end{cases}. \quad (4)$$

Решение этой системы может быть записано в виде суммы

$$a^m \equiv \sum_{i=1}^{n} \frac{p}{p_i} \cdot y_i \cdot r_i (mod\ p),$$

где y_i – решение сравнения $\frac{p}{p_i} \cdot y_i \equiv 1 (mod\ p_i)$.

Решение сравнений $bx \equiv 1 (mod\ p)$ для случая когда b и p взаимнопросты, осуществляется при помощи следствия теоремы Эйлера. Напомним, что функцией Эйлера $\varphi(\cdot)$ называется отображение множества натуральных чисел на себя, при котором натуральному числу ставиться в соответствие количество меньших взаимнопростых чисел с данным. В рамках сделанных предположений решение указанного сравнения дается формулой $x \equiv b^{\varphi(p)-1} (mod\ p)$.

На рис.2 приведен программный код реализации описанного алгоритма. Согласно оценкам [1] использование алгоритма позволяет сократить время модульного возведения в степень по меньшей мере в два раза.

```
lli pow3(lli a, lli m, const vector<lli> &p)
{
    lli lp = 1;
    for (std::size_t i = 0; i < p.size(); ++i)
    {
        lp *= p[i];
    }
    lli ans = 0;
    for (std::size_t i = 0; i < p.size(); ++i)
    {
        lli yi = pow1(lp / p[i], p[i] - 2, p[i]);
        lli addition = (lp / p[i]) * yi;
        addition %= lp;
        addition *= pow1(a, m, p[i]);
        addition %= lp;
        ans += addition;
        ans %= lp;
    }
    return ans % lp;
}
```

Рис.2. Программный код процедуры быстрого возведения в степень в случае факторизации модуля на попарно взаимнопростые множители.

Выводы

В работе приводятся алгоритмы быстрого модульного возведения в степень. На основе рассмотренных алгоритмов разработаны программные коды

их реализации для случаев простого и составного модуля. Применение данных алгоритмов позволяет ускорять работу соответствующих программных блоков при работе с многоразрядными данными.

Список литературы

1. Крэндалл Ричард, Померанс Карл. Простые числа: Криптографические и вычислительные аспекты / Под ред. и с предисл. В. Н. Чубарикова., М.: УРСС:: Книжный Дом «ЛИБРОКОМ», 2011. - 664 с 1.
2. Schneier B. Applied Cryptography: Protocols, Algorithms and Source Code in C. 2nd Ed. New York:JohnWiley&Sons, 1995. -662 p.
3. Omondi A., Premkumar B. Residue number systems. Theory and implementation. London: Imperial College Press, 2007
4. Мао В. Современная криптография: Теория и практика., М.: Вильямс, 2005.-768 с.

Поступила в редакцию 12.06.2018

Оптимізація роботи програм, що використовують процедуру піднесення в ступінь за модулем

У статті наведенні алгоритми модульного піднесення в ступінь. Розроблений за допомогою цих алгоритмів програмний код дозволяє прискорити роботу та уникнути переповнень.

Ключові слова: модульне піднесення до ступеню, швидкі алгоритми, натуральні числа, оптимізація, програма.

Optimization of the Program Algorithm for the Raising Large Numbers to a Power Modulo

The article presents the algorithms for modular exponentiation. The program code developed by these algorithms allows you to speed up work and avoid overflows.

Keywords: modular exponentiation, fast algorithms, natural numbers, optimization, program

Сведения об авторах:

Шпилинская Ольга Леонидовна – канд. физ.-мат. наук, старший преподаватель каф. 603 «Инженерии программного обеспечения», Национальный аэрокосмический университет им. Н.Е. Жуковского «Харьковский авиационный институт», Украина

Сафин Карим Русланович – студент группы 621пст, Национальный аэрокосмический университет им. Н.Е. Жуковского «Харьковский авиационный институт», Украина