Annales Univ. Sci. Budapest., Sect. Comp. 35 (2011) 83–93

ITERATING THE TAU-FUNCTION

Tímea Csajbók (Budapest, Hungary)
János Kasza(Budapest, Hungary)

Dedicated to Professor Antal Járai on the occasion of his 60th birthday

Abstract. For every natural number greater than 2, the sequence generated by iterating the tau-function is a strictly monotone decreasing sequence, it stabilizes and at the end reaches 2. The second but last value of the sequence is an odd prime. The question of Imre Kátai is what is the asymptotic distribution of these primes, if any.
Our goal was to analyze every tau-iteration sequence of all natural numbers up to a given bound. We also analyzed the tau-iteration sequence for randomly chosen set of large numbers. For calculating the tau-function, efficient factorization methods are necessary.

Tau-function. Let $n = p_1^{\alpha_1} p_2^{\alpha_2} ... p_r^{\alpha_r}$, where $r \in \mathbb{N}, \alpha_i > 0$ integer, $p_i > 0$ prime and $p_i \neq p_j$ if $i \neq j$. Let $\tau(n)$ denote the number of positive divisors of n. Then $\tau(n) = (\alpha_1 + 1)(\alpha_2 + 1) \cdots (\alpha_r + 1)$.

It is evident that $\tau(1) = 1, \tau(p) = 2$ and $\tau(n) < n$ if $n \geq 3$.

Tau-iteration. Consider the iterated sequence n, $\tau(n)$, $\tau^{(2)}(n) = \tau(\tau(n))$, , ..., where $n > 2$. This is a strictly monotone decreasing sequence until reaching 2 and stabilizing (it cannot reach 1). The value before 2 is an odd prime. We will call this number lasttau(n) from now on.

n	$\tau(n)$	$\tau^{(2)}(n)$	$\tau^{(3)}(n)$	lasttau(n)
$64 = 2^6$	7	2	2	7
$2541 = 3 \cdot 7 \cdot 11^2$	12	6	4	3
$3003 = 3 \cdot 7 \cdot 11 \cdot 13$	2^4	5	2	5

Table 1 – Examples for the iteration

As it is clear from the examples, the most difficult part is the first factorization. Since we want to work with 50–60-digit long numbers, we have to find efficient methods of tolerable running times.

Small factors $(2, 3, \ldots, 9973)$ can be found using trial division. Beyond that the Pollard ρ method is used up to 10^6.

For finding even larger factors, we use elliptic curves. Roughly speaking, the running time of the elliptic curve factorization depends only on the length of the second largest prime factor. This method is appropriate for finding factors of about 20–30 digits.

To guarantee that each found factor is prime, the Miller–Rabin primality test is used after these methods.

Elliptic curves. An elliptic curve over $\mathbb{R}$ is the set of all (x, y) pairs on the plane satisfying $y^2 = x^3 + ax + b$, where a and b are real constants and $4a^3 + 27b^2 \neq 0$.

It is obvious that if any point (x, y) is on the curve, then so is $(x, -y)$. The condition for the constants guarantees that a definite tangent exists at every point of the curve. If a (non-vertical) line intersects the curve at two points, (x_1, y_1) and (x_2, y_2), then it intersects the curve at a third point (x_3, y_3) as well. If slope of the line is $\lambda = (y_1 - y_2)/(x_1 - x_2)$ then it is not hard to prove that $x_3 = \lambda^2 - x_1 - x_2$ and $y_3 = \lambda(x_3 - x_1) + y_1$. We can define the addition operation by the formula $(x_1, y_1) + (x_2, y_2) = (x_3, -y_3)$. If the line is tangent to the curve then we consider the line to intersect the curve at two equal points, i. e., $x_1 = x_2$ and $y_1 = y_2$. In this case $\lambda = (3x_1^2 + a)/(2y_1)$. If the line is vertical we consider the third intersection point to be in the infinity; this point will be the zero element of the addition. With this addition operation the points of the elliptic curve form an Abelian group.

We can define elliptic curves over any field having characteristic different from 2 and 3. Even more generally, we can define "elliptic curves" but only with a partial addition operation above a commutative ring with identity element, for example, above $\mathbb{Z}/n\mathbb{Z}$ if $\gcd(n, 6) = 1$ and $\gcd(n, 4a^3 + 27b^2) = 1$. For any prime divisor p of n we also get an elliptic curve modulo p. If an addition is defined over $\mathbb{Z}/n\mathbb{Z}$ then it is also defined for any prime divisor p of n. A key observation here is that for any prime divisor p of n, doing the addition modulo n and reducing the result modulo p is the same as reducing the addends modulo p first and then adding the results modulo p. To factorize n we use "elliptic curves" over $\mathbb{Z}/n\mathbb{Z}$. Roughly speaking, for some point P on the curve, we calculate $k! \cdot P$ for a rather large k. During this calculation the gcd operation to compute λ will with high probability find a non-trivial factor of n.

We can use projective representation: Let the points of the curve be represented as equivalence classes of triplets (X, Y, Z) above $\mathbb{Z}/n\mathbb{Z}$. Point (X, Y, Z)

is equivalent to all points (cX, cY, cZ) where c has an inverse modulo n. The zero element of the "curve" is the equivalence class of $(0, 1, 0)$. In this representation the equation of the curve becomes the homogeneous equation

$$ZY^2 = X^3 + aXZ^2 + bZ^3.$$

First we tried the approach described as follows. We select a random curve above $\mathbb{Z}/n\mathbb{Z}$ with a random point P on it by choosing random x, y, a values and calculating b from them. Then we check that $\gcd(n, 4a^3 + 27b^2) = 1$ holds. If it does, we calculate $k! \cdot P$ for increasing values of k. If it is not successful, we have found one of the divisors of n.

We carry out the multiplication by $k!$ iteratively, by multiplying $Q = (k - 1)! \cdot P$ by k. We calculate kQ by another iteration starting from Q and $2Q$. The basic idea is to use only the X and Z coordinates. Let i be the number represented by the first l bits of multiplier k. After the lth step we have the X and Z coordinates of the points iQ and $(i+1)Q$. If the next bit, i. e., the $l+1$st bit of k, is zero then we calculate the X and Z coordinates of the points $2iQ$ and $(2i+1)Q$. If the next bit is one then we calculate the X and Z coordinates of $(2i+1)Q$ and $(2i+2)Q$. Therefore we need only two operations: duplication and the calculation of the X and Z coordinates of $(2i + 1)Q$ from the X and Z coordinates of iQ, $(i + 1)Q$ and Q.

The above approach could be more efficient with changing the curve parameter determination and calculation of coordinates of the new points. Therefore we switched to the representation proposed by Montgomery [1]:

Let the curve equation in homogeneous coordinates be

$$Y^2Z = X^3 + aX^2Z + bXZ^2 + cZ^3, \tag{1}$$

the two points of the curve $P_1 = (u_1/w_1^2, v_1/w_1^3)$ and $P_2 = (u_2/w_2^2, v_2/w_2^3)$, where $u_1/w_1^2 \neq u_2/w_2^2$.

Then $P_3 = P_1 + P_2$, where $P_3 = (u_3/w_3^2, v_3/w_3^3)$ can be determined the following way:

$$\begin{aligned}
u_3 &= (v_2w_1^3 - v_1w_2^3)^2 - aw_1^2w_2^2(u_2w_1^2 - u_1w_2^2)^2 \\
&\quad - (u_1w_2^2 + u_2w_1^2)(u_1w_2^2 - u_2w_1^2)^2, \\
v_3 &= -v_1w_2^3(u_2w_1^2 - u_1w_2^2)^3 - (v_2w_1^3 - v_1w_2^3)u_3 \\
&\quad + w_2^2(u_2w_1^2 - u_1w_2^2)^2u_1(v_2w_1^3 - v_1w_2^3), \\
w_3 &= w_1w_2(u_2w_1^2 - u_1w_2^2).
\end{aligned}$$

For the duplication $2P_1 = (u_3/w_3^2, v_3/w_3^3)$, the corresponding coordinates have

to be determined as well:

$$u_3 = (3u_1^2 + 2au_1w_1^2 + bw_1^4)^2 - 4(aw_1^2 + 2u_1)v_1^2,$$
$$v_3 = -8v_1^4 - (3u_1^2 + 2au_1w_1^2 + bw_1^4)(u_3 - 4u_1v_1^2),$$
$$w_3 = 2v_1w_1.$$

In this approach the calculation of kQ where $Q = (k-1)!P$ is simply done by employing the left-to-right binary method using only duplication and addition of Q.

It seems that the determination of the coordinates requires a lot of multiplication. If we determine the starting point and the parameters of the curve in an appropriate way, the above calculations can be simplified. Let the starting point of the curve be $(1, \alpha, -1)$, where the constants of the curve (1) are $a = 0$, $b = 0$, and $c = \alpha^2 - 2$. With this selection, we can save many calculations. There is only one curve parameter, α, which is selected by random for each curve.

The effieciency of the factorization depends on the number of iterations and the number of curves. The suggested values are the following [10]:

Digits	**Number of iterations**	**Number of curves**
15	2000	25
20	11000	90
25	50000	300
30	250000	700
35	1000000	1800
40	3000000	5100
45	11000000	10600
50	43000000	19300
55	110000000	49000
60	260000000	124000
65	850000000	210000
70	2900000000	340000

Table 2 – Suggested values for number of iterations and curves

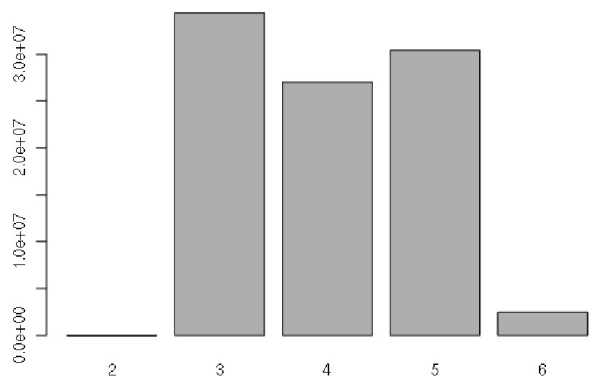
These values served well as good starting points for selecting the actual parameters. During the tests we had to tune them for finding the given length of factors.

With this simple flow control, we could find the lasttau(n) values:

procedure *lasttau*
  $t, last, i \leftarrow \tau(factors), -1, 1$
  while $(t \neq 2)$
    $last \leftarrow t$
    $ECM(t, factors)$
    $t \leftarrow \tau(factors)$
    $i \leftarrow i + 1$
  end
end

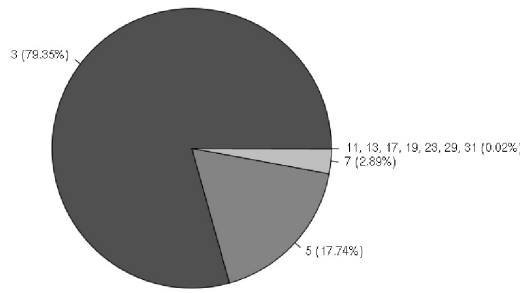
The implementation of the described methods has been done in C and C++ languages, with GNU GMP [12] multi-word arithmetic and with Condor workload management system. The program was run on a cluster of 64-bit AMD processors for several months.

In the next figure we can see how many times it is necessary to iterate the τ function for numbers up to 10^8 to get the lasttau(n) values. We can see that the most frequent value is 3 and it is never required to iterate more than 6 times.



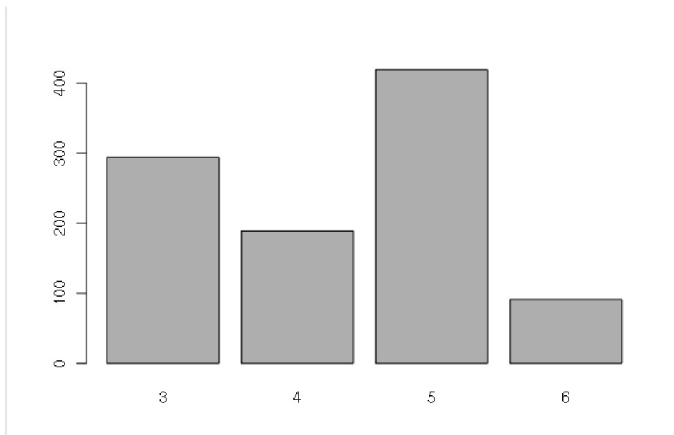
Required number of iterations for lasttau(n) calculations up to $n = 10^8$

The next diagram shows the distribution of lasttau values up to $n = 10^8$. The biggest lasttau value is 31. The occurrences of 3, 5 and 7 are the highest.



The ratio of lasttau(n) values up to $n = 10^8$

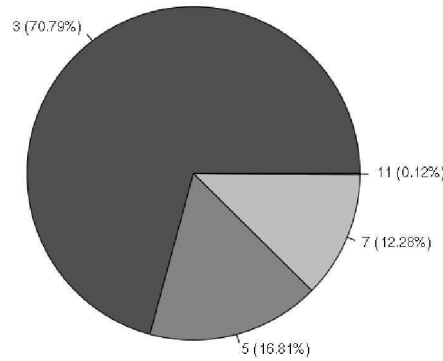
Let us see these ratios for numbers around 10^{50}. We chose randomly 1000 numbers and the distribution is the following:



Required number of iterations for calculating lasttau(n) for n around 10^{50}

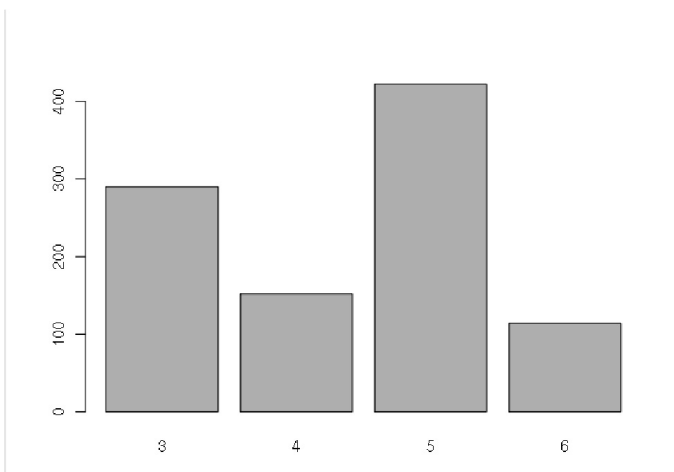
We can see that in this random sample the most frequent τ-iteration length is 5 and the most infrequent is 6.

The next diagram shows that the greatest lasttau value is 11 and the occurrence ratio is very similar to the case of smaller numbers.



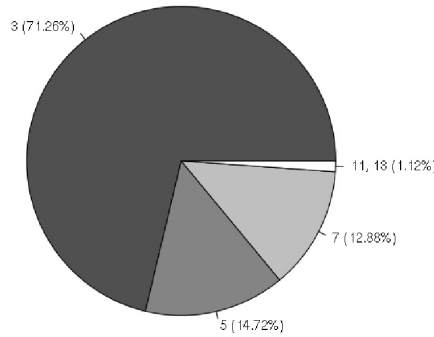
Ratio of lasttau(n) values for n around 10^{50}

Next, we chose the numbers in the interval $[10^{70}, 10^{70} + 1000)$. The distribution is still very similar to before. The most frequent iteration length in this case is also 5, and the most infrequent is also 6.



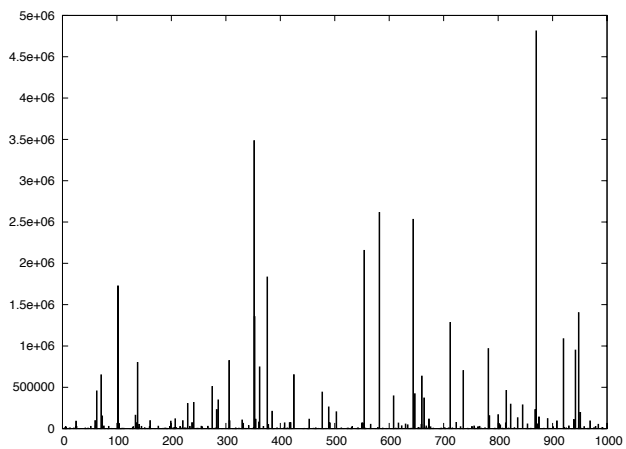
Required number of iterations for calculating lasttau(n)
between 10^{70} and $10^{70} + 1000$

If we analyze the occurrences of lasttau(n) values, we will see that 11 and 13 are the most frequented ones. The distribution of smaller primes is very similar to previous samples.



Ratio of lasttau(n) values between 10^{70} and $10^{70} + 1000$

The last diagram shows the time of factorization of 1000 numbers in seconds. We can see that there are extremely high values, and sometimes it was done very quickly. It depends on the number of curves that we are not able to determine any factor.



Factorization time for numbers between 10^{70} and $10^{70} + 1000$

Let us have a closer look at some numbers of this sample. In Tables 3 and 4 we can see for each n considered what its factors are, the value of lasttau(n) value (L), the number of iterations necessary (I), and the time the calculation took in minutes.

Number	Factors	L	I	Time
$10^{70}+1$	$29, 101, 281, 421, 3541, 27961, 3471301, 13489841, 121499449, 60368344121, 848654483879497562821$	3	6	0
$10^{70}+2$	$2, 3, 2417, 728771, 331844753, 1315441529, 216757689503480567071658372879852581111205513$	3	5	0
$10^{70}+3$	$7, 103, 13869625520110957004160887656033287101248266296809986130374479889043$	3	4	0
$10^{70}+4$	$2^2, 465761, 708845197, 735374140501601, 167342219028631 33, 615334987861198431900041$	3	6	2
$10^{70}+5$	$3, 5, 23, 109, 307, 297674527070399026203749, 2909875173333171111479969227134609531967$	3	5	142
$10^{70}+6$	$2, 1663, 147227767, 2384032997, 856595459059852666856707432875358753198266455623119$	3	5	0
$10^{70}+7$	$1621, 252073541474195849351629035309353, 24473141552192478478666014277117939$	3	4	424
$10^{70}+8$	$2^3, 3^2, 17, 16156512259, 567538992676497479567 63, 8909949913446915151529019420144601$	3	5	177
$10^{70}+9$	$233, 176144029, 243655462971284241469045332244275022188090622768643397118037$	3	4	0
$10^{70}+10$	$2, 5, 7, 11, 13, 47, 139, 2531, 31051, 143574021480139, 5497971844491917, 246494453476490591927 45899$	13	3	0
$10^{70}+11$	$3, 53, 433, 525404597, 8990767439, 5310944858510134877 59, 57896532578451563713869329$	3	5	2
$10^{70}+12$	$2^2, 8539, 119855917, 194568691, 125545328972902714113373191107146251387155936901 91$	3	5	0
$10^{70}+13$	$8009, 1097408550449, 166777194373804297 1066279, 6822078352993308595836369024 67$	5	3	36
$10^{70}+14$	$2, 3, 79, 191, 716848837, 304976983972605 1129, 505235247019871798150323203407222 78177$	3	5	46
$10^{70}+15$	$5, 19, 211, 586764512320294166 03, 2067397236615734055061, 4112502436486078502075149$	7	3	104
$10^{70}+16$	$2^4, 241, 259336009958506224066390041493775936360995850622406639004149377593361$	3	5	0
$10^{70}+17$	$3^4, 7, 409, 43121477514305550165370866267361784884197272135332445030896538639$	3	5	0
$10^{70}+18$	$2, 3881, 1454569, 183339272189671009, 4830994366338537465366958647013901853403609$	3	5	4
$10^{70}+19$	$674810659, 22393508323, 661753013393221095707734371385920431993559884567867$	3	4	0
$10^{70}+20$	$2^2, 3, 5, 127, 503, 3323, 101281, 299375505968569225 49, 258941975440540758891541779098500261$	3	6	1
$10^{70}+21$	$11, 4405961469831 7, 206332015228820139698611876104805217385963574297204818 83$	3	4	0
$10^{70}+22$	$2, 3697, 5281, 244671994467759, 12166397653726906185 13, 860316231948847300773028 69$	7	3	100
$10^{70}+23$	$3, 13, 9781, 7558907689, 35376899347, 1717124714191, 5709150444675349089768255266864 9$	3	5	0
$10^{70}+24$	$2^3, 7, 227, 563, 139726159084380068566420785886987246914008050462381805642030522 9$	7	3	0

Table 3 – Detailed results 1.

Number	Factors	L	I	Time
$10^{70}+976$	$2^4, 7, 73, 146477, 260671, 3369520352\overline{3}, 22445454879651, 42354588581183665588373211019\overline{61}$	5	4	7
$10^{70}+977$	$3, 17, 1410560625788052551233\overline{1}, 139007446956961142853460943668632702576932010017$	5	3	163
$10^{70}+978$	$2, 11, 167, 1181, 3192803, 721836481699563776769530010565242353651989830516110056379$	7	3	30
$10^{70}+979$	$10427, 21770568487823\overline{17}, 4405253010740831444480036113063386307215609595\overline{83}181$	3	4	40
$10^{70}+980$	$2^2, 3^2, 5, 241, 17948784330\overline{7}, 2839336706572166661400834\overline{67}, 45233325335895776847343238\overline{09}$	3	6	426
$10^{70}+981$	$59, 997, 808789, 3174287, 116108104\overline{3}, 57030721932015325310947921649073543009340003$	7	3	30
$10^{70}+982$	$2, 263, 477130943, 428411743423, 72740235062492\overline{33}, 12786173883475811425505044447661$	7	3	4
$10^{70}+983$	$3, 7, 41, 43, 139, 3862987, 5030258992164628464280563151245949189733352882682399756\overline{97}$	3	5	50
$10^{70}+984$	$2^3, 19, 83, 899209660\overline{9}, 42631426682162\overline{87}, 20676998140581370242980844143596327408253$	3	5	11
$10^{70}+985$	$5, 13, 2426789, 2186649450090759728942714\overline{9}, 2899181871473416749766870669758359929$	3	5	844
$10^{70}+986$	$2, 3, 109, 70849621, 3107608370, 92571301451963\overline{9}, 75020865170392960011074730724814605\overline{3}$	3	5	5
$10^{70}+987$	$29, 31, 8467, 1143561852296878\overline{79}, 1148817622934842465157527562245657691825339154\overline{1}$	3	5	11
$10^{70}+988$	$2^2, 67, 3731432835820895522388059701492537\overline{3}, 134328358208955223880597014925\overline{41}$	3	5	0
$10^{70}+989$	$3^4, 11, 37, 5835672122537, 5197921699628309518763130135876780402222845375\overline{83}491$	3	5	0
$10^{70}+990$	$2, 5, 7, 8392231, 17022546550153690614910045\overline{11}877030757886158553752188865426334\overline{7}$	3	5	0
$10^{70}+991$	$3393413, 7417051783859089773, 39731226287784319853841258581915322272573759$	3	4	80
$10^{70}+992$	$2^5, 3, 2383770887, 3087434689256921902709, 14153585608977250699315120651200019319$	3	6	101
$10^{70}+993$	$71, 1747, 7121, 249881, 587082217973913357287, 771585550672089772879565046423694\overline{7}$	7	3	153
$10^{70}+994$	$2, 17, 23, 7481, 65649718831\overline{7}, 26037586263260274679903884135037844823179832091385\overline{71}$	7	3	0
$10^{70}+995$	$3, 5, 97, 935096727371, 7349883741973753135321282883048735652618172438758967559$	3	5	0
$10^{70}+996$	$2^2, 1303, 1038237911603423576\overline{33}, 184798614023848590441592869381179719634126855\overline{51}$	3	5	10
$10^{70}+997$	$7, 47, 1036751, 8644661, 3391420742120581294422583829954267536281389505336658\overline{63}$	3	5	0
$10^{70}+998$	$2, 3^2, 13, 16831280293, 25390250765900387612738713681786782803436403228861270128379$	3	5	2
$10^{70}+1000$	$2^3, 5^3, 11, 909090909090909090909090909090909090909090909091$	7	3	0

Table 4 – Detailed results 2.

References

[1] **Niven, I., H.S. Zuckerman and H.L. Montgomery,** *An Introduction to the Theory of Numbers*, Wiley, 1991.

[2] **Zimmermann, P. and B. Dodson,** In: *20 Years of ECM*, Springer Berlin, Vol. **4076**, 2006, pp. 525–542.

[3] **Montgomery, P.L.,** Speeding the Pollard and elliptic curve methods of factorization, *Mathematics of Computation*, Vol. 48, 177, 1987, pp. 243–264.

[4] **Bressoud, D.M.,** *Factorization and Primality Testing*, Springer-Verlag, 1989.

[5] **Crandall, R.E.,** *Topics in Advanced Scientific Computations*, Springer TELOS, 1995.

[6] **Niven, I. and H.S. Zuckerman,** *Bevezetés a számelméletbe*, Műszaki Könyvkiadó, 1978.

[7] **Lenstra, H.W.,** Factoring integers with elliptic curves, *Annals of Mathematics*, **126** (1987), 649–673.

[8] **Brent, R.P.,** An improved Monte Carlo factorization algorithm, *BIT*, **20** (1980), 176–184.

[9] **Brent, R.P. and J.M. Pollard,** Factorization of the eighth Fermat number, *Mathematics of Computation*, **36** (1981), 627–630.

[10] `http://www.alpertron.com.ar/ECM.HTM`

[11] **Charron, T., N. Daminelli, T. Granlund, P. Leyland, and P. Zimmermann,** The ECMNET Project, `http://www.loria.fr/~zimmerma/ecmnet/`

[12] **Granlund, T.,** GNU MP: The GNU Multiple Precision Arithmetic Library, `http://www.swox.se/gmp/#DOC`

T. Csajbók and J. Kasza
Department of Computer Algebra
Eötvös Loránd University
H-1117 Budapest, Hungary
`timea.csajbok@compalg.inf.elte.hu`
`janos.kasza@compalg.inf.elte.hu`