УЧЕНЫЕ ЗАПИСКИ КАЗАНСКОГО УНИВЕРСИТЕТА.
СЕРИЯ ФИЗИКО-МАТЕМАТИЧЕСКИЕ НАУКИ

2019, Т. 161, кн. 1
С. 110–118

ISSN 2541-7746 (Print)
ISSN 2500-2198 (Online)

УДК 519.688

doi: 10.26907/2541-7746.2019.1.110-118

ОБ УСКОРЕНИИ k-АРНОГО АЛГОРИТМА ВЫЧИСЛЕНИЯ НОД НАТУРАЛЬНЫХ ЧИСЕЛ

И. Амер, Ш.Т. Ишмухаметов

Казанский (Приволжский) федеральный университет, г. Казань, 420008, Россия

Аннотация

В работе исследованы методы ускорения вычисления наибольшего общего делителя натуральных чисел на основе k-арного алгоритма, разработанного в 1990 г. Д. Соренсоном. Основная идея k-арного алгоритма состоит в вычислении для заданных натуральных чисел $A > B > 1$ пары целых чисел x и y таких, что выполнено соотношение $Ax + By \equiv 0 \bmod k$, где параметр k выбирается взаимно простым с A и B. Тогда $C = (Ax+By)/k$ примет значение, меньшее A, и следующая итерация выполняется с новой парой (B, C). k-арный алгоритм обеспечивает значительное уменьшение общего числа итераций по сравнению с классическим алгоритмом Евклида, однако общая производительность k-арного алгоритма проигрывает алгоритму Евклида.

Предложено использование заранее вычисленных таблиц параметров алгоритма и показано, что такой подход обеспечивает скорость, при которой k-арный алгоритм работает быстрее классического алгоритма Евклида.

Ключевые слова: наибольший общий делитель натуральных чисел, алгоритм Евклида, бинарный алгоритм, k-арный алгоритм

Введение

Задача вычисления наибольшего общего делителя (НОД) натуральных возникает при реализации многих математических и теоретико-числовых алгоритмов и их приложений. Например, в криптографии и теории чисел задача вычисления НОД появляется при разработке параметров криптографических методов шифрования, таких как RSA, El Gamal, Elliptic Curves Cyphering и др. (см. [1]). Вычисление обратных элементов в конечных полях также требует решения уравнения Безу $xA+yB = d$ в целых числах и реализуется с помощью расширенного алгоритма Евклида или соответствующего расширения k-арного алгоритма [2], поэтому задача ускорений вычисления НОД натуральных чисел является важной и актуальной задачей, имеющей многочисленные приложения. Цель настоящей работы – описать основные алгоритмы вычисления НОД и проанализировать их достоинства и недостатки.

1. Классический алгоритм Евклида

Пусть даны два натуральных числа A и B, $A > B > 1$. Требуется найти НОД d этих чисел. Классический алгоритм Евклида (КАЕ) представляет собой итерационную схему вычисления убывающих по величине пар (A_n, B_n), $n = 0, 1, 2, \ldots$ где $A_0 = A$, $B_0 = B$ – исходная пара чисел, а остальные пары вычисляются с помощью матричного равенства

$$\begin{pmatrix} A_{n+1} \\ B_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_n \end{pmatrix} \begin{pmatrix} A_n \\ B_n \end{pmatrix},$$

где $q_n = [A_n/B_n]$.

КАЕ имеет простую реализацию и широкое распространение. Главные недостатки этого алгоритма – это относительно медленная сходимость и невозможность распараллеливания (см. [5]). Сходимость КАЕ определяется значением коэффициента q_n. Однако среднее значение q_n, как отмечал еще Д. Кнут в своей фундаментальной монографии «Искусство программирования», в 70–80% итераций не превышает 3, что обеспечивает среднее число итераций, равное примерно бинарной длине меньшего из множества исходной пары (A, B) [4]. На каждой итерации КАЕ выполняется единственная операция вычисления остатка по модулю: $B_{n+1} = A_n \bmod B_n$.

2. Бинарный алгоритм вычисления НОД

Бинарный алгоритм вычисления НОД был разработан в 1967 г. Д. Стайном [3]. Предполагается, что исходные числа A и B являются нечетными. Если они таковыми не являются, то можно найти сначала наибольшую степень 2, на которую делятся оба числа, этот множитель войдет в НОД, затем сокращать каждое из чисел A и B на 2 до тех пор, пока каждое из них не станет нечетным.

Основная итерация бинарного алгоритма состоит в преобразовании $C_n = (A_n - B_n)/2$. Полученное C_n может оказаться четным числом, поэтому его следует сокращать на 2, пока оно не станет нечетным. Приведем пример вычисления НОД по бинарному алгоритму.

Пример 1. Даны числа $A = 183$ и $B = 172$. Сначала сократим B на степень двойки $172/4 = 43$. Затем начнем выполнять основное преобразование:

$$A_0 = 183, \quad B_0 = 43, \quad C_0 = (A_0 - B_0)/2 = (183 - 43)/2 = 70, \quad C := C/2 = 35.$$

$$A_1 = 43, \quad B_1 = 35, \quad C_1 = (43 - 35)/2 = 4, \quad C := C/4 = 1.$$

Вычисление закончено с ответом $\text{НОД}(183,\ 172) = 1$.

Число итераций в бинарном алгоритме примерно такое же, как и в КАЕ, однако выполнение одного шага итераций происходит заметно быстрее, чем в классическом алгоритме, так как вычисление C выполняется за линейное время относительно длины чисел A и B. Главным недостатком бинарного алгоритма является его плохая сходимость в случаях, когда значение B_n существенно меньше, чем A_n. В таких случаях значение C_n окажется больше B_n, и вычисление нового $C < B$ потребует нескольких итераций против одной в КАЕ. Поэтому в случае $A_n \gg B_n$ эффективнее выполнить одну итерацию классического алгоритма Евклида, после чего опять вернуться к выполнению бинарного алгоритма. Бинарный алгоритм также плохо поддается параллелизации. Д. Кнут в своей монографии [4] привел версию бинарного алгоритма примерно на 20%, превосходящего по скорости алгоритм Евклида.

Рассмотрим в следующем разделе k-арный алгоритм, являющийся обобщением бинарного алгоритма.

3. k-арный алгоритм вычисления НОД

3.1. Общее описание алгоритма. k-арный алгоритм вычисления НОД был разработан одновременно и независимо в начале 90-х годов XX в. американским математиком Дж. Соренсоном [6, 7] и австрийцем Т. Джебелеаном [8]. Алгоритм состоит из отдельных итераций. На произвольном шаге алгоритма на вход подаются нечетные числа A, B и $k = 2^{2s}$, $s > 1$, и выполняется поиск целых чисел

x и y таких, что $Ax + By \equiv 0 \bmod k$. Когда соответствующие x и y найдены, вычисляется множество $C = (Ax + By)/k$, проверяется, является ли оно четным, и C сокращается на 2, пока не станет нечетным. После этого переходят к новой паре (B, C). Вычисление повторяется до тех пор, пока очередное C_n не станет меньше k, после чего выполняется последняя итерация с использование алгоритма Евклида и исходных чисел

$$d = KAE(B, KAE(A, B_n)).$$

Последнее действие необходимо потому, что при вычислении C в нем может появиться дополнительный ложный фактор, не входящий в НОД исходных чисел A и B.

Сходимость алгоритма обеспечивается следующей теоремой Соренсона [6].

Теорема 1. *Для любых натуральных чисел A, B и $k > 1$, взаимно простого с A и B, найдутся целые числа x и y, $1 \le x \le s$, $-s \le y \le s$, где $s = \sqrt{k}$ такие, что*

$$Ax + By \equiv 0 \bmod k. \tag{1}$$

Непосредственно из теоремы следует, что значение A/C всегда меньше $2A/\sqrt{k}$, что обеспечивает быструю сходимость алгоритма при значениях k, сравнимых с размером машинного регистра. Рассмотрим, как можно найти подходящие значения x и y.

3.2. Поиск подходящих x и y перебором. Из условия (1) следует, что $y \equiv -qx \bmod k$, где $q = AB^{-1} \bmod k$. Искомую пару x и y можно искать перебором, рассматривая последовательно значения x от 1 до $\sqrt{k}$ и вычисляя отрицательное и положительное значения $y = -(qx \bmod k)$ и $y = k - (qx \bmod k)$, до тех пор, пока не будет найдено значение y из интервала $[-\sqrt{k}; \sqrt{k}]$. Такой алгоритм имеет асимптотическую оценку $O(\sqrt{k})$. Если значение k достаточно велико, то вычисление параметра q и поиск пары x и y займут существенную часть общего времени работы алгоритма. Рассмотрим более быстрый алгоритм поиска пары (x, y), основанный на рядах Фарея.

3.3. Поиск подходящих x и y с помощью рядов Фарея. Выполним поиск возможных значений q для заданных значений x. Разобьем работу алгоритма на несколько ветвей в зависимости от значений переменной x.

1. Пусть $x = 1$. Тогда $y = -q$, и из условия $-\sqrt{k} \le y \le \sqrt{k}$ получим, что $1 \le y \le \sqrt{k}$ или $k - \sqrt{k} \le y < k$. Пара (x, y) примет значение $(1, -q)$ в первом случае и $(1, k - q)$ во втором.

2. Пусть $x = 2$. В этом случае выполнится неравенство $(k - \sqrt{k})/2 \le y \le (k + \sqrt{k})/2$ и $(x, y) = (2, k - 2q)$ или $(x, y) = (2, 2q - k)$ в зависимости от того, в какой половине интервала $[(k - \sqrt{k})/2; (k + \sqrt{k})/2]$ находится q.

3. Пусть теперь x принимает произвольное значение $x = n$ из интервала $[1; \sqrt{k}]$. Решая неравенство $-\sqrt{k} \le -qn \bmod k \le \sqrt{k}$, получим область значений переменной q, состоящую из $n-1$ подынтервала длины $2\sqrt{k}/n$ с центрами в точках mk/n, $m = 1, 2, \ldots, n-1$. Например, при $x = 3$ получим два интервала значений q длины $2\sqrt{k}/3$ с центрами в точках $k/3$ и $2k/3$.

Таким образом, согласно вышесказанному имеем, что получить значение x можно, находя ближайшую к q дробь вида mk/n, где $0 < m < n < k$, и полагая значение x равным n. Если обозначить дробь q/k через α, то наша задача эквивалентна нахождению для заданного действительного числа α из интервала

$(0;\,1)$ правильной дроби m/n с числителем и знаменателем, меньшими параметра k, аппроксимирующей наиболее точно значение α.

Подходящую дробь найдем, используя ряды Фарея (Farey Series). Алгоритм нахождения подходящей дроби подробно описан в [9]. Дадим здесь его краткое описание.

1. Разобьем интервал $(0;\,1)$ на подынтервалы точками

$$\frac{1}{k-1},\quad \frac{1}{k-2},\ldots,\frac{1}{2},\quad \frac{2}{3},\quad \frac{3}{4},\quad \ldots,\quad \frac{k-2}{k-1}.$$

2. Найдем интервал $(m_1/n_1;\,m_2/n_2)$, содержащий значение α.

3. Вычислим медиану $m/n=(m_1+m_2)/(n_1+n_2)$ и проверим, находится α левее или правее этой медианы. Если выполнится $\alpha<m$, то определим уточненный интервал, содержащий α, равным $(m_1/n_1;\,m/n)$, иначе равным $(m/n;\,m_2/n_2)$.

4. Будем повторять шаги 2–3 до тех пор, пока значение медианы остается меньшим k.

Пример 2. Пусть $k=16$ и $q=45$. Тогда $\alpha=q/k\approx 0.703$. Выберем интервал, содержащий α: $\alpha\in(2/3;\,3/4)$. Строя медианы, получим цепочку интервалов, содержащих α:

$$\left(\frac{2}{3};\frac{3}{4}\right)\to\left(\frac{2}{3};\frac{5}{7}\right)\to\left(\frac{7}{10};\frac{5}{7}\right).$$

Процесс закончен, так как знаменатель следующей медианы превысит значение $k-1=15$. Наиболее точным приближением к $\alpha=0.703$ является дробь $7/10$.

Сформулируем полученный результат в виде следующей теоремы.

Теорема 2. *Для заданных натуральных чисел A, B и k поиск коэффициентов x и y, удовлетворяющих условию* (1) *теоремы Соренсона, может быть выполнен за время $O(\log_2 k)$.*

Доказательство следует из того факта, что построение новой медианы уменьшает оценку расстояния от ближайшей границы интервала до α вдвое.

3.4. Ускорение алгоритма путем использования предварительно вычисленных таблиц. Для ускорения работы можно выполнить предварительное вычисление значений пар (x,y) для каждого значения $q<k$. Эта таблица должна содержать k значений (по два значения для каждого нечетного числа, меньшего k). При k, сравнимым по размеру с 32-битовым машинным словом, таблица должна содержать $2^{32}\approx 4.3\cdot 10^9$ строк. Однако вычисление этой таблицы не займет много времени, а сама таблица может храниться в текстовом файле. Поскольку значение y может быть вычислено по значениям x, q и k, то достаточно хранить только пары (q,x).

Кроме того, отметим, что, поскольку пара (x,y), соответствующая $k-q$, отличается от соответствующей пары для q только знаком y, можно хранить только половину таблицу для $q<k/2$, а для значений $q>k/2$ брать пару $(x,-y)$, взяв значения (x,y) из таблицы для $k-q$.

Полезно также иметь заранее вычисленную таблицу обратных элементов по модулю k. Не обязательно при этом вычислять заранее всю таблицу обратных элементов. Так как $k=2^s$ представляет собой степень простого числа, существует простой алгоритм, позволяющий по значению обратного элемента по модулю 2^s найти значение обратного элемента по модулю 2^{s+1}. Этот алгоритм носит название в теории чисел «подъем Гензеля» (Hensel's Lifting). Его описание можно найти в статье [2].

Как и в случае бинарного алгоритма, единичная итерация k-арного алгоритма становится неэффективной, если длина A значительно больше длины B. В этом случае полезно использовать операцию dmod, введенную Вебером [10], которая по сути является единичной итерацией классического алгоритма Евклида и позволяет ускорить общее выполнение k-арного алгоритма.

В общем случае эффективность k-арного алгоритма незначительно лучше эффективности классического алгоритма. Следует отметить, что при увеличении значения k и размера пары A, B эффективность алгоритма существенно улучшается. Параллелизация этого алгоритма также невозможна.

3.5. Статистические данные скорости вычисления НОД и числа итераций. Нами были проведена серия вычислений НОД пар натуральных чисел длины 100 десятичных разрядов с использованием классического и k-арного алгоритмов вычисления НОД и собрана статистика по суммарному времени вычисления для 20 пар чисел. Подсчитывалось также среднее число итераций по серии из 20 чисел. Результаты приведены в следующих таблицах. Первый столбец содержит данные для алгоритма Евклида, а оставшиеся – данные для k-арного алгоритма для различных k.

Из табл. 1 видно, что по числу итераций классический алгоритм проигрывает k-арному, но выигрывает по времени. Эффективность k-арного алгоритма возрастает с ростом k.

Табл. 1

Время и число итераций КАЕ и k-арного алгоритма

	КАЕ	$k = 16$	$k = 256$	$k = 2^{16}$
N_{iter}	194	151	99	63
Время, мс	17	40	25	22

В табл. 2 мы собрали данные по вычислению НОД с использованием включения итераций классического алгоритма Евклида. Это включение было предложено Вебером [10] и выполняется в случае, когда число A в паре (A, B) значительно превышает B. В нашем вычислении эта операция применена в случае $A/B > k$.

Табл. 2

Время и число итераций при использовании dmod

	КАЕ	$k = 16$	$k = 256$	$k = 2^{16}$
N_{iter}	194	128	96	64
Время, мс	17	31	19	16

Мы видим, что по числу итераций и по времени k-арный алгоритм с модификацией Вебера немного выиграл по сравнению с алгоритмом Соренсона, и общее время выполнения стало сравнимым со временем работы классического алгоритма

Очевидно, что большая часть времени тратится на вычисление обратных по модулю k элементов при вычислении $q = AB^{-1} \bmod k$. Приведем данные времени вычисления (см. табл. 3), используя две предварительно вычисленные таблицы: таблицу обратных значений по модулю k и таблицу значений пар (x, y) для заданных значений k (число итераций в этом случае не меняется). В первой строчке приведены данные из табл. 2. Во второй строчке выполнены вычисление, использующее таблицы обратных элементов по модулю k. В третьей строчке используется таблица пар (x, y), вычисленная для каждого значения $q < k$.

Табл. 3
Время вычисления без и при использовании предварительно вычисленных таблиц

	КАЕ	$k = 16$	$k = 256$	$k = 2^{16}$
Время 1, мс	17	31	19	16
Время 2, мс	17	22	16	14
Время 3, см	17	20	15	10

Табл. 4
Время вычисления для пар чисел длины 200 десятичных разрядов

	КАЕ	$k = 16$	$k = 256$	$k = 2^{16}$
Время 1, мс	36	91	59	48
Время 2, мс	36	52	41	36
Время 3, мс	36	46	32	27

Табл. 5
Время вычисления для пар чисел длины 500 десятичных разрядов

	КАЕ	$k = 16$	$k = 256$	$k = 2^{16}$
Время 1, мс	108	271	187	149
Время 2, мс	108	189	154	134
Время 3, мс	108	171	137	88

Повторим последнее вычисление, увеличив длину входных пар до 200 десятичных разрядов (см. табл. 4). Для проверки сходимости приведем еще данные по сравнению скорости алгоритмов для чисел длины 500 десятичных разрядов (табл. 5).

Таким образом, длина входных данных не сильно влияет на соотношение скоростей выполнения классического и k-арного алгоритмов. Наиболее эффективными факторами ускорения k-арного алгоритма являются увеличение значения параметра k и использование таблиц обратных элементов.

Заключение

Вычисление НОД по классическому алгоритму Евклида является быстрым и эффективным даже для пар чисел большой длины, используемых в криптографии. Однако некоторые задачи такие, например, как вычисление функции распределения гладких чисел [11] или поиск строго псевдопростых чисел для улучшения теста простоты Миллера – Рабина [12] требуют вычисления НОД для огромного числа пар целых чисел, где использование k-арного алгоритма вполне оправдано и позволяет добиться существенной экономии времени вычисления.

Благодарности. Работа выполнена за счет средств субсидии, выделенной Казанскому федеральному университету для выполнения государственного задания в сфере научной деятельности, проект № 1.13556.2019/13.1. Работа также частично поддержана РФФИ (проект № 18-47-16005).

Литература

1. *Ишмухаметов Ш.Т.* Методы факторизации натуральных чисел .– Казань: Изд-во Казан. ун-та, 2011. – 189 с.

2. *Ишмухаметов Ш.Т., Маад К.А., Мубараков Б.Г.* Вычисление коэффициентов Безу для k-арного алгоритма нахождения НОД // Изв. вузов. Матем. – 2017. – № 11. – С. 30–38.

3. *Stein J.* Computational problems associated with Racah algebra // J. Comput. Phys. – 1967.– V. 1, No 3. – P. 397–405. – doi: 10.1016/0021-9991(67)90047-2.

4. *Knuth D.* The Art of Computer Programming. V. 2: Seminumerical Algorithms. – Addison-Wesley, 1997. – xiv, 762 p.

5. *Ishmukhametov S.T. Rubtsova R.G.* A parallel computation of the GCD of natural numbers // Сб. тр. «Параллельные вычислительные технологии – XI международная конференция», ПаВТ'2017. – Казань, 2017. – P. 120–129.

6. *Sorenson J.* The k-ary GCD Algorithm: Computer Sciences Technical Report CS-TR-90-979. – Madison: Univ. of Wisconsin, 1990. – 20 p.

7. *Sorenson J.* Two fast GCD Algorithms // J. Algorithms. – 1994. – V. 16, No 1. – P. 110–144. – doi: 10.1006/jagm.1994.1006.

8. *Jebelean T.* A generalization of the binary GCD algorithm // Proc. Int. Symp. on Symbolic and Algebraic Computation, SSAC'93. – N. Y.: ACM, 1993. – P. 111–116. – doi: 10.1145/164081.164102.

9. *Ishmukhametov S.T.* An approximating k-ary GCD algorithm // Lobachevskii J. Math. – 2016. – V. 37, No 6. – P. 723–729. – doi: 10.1134/S1995080216060147.

10. *Weber K.* The accelerated integer GCD algorithm // ACM Trans. Math. Software. – 1995. – V. 21, No 1. – P. 111–122. – doi: 10.1145/200979.201042.

11. *Maximov K.L., Ishmukhametov S.T.* About algorithm of smooth numbers calculation // Res. J. Appl. Sci. – 2015. – V. 10, No 8. – P. 376–380.

12. *Ishmukhametov S.T., Mubarakov B.G.* On practical aspects of the Miller–Rabin Primality Test // Lobachevskii J. Math. – 2013. – V. 34, No 4. – P. 304–312. – doi: 10.1134/S1995080213040100.

Поступила в редакцию
15.06.18

Амер Исмаил, стажер-исследователь кафедры системного анализа и информационных технологий

Казанский (Приволжский) федеральный университет
ул. Кремлевская, д. 18, г. Казань, 420008, Россия
E-mail: *safadi121979@yahoo.com*

Ишмухаметов Шамиль Талгатович, доктор физико-математических наук, профессор кафедры системного анализа и информационных технологий

Казанский (Приволжский) федеральный университет
ул. Кремлевская, д. 18, г. Казань, 420008, Россия
E-mail: *sishmukh@kpfu.ru*

ISSN 2541-7746 (Print)
ISSN 2500-2198 (Online)

**UCHENYE ZAPISKI KAZANSKOGO UNIVERSITETA.
SERIYA FIZIKO-MATEMATICHESKIE NAUKI
(Proceedings of Kazan University. Physics and Mathematics Series)**

2019, vol. 161, no. 1, pp. 110–118

doi: 10.26907/2541-7746.2019.1.110-118

On Acceleration of the k-ary GCD Algorithm

I. Amer, S.T. Ishmukhametov***

Kazan Federal University, Kazan, 420008 Russia
E-mail: **safadi121979@yahoo.com*, ***sishmukh@kpfu.ru*

Received June 15, 2018

Abstract

In this paper, methods of acceleration of GCD algorithms for natural numbers based on the k-ary GCD algorithm have been studied. The k-ary algorithm was elaborated by J. Sorenson in 1990. Its main idea is to find for given numbers A, B and a parameter k, co-prime to both A and B, integers x and y satisfying the equation $Ax + By \equiv 0 \bmod k$. Then, integer $C = (Ax+By)/k$ takes a value less than A. At the next iteration, a new pair (B, C) is formed. The k-ary GCD algorithm ensures a significant diminishing of the number of iterations against the classical Euclidian scheme, but the common productivity of the k-ary algorithm is less than the Euclidian method.

We have suggested a method of acceleration for the k-ary algorithm based on application of preliminary calculated tables of parameters like as inverse by module k. We have shown that the k-ary GCD algorithm overcomes the classical Euclidian algorithm at a sufficiently large k when such tables are used.

Keywords: greatest common divisor for natural numbers, Euclidian GCD algorithm, binary GCD algorithm, k-ary GCD algorithm

Acknowledgments. This work was funded by the subsidy allocated to Kazan Federal University for the state assignment in the sphere of scientific activities, project no. 1.13556.2019/13.1. The study was also supported in part by the Russian Foundation for Basic Research (project no. 18-47-16005).

References

1. Ishmukhametov Sh.T. *Metody faktorizatsii natural'nykh chisel* [Methods of Factorization of Positive Integers]. Kazan, Izd. Kazan. Univ., 2011. 189 p. (In Russian)
2. Ishmukhametov Sh.T., Mubarakov B.G., Maad Kamal Al-Anni Calculation of Bezout's coefficients for the k-ary algorithm of finding GCD. *Russ. Math.*, 2017, vol. 61, no. 11, pp. 26–33. doi: 10.3103/S1066369X17110044.
3. Stein J. Computational problems associated with Racah algebra. *J. Comput. Phys.*, 1967, vol. 1, no. 3, pp. 397–405. doi: 10.1016/0021-9991(67)90047-2.
4. Knuth D. *The Art of Computer Programming.* Vol. 2: *Seminumerical Algorithms.* Addison-Wesley, 1997. xiv, 762 p.

5. Ishmukhametov S.T. Rubtsova R.G. A parallel computation of the GCD of natural numbers. *Parallel'nye vychislitel'nye tekhnologii – XI mezhdunarodnaya konferentsiya, PaVT' 2017* [Parallel Computational Technologies: XI Int. Conf., PCT' 2017]. Kazan, 2017, pp. 120–129. (In Russian)

6. Sorenson J. *The k-ary GCD Algorithm: Computer Sciences Technical Report CS-TR-90-979*. Madison, Univ. of Wisconsin, 1990. 20 p.

7. Sorenson J. Two fast GCD algorithms. *J. Algorithms*, 1994, vol. 16, no. 1, pp. 110–144. doi: 10.1006/jagm.1994.1006.

8. Jebelean T. A generalization of the binary GCD algorithm. *Proc. Int. Symp. on Symbolic and Algebraic Computation, SSAC'93*. New York, ACM, 1993, pp. 111–116. doi: 10.1145/164081.164102.

9. Ishmukhametov S.T. An approximating k-ary GCD algorithm. *Lobachevskii J. Math.*, 2016, vol. 37, no. 6, pp. 723–729. doi: 10.1134/S1995080216060147.

10. Weber K. The accelerated integer GCD algorithm. *ACM Trans. Math. Software*, 1995, vol. 21, no. 1, pp. 111–122. doi: 10.1145/200979.201042.

11. Maximov K.L., Ishmukhametov S.T. About algorithm of smooth numbers calculation. *Res. J. Appl. Sci.*, 2015, vol. 10, no. 8, pp. 376–380.

12. Ishmukhametov S.T., Mubarakov B.G. On practical aspects of the Miller–Rabin Primality Test. *Lobachevskii J. Math.*, 2013, vol. 34, no. 4, pp. 304–312. doi: 10.1134/S1995080213040100.

⟨ ***Для цитирования:*** *Амер И., Ишмухаметов Ш.Т.* Об ускорении k-арного алгоритма вычисления НОД натуральных чисел // Учен. зап. Казан. ун-та. Сер. Физ.-матем. науки. – 2019. – Т. 161, кн. 1. – С. 110–118. – doi: 10.26907/2541-7746.2019.1.110-118. ⟩

⟨ ***For citation:*** Amer I., Ishmukhametov S.T. On acceleration of the k-ary GCD algorithm. *Uchenye Zapiski Kazanskogo Universiteta. Seriya Fiziko-Matematicheskie Nauki*, 2019, vol. 161, no. 1, pp. 110–118. doi: 10.26907/2541-7746.2019.1.110-118. (In Russian) ⟩