Annales Univ. Sci. Budapest., Sect. Comp. 43 (2014) 181–191

ON THE GRAPHS OF PRIMES DETERMINED BY LINEAR FUNCTIONS

János Kasza (Budapest, Hungary)

Communicated by Antal Járai

(Received February 20, 2014; accepted March 20, 2014)

Abstract. In this article graphs mapping prime numbers into the prime factors of their linear functions will be examined with computer programs [9] developed by the author. The main question is what can be said about the cycles in these graphs. First the $F(p) = 2p \pm 1$ functions, later the general $F(p) = ap + b$ functions will be analyzed.

1. Introduction

Let $F(n)$ denote a linear function in the $F(n) = an + b$ form, where $a \in \mathbb{N}$, $b \in \mathbb{Z}, (a, b) = 1$.

Let F^k be the iterates of F, where $F^1(n) = F(n)$ and $F^k(n) = F(F^{k-1}(n))$.

Let G_F denote the directed graph over the prime vertices connected to the edges in a way that each p prime node has a directed edge to the $q_1, \ldots, q_r$ nodes representing the prime numbers for that $F(p) = ap + b = q_1^{\alpha_1} \cdots q_r^{\alpha_r}$. The purpose of this article is to study these G_F graphs for some predefined a and b values and to determine their directed cycles – at least for all primes up to a limit.

In case of $a = 1$, the directed cycles of G_F are most probably b-dependent finite sets.

Key words and phrases: Number theory, prime, sieve, graph, primality, factorization.
2010 Mathematics Subject Classification: 11-04, 11Y11, 68W99.
https://doi.org/10.71352/ac.43.181

The $a = 2$, $b = \pm 1$ cases seem to be more interesting than others, because the p, $2p \pm 1$, $4p \pm 3$, ... numbers are the elements of a Cunningham chain of the first or second kind if all the consecutive numbers are primes starting with p. In the next Section the $a = 2$, $b = 1$ case will be analyzed first.

2. The $F(p) = 2p + 1$ case

2.1. First approach

To discover the cycles let us see the following algorithm as a first approach. For an arbitrary prime p – call it a starting prime – test the primality of $p_k := 2^k p + 2^k - 1$ starting from $k = 0$ until we find a composite p_k. Factorize that number, go through its factors and apply the same calculation iteratively what was done for p. In the following, let p_{k+1} denote the greatest prime factor of the iterated (composite) number. Create a directed graph in each step until there are no new primes left to be added to the graph. As a last step, apply the depth-first search (DFS) algorithm to detect its cycles [3]. Because of the time-consuming factorization, primality testing and graph operations, this algorithm can only be used as a starting point.

All graph figures in this paper were created with this approach implemented in C++ using our own number theory library [9] and the Boost Graph Library (BGL) [7] together with Graphviz [8], but to study greater numbers another approach is needed.

2.2. Second approach

The second algorithm starts like the first one, but instead of creating graphs it records only the current path and checks the recurrent elements in every step. When there are no new primes left to be processed, every cycle including the p starting point is discovered. This algorithm works fine, especially when any particular number is in the focus of the study and essential for smaller numbers that might be special cases of the generalizations below.

However, it still takes much time to find the cycles for every prime number in a given interval using this algorithm. The bulk of the time has to be spent on factorization and primality testing, but this can be reduced significantly.

2.3. Further improvements

The most important observation is the prerequisite to have (at least some) strictly increasing p_k prime sequences where p_k is the largest prime factor of

$F(p_{k-1})$ and $p_0 = p$, otherwise it is not possible to find any new cycles. Let us suppose that p_1 can be divided by 3, the smallest prime that can be a real divisor of an odd number. As $p_1/3 = (2p+1)/3 < p$ for every p prime, p_1 must be prime to consider it relevant and to go on with the next iteration. The case of p_2 is quite similar, except that $p_2 = 4p+3$ cannot be divided by 3, thus the smallest possible divisor is 5. As $(4p+3)/5 < p$ for every (odd) p prime, p_2 also must be prime to continue. There is a difference at $p_3 = 8p+7$ because both 3 and 5 can be its divisor, and both $(8p+7)/3 > p$ and $(8p+7)/5 > p$ if $p \geq 5$. As a result, p_3 does not have to be prime, it is sufficient to be the product of a prime and 3 or 5.

To sum up, both p, $2p+1$ and $4p+3$ needs to be prime to have strictly increasing p_k numbers. Thus the set of $p < P$ starting primes can be reduced by the order of $log(P)^2$ based on the Bateman-Horn conjecture [1], where P denotes the upper limit of the interval to be checked.

This observation gives the basis of the program used to check the cycles up to 2^{48}. To determine the reduced set of primes for which p, $2p+1$ and $4p+3$ are primes, it is recommended to use the sieve of Eratosthenes [2] with a modification: not only the solutions of the $x \equiv 0 \pmod{s}$ congruence are the numbers to be sieved out but also the solutions of $x \equiv 2p+1 \pmod{s}$ and $x \equiv 4p+3 \pmod{s}$ congruences, where s denotes the sieving prime. For this special sieve, the sieving primes needs to reach $\sqrt{4P+3}$ instead of $\sqrt{P}$. Thus the result of the sieve will be exactly the reduced set of primes. However, this sort of sieving takes three times longer than a usual sieve, but it reduces the number of factorizations and primality tests to a great extent.

For the reduced set of primes the factors of $8p+7$ need to be examined first. Fortunately, it is sufficient to check whether the number has any small factors up to a number-specific limit. The reason behind that is if a number has only small factors and its greatest prime factor u is greater than p, u needs to be iterated as well. In other words, the same examination needs to be done recursively for $2u+1$ until its greatest factor will be less than or equal to p. The smaller factors can be detached by trial divisions and after that, a deterministic version of the Miller–Rabin primality test described in [4], see also [6] and [5] can be used to determine if the remaining number is prime. If it is prime, it is the greatest factor of the examining number that (and only that) needs to be iterated further. The number-specific limit mentioned above is the minimum of p and the greatest possible number v for that $\lfloor u/v \rfloor > p$.

The cycles found with the described algorithm of strictly increasing iterations is almost equivalent to the first and second variant. Let us suppose that there is a p starting prime and the $q < p$ prime is an element of a cycle of p. Then, the enhanced algorithm stops when p_k falls below p and may not find q. But because q is an element of a cycle of p, there must be another starting prime what already reached p and q. The fact that only one factor

of the p_k's can be greater than p is taken advantage of. However, we cannot guarantee that after some incrementation and stepping back the chain will not increment above p^2, but its probability is very low and depends on the size of p. The bigger the p, the lower the probability, because for an l long iteration the $2^l p + 2^l - 1 > p^2$ inequality should be satisfied.

In the other case when $p < q$ and q is an element of a cycle of p, the algorithm reaches q by definition.

It is important to note that during the trial divisions it always has to be checked whether the p starting prime is a divisor of the iterated number or not. If so, the p is part of a cycle.

2.4. Statistics

The purpose of this paper was to collect every cycles up to 2^{48}. On the other hand, it seems to be also interesting to create statistics on how long the routes go above the starting primes. Because the routes need to start with Cunningham chains of the first kind, it was evident to collect the incremented chains and its lengths. These values can be seen on Figure 1.

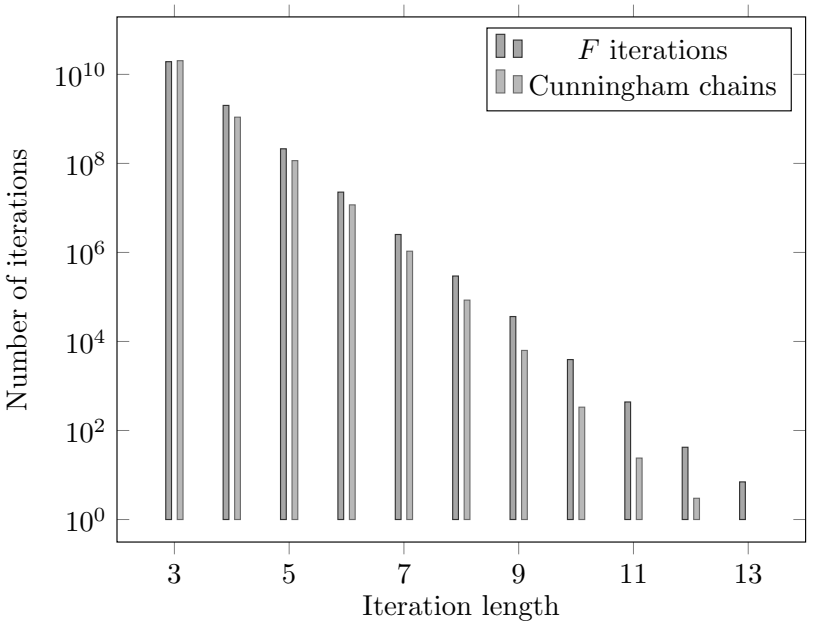

Figure 1. Number of iterations up to 2^{48}

Length	№ of F iterations	№ of Cunningham chains
3	19 295 197 476	20 320 683 726
4	2 008 345 618	1 092 882 580
5	212 992 947	115 612 273
6	22 643 793	11 709 518
7	2 529 529	1 065 672
8	295 207	84 857
9	36 298	6 289
10	3 922	334
11	437	24
12	42	3
13	7	0

Table 1. Number of route lengths up to 2^{48}

Using the implementation of the first approach, up to 2^{30} only two kind of cycles have found: the cycles of 3 ([3, 7, 3], [3, 7, 5, 11, 23, 47, 19, 3] and [3, 7, 5, 11, 23, 47, 19, 13, 3]) and the cycles of 5 ([5, 11, 23, 47, 5], [5, 11, 23, 47, 19, 3, 7, 5] and [5, 11, 23, 47, 19, 13, 3, 7, 5]). Most likely, these are the only cycles that can be produced by the iterates of the $F(p) = 2p + 1$ function. As it is seen on Figure 2, the iterates of 3 and 5 form a closed subgraph.

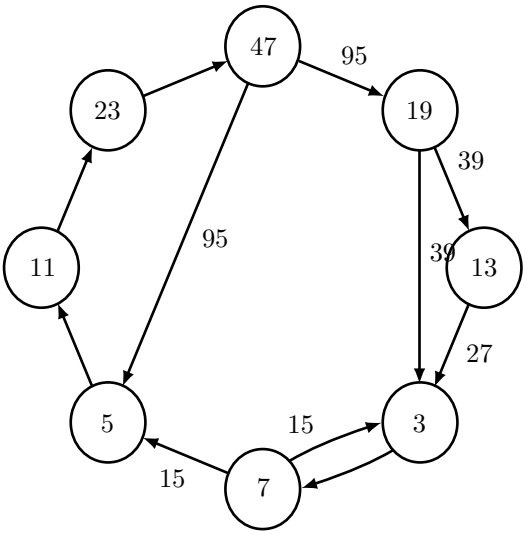

Figure 2. The graph of the iterates of 3 and 5

The lengths of the incremented routes have been collected up to 2^{48} and

the maximum length is 13. The first route with that length starts with the number 554 688 278 429.

Number	Length
5	4
11	5
359	6
179	7
89	8
14 145 539	9
16 911 299	10
2 966 476 949	11
260 559 395 669	12
554 688 278 429	13

Table 2. First occurrences of the found route lengths

Number	Attribute
554 688 278 429	p
1 109 376 556 859	p_1
2 218 753 113 719	p_2
4 437 506 227 439	p_3
8 875 012 454 879	p_4
17 750 024 909 759	p_5
35 500 049 819 519	p_6
71 000 099 639 039	p_7
142 000 199 278 079	p_8
284 000 398 556 159	p_9
568 000 797 112 319	p_{10}
1 136 001 594 224 639	p_{11}
119 579 115 181 541	$p_{12}/19$

Table 3. First occurrence of a 13 long route

3. The $F(p) = 2p - 1$ case

All the programs developed for the $F(p) = 2p + 1$ case can be used easily for the $F(p) = 2p - 1$ case with some obvious modifications. Therefore, let us see the results up to without further explanation.

Here also, we have found two kinds of cycles up to 2^{34}: the cycle of 3 (3, 5, 3) and the cycle of 19 (19, 37, 73 29, 19).

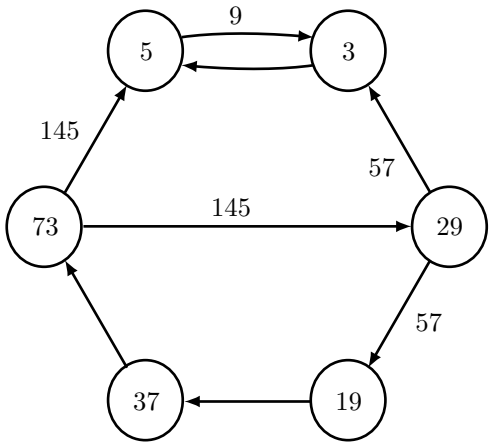

Figure 3. The graph of the iterates of 3 and 19

4. The $F(p) = ap + b$ case

To look into the $F(p) = ap + b$ case, some limitations need to be defined first. The most important is that a and b have to be coprime, otherwise the results will be dependent. For negative b the $F(p)$ values can be negative as well. As it is not defined how to interpret these numbers when applying on F, it was decided to stop the iteration for the negative numbers. Then, because the computations are based on 64-bit arithmetic, every p_k value needs to be less than 2^{64} and maybe the values greater than that limit are ignored. The algorithm for proving the primality can be used only for numbers below 341 550 071 728 321 [5]. Above that limit, we cannot be sure whether the number is a prime or a strong pseudoprime for several bases. For these reasons, there is a risk that the results are not absolutely precise.

The main goal was to compare the different (a, b) values by the number of cycles. For that, the algorithm described as the first approach was used with some experimenting of the sets of a, b and p. The first observation is that in most cases the algorithm finds every cycle of (a, b) with smaller p starting primes as well ($p < 100$). Taking this into account, the starting primes were used in the range of 2 to 10000. In the following some cycle statistics can be seen where a is between 0 and 100, and b is between -50 and 50.

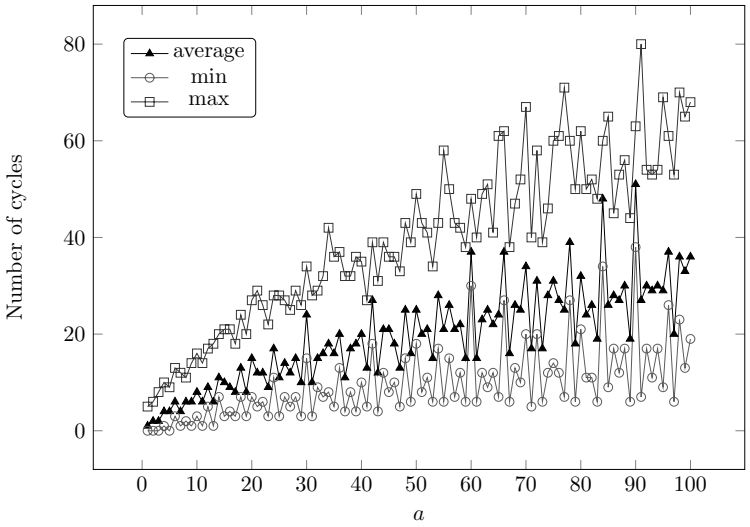

Figure 4. $F(p) = ap + b$ where $b \in [-50..50]$, $p \in [2..10000]$

On Figure 4 the minimum, maximum and average number of cycles are shown. The tendency in all three diagrams is increasing. The maximum number of cycles is found in the $a = 91$, $b = 30$ case where 80 cycles was counted. The top ten cases with the most of cycles was studied further with starting primes up to 1 000 000 to see whether additional cycles could be found or not. The results are the following: at $a = 98$, $b = 45$ the 71. cycle starts with 228731 as a new cycle; at $a = 100$, $b = -21$ the 69. new cycle starts with 15647; at $a = 70$, $b = 33$ the 69. new cycle starts with 35507, and probably there are no new cycles.

To have a closer look on the tendency of the maximum cycle values, an additional statistic was created with smaller starting primes where p is less then 100, but a is between 0 and 1000, and b is in the range of -50 and 50. The resulting diagram is on Figure 5. It is possible that not every cycle was collected, especially for the bigger a's, because the 2^{64} limit can be reached in a few steps even though the $F(p_k)$ values are composites very often what results fallbacks. To have an overview on the number of overflows, see Figure 6. The first overflow occurred at $a = 132$ (and $b = 25$), and 5022 overflows have been recorded in total.

Finally, the 3-dimensional distribution of cycles can be seen in Figure 7.

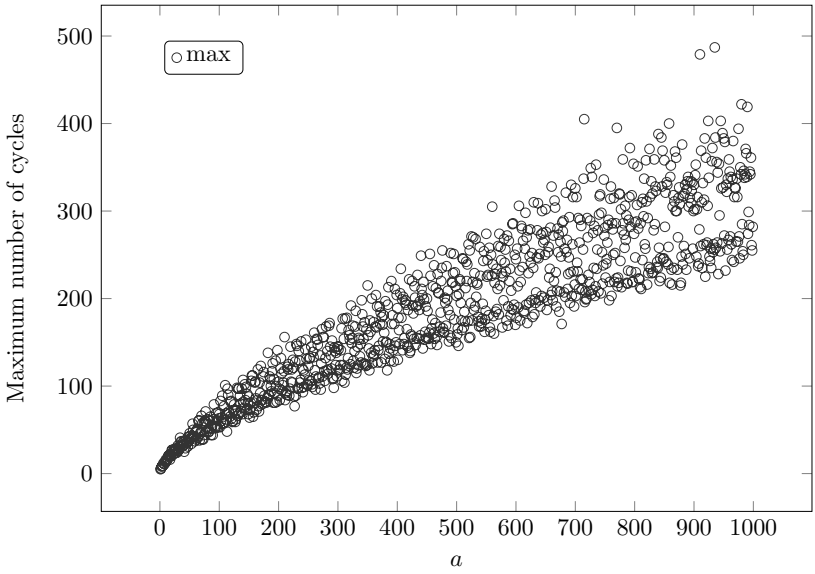

Figure 5. $F(p) = ap + b$ where $b \in [-50..50]$, $p \in [2..100]$

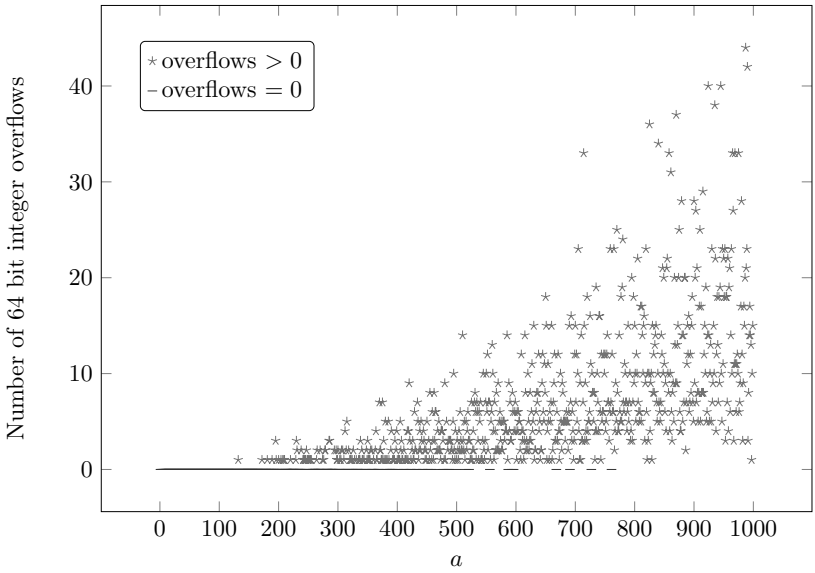

Figure 6. $F(p) = ap + b$ where $b \in [-50..50]$, $p \in [2..100]$

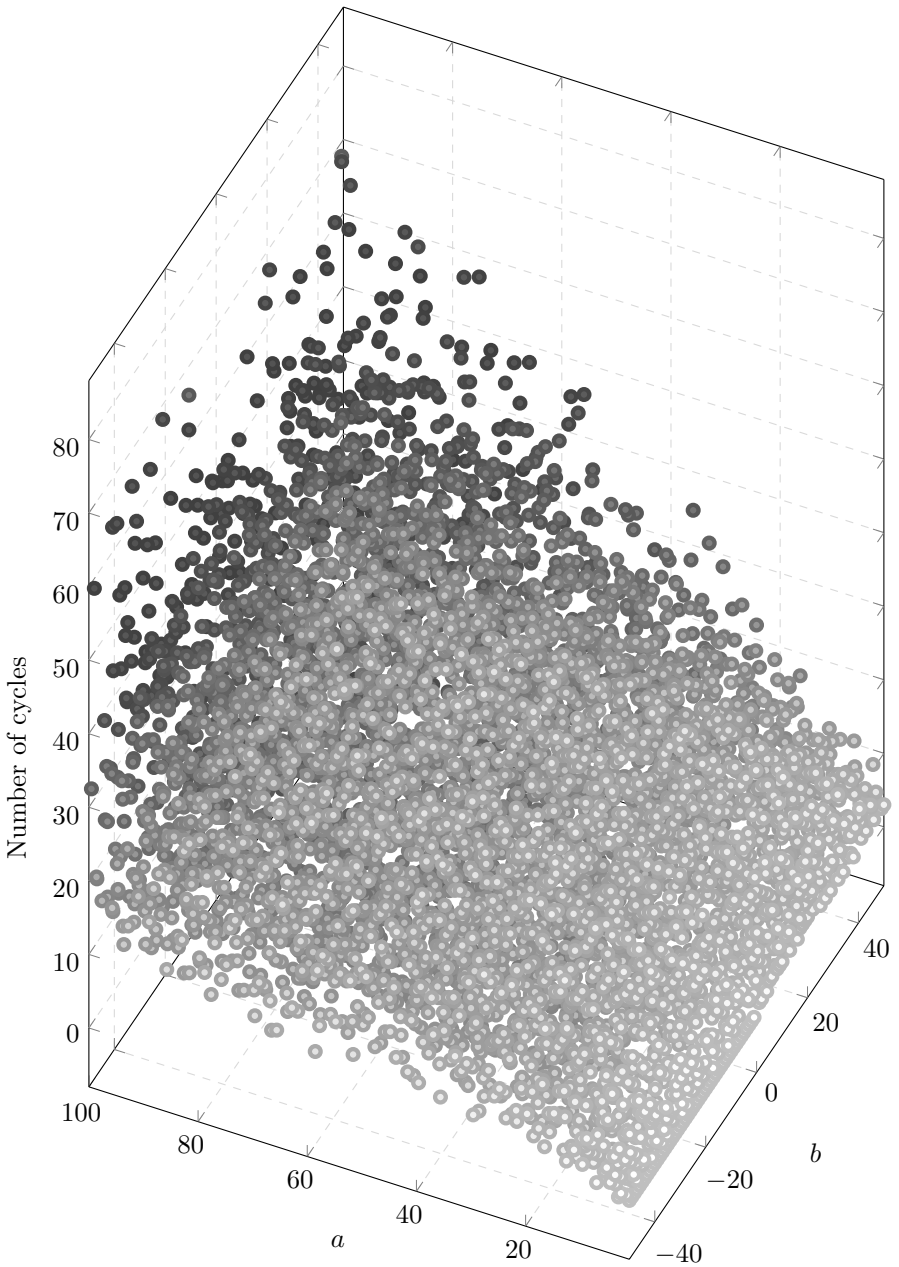

Figure 7. Distribution of cycles

Acknowledgement. I would like to give special thanks to Antal Járai, professor at Eötvös Loránd University, Department of Computer Algebra, for his guidance and help.

I would also like to thank Judit Cziczelszky for providing writing assistance.

References

[1] **P. T. Bateman and R. A. Horn**, A heuristic asymptotic formula concerning the distribution of prime numbers, *Math. Comp.* **16** (1962), 363–367.

[2] **Bressoud, D.M.,** *Factorization and Primality Testing*, Springer-Verlag, ISBN 978-0387970400, 1989.

[3] **Cormen, T.H., C.E. Leiserson, R.L. Rivest and C. Stein,** *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, ISBN 0-262-03293-7 (2001), 540–549.

[4] **Csajbók, T., A. Járai and J. Kasza,** On representing integers as quotients of shifted primes, *Annales Univ. Sci. Budapest, Sect. Comp.* **28** (2008), 157–174.

[5] **Jaeschke, G.,** On strong pseudoprimes to several bases, *Math. Comp.* **61** (1993), 915–926.

[6] **Pomerance, C., J.L. Selfridge and S.S. Wagstaff Jr,** The pseudoprimes to $25 \cdot 10^9$, *Math. Comp.* **35** (1980), 1003–1026.

[7] **The Boost Graph Library (BGL)**, `http://www.boost.org/doc/libs/release/libs/graph/`

[8] **Graphviz**, `http://www.graphviz.org/Documentation.php`

[9] **Open source programs developed by Janos Kasza**, `https://github.com/janoskk/computational-number-theory`

J. Kasza
Eötvös Loránd University
Department of Computer Algebra
H-1117 Budapest
Pázmány Péter sétány 1/C
Hungary
janos.kasza@compalg.inf.elte.hu