УДК 519.854

АЛГОРИТМ ФАКТОРИЗАЦИИ НА ОСНОВЕ РЕШЕНИЯ КВАДРАТНОГО НЕРАВЕНСТВА

к.т.н. В.Я. Певнев, к.т.н. Н.Ф. Логвиненко, к.т.н. А.В. Шостак
(представил д.ф.- м.н., проф. С.В. Смеляков)

Предложен алгоритм факторизации чисел, эффективно работающий с большими числами. Алгоритм может быть реализован как на последовательных, так и на параллельных структурах.

Эффективными системами криптографической защиты являются асимметричные криптосистемы. Использование асимметричных криптоалгоритмов при выборе соответствующих параметров позволяет обеспечить доказанную вычислительную стойкость. Стойкость асимметричной двухключевой системы зависит от сложности решения задачи разложения исходного числа на простые сомножители. Задача разложения числа на простые сомножители в математике называется задачей факторизации и относится к классу **NP** - полных [1]. Существует большое количество алгоритмов факторизации. Это алгоритмы Эвклида, Ленстры, Поларда и др. Все они эффективны до определенных размеров исходного числа.

Эффективный алгоритм факторизации предложен в [2]. Изложим его суть. Для этого введем следующие обозначения: **N** - число, которое необходимо факторизовать; $\mathbf{P_i}$, $\mathbf{Q_i}$ - переменные, которые зависят от шага алгоритма; Δ - разность между числом **N** и произведением переменных **P** и **Q**; **i** - шаг алгоритма.

На первом шаге алгоритма переменным **P**, **Q** и Δ присваиваются значения, которые равны соответственно:

$$\mathbf{P_1} = \lfloor \sqrt{\mathbf{N}} \rfloor ;$$

$$\mathbf{Q_1} = \lceil \sqrt{\mathbf{N}} \rceil = \mathbf{P_1} + \mathbf{1};$$

$$\Delta_1 = \mathbf{N} - \mathbf{P_1} \cdot \mathbf{Q_1}.$$

Очевидно, что Δ_1 может принимать положительные и отрицательные значения, а также равняться 0. В случае, если $\Delta_1 = \mathbf{0}$, то $\mathbf{P_1}$ и $\mathbf{Q_1}$ – искомые простые числа. Если $\Delta_1 \neq \mathbf{0}$, то необходимо изменить $\mathbf{P_1}$ и $\mathbf{Q_1}$.

Если $\Delta_1 < \mathbf{0}$, то необходимо уменьшить произведение $\mathbf{P_1} \cdot \mathbf{Q}$. (В дальнейшем будем говорить об **i** - м шаге, что не меняет сути изложения). Для этого выполняются следующие действия:

$$\mathbf{P_{i+1}} = \mathbf{P_i} - \mathbf{1} ;$$

$$\Delta_{i+1} = N - (P_i - 1)\cdot Q_i = N - P_i\cdot Q_i + Q_i = \Delta_i + Q_i . \quad (1)$$

В случае, когда $\Delta_i > 0$ имеем:

$$Q_{i+1} = Q_i + 1;$$

$$\Delta_{i+1} = N - P_i\cdot(Q_i + 1) = N - P_i\cdot Q_i - P_i = \Delta_i - P_i . \quad (2)$$

Достоинством изложенного выше алгоритма является быстродействие, так как при его реализации операции сложения - вычитания и умножения больших чисел заменяются операциями сложения - вычитания.

Рассмотрим работу предложенного алгоритма на примере факторизации числа 39. Начальные значения: **P** = 6, **Q** = 7. Результаты работы алгоритма представлены в табл. 1. При анализе можно заметить следующее. После того как Δ стала отрицательной, следующий шаг всегда изменяет знак Δ (шаги 1, 3, 6). Если Δ положительна, то об ее знаке на следующем шаге сказать ничего нельзя (отрицательна – шаги 2, 5; положительна – 4, 7, 8; равна 0 – шаг 9).

Таблица 1

Результаты работы алгоритма

Шаг	**P**	**Q**	**P·Q**	**Δ**
0	6	7	42	-3
1	5	7	35	4
2	5	8	40	-1
3	4	8	32	7
4	4	9	36	3
5	4	10	40	-1
6	3	10	30	9
7	3	11	33	6
8	3	12	36	3
9	3	13	39	0

Докажем, что в (1) $\Delta_{i+1} > 0$. Предположим, что на **i - 1** шаге $\Delta > 0$. Тогда $\Delta_i = N - P_{i-1}\cdot(Q_{i-1} + 1) = \Delta_{i-1} - P_{i-1}$.

По условию $\Delta_i < 0$. Согласно (1) $\Delta_{i+1} = \Delta_i + Q_i = \Delta_{i-1} - P_{i-1} + Q_i$.

Так как $\Delta_{i-1} > 0$, а **P** всегда меньше **Q**, то $\Delta_{i+1} > 0$. Легко показать, что предположение о выборе **i - 1** шага можно заменить любым предыдущим шагом, где $\Delta > 0$. Результат получится аналогичным.

Предположим, что на всех предыдущих шагах $\Delta < 0$. Тогда:

$$Q_i = P + 1 ;$$

$$\Delta_{i+1} = \Delta_i + Q_i = N - P_i\,Q_i + Q_i = N - P_i\cdot(P+1) + (P+1) = N - P_i\cdot P - P_i + P + 1.$$

Так как $P_i < P$ и $P < \sqrt{N}$, то $N - P_i\cdot P > 0$; $P + 1 - P_i > 0$, т.е. $\Delta_{i+1} > 0$.

Следовательно, можно сформулировать следующее утверждение: *если на **i** - м шаге Δ отрицательна, то на **i+1** шаге всегда $\Delta > 0$.*

В этом утверждении следует выделить слово *всегда*. Оно будет ключевым в наших дальнейших рассуждениях.

Введем понятие **j** - го шага. Под ним будем понимать пару **i** -х шагов, где на первом шаге **P:=P - 1** (в дальнейшем **i** - й шаг будем называть полушагом в **j** - м шаге или просто полушагом). Как было отмечено выше, после первого полушага в **j** - м шаге Δ становилась положительной. На втором полушаге **Q:=Q+1**. После второго полушага Δ могла оказаться как отрицательной, так и положительной или равной 0. При нормальной работе алгоритма Δ изменяет знак на каждом полушаге. Если по окончанию **j** - го шага Δ оказалась положительной, то в работе алгоритма имеет место так называемое проскальзывание. Используя понятие проскальзывания, можно составить неравенство

$$N - (P - j) \cdot (Q + j) \geq 0 . \qquad (3)$$

Рассмотрим неравенство (3). Мы предположили, что на каждом шаге алгоритма меняется знак ошибки. Это означает, что после каждого шага знак ошибки будет отрицательным. Если при определении сомножителей происходит проскальзывание, то после **j** - го шага знак ошибки будет положительным или равным нулю. Таким образом, решая неравенство (3) относительно **j**, мы получим количество шагов до первого проскальзывания. Преобразуем неравенство (3) к виду

$$j^2 + j \cdot (Q - P) + (N - Q \cdot P) \geq 0 . \qquad (4)$$

Исследуем неравенство (4). Решение это неравенство будет иметь только при отрицательном свободном члене, т.е. величина ошибки на первом шаге должна быть отрицательна. В том случае, если Δ положительна, то **Q** необходимо увеличить на единицу.

Рассмотрим вероятные значения **j**. Максимальное значение ошибки может быть равным -1. В этом случае **j** будет равна 1. Минимальное значение ошибки будет равно **1 - P**. В этом случае **j** будет равна

$$j = -1 + \sqrt{1 - \Delta} \approx \lfloor \sqrt{P} \rfloor .$$

Для составления следующего неравенства необходимо учесть проскальзывание. Для этого **Q** и **P** изменяем на **j** и **Q** увеличиваем на единицу. Неравенство для определения **j** будет иметь вид

$$j^2 + j \cdot ((Q + \lfloor \sqrt{P} \rfloor + 1) - (P - \lfloor \sqrt{P} \rfloor)) + (N - (Q + \lfloor \sqrt{P} \rfloor + 1) \cdot (P - \lfloor \sqrt{P} \rfloor)) \geq 0 .$$

Произведя несложные преобразования, получим

$$j^2 + 2 \cdot j \cdot (\lfloor \sqrt{P} \rfloor + 1) - (P - \lfloor \sqrt{P} \rfloor - 1)) \geq 0 .$$

При этом **j** будет примерна равна

$$j \approx \lfloor (\sqrt{2} - 1) \sqrt{P} \rfloor .$$

Составляя аналогичные уравнения для каждого из последующих шагов, получаем значения **j**, которые приведены в табл. 2.

Для определения суммарной величины шагов воспользуемся формулой

$$j_{\Sigma} = \sum_{i=1}^{M}\left(\sqrt{P}\cdot\left(\sqrt{i}-\sqrt{i-1}\right)\right) = \sqrt{P}\cdot\sqrt{M},$$

где **M** - количество шагов.

Таблица 2

Значения **j** на шаге **i**

Шаг **i**	1	2	3	4	5	…	**I**
J	$\sqrt{P}$	$\sqrt{P}\left(\sqrt{2}-1\right)$	$\sqrt{P}\left(\sqrt{3}-\sqrt{2}\right)$	$\sqrt{P}\left(\sqrt{4}-\sqrt{3}\right)$	$\sqrt{P}\left(\sqrt{5}-\sqrt{4}\right)$	…	$\sqrt{P}\left(\sqrt{i}-\sqrt{i-1}\right)$

Для того, чтобы перебрать все возможные варианты сомножителей, достаточно составить и решить $\sqrt{M} = \sqrt{P}$ неравенств. Такой подход с математической точки зрения будет правомочен, но если посмотреть на суть задачи, то можно найти некоторые возможности, позволяющие ускорить работу алгоритма.

Во-первых, при приближении **Q** к значению **2P**, величина шага будет уменьшаться и время, потраченное на решение неравенства, будет больше, чем время при решении задачи путем обычного перебора. Подсчитав и сравнив время, затрачиваемое на решение неравенства и при использовании алгоритма, приведенного в [2], можно сделать вывод, что при размере шага 4 можно приступать к решению задачи в «лоб». Во-вторых, когда **Q** становится больше **P** в **k** раз, то на каждом шаге это необходимо учитывать при составлении неравенства. При этом общее время поиска решения значительно уменьшается. В третьих, предложенный подход позволяет легко распараллелить процесс поиска решения. Для этого выбираются непересекающиеся области числовой оси. В одной из этих областей берется какое-либо произвольное число **P**, на которое делится исходное. В результате получается второе число **Q**, участвующее в составлении неравенства.

В статье предложен алгоритм факторизации чисел, основанный на решении квадратного уравнения. Эффективность этого алгоритма будет увеличиваться с ростом размерности факторизуемого числа. Достоинством алгоритма является и возможность его распараллеливания на любом количестве процессоров.

ЛИТЕРАТУРА

1. Гери М., Джонсон Д. Вычислительные машины и трудноразрешаемые задачи. – М.: Мир, 1982. – 416 с.

2. Певнев В.Я., Дулгер И. Алгоритм факторизации // Системи обробки інформації. – Харків: НАНУ, ПАНМ, ХВУ. – 2000. – Вип.1(7). – С. 154 - 157.

Поступила в редколлегию 28.02.2001