Uli Meyer

Reconstruction of D.H. Lehmer's number theory computer with LEGO®

In 1927 the american mathematician D.H. Lehmer constructed an electro-mechanical device that was able to find solutions for a system of linear equivalences. This machine, also known as the „bicycle chain sieve“, could be used to factor numbers or to proof a given number to be prime. In 1932 the machine was improved, the chains were replaced by gearwheels, the electro-mechanic stopping mechanism became photo-electric.

Probably this machine was the first „automatic“ factoring device, and it was then possible to factor numbers that were far out of reach for human calculators (even with the use of electro-mechanic calculators able to divide and multiply, that were available a few years earlier). In [1] Lehmer describes the factoring of two numbers with 16 and 17 decimals, but this required some extra math, which was possible, as these numbers had a special form. This sounds not very breath-taking – nowadays we are able to factor numbers of far more than 100 decimals – but it was 80 years ago.

I reconstructed the machine by the use of LEGO®-Bricks and some extra parts. My intention was not to build an historic correct replica of the original machine, but a functioning one. After programming modern factoring algorithms this was a funny and relaxing change... especially it was very interesting to see how hard the work must have been in the 1920s. I tried to factor a 13 digit number without actually even using a pocket calculator, to get the impression....

The reader does not need to be familiar with number theory to understand this text. The operator mod means the rest when dividing numbers, for example

$$17 \, mod \, 10 = 7 \quad \text{or} \quad -1 \, mod \, 23 = 22$$

In many places I use „congruences“:

$$\begin{aligned} & a \equiv b \pmod{m} \\ \Leftrightarrow \; & a \, mod \, m = b \, mod \, m \\ \Leftrightarrow \; & (\exists z \in \mathbb{Z})(a - b = z \cdot m) \end{aligned}$$

What the machine does.

Given $g \in \mathbb{N}$, (the number of chains), and for $i = 1, \ldots, g$ modules $m_i \in \mathbb{N}$ and subsets $S_i \subseteq \{0, 1, \ldots, m_i - 1\}$. The machine runs through $b = 0, 1, 2, 3, \ldots$ and stops for each b that satisfies

$$(\forall i \in \{1, \ldots, g\})(\exists y_i \in S_i)(b \equiv y_i (mod \, m_i)) \tag{1}$$

There is a motor driven shaft with (equal) sprockets for each module and a counter that counts the revolutions of the shaft (indicated in teeth of the sprockets, corresponding to b). Each module is represented by a chain of m_i links, one link of each chain is marked as the zero position. A pin can be put into each chain-link, and the subset S_i is represented by those chain-links, that don't hold a pin. A laser beam is positioned so that, once there is no pin in either of the topmost chain-links of all gear-wheels, it hits a photo-detector that makes the machine stop.

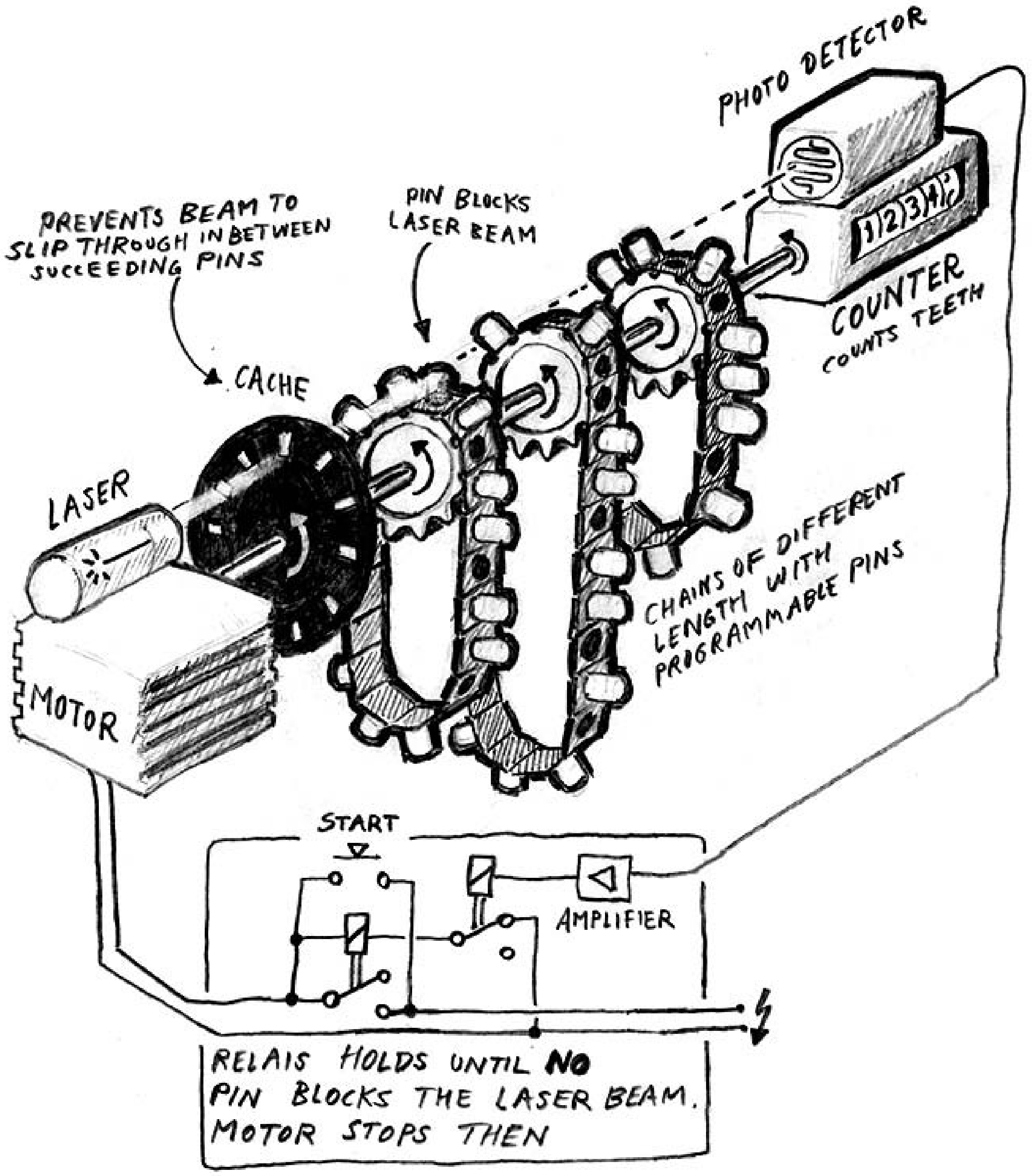


PHOTO DETECTOR
PREVENTS BEAM TO SLIP THROUGH IN BETWEEN SUCCEEDING PINS
PIN BLOCKS LASER BEAM
1 2 3 4
COUNTER
COUNTS TEETH
CACHE
LASER
CHAINS OF DIFFERENT LENGTH WITH PROGRAMMABLE PINS
MOTOR
START
AMPLIFIER
RELAIS HOLDS UNTIL NO PIN BLOCKS THE LASER BEAM. MOTOR STOPS THEN

How it can be used to factor a number

As already Fermat knew, an odd number N has two odd factors, whose difference is even (let's say $2b$), and so N can be expressed as a difference of squares of $a, b \in \mathbb{N}$

$$N = (a+b)\cdot(a-b) = a^2 - b^2 \tag{2}$$

So let's have a closer look to square numbers: 0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169... it looks like that the last digit is always one of 0, 1, 4, 5, 6, or 9 and never 2, 3, 7, or 8. This is because if you multiply, the last digit of the result depends only on the last digit of the multiplied numbers. We can use this fact as a kind of sieve to find candidates for square numbers without having to calculate square roots if the last digit is 2, 3, 7 or 8.

Looking at the last digit means looking at the rest if one divides by 10, and we have similar results for numbers $m_i \neq 10$. Each number is either *quadratic residue* or *non-residue modulo* m_i :

$$x \text{ quadratic residue modulo } m_i \Leftrightarrow (\exists y \in \mathbb{Z})(y^2 \equiv x (mod\, m_i)) \tag{3}$$

Now we can make the sieve more dense, and less non-square numbers will slip through, because a square number must be quadratic residue modulo **all** $m_i \in \mathbb{N}$. To find a factor of N we let the machine run over $b = 0, 1, 2...$ and we need to program the chains, so that it stops if $N + b^2$ is quadratic residue modulo all m_i .

E.g. let $m_i = 5$ and $N \equiv 2 (mod\, 5)$ for $b = 0, 1, 2, 3, 4$ we get the quadratic residues $b^2 \equiv 0, 1, 4, 4, 1$. Now $N + b^2 \equiv 2, 3, 1, 1, 3$ of which only 1 is quadratic residue modulo 5 and so we know that $b\, mod\, 5 \in \{2, 3\} = S_i$.

To stay formally correct, we summarize this

$$S_i = \{b \in \mathbb{Z}_i \mid (\exists a \in \mathbb{Z}_i)(N \equiv a^2 - b^2 (mod\, m_i))\}$$
$$\text{where } \mathbb{Z}_i := \{0, 1, \ldots, m_i\} \tag{4}$$

Now we supply the machine with pins, set the counter to zero and hang in the chains at their zero positions and let it run. It will stop at a possible candidate for b and to see if we have a solution we have to check that $N + b^2$ in fact is square, for example by taking the square root. Otherwise we start it again...

As already Lehmer did, we use different primes as m_i (or if there is a technical minimum for the chain length, a small multiple of the smaller primes). Then we have all m_i relatively prime to each other, and there will be no redundancy. If m_i is a prime, we have roughly $\# S_i \approx \frac{m_i}{2}$. If we furthermore assume the distribution of the elements among S_i to be random, we can estimate that the machine will stop in average every $\prod \frac{m_i}{\# S_i} \approx 2^g$ positions.

If we assume, that no prime less or equal to some boundary d divides N , we have a lower boundary for the smaller factor $a - b > d$, and putting this together with (2), we have an upper limit for the search

$$b < \tfrac{1}{2}\left(\tfrac{N}{d} - d\right) \leq \tfrac{N}{2\mathrm{d}} \tag{5}$$

For this method and fixed m_i , tables for the pin settings are provided in the appendix. Alternatively we could use the machine to run across $a = \sqrt{n}, \sqrt{n} + 1, \ldots$. Though we had to compute different subsets, but in principle its the same.

Now imagine that for some reason for some m_i we had $S_i=\{s\}$. Its obvious that we can make the machine run m_i times faster, if we omit the sprocket for m_i and substitute

$$b=b'\cdot m_i+s \tag{6}$$

But now we need to compute the other subsets S_j' accordingly. We replace $b\in S_j$ by $b'\equiv(b-s)\cdot\frac{1}{m_i}(mod\, m_j)$. Note, that the multiplicative inverse $\frac{1}{m_i}$ does exist if m_i and m_j are relatively prime to each other. We run the machine across $b'=0,1,2,3...$ and when it stops, recalculate b according to (6).

In his examples Lehmer makes use of number theoretic arguments or assumptions based on the special form of N to downsize the computation time this way. But as I found out, one can expect always to speed up the machine by at least factor six, because

$$\begin{aligned}
&N\equiv1(mod\,24) \Rightarrow b\equiv0(mod\,12)\\
&N\equiv5 \lor N\equiv17(mod\,24) \Rightarrow a\equiv3(mod\,6)\\
&N\equiv7 \lor N\equiv19(mod\,24) \Rightarrow b\equiv3(mod\,6)\\
&N\equiv11(mod\,24) \Rightarrow a\equiv6(mod\,12)\\
&N\equiv13(mod\,24) \Rightarrow b\equiv6(mod\,12)\\
&N\equiv23(mod\,24) \Rightarrow a\equiv0(mod\,12)
\end{aligned} \tag{7}$$

LEGO® considerations

The building bricks are quite exact, which was very helpful, because the major problem for proper operation of the machine is the exact adjustment of shaft, chain, pins, and laser beam. I used 24-teeth sprockets and #3873 chain-links from the Sandcrawler set #10144 (the set contains 274 chain-links, enough to build the machine). These chain-links are quite wide, so only three chains fit on one 12-long axle (the longest that exists), but the attached plate has holes that allow pins to fit in. There was no real good solution for making pins within the LEGO system so I decided to make them on my own from two types of plastic tubes, one of diameter 0,32mm glued into others of 0,48mm diameter. When I began to experiment, I tried to use only the smaller tubes, but I couldn't achieve the required exactness of caching and blocking the laser beam. Note, that one chain-link fits into a pair of two teeth, so one revolution of the shaft corresponds to twelve succeeding numbers.

Alternatively one could use the thick 70ies gearwheels, but the vintage chain-links (that have two studs, so one can use 1x2-bricks as pins) are hard to get and the old gearwheels have very weird numbers of teeth and the counting would take some extra effort.

In order not to let the laser beam slip through in between chain-links, there must be a cache disc on the shaft that has 12 rectangular holes which have to correspond exactly to the pin geometry. To achieve this exactness, I supplied a sprocket with 12 chain-links and pins, scanned its sideview and constructed the disc layout over this scan using vector graphic software. Note, that the chain-links are not symmetric (in direction of movement), but slightly biased, and so the pins. One should take this into account, and all chains have to be mounted in the same direction later on. I printed the scan, supplied it with self-adhesive plastic foil, cut the rectangles and the diameter, and punched a hole in the center. I supported the cache disk by two #4185 Technic Wedge Belt Wheels, and when I found it to be adjusted well, punched an extra hole and fixed it all together with a #4274 pin ½.

I used a 3V red laser with adjustable focus, that I had glued into a drilled hole in a 2x2-brick. It is extremely important that you can fix the laser extremly accurate in two dimensions of position and angle. The intial cheap solution was a small clod of dough. Finally I didn't trust the viscosity of the dough for eternity and found a good solution with some large ball joint and #32064 bricks with axlejoint. The laser beam was defined to go exactly through the center of a technic brick hole within the coordinate system. This, and a little piece of white cardboard was very helpful in adjusting the ray. Consequently the ray hits the photo-detector through a technic brick hole, in a kind of „black box“ that protects the detector from ambient light.

The photo-resistor and the electronics is all original „fischer-technik“ from the 1970s. This building system was a bit concurrent to LEGO, more designed for those who wanted things to really work, when LEGO was just rectangular bricks and some ready-made gimmicks. Unfortunately LEGO never got this fischer-technik-spirit, that supplies kids with cool parts like „relais with amplifier“... I mean, this worked all without PC and you don't need a micro-processor to make a motor stop by some signal. And when LEGO started with „mindstorms“, fischer-technik had computer interfaces since years, that could be programmed without kindergarden-software. And by the way, fisher-technik never had bricks with the odd ratio of 5 by 6 in dimensions x,y by z, and remember that 5 and 6 are relatively prime to each other, and what uggly consequences this had for the „universal brick toy“....

Talking about electric: The shaft is driven by the old #2838C01 motor. It wants ~9V. That's approximately what my fisher-technik relais stuff needs. There is one self-holded relais that makes the motor run once the start-button is pushed. The self-holding is interrupted by another relais that is controlled by the amplified photo-signal. If the latter relais switches a control light is illuminated and the motor stops but due to the moment of inertia of the rotating parts it does not stop abruptly but continues rotating a few teeth. There is a „hand wheel“ at the front end of the shaft, and one can rotate the shaft backwards a few numbers until the control light illuminates again to read out the solution on the counter.

The photo-dectector switches the relais if there is enough light energy going in. The higher the motor voltage, the faster the shaft rotates, the shorter the signal (no pin at all chains), the less the light energy. On the other hand, the higher the amplifier energy, the more sensitive the photo-detecting. Although these effects are not counteracting, you might nevertheless need two different voltages for motor and amplifier to achieve the maximum speed, that does neither stop by some erroneous signal (overall geometric inexactness) nor lets a solution slip through. One should test this thoroughly before proceeding. My construction achieved a maximum reliable speed of 3000 numbers per minute (like the original machine).

I used an the industrial counter UK35 from Hilß GmbH (~$50). It has five digits and counts 10 digits per revolution. If a 24-teeth wheel at the shaft drives a 20-teeth #32269 wheel at the counter, it counts twelve numbers when the shaft revolves once. After having made bad experiences with a counter from an old tape recorder, that originated from that those engineers had taken every imaginable effort to put some weird prime-factors into their gears that can't be compensated by lego gearwheels, I was really surprised by the reliability of this industrial counter. Worth its money.

The static part of the construction had to be exact and stable, and therefore the machine must be placed on a stable, level table.

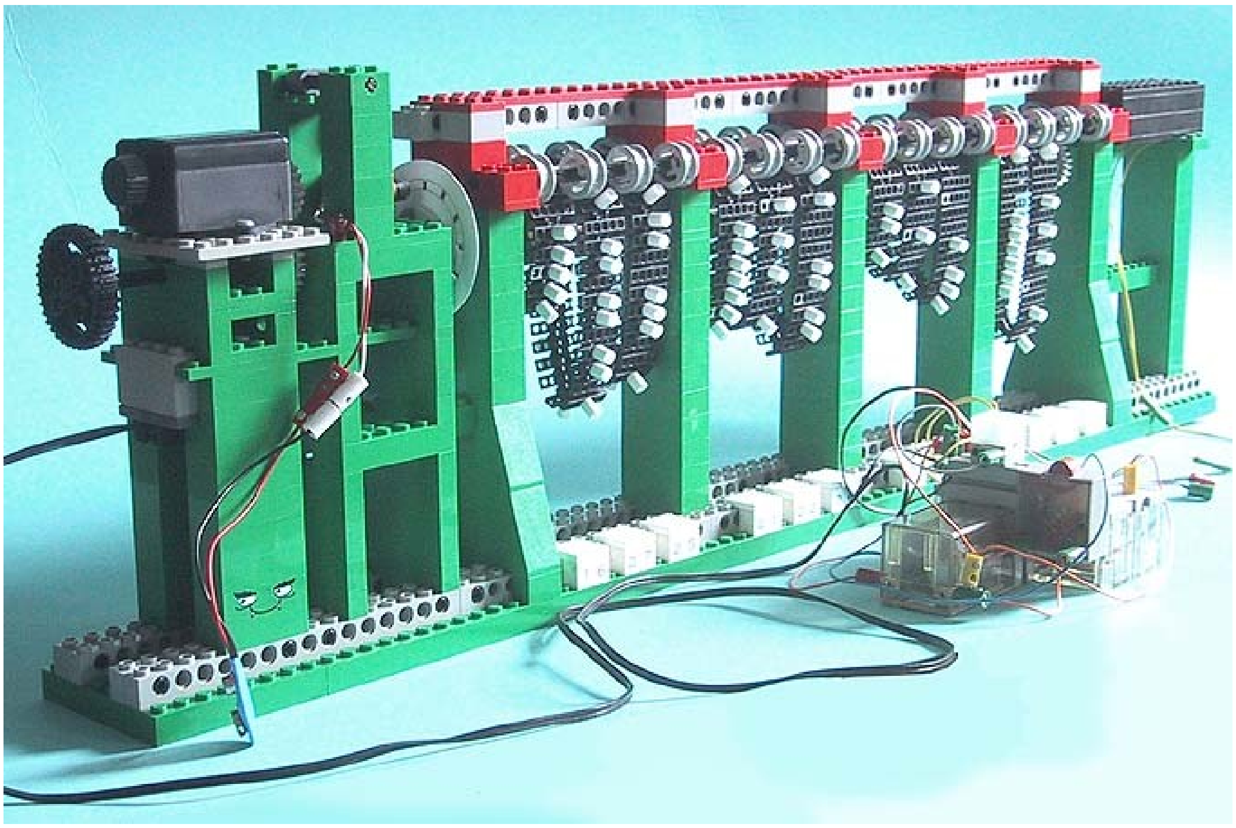
Finally there was this problem: if the shaft suddenly start or stops to rotate, the chains that hang loosely on the sprockets will swing, which might cause that they flip out of the teeth and fall back in a wrong position. I tried to solve this problem by placing loose sprockets into the hanging chain as ballast, but the final solution was a pair of axles with wheels parallel to the shaft, that prevent the chains to dismount. This part of the construction can be removed easily to allow access to the chains.

In the current version the machine has four groups of three sprockets each, with chain lengths of

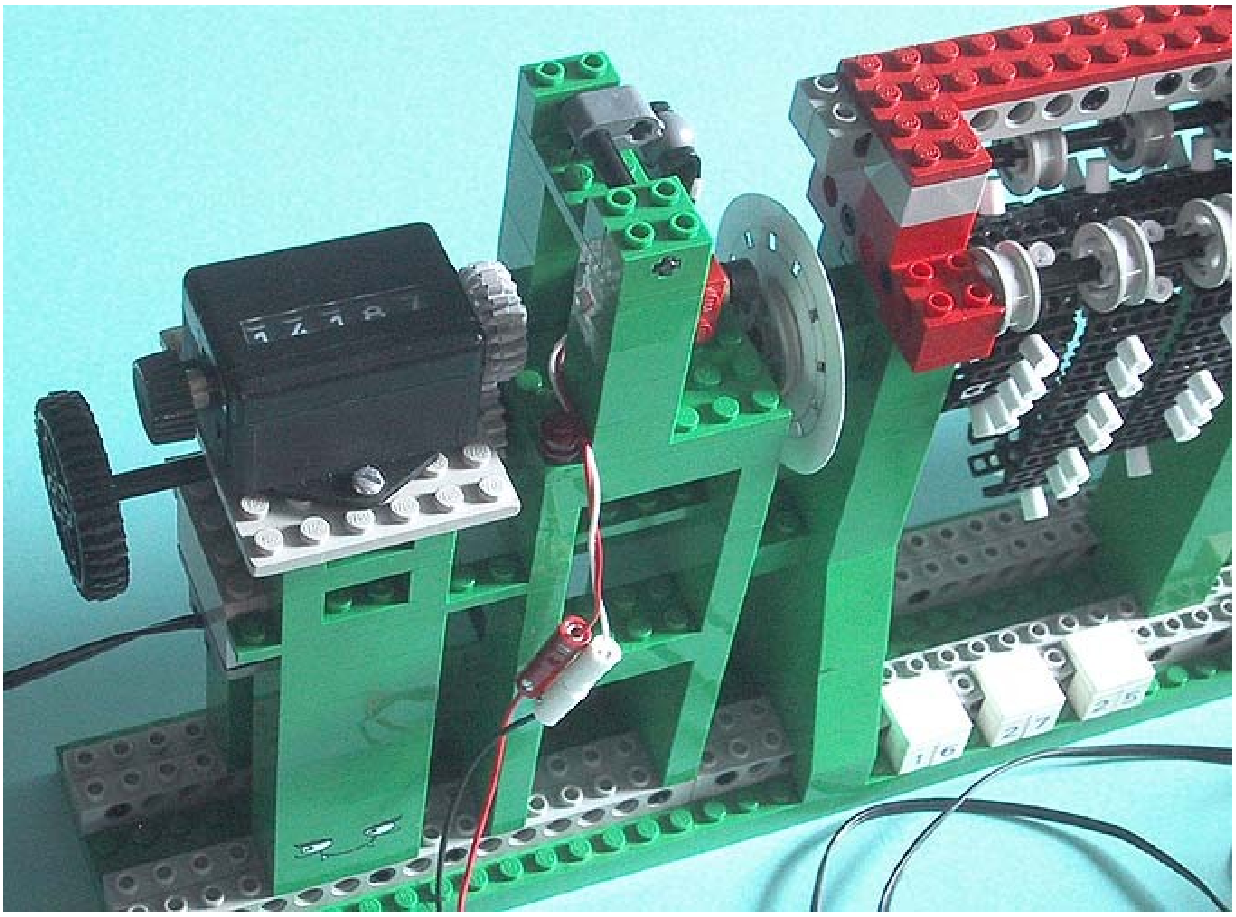
16, 27, 25, 21, 22, 26, 17, 19, 23, 29, 31, (37)

Due to lack of material the 37-link chain was not installed. A look at the „density“ of the tables for the pin-settings in the appendix, shows us, that we can estimate the machine to stop every ~22 000 numbers or every seven minutes. A closer look at the tables shows that this estimate can vary between 6 908 and 39 615 (between 2 and 13 minutes) depending on the individual $N \bmod m_i$.

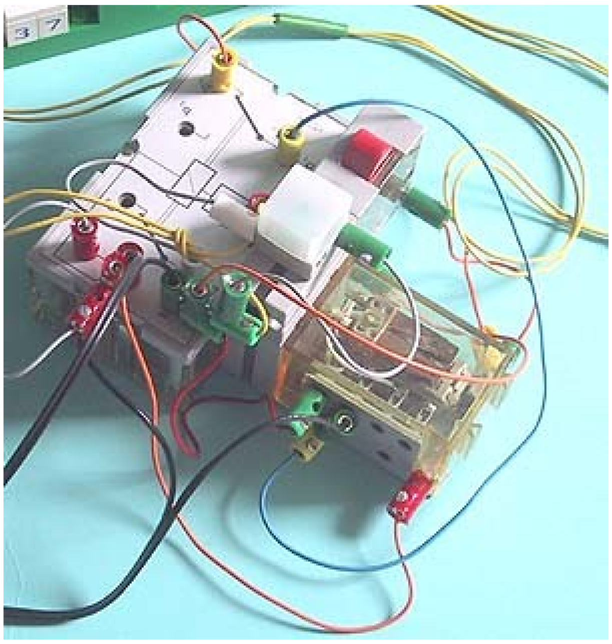
For comparison, Lehmer's machine had 19 sprockets, the chain lengths were 64, 27, 25, 49, 22, 26, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, and 67 corresponding to „a few hours“ between two successive stops.



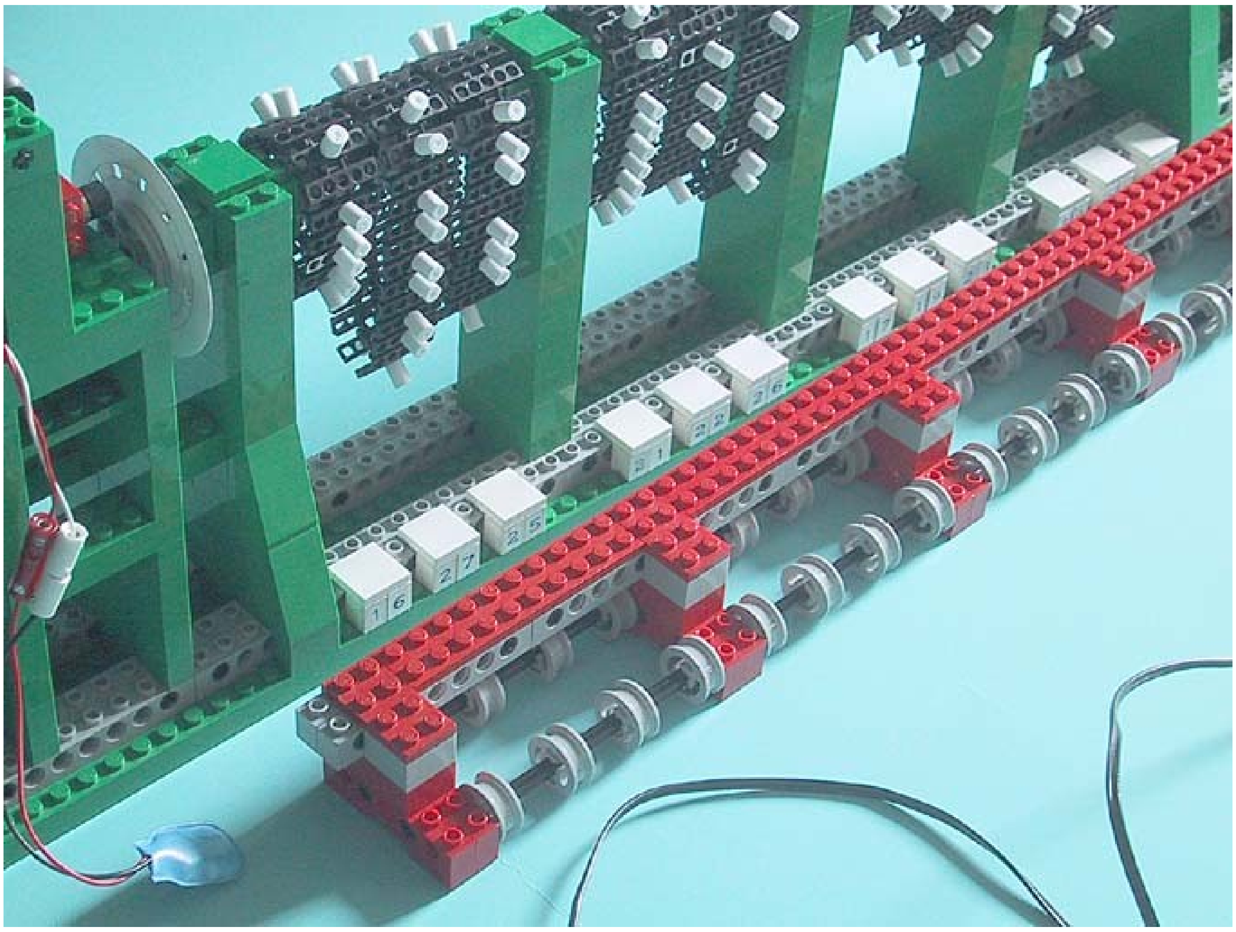
from left to right: handwheel, counter (motor below), ball-joint-fixed laser, cache disc, shaft with sprockets and chains, electric stuff, photo-detector in a black box.



the two fischer-technik relais, the control light and the start button



below: the anti-chain-flip device has been removed for access to the chains. the laser beam goes through the middle hole of the red 1x6 technic bricks.



Testing the machine

As an example I tried to factor a 13-digit number from the *Cunnigham-tables* (see [2])

$$N_{13} = 3^{27}+1 = 7\ 625\ 597\ 484\ 988$$

Three human beings calculated it by „continued squaring“ a few times, and I took the result that occured most frequently. It is almost impossible for human calculators to multiply numbers of this size without errors.

Due to its special form this number can be partially factored algebraically. The exponent 27 has a few odd factors, and with p odd

$$a^{p\cdot q}+1=(a^q)^p+1\equiv(-1)^p+1\equiv 0 \quad (mod\, a^q+1)$$

and a^q+1 divides $a^{pq}+1$ .This way we can show that 3+1 divides 3^3+1 which divides 3^9+1 which divides N_{13} and we let the machine hunt for factors of the 9-digit-number

$$N = \frac{3^{27}+1}{3^9+1} = \frac{N_{13}}{2^2\cdot 7\cdot 19\cdot 37} = 387\ 400\ 807$$

No prime below $d=100$ divides N (as a useful side-effect of these manual divisions I got the $N\, mod\, m_i$ that I needed to go into the tables). So I knew that $b<1\ 936\ 954$, and according to our estimate above, I could expect the machine to stop approximately 100 times within this range. It takes quite a while to manually check one of these resulting $N+b^2$ to be square, and so I was quite pessimistic if I would ever finish this task. But I had fortune. The machine stopped after a few seconds for the first time at $b=243$ and $N+b^2$ indeed was square and we obtain

$$N = 19\ 927 \cdot 19\ 441$$

Proving these factors to be prime again was easy, probe-division with all primes between 100 and $141\approx\sqrt{20000}$ showed that N was completely factored.

Conclusions

The test showed, how hard it must have been to factor numbers in the pre-digital era. The preparation of the test-run cost at least one day of human calculation time (not included the making of the tables). The programming of the pins took half an hour. The machine ran only a few seconds, but even if it had run several hours, the 1920s mathematician must have been quite happy, as long as they just had to wait. I think this was what was really new about this machine: *It was automatic*. The algorithm that the machine represents is a loop with a stopping criterium that is an AND-operation.

If I had tried to factor the number by probe-dividing manually by the first ~2222 primes (those smaller than $\sqrt{N}$), I would have needed weeks, and as I found out, it would have been practically impossible without doing errors. So I understand, that the machine was a break-through. On the other hand I over-estimated the power of the machine. The above example showed, that one should use maybe 19 instead of 11 chains to factor 9-digit numbers and that the machine run will be completed within 24 hours. What I like about the machine is, that the AND operation of the pins is done in parallel and extra chains should not slow it down.

Here I want to give you an idea where we are today: A few years ago I programmed a HMPQS (hypercube multi-polynomial quadra-tic sieve), a factoring algorithm invented in the 1990s (write an email to ulimy at freenet dot de, if you are interested in the software) that is the fastest up to ~110 decimal digits. It can factor an 80 digit number in 24 hours, the 9 digit number from above will be factored in less than one millisecond on an ordinary PC. [2] gives an excellent overview of the history of factoring methods, read [3] for a description of modern factoring algorithm.

Of course I programmed a simulator of the Lehmer-computer, and of course it is much faster than the real machine. One result, when playing with the simulator, was that if you use equations (7) you would need extra chains to compensate for the less density of the pins. But even so, the underlying algorithm of the machine is rather bad, O(N).

[1]
D. H. Lehmer
The mechanical combination of linear forms
in: The American Mathematical Monthly,
Vol. XXXV, 1928

[2]
John Brillhart, D. H. Lehmer et al:
Contemporary Mathematics Vol. 22
Factorizations of b^n±1, b = 2, 3, 5, 6, 7,10, 11, 12, Up to High Powers
Third Edition
American Mathematical Society, Providence, Rhode Island

[3]
Carl Pomerance
A Tale of Two Sieves
in: Notices of the AMS, December 1996

The pin settings

Asterisks * indicate the chain-link positions at which pins have to be placed. The number in parentheses at the end of each line is the number of pins supplied to the respecitve chain. The percentage in the head-line for each chain-length is the average percentage of chain-links supplied with pins.

```
 n mod 16 (75.0%)
 1 -***- ***-* **-** * (12)
 3 *-*** **-*- ***** - (12)
 5 **-** *-*** -***- * (12)
 7 ***-* -**** *-*-* * (12)
 9 -***- ***-* **-** * (12)
11 ***-* -**** *-*-* * (12)
13 **-** *-*** -***- * (12)
15 *-*** **-*- ***** - (12)
```

```
 n mod 27 (75.9%)
 1 -**-* *-**- **-** -**-* *-**- ** (18)
 2 ***** -**** ***-- ***** **-** ** (23)
 4 -**-* *-**- **-** -**-* *-**- ** (18)
 5 **-** **-** ***** ***** -**** -* (23)
 7 -**-* *-**- **-** -**-* *-**- ** (18)
 8 *-*** ***** -**** **-** ***** *- (23)
10 -**-* *-**- **-** -**-* *-**- ** (18)
11 ****- -**** ***** ***** **--* ** (23)
13 -**-* *-**- **-** -**-* *-**- ** (18)
14 ***** **-** *-*** *-*** -**** ** (23)
16 -**-* *-**- **-** -**-* *-**- ** (18)
17 ***** ***-* -**** **-*- ***** ** (23)
19 -**-* *-**- **-** -**-* *-**- ** (18)
20 ****- ***** ***-- ***** ***-* ** (23)
22 -**-* *-**- **-** -**-* *-**- ** (18)
23 **-** ***** *-*** *-*** ***** -* (23)
25 -**-* *-**- **-** -**-* *-**- ** (18)
26 *-*** ***-* ***** ****- ***** *- (23)
```

```
 n mod 25 (66.0%)
 1 -**** -*-** -**** -**-* -**** (18)
 2 **--* **--* **--* **--* **--* (15)
 3 *-**- *-**- *-**- *-**- *-**- (15)
 4 -**** -**** --**- -**** -**** (18)
 6 -**** -**** -*--* -**** -**** (18)
 7 **--* **--* **--* **--* **--* (15)
 8 *-**- *-**- *-**- *-**- *-**- (15)
 9 -***- -**** -**** -**** --*** (18)
11 -**** -**-* -**** -*-** -**** (18)
12 **--* **--* **--* **--* **--* (15)
13 *-**- *-**- *-**- *-**- *-**- (15)
14 -**** --*** -**** -***- -**** (18)
16 -**-* -**** -**** -**** -*-** (18)
17 **--* **--* **--* **--* **--* (15)
18 *-**- *-**- *-**- *-**- *-**- (15)
19 -**** -***- -**** --*** -**** (18)
21 -*-** -**** -**** -**** -**-* (18)
22 **--* **--* **--* **--* **--* (15)
23 *-**- *-**- *-**- *-**- *-**- (15)
24 --*** -**** -**** -**** -***- (18)
```

```
 n mod 21 (75.0%)
 1 -**** *-*** ***** -**** * (18)
 2 ****- **-** --**- **-** * (15)
 4 -**** ****- **-** ***** * (18)
 5 **-*- -**** --*** *--*- * (13)
 8 *-*** **--* ***-- ***** - (15)
10 ***** *-**- **-** -**** * (17)
11 **-** -*-** ****- *-**- * (15)
13 ***-* *-*** ***** -**-* * (17)
16 -**-* ***** ***** ***-* * (18)
17 *--** -**-* ***-* *-**- - (13)
19 ***-* ****- **-** ***-* * (17)
20 *-**- ***-* --*-* **-** - (13)
```

```
 n mod 22 (50.0%)
 1 -*-** --**- *-*-* *--** -* (12)
 3 --*-* ***-* ---*- ****- *- (12)
 5 -*--* ***-- *-*-- ****- -* (12)
 7 **--- **--- ***-- -**-- -* (10)
 9 -***- ---** *-*** ----* ** (12)
13 *-*-* --*-* -*-*- *--*- *- (10)
15 --**- **-** ---** -**-* *- (12)
17 ***-- ----* ****- ----- ** (10)
19 *--** --**- -*--* *--** -- (10)
21 *--*- **-*- -*--* -**-* -- (10)
```

```
 n mod 26 (50.0%)
 1 -**-- -**-- -**-* *---* *---* * (12)
 3 --*-* *--** -*--- *-**- -**-* - (12)
 5 **--* -**-* --*** --*-* *-*-- * (14)
 7 ***-- *--*- -**** *--*- -*--* * (14)
 9 ---*- ****- *---- -*-** **-*- - (12)
11 *-**- -**-- **-*- **--* *--** - (14)
15 *-*** ----* **-*- ***-- --*** - (14)
17 -**-* ----* -**-* *-*-- --*-* * (12)
19 **-*- *--*- *-*** -*-*- -*-*- * (14)
21 *---* ***** ---*- --*** ***-- - (14)
23 -*-*- -**-- *-*-* -*--* *--*- * (12)
25 ---** *--** *---- -***- -***- - (12)
```

```
 n mod 17 (50.0%)
 1 --**- -*-** -*--* *- (8)
 2 -***- *---- --*-* ** (8)
 3 *-**- **--- -**-* *- (9)
 4 -*--* **--- -***- -* (8)
 5 **-*- -**-- **--* -* (9)
 6 ***-* *---- --**- ** (9)
 7 *-*-* --*** *--*- *- (9)
 8 --*-* -**-- **-*- *- (8)
 9 -*--- -**** **--- -* (8)
10 ***-- -*-** -*--- ** (9)
11 **-** ---** ---** -* (9)
12 *---* ***-- ****- -- (9)
13 -*-** --*-- *--** -* (8)
14 *--*- *-*** *-*-* -- (9)
15 ---** *--** --*** -- (8)
16 --*-- *-*** *-*-- *- (8)
```

```
 n mod 19 (50.0%)
 1 -*-*- -***- -***- -*-* (10)
 2 **--* *-*-- --*-* *--* (9)
 3 *--*- --*** ***-- -*-- (9)
 4 --**- ***-- --*** -**- (10)
 5 ---** -*-** **-*- **-- (10)
 6 --*** *--*- -*--* ***- (10)
 7 -*--- *-*** ***-* ---* (10)
 8 *-*-- *--** **--* --*- (9)
 9 -***- *---* *---* -*** (10)
10 *-*-- -**-* *-**- --*- (9)
11 -**-* --**- -**-- *-** (10)
12 **-*- **--- ---** -*-* (9)
13 **-** ----* *---- **-* (9)
14 ***-- -*-*- -*-*- --** (9)
15 *---* **-*- -*-** *--- (9)
16 ----* ***-* *-*** *--- (10)
17 -**-* -*--* *--*- *-** (10)
18 *-*** --*-- --*-- ***- (9)
```

```
 n mod 23 (50.0%)
 1 --*** -*-*- -**-- *-*-* **- (12)
 2 ---*- -**** *--** ***-- *-- (12)
 3 --*-* *--** *--** *--** -*- (12)
 4 -*--* -***- -**-- ***-* --* (12)
 5 *--** *---* -**-* ---** *-- (11)
 6 -**** -*--- *--*- --*-* *** (12)
 7 *-*-- -**-* *--** -**-- -*- (11)
 8 ---*- ***-* -**-* -***- *-- (12)
 9 -*--- **--* ***** --**- --* (12)
10 ****- ---** ----* *---- *** (11)
11 *-**- --*-- ****- -*--- **- (11)
12 ---** *-**- *--*- **-** *-- (12)
13 -***- *---- ****- ---*- *** (12)
14 **--* --*-- ****- -*--* --* (11)
15 *-*-- **-*- -**-- *-**- -*- (11)
16 -**-- --*** -**-* **--- -** (12)
17 *-*-* ***-- ----- -**** -*- (11)
18 -**-* *-*-* ----* -*-** -** (12)
19 **-*- *-**- ----- **-*- *-* (11)
20 **--- **-*- *--*- *-**- --* (11)
21 **-** -*--* ----* --*-* *-* (11)
22 *---* ---** ***** *---* --- (11)
```

```
 n mod 29 (50.0%)
 1 -*-** ***-- *---- ----* --*** **-* (14)
 2 **-** *--*- *--*- -*--* -*--* **-* (15)
 3 *--** -*--* -***- -***- *--*- **-- (15)
 4 --*-- -*-** *-*-* *-*-* **-*- --*- (14)
 5 ---** -*-*- **--* *--** -*-*- **-- (14)
 6 --**- *-**- **--- ---** -**-* -**- (14)
 7 -**-- ***-- -*-*- -*-*- --*** --** (14)
 8 *-**- -*-** *--*- -*--* **-*- -**- (15)
 9 -*-*- ---** -**-* *-**- **--- -*-* (14)
10 ***** -*--- -*--* *--*- ---*- **** (15)
11 ***-* -***- --*-- --*-- -***- *-** (15)
12 *--*- ****- -**-- --**- -**** -*-- (15)
13 -**-- ---*- ****- -**** -*--- --** (14)
14 ***-- **--* **--- ---** *--** --** (15)
15 *-*-* *--*- --*** ***-- -*--* *-*- (15)
16 -*--* *---* -***- -***- *---* *--* (14)
17 ****- --*-* ---** **--- *-*-- -*** (15)
18 **-*- *---* *-*-* *-*-* *---* -*-* (15)
19 *---- ***-- *-*** ***-* --*** ---- (15)
20 -*--- -**** --**- -**-- ****- ---* (14)
21 *---* *-*** -*--* *--*- ***-* *--- (15)
22 --**- *---- -**** ****- ----* -**- (14)
23 --*** *-*-* ---*- -*--- *-*-* ***- (14)
24 ----* -**-* *--** **--* *-**- *--- (14)
25 -*--* **-** ----* *---- **-** *--* (14)
26 **--- --**- **-** **-** -**-- ---* (15)
27 *-*-* --*-* ***-- --*** *-*-- *-*- (15)
28 --*** --*-- *-*-* *-*-* --*-- ***- (14)
```

```
 n mod 31 (50.0%)
 1 ----* **-*- -**** --*** *--*- ***-- - (16)
 2 -***- *---* -**-* --*-* *-*-- -*-** * (16)
 3 *--*- --*-* --*** ***** --*-* ---*- - (15)
 4 ---*- *-*** *-*-- **--* -**** -*-*- - (16)
 5 -*--* *-*-* *---* ***-- -**-* -**-- * (16)
 6 *--** -**-- ***-- ----* **--* *-**- - (15)
 7 --*-* -*--* --*** ***** --*-- *-*-* - (16)
 8 --*** -**-* *--*- ---*- -**-* *-*** - (16)
 9 --*-- *-**- -***- **-** *--** -*--* - (16)
10 -*--* -*-** *-**- ---** -***- *-*-- * (16)
11 ***-* --*** ----* --*-- --*** --*-* * (15)
12 **-*- ***-- ---** --**- ----* **-*- * (15)
13 *-*** ---*- ---** ****- ---*- --*** - (15)
14 -*-** ----- ***-* ***-* **--- --**- * (16)
15 *--*- ---** **-*- **-*- ****- ---*- - (15)
16 -*--- -**-- **-** ****- **--* *---- * (16)
17 *-**- **--- *-*-* --*-* -*--- **-** - (15)
18 --*** **--* -*--- **--- *-*-- ***** - (16)
19 --*-- ****- *---* ***-- -*-** **-*- - (16)
20 -***- ---*- **-** --**- **-*- ---** * (16)
21 **-** **-*- ----- **--- ---*- ****- * (15)
22 ***-- -*-*- *-*-- **--* -*-*- *---* * (15)
23 ***-- *---* *-**- ---** -**-- -*--* * (15)
24 *---- -**** -**-* --*-* *-*** *---- - (15)
25 -***- -**** -*--- ----- *-*** *--** * (16)
26 ***-* --*-- -**-- **--* *---* --*-* * (15)
27 *---* *--** **--* --*-- ****- -**-- - (15)
28 -*-** *-**- --**- ---** ---** -***- * (16)
29 **--- **--* -*-*- **-*- *-*-- **--- * (15)
30 *-*-* *-*-- **-*- ---*- **--* -**-* - (15)
```

```
 n mod 37 (50.0%)
 1 -**-* --*-* ---*- -**** **--* ---*- *--*- ** (18)
 2 *-*-* --**- --*** *-*-- *-*** *---* *--*- *- (19)
 3 ----* -**-- **-** *--** --*** -**-- **-*- -- (18)
 4 -**** *--*- -*--* ---** ---*- -*--* --*** ** (18)
 5 **-*- --**- **--- ***-- ***-- -**-* *---* -* (19)
 6 *--** ***-* *-*-* ----- ---*- *-**- ***** -- (19)
 7 -*--* ***-* -*--* ---** ---*- -*-*- ****- -* (18)
 8 *--** --*** -*-** -*--- -*-** -*-** *--** -- (19)
 9 --**- -*--- ***-* -**-- **-*- ***-- -*--* *- (18)
10 --**- *-*-* -***- *---- --*-* **-*- *-*-* *- (18)
11 --**- --*-* ---** **-** -**** ---*- *---* *- (18)
12 -*--- --*** -**-* *-*-- *-**- **-** *---- -* (18)
13 ***** ----* **--* --*-- *--*- -***- ---** ** (19)
14 ****- **--- -*-*- *--** --*-* -*--- -**-* ** (19)
15 *-*** -*-** ----- *-*** *-*-- ---** -*-** *- (19)
16 -**-* -*-** *---- **--- -**-- --*** -*-*- ** (18)
17 **--- *-*-* *-*-- **-** -**-- *-**- *-*-- -* (19)
18 ***-- **-*- --*** -*--- -*-** *---* -**-- ** (19)
19 ***-* -**-- ***-- *---- --*-- ***-- **-*- ** (19)
20 *-**- ***-- ----* ***-- ****- ----- ***-* *- (19)
21 -*--- -*--- -**** ***-- ***** **--- -*--- -* (18)
22 **-*- ----* -***- -**** **--* **-*- ----* -* (19)
23 **-** -***- --*-- -*-** -*--- *---* **-** -* (19)
24 *---- -*-** *-*** --*** *--** *-*** -*--- -- (19)
25 --*-- **-** **--- -**-- **--- -**** -**-- *- (18)
26 ---** *---- *-*-- ***** ***-- *-*-- --*** -- (18)
27 --*-* *-**- --*-- -**** **--- *---* *-**- *- (18)
28 -**-- ---** *-*-* *--** --**- *-*** ----- ** (18)
29 *---* *--*- **-*- *-*** *-*-* -**-* --**- -- (19)
30 -*-*- ****- *--*- --*-- *---* --*-* ***-* -* (18)
31 ***-- *-*-- *--** --*** *--** --*-- *-*-- ** (19)
32 *---- **-** -*--* **-** -***- -*-** -**-- -- (19)
33 ---*- -***- ****- ---** ----* ***-* **--* -- (18)
34 -*-** *--*- *--** -*--- -*-** --*-* --*** -* (18)
35 *-*-* *---* ****- -*--- -*--* ****- --**- *- (19)
36 ---** **--* --**- *-*-- *-*-* *--*- -**** -- (18)
```