В.А. Минаев, ***Н.П. Васильев***, ***В.В. Лукьянов***, ***С.А. Никонов***, ***Д.В. Никеров***
(Российский новый университет; e-mail: m1va@yandex.ru)

ЗАЩИТА ИНФОРМАЦИИ: ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ АЛГОРИТМЫ ГЕНЕРАЦИИ ПРОСТЫХ ЧИСЕЛ С ПРИМЕНЕНИЕМ КОЛЬЦЕВОЙ ФАКТОРИЗАЦИИ

Разработаны алгоритмы генерации простых чисел с применением кольцевой факторизации. Материал может быть полезен для решения проблем безопасности конфиденциальной информации.

Ключевые слова: простые числа, генерация, защита информации.

V.A. Minaev, ***N.P. Vasil'ev***, ***V.V. Luk'janov***, ***S.A. Nikonov***, ***D.V. Nikerov***

INFORMATION PROTECTION: HIGH-PERFORMANCE ALGORITHMS FOR GENERATING PRIMES USING FACTORIZATION RING

Algorithms for generating primes using factorization ring was designed. The material can be useful in solving problems security of confidential information.

Key words: primes, generation, information protection.

Статья поступила в редакцию Интернет-журнала 10 сентября 2013 г.

Введение

Простые числа используются для шифрования информации. Так, например, известный алгоритм шифрования *RSA* использует два простых числа для формирования открытого ключа. При этом, чтобы расшифровать содержимое конфиденциального письма, надо провести процесс факторизации некоторого очень большого числа, то есть разложить его на сомножители, которые являются простыми числами.

Решение задачи факторизации при этом – исключительно вычислительно-ёмкая процедура, так как помимо нахождения сомножителей, каждый из них необходимо проверить на простоту.

Для генерации простых чисел методом просеивания (исключение составных чисел, так называемое "решето") применяются решета Эратосфена, Аткина и Сундарама в разных модификациях и производные от этих алгоритмов.

В 2011 году опубликована работа *В.А. Минаева* [1]**,** в которой описывается алгоритм**,** являющийся основой для создания новых высокопроизводительных алгоритмов, превосходящих по скорости вычисления простых чисел все предыдущие. Алгоритм основывается на теореме о полном множестве простых чисел.

Автор доказал, что полное множество простых чисел вида $\{6k + 1\}$, $k = 1, 2, 3\ldots$ (*по определению В.А. Минаева – плюс простые числа*) формируется путём вычитания из общего множества чисел $\{^{+}q_n\}$ вида $\{6k + 1\}$; $k = 1, 2, 3\ldots$,

определённого как $\{^{+}S\}$, подмножества составных чисел $^{+}c_{(q_i^{\pm})}$, определяемых с помощью уравнений:

$$^{+}c_{(q_i^{-})} = {}^{-}q_i \cdot {}^{-}q_i + {}^{-}q_i \cdot 6 \cdot m;$$
$$^{+}c_{(q_i^{+})} = {}^{+}q_i \cdot {}^{+}q_i + {}^{+}q_i \cdot 6 \cdot m, \quad (1)$$

где $m = 0, 1, 2 \ldots$;
$i = 1, 2, 3 \ldots$.

А полное множество простых чисел вида $\{6k - 1\}$; $k = 1, 2, 3\ldots$(*минус простые числа*) – путём вычитания из общего множества чисел $\{^{-}q_i\}$ вида $\{6k - 1\}$; $k = 1, 2, 3\ldots$, определённого как $\{^{-}S\}$, подмножества составных чисел $^{-}c_{(q_i^{\pm})}$, вычисляемых из соотношений:

$$^{-}c_{(q_i^{-})} = {}^{-}q_i \cdot {}^{+}q_i + {}^{-}q_i \cdot 6 \cdot m;$$
$$^{-}c_{(q_i^{+})} = {}^{+}q_i \cdot {}^{-}q_i + {}^{+}q_i \cdot 6 \cdot m, \quad (2)$$

где $m = 0, 1, 2\ldots$;
$i = 1, 2, 3\ldots$.

Смысл алгоритма, предложенного в [1], связан не с прямым вычислением простых чисел, а с нахождением полного множества составных того же вида $\{6k \pm 1\}$; $k = 1, 2, 3\ldots$, и последующим вычитанием соответствующих множеств.

В настоящей статье описываются новые *индексные алгоритмы генерации простых чисел*, на сегодняшний день самые быстрые, основой разработки которых является теорема, доказанная в [1], и правило знаков, сформулированное там же.

Индексные зависимости формирования составных чисел

Согласно правилу знаков [1], любые составные числа из множеств ^{-}S и ^{+}S представляются в виде произведений:

$$^{-}c(^{-}q_k, {}^{+}q_n) = {}^{-}q_k \cdot {}^{+}q_n;$$
$$^{-}c(^{+}q_k, {}^{-}q_n) = {}^{+}q_k \cdot {}^{-}q_n; \quad (3)$$

$$^{+}c(^{-}q_k, {}^{-}q_n) = {}^{-}q_k \cdot {}^{-}q_n;$$
$$^{+}c(^{+}q_k, {}^{+}q_n) = {}^{+}q_k \cdot {}^{+}q_n, \quad (4)$$

где $k, n = 1, 2, 3, \ldots$.

Подставляя соответствующие значения для $^{\pm}q_k$ и $^{\pm}q_n$, получим соотношения:

$$^{-}c(^{-}q_k, {}^{+}q_n) = 6 \cdot (n \cdot {}^{-}q_k + k) - 1; \quad (5)$$
$$^{-}c(^{+}q_k, {}^{-}q_n) = 6 \cdot (n \cdot {}^{+}q_k - k) - 1; \quad (6)$$
$$^{+}c(^{-}q_k, {}^{-}q_n) = 6 \cdot (n \cdot {}^{-}q_k - k) + 1; \quad (7)$$
$$^{+}c(^{+}q_k, {}^{+}q_n) = 6 \cdot (n \cdot {}^{+}q_k + k) + 1. \quad (8)$$

Из (5)-(8) следует, что всевозможные сочетания n и k (k, n = 1, 2, 3,....) позволяют получить весь набор составных чисел вида $\{6i \pm 1\}$, i = 1, 2, 3,...; в любом заданном интервале.

Так, например, при n = 1 выражение, стоящее в скобках формул (5)-(8), описывает получение следующего набора составных чисел:

$$\begin{aligned} {}^{-}c({}^{-}q_k, 7) &= 6 \cdot ({}^{-}q_k + k) - 1; \\ {}^{-}c({}^{+}q_k, 5) &= 6 \cdot ({}^{+}q_k - k) - 1; \\ {}^{+}c({}^{-}q_k, 5) &= 6 \cdot ({}^{-}q_k - k) + 1; \\ {}^{+}c({}^{+}q_k, 7) &= 6 \cdot ({}^{+}q_k + k) + 1. \end{aligned} \qquad (9)$$

Эти составные числа адресуются во множестве ${}^{-}S$ или ${}^{+}S$ следующими четырьмя индексами:

$$\begin{aligned} {}^{+}k_1 &= ({}^{-}q_k + k); \\ {}^{-}k_2 &= ({}^{+}q_k - k); \\ {}^{-}k_3 &= ({}^{-}q_k - k); \\ {}^{+}k_4 &= ({}^{+}q_k + k). \end{aligned} \qquad (10)$$

или в векторном виде (вектор включает по две компоненты):

$$\begin{aligned} \overline{{}^{+}k} &= ({}^{\pm}q_k + k) \\ \overline{{}^{-}k} &= ({}^{+}q_k - k); \; k = 1, 2, 3, \ldots. \end{aligned} \qquad (11)$$

Приведём примеры адресации (табл. 1 и 2), используя формулы (5-11), при n = 1.

Таблица 1

Адресация составных чисел с использованием ${}^{+}k$-индексной зависимости

k	$\pm q_k$	${}^{+}k_1$, ${}^{+}k_4$	${}^{\pm}c({}^{\pm}q_{k_1}, {}^{+}k_1)$ ${}^{\pm}c({}^{\pm}q_{k_1}, {}^{+}k_4)$
1	5 (множество ${}^{-}S$)	5+1 = 6	35 (множество ${}^{-}S$)
1	7 (множество ${}^{+}S$)	7+1 = 8	49 (множество ${}^{+}S$)
4	25 (множество ${}^{+}S$)	25+4 = 29	175 (множество ${}^{+}S$)
8	47 (множество ${}^{-}S$)	47+8 = 55	329 (множество ${}^{+}S$)
13	79 (множество ${}^{+}S$)	13+79 = 92	553 (множество ${}^{+}S$)
127	761 (множество ${}^{-}S$)	127+761 = 888	5327 (множество ${}^{-}S$)

Таблица 2

Адресация составных чисел с использованием ${}^{-}$k -индексной зависимости

k	$\pm q_k$	${}^{-}k_2$, ${}^{-}k_3$	${}^{\pm}c({}^{\pm}q_{k_1}, {}^{-}k_2)$ ${}^{\pm}c({}^{\pm}q_{k_1}, {}^{-}k_3)$
1	5 (во множестве ${}^{-}$**S**)	5-1 = 4	25 (во множестве ${}^{+}$**S**)
1	7 (во множестве ${}^{+}$**S**)	7-1 = 6	35 (во множестве ${}^{-}$**S**)
4	25 (во множестве ${}^{+}$**S**)	25-4 = 21	125 (во множестве ${}^{-}$**S**)
8	47 (во множестве ${}^{-}$**S**)	47-8 = 39	235 (во множестве ${}^{+}$**S**)
13	79 (во множестве ${}^{+}$**S**)	79-13 = 66	395 (во множестве ${}^{-}$**S**)
127	761 (во множестве ${}^{-}$**S**)	761-127 = 634	3805 (во множестве ${}^{+}$**S**)

Использование кольцевой факторизации

Для ускорения поиска простых чисел на определённом отрезке натурального ряда для предварительного отбора составных чисел в работе [2] реализован и исследован *индексный алгоритм*, построенный с использованием кольцевой факторизации для 3# = 6 (приориал простых чисел 2 · 3). Она позволила на предварительном этапе отсеять значительную часть составных чисел – 66,66…%. На рис. 1 приведено графическое изображение кольцевой факторизации для 3# и 5# [3-4].

Нужно отметить, что для 7# (приориал простых чисел равен 2 · 3 · 5 · 7) – отсеивается больше 77 % составных чисел, а для 251# – около 90 %.

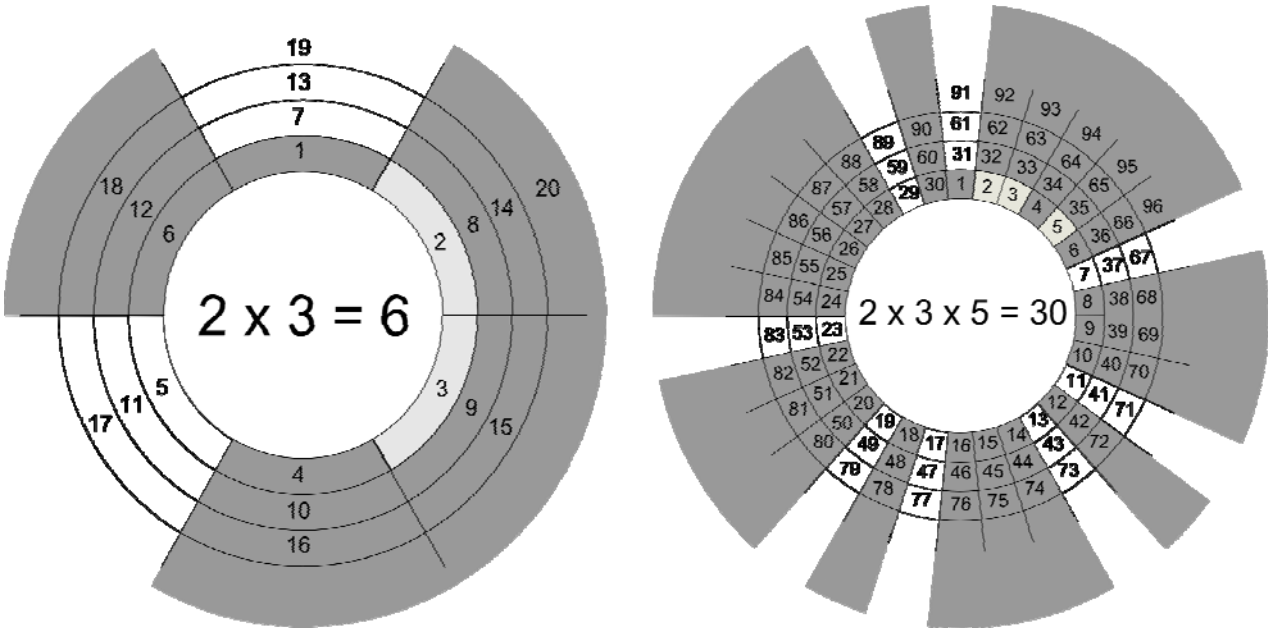

Рис. 1. Графическое изображение кольцевой факторизации для 3# и 5#

Учитывая важность предварительного просеивания, авторы обосновывают и исследуют индексные алгоритмы, построенные с применением кольцевой факторизации для примариалов, превышающих значение 6.

Введём понятие *порядок индексного алгоритма*, под которым подразумевается количество первых простых чисел, использованных при соответствующей кольцевой факторизации. Например, для 3# = 2 · 3 = 6 порядок индексного алгоритма равен 2, а для 5# = 2 · 3 · 5 = 30 – порядок равен 3.

Число k = 1, 2, 3..., участвующее в формировании пары чисел, каждое из которых принадлежит множеству ${}^{-}S$ *или* ${}^{+}S$, определим как *k-индекс*.

k-индекс адресует числа множеств ${}^{-}S$ и ${}^{+}S$ следующим образом: одному его значению одновременно соответствуют по одному элементу из каждого множества; при этом соответствие между *k-индексом* и элементом в каждом из множеств – взаимно однозначное.

Множества ${}^{-}S$ *и* ${}^{+}S$ будем называть *симметричными* по отношению друг к другу, поскольку их элементы формируются симметрично относительно одного и того же *k-индекса* с разностью 2, то есть для симметричных элементов множеств ${}^{-}S$ *и* ${}^{+}S$ разность ${}^{+}q_k - {}^{-}q_k = 2$.

В связи с тем, что каждое составное число есть член какой-либо арифметической прогрессии [5], был поставлен вопрос: можно ли адресовать все составные числа во множествах ${}^{-}S$ и ${}^{+}S$ через *k*-индексы и первые члены арифметических прогрессий, порождающих их подмножества?

Решение данного вопроса имеет существенную вычислительную и практическую ценность, так как определение массива индексов, соответствующих массиву составных чисел, позволяет избежать выделения огромной памяти под хранение последних в результате решений уравнений (1), (2).

Нужно отметить, что существующие до сегодняшнего дня методы получения очередного простого числа учитывали все найденные предыдущие простые и составные, в то время как индексный подход позволяет без учёта "предыстории" вычислять простые и составные числа в заданном произвольном интервале на множестве натуральных чисел при задании его нижней и верхней границы.

Предположим, требуется получить простые числа в диапазоне от одного миллиарда до двух. Для ранее известных методов приходилось задавать только верхнюю границу, то есть 2 *млрд*, выполнять ресурсоемкие расчёты, а затем отсекать ненужные значения простых и составных чисел. В нашем же случае задаются обе границы, и расчёт ведется только в их пределах для требуемых значений простых и составных.

Для произвольного *n* обобщим выражения (11) в виде уравнений:

$$\begin{aligned} {}^{+}k_n &= n \cdot {}^{\pm}q_k + k; \\ {}^{-}k_n &= n \cdot {}^{\mp}q_k - k; \quad k, n = 1, 2, 3,\ldots. \end{aligned} \qquad (12)$$

Члены этих арифметических прогрессий – это *k*-индексы, адресующие элементы множеств ${}^{-}S$ или ${}^{+}S$; При этом ${}^{\pm}q_k$ может быть как простым, так и составным числом. Члены прогрессии адресуют составные числа, некоторые из них могут адресоваться и другой k-индексной прогрессией, формируя "дубли".

В работе [5] отмечено, что все "дубли" игнорируются при вычитании составных вида $\{6k \pm 1\}$, $k = 1, 2, 3\ldots$, из объединения множеств ${}^{+}S$ и ${}^{-}S$, в результате остаются только простые числа вида $\{6k \pm 1\}$, $k = 1, 2, 3\ldots$.

Используя формулы (12), составим следующую таблицу примеров (табл. 3).

Анализируя столбцы табл. 3, можно заметить, что ${}^{+}k_n$ адресуют все составные числа в том множестве (${}^{-}S$ или ${}^{+}S$), которому принадлежит ${}^{\pm}q_k$, а ${}^{-}k_n$ – в симметричном множестве. Причём адресуемые составные числа кратны соответствующему ${}^{\pm}q_k$. Это дает возможность, определив граничные условия на ${}^{\pm}k_n$ и, соответственно, максимально необходимые для этого q_k, *n* и *k*, исходя из границ интервала, найти все простые числа.

Таблица 3

Примеры адресации составных чисел через их индексы

q_k	k	n	${}^{+}k_n$	${}^{\pm}c({}^{+}k_n)$	${}^{-}k_n$	${}^{\pm}c({}^{-}k_n)$
5 ($^{-}$S)	1	1	6	35 ($^{-}$S)	4	25 ($^{+}$S)
		2	11	65 ($^{-}$S)	9	55 ($^{+}$S)
		3	16	95 ($^{-}$S)	14	85 ($^{+}$S)
		4	21	125 ($^{-}$S)	19	115 ($^{+}$S)
		5	26	155 ($^{-}$S)	24	145 ($^{+}$S)
19 ($^{+}$S)	3	1	22	133 ($^{+}$S)	16	95 ($^{-}$S)
		2	41	247 ($^{+}$S)	35	209 ($^{-}$S)
		3	60	361 ($^{+}$S)	54	323 ($^{-}$S)
		4	79	475 ($^{+}$S)	71	437 ($^{-}$S)
25 ($^{+}$S)	4	1	29	175 ($^{+}$S)	21	125 ($^{-}$S)
		2	54	325 ($^{+}$S)	46	275 ($^{-}$S)

Оценка производительности работы индексного алгоритма второго порядка

Индексный алгоритм второго порядка был реализован на *C++* и протестирован на скорость вычислений и занимаемую при работе память. Результаты тестов отражены в табл. 4.

Таблица 4

Скорость вычислений и требуемая для работы алгоритма память на различных интервалах

Диапазон	Время, *с*	Память, *Мб*
$5 \div 1 \cdot 10^9$	22	317
$5 \div 2 \cdot 10^9$	48	635
$5 \div 3 \cdot 10^9$	74	953
$5 \div 4 \cdot 10^9$	101	1271
$5 \div 5 \cdot 10^9$	164	1589
$10^{20} \div 10^{20}+1 \cdot 10^9$	**105**	**317**
$10^{20} \div 10^{20}+2 \cdot 10^9$	**162**	**635**
$10^{20} \div 10^{20}+3 \cdot 10^9$	**222**	**953**
$10^{20} \div 10^{20}+4 \cdot 10^9$	**287**	**1271**
$10^{20} \div 10^{20}+5 \cdot 10^9$	**360**	**1589**

Корректность работы алгоритма проверена с использованием источников для 20-разрядных и менее чисел [6-8] в пределах первых 50 *млн* простых чисел.

Сравнение производительности алгоритмов

Проведём сравнительные тесты различных алгоритмов вычисления простых чисел, работающих в одинаковых условиях: каждый из них был реализован на языке C++, запускался на одном и том же компьютере, на вход подавались одни и те же интервалы от 1 до *n* и замерялось время, за которое алгоритм выделит все простые числа на данном интервале. Результаты измерений представлены на рис. 2.

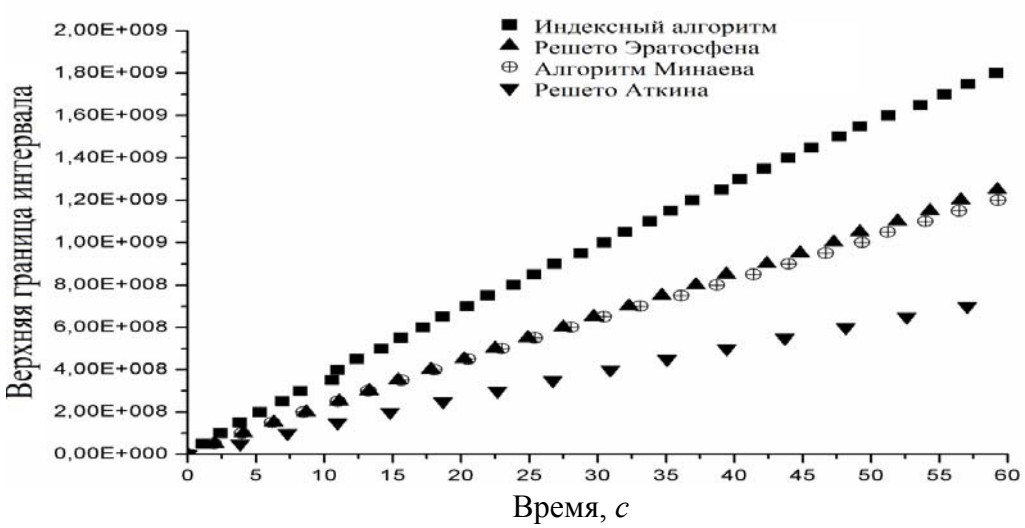

Рис. 2. Сравнение скоростей работы алгоритмов вычисления всех простых чисел на интервале от 1 до n

Предварительное просеивание с применением метода кольцевой факторизации в индексных алгоритмах произвольного порядка

Рассмотрим более детально индексный алгоритм второго порядка для нахождения простых чисел в заданном отрезке $[N_{\min}; N_{\max}]$, исходя из результатов работы [2]. Для этого применим кольцевую факторизацию для 3# к отрезку $[1; N_{\max}]$, введём обозначения $q_{6k}^{+1} = 6k+1$, $q_{6k}^{+5} = 6k+5$, где $k = 0, 1, 2, \ldots k_{\max}$ (для максимально возможного $k = k_{\max}$ должно быть выполнено условие $q_{6k_{\max}}^{+1} \le N_{\max}$), и занесём эти данные в табл. 5.

Таблица 5

Результаты кольцевой факторизации для 3#

Индекс	q_{6k}^{+1}	q_{6k}^{+5}
0	1	5
1	7	11
2	13	17
3	19	23
4	25	29
…	…	…
$k_{\max}$	$6k_{\max} + 1$	$6k_{\max} + 5$

В левом столбце таблицы приведена последовательность индексов $0, 1, 2, \ldots k_{\max}$, в двух других столбцах – отображения этой последовательности во множествах вида $\{6k+1\}$ и $\{6k+5\}$, содержащие как простые, так и составные числа, а также единицу.

Перейдём к следующему этапу – отсеву оставшихся после кольцевой факторизации составных чисел из множеств $\{q_{6k}^{+1}\}$ и $\{q_{6k}^{+5}\}$, где $k = 0, 1, 2, \ldots k_{\max}$.

Как показано в [2], все составные числа, кратные числам q_{6i}^{+1} или q_{6i}^{+5}, где $i = 0, 1, 2, \ldots i_{\max}$ (для максимально возможного $i = i_{\max}$ должно быть выполнено условие $(q_{6i_{\max}}^{+1})^2 \le N_{\max}$), можно исключить из таблицы 5, вычислив массив их индексов с помощью соответствующих соотношений.

Для фиксированного q_{6i}^{+1} последовательность $(m \cdot q_{6i}^{+1} + i)$, где $m = 1, 2, 3, \ldots$, индексирует все кратные ему составные числа во множестве $\{q_{6k}^{+1}\}$, а последовательность $(m \cdot q_{6i}^{+1} - i - 1)$, где $m = 1, 2, 3, \ldots$, индексирует все кратные q_{6i}^{+1} составные числа во множестве $\{q_{6k}^{+5}\}$ [2].

Для заданного отрезка $[N_{\min}; N_{\max}]$ из вышеупомянутых последовательностей нужно выбирать те их члены, которые индексируют числа, содержащиеся внутри него. Пример для $q_{6\cdot1}^{+1} = 7$ приведён в табл. 6, в левом столбце которой отмечены индексы $m \cdot q_{6\cdot1}^{+1}$, где $m = 1, 2, 3, \ldots$, а во втором и третьем столбцах отмечены кратные $q_{6\cdot1}^{+1}$ числа, индексируемые последовательностями соответственно $(m \cdot q_{6\cdot1}^{+1} + 1) = \{8, 22, 29, 36, \ldots\}$ и $(m \cdot q_{6\cdot1}^{+1} - 2) = \{5, 19, 26, 33, \ldots\}$.

Таблица 6

Пример адресации составных чисел при $q_{6\times1}^{+1} = 7$

k	q_{6k}^{+1}	q_{6k}^{+5}
0	1	5
…	…	…
5	31	**35**
6	37	41
7	43	47
8	**49**	53
…	…	…
19	115	**119**
20	121	125
21	127	131
22	**133**	137
23	139	143
24	145	149
25	151	155
26	157	**161**
27	163	167
28	169	173
29	**175**	179
30	181	185
31	187	191
32	193	197
33	199	**203**
34	205	209
35	211	215
36	**217**	221
…	…	…

Для фиксированного q_{6i}^{+5}, в свою очередь, уже другая последовательность $(m \cdot q_{6i}^{+5} - i)$, где $m = 1, 2, 3, \ldots$, индексирует все кратные этому q_{6i}^{+5} составные числа во множестве $\{q_{6k}^{+1}\}$, а последовательность $(m \cdot q_{6i}^{+5} + i - 1)$, где $m = 1, 2, 3, \ldots$, индексирует все кратные q_{6i}^{+5} составные числа во множестве $\{q_{6k}^{+5}\}$. Опять же для заданного отрезка $[N_{\min}; N_{\max}]$ из вышеупомянутых последовательностей нужно выбирать те их члены, которые индексируют числа, содержащиеся внутри этого отрезка. Пример для $q_{6\cdot0}^{+5} = 5$ приведён в табл. 7, в левом столбце которой отмечены индексы $m \cdot q_{6\cdot0}^{+5}$, где $m = 1, 2, 3, \ldots$, а во втором и третьем отмечены кратные $q_{6\cdot0}^{+5}$ числа, индексируемые последовательностями соответственно $(m \cdot q_{6\cdot0}^{+1} - 1) = \{4, 24, 29, 34, \ldots\}$ и $(m \cdot q_{6\cdot0}^{+1}) = \{5, 25, 30, 35, \ldots\}$.

Таблица 7

Пример адресации составных чисел при $q_{6\cdot0}^{+5} = 5$

k	q_{6k}^{+1}	q_{6k}^{+5}
0	1	5
…	…	…
4	**25**	29
5	31	**35**
…	…	…
24	**145**	149
25	151	*155*
26	157	161
27	163	167
28	169	173
29	**175**	179
30	181	**185**
31	187	191
32	193	197
33	199	203
34	**205**	209
35	211	**215**
…	…	…

Таким образом, если перебрать все подходящие q_{6i}^{+1} и q_{6i}^{+5}, то из заданного отрезка $[N_{\min}; N_{\max}]$ можно отсеять все составные числа и тем самым определить в нем все простые.

Обобщение алгоритма второго порядка дало возможность сформировать индексный алгоритм произвольного порядка для нахождения простых чисел в заданном отрезке $[N_{\min}; N_{\max}]$. При этом были обнаружены *паттерны*, под которыми авторы понимают строго повторяющиеся схемы размещения составных чисел в натуральном ряду.

Таким образом, если в заданном отрезке $[N_{\min}; N_{\max}]$ с использованием соответствующих паттернов отсеять составные числа, то в нём остаются только простые числа.

Исследование индексных алгоритмов различного порядка

Рассмотренные индексные алгоритмы реализованы на языке C++ с использованием библиотеки GMP для работы с большими числами и проверены на совпадение результатов генерации с первыми 50 *млн* простых чисел [6-8]. Индексные алгоритмы второго, третьего и четвертого порядков исследовались на время работы при различных отрезках и различных нижних границах на компьютере с процессором Intel Core i3 2,93 *ГГц*. На рис. 3, для примера, представлены результаты работы алгоритмов для отрезка 10^6 при различных значениях нижней границы отрезка.

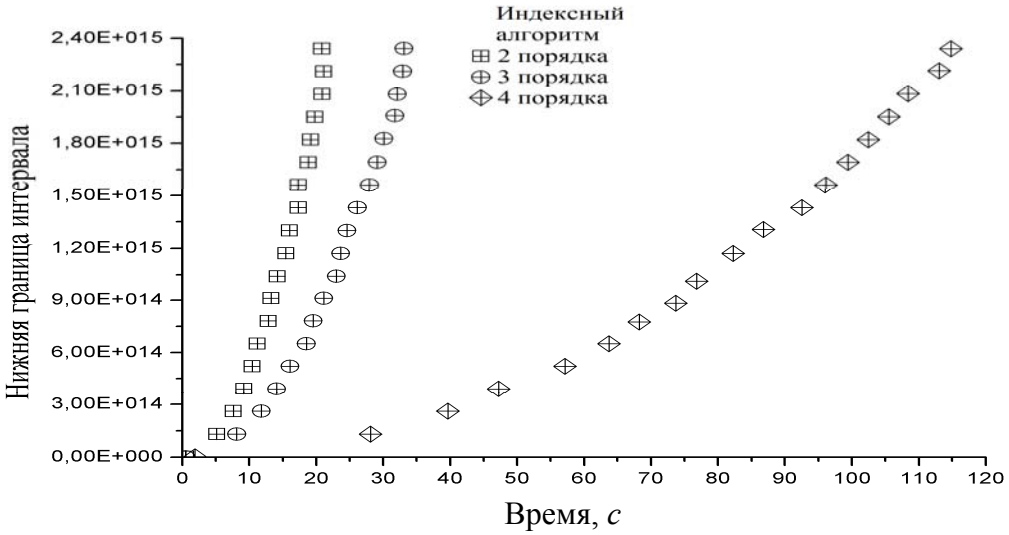

Рис. 3. Сравнение скоростей работы индексных алгоритмов генерации простых чисел на отрезке размером 10^6 (считая от значения ординаты)

Выводы

1. Развитие положений работ [1-2] связано с реализацией *индексного принципа* нахождения составных чисел, заключающегося в вычислении не самих чисел, а массива соответствующих им индексов. Такой подход способен существенно увеличить производительность алгоритмов вычисления простых чисел, а, следовательно, скорость факторизации составных чисел, что напрямую связано с алгоритмами шифрования конфиденциальной информации.

2. Индексные алгоритмы ориентированы на решение задач:

- дальнейших исследований в области простых чисел, в частности – оценки распределения простых чисел в произвольном диапазоне, проверки решения открытых математических проблем;
- создания генератора случайных простых чисел для криптографических приложений;
- поиска оптимальных значений простых чисел при использовании хеш-таблиц в базах данных и других задач, связанных с их применением.

3. Индексные алгоритмы в их текущей реализации представляют принципиально новый подход к поиску простых чисел. Однако требуется их оптимизация применительно к объёму кэш-памяти используемого процессора.

В то же время, индексные алгоритмы с использованием кольцевой факторизации не требовательны к вычислительным ресурсам, поскольку оперируют с булевым массивом. Благодаря своей многозадачной-структуре алгоритм легко модифицируется как многопоточный с реализацией на GPU. В частности они легко реализуемы при параллельном вычислении, используя кластерные вычислительные технологии NVIDIA CUDA, Open MP, Open MPI и другие.

Литература

1. ***Минаев В.А.*** Теорема о полном множестве простых чисел. М.: НИЯУ МИФИ, 2011. 24 с.

2. ***Минаев В.А., Васильев Н.П., Лукьянов В.В., Никонов С.А., Никеров Д.В.*** Высокопроизводительный алгоритм генерации простых чисел в произвольном диапазоне // Матер. XIV междунар. науч. конф. "Цивилизация знаний: проблемы и смыслы образования". 2013. М.: РосНОУ.

3. ***Wheel*** factorization. http://primes.utm.edu/glossary/xpage/WheelFactorization.html (дата обращения: 13.06.2013).

4. ***Wheel*** factorization. http://en.wikipedia.org/wiki/Wheel_factorization (дата обращения: 18.06.2013).

5. ***Минаев В.А.*** Простые числа: новый взгляд на закономерности формирования. М.: Логос, 2011. 80 с.

6. The first fifty million primes. http://primes.utm.edu/lists/small/millions (дата обращения: 10.06.2013).

7. The Prime Pages (prime number research, records and resources). http://primes.utm.edu (дата обращения: 17.02.2013).

8. ***Простые*** числа. http://ru.numberempire.com/primenumbers.php (дата обращения 25.04.2013).