Stud. Univ. Babeş-Bolyai Math. 59(2014), No. 4, 457–462

The largest known Cunningham chain of length 3 of the first kind

Gábor Farkas, Gábor E. Gévay, Antal Járai and Emil Vatai

Abstract. Cunningham chains of length n of the first kind are n long sequences of prime numbers $p_1, p_2, \ldots, p_n$ so that $p_{i+1} = 2p_i + 1$ (for $1 \leq i < n$). In [3] we have devised a plan to find large Cunningham chains of the first kind of length 3 where the primes are of the form $p_{i+1} = (h_0 + cx) \cdot 2^{e+i} - 1$ for some integer x with $h_0 = 5\,775$, $c = 30\,030$ and $e = 34\,944$. The project was executed on the non-uniform memory access (NUMA) supercomputer of NIIF in Pécs, Hungary. In this paper we report on the obtained results and discuss the implementation details. The search consisted of two stages: sieving and the Fermat test. The sieving stage was implemented in a concurrent manner using lockfree queues, while the Fermat test was trivially parallel. On the 27th of April, 2014 we have found the largest known Cunningham chain of length 3 of the first kind which consists of the numbers $5110664609396115 \cdot 2^{34944+j} - 1$ for $j = 0, 1, 2$.

Mathematics Subject Classification (2010): 11Y11.

Keywords: Cunningham chains, primality, computational number theory.

1. Cunningham chain search project

Cunningham chains of length n of the first kind are n long sequences of prime numbers $p_1, p_2, \ldots, p_n$ so that $p_{i+1} = 2p_i + 1$ (for $1 \leq i < n$). In 2013 we set out to find the largest primes which form a Cunningham chain of the first kind of length 3. The first stage of the plan was to take a large number of candidates, each representing one chain, i.e. three primes, and eliminate most of them using a sieve similar to the sieve of Eratosthenes. In the second stage the Fermat test was used to check if the remaining candidates are probable primes. Finally, that the numbers were actually primes was proven using the OpenPFGW open source implementation of the Brillhart-Lehmer-Selfridge test.

This paper was presented at the 10th Joint Conference on Mathematics and Computer Science (MaCS 2014), May 21-25, 2014, Cluj-Napoca, Romania.

The program was looking for primes of the form

$$p_{i+1} = p_{i+1}(x) = (h_0 + cx) \cdot 2^{e+i} - 1 \text{ for } i = 0, 1, 2, \tag{1.1}$$

therefore the parameters of the program were e which determined the magnitude of the primes and h_0 and c which were required to ensure the deterministic Riesel test would prove primality fast enough.

1.1. The sieve

The candidates were different values of x, and these candidates were represented by bits in a large sieve table H which was sieved with a set of primes. The size of the sieve table, denoted by h, and the upper bound of primes P were also parameters specific to the sieve program, so the values of $0 \leq x < h$ were sieved with primes $p < P$.

Sieving with a prime $p < P$ means finding the first $0 \leq x_i < h$ for which $p | p_i(x_i)$ and eliminating x_i (for $i = 1, 2, 3$). Then the candidates $x_i + kp$ (for any $k \in \mathbb{Z}$) can also be eliminated since

$$\begin{aligned} p_i(x_i + kp) &= (h_0 + c(x_i + kp)) \cdot 2^{e+i} - 1 \\ &= (h_0 + cx_i) \cdot 2^{e+i} - 1 + ckp \cdot 2^{e+i} \\ &= p_i(x_i) + ckp \cdot 2^{e+i} \end{aligned}$$

that is, $p | p_i(x_i)$ implies $p | p_i(x_i + kp)$, therefore $p_i(x_i + kp)$ is composite (for $k \in \mathbb{Z}$ and $i = 1, 2, 3$). More details can be found in [3].

1.2. Probabilistic and deterministic primality tests

For the probabilistic primality test the Fermat test was used with base 3. For each candidate x, not eliminated by the sieve, the program checked if

$$3^{p_i - 1} \equiv 1 \pmod{p_i} \text{ where } p_i = p_i(x) \text{ for } i = 1, 2, 3. \tag{1.2}$$

The probability of a false positive result of the Fermat test for all three p_i's is close to none, so practically, after finding a candidate x for which (1.2) is true, the search would be over.

However this would not prove that these numbers were prime, but this was not a problem, because the Riesel test provides a very fast and efficient way to verify the primality of the numbers of the form $k \cdot 2^e - 1$ so it can be used for (1.1).

2. Implementation

As described in [3], the above mentioned parameters were chosen as follows:

- the number of initial candidates, i.e. the size of the sieve table $H = 2^{37}$;
- the upper bound of the sieving primes $P = 2^{48}$;
- the parameters for the $p_i(x)$ polynomials were $h_0 = 5\,775$, $c = 30\,030$ and $e = 34\,944$.

2.1. The implementation of the sieve

The sieve implementation was written by Gábor E. Gévay in C++ and compiled with GCC version 4.8.1. To implement concurrent execution of the sieve program, OpenMP and lock-free queues from the Boost [1] library were used.

The primes up to $\sqrt{P} = 2^{24}$ were generated as the initial step, then each CPU core generated a "chunk" of the remaining sieving primes up to $P = 2^{48}$. The size of one chunk was 2^{31} bits, without representing the even numbers, so the effective chunk size was actually 2^{32}. Ergo for each chunk a sieve of Eratosthenes was executed on an interval of 2^{32} numbers and because all the primes up to $P = 2^{48}$ were to be sieved with, the number of chunks was $2^{16} = 65\,536$.

After generating the primes for the given chunk, each CPU core proceeded with sieving the main sieve table H representing the candidates with the primes found. Thus, several hundred cpu cores were doing bit operations on a shared 16 GB bitset at the same time, which required some synchronization. There were 32 special threads that were actually writing to the bitset. Each of these was responsible for doing operations on a 1/32th chunk of the bitset, and each had a queue (`boost::lockfree::queue`) to which the computing threads pushed the bit operations. Furthermore, each computing thread had a thread-local proxy object of the bitset, and used a method of this proxy object to request the bit operations. These objects were responsible for calculating the index of the writer thread to which the operation is to be forwarded, and also for buffering the operations to prevent the synchronization of the queue from incurring too much overhead. (Note that sequential consistency of the bit operations was not required.) The supercomputer used has a NUMA architecture. The above scheme requires remote memory accesses only for the queue operations, while both writing to the buffers in the proxy objects and executing the bit operations on the sieve table involves only local memory access. Each prime sieved three times into H, once for each polynomial $p_i(x_i)$.

Finally the sieve program converted the sieve table into a `vector` of 64bit `long long int`s and wrote them into a (binary) file.

2.2. The Fermat test

The Fermat test was written as a function with two parameters: x the candidate and i the index of the polynomial. It calculated the value of $p = p_i(x)$, and then checked if $3^{p-1} \equiv 1 \pmod{p}$. So when $3^{p-1} \bmod p = 1$ the function returned `true`, indicating that p was a probable prime, otherwise it returned `false` which meant p was certainly composite.

The output of the sieve program, containing the x candidates as `long long int`s, was the input for the Fermat test, which was implemented using the *GNU Multiple Precision (GMP)* library. Concurrent execution was not a problem: each thread read a different candidate x, calculated $p = p_1(x)$ and checked if $3^{p-1} \equiv 1 \pmod{p}$, and if the test returned `true`, the test is executed for $p_2(x)$ and if it was still `true` then for $p_3(x)$ also.

2.3. Execution

The programs were written for and executed on the supercomputers of NIIF institute [2] in Pécs. The NIIF institute provided us with access to other supercomputers, including the ones in Budapest, Debrecen and Szeged, but the supercomputer of the University in Pécs was targeted, because of it's shared memory which could hold the sieve table of $H = 2^{37}$ bits i.e. 16GiB in size.

To provide us with the advantages of C++11, we used GCC version 4.8.1. Performance of the available GMP library was suboptimal, so we compiled a newer, 5.1.2 version, which provided better performance. We also tried to use the `tuneup` utility of GMP to optimize it, but it did not improve performance.

3. Results

3.1. The first run

The sieve was executed on the 8th October 2013 and ran for about 41 hours with 352 threads. After sieving, 88 573 926 candidates were left. In [3] the number of remaining candidates after sieving was estimated using the Bateman-Horn conjecture and was to be approximately 88 570 684. The number of the actual candidates not eliminated during sieving came very close to this value, the error was only about 0.003%.

The generated output file, `fermat_in` was about 708MB in size. This file was divided into smaller parts using the `split` command to be processed by the Fermat test. The Fermat tests running on these parts of the `fermat_in` file were started immediately after the sieve program finished on the 10th of October. The estimate of the time it took to finish the Fermat tests was roughly 4 weeks, however there were some unanticipated slowdowns in the execution, which were only later solved.

On November 16th 2013. the Fermat test finished, without finding any Cunningham chains and the project came to a temporary halt.

3.2. The second run

The estimated number of Cunningham chains found (based on the above mentioned parameters) should have been ≈ 1.3, and it implied that we might have just been out of luck, i.e. we needed to continue with more candidates, with an extension of the sieve table. The project was resumed in March of 2014. The program had to be modified, because in the first run the candidates were $0 \leq x < 2^{37}$, and now the search was to be extended to the candidates $2^{37} \leq x < 3 \cdot 2^{37}$. The upper bound for the primes P was also modified from 2^{48} to 2^{50} to save a little time on the Fermat tests. The number of candidates not eliminated after sieving was 156 743 147. Using calculations similar to the ones described in [3], this value was expected to be 156 722 877. Again, the actual value was very close to the expected one, the error was only about 0.013%.

The sieve was again run on the NIIF supercomputer in Pécs, but to find the primes more quickly, the Fermat test was executed on all available computers of the NIIF institute, including the ones in Debrecen and Szeged.

On Friday, April 25th 2014. at 02:59:14 on the supercomputer in Pécs the Fermat test finished with a positive result. The result was verified with a quick reimplementation of the Fermat test in the Maple computer algebra system, and there was no mistake, we have found three probable primes for the candidate $x = 170\,185\,301\,678$. Afterward we verified the results using the OpenPFGW program, which is an open source implementation of the very fast and, most importantly deterministic, BrillhartLehmerSelfridge test for primes of the form $k \cdot 2^e - 1$.

So the largest know Cunningham chains of the first kind of length 3, according to "The Top Twenty: Cunningham Chains (1st kind)" [4], where we submitted our findings, consist of the following three primes:

$$p_{i+1}(x) = (5775 + 30030 \cdot x) \cdot 2^{34944-i} - 1 \text{ for } x = 170185301678$$

for $i = 1, 2, 3$, that is:

$$p_1(x) = 5110664609396115 \cdot 2^{34944} - 1 \qquad 10535 \text{ digits}$$
$$p_2(x) = 5110664609396115 \cdot 2^{34945} - 1 \qquad 10536 \text{ digits}$$
$$p_3(x) = 5110664609396115 \cdot 2^{34946} - 1 \qquad 10536 \text{ digits}$$

Acknowledgment. We are thankful to the operators at NIIF, who wrote some very useful wiki pages, which were of great help to us, and we are thankful to Gábor Kőszegi, for having the idea to use the supercomputer at the University of Pécs and helping us obtain access to it.

References

[1] *Chapter 18. Boost.Lockfree*, http://www.boost.org/doc/libs/1_55_0/doc/html/lockfree.html

[2] *National Information Infrastructure development Institute* (Nemzeti Információs Infrastruktúra fejlesztési Intézet), http://www.niif.hu/

[3] Farkas, G., Vatai, E., *Sieving for large Cunningham chains of length 3 of the first kind*, Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Computatorica, **40**(2013), 215–222.

[4] *The Top Twenty: Cunningham Chains (1st kind)*, http://primes.utm.edu/top20/page.php?id=19

Gábor Farkas
"Eötvös Loránd" University
Faculty of Informatics
Pázmány Péter sétány 1/C, 1117 Budapest, Hungary
e-mail: farkasg@compalg.inf.elte.hu

Gábor E. Gévay
"Eötvös Loránd" University
Faculty of Informatics
Pázmány Péter sétány 1/C, 1117 Budapest, Hungary
e-mail: ggab90@gmail.com

Antal Járai
"Eötvös Loránd" University
Faculty of Informatics
Pázmány Péter sétány 1/C, 1117 Budapest, Hungary
e-mail: ajarai@moon.inf.elte.hu

Emil Vatai
"Eötvös Loránd" University
Faculty of Informatics
Pázmány Péter sétány 1/C, 1117 Budapest, Hungary
e-mail: vatai@inf.elte.hu