ЧЕБЫШЕВСКИЙ СБОРНИК

Том 19. Выпуск 2

УДК 511.17

DOI 10.22405/2226-8383-2018-19-2-420-430

**О расширенном алгоритме Джебелеана–Вебера–Седжелмасы
вычисления наибольшего общего делителя**

Долгов Дмитрий Александрович — аспирант кафедры системного анализа и информационных технологий, Казанский Федеральный Университет.

e-mail: Dolgov.kfu@gmail.com

Аннотация

Существует большое количество различных алгоритмов вычисления Н.О.Д. Прежде всего стоит отметить алгоритмы типа Шонхаге. Они используются для очень больших чисел и имеют наилучшую асимптотическую сложность в худшем случае — $O(n \log^2(n) \log(\log(n)))$. Для чисел поменьше используются обобщенный бинарные алгоритмы. Все они основаны на k -арной редукции: $\alpha \gcd(u, v) = \gcd(v, \frac{|au \pm bv|}{k})$, целые $u > v > 0$, $\gcd(u, k) = \gcd(v, k) = 1$, $\alpha \geq 1$. Знак $+$ или $-$ ставится в зависимости от версии выбранного алгоритма. Основная задача — подобрать коэффициенты a, b таким образом, чтобы выполнялось $au + bv = 0 \pmod k$. Число k обычно выбирают равным простому числу или степени простого числа. Недостаток алгоритмов в том, что в ходе вычислений могут накапливаться дополнительные множители, поэтому в рекуррентном соотношении в начале стоит множитель $\alpha \geq 1$. Если $k = 2^s$, то получаем обобщенные бинарные алгоритмы. Вебер первым предложил алгоритм поиска коэффициентов на основе подаваемых чисел, его обобщенный бинарный алгоритм работает в пять раз быстрее, чем бинарный алгоритм. Седжелмасы модифицировал алгоритм Джебелеана–Вебера, избавив его от дополнительных множителей, асимптотическая сложность алгоритма в худшем случае — $O(n^2 / \log(n))$.

Коэффициенты Безу — представление Н.О.Д. d чисел A, B в виде линейной комбинации $Ax + By = d$, где x и y — целые числа, называемые коэффициентами Безу. В этой статье предложен расширенный алгоритм Джебелеана–Вебера–Седжелмасы вычисления Н.О.Д. двух натуральных чисел, выводятся необходимые формулы и приводятся примеры, показывающие как можно вычислять обратные элементы.

Ключевые слова: НОД, алгоритм Евклида, бинарный алгоритм, k -арный алгоритм, алгоритм Шонхаге, алгоритм Вебера, алгоритм Лемера

Библиография: 19 названий.

Для цитирования:

Д. А. Долгов. О расширенном алгоритме Джебелеана–Вебера–Седжелмасы вычисления наибольшего общего делителя // Чебышевский сб. 2018. Т. 19, вып. 2. С. 420–430.

CHEBYSHEVSKII SBORNIK

Vol. 19. No. 2

UDC 511.17

DOI 10.22405/2226-8383-2018-19-2-420-430

An extended Jebelean–Weber–Sedjelmaci GCD algorithm

Dolgov Dmitry Alexandrovich — Postgraduate student of the Department of System analysis and Information Technologies, Kazan Federal University.

e-mail: Dolgov.kfu@gmail.com

Abstract

There are large number of different algorithms for gcd computation. First of all, it is the Shönhage type gcd algorithms. They are used for very large numbers and have the best asymptotic complexity in the worst case — $O(n \log^2(n) \log(\log(n)))$. For lesser numbers, generalized binary algorithms are used. They are based on the k -ary reduction: $\alpha \gcd(u, v) = \gcd(v, \frac{|au \pm bv|}{k})$, integers $u > v > 0$, $\gcd(u, k) = \gcd(v, k) = 1$, $\alpha \geq 1$. We use $+$ or $-$ depending on the type of gcd algorithm. The main task is to select the coefficients a, b such that $au + bv = 0 \pmod k$. The number k is usually chosen as a prime number or a power of a prime number. Unfortunately, spurious factors can accumulate during the computation, so $\alpha \geq 1$. If $k = 2^s$, then we obtain generalized binary algorithms. It's five times faster than the binary gcd algorithm. Sedjelmaci modified the Jebelean–Weber's algorithm, he removed spurious factors. Asymptotic complexity of the algorithm in the worst case is $O(n^2/\log(n))$.

Bezout Coefficients is a representation of gcd d of the numbers A, B of a linear combination $Ax + By = d$, where x and y are integer numbers, called Bezout coefficients. In this paper we introduce the extended Jebelean–Weber–Sedjelmaci algorithm of two natural numbers, formulas and give examples that showing how to calculate inverse elements.

Keywords: GCD, Euclidean gcd, binary gcd, k -ary gcd, Shönhage gcd, Weber gcd, Lehmer gcd.

Bibliography: 19 titles.

For citation:

D. A. Dolgov. 2018, "An extended Jebelean–Weber–Sedjelmaci GCD algorithm", *Chebyshevskii sbornik*, vol. 19, no. 2. pp. 420–430.

1. Введение

Наибольшим общим делителем (Н.О.Д., $\gcd$) двух натуральных чисел u, v отличных от 0, называется наибольшее целое число d , такое что $d|u$ и $d|v$. Все алгоритмы вычисления Н.О.Д. следуют одной и той же идее сокращения u и v до u' и v' , так что $\gcd(u', v') = \gcd(u, v)$. Один из наиболее известных алгоритмов вычисления Н.О.Д. — алгоритм Евклида (АЕ), основанный на рекуррентном соотношении с использованием модульной редукции: $\gcd(u, v) = \gcd(v, u \bmod v)$, где $u > v > 0$, которое выполняется до тех пор пока второй аргумент не станет равным 0. Тогда первый аргумент пары равен Н.О.Д. Асимптотическая временная сложность алгоритма в худшем случае составляет $O(n^2)$ для двух n -битных чисел [3].

Расширенная версия АЕ (РАЕ) используется для нахождения вместе с Н.О.Д. d так называемых коэффициентов Безу, которые являются коэффициентами линейной комбинации $au + bv = d$. Работа расширенного алгоритма Евклида состоит из двух этапов. Первый этап совпадает со стандартным алгоритмом с накоплением целых чисел $DIV = [u/v]$. На втором этапе коэффициенты Безу вычисляются по формулам (n — значение последней итерации):

$$u_n = 0, v_n = 1, u_i = v_{i+1}, v_i = u_{i+1} - v_{i+1} \cdot [A/B]_i \quad (1)$$

В 1967 Джозефом Стайном был получен бинарный алгоритм Н.О.Д., основанный на рекуррентном соотношении: $\gcd(u, v) = \gcd(v, (u - v)/2)$, где u, v — нечетные и $u > v > 0$ [3,4]. Как и в АЕ, алгоритм выполняется пока второй аргумент не станет равен 0, в первом будет стоять Н.О.Д. Асимптотическая временная сложность алгоритма в худшем случае составляет $O(n^2)$ для двух n -битных чисел. Преимуществом данного алгоритма перед АЕ является скорость работы: при одинаковой асимптотической временной сложности операция деления на произвольное нечетное число выполняется медленнее, чем операция деления на степень двойки, т.к. данную операцию можно заменить правым битовым сдвигом.

Алгоритм Лемера [1] — усовершенствованный алгоритм Евклида. Алгоритм работает со старшими разрядами больших чисел. Большая часть частных, полученных в ходе деления на каждом шаге, довольно мала. Например, Кнут заметил, что частные один, два и три составляют 67,7% всех частных [5]. Эти небольшие частные могут быть получены только из старших разрядов исходных чисел. Таким образом, алгоритм начинается с выделения старших разрядов и вычисления последовательности частных. Это позволяет выполнять большую часть вычислений с помощью арифметики однократной точности и существенно сэкономить на операциях многократной точности. Оценка сложности алгоритма Лемера была улучшена Соренсоном: n — количество бит чисел u и v , k — число старших бит, необходимых для подсчета частных, получаемых в ходе делений на каждом шаге алгоритма. Если $k = \lfloor \log(n)/4 \rfloor$, то сложность алгоритма в худшем случае составит $O(n^2 / \log(n))$ [6].

Одним из самых асимптотически быстрых алгоритмов вычисления Н.О.Д. является алгоритм Шонхаге [2], его асимптотическая временная сложность — $O(n \log^2(n) \log(\log(n)))$. Это классический рекурсивный алгоритм вычисления Н.О.Д., применяемый для очень больших чисел. На практике обнаруживается, что рекурсивные схемы Н.О.Д. превосходят все известные альтернативы (например, различные бинарные алгоритмы), когда входные аргументы u, v достаточно велики, скажем, порядка десятков тысяч бит [3]. Классические алгоритмы на основе АЕ вычисляют частные q_j и остатки r_j на каждом шаге. В худшем случае сумма битовой длины последовательности $\{r_j\}$ пропорциональна n^2 . Таким образом, любой алгоритм, который явно вычисляет каждый член последовательности $\{r_j\}$, имеет, по меньшей мере, квадратичную сложность [7]. С другой стороны, сумма битовых длин последовательности $\{q_j\}$, которая связана с $\{r_j\}$, пропорциональна n , т.к. большинство членов $\{q_j\}$ довольно малы. Поэтому члены r_j лучше получать с помощью членов q_j [3,7].

Стэлле в 2004 получил бинарный рекурсивный алгоритм, имеющий такую же слож-

ность, как и алгоритм Шонхаге [8]. У нового алгоритма получается меньшая эффективная константа в "big-O". Основная идея нового алгоритма заключается в расширении рационального числа в цепную дробь, элементы которой берутся не из обычных натуральных чисел, а скорее из набора, построенного следующим образом: $\{a_i/b_i\} = (\pm\frac{1}{2}, \pm\frac{1}{4}, \pm\frac{3}{4}, \pm\frac{1}{8}, \pm\frac{3}{8}, \pm\frac{5}{8}, \pm\frac{7}{8}, \pm\frac{1}{16}, \pm\frac{3}{16}, \dots)$. При подсчете $\gcd(u, v)$ необходимо выражать на каждом шаге $u_1 = (a_t/b_t) \cdot v_1 + r_1$, где r_i — некоторый остаток. Далее рекурсивно вычисляется $v_1 = (a_l/b_l) \cdot r_1 + r_2$ и так до тех пор, пока $r_k \neq 0$. Для повышения эффективности алгоритма для небольших операндов используется обычно умножение, для больших операндов можно использовать алгоритм Карацубы или Тома-Кука, для очень больших операндов используется быстрое преобразование Фурье [3].

Но есть и другие более современные алгоритмы, работающие с числами произвольной длины. В 1990-х Соренсон получил обобщение бинарного алгоритма Н.О.Д., назвав его k -арным алгоритмом [9,10]. Сначала была описана версия с использованием "правого сдвига" (right-shift), потом с использованием левого сдвига (left-shift). Любой k -арный алгоритм использует рекуррентное соотношение: $\alpha \gcd(u, v) = \gcd(v, \frac{au \pm bv}{k})$, целые $u > v > 0$, $\gcd(u, k) = \gcd(v, k) = 1$, $\alpha \geq 1$, знак $+$ или $-$ ставится в зависимости от версии выбранного алгоритма: с левым или правым сдвигом. Основная задача — подобрать коэффициенты a, b таким образом, чтобы выполнялось $au + bv = 0 \pmod k$. Число k обычно выбирают как простое число или степень простого числа. Соренсон показал, что всегда можно найти a, b : $0 < |a|, |b| < \sqrt{k}$. Минус алгоритма в том, что в ходе вычислений могут накапливаться дополнительные множители, поэтому в рекуррентном соотношении в начале стоит $\alpha \geq 1$. По сравнению с АЕ алгоритм Соренсона позволяет значительно уменьшить сложность вычислений с помощью выбора параметра k . Если u, v — два n битных числа, k — простое и $k = O(n^2/\log^3(n))$, то сложность алгоритма $O(n^2/\log(k))$ [10]. В дальнейшем были получены обобщенные бинарные алгоритмы с $k = 2^s$ [12,13]. Вебер первым предложил алгоритм поиска коэффициентов на основе подаваемых чисел, его обобщенный бинарный алгоритм работает в пять раз быстрее, чем бинарный алгоритм [3,13]. Также он ввел операцию dmod , которая позволяет получить новое число необходимой битовой длины, содержащее Н.О.Д. В [11] были получены обобщения данной операции для многочленов над конечными полями. Седжелмаси модифицировал алгоритм Вебера, избавив его от дополнительных множителей [14].

Вычисление Н.О.Д. — распространенная операция, имеющая много приложений. Одно из таких приложений — задача факторизации натуральных чисел. В первую очередь, стоит отметить ρ алгоритм Полларда, $p-1$ алгоритм Полларда, позволяющие эффективно факторизовать числа, в состав которых входят небольшие по размеру множители [3]. Еще стоит отметить субэкспоненциальный алгоритм факторизации Ленстры с использованием эллиптических кривых. Данный метод лучше всего подходит для нахождения небольших простых делителей числа до 83 бит (текущий рекорд принадлежит Р. Пропперу [15]). Сложность метода сильно зависит от размера наименьшего простого множителя факторизируемого n и только слабо от самого n . По этой причине, в последние годы было обнаружено много множителей поистине гигантских чисел. Многие из этих чисел лежат далеко за пределами возможностей алгоритмов квадратичного решета и общего метода решета числового поля [3,15]. Еще одно применение данной операции — вычисление псевдопростых и сильнопсевдопростых чисел [12].

Вычисление Н.О.Д. можно обобщить на многочлены. Данная операция применима для решения линейных диофантовых уравнений, следовательно, для решения линейных уравнений над кольцами вычетов. Например, для вычисления мультипликативного обратного для элементов поля классов вычетов по модулю некоторого неприводимого многочлена (для этого используются расширенные алгоритмы, например РАЕ); для разложения многочленов в произведение степеней многочленов. Также такие вычисления используются в задачах теории кодирования [18,19]. Одно из применений — декодирование кодов Рида-Соломона.

При декодировании необходимо решить ключевое уравнение $S(z)\sigma(z) = z^{2t}\Phi(z) - \omega(z)$, или $-\omega(z) = \sigma(z)S(z) \pmod{z^{2t}}$, где $\sigma(z)$ - многочлен локаторов ошибок, а $\omega(z)$ - многочлен значений ошибок. Декодирование с помощью расширенного алгоритма вычисления Н.О.Д. многочленов над $GF(q)$ длины n является не только наиболее простым для понимания, но и по быстрдействию оно по меньшей мере сравнимо с другими методами при $n < 10^6$ [16].

Впервые формулы для вычисления коэффициентов Безу (КБ) появились в [17]. Там рассматривалось два случая. В первом случае $\gcd(C_i, k) = 1$ и $C_i < B_i$, где $C_i = (x_i A_i + y_i B_i)/k$. В этом случае новая пара (A_{i+1}, B_{i+1}) равнялась $(B_i, (x_i A_i + y_i B_i)/k)$. Во втором случае $\gcd(C_i, k) \neq 1$. Случай, где $C_i \geq B_i$ и переложение формул для алгоритма Вебера (или его модификаций), не рассматривался. В алгоритме Джебелеана–Вебера–Седжелмаси не накапливаются дополнительные множители во время прямого хода, он обладает одной из лучших асимптотических сложностей среди остальных обобщенных бинарных алгоритмов. В данной статье автором вводится расширенный алгоритм Джебелеана–Вебера–Седжелмаси вычисления Н.О.Д двух натуральных чисел, выводятся необходимые формулы и приводятся примеры, показывающие как можно вычислять обратные элементы.

2. Алгоритм Джебелеана–Вебера–Седжелмаси

Ниже представлен алгоритм Седжелмаси выбора коэффициентов для обобщенного алгоритма Вебера–Джебелеана (JWA). Это модифицированная версия алгоритма Вебера [13]. В отличие от алгоритма Вебера здесь также возвращаем результат предыдущей итерации. Следующая теорема доказывает отсутствие дополнительных множителей, возникающих в ходе работы других k -арных алгоритмов [14].

ТЕОРЕМА 1. (Седжелмаси) Пусть $u \geq v \geq 1$ и $k \geq 1$ — три целых, $\gcd(u, k) = \gcd(v, k) = 1$. Пусть $M = \begin{pmatrix} n_1 & d_1 \\ n_2 & d_2 \end{pmatrix}$ — целочисленная матрица, полученная из алгоритма SJWA, $u/v < \sqrt{k}$. Если $\begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = \begin{pmatrix} |n_1 v - d_1 u|/k \\ |n_2 v - d_2 u|/k \end{pmatrix}$, тогда R_1, R_2 — целые, удовлетворяющие $0 \leq R_1 \leq v$, $0 \leq R_2 \leq 2u/\sqrt{k}$ и $\gcd(u, v) = \gcd(R_1, R_2)$.

Ниже представлен алгоритм Джебелеана–Вебера–Седжелмаси SJWA-GCD. Выбор k обусловлен размером машинного слова. Также автор предложил использовать предвычисленные таблицы (precompute tables), хранящие всевозможные произведения, частные и остатки пар чисел $\leq 2^m$, см. Лемма 3 [6] или Лемма 4.1 [10]. При выполнении вычислений с большими числами, необходимо разбить каждое из чисел на m двоичных блоков и выполнять всю арифметику поблочно, используя таблицы для вычисления произведений и частных чисел, ограниченных 2^m . Функция $makeodd(u)$ приводит число к нечетному виду: пока u — нечетно, то $u = u/2$.

3. Расширенный алгоритм Джебелеана–Вебера–Седжелмаси

Сначала выведем для SJWA-GCD формулы, аналогичные (1). Пусть (A_{i+1}, B_{i+1}) — пара чисел, рассматриваемая на текущем шаге алгоритма и $k = 2^s$, $s > 0$. $A_{i+1}/B_{i+1} < \sqrt{k}$, значит, запускаем SJWA-GCD.

3.1. Отсутствие дополнительных множителей

Рассмотрим случай, когда дополнительные множители не появляются в ходе работы алгоритма. Для каждой пары чисел можно записать соотношение Безу: $u_{i+1}A_{i+1} + v_{i+1}B_{i+1} = d$.

Algorithm: SJWA-GCD*Input:* 2 нечетных числа $u \geq v \geq 3$ *Output:* $\gcd(u, v)$

```

 $k = 2^{32}$  или  $k = 2^{64}$ 
 $n = \lfloor \log_2(u) \rfloor + 1$ 
if  $n^{0.4} > k$  then
   $m = \lfloor 0.4 \log_2(n) \rfloor + 1$ 
   $m = m + (m \bmod 2); k = 2^m;$ 
  Precompute tables
end
while  $uv \neq 0$  do
  if  $u < v$  then
     $(u, v) = (v, u)$ 
  end
  if  $u/v < \sqrt{k}$  then
     $\begin{pmatrix} c & d \\ a & b \end{pmatrix} = SJWA(u, v, k)$ 
     $u = |du - cv|/k;$ 
     $v = |bu - av|/k$ 
  else
     $(u, v) = (v, u \bmod v)$ 
  end
   $makeodd(u); makeodd(v)$ 
end
return  $u + v$ 

```

Algorithm: SJWA*Input:* $x, y > 0, m \geq 4:$ $\gcd(k, x) = \gcd(k, y) = 1$ *Output:* 2×2 целочисленнаяматрица $M = \begin{pmatrix} n_1 & d_1 \\ n_2 & d_2 \end{pmatrix},$ где $0 < n_2, |d_2| < \sqrt{k},$ $n_2 y = d_2 x \pmod{k}$ и $n_1 y = d_1 x \pmod{k}$ $r = x/y \bmod k$ $f_1 = (n_1, d_1) = (k, 0)$ $f_2 = (n_2, d_2) = (r, 1)$ **while** $n_2 \geq \sqrt{k}$ **do**
 $f_1 = f_1 - \lfloor n_1/n_2 \rfloor f_2$
 $swap(f_1, f_2)$
end**return** $M = \begin{pmatrix} n_1 & d_1 \\ n_2 & d_2 \end{pmatrix}$

На каждом шаге идет подсчет двух k -арных редукций $KR_1 = \frac{|n_2 v - d_2 u|}{k}$, $KR_2 = \frac{|n_1 v - d_1 u|}{k}$. $A_{i+1} = \max(KR_1, KR_2)$, $B_{i+1} = \min(KR_1, KR_2)$. Сгруппировав члены A_{i+1}, B_{i+1} получим формулы, для случая $KR_2 \geq KR_1$:

$$B_i \frac{(n_1 u_{i+1} + n_2 v_{i+1})}{k} - A_i \frac{d_1 u_{i+1} + d_2 v_{i+1}}{k} = d$$

Соответствующие формулы u_i, v_i имеют вид:

$$u_n = 0, v_n = 1, u_i = (u_{i+1} n_1 + v_{i+1} n_2)/k, v_i = (-u_{i+1} d_1 - v_{i+1} d_2)/k, 1 \leq i < n. \quad (2)$$

Если $KR_1 > KR_2$:

$$u_n = 0, v_n = 1, u_i = (u_{i+1} n_2 + v_{i+1} n_1)/k, v_i = (-u_{i+1} d_2 - v_{i+1} d_1)/k, 1 \leq i < n. \quad (3)$$

В начале обратного хода u_n, v_n присваиваются нуль и один, соответственно. u_i, v_i — рациональные числа, в ходе расчетов идет накопление k^{n-i} , где i — текущая итерация. КБ — целые числа. Формулы (2) дают нам дроби, тогда как КБ должны быть целые числа. В ходе расчетов идет накопление k^{n-i} , где i — текущая итерация. Записав соотношение Безу для текущей итерации получим $u_i A_i + v_i B_i = d$, возьмем остаток от деления на A_i [17]. Тогда можно переписать формулы:

$$u'_i = u_i \bmod A_i, v'_i = v_i \bmod A_i, 1 \leq i < n \quad (4)$$

$$u'_i A_i + v'_i B_i = d \pmod{A_i}, u'_i A_i + v'_i B_i = d + t A_i, t \in \mathbb{Z} \quad (5)$$

Из последнего выводится требуемое выражение для коэффициентов Безу:

$$u_i A_i + v_i B_i = d, \text{ где } u = u'_i - t, v = v'_i, u, v \in \mathbb{Z} \quad (6)$$

u, v — ответ. Если $A_{i+1}/B_{i+1} \geq \sqrt{k}$, то запускается АЕ с формулами (1). Покажем, как можно эффективно вычислять $x = k^{-1} \bmod A$, см. [17]. По определению,

$$kx \bmod A = 1 \Leftrightarrow kx = 1 + pA \quad (7)$$

Возьмем $a = A \bmod k$. Если в (7) взять остаток от деления на k , получим $pA = -1 \bmod k \Leftrightarrow p = -a^{-1} \bmod k$, $p = -a^{-1} + rk$, $r \in \mathbb{Z}$.

$$x = \frac{1 + (-a^{-1} + rk)A}{k} = rA + \frac{1 - a^{-1}A}{k} \Rightarrow k^{-1} \bmod A = \frac{1 - a^{-1}A}{k} \pmod{A} \quad (8)$$

3.2. Наличие дополнительных множителей

По теореме 1 в ходе прямого хода алгоритма SJWA-GCD не возникает никаких дополнительных множителей. Тем не менее в рамках обратного хода при вычислении u' , v' в знаменателе может стоять не обязательно k^s , а $2^r k^s$.

Пусть для $i+1$ выполнено $A_{i+1}u_{i+1} + B_{i+1}v_{i+1} = d$. Первый случай: на i шаге выполнялся SJWA-GCD. Пусть KR_1 и KR_2 при сокращении получились четными, приведение их к нечетному — деление на 2^l , 2^m соответственно. Подставив значения KR_1, KR_2 можно найти u_i, v_i . Если на i шаге (обозначим следующий список (9)):

- $KR_2 \geq KR_1$, $l \geq m$, то $u_i = \frac{u_{i+1}n_1 + 2^{l-m}v_{i+1}n_2}{2^l k}$, $v_i = \frac{u_{i+1}d_1 + 2^{l-m}v_{i+1}d_2}{2^l k}$
- $KR_2 \geq KR_1$, $l < m$, то $u_i = \frac{2^{m-l}u_{i+1}n_1 + v_{i+1}n_2}{2^m k}$, $v_i = \frac{2^{m-l}u_{i+1}d_1 + v_{i+1}d_2}{2^m k}$
- $KR_1 > KR_2$, $l \geq m$, то $u_i = \frac{u_{i+1}n_2 + 2^{l-m}v_{i+1}n_1}{2^l k}$, $v_i = \frac{u_{i+1}d_2 + 2^{l-m}v_{i+1}d_1}{2^l k}$
- $KR_1 > KR_2$, $l < m$, то $u_i = \frac{2^{m-l}u_{i+1}n_2 + v_{i+1}n_1}{2^m k}$, $v_i = \frac{2^{m-l}u_{i+1}d_2 + v_{i+1}d_1}{2^m k}$

После расчета необходимых коэффициентов может оказаться, что $A_i u_i + B_i v_i = 2^{r_i} d$, т.е. присутствует дополнительный множитель в соотношении Безу. В этом случае необходимо: $u_i = u_i / 2^{r_i}$, $v_i = v_i / 2^{r_i}$. А дальше необходимо пользоваться формулами (4)–(6).

Второй случай: на i шаге выполнялся АЕ. Аналогично можно вывести формулы для случая, если MOD — нечетно и необходимо сократить на 2^l . Тогда,

$$u_i = v_{i+1}, v_i = 2^l u_{i+1} - v_{i+1} [A/B]_i \quad (10)$$

Аналогично, если после этого соотношение Безу будет нечетным и равняться 2^{r_i} : $u_i = u_i / 2^{r_i}$, $v_i = v_i / 2^{r_i}$. Если на всех шагах алгоритма выполняется $A_i/B_i \geq \sqrt{k}$, то выполняется всегда АЕ, и в столбцах u_i, v_i будут храниться коэффициенты Безу.

3.3. Корректность расширенного алгоритма

ТЕОРЕМА 2. Пусть A, B — целые числа, для которых ищем коэффициенты Безу, n — число шагов SJWA-GCD, $d = \gcd(A, B)$, i — номер текущего шага, следуя формулам (2)–(6), (9), (10), получим для $i = t$ $uA_t + vB_t = d$.

ДОКАЗАТЕЛЬСТВО.

- $i = n$. Очевидно: $u_n A_n + v_n B_n = d$.
- $i = t$ и для $i = t - 1$ выполняется соотношение Безу. Если при $i = t$ выполнялся алг. SJWA-GCD, то выразив A_{i+1}, B_{i+1} через A_i, B_i получим формулы (2) или (9). Разница формул заключается в том, что на каждом шаге необходимы нечетные числа. И при вычислении редукций могут возникнуть четные числа.

Аналогично, если выполнялся АЕ, получим формулы (1), (10).

По теореме 1 в рамках прямого хода никаких дополнительных множителей не возникает. Но, если $u_k A_k, v_k B_k$ — нечетные, то $u_k A_k + v_k B_k = 2^l d$. Значит, $u_t = u_t / 2^l$, $v_t = v_t / 2^l$. u_t, v_t — рациональные числа. Избавляемся от $2^l k^s$ в знаменателе u_t, v_t по формуле (4), (5). $(u'_i - t)A_i + v'_i B_i = d$, и получили формулы (6). $\square$

3.4. Примеры

ПРИМЕР 1. Решить уравнение $89x = 1 \pmod{97}$, 97 и 89 — простые числа, $k = 16$.

Таблица 1: пошаговое решение

№	A_i	B_i	$coef$	KR_1	KR_2	/	%	u_n	v_n	u'	v'	u	v
1	97	89	$\begin{pmatrix} 7 & -1 \\ 2 & 2 \end{pmatrix}$	1	45	—	—	$2/k$	$-2/k$	85	12	-11	12
2	45	1	—	—	—	45	0	0	1	—	—	—	—

Знак / означает $[A_i/B_i]$, % означает $A_i \bmod B_i$. На первом шаге запускается алгоритм SJWA-GCD. На втором шаге запускается алгоритм Евклида. Условие выхода: $MOD = 0$. Коэффициенты $coef$ из алгоритма SJWA.

В рамках обратного хода на первом шаге выбраны формулы $KR_2 > KR_1$: $u_1 = (u_2n_1 + v_2n_2)/k = 2/k$, $v_1 = (u_2d_1 + v_2d_2)/k = -2/k$. $u'_1 = u_1 \bmod A_1 = 85$, $v'_1 = v_1 \bmod A_1 = 12$.

$u'_1A_1 + v'_1B_1 = 8245 + 1068 = 9313 = 1 + tA_1$, $t = 96$. $u = u'_1 - t = 85 - 96 = -11$, $v = v'_1 = 12$. Проверка: $89 \cdot 12 - 97 \cdot 11 = 1$. Ответ: 12.

ПРИМЕР 2. Решить уравнение $405x = 1 \pmod{971}$, $\gcd(971, 405) = 1$, $k = 16$.

На первом шаге запускается алгоритм SJWA-GCD. На втором и третьем шаге запускается алгоритм Евклида. На четвертом и пятом запускается алгоритм SJWA-GCD. Условие выхода: $KR_1 = 0$ (в рамках другого примера может быть $KR_2 = 0$). Коэффициенты $coef$ из алгоритма SJWA.

Таблица 2: пошаговое решение

№	A_i	B_i	$coef$	KR_1	KR_2	/	%	u_n	v_n	u'	v'	u	v
1	971	405	$\begin{pmatrix} 1 & -1 \\ 15 & 1 \end{pmatrix}$	43	319	—	—	$\frac{-173}{8192}$	$\frac{435}{8192}$	437	585	-244	585
2	319	43	—	—	—	7	9	$\frac{19}{256}$	$\frac{-135}{256}$	263	230	-31	230
3	43	9	—	—	—	4	7	$\frac{-1}{128}$	$\frac{19}{128}$	1	24	-5	24
4	9	7	$\begin{pmatrix} 1 & -1 \\ 15 & 1 \end{pmatrix}$	1	3	—	—	$\frac{15}{128}$	$\frac{-1}{128}$	3	4	-3	4
5	3	1	$\begin{pmatrix} 3 & 1 \\ 16 & 0 \end{pmatrix}$	0	1	—	—	0	1	—	—	—	—

Знак / означает $[A_i/B_i]$, % означает $A_i \bmod B_i$. По умолчанию $u_5 = 0$, $v_5 = 1$. $u_4 = (u_5n_1 + v_5n_2)/k = 15/16$, $v_4 = (u_5d_1 + v_5d_2)/k = -1/16$. На данном шаге произошло лишнее сокращение на два в KR_2 , т.к там было четное число. Запишем соотношение Безу: $(9 \cdot 15 - 7)/16 = 8$. Возник доп. множитель — степень двойки. Увеличим знаменатель до $8 \cdot 16 = 128$, т.е. $u_4 = 15/128$, $v_4 = -1/128$.

Для проверки найдем коэффициенты Безу (КБ) для четвертого шага алгоритма. $u'_4 = u_4 \bmod A_4 = 15/128 \bmod 9 = 15 \cdot 5 \bmod 9 = 3$. $v'_4 = v_4 \bmod A_4 = -1/128 \bmod 9 = 4$. Найдем t . $9 \cdot 3 + 4 \cdot 7 = 55 = 1 + tA_4$, $t = 6$. $u = u'_4 - t = -3$. $v = v'_4 = 4$. Соотношение Безу: $-3 \cdot 9 + 4 \cdot 7 = 1$.

На третьем шаге применяем формулы из РАЕ. $u_3 = -1/128$, $v_3 = (15 + 4 \cdot 1)/128 = 19/128$. Проверка: $(19 \cdot 9 - 43 \cdot 1)/128 = 1$. Найдем КБ. $u'_3 = -1/128 \bmod 43 = 1$, $v'_3 = 19/128 \bmod 43 = 24$. $43 + 24 \cdot 9 = 259 = 1 + 43t$, $t = 6$, $u = 1 - 6 = -5$, $v = 24$.

На втором шаге вновь применяем формулы из РАЕ. $u_2 = 19/256$, $v_2 = -(1 \cdot 2 + 19 \cdot 7)/256$. На этом шаге произошло лишнее сокращение на двойку в %. Поэтому присутствует лишняя двойка в числителе v_2 , записав соотношение Безу, добавили еще одну двойку в знаменатель и получилось 256 вместо 128. Найдем КБ. $u'_2 = 19/256 \bmod 319 = 263$, $v'_2 = -135/256 \bmod 319 = 230$. $263 \cdot 319 + 230 \cdot 43 = 93787 = 1 + 319t$, $t = 294$, $u = -31$, $v = 230$.

На первом шаге снова найдем $u_1 = (2 \cdot 19 \cdot 15 - 135)/(32 \cdot 256) = 435/8192$, $v_1 = (-2 \cdot 19 - 135)/(32 \cdot 256) = -173/8192$. В KR_1 появляется лишнее сокращение на двойку, поэтому в числителе u_1, v_1 появляется два. Записав соотношение Безу также появляется два в знаменателе, поэтому вместо $16 \cdot 256$ появляется $32 \cdot 256$. $u'_1 = -173/8192 \bmod 971 = 437$, $v'_1 = 435/8192 \bmod 971 = 585$. $437 \cdot 971 + 405 \cdot 585 = 661252 = 1 + 971t$, $t = 681$, $u = 437 - 6881 = -244$, $v = 585$. Проверка: $-244 \cdot 971 + 405 \cdot 585 = 1$. Ответ: 585.

Поиск КБ на каждом шаге был сделан для наглядности. В действительности эту процедуру (заполнение столбцов u', v', u, v) нужно провести только для первого шага.

4. Заключение

В данной статье описывается схема расширенного алгоритма Джебелеана–Вебера–Седжелмасы, позволяющая вычислять коэффициенты Безу и вычислять обратные элементы быстрее, чем в случае классического алгоритма Евклида. Сложность алгоритма Джебелеана–Вебера–Седжелмасы в худшем случае составляет $O(n^2/\log(n))$ для двух n битных чисел, когда сложность алгоритма Евклида в худшем случае составляет $O(n^2)$. Практический расчет при k , не превышающем размер машинного слова, реализуется быстрее против классического алгоритма Евклида. С другой стороны в случае очень больших чисел предпочтительнее использовать алгоритмы типа Шонхаге, работающие за $O(n \log^2(n) \log(\log(n)))$. Для небольших чисел можно использовать алгоритм Евклида или бинарный алгоритм. Текущие исследования автора связаны с обобщением данного алгоритма на многочлены для вычисления мультипликативного обратного для элементов поля классов вычетов по модулю некоторого неприводимого многочлена.

СПИСОК ЦИТИРОВАННОЙ ЛИТЕРАТУРЫ

1. Lehmer D. H. Euclid's algorithm for large numbers // American Mathematical Monthly. 1938. Vol. 45, №4. P. 227-233.
2. Schönhage A. Schnelle Berechnung von Kettenbruchentwicklungen // Acta Informat. 1971. Vol. 1, №2. P. 139-144.
3. Crandall R. D., Pomerance C. Prime Numbers. Springer-Verlag New York. 2005. P. 597.
4. Stein J., Computational problems associated with Racah algebra // J. Comp. Phys. 1967. Vol. 1, №3. P. 397-405.
5. Knuth D. E. The Art of Computer Programming: Seminumerical Algorithms, volume 2 // Addison-Wesley Professional, 3 edition. 1997. P. 784.
6. Sorenson J. An Analysis of Lehmer's Euclidean GCD Algorithm // Proc. of the International Symposium on Symbolic and Algebraic Computation. 1995. P. 254-258.

7. Cesari G. Parallel implementation of Schönhage's integer GCD algorithm // Algorithmic Number Theory. ANTS 1998. Lecture Notes in Computer Science. 1998. Vol. 1423. P. 64-76.
8. Stehle D., Zimmermann P. A Binary Recursive Gcd Algorithm. // Algorithmic Number Theory. ANTS 2004. Lecture Notes in Computer Science. 2004. Vol. 3076. P. 411-425.
9. Sorenson J. The k -ary gcd algorithm // Univ. of Wisconsin-Madison, Comp. Sciences Tech. Rep. 1990. №979. P. 1-20.
10. Sorenson J. Two Fast GCD Algorithms // J. of Algorithms. 1994. Vol. 16, №1. P. 110-144.
11. Dolgov D. Polynomial GCD as a solution of system of linear equations // Lobachevskii J Math. 2018. Vol. 39, №7.
12. Dolgov D. GCD calculation in the search task of pseudoprime and strong pseudoprime numbers // Lobachevskii J Math. 2016. Vol. 37, №6. P. 734-739.
13. Weber K. The Accelerated Integer GCD Algorithm // ACM Trans. Math. Software. 1995. Vol. 21, №1. P. 111-122.
14. Sedjelmaci S.M. Jebelean-Weber's algorithm without spurious factors // Information Processing Letters. 2007. Vol. 102, №6. P. 247-252.
15. Zimmermann P. 50 largest factors found by ECM // <https://members.loria.fr/PZimmermann/records/top50.html>
16. Сагалович Ю. Л. ВВЕДЕНИЕ В АЛГЕБРАИЧЕСКИЕ КОДЫ. ИППИ РАН. 2011. Р. 302.
17. Ishmukhametov S. T., Mubarakov B. G., Al-Anni K. M. Calculation of Bezout's Coefficients for the k -Ary Algorithm of Finding GCD // Russ Math. 2017. Vol. 61, №11. P. 26-33.
18. Бассалыго Л. А., Зиновьев В. А. Замечание об уравновешенных неполных блок-схемах, почти разрешимых блок-схемах и q -ичных равновесных кодах // Проблемы передачи информации. 2017. Т. 53, №1. С. 56-59.
19. Беспалов Е. А., Кротов Д. С. МДР-коды в графах Дуба // Проблемы передачи информации. 2017. Т. 53, №2. С. 40-59.

REFERENCES

1. Lehmer, D. H. 1938, "Euclid's algorithm for large numbers", *American Mathematical Monthly*, vol. 45, no. 4, pp. 227-233.
2. Schönhage, A. 1971, "Schnelle Berechnung von Kettenbruchentwicklungen", *Acta Informat*, vol. 1, no. 2, pp. 139-144.
3. Crandall, R., Pomerance, C. 2005, "Prime Numbers", *Springer-Verlag New York*, pp. 579.
4. Stein, J. 1967, "Computational problems associated with Racah algebra", *J. Comp. Phys.*, vol. 1, no. 3, pp. 397-405.
5. Knuth, D. E. 1997, "The Art of Computer Programming: Seminumerical Algorithms, volume 2", *Addison-Wesley Professional, 3rd edition*, pp. 784.
6. Sorenson, J. 1995, "An Analysis of Lehmer's Euclidean GCD Algorithm", *Proc. of the International Symposium on Symbolic and Algebraic Computation*, pp. 254-258.

7. Cesari, G. 1998, "Parallel implementation of Schönhage's integer GCD algorithm", *Algorithmic Number Theory. ANTS 1998. Lecture Notes in Computer Science*, vol. 1423, pp. 64-76.
8. Stehle, D., Zimmermann, P. 2004, "A Binary Recursive Gcd Algorithm", *Algorithmic Number Theory. ANTS 2004. Lect. Notes in Comp. Science*, vol. 3076, pp. 411-425.
9. Sorenson, J. 1990, "The k -ary gcd algorithm", *Univ. of Wisconsin-Madison, Comp. Sciences Tech. Rep.*, no. 979, pp. 1-20.
10. Sorenson, J. 1994, "Two Fast GCD Algorithms ", *J. of Algorithms*, vol. 16, no. 1, pp. 110-144.
11. Dolgov, D. 2018, "Polynomial GCD as a solution of system of linear equations", *Lobachevskii J Math*, vol. 39, no. 7.
12. Dolgov, D. 2016, "GCD calculation in the search task of pseudoprime and strong pseudoprime numbers", *Lobachevskii J Math*, vol. 37, no. 6, pp. 734-739.
13. Weber, K. 1995, "The Accelerated Integer GCD Algorithm", *ACMTrans.Math. Software*, vol. 21, no. 1, pp. 111-122.
14. Sedjelmaci, S. M. 2007, "Jebelean-Weber's algorithm without spurious factors", *Information Processing Letters*, vol. 102, no. 6, pp. 247-252.
15. Zimmermann, P. "50 largest factors found by ECM", <https://members.loria.fr/PZimmermann/records/top50.html>
16. Sagalovich, J. L. 2011, "INTRODUCTION TO ALGEBRAIC CODES [IN RUSSIAN]", *IITP RAS*, pp. 302.
17. Ishmukhametov, S. T., Mubarakov, B. G., Al-Anni, K. M. 2017, "Calculation of Bezout's Coefficients for the k -Ary Algorithm of Finding GCD", *Russ Math*, vol. 61, no. 11, pp. 26-33.
18. Bassalygo, L. A., Zinoviev, V. A., Al-Anni, K. M. 2017, "Remark on balanced incomplete block designs, near-resolvable block designs, and q -ary constant-weight codes", *Problems of Information Transmission*, vol. 53, no. 1, pp. 51-54.
19. Bespalov, E. A., Krotov, D. S., Al-Anni, K. M. 2017, "MDS codes in Doob graphs", *Problems of Information Transmission*, vol. 53, no. 2, pp. 136-154.

Получено 14.06.2018

Принято в печать 17.08.2018