Comparison of SS, MR, AKS and Trial Division for Primality

Kevin van As
4076311
MSc Applied Physics

Laurent Verweijen
4030281
MSc Media and Knowledge

ABSTRACT

In this report, we will study common primality tests. We will look at both deterministic primality tests and probablisic primality tests.

Keywords

randomized algorithms, primality, SS, MR, AKS

1. INTRODUCTION

Prime numbers are of interest to mathematicians and cryptographers. Many encryption algorithms, e.g. RSA, rely on the fact that factoring large numbers is difficult. However, since every number may be factorised using prime numbers, we can greatly reduce the computational effort if we know the prime numbers smaller than n, the number to be factorised.

Recently, Angrawal, Kayal and Saxena (AKS) [1] have proven that the primality problem can be solved deterministically in polynomial time. However, in practice randomized algorithms are preferred because of their speed and low error probability.

In this report, we will compare several simple deterministic algorithms (TD, WS) and randomized algorithms (SS, MR) with the AKS algorithm.

2. THEORY

There exist several algorithms to check whether a given integer, n, is a prime number. The easiest among them all is the "trial division" (TD) algorithm, described in Sec. 2.1. It is a deterministic algorithm with a $O(\sqrt{n})$ complexity. Several different algorithms have been devised, both deterministic and random, to beat this complexity. All randomized algorithms below are based on "Fermat's Theorem" for primality [2], which says that "for every prime number, n,

$$a^{n-1} \equiv 1 (\bmod\, n), \tag{1}$$

$\forall a \in Z_n^*$". Sadly, there does as well exist an infinite set of composite numbers which satisfy this criterion. They are the so-called "Carmichael numbers". Each of the randomized algorithms below deals with these numbers in their own way.

2.1 Trial Division

In the trial division (TD) algorithm, we start with the very definition of a prime number: it is only dividable by 1 and itself. To test this statement, we divide by every integer up to $\sqrt{n}$. If any of those divisions result in an integer, the number is not a prime. Otherwise, it must necessarily be a prime, from the very definition. TD is a deterministic algorithm with $O(\sqrt{n})$ complexity.

2.2 Wheel-Sieve

The Wheel-Sieve (WS) algorithm is an optimized version of TD. The algorithm takes the first k prime numbers (hard-coded, or with a cheap primality algorithm) $p_1 = 2, p_2, \ldots, p_k$ as initial prime numbers. First the algorithm uses trial division for these k numbers to see whether n is dividable by any of these numbers.

Since we only take the first k prime numbers, which may be very far off from n, further steps are required. Let, $m = \prod p_i$, then the algorithm will test divisibility of n by all numbers l, such that $\forall p_i : l \neq p_i (\mod m)$. Unfortunately, this algorithm gives only a linear speedup in comparison with trial division.

2.3 Solovay-Strassen

The Solovoy-Strassen (SS) algorithm is a randomized primality test based on Fermat's Theorem. The algorithm will never error for prime n, but it has a probability of at most $(\frac{1}{2})^k$ of incorrectly identifying a composite number as prime when the algorithm is repeated k times. Each trial of the algorithm has a runtime of $O((\log n)^3)$ as this is the time needed for modular exponentiation.

2.4 Miller-Rabin

The Miller-Rabin (MR) primality test is also based on Fermat's Theorem and will never error for prime n either. On composite numbers it will error with a probability smaller than $\frac{1}{4}$. [3] Again each trial of the algorithm has a runtime of $O((\log n)^3)$ as this is the time needed for modular exponentiation.

2.5 Angrawal-Kayal-Saxena

Angrawal-Kayal-Saxena (AKS) is a polynomial deterministic primality test, published in 2002. They were the first

Table 1: Execution time as a function of $\log n$ in milliseconds

$\log n$	SS	AKS	MR	TD	WS
1	8	1726	13	12	8
2	110	19326	128	8	8
3	110	32415	130	14	16
4	122	689248	140	20	14
5	156	582623	184	23	18
6	169	3735301	190	26	18
7	192	10045207	212	29	20
8	211	21503625	217	32	21
9	234	60103404	237	36	22
10	254	127910717	245	40	25
11	275	292019772	256	46	27
12	297	581977224	269	54	30
13	321	1014476731	278	64	34
14	726.0	-	292.0	78.0	39.0
15	1278.0	-	304.0	94.0	44.0
16	702.0	-	316.0	117.0	52.0
17	749.0	-	330.0	149.0	63.0
18	790.0	-	342.0	189.0	77.0
19	830.0	-	354.0	251.0	94.0
20	873.0	-	369.0	325.0	118.0
21	914.0	-	382.0	1499.0	154.0
22	954.0	-	409.0	1005.0	513.0
23	997.0	-	747.0	1332.0	727.0
24	1037.0	-	772.0	1798.0	624.0
25	1075.0	-	794.0	2399.0	828.0
26	1120.0	-	819.0	3252.0	1112.0
27	1157.0	-	844.0	4438.0	1512.0
28	1197.0	-	871.0	6130.0	2073.0
29	1238.0	-	897.0	7479.0	2841.0
30	1017.0	-	922.0	11676.0	3915.0
31	1318.0	-	947.0	14595.0	5036.

to show that primality is in P. The authors state that the runtime of this algorithm is at most $O((\log n)^{\frac{21}{2}})$ [1].

3. EXPERIMENTS

Experiments have been performed by trying input sizes of different lengths. For each length k, a sample of 1000000 random numbers was generated in the interval $[2^{k-1}, 2^k >$ and for these numbers we recorded the runtime and the fraction of samples that was predicted correctly. For AKS these experiments had to be aborted at a certain point, because this algorithm turned out to be very slow.

4. RESULTS

We can see the runtime in Table 1 and we can see the runtime plotted in Figure 1.

5. CONCLUSIONS

As we see the Trial Divion and Wheel Sieve algorithm perform well for small data, but for big numbers they are exponential in the input size. They do have an error rate of 0%. For bigger data SS and MR perform better. MR seems to outperform SS both in runtime and in error rate. However for the runtime this might depend on how we implemented those since both have the same time complexity. Although AKS would theoretically run in polynomial time, it really is

Table 2: Error as a function of $\log n$

$\log n$	SS	AKS	MR	TD	WS
1	1.0	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	1.0	1.0
3	1.0	1.0	1.0	1.0	1.0
4	0.951203	1.0	0.95053	1.0	1.0
5	0.978582	1.0	0.978538	1.0	1.0
6	0.983269	1.0	0.984793	1.0	1.0
7	0.985784	1.0	0.987807	1.0	1.0
8	0.991804	1.0	0.992822	1.0	1.0
9	0.993558	1.0	0.99467	1.0	1.0
10	0.995892	1.0	0.996872	1.0	1.0
11	0.996757	1.0	0.997659	1.0	1.0
12	0.997884	1.0	0.998644	1.0	1.0
13	0.998752	1.0	0.999046	1.0	1.0
14	0.999077	-	0.999402	1.0	1.0
15	0.999396	-	0.999603	1.0	1.0
16	0.999572	-	0.999767	1.0	1.0
17	0.999736	-	0.999844	1.0	1.0
18	0.999832	-	0.999898	1.0	1.0
19	0.999872	-	0.999928	1.0	1.0
20	0.999923	-	0.999947	1.0	1.0
21	0.99994	-	0.999969	1.0	1.0
22	0.999975	-	0.999988	1.0	1.0
23	0.999971	-	0.999994	1.0	1.0
24	0.999983	-	0.99999	1.0	1.0
25	0.999991	-	0.999992	1.0	1.0
26	0.99999	-	0.999997	1.0	1.0
27	0.999996	-	0.999997	1.0	1.0
28	0.999996	-	1.0	1.0	1.0
29	0.999997	-	1.0	1.0	1.0
30	0.999999	-	1.0	1.0	1.0
31	1.0	-	1.0	1.0	1.0

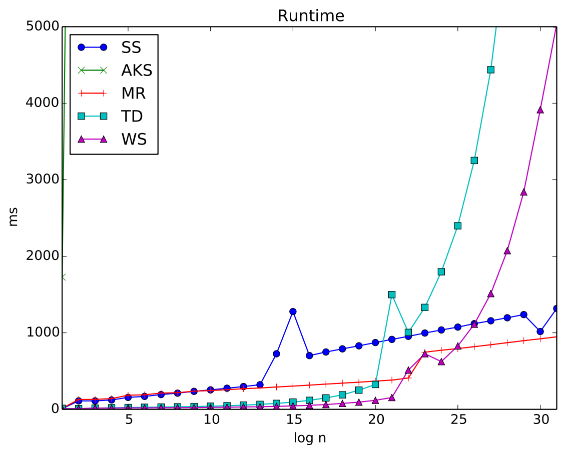

Figure 1: Execution time as function of $\log n$

Table 3: Execution time for AKS in the number range 2-500. Each datapoint consists of 10,000 samples.

Prime numbers included	$O(10^9)$s
Composite numbers only	$O(1)$s

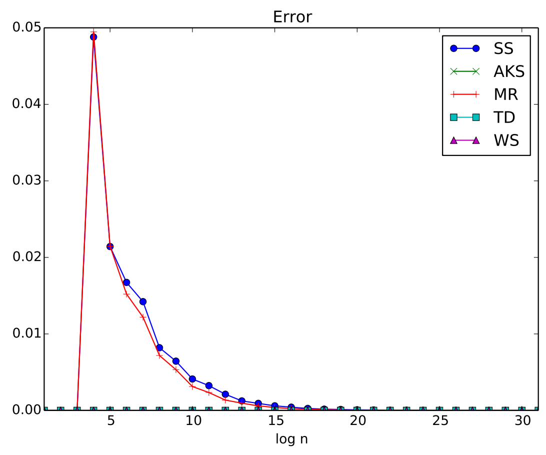

Figure 2: Execution time as function of $\log n$

the snail of the bunch and therefore it has been decided to drop the results for integers above 2^{13}.

Something else that looks interesting is that the error is low for both small inputs and large inputs. For small inputs, it's probably that there are more random numbers for which the SS and MR tests succeed. For large inputs the error is probably small, because for larger numbers, primes get more rare and it is more likely that the algorithm is being tested on a lot of composite numbers.

6. REFERENCES

[1] M. Agrawal, N. Kayal, and N. Saxena. Primes is in p. *Ann. of Math*, 2:781–793, 2002.

[2] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 2007.

[3] M. O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128 – 138, 1980.