PRIMALITY TESTING

INTERMEDIATE SUMMARY

Team – Arijit Banerjee, Suchit Maindola, Srikanth Manikarnike

Problem Statement

The aim of this project is to implement a very widely studied problem of Mathematics – primality testing – as efficiently as possible. In saying so, our main focus is to successfully implement a breakthrough achieved in 2002 by Agrawal, Kayal and Saxena (AKS)[1][8] who came up with the "first published primality-proving algorithm to be simultaneously *general*, *polynomial*, *deterministic*, and *unconditional"* [2]. Although, this is a worst-case polynomial time algorithm that works on any general number deterministically and is not dependent on yet unproven hypothesis, it does not necessarily give the best running time for all possible inputs. There are other polynomial time algorithms like the Miller test, the Lucas-Lehmer test for Mersenne numbers and Pepin's test but these algorithms are "constrained" in some way or the other.

The Miller test (deterministic version of Miller-Rabin algorithm) is fully deterministic and runs in polynomial time over all inputs, but its correctness depends on the truth of the yet-unproven Generalized Reimann Hypothesis (GRH) [2]. The Lucas-Lehmer test for Mersenne numbers, as the name suggests, works only for Mersenne numbers. Similarly Pepin's test can be applied on Fermat's number only.

In course of the project, we want to explore the fastest possible way to determine whether a number is prime or not, AKS algorithm will definitely be our main focus area but we are also interested in building a tunable primality testing system which will combine the completeness of AKS and along with that utilize any special property (e.g. Mersenne numbers) inherent to the number by trying a combination of different methods that work best under the given conditions.

Literature Review

A natural number is said to a prime number if it is greater than one and has no other divisors other than 1 and itself [3]. Prime numbers are used extensively. Public key cryptography, modular arithmetic, pattern recognition are a few examples. The problem of testing the Primality of a given number has existed for 2300 years. Numerous approaches have been tried till date to determine if a given number is prime. Of these, some naïve techniques include – Sieve of Eratosthenes, Pascal's triangle, Winston's theorem [4]. There are probabilistic tests e.g. Fermat's primality test, Miller-Rabin primality test, Solovay-Strassen primality test and deterministic tests such as Pocklington primality test, Elliptic curve primality test, Miller test (which is the deterministic version of the Miller-Rabin primality test under the assumption that GRH is true) etc. [3]. All these methods either have unacceptable runtimes, or have some probability of error in detection, or based on some yet to be proven hypothesis. AKS were the first to come up with a fully deterministic polynomial time algorithm that is not constrained by any limitations [1].

After substantial literature study, we have decided to implement the following three methods

1. AKS Algorithm
   Salembier and Southerington [5] describe optimizations for implementing AKS Primality Test using LiDLA library in C++. They use dgcd() and bgcd() functions for Euclidian division and binary classification respectively. We are planning to use a similar approach for our implementation and later suggest optimizations if time permits.

2. Lucas-Lehmer test for Mersenne numbers
   Crandall and Pomerance [6], in their book "Prime Numbers: A Computational Perspective" provide information on implementing this algorithm. This will be our second strategy, which will be limited only to Mersenne numbers.

3. Miller-Test
   This algorithm is fully deterministic and runs in polynomial time over all inputs, but its correctness depends on the truth of the yet-unproven generalized Reimann hypothesis [7]. Implementation of this algorithm is fairly easier than the previous two strategies.

Our approach

We had planned to implement all the above approaches. Details of each of the approaches is described below:

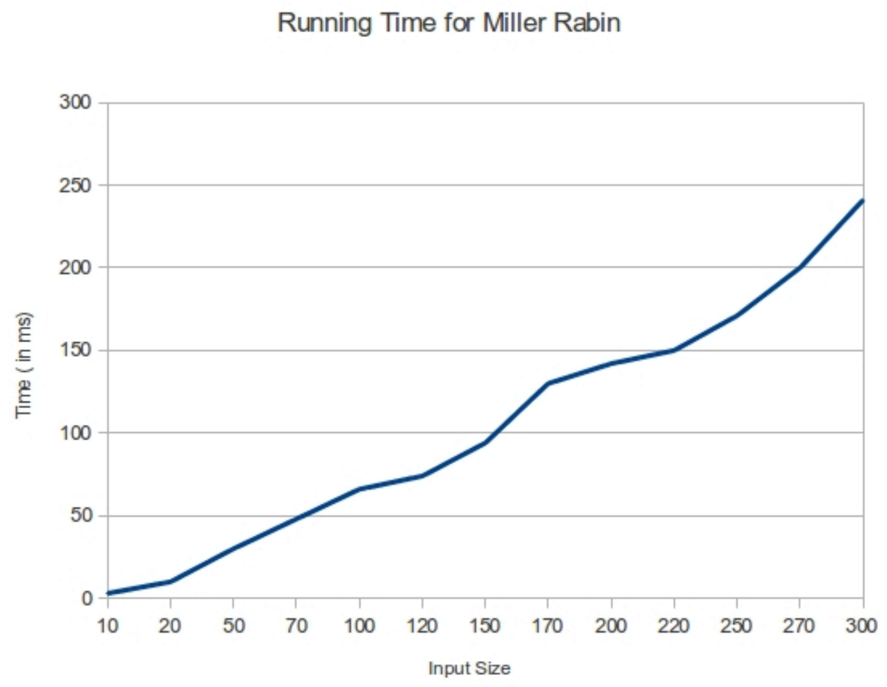
i. Mersenne numbers -
   Identifying a number $M_p = 2^p - 1$ as prime, requires the exponent p to be prime as well. However, the converse is not true always. As of October 2009, 47 Mersenne primes are known with $2^{43,112,609} - 1$ being the largest. Computing Mersenne primes of this sequence is a centuries-old research problem. We propose to handle this issue by maintaining a table of all known Mersenne primes and report primality based on a look up from this table. Determining if a given number is of the type $2^p - 1$ is linear in the size of the input. This approach would also give us an edge while checking for primality

of extremely large numbers of the form 2^p-1. These 47 numbers form a very small part of the input set.

ii. Miller-Rabin Test –
The determinism of the Miller-Rabin Test depends on yet unproven Reimann Hypothesis. Although the full power of GRH is not needed to ensure correctness of this test, we can claim that this supersedes the AKS primality test only for theoretical purposes.

For smaller numbers, trying all values of $a < 2(\ln n)^2$ is not necessary. Much smaller sets will suffice. For instance, inputs of the order of 2^{32} or less, it was proven that, it is sufficient to test for values of a = 2, 7 and 61[7].

We have implemented a non-deterministic version of the algorithm. The results of our tests are attached below.

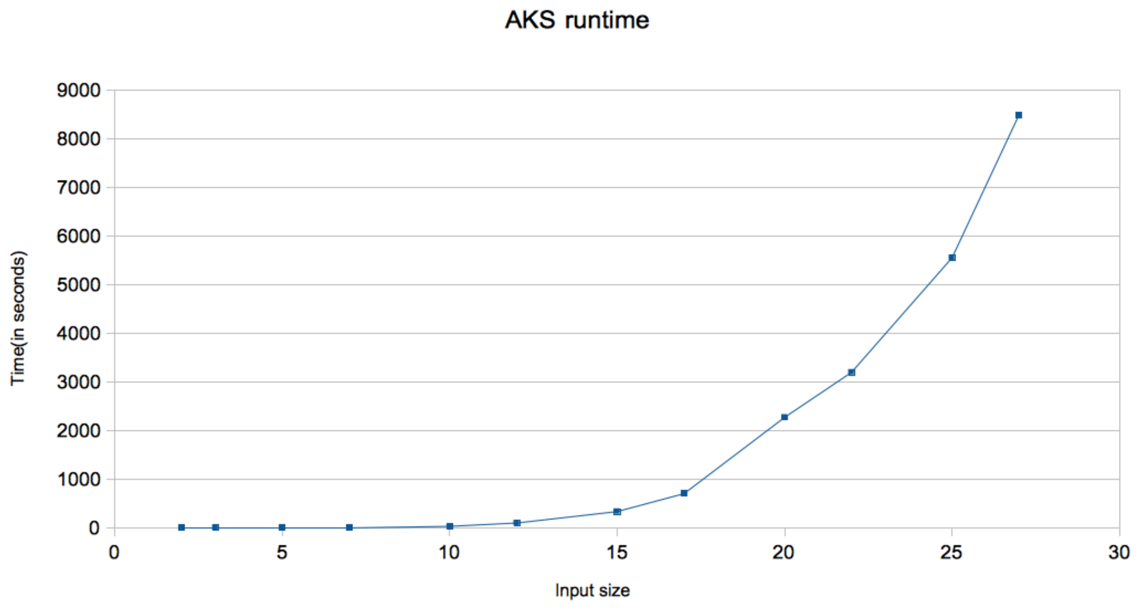


Input size indicates the number of decimal digits in the input and has been taken from [9]. Our implementation has been verified to be true for all our inputs. We propose to enhance its determinism and efficiency by implementing checks on *n* for its corresponding values of *a* respectively.

Value of *n*	**Range of *a***
2,152,302,898,747	2, 3, 5, 7, and 11
3,474,749,660,383	2, 3, 5, 7, 11, and 13
341,550,071,728,321	2, 3, 5, 7, 11, 13, and 17

Test System Configuration
Processor: 64-bit Intel ™Core® i7 ~2.67GHz
Memory: 4GBytes
Operating System: Ubuntu 11.04

iii. AKS Algorithm –
We tried implementing the AKS algorithm using inputs and ideas from [1] and [5] using the NTL library. NTL is a high-performance, portable C++ library providing data structures and algorithms for manipulating arbitrary length integers and implementing time-intensive algorithms [10]. Additionally, we made use of the GMP to provide support for large integers [11]. The running time of AKS algorithm is a lot more than the Miller-Rabin test. The results of our experiments are given below.



Test System Configuration
Processor: 64-bit Intel ™Core® i7 ~2.8GHz
Memory: 4GBytes
Operating System: Ubuntu 10.04

Testing
We have tested the correctness of our AKS algorithm with random primes generated by [9]. Our AKS implementation has been verified to be true for all inputs provided. Although the Miller-Rabin Test's determinism depends on GRH, we have verified the correctness of our Miller-Rabin Test implementation using previously proven results that only a small set of values for *a* that n needs to be tested against to determine whether it is prime or not.

Timeline

- Our initial results suggest that the Miller-Rabin Test is way faster than the AKS implementation. Hence our approach is to integrate the three strategies so that our algorithm uses Miller-Rabin Test up to a point till its deterministic and then using AKS for the rest of the input. In order to get an edge, we could also maintain a table for Mersenne number for extremely large inputs.

- We would also try out other implementations of AKS to improve its efficiency.

References

[1] – Agrawal, Kayal and Saxena. Primes is in p. *Annals of Mathematics* (2004), 781—793.

[2] – Wikipedia. AKS Primality Test. http://en.wikipedia.org/wiki/AKS_primality_test#Importance.

[3] – Wikipedia. Prime number. http://en.wikipedia.org/wiki/Prime_number.

[4] – Scott Aaronson. The Prime Facts: From Euclid to AKS. 2003

[5] – Salembier and Southerington. An Implementation of AKS Primality Test. 2005.

[6] – Crandall and Pomerance. *Prime Numbers: A Computational Perspective*. Springer Verlag. 2005.

[7] – Wikipedia. Miller-Rabin Primality Testing. http://en.wikipedia.org/wiki/Miller_Rabin_primality_test#Deterministic_variants_of_the_test

[8] – Advanced Algorithms, University of Utah, Fall 2011. https://learn-uu.uen.org/courses/48426/files/3870539/download?wrap=1.

[9] – Random Small Primes. http://primes.utm.edu/lists/small/small.html

[10] – NTL: A library for doing Number Theory. http://www.shoup.net/ntl/

[11] – The GNU Multi Precision Arithmetic Library. http://gmplib.org/