PRIMALITY TESTING

INITIAL IMPLEMENTATION PLAN

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 plan to implement all three approaches. Implementing strategy 1 is fairly complicated and would require significant time and effort as compared to strategies 2 and 3. Since the Lucas-Lehmer and Miller-tests work better than the AKS algorithm for certain sets of inputs, we believe that if we can determine whether an input is a Mersenne number or if GRH is true for it, in deterministic polynomial time, we can decide which of the three strategies to apply to our input. This, however, is still conjecture and an approach. We believe we need to explore this idea further for it to improve the efficiency of our primality tester. After we

implement a basic version of the primality tester we will have an idea of the expected running time of our approach.
Any feedback/suggestions on whether this could prove to be a viable option are really appreciated!

Timeline

As of November 1, 3011, we plan to

- Explore our approach to compute the complexity in figuring out whether the input is a Mersenne number and GRH is true for it.
- Implement all three of our strategies.

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.