Annales Univ. Sci. Budapest., Sect. Comp. 41 (2013) 355–360

INVERSE SIEVE

Emil Vatai (Budapest, Hungary)

Dedicated to Professors Zoltán Daróczy and Imre Kátai on their 75th birthday

Communicated by Antal Járai

(Received March 08, 2013; accepted May 28, 2013)

Abstract. The paper suggests a method to speed up distributed primality testing by compressing the sieve table and so reducing data traffic. Since smaller primes sieve out most of the candidates, most of the sieving done by larger primes is redundant. Instead of doing such unnecessary sieving on a complete and uncompressed table (or an array of indices), one can eliminate most of the redundant offsets based on a compressed sieve table. The compression is done by "smearing" the exact locations of the potential primes, that is compressing the sieve to one N-th of its original size by encoding each N long interval of the table with a 0 bit if it contains no potential primes, and with a 1 bit otherwise. If N is chosen to be a power of two, calculating the offset in the compressed table requires virtually no extra effort.

1. The problem

Researchers at Eötvös Loránd University in Budapest have more then once set world records in finding the largest known primes of a certain kind e.g. twin primes (p and $p+2$ are prime) [3, 4, 5, 6, 8] or Sophie-Germain primes

Key words and phrases: Sieve, number theory, primality, parallel.
2010 Mathematics Subject Classification: 11-04, 11Y11, 68W99.
https://doi.org/10.71352/ac.41.355

(p and $2p+1$ are prime) [7, 9, 10]. Further possibilities are finding Cunningham chains of the first kind ($p_1, p_2 = 2p_1 + 1, p_3 = 2p_2 + 1, \ldots$ are primes) or second kind ($p_1, p_2 = 2p_1 - 1, p_3 = 2p_2 - 1, \ldots$ are prime).

The process of finding such (large) primes usually goes as follows [11]:

1. Determining the interval size where to look for the primes and other parameters: using mathematical methods, the size of the interval is found, where the primes can be found with large enough certainty. The Bateman-Horn conjecture [1] is usually used. This interval is represented by a large bit array, referred to as the *sieve table*.

   One parameter to determine is the size of the sieve table, i. e. the number of candidates, denoted by H, the other parameter is a boundary for the primes which will sieve the sieve table, denoted by B_0.

2. Sieving: Each bit in the sieve table represents a candidate number which might or might not be a prime, and by sieving out (or eliminating) the ones which are certainly not prime, one is left with a reasonably smaller number of candidates.

3. Probabilistic test: After the number of potential primes has dropped low enough, and it becomes cheaper (i. e. more efficient) to determine each candidate's primality by running a Fermat test [2] (or some other primality test) instead of sieving, the test continues with the formal approach.

4. Exact test: To prove that the remaining numbers found are indeed primes, an exact test has to be executed.

This paper suggests a performance improvement when executing the sieving stage in a distributed fashion.

1.1. Distributed sieving

Let B_0 be the upper limit of primes we want to sieve with. Finding primes $p < \sqrt{B_0} = 2^{24}$ is a feasible task on an average computer at the time of writing this paper and one can distribute the effort of finding the other primes among separate computers (nodes). After all primes up to B_0 have been found, the composites from the sieving table have to be discarded, again via a sieve similar to the sieve of Eratosthenes. This introduces some problems, since the entire sieve table has to be sieved with all the primes (found on different computers). This would generate a lot of data traffic between nodes, because after obtaining the sieved table from one node, it has to be "sent back" and merged with the sieved tables from the other nodes.

1.2. A case study: Twin primes and Cunningham chains

As an example, to find twin primes and Cunningham chains of the second kind of length 2, primes with more than 75000 decimal digits as described in [11], one needs to sieve with primes up to $B_0 = 2^{48}$ on a sieve table of $H = 2^{35}$ candidates.

This case is interesting, because of two reasons:

1. Small primes ($p < 2^{24}$) sieve out most of the sieve table, i.e. the sieve table becomes quickly sparse,
2. Large primes ($2^{35} < p < 2^{48}$) will sieve at most once in the sieve table.

Therefore, a larger amount of vital information is located in the prime tables than in the sieve table, ergo compressing the sieve table can increase performance.

2. A proposed solution for minimizing data traffic

The usual solution is to compress the sieve table by switching from a bit table representation to a representation where the actual candidates are stored as an array of unsigned integers. This already yields a considerate compression of the data needed to be transferred to all the nodes, but at the price of slowing down and complicating the process of eliminating candidates from the sieve table, so the following idea can help improve performance.

2.1. The basic idea

Let us assume the sieve table contains a multiple of N bits (candidates), and the composites are represented by 0's and the potential primes are represented by 1's. The table can be compressed to one N-th of its original size, by representing each interval of N bits with only one bit according to the following simple rules:

1. if any of the N bits (candidates) are potential primes (i. e. set to 1), the N bits are represented by one potential prime (one 1 bit).
2. if all of the N bits (candidates) are already known to be composite (i. e. set to 0), the N bits are represented by one composite (one 0 bit).

This can rephrased as "smearing" the 1 bits over the N long intervals, so if all bits in the interval are 0's, then the smeared interval is a 0 bit, if there are any 1's in the interval, then the smeared interval is a 1 bit.

After compressing the sieve table, it can be sent to all the different nodes. On the nodes, the offsets for primes can be calculated, and if it sieves to a 0 interval (with all composites), then the sieve should mark a composite which is already marked so no effective sieving is done, therefore the offset (and if it was a large prime, the prime too) can be thrown away, i. e. it is known for sure, it does not need to be sent back.

2.2. Benefits and implementation details

If N is chosen to be a power of two ($N = 2^n$), determining the offset in the compressed table from the offset in the uncompressed table becomes a bitwise shift. Therefore, the nodes should have a copy of the compressed sieve table and some of the (larger) primes. The information to calculate the offset, that is the index of the candidate which will be eliminated, can be obtained from the primes, and shifting it right by n (which is the equivalent of dividing it by $N = 2^n$) yields the offset in the compressed table.

The assumption is that even after compressing the sieve table using the above mentioned method, the compressed sieve table remains fairly sparse, and most of the elimination would be superfluous, because the offset would usually eliminate a empty smear which contains only candidates, already eliminated by smaller primes. Primes (or offsets) which would perform such redundant eliminations can be eliminated themselves, hence the name *inverse sieve*, because it eliminates not the candidates but the primes which would sieve the sieve table.

On the other hand, some offsets will fall on smears which represent intervals, containing at least one potential prime. In this case, it is not known if actual sieving is to be done, so two things have to be considered:

1. The original offset (not the shifted one, calculated for the compressed sieve table) has to be stored, and sent back or united in some other way with the information from the uncompressed sieve.

2. The compressed sieve table must not be changed, i. e. the smear must not be cleared, because the offset might not eliminate any candidates from that interval at all, or there might be more potential primes in one smeared interval.

3. Summary

As mentioned above, small primes sieve out most of the candidates from the sieve table, thus it becomes very sparse and contains a minimal amount of information, that is in the later stages of the algorithm the computational effort is shifted from the sieve table to the primes.

By distributing a smeared (i.e. compressed) version of the sieve table to the nodes, they can approximately determine which candidate in the original table will be eliminated, *if* it is not eliminated already. This can be achieved without effectively increasing the cost of computation (if the table is compressed by a factor which is a power of two, the extra work can be done with only an extra bitwise shift right instruction).

As a result, instead of collecting *all* the offsets, and merging them with the uncompressed sieve table, most of the superfluous offsets can immediately be discarded, thus eliminating a great deal of unnecessary work and data traffic.

4. Future work

Since this is just a theoretical discussion, the first step will be of course to implement the inverse sieve as part of an actual prime searching project.

This approach, of coarse, is just an improvement, and does not replace any of the techniques used before, it supplements them. So another important task, is to figure out, in what stage of the sieve procedure, should the inverse sieve be introduced. Its benefits are obvious for large primes which would sieve only once, but maybe it can be applied for primes which sieve more than once, but that would complicate the process, because if a prime sieves the table more than once, after calculating the (first) offset, it might result in a unnecessary elimination, but the offset and the prime can not be thrown away, because the other offsets will be calculated from them.

Yet another question will be finding the optimal parameter n. Because if the table is compressed too much i.e. n is too large, the table becomes too dense, and the inverse sieve becomes useless. If n is too small, the table will not shrink enough, so sending to the remote nodes and storing it on the hard drive will become too costly.

Acknowledgement. I would like to thank Antal Járai and Gábor Farkas for their help on working out the details of this method and making it fit in the prime search project.

References

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

[2] **Aczél, J.,** *Lectures on Functional Equations and their Applications*, Academic Press, New York and London, 1966.

[3] **Csajbók, T., G. Farkas, A. Járai, Z. Járai and J. Kasza,** Report on the largest known twin primes, *Annales Univ. Sci. Budapest, Sect. Comp.*, **25** (2005), 247–248.

[4] **Csajbók, T., G. Farkas, A. Járai, Z. Járai and J. Kasza,** Report on the largest known Sophie Germain and twin primes, *Annales Univ. Sci. Budapest, Sect. Comp.*, **26** (2006), 181–183.

[5] **Csajbók, T., G. Farkas, A. Járai, Z. Járai and J. Kasza,** The largest known twin primes of the World, $16869987339975 \cdot 2^{171960} \pm 1$ (51779 digits), `http://primes.utm.edu/top20/page.php?id=1`, (2005).

[6] **Csajbók, T., G. Farkas, A. Járai, Z. Járai and J. Kasza,** The largest known twin primes of the World, $100314512544015 \cdot 2^{171960} \pm 1$ (51780 digits), `http://primes.utm.edu/top20/page.php?id=1`, (2006).

[7] **Csajbók, T., G. Farkas, A. Járai, Z. Járai and J. Kasza,** The largest known Sophie Germain prime of the World, $137211941292195 \cdot 2^{171960} - 1$ (51780 digits), `http://primes.utm.edu/top20/page.php?id=2`, (2006).

[8] **Csajbók, T., G. Farkas, A. Járai, Z. Járai and J. Kasza,** The largest known twin primes of the World, $194772106074315 \cdot 2^{171960} \pm 1$ (51780 digits), `http://primes.utm.edu/top20/page.php?id=1`, (2007).

[9] **Csajbók, T., G. Farkas, A. Járai, Z. Járai and J. Kasza,**The largest known Sophie Germain prime of the World, $620366307356565 \cdot 2^{253824} - 1$ (76424 digits), `http://primes.utm.edu/top20/page.php?id=2`, (2009).

[10] **Csajbók, T., G. Farkas, A. Járai, Z. Járai and J. Kasza,** The largest known Sophie Germain prime of the World, $648621027630345 \cdot 2^{253824} - 1$ (76424 digits), `http://primes.utm.edu/top20/page.php?id=2`, (2009).

[11] **Csajbók, T., G. Farkas and J. Kasza,** Sieving for large twin primes and Cunningham chains of length 2 of the second kind, *Annales Univ. Sci. Budapest., Sect. Comp.*, **38** (2012), 117–128.

E. Vatai
ELTE IK, Department of computeralgebra
Budapest
Hungary
`emil.vatai@gmail.com`