Prime Numbers and Circuits

Nitin Saxena

Hausdorff Center for Mathematics
Bonn

23rd European Conference on Operational Research
Bonn, 8th July 2009

1. Brief History of Primes
2. Primality testing
3. Derandomization?
4. Circuits
5. Primality Derandomized
6. Questions

Outline

Prime Numbers

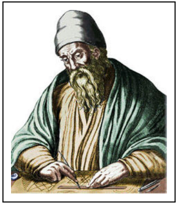


Fig: Euclid

- An integer $n > 1$ is *prime* if its divisors are only 1 and n.
- They are the building blocks of numbers and this means, as Euclid demonstrated in 300 B.C., primes are infinitely many.
- Not only are they pervasive in Mathematics but also appear in practice eg. Cryptography, Communication, ...
- So how do we check and find primes?

Prime Numbers

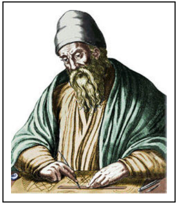


Fig: Euclid

- An integer $n > 1$ is *prime* if its divisors are only 1 and n.
- They are the building blocks of numbers and this means, as Euclid demonstrated in 300 B.C., primes are infinitely many.
- Not only are they pervasive in Mathematics but also appear in practice eg. Cryptography, Communication, ...
- So how do we check and find primes?

Prime Numbers

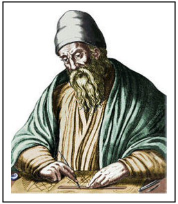


Fig: Euclid

- An integer $n > 1$ is *prime* if its divisors are only 1 and n.
- They are the building blocks of numbers and this means, as Euclid demonstrated in 300 B.C., primes are infinitely many.
- Not only are they pervasive in Mathematics but also appear in practice eg. Cryptography, Communication, ...
- So how do we check and find primes?

Prime Numbers

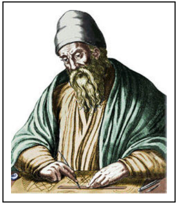


Fig: Euclid

- An integer $n > 1$ is *prime* if its divisors are only 1 and n.
- They are the building blocks of numbers and this means, as Euclid demonstrated in 300 B.C., primes are infinitely many.
- Not only are they pervasive in Mathematics but also appear in practice eg. Cryptography, Communication, ...
- So how do we check and find primes?

Prime Numbers

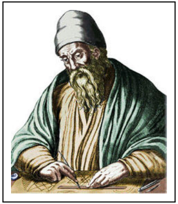


Fig: Euclid

- An integer $n > 1$ is *prime* if its divisors are only 1 and n.
- They are the building blocks of numbers and this means, as Euclid demonstrated in 300 B.C., primes are infinitely many.
- Not only are they pervasive in Mathematics but also appear in practice eg. Cryptography, Communication, ...
- So how do we check and find primes?

Eratosthenes & his sieve

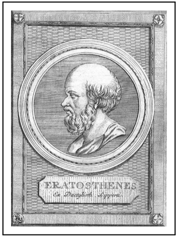


Fig: Eratosthenes

Fig: The Sieve

Eratosthenes & his sieve

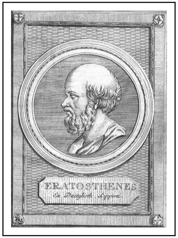


Fig: Eratosthenes

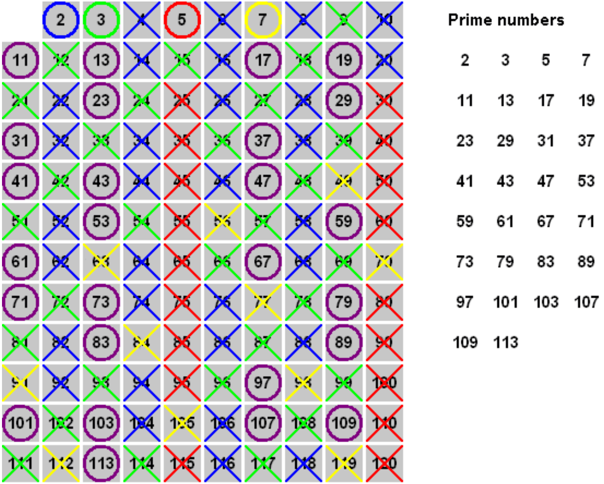


Fig: The Sieve

Eratosthenes Sieve

"Sift the Twos and sift the Threes, The Sieve of Eratosthenes. When the multiples sublime, The numbers that remain are Prime."

- This is the high school method to test primes, attributed to Eratosthenes 200 B.C.
- For a number n, it is sufficient to divide by numbers upto $\sqrt{n}$.
- Thus, it takes around $O(\sqrt{n})$ steps. For a 100-bit number this means 2^{50} steps!

Eratosthenes Sieve

"Sift the Twos and sift the Threes, The Sieve of Eratosthenes. When the multiples sublime, The numbers that remain are Prime."

- This is the high school method to test primes, attributed to Eratosthenes 200 B.C.
- For a number n, it is sufficient to divide by numbers upto $\sqrt{n}$.
- Thus, it takes around $O(\sqrt{n})$ steps. For a 100-bit number this means 2^{50} steps!

Eratosthenes Sieve

"Sift the Twos and sift the Threes, The Sieve of Eratosthenes. When the multiples sublime, The numbers that remain are Prime."

- This is the high school method to test primes, attributed to Eratosthenes 200 B.C.
- For a number n, it is sufficient to divide by numbers upto $\sqrt{n}$.
- Thus, it takes around $O(\sqrt{n})$ steps. For a 100-bit number this means 2^{50} steps!

Eratosthenes Sieve

"Sift the Twos and sift the Threes, The Sieve of Eratosthenes. When the multiples sublime, The numbers that remain are Prime."

- This is the high school method to test primes, attributed to Eratosthenes 200 B.C.
- For a number n, it is sufficient to divide by numbers upto $\sqrt{n}$.
- Thus, it takes around $O(\sqrt{n})$ steps. For a 100-bit number this means 2^{50} steps!

Fermat & his Little Theorem

Theorem (Fermat, 1660s)

If n is prime then for every a, $a^n = a \pmod{n}$.

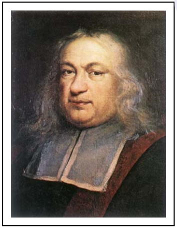


Fig: Fermat

- It is easy to compute $a^n \pmod{n}$ using repeated squaring (i.e. compute sequentially $a \pmod{n}$, $a^2 \pmod{n}$, $a^4 \pmod{n}$,...) this takes time $\log^2 n$, which for a 100-bit number is only 100^2 steps.
- Can we ascertain the primality of n by checking $a^n = a \pmod{n}$ for few *magical* a?
- No! Even if we check it for *most* a (Carmichael, 1910).
- But Fermat gives a starting point!

Fermat & his Little Theorem

Theorem (Fermat, 1660s)

If n is prime then for every a, $a^n = a \pmod n$.

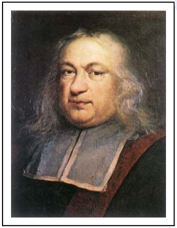


Fig: Fermat

- It is easy to compute $a^n \pmod n$ using repeated squaring (i.e. compute sequentially $a \pmod n$, $a^2 \pmod n$, $a^4 \pmod n$,...) this takes time $\log^2 n$, which for a 100-bit number is only 100^2 steps.
- Can we ascertain the primality of n by checking $a^n = a \pmod n$ for few *magical* a?
- No! Even if we check it for *most* a (Carmichael, 1910).
- But Fermat gives a starting point!

Fermat & his Little Theorem

Theorem (Fermat, 1660s)

If n is prime then for every a, $a^n = a \pmod{n}$.

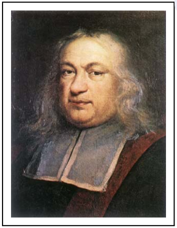


Fig: Fermat

- It is easy to compute $a^n \pmod{n}$ using repeated squaring (i.e. compute sequentially $a \pmod{n}$, $a^2 \pmod{n}$, $a^4 \pmod{n}$,...) this takes time $\log^2 n$, which for a 100-bit number is only 100^2 steps.
- Can we ascertain the primality of n by checking $a^n = a \pmod{n}$ for few *magical* a?
- No! Even if we check it for *most* a (Carmichael, 1910).
- But Fermat gives a starting point!

Fermat & his Little Theorem

Theorem (Fermat, 1660s)

If n is prime then for every a, $a^n = a \pmod{n}$.

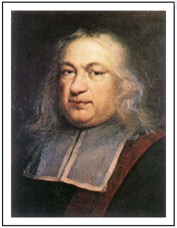


Fig: Fermat

- It is easy to compute $a^n (\text{mod } n)$ using repeated squaring (i.e. compute sequentially $a (\text{mod } n)$, $a^2 (\text{mod } n)$, $a^4 (\text{mod } n)$,...) this takes time $\log^2 n$, which for a 100-bit number is only 100^2 steps.
- Can we ascertain the primality of n by checking $a^n = a \pmod{n}$ for few *magical* a?
- No! Even if we check it for *most* a (Carmichael, 1910).
- But Fermat gives a starting point!

Fermat & his Little Theorem

Theorem (Fermat, 1660s)

If n is prime then for every a, $a^n = a \pmod n$.

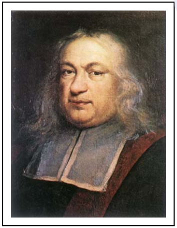


Fig: Fermat

- It is easy to compute $a^n \pmod n$ using repeated squaring (i.e. compute sequentially $a \pmod n$, $a^2 \pmod n$, $a^4 \pmod n$,...) this takes time $\log^2 n$, which for a 100-bit number is only 100^2 steps.
- Can we ascertain the primality of n by checking $a^n = a \pmod n$ for few *magical* a?
- No! Even if we check it for *most* a (Carmichael, 1910).
- But Fermat gives a starting point!

Fermat & his Little Theorem

Theorem (Fermat, 1660s)

If n is prime then for every a, $a^n = a \pmod{n}$.

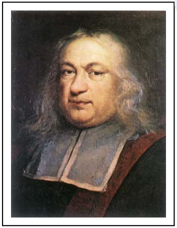


Fig: Fermat

- It is easy to compute $a^n \pmod{n}$ using repeated squaring (i.e. compute sequentially $a \pmod{n}$, $a^2 \pmod{n}$, $a^4 \pmod{n}$, ...) this takes time $\log^2 n$, which for a 100-bit number is only 100^2 steps.
- Can we ascertain the primality of n by checking $a^n = a \pmod{n}$ for few *magical* a?
- No! Even if we check it for *most* a (Carmichael, 1910).
- But Fermat gives a starting point!

Fermat & his Little Theorem

Theorem (Fermat, 1660s)

If n is prime then for every a, $a^n = a \pmod{n}$.

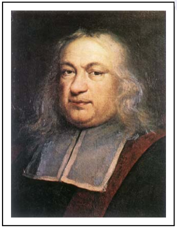


Fig: Fermat

- It is easy to compute $a^n \pmod{n}$ using repeated squaring (i.e. compute sequentially $a \pmod{n}$, $a^2 \pmod{n}$, $a^4 \pmod{n}$,...) this takes time $\log^2 n$, which for a 100-bit number is only 100^2 steps.
- Can we ascertain the primality of n by checking $a^n = a \pmod{n}$ for few *magical* a?
- No! Even if we check it for *most* a (Carmichael, 1910).
- But Fermat gives a starting point!

Prime Number Estimates

Fig: Gauss

- For any real $x > 1$, let $\pi(x)$ be the number of primes $p \leq x$.
- By looking at the tables of primes Legendre and Gauss (independently) conjectured in 1796 that:

$$\pi(x) \text{ might be appoximated by } \frac{x}{\ln x}.$$

Prime Number Estimates

Fig: Gauss

- For any real $x > 1$, let $\pi(x)$ be the number of primes $p \leq x$.
- By looking at the tables of primes Legendre and Gauss (independently) conjectured in 1796 that:

$$\pi(x) \text{ might be appoximated by } \frac{x}{\ln x}.$$

Prime Number Estimates

Fig: Gauss

- For any real $x > 1$, let $\pi(x)$ be the number of primes $p \leq x$.
- By looking at the tables of primes Legendre and Gauss (independently) conjectured in 1796 that:

$\pi(x)$ might be appoximated by $\frac{x}{\ln x}$.

Prime Number Estimates

Fig: Gauss

- For any real $x > 1$, let $\pi(x)$ be the number of primes $p \leq x$.
- By looking at the tables of primes Legendre and Gauss (independently) conjectured in 1796 that:

$$\pi(x) \text{ might be appoximated by } \frac{x}{\ln x}.$$

Prime Number Theorem

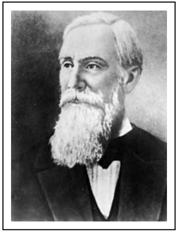


Fig: Chebyshev

- This conjectured estimate was proved by Chebyshev in 1848.
- He found explicit constants c, d around 1 such that:

$$\frac{cx}{\ln x} \leq \pi(x) \leq \frac{dx}{\ln x}$$

- Interestingly, using this he was able to show that there is always a prime between n and $2n$, for any $n \geq 2$.

Prime Number Theorem

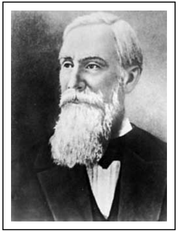


Fig: Chebyshev

- This conjectured estimate was proved by Chebyshev in 1848.
- He found explicit constants c, d around 1 such that:

$$\frac{cx}{\ln x} \leq \pi(x) \leq \frac{dx}{\ln x}$$

- Interestingly, using this he was able to show that there is always a prime between n and $2n$, for any $n \geq 2$.

Prime Number Theorem

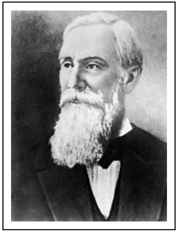


Fig: Chebyshev

- This conjectured estimate was proved by Chebyshev in 1848.
- He found explicit constants c, d around 1 such that:

$$\frac{cx}{\ln x} \leq \pi(x) \leq \frac{dx}{\ln x}$$

- Interestingly, using this he was able to show that there is always a prime between n and $2n$, for any $n \geq 2$.

Prime Number Theorem

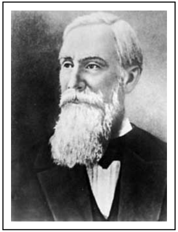


Fig: Chebyshev

- This conjectured estimate was proved by Chebyshev in 1848.
- He found explicit constants c, d around 1 such that:

$$\frac{cx}{\ln x} \leq \pi(x) \leq \frac{dx}{\ln x}$$

- Interestingly, using this he was able to show that there is always a prime between n and $2n$, for any $n \geq 2$.

Outline

Defining Efficiency

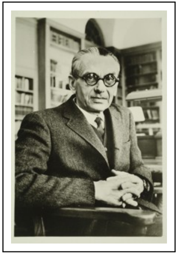


Fig: Gödel

- Kurt Gödel was probably the first to define the question of *primality testing*, and with it a notion of computational *efficiency* itself.
- In 1956, he asked in a letter to John von Neumann: Can we check whether n is a prime in time polynomial in $\log n$.
- This gave the modern question: Is there a polynomial time algorithm for primality?

DEFINING EFFICIENCY

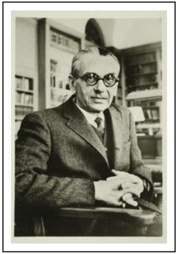


FIG: Gödel

- Kurt Gödel was probably the first to define the question of *primality testing*, and with it a notion of computational *efficiency* itself.
- In 1956, he asked in a letter to John von Neumann: Can we check whether n is a prime in time polynomial in $\log n$.
- This gave the modern question: Is there a polynomial time algorithm for primality?

Defining Efficiency

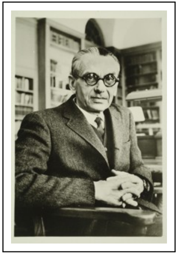


Fig: Gödel

- Kurt Gödel was probably the first to define the question of *primality testing*, and with it a notion of computational *efficiency* itself.
- In 1956, he asked in a letter to John von Neumann: Can we check whether n is a prime in time polynomial in $\log n$.
- This gave the modern question: Is there a polynomial time algorithm for primality?

Defining Efficiency

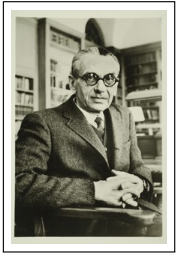


Fig: Gödel

- Kurt Gödel was probably the first to define the question of *primality testing*, and with it a notion of computational *efficiency* itself.
- In 1956, he asked in a letter to John von Neumann: Can we check whether n is a prime in time polynomial in $\log n$.
- This gave the modern question: Is there a polynomial time algorithm for primality?

Can't decide? Toss a coin!

Theorem (Solovay-Strassen, 1977)

An odd number n is prime iff for most a, $a^{\frac{n-1}{2}} = \left(\frac{a}{n}\right) \pmod{n}$.

- Jacobi symbol $\left(\frac{a}{n}\right)$ is computable in time $O^{\sim}(\log^2 n)$.
- We check the above equation for a random a.
- This gives a randomized test that takes time $O^{\sim}(\log^2 n)$.
- It errs with probability at most $\frac{1}{2}$.
- Thus, repeating this process 100 times makes the error probability $\frac{1}{2^{100}}$.

Can't decide? Toss a coin!

Theorem (Solovay-Strassen, 1977)

An odd number n is prime iff for most a, $a^{\frac{n-1}{2}} = \left(\frac{a}{n}\right) \pmod{n}$.

- Jacobi symbol $\left(\frac{a}{n}\right)$ is computable in time $O^{\sim}(\log^2 n)$.
- We check the above equation for a random a.
- This gives a randomized test that takes time $O^{\sim}(\log^2 n)$.
- It errs with probability at most $\frac{1}{2}$.
- Thus, repeating this process 100 times makes the error probability $\frac{1}{2^{100}}$.

Can't decide? Toss a coin!

Theorem (Solovay-Strassen, 1977)

An odd number n is prime iff for most a, $a^{\frac{n-1}{2}} = \left(\frac{a}{n}\right) \pmod{n}$.

- Jacobi symbol $\left(\frac{a}{n}\right)$ is computable in time $O^{\sim}(\log^2 n)$.
- We check the above equation for a random a.
- This gives a randomized test that takes time $O^{\sim}(\log^2 n)$.
- It errs with probability at most $\frac{1}{2}$.
- Thus, repeating this process 100 times makes the error probability $\frac{1}{2^{100}}$.

Can't decide? Toss a coin!

Theorem (Solovay-Strassen, 1977)

An odd number n is prime iff for most a, $a^{\frac{n-1}{2}} = \left(\frac{a}{n}\right) \pmod{n}$.

- Jacobi symbol $\left(\frac{a}{n}\right)$ is computable in time $O^{\sim}(\log^2 n)$.
- We check the above equation for a random a.
- This gives a randomized test that takes time $O^{\sim}(\log^2 n)$.
- It errs with probability at most $\frac{1}{2}$.
- Thus, repeating this process 100 times makes the error probability $\frac{1}{2^{100}}$.

Can't decide? Toss a coin!

Theorem (Solovay-Strassen, 1977)

An odd number n is prime iff for most a, $a^{\frac{n-1}{2}} = \left(\frac{a}{n}\right) \pmod{n}$.

- Jacobi symbol $\left(\frac{a}{n}\right)$ is computable in time $O^{\sim}(\log^2 n)$.
- We check the above equation for a random a.
- This gives a randomized test that takes time $O^{\sim}(\log^2 n)$.
- It errs with probability at most $\frac{1}{2}$.
- Thus, repeating this process 100 times makes the error probability $\frac{1}{2^{100}}$.

Can't decide? Toss a coin!

Theorem (Solovay-Strassen, 1977)

An odd number n is prime iff for most a, $a^{\frac{n-1}{2}} = \left(\frac{a}{n}\right) \pmod{n}$.

- Jacobi symbol $\left(\frac{a}{n}\right)$ is computable in time $O^{\sim}(\log^2 n)$.
- We check the above equation for a random a.
- This gives a randomized test that takes time $O^{\sim}(\log^2 n)$.
- It errs with probability at most $\frac{1}{2}$.
- Thus, repeating this process 100 times makes the error probability $\frac{1}{2^{100}}$.

Primality: A Practical Solution

Theorem (Miller-Rabin, 1980)

An odd number $n = 1 + 2^s \cdot t$ (odd t) is prime iff for most $a \in \mathbb{Z}_n$, the sequence $a^{2^{s-1} \cdot t}, a^{2^{s-2} \cdot t}, \ldots, a^t$ has either a -1 or all 1's.

- We check the above condition for a random a.
- This gives a randomized test that takes time $O^{\sim}(\log^2 n)$.
- It errs with probability at most $\frac{1}{4}$.
- The most popular primality test!

Primality: A Practical Solution

Theorem (Miller-Rabin, 1980)

An odd number $n = 1 + 2^s \cdot t$ (odd t) is prime iff for most $a \in \mathbb{Z}_n$, the sequence $a^{2^{s-1} \cdot t}, a^{2^{s-2} \cdot t}, \ldots, a^t$ has either a -1 or all 1's.

- We check the above condition for a random a.
- This gives a randomized test that takes time $O^{\sim}(\log^2 n)$.
- It errs with probability at most $\frac{1}{4}$.
- The most popular primality test!

Primality: A Practical Solution

Theorem (Miller-Rabin, 1980)

An odd number $n = 1 + 2^s \cdot t$ (odd t) is prime iff for most $a \in \mathbb{Z}_n$, the sequence $a^{2^{s-1}\cdot t}, a^{2^{s-2}\cdot t}, \ldots, a^t$ has either a -1 or all 1's.

- We check the above condition for a random a.
- This gives a randomized test that takes time $O^{\sim}(\log^2 n)$.
- It errs with probability at most $\frac{1}{4}$.
- The most popular primality test!

Primality: A Practical Solution

Theorem (Miller-Rabin, 1980)

An odd number $n = 1 + 2^s \cdot t$ (odd t) is prime iff for most $a \in \mathbb{Z}_n$, the sequence $a^{2^{s-1}\cdot t}, a^{2^{s-2}\cdot t}, \ldots, a^t$ has either a -1 or all 1's.

- We check the above condition for a random a.
- This gives a randomized test that takes time $O^{\sim}(\log^2 n)$.
- It errs with probability at most $\frac{1}{4}$.
- The most popular primality test!

Primality: A Practical Solution

Theorem (Miller-Rabin, 1980)

An odd number $n = 1 + 2^s \cdot t$ (odd t) is prime iff for most $a \in \mathbb{Z}_n$, the sequence $a^{2^{s-1} \cdot t}, a^{2^{s-2} \cdot t}, \ldots, a^t$ has either a -1 or all 1's.

- We check the above condition for a random a.
- This gives a randomized test that takes time $O^{\sim}(\log^2 n)$.
- It errs with probability at most $\frac{1}{4}$.
- The most popular primality test!

Outline

Determinism and Randomness: Different?

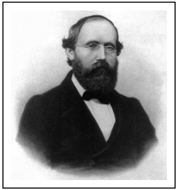


Fig: Riemann

- Can we select the random bits carefully in a randomized algorithm such that there is no error?
- For example, if we assume generalized Riemann Hypothesis (GRH) then the first $(2\log^2 n)$ a's suffice to test primality of n in Solovay-Strassen and Miller-Rabin tests.
- Can we derandomize any randomized polynomial time algorithm?
- Is BPP=P? or

"God does not play dice...."??

Determinism and Randomness: Different?

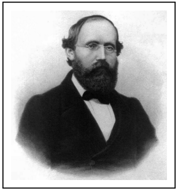


Fig: Riemann

- Can we select the random bits carefully in a randomized algorithm such that there is no error?
- For example, if we assume generalized Riemann Hypothesis (GRH) then the first $(2\log^2 n)$ a's suffice to test primality of n in Solovay-Strassen and Miller-Rabin tests.
- Can we derandomize any randomized polynomial time algorithm?
- Is BPP=P? or

"God does not play dice...."??

Determinism and Randomness: Different?

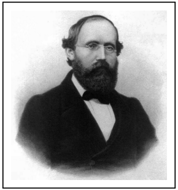


Fig: Riemann

- Can we select the random bits carefully in a randomized algorithm such that there is no error?
- For example, if we assume generalized Riemann Hypothesis (GRH) then the first $(2 \log^2 n)$ a's suffice to test primality of n in Solovay-Strassen and Miller-Rabin tests.
- Can we derandomize any randomized polynomial time algorithm?
- Is BPP=P? or

"God does not play dice...."??

Determinism and Randomness: Different?

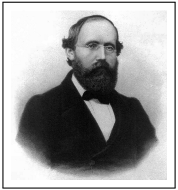


Fig: Riemann

- Can we select the random bits carefully in a randomized algorithm such that there is no error?
- For example, if we assume generalized Riemann Hypothesis (GRH) then the first $(2\log^2 n)$ a's suffice to test primality of n in Solovay-Strassen and Miller-Rabin tests.
- Can we derandomize any randomized polynomial time algorithm?
- Is BPP=P? or

"God does not play dice...." ??

Determinism and Randomness: Different?

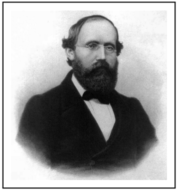


Fig: Riemann

- Can we select the random bits carefully in a randomized algorithm such that there is no error?
- For example, if we assume generalized Riemann Hypothesis (GRH) then the first $(2\log^2 n)$ a's suffice to test primality of n in Solovay-Strassen and Miller-Rabin tests.
- Can we derandomize any randomized polynomial time algorithm?
- Is BPP$=$P? or

"God does not play dice...."??

Determinism and Randomness: Hardness Enters

- In the 1990s it was observed that if there are hard problems then they can be used to derandomize.
- Specifically, Impagliazzo & Wigderson showed in 1997 that BPP=P if E has exponentially hard functions.
- But proving hardness has always been a hard problem!
- Some hoped that Primality might have an easier proof. After all, there were several intermediate results in that direction.

Determinism and Randomness: Hardness Enters

- In the 1990s it was observed that if there are hard problems then they can be used to derandomize.
- Specifically, Impagliazzo & Wigderson showed in 1997 that BPP=P if E has exponentially hard functions.
- But proving hardness has always been a hard problem!
- Some hoped that Primality might have an easier proof. After all, there were several intermediate results in that direction.

Determinism and Randomness: Hardness Enters

- In the 1990s it was observed that if there are hard problems then they can be used to derandomize.
- Specifically, Impagliazzo & Wigderson showed in 1997 that BPP=P if E has exponentially hard functions.
- But proving hardness has always been a hard problem!
- Some hoped that Primality might have an easier proof. After all, there were several intermediate results in that direction.

Determinism and Randomness: Hardness Enters

- In the 1990s it was observed that if there are hard problems then they can be used to derandomize.
- Specifically, Impagliazzo & Wigderson showed in 1997 that BPP=P if E has exponentially hard functions.
- But proving hardness has always been a hard problem!
- Some hoped that Primality might have an easier proof. After all, there were several intermediate results in that direction.

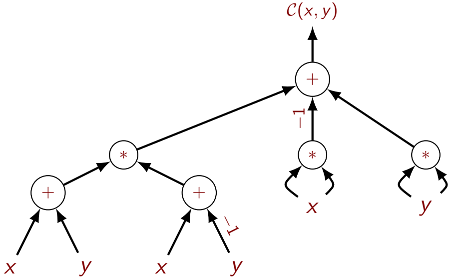
Outline

Primality Testing & Circuits

- Finally, the answer came forth by a rephrasal of primality testing in terms of an *arithmetic circuit*.
- A circuit C over a ring R is a directed acyclic graph with inputs at the leaves, output at the root, $+$ and $*$ as internal nodes, and constants from R at the edges.

Primality Testing & Circuits

- Finally, the answer came forth by a rephrasal of primality testing in terms of an *arithmetic circuit*.
- A circuit $\mathcal{C}$ over a ring R is a directed acyclic graph with inputs at the leaves, output at the root, $+$ and $*$ as internal nodes, and constants from R at the edges.



Primality & Zero Circuits

- For any integers $n > 0$ and $1 \leq a \leq n$ define a circuit $C_{n,a}(x) := (x+a)^n - (x^n + a) \pmod{n}$.
- Note that, using repeated squaring, circuit $C_{n,a}$ can be expressed as a directed acyclic graph of size $O(\log n)$.
- It is a simple property of binomial coefficients that:

$$n \text{ is prime iff } C_{n,1}(x) = 0.$$

- It can be viewed as a generalization of Fermat's little theorem.
- It was used by Agrawal & Biswas (1999) to give a new kind of randomized primality test.

Primality & Zero Circuits

- For any integers $n > 0$ and $1 \leq a \leq n$ define a circuit $C_{n,a}(x) := (x+a)^n - (x^n + a) \pmod{n}$.
- Note that, using repeated squaring, circuit $C_{n,a}$ can be expressed as a directed acyclic graph of size $O(\log n)$.
- It is a simple property of binomial coefficients that:

$$n \text{ is prime iff } C_{n,1}(x) = 0.$$

- It can be viewed as a generalization of Fermat's little theorem.
- It was used by Agrawal & Biswas (1999) to give a new kind of randomized primality test.

Primality & Zero Circuits

- For any integers $n > 0$ and $1 \leq a \leq n$ define a circuit $C_{n,a}(x) := (x + a)^n - (x^n + a) \pmod{n}$.
- Note that, using repeated squaring, circuit $C_{n,a}$ can be expressed as a directed acyclic graph of size $O(\log n)$.
- It is a simple property of binomial coefficients that:

$$n \text{ is prime iff } C_{n,1}(x) = 0.$$

- It can be viewed as a generalization of Fermat's little theorem.
- It was used by Agrawal & Biswas (1999) to give a new kind of randomized primality test.

Primality & Zero Circuits

- For any integers $n > 0$ and $1 \leq a \leq n$ define a circuit $C_{n,a}(x) := (x+a)^n - (x^n + a) \pmod{n}$.
- Note that, using repeated squaring, circuit $C_{n,a}$ can be expressed as a directed acyclic graph of size $O(\log n)$.
- It is a simple property of binomial coefficients that:

$$n \text{ is prime iff } C_{n,1}(x) = 0.$$

- It can be viewed as a generalization of Fermat's little theorem.
- It was used by Agrawal & Biswas (1999) to give a new kind of randomized primality test.

Primality & Zero Circuits

- For any integers $n > 0$ and $1 \leq a \leq n$ define a circuit $C_{n,a}(x) := (x+a)^n - (x^n + a) \pmod{n}$.
- Note that, using repeated squaring, circuit $C_{n,a}$ can be expressed as a directed acyclic graph of size $O(\log n)$.
- It is a simple property of binomial coefficients that:

$$n \text{ is prime iff } C_{n,1}(x) = 0.$$

- It can be viewed as a generalization of Fermat's little theorem.
- It was used by Agrawal & Biswas (1999) to give a new kind of randomized primality test.

Primality & Zero Circuits

- For any integers $n > 0$ and $1 \leq a \leq n$ define a circuit $C_{n,a}(x) := (x+a)^n - (x^n + a) \pmod{n}$.
- Note that, using repeated squaring, circuit $C_{n,a}$ can be expressed as a directed acyclic graph of size $O(\log n)$.
- It is a simple property of binomial coefficients that:

$$n \text{ is prime iff } C_{n,1}(x) = 0.$$

- It can be viewed as a generalization of Fermat's little theorem.
- It was used by Agrawal & Biswas (1999) to give a new kind of randomized primality test.

Outline

The Idea

- Although $C_{n,a}(x) := (x+a)^n - (x^n + a) \pmod{n}$ is a $O(\log n)$ sized circuit, checking it for zeroness seems to require computing all the n terms in the expansion of $(x+a)^n$.
- However, if r is "small" we can check $C_{n,a}(x) = 0 \pmod{x^r - 1}$ efficiently.
- Does checking this for few different a & r imply $C_{n,1}(x) = 0$?
- Agrawal, Kayal & Saxena (2002) showed that a, r below $(\log n)^5$ will do!
- It was the first unconditional, deterministic and polynomial time primality test.

The Idea

- Although $C_{n,a}(x) := (x+a)^n - (x^n + a) \pmod{n}$ is a $O(\log n)$ sized circuit, checking it for zeroness seems to require computing all the n terms in the expansion of $(x+a)^n$.
- However, if r is "small" we can check $C_{n,a}(x) = 0 \pmod{x^r - 1}$ efficiently.
- Does checking this for few different a & r imply $C_{n,1}(x) = 0$?
- Agrawal, Kayal & Saxena (2002) showed that a, r below $(\log n)^5$ will do!
- It was the first unconditional, deterministic and polynomial time primality test.

THE IDEA

- Although $C_{n,a}(x) := (x+a)^n - (x^n + a) \pmod{n}$ is a $O(\log n)$ sized circuit, checking it for zeroness seems to require computing all the n terms in the expansion of $(x+a)^n$.
- However, if r is "small" we can check $C_{n,a}(x) = 0 \pmod{x^r - 1}$ efficiently.
- Does checking this for few different a & r imply $C_{n,1}(x) = 0$?
- Agrawal, Kayal & Saxena (2002) showed that a, r below $(\log n)^5$ will do!
- It was the first unconditional, deterministic and polynomial time primality test.

The Idea

- Although $C_{n,a}(x) := (x+a)^n - (x^n + a) \pmod{n}$ is a $O(\log n)$ sized circuit, checking it for zeroness seems to require computing all the n terms in the expansion of $(x+a)^n$.
- However, if r is "small" we can check $C_{n,a}(x) = 0 \pmod{x^r - 1}$ efficiently.
- Does checking this for few different a & r imply $C_{n,1}(x) = 0$?
- Agrawal, Kayal & Saxena (2002) showed that a, r below $(\log n)^5$ will do!
- It was the first unconditional, deterministic and polynomial time primality test.

The Idea

- Although $C_{n,a}(x) := (x+a)^n - (x^n + a) \pmod{n}$ is a $O(\log n)$ sized circuit, checking it for zeroness seems to require computing all the n terms in the expansion of $(x+a)^n$.
- However, if r is "small" we can check $C_{n,a}(x) = 0 \pmod{x^r - 1}$ efficiently.
- Does checking this for few different a & r imply $C_{n,1}(x) = 0$?
- Agrawal, Kayal & Saxena (2002) showed that a, r below $(\log n)^5$ will do!
- It was the first unconditional, deterministic and polynomial time primality test.

Agrawal-Kayal-S Test

1. If n is a^b $(b > 1)$, it is composite.
2. Select an r such that $\mathrm{ord}_r(n) > 4\log^2 n$ and work in the ring $R := \mathbb{Z}_n[x]/(x^r - 1)$.
3. For each a, $1 \leq a \leq \ell := \lceil 2\sqrt{r}\log n \rceil$, check if $(x+a)^n = (x^n + a)$.
4. If yes then n is prime else composite.

Agrawal-Kayal-S Test

1. If n is a^b $(b > 1)$, it is composite.
2. Select an r such that $\mathrm{ord}_r(n) > 4\log^2 n$ and work in the ring $R := \mathbb{Z}_n[x]/(x^r - 1)$.
3. For each a, $1 \leq a \leq \ell := \lceil 2\sqrt{r}\log n \rceil$, check if $(x+a)^n = (x^n + a)$.
4. If yes then n is prime else composite.

Agrawal-Kayal-S Test

1. If n is a^b $(b > 1)$, it is composite.
2. Select an r such that $\text{ord}_r(n) > 4\log^2 n$ and work in the ring $R := \mathbb{Z}_n[x]/(x^r - 1)$.
3. For each a, $1 \le a \le \ell := \lceil 2\sqrt{r}\log n \rceil$, check if $(x+a)^n = (x^n + a)$.
4. If yes then n is prime else composite.

Agrawal-Kayal-S Test

1. If n is a^b $(b > 1)$, it is composite.
2. Select an r such that $\mathrm{ord}_r(n) > 4\log^2 n$ and work in the ring $R := \mathbb{Z}_n[x]/(x^r - 1)$.
3. For each a, $1 \le a \le \ell := \lceil 2\sqrt{r}\log n \rceil$, check if $(x+a)^n = (x^n + a)$.
4. If yes then n is prime else composite.

AKS Test: The Proof

- Suppose all the congruences hold and p is a prime factor of n.
- The group $I := \langle n, p \pmod{r} \rangle$. $t := \#I \geq \mathrm{ord}_r(n) \geq 4 \log^2 n$.
- The group $J := \langle (x+1), \ldots, (x+\ell) \pmod{p, h(x)} \rangle$ where $h(x)$ is an irreducible factor of $\frac{x^r-1}{x-1}$ modulo p.

  $\#J \geq 2^{\min\{t,\ell\}} > 2^{2\sqrt{t}\log n} \geq n^{2\sqrt{t}}$.
- *Proof:* Let $f(x), g(x)$ be two different products of $(x+a)$'s, having degree $< t$. Suppose $f(x) = g(x) \pmod{p, h(x)}$.
  - The test tells us that $f(x^{n^i \cdot p^j}) = g(x^{n^i \cdot p^j}) \pmod{p, h(x)}$.
  - But this means that $f(z) - g(z)$ has atleast t roots in the field $\mathbb{F}_p[x]/(h(x))$, which is a contradiction.

AKS Test: The Proof

- Suppose all the congruences hold and p is a prime factor of n.
- The group $I := \langle n, p \pmod r \rangle$. $t := \#I \geq \mathrm{ord}_r(n) \geq 4\log^2 n$.
- The group $J := \langle (x+1), \ldots, (x+\ell) \pmod{p, h(x)} \rangle$ where $h(x)$ is an irreducible factor of $\frac{x^r-1}{x-1}$ modulo p.

  $\#J \geq 2^{\min\{t,\ell\}} > 2^{2\sqrt{t}\log n} \geq n^{2\sqrt{t}}$.
- *Proof:* Let $f(x), g(x)$ be two different products of $(x+a)$'s, having degree $< t$. Suppose $f(x) = g(x) \pmod{p, h(x)}$.
  - The test tells us that $f(x^{n^i \cdot p^j}) = g(x^{n^i \cdot p^j}) \pmod{p, h(x)}$.
  - But this means that $f(z) - g(z)$ has atleast t roots in the field $\mathbb{F}_p[x]/(h(x))$, which is a contradiction.

AKS Test: The Proof

- Suppose all the congruences hold and p is a prime factor of n.
- The group $I := \langle n, p \pmod{r} \rangle$. $t := \#I \geq \text{ord}_r(n) \geq 4\log^2 n$.
- The group $J := \langle (x+1), \ldots, (x+\ell) \pmod{p, h(x)} \rangle$ where $h(x)$ is an irreducible factor of $\frac{x^r-1}{x-1}$ modulo p.

  $\#J \geq 2^{\min\{t,\ell\}} \geq 2^{2\sqrt{t}\log n} \geq n^{2\sqrt{t}}$.
- *Proof:* Let $f(x), g(x)$ be two different products of $(x+a)$'s, having degree $< t$. Suppose $f(x) = g(x) \pmod{p, h(x)}$.
  - The test tells us that $f(x^{n^i \cdot p^j}) = g(x^{n^i \cdot p^j}) \pmod{p, h(x)}$.
  - But this means that $f(z) - g(z)$ has atleast t roots in the field $\mathbb{F}_p[x]/(h(x))$, which is a contradiction.

AKS Test: The Proof

- Suppose all the congruences hold and p is a prime factor of n.
- The group $I := \langle n, p \pmod{r} \rangle$. $t := \#I \geq \mathrm{ord}_r(n) \geq 4\log^2 n$.
- The group $J := \langle (x+1), \ldots, (x+\ell) \pmod{p, h(x)} \rangle$ where $h(x)$ is an irreducible factor of $\frac{x^r-1}{x-1}$ modulo p.
  $\#J \geq 2^{\min\{t,\ell\}} > 2^{2\sqrt{t}\log n} \geq n^{2\sqrt{t}}$.
- *Proof:* Let $f(x), g(x)$ be two different products of $(x+a)$'s, having degree $< t$. Suppose $f(x) = g(x) \pmod{p, h(x)}$.
  - The test tells us that $f(x^{n^i \cdot p^j}) = g(x^{n^i \cdot p^j}) \pmod{p, h(x)}$.
  - But this means that $f(z) - g(z)$ has atleast t roots in the field $\mathbb{F}_p[x]/(h(x))$, which is a contradiction.

AKS Test: The Proof

- Suppose all the congruences hold and p is a prime factor of n.
- The group $I := \langle n, p \pmod{r} \rangle$. $t := \#I \geq \mathrm{ord}_r(n) \geq 4\log^2 n$.
- The group $J := \langle (x+1), \ldots, (x+\ell) \pmod{p, h(x)} \rangle$ where $h(x)$ is an irreducible factor of $\frac{x^r-1}{x-1}$ modulo p.
  $\#J \geq 2^{\min\{t,\ell\}} > 2^{2\sqrt{t}\log n} \geq n^{2\sqrt{t}}$.
- *Proof:* Let $f(x), g(x)$ be two different products of $(x+a)$'s, having degree $< t$. Suppose $f(x) = g(x) \pmod{p, h(x)}$.
  - The test tells us that $f(x^{n^i \cdot p^j}) = g(x^{n^i \cdot p^j}) \pmod{p, h(x)}$.
  - But this means that $f(z) - g(z)$ has atleast t roots in the field $\mathbb{F}_p[x]/(h(x))$, which is a contradiction.

AKS Test: The Proof

- Suppose all the congruences hold and p is a prime factor of n.
- The group $I := \langle n, p \pmod r \rangle$. $t := \#I \geq \mathrm{ord}_r(n) \geq 4\log^2 n$.
- The group $J := \langle (x+1), \ldots, (x+\ell) \pmod{p, h(x)} \rangle$ where $h(x)$ is an irreducible factor of $\frac{x^r-1}{x-1}$ modulo p.
  $\#J \geq 2^{\min\{t,\ell\}} > 2^{2\sqrt{t}\log n} \geq n^{2\sqrt{t}}$.
- *Proof:* Let $f(x), g(x)$ be two different products of $(x+a)$'s, having degree $< t$. Suppose $f(x) = g(x) \pmod{p, h(x)}$.
  - The test tells us that $f(x^{n^i \cdot p^j}) = g(x^{n^i \cdot p^j}) \pmod{p, h(x)}$.
  - But this means that $f(z) - g(z)$ has atleast t roots in the field $\mathbb{F}_p[x]/(h(x))$, which is a contradiction.

AKS Test: The Proof

- Suppose all the congruences hold and p is a prime factor of n.
- The group $I := \langle n, p \pmod{r} \rangle$. $t := \#I \geq \mathrm{ord}_r(n) \geq 4 \log^2 n$.
- The group $J := \langle (x+1), \ldots, (x+\ell) \pmod{p, h(x)} \rangle$ where $h(x)$ is an irreducible factor of $\frac{x^r-1}{x-1}$ modulo p.
  $\#J \geq 2^{\min\{t,\ell\}} > 2^{2\sqrt{t}\log n} \geq n^{2\sqrt{t}}$.
- *Proof:* Let $f(x), g(x)$ be two different products of $(x+a)$'s, having degree $< t$. Suppose $f(x) = g(x) \pmod{p, h(x)}$.
  - The test tells us that $f(x^{n^i \cdot p^j}) = g(x^{n^i \cdot p^j}) \pmod{p, h(x)}$.
  - But this means that $f(z) - g(z)$ has atleast t roots in the field $\mathbb{F}_p[x]/(h(x))$, which is a contradiction.

AKS Test: The Proof

- Suppose all the congruences hold and p is a prime factor of n.
- The group $I := \langle n, p \pmod{r} \rangle$. $t := \#I \geq \text{ord}_r(n) \geq 4 \log^2 n$.
- The group $J := \langle (x+1), \ldots, (x+\ell) \pmod{p, h(x)} \rangle$ where $h(x)$ is an irreducible factor of $\frac{x^r - 1}{x-1}$ modulo p.
  $\#J \geq 2^{\min\{t, \ell\}} > 2^{2\sqrt{t} \log n} \geq n^{2\sqrt{t}}$.
- *Proof:* Let $f(x), g(x)$ be two different products of $(x+a)$'s, having degree $< t$. Suppose $f(x) = g(x) \pmod{p, h(x)}$.
  - The test tells us that $f(x^{n^i \cdot p^j}) = g(x^{n^i \cdot p^j}) \pmod{p, h(x)}$.
  - But this means that $f(z) - g(z)$ has atleast t roots in the field $\mathbb{F}_p[x]/(h(x))$, which is a contradiction.

AKS Test: The Proof

- Suppose all the congruences hold and p is a prime factor of n.
- The group $I := \langle n, p \pmod{r} \rangle$. $t := \#I \geq \text{ord}_r(n) \geq 4\log^2 n$.
- The group $J := \langle (x+1), \ldots, (x+\ell) \pmod{p, h(x)} \rangle$ where $h(x)$ is an irreducible factor of $\frac{x^r-1}{x-1}$ modulo p.
  $\#J \geq 2^{\min\{t,\ell\}} > 2^{2\sqrt{t}\log n} \geq n^{2\sqrt{t}}$.
- *Proof:* Let $f(x), g(x)$ be two different products of $(x+a)$'s, having degree $< t$. Suppose $f(x) = g(x) \pmod{p, h(x)}$.
  - The test tells us that $f(x^{n^i \cdot p^j}) = g(x^{n^i \cdot p^j}) \pmod{p, h(x)}$.
  - But this means that $f(z) - g(z)$ has atleast t roots in the field $\mathbb{F}_p[x]/(h(x))$, which is a contradiction.

AKS Test: The Proof

The Two Groups

Group $I := \langle n, p \pmod{r} \rangle$ is of size $t > 4\log^2 n$.
Group $J := \langle (x+1), \ldots, (x+\ell) \pmod{p, h(x)} \rangle$ is of size $> n^{2\sqrt{t}}$.

- There exist tuples $(i,j) \neq (i',j')$ such that $0 \leq i,j,i',j' \leq \sqrt{t}$ and $n^i \cdot p^j \equiv n^{i'} \cdot p^{j'} \pmod{r}$.
- The test tells us that for all $f(x) \in J$, $f(x)^{n^i \cdot p^j} = f(x^{n^i \cdot p^j})$ and $f(x)^{n^{i'} \cdot p^{j'}} = f(x^{n^{i'} \cdot p^{j'}})$.
- Thus, for all $f(x) \in J$, $f(x)^{n^i \cdot p^j} = f(x)^{n^{i'} \cdot p^{j'}}$.
- As J is a cyclic group: $n^i \cdot p^j \equiv n^{i'} \cdot p^{j'} \pmod{\#J}$.
- As $\#J$ is large, $n^i \cdot p^j = n^{i'} \cdot p^{j'}$. Hence, $n = p$ a prime.

AKS Test: The Proof

The Two Groups

Group $I := \langle n, p \pmod{r} \rangle$ is of size $t > 4\log^2 n$.
Group $J := \langle (x+1), \ldots, (x+\ell) \pmod{p, h(x)} \rangle$ is of size $> n^{2\sqrt{t}}$.

- There exist tuples $(i,j) \neq (i',j')$ such that $0 \leq i, j, i', j' \leq \sqrt{t}$ and $n^i \cdot p^j \equiv n^{i'} \cdot p^{j'} \pmod{r}$.
- The test tells us that for all $f(x) \in J$, $f(x)^{n^i \cdot p^j} = f(x^{n^i \cdot p^j})$ and $f(x)^{n^{i'} \cdot p^{j'}} = f(x^{n^{i'} \cdot p^{j'}})$.
- Thus, for all $f(x) \in J$, $f(x)^{n^i \cdot p^j} = f(x)^{n^{i'} \cdot p^{j'}}$.
- As J is a cyclic group: $n^i \cdot p^j \equiv n^{i'} \cdot p^{j'} \pmod{\#J}$.
- As $\#J$ is large, $n^i \cdot p^j = n^{i'} \cdot p^{j'}$. Hence, $n = p$ a prime.

AKS Test: The Proof

The Two Groups

Group $I := \langle n, p \pmod{r} \rangle$ is of size $t > 4\log^2 n$.
Group $J := \langle (x+1), \ldots, (x+\ell) \pmod{p, h(x)} \rangle$ is of size $> n^{2\sqrt{t}}$.

- There exist tuples $(i, j) \neq (i', j')$ such that $0 \leq i, j, i', j' \leq \sqrt{t}$ and $n^i \cdot p^j \equiv n^{i'} \cdot p^{j'} \pmod{r}$.
- The test tells us that for all $f(x) \in J$, $f(x)^{n^i \cdot p^j} = f(x^{n^i \cdot p^j})$ and $f(x)^{n^{i'} \cdot p^{j'}} = f(x^{n^{i'} \cdot p^{j'}})$.
- Thus, for all $f(x) \in J$, $f(x)^{n^i \cdot p^j} = f(x)^{n^{i'} \cdot p^{j'}}$.
- As J is a cyclic group: $n^i \cdot p^j \equiv n^{i'} \cdot p^{j'} \pmod{\#J}$.
- As $\#J$ is large, $n^i \cdot p^j = n^{i'} \cdot p^{j'}$. Hence, $n = p$ a prime.

AKS Test: The Proof

The Two Groups

Group $I := \langle n, p \pmod{r} \rangle$ is of size $t > 4\log^2 n$.
Group $J := \langle (x+1), \ldots, (x+\ell) \pmod{p, h(x)} \rangle$ is of size $> n^{2\sqrt{t}}$.

- There exist tuples $(i,j) \neq (i',j')$ such that $0 \leq i,j,i',j' \leq \sqrt{t}$ and $n^i \cdot p^j \equiv n^{i'} \cdot p^{j'} \pmod{r}$.
- The test tells us that for all $f(x) \in J$, $f(x)^{n^i \cdot p^j} = f(x^{n^i \cdot p^j})$ and $f(x)^{n^{i'} \cdot p^{j'}} = f(x^{n^{i'} \cdot p^{j'}})$.
- Thus, for all $f(x) \in J$, $f(x)^{n^i \cdot p^j} = f(x)^{n^{i'} \cdot p^{j'}}$.
- As J is a cyclic group: $n^i \cdot p^j \equiv n^{i'} \cdot p^{j'} \pmod{\#J}$.
- As $\#J$ is large, $n^i \cdot p^j = n^{i'} \cdot p^{j'}$. Hence, $n = p$ a prime.

AKS Test: The Proof

The Two Groups

Group $I := \langle n, p \pmod{r} \rangle$ is of size $t > 4\log^2 n$.
Group $J := \langle (x+1), \ldots, (x+\ell) \pmod{p, h(x)} \rangle$ is of size $> n^{2\sqrt{t}}$.

- There exist tuples $(i, j) \neq (i', j')$ such that $0 \leq i, j, i', j' \leq \sqrt{t}$ and $n^i \cdot p^j \equiv n^{i'} \cdot p^{j'} \pmod{r}$.
- The test tells us that for all $f(x) \in J$, $f(x)^{n^i \cdot p^j} = f(x^{n^i \cdot p^j})$ and $f(x)^{n^{i'} \cdot p^{j'}} = f(x^{n^{i'} \cdot p^{j'}})$.
- Thus, for all $f(x) \in J$, $f(x)^{n^i \cdot p^j} = f(x)^{n^{i'} \cdot p^{j'}}$.
- As J is a cyclic group: $n^i \cdot p^j \equiv n^{i'} \cdot p^{j'} \pmod{\#J}$.
- As $\#J$ is large, $n^i \cdot p^j = n^{i'} \cdot p^{j'}$. Hence, $n = p$ a prime.

AKS Test: The Proof

The Two Groups

Group $I := \langle n, p \pmod{r} \rangle$ is of size $t > 4\log^2 n$.
Group $J := \langle (x+1), \ldots, (x+\ell) \pmod{p, h(x)} \rangle$ is of size $> n^{2\sqrt{t}}$.

- There exist tuples $(i,j) \neq (i',j')$ such that $0 \leq i, j, i', j' \leq \sqrt{t}$ and $n^i \cdot p^j \equiv n^{i'} \cdot p^{j'} \pmod{r}$.
- The test tells us that for all $f(x) \in J$, $f(x)^{n^i \cdot p^j} = f(x^{n^i \cdot p^j})$ and $f(x)^{n^{i'} \cdot p^{j'}} = f(x^{n^{i'} \cdot p^{j'}})$.
- Thus, for all $f(x) \in J$, $f(x)^{n^i \cdot p^j} = f(x)^{n^{i'} \cdot p^{j'}}$.
- As J is a cyclic group: $n^i \cdot p^j \equiv n^{i'} \cdot p^{j'} \pmod{\#J}$.
- As $\#J$ is large, $n^i \cdot p^j = n^{i'} \cdot p^{j'}$. Hence, $n = p$ a prime.

AKS Test: The Proof

The Two Groups

Group $I := \langle n, p \pmod{r} \rangle$ is of size $t > 4 \log^2 n$.

Group $J := \langle (x+1), \ldots, (x+\ell) \pmod{p, h(x)} \rangle$ is of size $> n^{2\sqrt{t}}$.

- There exist tuples $(i, j) \neq (i', j')$ such that $0 \leq i, j, i', j' \leq \sqrt{t}$ and $n^i \cdot p^j \equiv n^{i'} \cdot p^{j'} \pmod{r}$.
- The test tells us that for all $f(x) \in J$, $f(x)^{n^i \cdot p^j} = f(x^{n^i \cdot p^j})$ and $f(x)^{n^{i'} \cdot p^{j'}} = f(x^{n^{i'} \cdot p^{j'}})$.
- Thus, for all $f(x) \in J$, $f(x)^{n^i \cdot p^j} = f(x)^{n^{i'} \cdot p^{j'}}$.
- As J is a cyclic group: $n^i \cdot p^j \equiv n^{i'} \cdot p^{j'} \pmod{\#J}$.
- As $\#J$ is large, $n^i \cdot p^j = n^{i'} \cdot p^{j'}$. Hence, $n = p$ a prime.

AKS Test: Time Complexity

- Recall that r is the least number such that $\text{ord}_r(n) > 4\log^2 n$.
- Prime number theorem gives $r = O(\log^5 n)$ and the algorithm takes time $O^{\sim}(\log^{10.5} n)$.
- Lenstra and Pomerance (2003) further reduced the time complexity to $O^{\sim}(\log^6 n)$.

AKS Test: Time Complexity

- Recall that r is the least number such that $\mathrm{ord}_r(n) > 4\log^2 n$.
- Prime number theorem gives $r = O(\log^5 n)$ and the algorithm takes time $O^{\sim}(\log^{10.5} n)$.
- Lenstra and Pomerance (2003) further reduced the time complexity to $O^{\sim}(\log^6 n)$.

AKS Test: Time Complexity

- Recall that r is the least number such that $\mathrm{ord}_r(n) > 4\log^2 n$.
- Prime number theorem gives $r = O(\log^5 n)$ and the algorithm takes time $O^\sim(\log^{10.5} n)$.
- Lenstra and Pomerance (2003) further reduced the time complexity to $O^\sim(\log^6 n)$.

Outline

Questions

- The AKS primality test solves a long-standing open question but cannot compete with the randomized tests used in practice.
- However, several modifications have been suggested to AKS test that are faster than the original proposal.
- Can we reduce the number of a for which the test is performed? Here is a conjecture that can bring down the complexity to $O^{\sim}(\log^3 n)$:

Conjecture: (Bhattacharjee-Pandey 2001; AKS 2004)

Let $r > \log n$ be a prime number that does not divide $(n^3 - n)$. Then $(x-1)^n \equiv (x^n - 1) \pmod{n, x^r - 1}$ iff n is prime.

Questions

- The AKS primality test solves a long-standing open question but cannot compete with the randomized tests used in practice.
- However, several modifications have been suggested to AKS test that are faster than the original proposal.
- Can we reduce the number of a for which the test is performed? Here is a conjecture that can bring down the complexity to $O^{\sim}(\log^3 n)$:

Conjecture: (Bhattacharjee-Pandey 2001; AKS 2004)

Let $r > \log n$ be a prime number that does not divide $(n^3 - n)$. Then $(x-1)^n \equiv (x^n - 1) \pmod{n, x^r - 1}$ iff n is prime.

Questions

- The AKS primality test solves a long-standing open question but cannot compete with the randomized tests used in practice.
- However, several modifications have been suggested to AKS test that are faster than the original proposal.
- Can we reduce the number of a for which the test is performed? Here is a conjecture that can bring down the complexity to $O^{\sim}(\log^3 n)$:

Conjecture: (Bhattacharjee-Pandey 2001; AKS 2004)

Let $r > \log n$ be a prime number that does not divide $(n^3 - n)$. Then $(x-1)^n \equiv (x^n - 1) \pmod{n, x^r - 1}$ iff n is prime.

Questions

- The AKS primality test solves a long-standing open question but cannot compete with the randomized tests used in practice.
- However, several modifications have been suggested to AKS test that are faster than the original proposal.
- Can we reduce the number of a for which the test is performed? Here is a conjecture that can bring down the complexity to $O^{\sim}(\log^3 n)$:

Conjecture: (Bhattacharjee-Pandey 2001; AKS 2004)

Let $r > \log n$ be a prime number that does not divide $(n^3 - n)$. Then $(x-1)^n \equiv (x^n - 1) \pmod{n, x^r - 1}$ iff n is prime.

Questions

- An even more interesting question is that of Polynomial Identity Testing (PIT).
- Given a circuit $C(x_1, \ldots, x_n)$, determine whether it is the zero circuit in time polynomial in the size of C ??
- Note that AKS primality test solved this question for the special circuit $C(x) = (x+1)^n - (x^n+1) \pmod{n}$.
- There has been some progress but the big question of PIT is very much open.
- It has also been shown that PIT is related to the "holy-grail" of complexity theory: proving lower bounds.

Questions

- An even more interesting question is that of Polynomial Identity Testing (PIT).
- Given a circuit $C(x_1, \ldots, x_n)$, determine whether it is the zero circuit in time polynomial in the size of C ??
- Note that AKS primality test solved this question for the special circuit $C(x) = (x+1)^n - (x^n + 1) \pmod{n}$.
- There has been some progress but the big question of PIT is very much open.
- It has also been shown that PIT is related to the "holy-grail" of complexity theory: proving lower bounds.

Questions

- An even more interesting question is that of Polynomial Identity Testing (PIT).
- Given a circuit $C(x_1, \ldots, x_n)$, determine whether it is the zero circuit in time polynomial in the size of C ??
- Note that AKS primality test solved this question for the special circuit $C(x) = (x+1)^n - (x^n+1) \pmod{n}$.
- There has been some progress but the big question of PIT is very much open.
- It has also been shown that PIT is related to the "holy-grail" of complexity theory: proving lower bounds.

Questions

- An even more interesting question is that of Polynomial Identity Testing (PIT).
- Given a circuit $C(x_1, \ldots, x_n)$, determine whether it is the zero circuit in time polynomial in the size of C ??
- Note that AKS primality test solved this question for the special circuit $C(x) = (x+1)^n - (x^n+1) \pmod{n}$.
- There has been some progress but the big question of PIT is very much open.
- It has also been shown that PIT is related to the "holy-grail" of complexity theory: proving lower bounds.

Questions

- An even more interesting question is that of Polynomial Identity Testing (PIT).
- Given a circuit $C(x_1, \ldots, x_n)$, determine whether it is the zero circuit in time polynomial in the size of C ??
- Note that AKS primality test solved this question for the special circuit $C(x) = (x+1)^n - (x^n+1) \pmod{n}$.
- There has been some progress but the big question of PIT is very much open.
- It has also been shown that PIT is related to the "holy-grail" of complexity theory: proving lower bounds.

Questions

- An even more interesting question is that of Polynomial Identity Testing (PIT).
- Given a circuit $C(x_1, \ldots, x_n)$, determine whether it is the zero circuit in time polynomial in the size of C ??
- Note that AKS primality test solved this question for the special circuit $C(x) = (x+1)^n - (x^n+1) \pmod{n}$.
- There has been some progress but the big question of PIT is very much open.
- It has also been shown that PIT is related to the “holy-grail” of complexity theory: proving lower bounds.