Computer Science & Engineering Department

IIT Kharagpur

Computational Number Theory: CS60094

Lecture VII

Instructor: Goutam Biswas *Spring Semester 2014-2015*

1 Test for prime III *(Agrawal, Kayal, and Saxena)*

Miller's polynomial time prime test algorithm depends on the *truth* of the *Extended Riemann Hypothesis*[1] which is unknown. In August 6, 2002, Agrawal, Kayal and Saxena from IIT Kanpur proposed the first polynomial time algorithm for testing prime. This is a remarkable discovery in theory of computing, but it has little practical utility. Probabilistic decision procedures e.g. Miller-Rabin is much more suitable for application, as the probability of error can be made sufficiently small and it runs in lower degree polynomial time[2]. Following is an overview of the AKS-algorithm.

1.1 Polynomial Over a Ring

Let $(R, +, \times, 0, 1)$, be a commutative ring with identity. The set of polynomials of one variable over R, $R[X]$, is defined to be the collection of sequences $\{a_i\}_{i=0}^{\infty}$, where $a_i \in R$, and only finite number of them are non-zero. The largest value of i for a non-zero a_i is called the *degree* of the polynomial.

A polynomial is usually written as

$$a_0 + a_1X + a_2X^2 + \cdots + a_nX^n,$$

where $a_n \neq 0$ and the degree of the polynomial is n. A polynomial of degree zero is an element of R.

If p and q are two polynomials, their addition and multiplication are defined in the usual way over the underlying R.

Example 1. Let $R = \mathbb{Z}_6$ and $p = 4X^2 + 3$ and $q = 3X^5 + 4X^2 + 5X + 4$. Then, $p + q = 3X^5 + [(4+4) \bmod 6]X^2 + 5X + [(3+4) \bmod 6] = 3X^5 + 2X^2 + 5X + 1$. And,
$p \times q = 3X^5 + 4X^4 + 2X^3 + 4X^2 + 3X$.

It is not difficult to prove that $(R[X], +, \times, 0, 1)$ is also a commutative *ring* with identity.

1.2 AKS Algorithm

We shall present the outline of the basic idea of the *Agrawal, Kayal, Saxena*-algorithm for testing prime. We follow [MD] and [VS].

[1] For all real number $s > 1$, the *zeta function* is defined as $\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$. The infinite series converges as $s > 1$. The connection of $\zeta(s)$ with the set of primes is given by *Euler's identity*, $\zeta(s) = \prod_{\text{prime } p} \left(1 - \frac{1}{p^s}\right)^{-1}$. Right-hand side is actually $\lim_{k\to\infty} \prod_{i=1}^{k} \left(1 - \frac{1}{p_i^s}\right)^{-1}$, where p_i is the i^{th} prime.
The thing turns out to be more interesting if we take the domain of $\zeta()$ as complex number i.e. $s \in \mathbb{C}$. Now the series converges absolutely if *Re* $s > 1$. In fact one can extend $\zeta(s)$ nicely only by excluding $s = 1$.
The *Riemann Hypothesis* is as follows: if $s \in \mathbb{C}$, such that $s = (x, y)$, $0 < x < 1$, $\zeta(s) = 0$, then $x = \frac{1}{2}$. So the *non-trivial zeros* of $\zeta()$ should be on the *critical line* $(1/2, y)$. Note that there are *trivial zeros* of $\zeta()$ when s is a negative even integer.

[2] The complexity of the AKS-algorithm using simple implementation of the basic operations is $O((\log n)^{16.5})$ in terms of bit operations. With more sophisticated implementation of basic operations it is $O((\log n)^{10.5})$ bit operations. More sophisticated analysis of the algorithm gives a running time $O((\log n)^{7.5})$. Using some conjecture related to *Sophie Germain prime* this is estimated to $O((\log n)^6)$.

1.2.1 Outline of the Algorithm

Following propositions about the polynomial ring $\mathbb{Z}_n[X]$ characterises the primality of n.
Proposition 1. If n is prime and $a \in \mathbb{Z}_n$, then $(X+a)^n = X^n + a$ in $\mathbb{Z}_n[X]$.
Proof: We use binomial theorem to expand $(X+a)^n$.

$$(X+a)^n = X^n + \sum_{r=1}^{n-1} \binom{n}{r} a^r X^{n-r} + a^n$$

It is known that, if n is prime then $n|\binom{n}{r}$, for $r = 1, \cdots, n-1$. So we have

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

in $\mathbb{Z}_n[X]$. From the Fermat's little theorem, we also have $a^n = a$ if n is a prime. QED.

Proposition 2. If n is not a prime and p is a prime factor of n, then n does not divide $\binom{n}{p}$.
Proof: Let $n = p^k \times m$, where $p \not| m$. Consider

$$\binom{n}{p} = \frac{n(n-1)\cdots(n-p+1)}{p!}.$$

Clearly n in the numerator is divisible by p^k but no other terms in the numerator is divisible by p. The denominator is divisible only by p. So $\binom{n}{p}$ is divisible by p^{k-1} and not by p^k. So it is not divisible by n. QED.

Proposition 3. If n is not prime and $a \in \mathbb{Z}_n^*$, then $(X+a)^n \neq X^n + a$ in $\mathbb{Z}_n[X]$.
Proof: We have already proved that n does not divide $\binom{n}{p}$ where p is a prime factor n. Again $gcd(a, n) = 1$, so $gcd(a^p, n) = 1$. So $\binom{n}{p} a^p \not\equiv 0 (\text{mod } n)$ and a term like $\binom{n}{p} a^p X^{n-p}$ will not be zero in the expansion of $(X+a)^n$. QED.

We may use this characterization to test prime. We choose $a = 1$, use fast exponentiation algorithm to compute $(X+1)^n$ and see whether it is equal to $X^n + 1$. Unfortunately the method is not efficient as there may be many $(O(n))$ non-zero terms in the pre-final stage and its time complexity is worst than trial division which is $O(\sqrt{n})$.

Example 2. Let $n = 7 = 111_2$. We compute in $\mathbb{Z}_7$.

$$\begin{aligned}
& (X+1)^7 \\
= \; & (X+1)^{111_2} \\
= \; & (X+1)^4 \times (X+1)^2 \times (X+1)^1 \\
= \; & (X^4 + 4X^3 + 6X^2 + 4X + 1) \times (X^3 + 3X^2 + 3X + 1) \\
= \; & X^7 + 1
\end{aligned}$$

Now take $n = 6 = 110_6$. The computation is in $\mathbb{Z}_6$.

$$\begin{aligned}
& (X+1)^6 \\
= \; & (X+1)^{110_2} \\
= \; & (X+1)^4 \times (X+1)^2 \\
= \; & (X^4 + 4X^3 + 6X^2 + 4X + 1) \times (X^2 + 2X + 1) \\
= \; & X^6 + 3X^4 + 2X^3 + 3X^2 + 1
\end{aligned}$$

The computation cost is heavy. But instead of this equality in $\mathbb{Z}_n[X]$ one may compute

$$(X+a)^n \equiv X^n + a \;(\text{mod } X^r - 1),$$

with a suitable choice of r. In this case we have to compute $(X+a)^n \bmod (X^r - 1)$ and $(X^n + a) \bmod (X^r - 1)$. The second computation gives us $X^{n \bmod r} + a$.

Example 3.

$$\frac{X^n + a}{X^r - 1} = X^{n-r} + \frac{X^{n-r} + a}{X^r - 1} = \cdots = X^{n-r} + X^{n-2r} + \cdots + X^{n-qr} + \frac{X^{n \bmod r} + a}{X^r - 1}.$$

So $(X^n + a) \bmod (X^r - 1) = X^{n \bmod r} + a$. If $a = 0$, then $X^n \bmod (X^r - 1) = X^{n \bmod r}$.

In the computation of $(X + a)^n \bmod (X^r - 1)$, all coefficients are modulo n and $X^m \equiv X^{m \bmod r} (\bmod\ X^r - 1)$ as $X^m + a\bmod \equiv X^{m \bmod r} + a$ (mod $X^r - 1$), and $a = 0$ gives the result. In the computation process the degree of intermediate polynomials can be kept less than r and the size of the coefficients less than n.

Example 4. Let $n = 7 = 111_2$ and $r = 2$. We compute in $\mathbb{Z}_7$ and $\bmod (X^2 - 1)$.

$$\begin{aligned} & (X+1)^7 \\ \equiv\ & (X+1)^{111_2} \\ \equiv\ & [(X+1)^4 \times (X+1)^2 \times (X+1)^1] \\ \equiv\ & [(X+1) \times 2(X+1) \times (X+1)] \\ \equiv\ & X + 1 \\ \equiv\ & (X^7 + 1) \bmod (X^2 - 1). \end{aligned}$$

Now take $n = 6 = 110_6$. The computation is in $\mathbb{Z}_6$ and mod $(X^2 - 1)$.

$$\begin{aligned} & (X+1)^6 \\ \equiv\ & (X+1)^{110_2} \\ \equiv\ & [(X+1)^4 \times (X+1)^2] \\ \equiv\ & [(X+1) \times 2(X+1)] \\ \equiv\ & 4(X+1) \\ \not\equiv\ & 2 \equiv (X^6 + 1) \bmod (X^2 - 1) \end{aligned}$$

If r is within $O((\log n)^c)$, the computation time is bounded by some polynomial of the length of input. If n is a prime number and $a < n$, then $(X + a)^n \equiv X^{n \bmod r} + a$ (mod $X^r - 1$), for all a and r.

It will be nice to get a single suitable r, not too large in size, as a witness of n as a prime. The theory of AKS-algorithm establishes the sufficiency condition for a suitable r that can be tested in polynomial time. They proved that there is such an $r < n$'s. In fact there is an $r \leq 4\lceil \log n \rceil^2$. As the value r is polynomial in the size of input, it may be found in polynomial time by exhaustive search.

But even with a suitable r it is necessary to check for the equivalence of $(X + a)^n$ and $X^n + a$ modulo $X^r - 1$ in $\mathbb{Z}_n$ for a sequence of a's. They proved that the number of a's are polynomial bounded. But even then the conclusion of the main theorem about n is *not a prime*, but some *power of a prime*. But in algorithm, testing a perfect power can be done in polynomial time, so a prime can be tested in polynomial time.

While searching for r, if it is found that $r|n$, then n is *composite* with r as a factor. Similarly, while going through the sequence of a's if it is found that $(X+a)^n \not\equiv X^{n \bmod r} + a$ (mod $(X^r - 1, n)$), then also n is composite with (r, a) as a witness. The main theorem of the AKS-algorithm is as follows.

Theorem 4. (*Main Theorem*) Let n and r be integers such that

1. $n \geq 3$,
2. $r < n$ is a prime,
3. $a \not| n$, for $2 \leq a \leq r$,
4. order of n in $\mathbb{Z}_r^* > 4(\log n)^2$,
5. $(X + a)^n \equiv X^n + a(\bmod\ X^r - 1)$ in $\mathbb{Z}_n[X]$, for $1 \leq a \leq 2\sqrt{r} \log n$,

then n is a power of a prime.

Following is the AKS-algorithm.

```
isPrimeAKS(n) // n ≥ 2
1   if n = a^b, where a, b ≥ 2, then return 0
2   r ← 2
3   while r < n do
4       if r|n return 0
5       if isPrime(r) then
6           if n^i mod r ≠ 1, ∀i, 1 ≤ i ≤ 4⌈log n⌉^2 then break
7       r ← r + 1
8   if r = n then return 1
9   for a ← 1 to 2⌈√r⌉⌈log n⌉ do
10      if (X + a)^n mod (X^r − 1, n) ≠ X^(n mod r) + a then return 0
11  return 1
```

We assume the correctness of the main theorem and argue that the time complexity of the algorithm is bounded by a polynomial of $\log n$, the input length. We shall not go for any sophisticated analysis.

1.2.2 Analysis of the Algorithm

We assume that the number is represented in binary. For an input n, the largest number generated during computation will not exceed n^2. The size of all intermediate data are bounded by length $2\log n$. So the number of bit operations for all basic arithmetic operations on this data is quadratic of input length, $O((\log n)^2)$.

1. The number of arithmetic operations to test perfect power is $O((\log n)^2 \log\log n)$. The algorithm to test a perfect power and its analysis is given afterward.

2. We claim (without any proof at this point) that the loop of *line 3-7* will be executed for $O((\log n)^5)$ times. We take the number of iteration as $l(n)$. The value of r is incremented in each iteration of the loop and is bounded by $l(n) + 2$.

3. The number of divisions in *line-4* is also $l(n)$.

4. In *line-5* we test whether r is prime. The value of r is bounded by $l(n)$. Even if we use trial division, it takes $O(\sqrt{r})$ trials for each r. So the total number of trials will be $O(l(n)^{3/2})$. If $l(n) = O((\log n)^5)$, the number of iterations is a polynomial of input length[3].

5. If r is a prime, then in *line-6* we test whether the order of n in $\mathbb{Z}_r^*$ is greater than $4\lceil\log n\rceil^2$. If the order of n is $\leq 4\lceil\log n\rceil^2$, we go for the next r. Otherwise we break.

   For each r, we calculate $n^i \bmod r$, for $i = 1, \cdots 4\lceil\log n\rceil^2$. Given an r, the number of multiplications modulo r are $O((\log n)^2)$ ($n_0 = n \bmod r, n_1 = (n_0 \times n) \bmod r, (n_1 \times n) \bmod r, \cdots, n_k = (n_{k-1} \times n) \bmod r$), where $k = 4\lceil\log n\rceil^2$). Considering all iterations (r's), the total number of multiplications are $O(l(n)(\log n)^2)$.

6. If the loop in *line 3-7* terminates at *line 3* (for small values of n), then *line-8* returns 1, indicating n as prime. But when the loop terminates by `break`, the loop of *line 9-10* will be executed with the corresponding value of r.

7. Computation of $\sqrt{r}$ takes $O(r)$ time.
   For each a, $1 \leq a \leq 2\lceil\sqrt{r}\rceil\lceil\log n\rceil$, the calculation of $(X+a)^n \bmod (X^r - 1, n)$ takes place and is compared with $X^{n \bmod r} + a$.

   $(X+a)^n$ takes $O(\log n)$ number of polynomial multiplications over the ring $\mathbb{Z}_n[X]/(X^r - 1)$. Computation of *modulo* $X^r - 1$ is a simple, it replaces X^s by X^{s-r}, whenever $r \leq s < 2r - 1$. So the degree of a polynomial is always smaller than r. Each polynomial multiplication and addition over the ring

[3]A better method is to prepare a *prime table* incrementally. The table contains the primes from 2 to 2^i, when $2^{i-1} < r \leq 2^i$. A variation of the *Sieve of Eratosthenes* is used for the purpose. When the value of r exceeds 2^i, the table is augment with primes up to 2^{i+1}. The total table building cost can be shown to be equal to $O(l(n)\log l(n))$.

$\mathbb{Z}_n[X]/(X^r-1)$ takes $O(r^2)$ multiplication and addition operations of the coefficients in $\mathbb{Z}_n$. So the overall cost of $O(\log n)$ polynomial multiplication is $O((\log n)r^2)$. Taking all a's together we have the following bounds of the number of arithmetic operations. We take the size of r as $l(n)$.

$$O(\sqrt{l(n)}(\log n) \times l(n)^2 \log n) = O(l(n)^{5/2}(\log n)^2.$$

Clearly the computation cost of loop in *line 9-10* dominates. If we take $l(n) = O((\log n)^5)$ which we shall prove, the number of arithmetic operations are $O((\log n)^{14.5})$ and the number of bit operations are $O((\log n)^{16.5})$, as numbers are bounded by n^2.

A sophisticated implementation of the operations and their analysis will give the corresponding figures as $O^{\sim}((\log n)^{9.5})$ and $O^{\sim}((\log n)^{10.5})$.

1.2.3 Small Witness r

AKS proved that the loop of *line 3-7* of their prime testing algorithm will terminate within $20\lceil\log n\rceil^5$ steps. For *small* n where $n < 20\lceil\log n\rceil^5$ e.g. $2^{28} < 20(\lceil\log 2^{28}\rceil)^5$ but $2^{29} < 20(\lceil\log 2^{29}\rceil)^5$, it may terminate when $r = n$ at *line-3*.

For a *large* n there are two possibilities of termination - either r is a divisor of n (*line-4*) or there is a prime number r ($r < 20\lceil\log n\rceil^5$) such that the order of n in $\mathbb{Z}_r^*$ is greater than $4\lceil\log n\rceil^2$. Following is the proposition.

Proposition 5. (A) For $n \geq 2$ there is a prime number r, $2 \leq r \leq 20\lceil\log n\rceil^5$ so that, either $r|n$, or if $r \not| n$, then the order of n in $\mathbb{Z}_r^*$ (smallest i for which $n^i \equiv 1 (\bmod\ r)$) is larger than $4\lceil\log n\rceil^2$.

We shall use the following proposition without proof.

Lemma 6. (B) If $n \geq 2$,

$$\prod_{p \text{ is a prime } \leq 2n} p > 2^n.$$

Proof: (A) If n is "*small*" i.e if $n < 20\lceil\log n\rceil^5$, then n has a prime divisor $< 20\lceil\log n\rceil^5$.

For larger n we shall argue that there is a prime r in the range of $2 \leq r \leq 20\lceil\log n\rceil^5$, such that $n^i \not\equiv 1 (\bmod\ r)$, for all i, $1 \leq i \leq 4\lceil\log n\rceil^2$.

We define

$$P = \prod_{1 \leq i \leq 4\lceil\log n\rceil^2} (n^i - 1).$$

We have

$$\begin{aligned} P &< \prod_{1 \leq i \leq 4\lceil\log n\rceil^2} n^i \\ &= n^{1+2+\cdots+4\lceil\log n\rceil^2} \\ &= n^{\frac{1}{2}4\lceil\log n\rceil^2(4\lceil\log n\rceil^2+1)} \\ &= n^{8\lceil\log n\rceil^4+2\lceil\log n\rceil^2} \\ &< 2^{10\lceil\log n\rceil^5}, \text{for } n \geq 4. \end{aligned}$$

[4] Using the proposition (B) we get

$$\prod_{p \text{ is a prime } \leq 20\lceil\log n\rceil^5} p > 2^{10\lceil\log n\rceil^5} > P.$$

The product of all the primes $\leq 20\lceil\log n\rceil^5$ exceeds P, so there is a prime r that does not divide P - if all of them divides P, then their product also divides P.

As r does not divide $P = \prod_{1 \leq i \leq 4\lceil\log n\rceil^2}(n^i - 1)$, it does not divide $n^i - 1$, for any $i = 1, \cdots, 4\lceil\log n\rceil^2$.

If $r|n$, then n is composite; otherwise $n^i \not\equiv 1(\bmod\ r)$, for any $i = 1, \cdots, 4\lceil\log n\rceil^2$

QED.

[4] If $n^{8(\log n)^4+2(\log n)^2} < 2^{10(\log n)^5}$, then $(8(\log n)^4 + 2(\log n)^2)\log n < 10(\log n)^5$ i.e. $8(\log n)^5 + 2(\log n)^3 < 10(\log n)^5$. If $n = 4$, the left-hand side of the inequality is $8 \cdot 2^5 + 2 \cdot 2^3 = 256 + 16 = 272$ and the right-hand side is $10 \cdot 32 = 320$.

1.2.4 Correctness Proof

If the main theorem is correct, we have the correctness proof of the algorithm.
Theorem 7. If `isPrimeAKS(n)` runs on $n \geq 2$, then it returns 1 if and only if n is a prime.
Proof: n is a prime number ($\Leftarrow$):

- n is not a perfect power, the test of *line 1* fails.
- When within the loop of *line 3-7*, $r < n$ and n is a prime, so $r \not| n$, and the test in *line 4* fails.
- If the loop of *line 3-7* terminates at *line 3* (for small n), then $r = n$ is prime and 1 is returned in *line 8*.
- If the loop terminates at `break`, the order of n in $\mathbb{Z}_r^*$ is greater than $4(\log n)^2$. And the order of n in $\mathbb{Z}_r^*$ must be less than r. So $r > 4(\log n)^2$ i.e. $\sqrt{r} > 2(\log n)$. Therefore $n > r > 2\sqrt{r}\log n$.
- As n is prime, the inequality of *line 9-10* does not holds for any a, $1 \leq a \leq 2\lceil\sqrt{r}\rceil\lceil\log n\rceil$. So 1 is returned in *line-11*.

The algorithm returns 1: This can happen in *line 8* and *line 11*.

- *Line 8* returns 1, only if the exit from the loop is from *line 3* i.e. $r = n$ and no integer in the range 2 to $n - 1$ divides n. So n is prime.
- *Line 11* returns 1, only if the loop of *line 3-7* is terminated by a `break`. We claim that n and r satisfies all the conditions of the *main theorem*. So n should be power of a prime, but perfect power is excluded in *line 1*. So n is a prime.

1. The initial value of r was 2 and at `break` $r < n$. So $n \geq 3$.
2. It is tested whether $r < n$ is a prime at *line 5*.
3. The value of a in the loop of *line 9-10* are in the range of $1, \cdots, 2\sqrt{r}\log n < r$ i.e. the values of a are the values of r in earlier iterations of the loop of *line 3-7*. None of these values divide n, otherwise the loop would have been terminated at *line 4*.
4. The order of n in $\mathbb{Z}_r^* > 4(\log n)^2$ is tested in *line 6* as the condition for `break`.
5. The condition $(X + a)^n \equiv X^n + a(\text{mod } X^r - 1)$ in $\mathbb{Z}_n[X]$, for $1 \leq a \leq 2\sqrt{r}\log n$ is also tested in the loop of *line 9-10*.

QED.

1.3 Perfect Power Test

Following algorithm can be used for perfect power test. We want to see whether the input $n = a^b$, where $a, b \geq 2$.

```
isperfectPower(n) // n ≥ 2
1   b ← 2
2   while 2^b ≤ n do
3       l ← 1, h ← n
4       while h − l ≥ 2 do
5           mid ← (l+h)/2
6           temp ← min{mid^b, n + 1}
7           if n = temp then return (mid, b)
8           if temp < n then l ← mid
9           else h ← mid
10      b ← b + 1
11  return (−1, −1).
```

It is not necessary to calculate mid^b beyond $n + 1$ in *line-6*. The outer loop is executed $O(\log n)$ times. The inner loop is executed $O(\log n)$ times. The exponentiation takes $O(\log b) = O(\log\log n)$ operations. So the number of operations are $O((\log n)^2 \log\log n)$.

References

[AB] *Computational Number Theory* by Abhijit Das, (will be published from CRC Press).

[AKS1] *PRIMES is in P*, by M Agrawal, N Kayal, and Saxena N, preprint, http://www.cse.iitk.ac.in/news/primality.ps August, 8, 2002.

[AKS2] *PRIMES is in P*, by M Agrawal, N Kayal, and Saxena N, preprint (revised), http://www.cse.iitk.ac.in/news/primality_v3.ps March, 1, 2003.

[DGB] *Proving Primality after Agrawal, Kayal and Saxena*, by D G Bernstein, preprint http://cr.yp.to/papers#aks, January 25, 2003.

[MD] *Primality Testing in Polynomial Time From Randomized Algorithms to "PRIMES is in P"* by Martin Dietzfelbinger, LNCS 3000, Pub. Springer, 2004, ISBN 3-540-40344-2.

[RCCP] *Prime Numbers A Computational Perspective* by Richar Crandall & Carl Pomerance, 2nd ed., Pub. Springer, 2010, 978-1-4419-2050-8.

[VS] *A Computational Introduction to Number Theory and Algebra* by Victor Shoup, 2nd ed., Pub. Cambridge University Press, 2009, ISBN 978-0-521-51644-0.