УДК 511.333;511.337

В.А. Минаев[1]
Н.П. Васильев[2]
В.В. Лукьянов[3]
С.А. Никонов[4]
Д.В. Никеров[5]

ИНДЕКСНЫЕ АЛГОРИТМЫ ВЫЧИСЛЕНИЯ ПРОСТЫХ ЧИСЕЛ С ИСПОЛЬЗОВАНИЕМ МЕТОДА КОЛЬЦЕВОЙ ФАКТОРИЗАЦИИ

Рассмотрены индексные алгоритмы вычисления простых чисел в сочетании с методом кольцевой факторизации для предварительного отбора составных чисел. Даются определения порядка индексного алгоритма и паттерна размещения составных чисел. Производится сравнение индексных алгоритмов различного порядка.

Ключевые слова: *простые числа, кольцевая факторизация, индексный алгоритм.*

V.A. Minaev
N.P. Vasiliev
V.V. Lukyanov
S.A. Nikonov
D.V. Nikerov

INDEX ALGORITHMS FOR COMPUTING THE PRIME NUMBERS USING THE METHOD OF WHEEL FACTORIZATION

The Primes index calculation algorithms in combination with the method of wheel factorization for pre-selection of composite numbers are considered. The definitions of the index algorithm order and pattern layouts of composite numbers are given. Comparison of different order index algorithms is made.

Keywords: *Primes, wheel factorization, index algorithm.*

Введение

При решении задачи нахождения полного множества простых чисел на определенном отрезке натурального ряда для предварительного отбора составных чисел применяется метод кольцевой факторизации [1].

Существо метода заключается в следующем: перемножаются несколько первых простых чисел, идущих подряд (математическая операция, известная как примориал), например, 3# = 2 × 3 = 6, 5# = 2 × 3 × 5 = 30 или 7# = 2 × 3 × 5 × 7 = 210. Затем строится кольцевая диаграмма, разделенная на количество секторов, равное полученному примориалу. Каждая часть кольца, отсеченная сектором, нумеруется так, как показано на рис. 1 [2].

Отметим сектора, которые начинаются: с единицы; с простых чисел, не участвовавших в получении примориала; а также со всевозможных, но меньших примориала, произведений этих простых чисел. Согласно методу кольцевой факторизации, в этих секторах лежат простые и составные числа, а в остальных (выделенных серым цветом на рис. 1) – только составные.

Для 7# = 210 начала секторов будут следующими: 1, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, **121** = 11 × 11, 127, 131, 137, 139, **143** = 11 × 13, 149, 151, 157, 163, 167, **169** = 13 × 13, 173, 179, 181, **187** = 11 × 17, 191, 193, 197, 199, **209** = 11 × 19.

Метод кольцевой факторизации позволяет отсеять значительную часть составных чисел [1], а именно: для 3# отсеивается 66,66…% всех составных чисел, для 7# – больше 77%, а для 251# – около 90%.

На следующем этапе для исключения оставшихся составных чисел применяют различные

[1] Доктор технических наук, профессор, проректор НОУ ВПО «Российский новый университет».

[2] Кандидат технических наук, доцент Национального исследовательского ядерного университета «МИФИ».

[3] Аспирант Национального исследовательского ядерного университета «МИФИ».

[4] Аспирант НОУ ВПО «Российский новый университет».

[5] Аспирант НОУ ВПО «Российский новый университет».

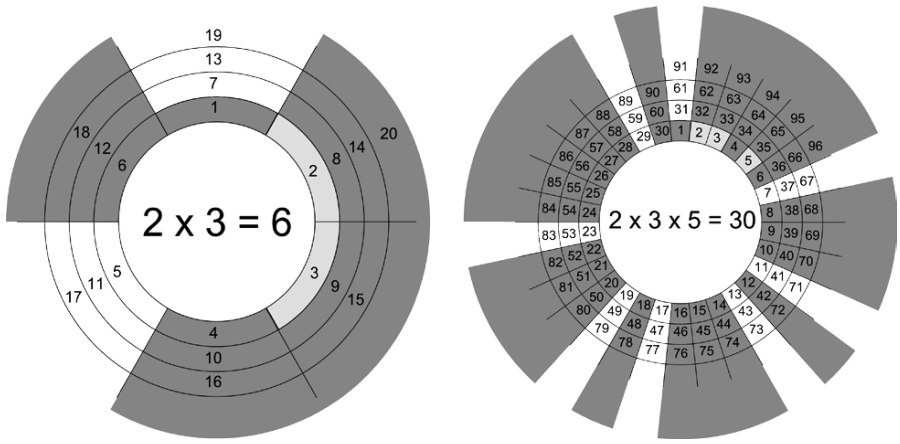

Рис. 1. Графическое изображение кольцевой факторизации для 3# и 5#

алгоритмы. В частности – решето Эратосфена, решето Аткина и другие. Появились и современные алгоритмы [3; 4].

Так, в работе [3] для нахождения полного множества простых чисел на определенном отрезке натурального ряда применяется аддитивное представление составных чисел вида $\{6k \pm 1\}$, где $k = 1, 2, 3, \ldots$ через простые числа того же вида.

Индексные алгоритмы с использованием кольцевой факторизации

Развитие положений работы [3] связано с реализацией *индексного принципа* нахождения составных чисел, заключающегося в вычислении не самих чисел, а массива соответствующих им индексов. Такой подход способен увеличить производительность алгоритмов вычисления простых чисел. В работе [4] реализован и исследован *индексный алгоритм*, построенный с использованием кольцевой факторизации для 3# = 6.

Целью настоящей работы является обоснование, разработка и исследование индексных алгоритмов, построенных с применением кольцевой факторизации для приморialов, превышающих значение 6.

Для удобства изложения введем такое понятие, как *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.

По аналогии можно также определить индексные алгоритмы первого и нулевого порядков. А именно, индексный алгоритм первого порядка есть отсев составных чисел из множества вида $\{2k + 1\}$, а нулевого – из чисел вида $\{1\ k + 1\}$, где $k = 0, 1, 2, \ldots$. Очевидно, что эти индексные алгоритмы являются решетами Эратосфена для нечетных натуральных чисел в первом случае и для всех натуральных чисел – во втором.

Сформируем индексный алгоритм второго порядка для нахождения простых чисел в заданном отрезке $[N_{\min}; N_{\max}]$, исходя из результатов работы [4]. Для этого применим кольцевую факторизацию для 3# к отрезку $[1; N_{\max}]$, *введем 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 занесем эти данные в таблицу 1:

Таблица 1

Результаты кольцевой факторизации для 3# в табличном виде

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 в множества вида $\{6k + 1\}$ и $\{6k + 5\}$, содержащие как простые, так и составные числа, а также единицу.

Перейдем 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 [4], 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 1, вычислив массив их индексов с помощью следующих 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 $\{q_{6k}^{+5}\}$. Для заданного отрезка $[N_{min}; N_{max}]$ из вышеупомянутых последовательностей нужно выбирать те их члены, которые индексируют числа, содержащиеся внутри этого отрезка. Пример для $q_{6\cdot1}^{+1} = 7$ приведен в показательных частях таблицы 1, сформированных в таблицу 2, 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 2

Пример адресации составных чисел при $q_{6\cdot1}^{+1} = 7$

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 $\{q_{6k}^{+5}\}$.

Для заданного отрезка $[N_{min}; N_{max}]$ из вышеупомянутых последовательностей нужно выбирать те их члены, которые индексируют числа, содержащиеся внутри этого отрезка. Пример для $q_{6\cdot0}^{+5} = 5$ приведен в таблице 3, в левом столбце которой представлены индексы $m \cdot q_{6\cdot0}^{+5}$, где $m = 1, 2, 3, \ldots$, а во втором и третьем – 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 3

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.

Индексный алгоритм произвольного порядка

Сформируем индексный алгоритм произвольного порядка n для нахождения простых чисел в заданном отрезке $[N_{min}; N_{max}]$, обобщив изложенный алгоритм второго порядка.

Применим кольцевую факторизацию для $p_n\#$

к отрезку [1; $N_{\max}$], *примем обозначения*, аналогичные принятым выше $q^{+1}_{p_n\#\cdot k} = p_n\#\cdot k + 1$, $q^{+p_{n+1}}_{p_n\#\cdot k} = p_n\#\cdot k + p_{n+1}$, … $q^{+p_{n+s}}_{p_n\#\cdot k} = p_n\#\cdot k + p_{n+s}$, где $k = 0, 1, 2, \ldots k_{\max}$ (для максимально возможного $k = k_{\max}$ должно быть выполнено условие $q^{+1}_{p_n\#\cdot k_{\max}} \le N_{\max}$), и занесем эти данные в таблицу 4:

Таблица 4

Результаты кольцевой факторизации для $p_n\#$ в табличном виде

Индекс	$q^{+1}_{p_n\#\cdot k}$	$q^{+p_{n+1}}_{p_n\#\cdot k}$	…	$q^{+p_{n+r}}_{p_n\#\cdot k}$	…	$q^{+p_{n+s}}_{p_n\#\cdot k}$
0	1	p_{n+1}	…	p_{n+r}	…	p_{n+s}
1	$p_n\#\cdot 1 + 1$	$p_n\#\cdot 1 + p_{n+1}$	…	$p_n\#\cdot 1 + p_{n+r}$	…	$p_n\#\cdot 1 + p_{n+s}$
…	…	…	…	…	…	…
$k_{\max}$	$p_n\#\cdot k_{\max} + 1$	$p_n\#\cdot k_{\max} + p_{n+1}$	…	$p_n\#\cdot k_{\max} + p_{n+r}$	…	$p_n\#\cdot k_{\max} + p_{n+s}$

В левом столбце таблицы приведена последовательность индексов 0, 1, 2, … $k_{\max}$, в двух других столбцах – отображения этой последовательности в множества вида $\{p_n\#\cdot k + 1\}$, $\{p_n\#\cdot k + p_{n+1}\}$, … $\{p_n\#\cdot k + p_{n+r}\}$, … $\{p_n\#\cdot k + p_{n+s}\}$, содержащие как простые, так и составные числа, а также единицу.

Использованные обозначения: $p_1 = 2$, $p_2 = 3$, … p_n – записанные в порядке возрастания простые числа; $p_n\#$ – приориал (произведение всех простых чисел, меньших либо равных p_n); p_{n+1}, … p_{n+r}, … p_{n+s} – записанные по возрастанию числа из интервала (p_n; $p_n\#$), являющиеся простыми или всевозможными произведениями простых чисел из этого же интервала; s – количество простых чисел и их произведений на интервале (p_n; $p_n\#$); $r = 1, 2, 3, \ldots s$.

Перейдем к следующему этапу – отсеву оставшихся после кольцевой факторизации составных чисел из множеств $\{q^{+1}_{p_n\#\cdot k}\}$, $\{q^{+p_{n+1}}_{p_n\#\cdot k}\}$, … $\{q^{+p_{n+r}}_{p_n\#\cdot k}\}$, … $\{q^{+p_{n+s}}_{p_n\#\cdot k}\}$, где $k = 0, 1, 2, \ldots k_{\max}$. Кратные числу q составные числа можно исключить, вычислив их индексы, для чего обобщим полученные для индексного алгоритма второго порядка соотношения на произвольный порядок. При этом $q \in \{Q\} = \{p_n\#\cdot i + 1\} \cup \{p_n\#\cdot i + p_{n+1}\} \cup \cup \ldots \cup \{p_n\#\cdot i + p_{n+r}\} \cup \ldots \cup \{p_n\#\cdot i + p_{n+s}\}$, где $i = 0, 1, 2, \ldots i_{\max}$ (для максимально возможного $i = i_{\max}$ должно быть выполнено условие $(q^{+1}_{p_n\#\cdot i_{\max}})^2 \le N_{\max}$), $r = 1, 2, 3, \ldots s$. Чтобы вычислить эти индексы, введем для фиксированного $q \in \{Q\}$ числа t^j_q, где $j = 1, 2, \ldots (s + 1)$. Числа t^j_q определяются в диапазоне [0; $q - 1$] и принимают значения из первого столбца таблицы 4 соответственно тому, на какой строке в (j + + 1)-ом столбце находится кратное q число. Это означает, что для каждого q определяется свой *паттерн*

$$t^j_q, q \in \{Q\}, \quad (1)$$

под которым будем понимать схему размещения кратных q чисел в строках с индексами 0, 1, … ($q - 1$) таблицы 4.

Покажем, что этот паттерн повторяется в строках с индексами $m\cdot q$, … , $(m + 1)\cdot q - 1$, где $m = 0, 1, 2, \ldots$. Число, стоящее в (j + 1)-ом столбце таблицы 1 на строке с индексом

$$m \cdot q + t^j_q, \quad (2)$$

при $j > 1$ равно

$$p_n\#\cdot(m\cdot q + t^j_q) + p_{n+j-1} = \\ = p_n\#\cdot t^j_q + p_{n+j-1} + p_n\#\cdot m\cdot q, \quad (3)$$

или (при $j = 1$)

$$p_n\#\cdot(m\cdot q + t^j_q) + 1 = p_n\#\cdot t^j_q + 1 + p_n\#\cdot m\cdot q, \quad (3^*)$$

а значит, отличается от стоящего в (j + 1)-ом столбце таблицы 4 на t^j_q-ой строке числа (при $j > 1$ – $p_n\#\cdot t^j_q + p_{n+j-1}$, или при $j = 1$ – $p_n\#\cdot t^j_q + 1$ по определению кратного q) на $p_n\#\cdot m\cdot q$ и, следовательно, делится на q. Очевидно, что для определенного отрезка [$N_{\min}$; $N_{\max}$] из последовательностей (2) нужно выбирать те их члены, которые индексируют числа, содержащиеся внутри этого отрезка в таблице 4.

Рассмотрим в качестве примера отсеивание кратных $q = q^{+7}_{30\cdot 0} = 7$ чисел при выполнении индексного алгоритма третьего порядка. Третий порядок индексного алгоритма означает, что нужно взять 3 первых простых числа и перемножить их: 2 × 3 × 5 = 30, после чего составить таблицу кольцевой факторизации, содержащую все простые числа и состоящую из членов прогрессий {30k + 1}, {30k + 7}, {30k + 11}, {30k + 13},

{30k + 17}, {30k + 19}, {30k + 23}, {30k + 29}. Для того чтобы отсеять все числа, кратные 7, нужно в диапазоне индексов [0; 6] определить паттерн (1) для числа 7:

$$t_q^j = \{3,0,5,4,2,1,6,3\}, \text{ для } j = 1, \ldots 8, q = 7. \quad (4)$$

Далее следует во множестве диапазонов индексов {[7; 13], [14; 20], …} исключить соответственно паттерну (4) кратные 7 числа, которые будут находиться по формуле (2) на тех же местах, что и в диапазоне [0; 6]. Описанный процесс показан в таблице 5, полученной из таблицы 4 при n = 3:

Таблица 5

Пример адресации составных чисел при $q = q_{30\cdot 0}^{+7} = 7$

k	30k + 1	30k + 7	30k + 11	30k + 13	30k + 17	30k + 19	30k + 23	30k + 29
0	1	7	11	13	17	19	23	29
1	31	37	41	43	47	49	53	59
2	61	67	71	73	77	79	83	89
3	91	97	101	103	107	109	113	119
4	121	127	131	133	137	139	143	149
5	151	157	161	163	167	169	173	179
6	181	187	191	193	197	199	203	209
7	211	217	221	223	227	229	233	239
8	241	247	251	253	257	259	263	269
9	271	277	281	283	287	289	293	299
10	301	307	311	313	317	319	323	329
11	331	337	341	343	347	349	353	359
12	361	367	371	373	377	379	383	389
13	391	397	401	403	407	409	413	419
14	421	427	431	433	437	439	443	449
15	451	457	461	463	467	469	473	479
16	481	487	491	493	497	499	503	509
17	511	517	521	523	527	529	533	539
18	541	547	551	553	557	559	563	569
19	571	577	581	583	587	589	593	599
20	601	607	611	613	617	619	623	629
…	…	…	…	…	…	…	…	…

Если в заданном отрезке [N_{min}; N_{max}] с помощью соответствующих паттернов (1) отсеять составные числа (3) и (3*), кратные всем подходящим q (1), то в нем останутся только простые числа. Отметим, что количество паттернов равно количеству элементов множества {Q}, которое зависит от значения N_{max}. Следовательно, на отрезках равного размера с различным N_{max} количество паттернов будет больше у отрезка с большим N_{max}. Перейдем к исследованию полученного алгоритма.

Результаты

Изложенный алгоритм реализован на языке программирования C++ с использованием библиотеки GMP для работы с большими числами и проверен на совпадение результатов генерации с первыми 50 миллионами простых чисел [5]. 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. Увеличившееся время работы по сравнению с [4] обусловлено 64-битным представлением чисел в упомянутой статье. Использование представления mpz_t из GMP позволяет судить о поведении алгоритмов за пределом 64-битного представления.

На рис. 2–5 представлены результаты работы алгоритмов для отрезков, указанных в подписях. Из рис. 2 видно, что для миллионного отрезка индексный алгоритм второго порядка работает быстрее остальных, а четвертого – сильно отстает. Однако уже на отрезке $5\cdot10^7$ (рис. 3) видно, что индексный алгоритм третьего порядка по времени работы практически сравнялся по скорости с алгоритмом второго, а на больших отрезках – стал быстрее. Аналогичная ситуация повторяется на отрезке 10^9 (рис. 4): индексный алгоритм четвертого порядка по времени работы сравнялся с индексным алгоритмом третьего порядка, а на отрезке $4\cdot10^9$ (рис. 5) – стал быстрее.

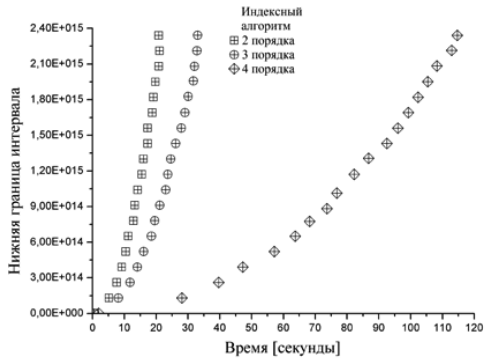

Рис. 2. Сравнение скоростей работы индексных алгоритмов генерации простых чисел на отрезке размером 10^6 (считая от значения ординаты)

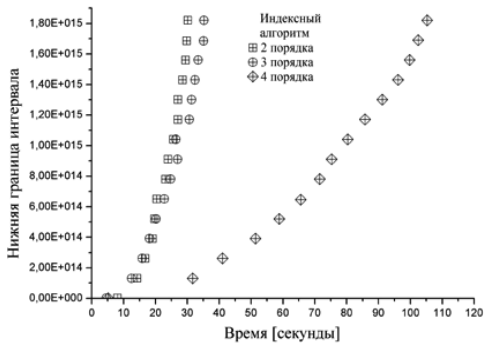

Рис. 3. Сравнение скоростей работы индексных алгоритмов генерации простых чисел на отрезке размером $5 \cdot 10^7$ (считая от значения ординаты)

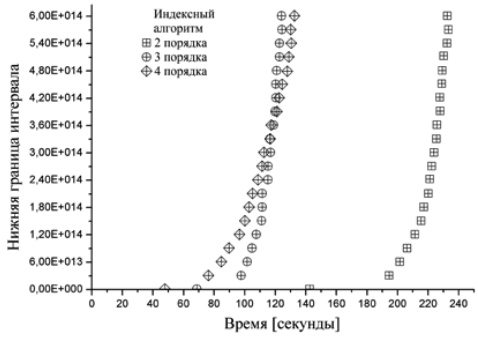

Рис. 4. Сравнение скоростей работы индексных алгоритмов генерации простых чисел на отрезке размером 10^9 (считая от значения ординаты)

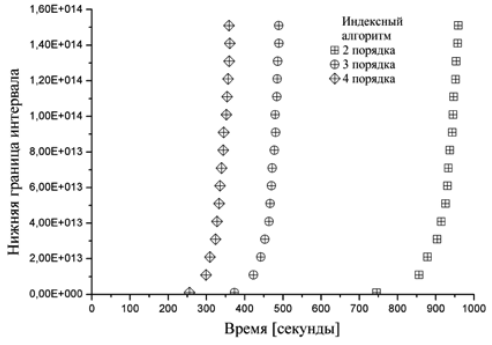

Рис. 5. Сравнение скоростей работы индексных алгоритмов генерации простых чисел на отрезке размером $4 \cdot 10^9$ (считая от значения ординаты)

Таким образом, для бóльших отрезков индексные алгоритмы генерации простых чисел большего порядка работают лучше, чем для меньших. Напротив, для меньших отрезков индексные алгоритмы генерации простых чисел меньшего порядка работают лучше, чем для бóльших. Происходит это потому, что для большего отрезка увеличивается время определения соответствующих ему паттернов, согласно которым отсеиваются составные числа. Для меньшего отрезка наблюдается обратная ситуация – паттерны находятся быстрее.

Выводы

Индексный алгоритм в его текущей реализации представляет принципиально новый подход к поиску простых чисел. Однако требуется его оптимизация применительно к объему кэш-памяти используемого процессора. В то же время, индексный алгоритм с использованием кольцевой факторизации нетребователен к вычислительным ресурсам, поскольку он оперирует с булевым массивом. Благодаря своей многозадачной структуре алгоритм легко модифицируется как многопоточный с реализацией на GPU.

Литература

1. Wheel factorization – [Электронный ресурс]. URL: http://primes.utm.edu/glossary/xpage/WheelFactorization.html

2. Wheel factorization – [Электронный ресурс]. URL: http://en.wikipedia.org/wiki/Wheel_factorization

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

4. Минаев В.А., Васильев Н.П., Лукьянов В.В., Никонов С.А., Никеров Д.В. Высокопроизводительный алгоритм генерации простых чисел в произвольном диапазоне // Материалы XIV Международной научной конференции «Цивилизация знаний: проблемы и смыслы образования». – 2013. – М. : РосНОУ.

5. The first fifty million primes – [Электронный ресурс]. URL: http://primes.utm.edu/lists/small/millions