Анализ данных и интеллектуальные системы

DOI: 10.25586/RNU.V9187.21.01.P.150 УДК 681.3

И.Ф. Амер, А.М. Аль Халиди

ДВА БЫСТРЫХ МЕТОДА НАХОЖДЕНИЯ НАИБОЛЬШЕГО ОБЩЕГО ДЕЛИТЕЛЯ

Рассмотрены новые высокопроизводительные алгоритмы нахождения наибольшего общего делителя (НОД), разработанные на основе вавилонской системы исчисления и базиса Крестенсона. Произведены оценки вычислительной сложности основных операций усовершенствованного алгоритма поиска НОД в базисе Крестенсона. Представлены результаты сравнительного анализа усовершенствованного алгоритма поиска НОД в базисе Крестенсона со стандартным алгоритмом Евклида в вавилонской системе счисления и алгоритмом поиска НОД с помощью базиса Крестенсона.
Ключевые слова: наибольший общий делитель, теория чисел, алгоритм Евклида, вычислительная сложность, метод нахождения.

I.F. Amer, A.M. Al Khalidi

TWO FAST METHODS FOR FINDING THE GREATEST COMMON DIVISOR

New high-performance algorithms for finding the greatest common divisor (GCD), developed on the basis of the Babylonian calculus and the Chrestenson basis, are considered. Estimates of the computational complexity of the basic operations of the improved GCD search algorithm in the Chrestenson basis are made. The results of a comparative analysis of the improved algorithm for finding GCD in the Chrestenson basis with the standard Euclidean algorithm in the Babylonian number system and the algorithm for searching for GCD using the Chrestenson basis are presented.
Keywords: the greatest common divisor, number theory, Euclidean algorithm, computational complexity, method of finding.

Актуальность работы

Наиболее распространенным методом нахождения НОД является один из древнейших математических алгоритмов – алгоритм Евклида, согласно которому для нахождения НОД двух чисел необходимо несколько раз от большего числа отнять меньшее, пока разница не станет меньше вычитателя [8–10]. Тогда эту же самую процедуру требуется выполнить с вычитаемым и разностью. Процесс вычитания будет продолжаться до тех пор, пока вычитаемое и разность не станут одинаковы. Поскольку числа, над которыми выполняются операции, на каждом шаге уменьшаются, то такой процесс не может продолжаться бесконечно, а закончится через некоторое число шагов [2–5].

Современная математическая запись алгоритма Евклида имеет следующий вид [1, 6, 7]: для любого $a > b = r_0$, где a и b – целые числа, выполняется система уравнений

Два быстрых метода нахождения наибольшего общего делителя

$$\begin{gathered} a = r_0 \times q_1 + r_1,\ 0 \le r_1 < r_0; \\ r_0 = r_1 \times q_2 + r_2,\ 0 \le r_2 < r_1 \\ r_0 = r_1 \times q_2 + r_2,\ 0 \le r_2 < r_1 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ r_{n-2} = r_{n-1} \times q_n + r_n,\ 0 \le r_n < r_{n-1}; \\ r_{n-1} = r_n \times q_{n+1} + 0. \end{gathered} \qquad (1)$$

НОД (a, b) равен r_n, то есть последнему ненулевому члену последовательности r_i. Поскольку $r_0 > r_1 > r_2 \ldots > r_n > 0$, то данный процесс закончится минимум через b шагов, то есть нужно выполнить b делений с остатком. Однако данная оценка является неудовлетворительной, поскольку для нахождения НОД двух чисел (или проверки на простоту), которые используются в современных асимметричных криптосистемах, требуется очень большой объем времени. В [8] показано, что для нахождения НОД с помощью алгоритма Евклида нужно выполнить не более чем $5k$ операций деления с остатком, где k – количество цифр в десятичной записи числа a. В [9] показано, что количество шагов не превышает $2\log_2 b + 1$. Иная оценка следует из теоремы Ламе [10], согласно которой количество шагов алгоритма Евклида не превышает $\lfloor \log\phi(\sqrt{5}a) \rfloor - 2$, где $[a]$ – верхнее целое a; $\Phi = (1+\sqrt{5})/2$ – больший корень характеристического уравнения последовательности Фибоначчи.

Несмотря на указанные оценки и простоту программной реализации алгоритма Евклида, его вычислительная сложность остается большой, так как операция деления является достаточно трудоемкой. Расчеты показывают, что поиск НОД с использованием алгоритма Евклида в базисе Радемахера [8] характеризуется вычислительной сложностью по меньшей мере $17{,}5 \times n(n+1)^2$, где n – разрядность числа a в двоичном коде.

Другим недостатком алгоритма Евклида является последовательное выполнение операции деления одна за другой, то есть невозможность его распараллеливания.

Перспективным направлением совершенствования исследуемых алгоритмов является разработка теории их реализации на основе вавилонской системы счисления остаточных классов.

Цель работы

Разработать новые высокопроизводительные алгоритмы нахождения НОД с помощью вавилонской системы исчисления и базиса Крестенсона, которые характеризуются значительно меньшей вычислительной сложностью, чем стандартный алгоритм Евклида.

Алгоритм Евклида в вавилонской системе счисления. Пусть требуется найти НОД чисел a и b:

$$a = a_{n-1}2^{n-1} + \ldots + a_i 2^i + \ldots + a_1 2 + a_0; \qquad (2)$$

$$b = b_{n-1}2^{n-1} + \ldots + b_i 2^i + \ldots + b_1 2 + b_0, \qquad (3)$$

причем $a > b = r_0$; $a, b = 0, 1$.

Исходя из (1)

$$r_1 = a \bmod b = (a_{(n-1)}2^{(n-1)} + \ldots + a_i 2^i + \ldots + a_1 2 + a_0) \bmod b = \\ = \left(\sum_{i=0}^{n-1} (a_i 2^i \bmod b) \right) \bmod b = \left(\sum_{i=0}^{n-1} (a_i r_{1i}) \right) \bmod b,$$

где $r_{1i} = 2^i \bmod b$.

Это означает, что искомый остаток будет равен сумме трех степеней двойки, для которых $a_i = 1$.

Следует отметить также, что два последовательных значения – r_{1i} и r_{1i+1} – связаны рекуррентным соотношением $r_{1i+1} = (2 \times r_{1i}) \bmod b$.

Для нахождения остатка по модулю b необязательно выполнять деление с остатком, а можно ограничиться вычитанием: если $r_{1i+1} < b$, то оно остается неизменным, в противном случае $r_{1i+1} = r_{1i+1} - b$. Проще всего реализовать описанный шаг алгоритма Евклида в вавилонской системе счисления с помощью таблицы 1.

Таблица 1

Нахождение остатка $a \bmod b$

a_{n-1}	a_{n-2}	…	a_i	…	a_2	a_1	a_0
r_{1n-1}	r_{1n-2}	…	r_{1i}	…	r_{12}	r_{11}	r_{10}

В соответствии с таблицей 1 r_1 ищем как сумму r_{1i} по модулю b, над которыми в верхней строке размещена 1, то есть $r_1 = \left(\sum_{i=0}^{n-1} r_{1i}\right) \bmod b$ при условии $a_i = 1$.

Аналогично строим таблицу 2.

Таблица 2

Нахождение остатка $b \bmod r_1$

$b_{n-1} = r_{0n-1}$	$b_{n-2} = r_{0n-2}$	…	$b_i = r_{0i}$	…	$b_2 = r_{02}$	$b_1 = r_{01}$	$b_0 = r_{00}$
r_{2n-1}	r_{2n-2}	…	r_{2i}	…	r_{22}	r_{21}	r_{20}

Соответственно, $r_2 = \left(\sum_{i=0}^{n-1} r_{2i}\right) \bmod r_1$ при условии $b_i = 1$.

Обобщая полученные результаты, запишем выражение для нахождения любой

$$r_j = \left(\sum_{i=1}^{n-1} r_{j-2i} r_{ji}\right) \bmod r_{j-1},$$

где $r_{j-2i} = 0,1$; $r_{ji} = 2^i mj = mod r_{i-1}$.

Отметим, что количество шагов стандартного алгоритма Евклида и алгоритма Евклида в вавилонской системе счисления одинаковое. Однако вычислительная сложность выполнения каждого шага существенно уменьшается и составляет $\log_2 n$. Общая вычислительная сложность алгоритма Евклида в вавилонской системе счисления оценивается выражением $O(17{,}5n(\log_2 n))$. Кроме того, он исключает возможность распараллеливания.

Алгоритм поиска НОД с помощью базиса Крестенсона

Предложен алгоритм нахождения НОД, основанный на поиске остатков от деления чисел a и b $(a > b)$ в вавилонской системе счисления (2), (3) на все простые числа $\sqrt{b}$.

Общим делителем чисел a и b будет модуль p_j^k, который находим из условия

$$\left(\sum_{i=0}^{n-1} a_{ij}\right) \bmod p_j^k = \left(\sum_{i=0}^{n-1} b_{ij}\right) \bmod p_j^k = 0, \qquad (4)$$

где $a_{ij} = a_i \times 2^i \bmod p^k{}_j$; $b_{ij} = b_i \times 2^i \bmod p_j^k$; p_j – простое число, меньшее $\sqrt{b}$; $k = 1, 2, 3, \ldots$ – степень p_j.

Следует отметить, что при $k = 1$ и выполнении (4) проверяется условие при $k = 2$, и так далее. Таким образом, учитывается общий делитель, который является степенью простого числа. Искомый наибольший общий делитель находим как произведение полученных с помощью (4) общих делителей.

Предложенный алгоритм характеризуется логарифмической вычислительной

$$O(m \times \log_2 n),$$

где $m = \int_2^{\sqrt{b}} \frac{dt}{\ln t}$ – количество простых чисел в диапазоне от 2 до сложностью $\sqrt{b}$.

Кроме того, он позволяет распараллелить выполнение всех операций по каждому модулю и выполнить факторизацию чисел a и b.

Усовершенствованный алгоритм поиска НОД с помощью базиса Крестенсона. Предложенный выше алгоритм можно существенно усовершенствовать путем сокращения количества модулей (то есть количества шагов), по которым нужно искать остатки. Пусть имеем систему модулей, которая состоит из простых чисел $p_1, p_2, p_3, ...$, меньших $\sqrt{b}$, и для некоторого p^k_j выполняется условие (4). Следующий шаг заключается в последовательной проверке условия (4) для модуля $(p^k_j p^k_{j+1})$, $k_1 = 1, 2, 3,...; i = 1, 2, ...$

Таким образом, при последовательном умножении модулей получаем, что НОД $(a,b) = \prod_{j=1}^{s} p^k_j$, для которых выполняется условие (4).

По сравнению с предыдущим данный алгоритм использует меньшее количество шагов, однако он не поддается распараллеливанию и не решает задачу факторизации чисел.

Примеры применения алгоритмов поиска НОД. Пусть нужно вычислить НОД (3843, 1449).

1. Стандартный алгоритм Евклида:

3843 = 1449·1+945

1449 = 945·1+504

945 = 504·1+441

504 = 441·1+63

441 = 63·7+0.

Следовательно, НОД (3843, 1449) = 63.

Алгоритм Евклида в вавилонской системе счисления

Данный алгоритм удобно представить в виде таблицы 3.

Таблица 3

Алгоритм Евклида в вавилонской системе счисления

1	3843	1	1	1	1	0	0	0	0	0	0	1	1
2	2^imod1449	599	1024	512	256	128	64	32	16	8	4	2	1
3	1449		1	0	1	1	0	1	0	1	0	0	1
4	2^imod945		79	512	256	128	64	32	16	8	4	2	1
5	945			1	1	1	0	1	1	0	0	0	1
6	2^imod504			8	256	128	64	32	16	8	4	2	1
7	504				1	1	1	1	1	1	0	0	0
8	2^imod441				256	128	64	32	16	8	4	2	1
9	441				1	1	0	1	1	1	0	0	1
10	2^imod63				4	2	1	32	16	8	4	2	1

Строка 2: (599+1024+512+256+2+1)mod1449 = 945.
Строка 4: (79+256+128+32+8+1)mod945 = 504.
Строка 6: (8+256+128+32+16+1)mod504 = 441.
Строка 8: (256+128+64+32+16+8)mod441 = 63.
Строка 10: (4+2+32+16+8+1)mod63 = 0.
Таким образом можно получить НОД, избежав громоздкой операции деления.

Поиск НОД в базисе Крестенсона. Данную задачу также удобно представить в виде таблицы 4, учитывая, что $\sqrt{14449} \approx 38$.

Таблица 4

Нахождение остатков за простыми модулями

1		2048	1024	512	256	128	64	32	16	8	4	2	1
2	3843	1	1	1	1	0	0	0	0	0	0	1	1
3	1449		1	0	1	1	0	1	0	1	0	0	1
4	2^imod2	0	0	0	0	0	0	0	0	0	0	0	1
5	2^imod3	2	1	2	1	2	1	2	1	2	1	2	1
6	2^imod5	3	4	2	1	3	4	2	1	3	4	2	1
7	2^imod7	4	2	1	4	2	1	4	2	1	4	2	1
8	2^imod11	2	1	6	3	7	9	10	5	8	4	2	1
9	2^imod13	7	10	5	9	11	12	6	3	8	4	2	1
10	2^imod17	8	4	2	1	9	13	15	16	8	4	2	1
11	2^imod19	15	17	18	9	14	7	13	16	8	4	2	1
12	2^imod23	1	12	6	3	13	18	9	16	8	4	2	1
13	2^imod29	18	9	19	24	12	6	3	16	8	4	2	1
14	2^imod31	2	1	16	8	4	2	1	16	8	4	2	1
15	2^imod37	13	25	31	34	17	27	32	16	8	4	2	1
16	2^imod9	5	7	8	4	2	1	5	7	8	4	2	1
17	2^imod27	23	25	26	13	20	10	5	16	8	4	2	1

Из таблицы 4 ищем остатки за простыми модулями.
Строка 4: 3843mod2 = 1; 1449mod2 = 1.
Строка 5: 3843mod3 = (2+1+2+1+2+1)mod3 = 0;
1449mod3 = (1+1+2+2+2+1)mod3 = 0.
Строка 6: 3843mod5 = (3+4+2+1+2+1)mod5 = 3;
1449mod5 = (4+1+3+2+3+1)mod5 = 4.
Строка 7: 3843mod7 = (4+2+1+4+2+1)mod7 = 0;
1449mod7 = (2+4+2+4+1+1)mod7 = 0.
Строка 8: 3843mod1 = (2+1+6+3+2+1)mod11 = 4;
1449mod11 = (1+3+7+10+8+1)mod11 = 8.
Строка 9: 3843mod13 = (7+10+5+9+2+1)mod13 = 8;
1449mod13 = (10+9+11+6+8+1)mod13 = 6.
Строка 10: 3843mod17 = (8+4+2+1+2+1)mod17 = 1;
1449mod17 = (4+1+9+15+8+1)mod17 = 4.

Строка 11: 3843 mod19 = (15+17+18+9+2+1)mod19 = 5;
1449mod19 = (17+9+14+13+8+1)mod19 = 5.
Строка 12: 384mod23 = (1+12+6+3+2+1)mod23 = 2;
1449mod23 = (12+3+13+9+8+1)mod23 = 0.
Строка 13: 3843mod29 = (18+9+19+24+2+1)mod29 = 15;
1449mod29 = (9+24+12+3+8+1)mod29 = 28.
Строка 14: 3843mod31 = (2+1+16+8+2+1)mod31 = 30;
1449mod31 = (1+8+4+1+8+1)mod31 = 23.
Строка 15: 3843mod37 = (13+25+31+34+2+1)mod37 = 32;
1449mod37 = (25+34+17+32+8+1)mod37 = 6.

Данные расчеты показывают, что общими простыми делителями являются числа 3 и 7. Для нахождения НОД нужно проверить их степени:

Строка 16:
$3843\text{mod}3^2 = (5+7+8+4+2+1)\text{mod}3^2 = 0$; $1449\text{mod}3^2 = (7+4+2+5+8+1)\text{mod}3^2 = 0$.
Строка 17:
$3843\text{mod}3^3 = (23+25+26+13+2+1)\text{mod}3^3 = 9$;
$1449\text{mod}3^3 = (25+13+20+5+8+1)\text{mod}3^3 = 18$.
Число $7^2 = 49 > 38$, и его можно не проверять.

Следовательно, НОД (3843, 1449) = $3^2 \cdot 7 = 63$. Данный метод позволяет провести факторизацию чисел, например, $1449 = 3^2 \cdot 7 \cdot 23$. Кроме того, вычисления можно выполнять параллельно с различными модулями.

Усовершенствованный алгоритм поиска НОД в базисе Крестенсона

Строим таблицу 5.

Таблица 5

Нахождение остатков в усовершенствованном алгоритме

1		2048	1024	512	256	128	64	32	16	8	4	2	1
2	3843	1	1	1	1	0	0	0	0	0	0	1	1
3	1449		1	0	1	1	0	1	0	1	0	0	1
4	2^imod2	0	0	0	0	0	0	0	0	0	0	0	1
5	2^imod3	2	1	2	1	2	1	2	1	2	1	2	1
6	2^imod9	5	7	8	4	2	1	5	7	8	4	2	1
7	2^imod27	23	25	26	13	20	10	5	16	8	4	2	1
8	2^imod45	23	34	17	31	38	19	32	16	8	4	2	1
9	2^imod63	32	16	8	4	2	1	32	16	8	4	2	1
10	2^imod441	284	142	71	256	128	64	32	16	8	4	2	1
11	2^imod693	662	331	512	256	128	64	32	16	8	4	2	1

Анализируем таблицу 5.
Строка 4: 3843mod2 = 1; 1449mod2 = 1.
Строка 5: 3843mod3 = (2+1+2+1+2+1)mod3 = 0;
1449mod3 = (1+1+2+2+2+1)mod3 = 0.

Строка 6: 3843mod9 = (5+7+8+4+2+1)mod9 = 0;
1449mod9 = (7+4+2+5+8+1)mod9 = 0.
Строка 7: 3843mod27 = (23+25+26+13+2+1)mod27 = 9;
1449mod27 = (25+13+20+5+8+1)mod27 = 18.
Строка 8: 3843mod45 = (23+34+17+31+2+1)mod45 = 18;
1449mod45 = (34+31+38+32+8+1)mod45 = 9.
Строка 9: 3843mod63 = (32+16+8+4+2+1)mod63 = 0;
1449mod63 = (16+4+2+32+8+1)mod63 = 0.
Строка 10: 3843mod441 = (284+142+71+256+2+1)mod441 = 315;
1449mod441 = (142+256+128+32+8+1)mod441 = 126.
Строка 11: 3843mod693 = (662+331+512+256+2+1)mod693 = 378;
449mod693 = (331+256+128+32+8+1)mod693 = 63.
Из расчетов также следует, что НОД (3843, 1449) = 63.

Кроме того, отметим, что в двух последних алгоритмах необязательно искать остатки от обоих чисел *a* и *b*. Достаточно найти остатки от меньшего числа и только при их равенстве 0 проверять второе число.

Оценка вычислительной сложности известного и предложенных алгоритмов поиска НОД. Сложность предложенных алгоритмов определяется вычислительной сложностью следующих операций:

1) нахождение остатков a_j, b_j чисел X,Y по простым модулям p_j^{mi}, для которых выполняется условие $a_j = bj = 0$;

2) вычисление произведения модулей $Z = \text{НОД}(X,Y) = \prod_{j=1}^{k} p_j^{m_j}$.

В таблице 6 приведены оценки вычислительной сложности основных операций алгоритма поиска НОД в базисе Крестенсона, что позволяет сделать сравнительный анализ с приведенными выше алгоритмами поиска наибольшего общего делителя.

Таблица 6

Вычислительная сложность основных операций алгоритма поиска НОД в базисе Крестенсона и его совершенствование

№ п/п	Основные операции	Вычислительная сложность
1	$p^{mj}{}_j$	$\mathrm{O}\left(\log_2 n \times \left(\log_2 n + \frac{n}{2}\right)\right)$
2	$a_j^{(m)} = \mathrm{res}\left(\sum_{i=1}^{n-1} a_{ij} (\mathrm{mod}\, p_j^m)\right)$, $a_j^{(m)} = \mathrm{res}\left(\sum_{i=1}^{n-1} b_{ij} (\mathrm{mod}\, p_j^m)\right)$	$\mathrm{O}\left(\log_2 n + \frac{n}{2}\right)$
3	$Z = \prod_{j=1}^{k} p_j^{m_j}$	$0(k \times \log_2 n)$
k – количество модулей, для которых выполняется условие $a_j = b_j = 0$.		

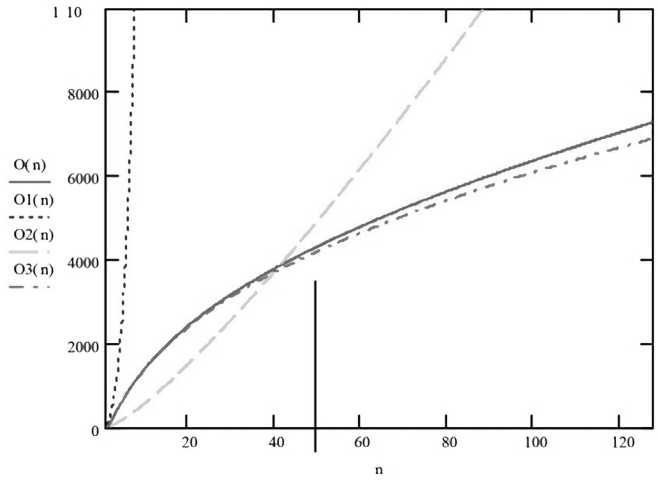
С учетом данных таблицы 6 общая вычислительная сложность предложенного алгоритма поиска НОД в базисе Крестенсона и его усовершенствования будет определяться суммой сложностей основных операций, соответственно,

Два быстрых метода нахождения наибольшего общего делителя

$$O\left(\log_2 n\times\left(\log_2 n+\frac{n}{2}+k\times\log_2 n\right)+n\times\log_2\frac{n}{2}\right);$$

$$O3\left(\log_2 n\left(\log_2 n+k\times\log_2 n+\frac{n}{2}\right)+\frac{n}{2}\log_2\frac{n}{2}\right).$$

На рисунке изображены графики, которые характеризуют сложности существующего и предложенных алгоритмов в зависимости от разрядности компонентов Z.



Сложности алгоритмов поиска НОД(X,Y):
$O(n)$ – в базисе Крестенсона; $O1(n)$ – алгоритма Евклида;
$O2(n)$ – совершенствование реализации алгоритма Евклида
с использованием вавилонской системы счисления;
$O3(n)$ – усовершенствованный алгоритм поиска НОД в базисе Крестенсона

Численный эксперимент оценки сложности предложенных алгоритмов поиска НОД показывает, что в диапазоне двоичных разрядов от 0 до 42 бит следует использовать усовершенствования реализации алгоритма Евклида с использованием вавилонской системы счисления Радемахера – Крестенсона, а при увеличении разрядности чисел – алгоритм поиска НОД в базисе Крестенсона и его совершенствование.

Заключение

Предложенные теоретические основы поиска НОД с применением теоретико-числового базиса Крестенсона и вавилонской системы счисления остаточных классов позволяют уменьшить вычислительную сложность на 1–2 порядка, проводить вычисления с использованием параллельной технологии и эффективно использовать для реализации высокопроизводительных алгоритмов обработки и защиты информационных потоков на основе асимметричных алгоритмов шифрования.

Литература

1. *Амер И., Ишмухаметов Ш.Т.* Об ускорении *k*-арного алгоритма вычисления НОД натуральных чисел // Ученые записки Казан. ун-та. Серия «Физико-математические науки». 2019. Т. 161, кн. 1. С. 110–118. DOI: 10.26907/2541-7746.2019.1.110-118.

2. *Гашков С.Б., Сергеев И.С.* Об аддитивной сложности матриц НОД и ОК // Математические заметки. 2016. Т. 100, № 2. С. 196–211.
3. *Корюкин А.Н., Себельдин А.М., Силла А.Л.* Кольца с наибольшим общим делителем // Фундаментальная и прикладная математика. 2010. Т. 16, № 7. С. 69–74.
4. *Малашонок Н.А.* НОД многочленов дуального переменного // Вестник Тамбовского ун-та. Серия «Естественные и технические науки». 2012. Т. 17, № 1. С. 91–92.
5. *Малашонок Н.А.* Результант на алгебре дуальных чисел // Вестник Тамбовского ун-та. Серия «Естественные и технические науки». 2013. Т. 18, № 1. С. 110–111.
6. *Окулов С.М., Лялин А.В.* Расширенный алгоритм Евклида // Информатика и образование. 2011. № 8. С. 37–41.
7. *Оленев А.А.* Особенности реализации алгоритма Евклида в MAPLE // Актуальные вопросы инженерного образования – 2015: сборник науч. трудов Междунар. науч.-метод. конф. (Октябрьский, 27 ноября 2015 г.). Ставрополь: Альфа Принт, 2016. С. 160–167.
8. *Павлова Т.В., Токц Н.А.* Применение теоремы о линейной форме наибольшего общего делителя к решению сравнений первой степени // Проблемы и перспективы физико-математического и технического образования: сборник материалов Всерос. науч.-практ. конф. / отв. ред. Т.С. Мамонтова. 2015. С. 158–164.
9. *Фалин Г., Фалин А.* Избранные задачи на делимость целых чисел // Математика. Первое сентября. 2013. № 5. С. 43–50.
10. *Цирулик В.Г.* Неевклидов алгоритм отыскания наибольших общих делителей систем многочленов // Научные труды SWorld. 2014. Т. 29, № 1. С. 89–91.

Literatura

1. *Amer I., Ishmukhametov Sh.T.* Ob uskorenii k-arnogo algoritma vychisleniya NOD natural'nykh chisel // Uchenye zapiski Kazan. un-ta. Seriya «Fiziko-matematicheskie nauki». 2019. T. 161, kn. 1. S. 110–118. DOI: 10.26907/2541-7746.2019.1.110-118.
2. *Gashkov S.B., Sergeev I.S.* Ob additivnoj slozhnosti matrits NOD i OK // Matematicheskie zametki. 2016. T. 100, № 2. S. 196–211.
3. *Koryukin A.N., Sebel'din A.M., Silla A.L.* Kol'tsa s naibol'shim obshchim delitelem // Fundamental'naya i prikladnaya matematika. 2010. T. 16, № 7. S. 69–74.
4. *Malashonok N.A.* NOD mnogochlenov dual'nogo peremennogo // Vestnik Tambovskogo un-ta. Seriya «Estestvennye i tekhnicheskie nauki». 2012. T. 17, № 1. S. 91–92.
5. *Malashonok N.A.* Rezul'tant na algebre dual'nykh chisel // Vestnik Tambovskogo un-ta. Seriya «Estestvennye i tekhnicheskie nauki». 2013. T. 18, № 1. S. 110–111.
6. *Okulov S.M., Lyalin A.V.* Rasshirennyj algoritm Evklida // Informatika i obrazovanie. 2011. № 8. S. 37–41.
7. *Olenev A.A.* Osobennosti realizatsii algoritma Evklida v MAPLE // Aktual'nye voprosy inzhenernogo obrazovaniya – 2015: sbornik nauch. Trudov Mezhdunar. nauch.-metod. konf. (Oktyabr'skij, 27 noyabrya 2015 g.). Stavropol': Al'fa Print, 2016. S. 160–167.
8. *Pavlova T.V., Tokts N.A.* Primenenie teoremy o linejnoj forme naibol'shego obshchego delitelya k resheniyu sravnenij pervoj stepeni // Problemy i perspektivy fiziko-matematicheskogo i tekhnicheskogo obrazovaniya: sbornik materialov Vseros. nauch.-prakt. konf. / otv. red. T.S. Mamontova. 2015. S. 158–164.
9. *Falin G., Falin A.* Izbrannye zadachi na delimost' tselykh chisel // Matematika. Pervoe sentyabrya. 2013. № 5. S. 43–50.
10. *Tsirulik V.G.* Neevklidov algoritm otyskaniya naibol'shikh obshchikh delitelej system mnogochlenov // Nauchnye trudy SWorld. 2014. T. 29, № 1. S. 89–91.