CS 6150 Project: Primes in P

Aishwarya Asesh (U1063384), Sonam Choudhary (U1069221)

Introduction

The struggle to deterministically determine if a number is Prime is being dealt from a long time. Since the time of ancient Greeks, the classic approach was to check divisibility of n by every number $m \leq \sqrt{n} - 1$. But this proved inefficient as it takes $\Omega(\sqrt{n})$ steps. Fremat's Little theorem needed only polynomial steps to do primality testing. However it wasn't 100% correct as some composites also satisfied the test. Classical primality testing is unique as it is in class co-NP, when n is composite. As observed by Pratt in 1974 this problem is in class NP too. Thus making it NP $\cap$ co-NP.

Miller obtained a deterministic poly time algo for primality testing in 1975 [1]. Soon Robin modified Miller's algo to an unconditional but randomized poly time algo [2]. Solovay and Strassen, extended the approach using a different randomized poly time algo, using property that for a prime n, $(a/n) = a^{(n-1)/2}$ (mod n) for every a [3]. Finally in 1983, Adleman, Pomerance, and Rumely gave a deterministic poly time algo that runs in $(logn)^{O(loglogn)}$ time. In 1986, Goldwasser and Kilian proposed randomized algo based on Elliptic Curves running in poly time for almost all inputs [4]. Later, Adleman and Huang modified Goldwasser-Kilian algo to run it on all possible inputs [5].

In this paper, an unconditional deterministic poly time algo for primality testing, has been achieved in $O^{\sim}(log^{15/2}n)$ time. Main basis of the proposed algo is based on Sophie Germain primes (primes p such that 2p+1 is also a prime). Correctness of the proposed algo can be determined using simple certficates.

Notations and Main Result

The Testing method used is based on identity for pirme numbers which is generalization of Fermat's Little Theorem.
Lemma 1: Let $a \epsilon Z, n \epsilon N, n \geq 2$ and $(a, n) = 1$. Then n is prime if and only if:

$$(X + a)^n = X^n + a(mod n)$$

The above equation is a simple test for primality, for any input n, choose an a and test if above equation is satisfied. But this takes $\Omega(n)$ as n coefficients are evaluated in LHS (for worst case). To improve time complexity we check for satisfiablity:

$$(X + a)^a = X^n + a(mod X^r - 1, n)$$

Thus all primes n satisfy the modified equation for all values of a and r. But some composites also satisfy the condition. Choice of r is appropriate so we know if above equation is satisfied by several a then n must be prime. The number of a and appropriate r are both bounded by a poly in log n, therefore deterministic poly time algo is guaranteed.

Z_n denotes ring of numbers modulo n. F_p denotes finite field with p (prime) elements. h(X) denotes poly of degree d, irreducible in F_p, then $F_p[X]/(h(X))$= finite field of order p^d. Notation $f(X) = g(X)(\text{mod } h(X), n)$ to represent equation $f(X) = g(X)$ in the ring $Z_n[X]/(h(X))$. Symbol $O^{\sim}(t(n) for O(t(n).\text{poly}(logt(n))$, where t(n) is any function of n. Log base 2 is used as default. N and Z denote the set of natural numbers and integers. Order of a modulo r is smallest number k such that $a^k = 1$(mod r). It is denoted as $o_r(a)$. For $r \epsilon N$, $\phi(r)$ is Euler's totient function giving number of numbers less than r that are relatively prime to r. We can observe $o_r(a) | \phi(r)$ for any a, $(a, r) = 1$.

Algorithm for Primality Testing

Input: integer $n > 1$.
1. If $(n = a^b$ for $n \epsilon N$ and $b > 1)$, output COMPOSITE.
2. Find the smallest r such that $o_r(n) > log^2 n$.
3. If $1 < (a, n) < n$ for some $a \leq r$, output COMPOSITE.
4. If $n \leq r$ output PRIME.
5. For a=1 to $\lfloor \sqrt{\phi(r)} \log n \rfloor$ do.
   if $((X + a)^n \neq X^n + a(\text{mod } X^r - 1, n))$, output COMPOSITE.
6. Output PRIME.

Outline of the analysis

The main lemmas are the following:
Lemma 2 If n is prime, the algorithm returns PRIME.
To bound the magnitude of appropriate r, we have lemma 3.
Lemma 3 There exist an $r \leq max\{3, \lceil \log 5n \rceil\}$ such that $o_r(n) > \log^2 n$.
By Lemma 3, we can conclude that $r \leq [log^5 n]$.
Definition 1 For polynomial $f(X)$ and number $m \epsilon$ N , we say that m is introspective for $f(X)$ if

$$[f(X)]^m = f(X^m)(modX^r - 1, p)$$

Lemma 4 If m and m' are introspective numbers for $f(X)$ then so is $m.m'$.
Lemma 5 If m is introspective for $f(X)$ and $g(X)$ then it is also introspective for $f(X)g(X)$.
Lemma 6 (Hendrik Lenstra Jr.) $|G| \geq \binom{t+l}{t-1}$.
Starting by assuming, $f(X)$ and $g(X)$ be 2 polynomials in P, since m is introspective for both f and g, and $h(X)$ divides $X^r - 1$, Lemma 7 subsiquently proves that there exist at least $\binom{t+l}{t-1}$ distinct poly of degree $< t$ in G.
Lemma 7 If n is not a power of p then $|G| \leq n^{\sqrt{t}}$.
From a subset I, where $\hat{I} = \{(n/p)^i.p^j | 0 \leq i, j \leq |\sqrt{t}|\}$, We can say that if n is not a power of p then the set $\hat{I}$ has $(|\sqrt{t}+1)^2 > t$ distinct numbers, which further implies $[f(X)]^{m1} = [f(X)]^{m2}$ which leads to conclusion $|G| \leq n^{\sqrt{t}}$.
Lemma 8 If the algorithm returns PRIME then n is prime.

Main algorithm is governed through the above lemmas. Important lemmas namely, lemma 3 ,6 and 7 bound the values of r and G which subsequently leads to the proof of the algorithm.

Time Complexity Analysis and Improvements

To calculate the time complexity, we note that addition, multiplication, and division operations between two m bits numbers takes time $O(m)$. Similar operations on two degree d polynomials with coefficients at most m bits (size) takes time $O(dm)$ steps. Below mentioned theorems state the shift in the time complexity from $O(\log^{21/2} n)$ to $O(\log^{15/2} n)$. However recently it has been reduced to $O(\log^6 n)$ with modifications in this algorithm by Hendrik Lenstra and Carl Pomerance [6].
Theorem 1 The asymptotic time complexity of the algorithm is $O(\log^{21/2} n)$.
Analysing each step of algorithm, first step takes asymptotic time $O^{\sim}(\log^3 n)$. In second step for a specific value of r, at most $O(\log^2 n)$ multiplications are involved and thus a time complexity of $O^{\sim}(\log^2 n \log r)$. Total time complexity of step 2 is $O^{\sim}(log^7 n)$ as from lemma 3 only $O(log^5 n)$ different r need to be tried. Since each gcd computation takes $O(logn)$time, complexity of step 3 is $O(rlogn) = O(log^6 n)$. And time complexity of step 4 is $O(logn)$. In step 5 $\lfloor \sqrt{\phi(r)} logn \rfloor$ equations are verified, and each equation takes $O^{\sim}(rlog^2 n)$ steps. So the total time complexity comes out to be $O^{\sim}(r\sqrt{\phi(r)} log^3 n) = O^{\sim}(r^{3/2} log^3 n)$ $= O(\log^{21/2} n)$. Hence, the time complexity is $O(\log^{21/2} n)$.
The time complexity was further improved by improving the estimate for r (lemma 3). **Artins Conjecture** and **Sophie-Germain Prime Density Conjecture** supports the idea that for the best case scenerio r would be $O(\log^2 n)$ and the time complexity would be reduced to $O^{\sim}(\log^6 n)$.
Lemma 9 There exist constants $c > 0$ and n such that, for all $x \geq n_0$:

$$|\{q | q = prime, q \leq x \& P(q-1) > q^{2/3}\}| \geq c\frac{x}{lnx}$$

Using the above lemma which holds true for exponents up to 0.6683 the analysis of algorithm is improved.
Theorem 3 The asymptotic time complexity of the algorithm is $O(log^{15/2} n)$.
Proof: From **Artins Conjecture** and **Sophie-Germain Prime Density Conjecture** it is believed that a high density of primes q with $P(q-1) > q^{2/3}$ which implies that step 2 of the algorithm will find an $r = O(log^3 n)$ with $o_r(n) > log^2 n$.This reduces the complexity of the algorithm to $O^{\sim}(log^{15/2} n)$.

Future Work and Extension

Complexity can be improved by reducing the number of iterations of step 5 of the algorithm which runs for $\lfloor \sqrt{\phi(r)} \log n \rfloor$ times to ensure large enough value of G.
Conjecture : If r is prime number that does not divide n and if:

$$(X-1)^n = X^n - 1(modX^r - 1, n)$$

then either n is prime or $n^2 = 1(modr)$.
If above conjecture is true time can be reduced to $O^{\sim}(\log^3 n)$.

Resources / References

1. G. L. Miller. Riemanns hypothesis and tests for primality. J. Comput. Sys. Sci., 13:300317, 1976.
2. M. O. Rabin. Probabilistic algorithm for testing primality. J. Number Theory, 12:128138, 1980.
3. R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. SIAM Journal on Computing, 6:8486, 1977.
4. S. Goldwasser and J Kilian. Almost all primes can be quickly certified. In Proceedings of Annual ACM Symposium on the Theory of Computing, pages 316329, 1986.
5. L. M. Adleman and M.-D. Huang. Primality testing and two dimensional Abelian varieties over finite fields. Lecture Notes in Mathematics, 1512, 1992.
6. H. W. Lenstra, Jr. and Carl Pomerance. Primality testing with gaussian periods. Private communication, March 2003.

PLEASE FIND THE IMPLEMENTATION OF THE ALGORITHM IN THE ATTACHED FILE