SPCS Cryptography Class Lecture 7

June 30, 2015

Primality Testing

Goal: Given a (large) number n, want to test whether n is a prime efficiently.

Primality Testing

Goal: Given a (large) number n, want to test whether n is a prime efficiently. Naive way:

Primality Testing

Goal: Given a (large) number n, want to test whether n is a prime efficiently. Naive way:

- Test all the numbers 2, 3, 4, all the way up to $n - 1$, to see if any of them is a divisor of n.

Primality Testing

Goal: Given a (large) number n, want to test whether n is a prime efficiently. Naive way:

- Test all the numbers 2, 3, 4, all the way up to $n-1$, to see if any of them is a divisor of n.
- This takes $O(n)$ divisions.

Primality Testing

Goal: Given a (large) number n, want to test whether n is a prime efficiently. Naive way:

- Test all the numbers 2, 3, 4, all the way up to $n - 1$, to see if any of them is a divisor of n.
- This takes $O(n)$ divisions.

Less naive way:

Primality Testing

Goal: Given a (large) number n, want to test whether n is a prime efficiently. Naive way:

- Test all the numbers 2, 3, 4, all the way up to $n-1$, to see if any of them is a divisor of n.
- This takes $O(n)$ divisions.

Less naive way:

- Test all the numbers 2, 3, 4, all the way up to $[\sqrt{n}]$, to see if any of them is a divisor of n.

Primality Testing

Goal: Given a (large) number n, want to test whether n is a prime efficiently. Naive way:

- Test all the numbers 2, 3, 4, all the way up to $n - 1$, to see if any of them is a divisor of n.
- This takes $O(n)$ divisions.

Less naive way:

- Test all the numbers 2, 3, 4, all the way up to $[\sqrt{n}]$, to see if any of them is a divisor of n.
- This takes $O(\sqrt{n})$ divisions.

Primality Testing

Goal: Given a (large) number n, want to test whether n is a prime efficiently. Naive way:

- Test all the numbers 2, 3, 4, all the way up to $n-1$, to see if any of them is a divisor of n.
- This takes $O(n)$ divisions.

Less naive way:

- Test all the numbers 2, 3, 4, all the way up to $[\sqrt{n}]$, to see if any of them is a divisor of n.
- This takes $O(\sqrt{n})$ divisions.

Is this good?

Primality Testing

Goal: Given a (large) number n, want to test whether n is a prime efficiently. Naive way:

- Test all the numbers 2, 3, 4, all the way up to $n-1$, to see if any of them is a divisor of n.
- This takes $O(n)$ divisions.

Less naive way:

- Test all the numbers 2, 3, 4, all the way up to $[\sqrt{n}]$, to see if any of them is a divisor of n.
- This takes $O(\sqrt{n})$ divisions.

Is this good? No, because for a number n, in base 2 there are $\log_2 n$ digits, so these are all exponential time algorithms.

Fermat's test

- The first test we will discuss is based on Fermat's little theorem, hence called the Fermat test.

Fermat's test

- The first test we will discuss is based on Fermat's little theorem, hence called the Fermat test.
- Recall Fermat's little theorem

Theorem (Fermat's little theorem)

Let p be a prime. Then for all integers a relatively prime to p,

$$a^{p-1} \equiv 1 \mod p$$

- The *contrapositive* of the above statement is

Fermat's test

- The first test we will discuss is based on Fermat's little theorem, hence called the Fermat test.
- Recall Fermat's little theorem

Theorem (Fermat's little theorem)

Let p be a prime. Then for all integers a relatively prime to p,

$$a^{p-1} \equiv 1 \mod p$$

- The *contrapositive* of the above statement is

Theorem

Given a positive integer n. If you can find x such that $\gcd(x, n) = 1$, and $x^{n-1} \not\equiv 1 \mod n$, then n is NOT a prime.

Fermat's test

Example

What is $2^{14} \mod 15$?

Fermat's test

Example

What is $2^{14} \mod 15$? Is 15 a prime?

Fermat's test

Example

What is $2^{14} \mod 15$? Is 15 a prime?

Example

What is $2^{16} \mod 17$?

Fermat's test

Example

What is $2^{14} \mod 15$? Is 15 a prime?

Example

What is $2^{16} \mod 17$? Does it imply that 17 is a prime?

Fermat's test

Example

What is $2^{14} \mod 15$? Is 15 a prime?

Example

What is $2^{16} \mod 17$? Does it imply that 17 is a prime?

Example

What is $2^{1386} \mod 1387$?

Fermat's test

Example

What is $2^{14} \mod 15$? Is 15 a prime?

Example

What is $2^{16} \mod 17$? Does it imply that 17 is a prime?

Example

What is $2^{1386} \mod 1387$? Let's get some help.. `https://cloud.sagemath.com/projects/113f54cf-25b5-4bf4-803f-c9bdef8b6eec/files/fermat_prime.sagews`

Fermat's test

Example

What is $2^{14} \mod 15$? Is 15 a prime?

Example

What is $2^{16} \mod 17$? Does it imply that 17 is a prime?

Example

What is $2^{1386} \mod 1387$? Let's get some help.. `https://cloud.sagemath.com/projects/113f54cf-25b5-4bf4-803f-c9bdef8b6eec/files/fermat_prime.sagews` Does it imply anything interesting about 1387?

Fermat's test

Example

What is $2^{14} \mod 15$? Is 15 a prime?

Example

What is $2^{16} \mod 17$? Does it imply that 17 is a prime?

Example

What is $2^{1386} \mod 1387$? Let's get some help.. `https://cloud.sagemath.com/projects/113f54cf-25b5-4bf4-803f-c9bdef8b6eec/files/fermat_prime.sagews` Does it imply anything interesting about 1387? What about $3^{1386} \mod 1387$?

Fermat's test

Fermat test

Fermat's test

Fermat test

To test whether n is a prime, choose a few random numbers $x \neq 0 \mod n$ and see if $x^{n-1} \equiv 1 \mod n$.

Fermat's test

Fermat test

To test whether n is a prime, choose a few random numbers $x \neq 0 \mod n$ and see if $x^{n-1} \equiv 1 \mod n$.

- If some of the x satisfy $(x, n) > 1$, congratulations! You found a factor of n.

Fermat's test

Fermat test

To test whether n is a prime, choose a few random numbers $x \neq 0 \bmod n$ and see if $x^{n-1} \equiv 1 \bmod n$.

- If some of the x satisfy $(x, n) > 1$, congratulations! You found a factor of n.
- If for some x, $x^{n-1} \not\equiv 1 \bmod n$, we know that n is composite, and that x is a *Fermat witness*.

Fermat's test

Fermat test

To test whether n is a prime, choose a few random numbers $x \neq 0 \mod n$ and see if $x^{n-1} \equiv 1 \mod n$.

- If some of the x satisfy $(x, n) > 1$, congratulations! You found a factor of n.
- If for some x, $x^{n-1} \not\equiv 1 \mod n$, we know that n is composite, and that x is a *Fermat witness*.
- If for all x we tested, $x^{n-1} \equiv 1 \mod n$, we declare that n passes the Fermat test and is probably a prime.

Fermat's test

Fermat test

To test whether n is a prime, choose a few random numbers $x \neq 0 \mod n$ and see if $x^{n-1} \equiv 1 \mod n$.

- If some of the x satisfy $(x, n) > 1$, congratulations! You found a factor of n.
- If for some x, $x^{n-1} \not\equiv 1 \mod n$, we know that n is composite, and that x is a *Fermat witness*.
- If for all x we tested, $x^{n-1} \equiv 1 \mod n$, we declare that n passes the Fermat test and is probably a prime.

- Why just a few random x? Why not all $x < n$?

Fermat's test

Some issues:

- Correctness:

Fermat's test

Some issues:

- Correctness:
  - if n passes the Fermat test for all $x \bmod n$ - is it a prime?

Fermat's test

Some issues:

- Correctness:
  - if n passes the Fermat test for all $x \bmod n$ - is it a prime? Yes.

Fermat's test

Some issues:

- Correctness:
  - if n passes the Fermat test for all $x \mod n$ - is it a prime? Yes.
  - If n passes the Fermat test for all $x \mod n$ such that $gcd(x, n) = 1$ - is it a prime?

Fermat's test

Some issues:

- Correctness:
  - if n passes the Fermat test for all $x \mod n$ - is it a prime? Yes.
  - If n passes the Fermat test for all $x \mod n$ such that $gcd(x, n) = 1$ - is it a prime?

  No! There are Carmichael numbers.

Fermat's test

Some issues:

- Correctness:
  - if n passes the Fermat test for all $x \mod n$ - is it a prime? Yes.
  - If n passes the Fermat test for all $x \mod n$ such that $gcd(x, n) = 1$ - is it a prime?

  No! There are Carmichael numbers.
  - They are precisely the numbers that pass the Fermat test for all $gcd(x, n) = 1$, yet is not a prime.

Fermat's test

Some issues:

- Correctness:
  - if n passes the Fermat test for all $x \mod n$ - is it a prime? Yes.
  - If n passes the Fermat test for all $x \mod n$ such that $gcd(x, n) = 1$ - is it a prime?

  No! There are Carmichael numbers.
  - They are precisely the numbers that pass the Fermat test for all $gcd(x, n) = 1$, yet is not a prime.
  - The smallest Carmichael number is 561.

Fermat's test

Some issues:

- Correctness:
  - if n passes the Fermat test for all $x \mod n$ - is it a prime? Yes.
  - If n passes the Fermat test for all $x \mod n$ such that $gcd(x, n) = 1$ - is it a prime?

No! There are Carmichael numbers.

  - They are precisely the numbers that pass the Fermat test for all $gcd(x, n) = 1$, yet is not a prime.
  - The smallest Carmichael number is 561.
  - There are infinitely many Carmichael numbers. (Alford, Granville, Pomerance 1994)

Fermat's test

Running time. Let's look at one random x first.

Fermat's test

Running time. Let's look at one random x first.

- Checking $gcd(x, n)$.

Fermat's test

Running time. Let's look at one random x first.

- Checking $gcd(x, n)$. The naive Euclidean algorithm would give $O(\log_2 x + \log_2 n) = O(\log n)$ runtime.

Fermat's test

Running time. Let's look at one random x first.

- Checking $gcd(x, n)$. The naive Euclidean algorithm would give $O(\log_2 x + \log_2 n) = O(\log n)$ runtime.
- Computing $x^{n-1} \mod n$.

Fermat's test

Running time. Let's look at one random x first.

- Checking $gcd(x, n)$. The naive Euclidean algorithm would give $O(\log_2 x + \log_2 n) = O(\log n)$ runtime.
- Computing $x^{n-1} \mod n$. Fast exponentiation takes $O(\log n)$ steps - where one step = multiply mod n.

Fermat's test

Running time. Let's look at one random x first.

- Checking $gcd(x, n)$. The naive Euclidean algorithm would give $O(\log_2 x + \log_2 n) = O(\log n)$ runtime.
- Computing $x^{n-1} \mod n$. Fast exponentiation takes $O(\log n)$ steps - where one step = multiply mod n.
- Note that in this case we CANNOT ignore the modulus n, since it is also getting larger.

Fermat's test

Running time. Let's look at one random x first.

- Checking $gcd(x, n)$. The naive Euclidean algorithm would give $O(\log_2 x + \log_2 n) = O(\log n)$ runtime.
- Computing $x^{n-1} \mod n$. Fast exponentiation takes $O(\log n)$ steps - where one step = multiply mod n.
- Note that in this case we CANNOT ignore the modulus n, since it is also getting larger.
    - $x \mod n$ has $O(\log n)$ digits.

Fermat's test

Running time. Let's look at one random x first.

- Checking $gcd(x, n)$. The naive Euclidean algorithm would give $O(\log_2 x + \log_2 n) = O(\log n)$ runtime.
- Computing $x^{n-1} \mod n$. Fast exponentiation takes $O(\log n)$ steps - where one step = multiply mod n.
- Note that in this case we CANNOT ignore the modulus n, since it is also getting larger.
    - $x \mod n$ has $O(\log n)$ digits.
    - We are multiplying two numbers of this size then mod n.

Fermat's test

Running time. Let's look at one random x first.

- Checking $gcd(x, n)$. The naive Euclidean algorithm would give $O(\log_2 x + \log_2 n) = O(\log n)$ runtime.
- Computing $x^{n-1} \mod n$. Fast exponentiation takes $O(\log n)$ steps - where one step = multiply mod n.
- Note that in this case we CANNOT ignore the modulus n, since it is also getting larger.
  - $x \mod n$ has $O(\log n)$ digits.
  - We are multiplying two numbers of this size then mod n.
  - Naive algorithm gives $O((\log n)^2)$ time for each step in fast exponentiation.

Fermat's test

Running time. Let's look at one random x first.

- Checking $gcd(x, n)$. The naive Euclidean algorithm would give $O(\log_2 x + \log_2 n) = O(\log n)$ runtime.
- Computing $x^{n-1} \mod n$. Fast exponentiation takes $O(\log n)$ steps - where one step = multiply $\mod n$.
- Note that in this case we CANNOT ignore the modulus n, since it is also getting larger.
  - $x \mod n$ has $O(\log n)$ digits.
  - We are multiplying two numbers of this size then $\mod n$.
  - Naive algorithm gives $O((\log n)^2)$ time for each step in fast exponentiation.
  - Thus fast exponentiation takes $O((\log n)^3)$ time.

Fermat's test

Running time. Let's look at one random x first.

- Checking $gcd(x, n)$. The naive Euclidean algorithm would give $O(\log_2 x + \log_2 n) = O(\log n)$ runtime.
- Computing $x^{n-1} \mod n$. Fast exponentiation takes $O(\log n)$ steps - where one step = multiply mod n.
- Note that in this case we CANNOT ignore the modulus n, since it is also getting larger.
  - $x \mod n$ has $O(\log n)$ digits.
  - We are multiplying two numbers of this size then mod n.
  - Naive algorithm gives $O((\log n)^2)$ time for each step in fast exponentiation.
  - Thus fast exponentiation takes $O((\log n)^3)$ time.
- We then check if $x^{n-1} \equiv 1 \mod n$ or not.

Fermat's test

Running time. Let's look at one random x first.

- Checking $gcd(x, n)$. The naive Euclidean algorithm would give $O(\log_2 x + \log_2 n) = O(\log n)$ runtime.
- Computing $x^{n-1} \mod n$. Fast exponentiation takes $O(\log n)$ steps - where one step = multiply mod n.
- Note that in this case we CANNOT ignore the modulus n, since it is also getting larger.
  - $x \mod n$ has $O(\log n)$ digits.
  - We are multiplying two numbers of this size then mod n.
  - Naive algorithm gives $O((\log n)^2)$ time for each step in fast exponentiation.
  - Thus fast exponentiation takes $O((\log n)^3)$ time.
- We then check if $x^{n-1} \equiv 1 \mod n$ or not. This takes $O(1)$ time.
- Thus the total running time for ONE random x is

$$O(\log n) + O((\log n)^3) + O(1) = O((\log n)^3)$$

- So if we test k random x, the run time is $O(k(\log n)^3)$.

Fermat's test

Running time. Let's look at one random x first.

- Checking $gcd(x, n)$. The naive Euclidean algorithm would give $O(\log_2 x + \log_2 n) = O(\log n)$ runtime.
- Computing $x^{n-1} \mod n$. Fast exponentiation takes $O(\log n)$ steps - where one step = multiply mod n.
- Note that in this case we CANNOT ignore the modulus n, since it is also getting larger.
  - $x \mod n$ has $O(\log n)$ digits.
  - We are multiplying two numbers of this size then mod n.
  - Naive algorithm gives $O((\log n)^2)$ time for each step in fast exponentiation.
  - Thus fast exponentiation takes $O((\log n)^3)$ time.
- We then check if $x^{n-1} \equiv 1 \mod n$ or not. This takes $O(1)$ time.
- Thus the total running time for ONE random x is

$$O(\log n) + O((\log n)^3) + O(1) = O((\log n)^3)$$

- So if we test k random x, the run time is $O(k(\log n)^3)$.

Is this polynomial time?

Fermat's test

Example (Fermat primes)

We once mentioned that $65537 = 2^{16} + 1$ is a prime. In general, numbers of the form $2^{2^n} + 1$ are called Fermat numbers, and those that are also primes are called Fermat prime.

Fermat's test

Example (Fermat primes)

We once mentioned that $65537 = 2^{16} + 1$ is a prime. In general, numbers of the form $2^{2^n} + 1$ are called Fermat numbers, and those that are also primes are called Fermat prime.

- 65537 is the largest known Fermat prime.

Fermat's test

Example (Fermat primes)

We once mentioned that $65537 = 2^{16} + 1$ is a prime. In general, numbers of the form $2^{2^n} + 1$ are called Fermat numbers, and those that are also primes are called Fermat prime.

- 65537 is the largest known Fermat prime.
- Let's look at $2^{2^5} + 1 = 4294967297$. You can calculate this in Sage.

Fermat's test

Example (Fermat primes)

We once mentioned that $65537 = 2^{16} + 1$ is a prime. In general, numbers of the form $2^{2^n} + 1$ are called Fermat numbers, and those that are also primes are called Fermat prime.

- 65537 is the largest known Fermat prime.
- Let's look at $2^{2^5} + 1 = 4294967297$. You can calculate this in Sage. It passes the Fermat test for 2 but fails for 3.

Fermat's test

Example (Fermat primes)

We once mentioned that $65537 = 2^{16} + 1$ is a prime. In general, numbers of the form $2^{2^n} + 1$ are called Fermat numbers, and those that are also primes are called Fermat prime.

- 65537 is the largest known Fermat prime.
- Let's look at $2^{2^5} + 1 = 4294967297$. You can calculate this in Sage. It passes the Fermat test for 2 but fails for 3.

Does it tell you how $2^{2^5} + 1$ factors?

Fermat's test

Example

A larger number is $2^{2^{14}} + 1$. This number has around 5000 digits.

Fermat's test

Example

A larger number is $2^{2^{14}}+1$. This number has around 5000 digits. It passes the Fermat test for 2 but fails for 3.

Fermat's test

Example

A larger number is $2^{2^{14}}+1$. This number has around 5000 digits. It passes the Fermat test for 2 but fails for 3. Compositeness of this number is known in 1961. But the first non-trivial factor of it is only found in 2010.

Fermat's test

Example

A larger number is $2^{2^{14}} + 1$. This number has around 5000 digits. It passes the Fermat test for 2 but fails for 3. Compositeness of this number is known in 1961. But the first non-trivial factor of it is only found in 2010. It is a 53-digit number,

$$116928085873074369829035993834596371340386703423373313$$

Fermat's test

One final remark:

Fermat's test

One final remark:

- There are Carmichael numbers that mess things up - what about the other numbers?

Fermat's test

One final remark:

- There are Carmichael numbers that mess things up - what about the other numbers?
- How many random x is enough to tell the compositeness of n?

Fermat's test

One final remark:

- There are Carmichael numbers that mess things up - what about the other numbers?
- How many random x is enough to tell the compositeness of n?

Theorem

Let $n \geq 2$ be a composite number which is not Carmichael. Then the number of Fermat witnesses is greater than $\frac{n-1}{2}$.

Fermat's test

One final remark:

- There are Carmichael numbers that mess things up - what about the other numbers?
- How many random x is enough to tell the compositeness of n?

Theorem

Let $n \geq 2$ be a composite number which is not Carmichael. Then the number of Fermat witnesses is greater than $\frac{n-1}{2}$.

Summary:

- Fermat test is a primality test based on Fermat's little theorem.

Fermat's test

One final remark:

- There are Carmichael numbers that mess things up - what about the other numbers?
- How many random x is enough to tell the compositeness of n?

Theorem

Let $n \geq 2$ be a composite number which is not Carmichael. Then the number of Fermat witnesses is greater than $\frac{n-1}{2}$.

Summary:

- Fermat test is a primality test based on Fermat's little theorem.
- It is probabilistic - a number that passes the Fermat's test is probably prime, but still may not be prime.

Miller-Rabin test

Question

Can we tweak Fermat test to make it deterministic? (i.e. if it passes the test, it REALLY is a prime.)

Miller-Rabin test

Question

Can we tweak Fermat test to make it deterministic? (i.e. if it passes the test, it REALLY is a prime.)

Yes! The main problem with Fermat's little theorem is, the converse is false. We can try to refine that.

Miller-Rabin test

Question

Can we tweak Fermat test to make it deterministic? (i.e. if it passes the test, it REALLY is a prime.)

Yes! The main problem with Fermat's little theorem is, the converse is false. We can try to refine that. This is based on the observation,

$$x^2 \equiv 1 \mod p \Leftrightarrow x \equiv \pm 1 \mod p$$

Miller-Rabin test

Question

Can we tweak Fermat test to make it deterministic? (i.e. if it passes the test, it REALLY is a prime.)

Yes! The main problem with Fermat's little theorem is, the converse is false. We can try to refine that. This is based on the observation,

$$x^2 \equiv 1 \mod p \Leftrightarrow x \equiv \pm 1 \mod p$$

One can push it further, for example,

$$\begin{aligned} x^4 \equiv 1 \mod p &\Leftrightarrow (x^2 - 1)(x^2 + 1) \equiv 0 \mod p \\ &\Leftrightarrow x^2 \equiv 1 \mod p \text{ or } x^2 \equiv -1 \mod p \\ &\Leftrightarrow x \equiv 1 \mod p \text{ or } x \equiv -1 \mod p \text{ or } x^2 \equiv -1 \mod p \end{aligned}$$

Miller-Rabin test

Question

Can we tweak Fermat test to make it deterministic? (i.e. if it passes the test, it REALLY is a prime.)

Yes! The main problem with Fermat's little theorem is, the converse is false. We can try to refine that. This is based on the observation,

$$x^2 \equiv 1 \mod p \Leftrightarrow x \equiv \pm 1 \mod p$$

One can push it further, for example,

$$x^4 \equiv 1 \mod p \Leftrightarrow (x^2 - 1)(x^2 + 1) \equiv 0 \mod p$$

$$\Leftrightarrow x \equiv 1 \mod p \text{ or } x \equiv -1 \mod p \text{ or } x^2 \equiv -1 \mod p$$

Miller-Rabin test

Question

Can we tweak Fermat test to make it deterministic? (i.e. if it passes the test, it REALLY is a prime.)

Yes! The main problem with Fermat's little theorem is, the converse is false. We can try to refine that. This is based on the observation,

$$x^2 \equiv 1 \mod p \Leftrightarrow x \equiv \pm 1 \mod p$$

One can push it further, for example,

$$\begin{aligned} x^4 \equiv 1 \mod p &\Leftrightarrow (x^2 - 1)(x^2 + 1) \equiv 0 \mod p \\ &\Leftrightarrow x^2 \equiv 1 \mod p \text{ or } x^2 \equiv -1 \mod p \\ &\Leftrightarrow x \equiv 1 \mod p \text{ or } x \equiv -1 \mod p \text{ or } x^2 \equiv -1 \mod p \end{aligned}$$

Miller-Rabin test

Or even more,

$$\begin{aligned} x^{12} \equiv 1 \bmod p &\Leftrightarrow (x^3)^4 - 1 \equiv 0 \bmod p \\ &\Leftrightarrow (x^3)^2 \equiv 1 \bmod p \text{ or } (x^3)^2 \equiv -1 \bmod p \\ &\Leftrightarrow x^3 \equiv 1 \bmod p \text{ or } x^3 \equiv -1 \bmod p \text{ or } (x^3)^2 \equiv -1 \bmod p \end{aligned}$$

This generalizes to the following theorem.

Theorem

Given n. Factorize $n - 1 = 2^s t$, where t is an odd number. Then n is a prime if and only if for all $0 < x < n$,

- $x^t \equiv 1 \bmod n$ *or*
- $x^t \equiv -1 \bmod n$ *or*
- $x^{2t} \equiv -1 \bmod n$ *or*
- ...
- $x^{2^{s-1}t} \equiv -1 \bmod n$

Miller-Rabin test

Or even more,

$x^{12} \equiv 1 \mod p \Leftrightarrow (x^3)^4 - 1 \equiv 0 \mod p$

$$\Leftrightarrow x^3 \equiv 1 \mod p \text{ or } x^3 \equiv -1 \mod p \text{ or } (x^3)^2 \equiv -1 \mod p$$

This generalizes to the following theorem.

Theorem

Given n. Factorize $n - 1 = 2^s t$, where t is an odd number. Then n is a prime if and only if for all $0 < x < n$,

- $x^t \equiv 1 \mod n$ *or*
- $x^t \equiv -1 \mod n$ *or*
- $x^{2t} \equiv -1 \mod n$ *or*
- ...
- $x^{2^{s-1}t} \equiv -1 \mod n$

Miller-Rabin test

Or even more,

$$
\begin{aligned}
x^{12} \equiv 1 \bmod p &\Leftrightarrow (x^3)^4 - 1 \equiv 0 \bmod p \\
&\Leftrightarrow (x^3)^2 \equiv 1 \bmod p \text{ or } (x^3)^2 \equiv -1 \bmod p \\
&\Leftrightarrow x^3 \equiv 1 \bmod p \text{ or } x^3 \equiv -1 \bmod p \text{ or } (x^3)^2 \equiv -1 \bmod p
\end{aligned}
$$

Or even more,

$$\begin{aligned} x^{12} \equiv 1 \bmod p &\Leftrightarrow (x^3)^4 - 1 \equiv 0 \bmod p \\ &\Leftrightarrow (x^3)^2 \equiv 1 \bmod p \text{ or } (x^3)^2 \equiv -1 \bmod p \\ &\Leftrightarrow x^3 \equiv 1 \bmod p \text{ or } x^3 \equiv -1 \bmod p \text{ or } (x^3)^2 \equiv -1 \bmod p \end{aligned}$$

This generalizes to the following theorem.

Theorem

Given n. Factorize $n - 1 = 2^s t$, where t is an odd number. Then n is a prime if and only if for all $0 < x < n$,

Miller-Rabin test

Or even more,

$$\begin{aligned} x^{12} \equiv 1 \bmod p &\Leftrightarrow (x^3)^4 - 1 \equiv 0 \bmod p \\ &\Leftrightarrow (x^3)^2 \equiv 1 \bmod p \text{ or } (x^3)^2 \equiv -1 \bmod p \\ &\Leftrightarrow x^3 \equiv 1 \bmod p \text{ or } x^3 \equiv -1 \bmod p \text{ or } (x^3)^2 \equiv -1 \bmod p \end{aligned}$$

This generalizes to the following theorem.

Theorem

Given n. Factorize $n - 1 = 2^s t$, where t is an odd number. Then n is a prime if and only if for all $0 < x < n$,

- $x^t \equiv 1 \bmod n$ *or*

Miller-Rabin test

Or even more,

$$\begin{aligned} x^{12} \equiv 1 \bmod p &\Leftrightarrow (x^3)^4 - 1 \equiv 0 \bmod p \\ &\Leftrightarrow (x^3)^2 \equiv 1 \bmod p \text{ or } (x^3)^2 \equiv -1 \bmod p \\ &\Leftrightarrow x^3 \equiv 1 \bmod p \text{ or } x^3 \equiv -1 \bmod p \text{ or } (x^3)^2 \equiv -1 \bmod p \end{aligned}$$

This generalizes to the following theorem.

Theorem

Given n. Factorize $n - 1 = 2^s t$, where t is an odd number. Then n is a prime if and only if for all $0 < x < n$,

- $x^t \equiv 1 \bmod n$ *or*
- $x^t \equiv -1 \bmod n$ *or*

Miller-Rabin test

Or even more,

$$\begin{aligned} x^{12} \equiv 1 \bmod p &\Leftrightarrow (x^3)^4 - 1 \equiv 0 \bmod p \\ &\Leftrightarrow (x^3)^2 \equiv 1 \bmod p \text{ or } (x^3)^2 \equiv -1 \bmod p \\ &\Leftrightarrow x^3 \equiv 1 \bmod p \text{ or } x^3 \equiv -1 \bmod p \text{ or } (x^3)^2 \equiv -1 \bmod p \end{aligned}$$

This generalizes to the following theorem.

Theorem

Given n. Factorize $n - 1 = 2^s t$, where t is an odd number. Then n is a prime if and only if for all $0 < x < n$,

- $x^t \equiv 1 \bmod n$ *or*
- $x^t \equiv -1 \bmod n$ *or*
- $x^{2t} \equiv -1 \bmod n$ *or*

Or even more,

$$
\begin{aligned}
x^{12} \equiv 1 \bmod p &\Leftrightarrow (x^3)^4 - 1 \equiv 0 \bmod p \\
&\Leftrightarrow (x^3)^2 \equiv 1 \bmod p \text{ or } (x^3)^2 \equiv -1 \bmod p \\
&\Leftrightarrow x^3 \equiv 1 \bmod p \text{ or } x^3 \equiv -1 \bmod p \text{ or } (x^3)^2 \equiv -1 \bmod p
\end{aligned}
$$

This generalizes to the following theorem.

Theorem

Given n. Factorize $n - 1 = 2^s t$, where t is an odd number. Then n is a prime if and only if for all $0 < x < n$,

- $x^t \equiv 1 \bmod n$ *or*
- $x^t \equiv -1 \bmod n$ *or*
- $x^{2t} \equiv -1 \bmod n$ *or*
- ...
- $x^{2^{s-1}t} \equiv -1 \bmod n$

Miller-Rabin test

To test whether n is a prime, choose a few random numbers $x \neq 0 \mod n$.
Let $n - 1 = 2^s t$ where t is an odd number.

Miller-Rabin test

Miller-Rabin test

To test whether n is a prime, choose a few random numbers $x \neq 0 \mod n$.
Let $n - 1 = 2^s t$ where t is an odd number.

- If some of the x satisfy $(x, n) > 1$, congratulations! You found a factor of n.

Miller-Rabin test

To test whether n is a prime, choose a few random numbers $x \neq 0 \mod n$. Let $n - 1 = 2^s t$ where t is an odd number.

- If some of the x satisfy $(x, n) > 1$, congratulations! You found a factor of n.
- Compute $x^t \mod n$. If $x^t \equiv \pm 1 \mod n$, it passes the Miller-Rabin test.

Miller-Rabin test

Miller-Rabin test

To test whether n is a prime, choose a few random numbers $x \neq 0 \mod n$. Let $n - 1 = 2^s t$ where t is an odd number.

- If some of the x satisfy $(x, n) > 1$, congratulations! You found a factor of n.
- Compute $x^t \mod n$. If $x^t \equiv \pm 1 \mod n$, it passes the Miller-Rabin test.
- If not, try $x^{2t} \mod n$. If $x^{2t} \equiv -1 \mod n$, it passes the Miller-Rabin test.

Miller-Rabin test

Miller-Rabin test

To test whether n is a prime, choose a few random numbers $x \neq 0 \mod n$. Let $n - 1 = 2^s t$ where t is an odd number.

- If some of the x satisfy $(x, n) > 1$, congratulations! You found a factor of n.
- Compute $x^t \mod n$. If $x^t \equiv \pm 1 \mod n$, it passes the Miller-Rabin test.
- If not, try $x^{2t} \mod n$. If $x^{2t} \equiv -1 \mod n$, it passes the Miller-Rabin test. If not, repeat for $x^{2^2 t} \mod n$ all the way up to $x^{n-1} \mod n$.

Miller-Rabin test

Miller-Rabin test

To test whether n is a prime, choose a few random numbers $x \neq 0 \mod n$. Let $n - 1 = 2^s t$ where t is an odd number.

- If some of the x satisfy $(x, n) > 1$, congratulations! You found a factor of n.
- Compute $x^t \mod n$. If $x^t \equiv \pm 1 \mod n$, it passes the Miller-Rabin test.
- If not, try $x^{2t} \mod n$. If $x^{2t} \equiv -1 \mod n$, it passes the Miller-Rabin test. If not, repeat for $x^{2^2 t} \mod n$ all the way up to $x^{n-1} \mod n$.
- If for all x we tested, n passes the Miller-Rabin test, we declare that it is probably a prime.

Miller-Rabin test

Miller-Rabin test

To test whether n is a prime, choose a few random numbers $x \neq 0 \mod n$. Let $n - 1 = 2^s t$ where t is an odd number.

- If some of the x satisfy $(x, n) > 1$, congratulations! You found a factor of n.
- Compute $x^t \mod n$. If $x^t \equiv \pm 1 \mod n$, it passes the Miller-Rabin test.
- If not, try $x^{2t} \mod n$. If $x^{2t} \equiv -1 \mod n$, it passes the Miller-Rabin test. If not, repeat for $x^{2^2 t} \mod n$ all the way up to $x^{n-1} \mod n$.
- If for all x we tested, n passes the Miller-Rabin test, we declare that it is probably a prime.
- If for some x it does not pass the Miller-Rabin test, we declare that n is composite, with x a Miller-Rabin witness.

Miller-Rabin test

Example

Let's take $n = 13$. We factorize: $13 - 1 = 2^2 \cdot 3$. Let's take our $x = 2$.

Miller-Rabin test

Example

Let's take $n = 13$. We factorize: $13 - 1 = 2^2 \cdot 3$. Let's take our $x = 2$. We compute

$$2^3 = 8 \not\equiv \pm 1 \mod 13,$$

Miller-Rabin test

Example

Let's take $n = 13$. We factorize: $13 - 1 = 2^2 \cdot 3$. Let's take our $x = 2$. We compute

$$2^3 = 8 \not\equiv \pm 1 \mod 13, \text{ but } 2^6 = 64 \equiv -1 \mod 13$$

So it passes Miller-Rabin for $x = 2$.

Example

Let's take $n = 13$. We factorize: $13 - 1 = 2^2 \cdot 3$. Let's take our $x = 2$. We compute

$$2^3 = 8 \not\equiv \pm 1 \mod 13, \text{ but } 2^6 = 64 \equiv -1 \mod 13$$

So it passes Miller-Rabin for $x = 2$. If we take $x = 3$,

Miller-Rabin test

Example

Let's take $n = 13$. We factorize: $13 - 1 = 2^2 \cdot 3$. Let's take our $x = 2$. We compute

$$2^3 = 8 \not\equiv \pm 1 \mod 13, \text{ but } 2^6 = 64 \equiv -1 \mod 13$$

So it passes Miller-Rabin for $x = 2$. If we take $x = 3$,

$$3^3 = 27 \equiv 1 \mod 13$$

Miller-Rabin test

Example

Let's take $n = 13$. We factorize: $13 - 1 = 2^2 \cdot 3$. Let's take our $x = 2$. We compute

$$2^3 = 8 \not\equiv \pm 1 \mod 13, \text{ but } 2^6 = 64 \equiv -1 \mod 13$$

So it passes Miller-Rabin for $x = 2$. If we take $x = 3$,

$$3^3 = 27 \equiv 1 \mod 13$$

so it also passes Miller-Rabin for $x = 3$. Of course, we know that 13 is a prime.

Miller-Rabin test

Example

What happens to Carmichael numbers? Let's take $n = 561$, the smallest Carmichael number.

Miller-Rabin test

Example

What happens to Carmichael numbers? Let's take $n = 561$, the smallest Carmichael number. https://cloud.sagemath.com/projects/113f54cf-25b5-4bf4-803f-c9bdef8b6eec/files/miller_rabin.sagews

Miller-Rabin test

Example

What happens to Carmichael numbers? Let's take $n = 561$, the smallest Carmichael number. `https://cloud.sagemath.com/projects/113f54cf-25b5-4bf4-803f-c9bdef8b6eec/files/miller_rabin.sagews` Factorize $561 - 1 = 2^4 \cdot 35$.

Miller-Rabin test

Example

What happens to Carmichael numbers? Let's take $n = 561$, the smallest Carmichael number. `https://cloud.sagemath.com/projects/113f54cf-25b5-4bf4-803f-c9bdef8b6eec/files/miller_rabin.sagews` Factorize $561 - 1 = 2^4 \cdot 35$. Let's take our $x = 2$, then we compute

Miller-Rabin test

Example

What happens to Carmichael numbers? Let's take $n = 561$, the smallest Carmichael number. `https://cloud.sagemath.com/projects/113f54cf-25b5-4bf4-803f-c9bdef8b6eec/files/miller_rabin.sagews` Factorize $561 - 1 = 2^4 \cdot 35$. Let's take our $x = 2$, then we compute

$$2^{35} \equiv 263 \not\equiv \pm 1 \mod 561$$

Example

What happens to Carmichael numbers? Let's take $n = 561$, the smallest Carmichael number. `https://cloud.sagemath.com/projects/113f54cf-25b5-4bf4-803f-c9bdef8b6eec/files/miller_rabin.sagews` Factorize $561 - 1 = 2^4 \cdot 35$. Let's take our $x = 2$, then we compute

$$2^{35} \equiv 263 \not\equiv \pm 1 \mod 561$$

$$2^{2\cdot 35} \equiv 263^2 \equiv 166 \not\equiv -1 \mod 561$$

Miller-Rabin test

Example

What happens to Carmichael numbers? Let's take $n = 561$, the smallest Carmichael number. `https://cloud.sagemath.com/projects/113f54cf-25b5-4bf4-803f-c9bdef8b6eec/files/miller_rabin.sagews` Factorize $561 - 1 = 2^4 \cdot 35$. Let's take our $x = 2$, then we compute

$$2^{35} \equiv 263 \not\equiv \pm 1 \mod 561$$

$$2^{2 \cdot 35} \equiv 263^2 \equiv 166 \not\equiv -1 \mod 561$$

$$2^{2^2 \cdot 35} \equiv 166^2 \equiv 67 \not\equiv -1 \mod 561$$

Miller-Rabin test

Example

What happens to Carmichael numbers? Let's take $n = 561$, the smallest Carmichael number. `https://cloud.sagemath.com/projects/113f54cf-25b5-4bf4-803f-c9bdef8b6eec/files/miller_rabin.sagews` Factorize $561 - 1 = 2^4 \cdot 35$. Let's take our $x = 2$, then we compute

$$2^{35} \equiv 263 \not\equiv \pm 1 \mod 561$$

$$2^{2 \cdot 35} \equiv 263^2 \equiv 166 \not\equiv -1 \mod 561$$

$$2^{2^2 \cdot 35} \equiv 166^2 \equiv 67 \not\equiv -1 \mod 561$$

$$2^{2^3 \cdot 35} \equiv 67^2 \equiv 1 \not\equiv -1 \mod 561$$

Miller-Rabin test

Example

What happens to Carmichael numbers? Let's take $n = 561$, the smallest Carmichael number. `https://cloud.sagemath.com/projects/113f54cf-25b5-4bf4-803f-c9bdef8b6eec/files/miller_rabin.sagews` Factorize $561 - 1 = 2^4 \cdot 35$. Let's take our $x = 2$, then we compute

$$2^{35} \equiv 263 \not\equiv \pm 1 \mod 561$$

$$2^{2 \cdot 35} \equiv 263^2 \equiv 166 \not\equiv -1 \mod 561$$

$$2^{2^2 \cdot 35} \equiv 166^2 \equiv 67 \not\equiv -1 \mod 561$$

$$2^{2^3 \cdot 35} \equiv 67^2 \equiv 1 \not\equiv -1 \mod 561$$

So 561 does not pass Miller-Rabin, and so it's not a prime!

Miller-Rabin test

Some issues:

Question

Are there numbers that would pass Miller-Rabin for all $(x, n) = 1$*, yet is not a prime?*

Miller-Rabin test

Some issues:

Question

Are there numbers that would pass Miller-Rabin for all $(x, n) = 1$, *yet is not a prime?*

- The answer is NO this time, because of the theorem before Miller-Rabin.

Miller-Rabin test

Some issues:

Question

Are there numbers that would pass Miller-Rabin for all $(x, n) = 1$, yet is not a prime?

- The answer is NO this time, because of the theorem before Miller-Rabin.
- As stated Miller-Rabin is still *probabilistic*, because it depends on whether you hit an x that happens to be a witness. This raises the question,

Miller-Rabin test

Miller-Rabin test

Question

How many random x do we need to guarantee detection of a composite number n?

Miller-Rabin test

Question

How many random x do we need to guarantee detection of a composite number n?

Theorem

Let n be an odd composite number, then at least 75% of the numbers $0 < x < n$ are Miller-Rabin witness!

Miller-Rabin test

Question

How many random x do we need to guarantee detection of a composite number n?

Theorem

Let n be an odd composite number, then at least 75% of the numbers $0 < x < n$ are Miller-Rabin witness!

Conditional on Generalized Riemann Hypothesis (GRH), one can prove that for composite n, one of $2, 3, \cdots, (\log n)^2$ gives a Miller-Rabin witness.

Miller-Rabin test

Question

How many random x do we need to guarantee detection of a composite number n?

Theorem

Let n be an odd composite number, then at least 75% of the numbers $0 < x < n$ are Miller-Rabin witness!

Conditional on Generalized Riemann Hypothesis (GRH), one can prove that for composite n, one of $2, 3, \cdots, (\log n)^2$ gives a Miller-Rabin witness.So *conditionally* you only need to use $(\log n)^2$ witness, making it a (conditional) *deterministic polynomial time* test.

Other Primality Test?

- The two tests we mentioned both based on some characterization of primes using Fermat's little theorem.

Other Primality Test?

- The two tests we mentioned both based on some characterization of primes using Fermat's little theorem.
- If you can find other ways to characterize a prime, you may be able to make it into a primality test.

Other Primality Test?

- The two tests we mentioned both based on some characterization of primes using Fermat's little theorem.
- If you can find other ways to characterize a prime, you may be able to make it into a primality test.
- Examples: Lucas-Lehmer test (Lucas sequence),

Other Primality Test?

- The two tests we mentioned both based on some characterization of primes using Fermat's little theorem.
- If you can find other ways to characterize a prime, you may be able to make it into a primality test.
- Examples: Lucas-Lehmer test (Lucas sequence),Solovay-Strassen test (Euler's criterion).

Other Primality Test?

- The two tests we mentioned both based on some characterization of primes using Fermat's little theorem.
- If you can find other ways to characterize a prime, you may be able to make it into a primality test.
- Examples: Lucas-Lehmer test (Lucas sequence),Solovay-Strassen test (Euler's criterion).
- Are there tests that are provably deterministic and polynomial time? Yes!

Other Primality Test?

- The two tests we mentioned both based on some characterization of primes using Fermat's little theorem.
- If you can find other ways to characterize a prime, you may be able to make it into a primality test.
- Examples: Lucas-Lehmer test (Lucas sequence),Solovay-Strassen test (Euler's criterion).
- Are there tests that are provably deterministic and polynomial time? Yes!The AKS test was proposed by Agrawal, Kayal, Saxena in 2002 at IIT Kanpur, coming out of an undergraduate research project (!)

Other Primality Test?

- The two tests we mentioned both based on some characterization of primes using Fermat's little theorem.
- If you can find other ways to characterize a prime, you may be able to make it into a primality test.
- Examples: Lucas-Lehmer test (Lucas sequence),Solovay-Strassen test (Euler's criterion).
- Are there tests that are provably deterministic and polynomial time? Yes!The AKS test was proposed by Agrawal, Kayal, Saxena in 2002 at IIT Kanpur, coming out of an undergraduate research project (!)It is an $O((\log n)^{12})$ algorithm, which was later improved to $O((\log n)^6)$ by Lenstra and Pomerance.

Other Primality Test?

- The two tests we mentioned both based on some characterization of primes using Fermat's little theorem.
- If you can find other ways to characterize a prime, you may be able to make it into a primality test.
- Examples: Lucas-Lehmer test (Lucas sequence),Solovay-Strassen test (Euler's criterion).
- Are there tests that are provably deterministic and polynomial time? Yes!The AKS test was proposed by Agrawal, Kayal, Saxena in 2002 at IIT Kanpur, coming out of an undergraduate research project (!)It is an $O((\log n)^{12})$ algorithm, which was later improved to $O((\log n)^6)$ by Lenstra and Pomerance.
- The currently fastest test is Elliptic Curve Primality Proving (ECPP), which runs heuristically in time $O((\log n)^{4+\epsilon})$, although the exact worst execution time is not known.

How to get large primes?

Let's say we want to find a 1024-bit prime, which has $\sim \log_{10}(2^{1024}) \sim 309$ digits. How can we find a prime of that size?

How to get large primes?

Let's say we want to find a 1024-bit prime, which has $\sim \log_{10}(2^{1024}) \sim 309$ digits. How can we find a prime of that size?

- We can generate a random number of 309 digits and check if it is a prime.

How to get large primes?

Let's say we want to find a 1024-bit prime, which has $\sim \log_{10}(2^{1024}) \sim 309$ digits. How can we find a prime of that size?

- We can generate a random number of 309 digits and check if it is a prime.If not, keep going.... but when can we stop?

How to get large primes?

Let's say we want to find a 1024-bit prime, which has $\sim \log_{10}(2^{1024}) \sim 309$ digits. How can we find a prime of that size?

- We can generate a random number of 309 digits and check if it is a prime.If not, keep going.... but when can we stop?
- The real question here is then

How to get large primes?

Let's say we want to find a 1024-bit prime, which has $\sim \log_{10}(2^{1024}) \sim 309$ digits. How can we find a prime of that size?

- We can generate a random number of 309 digits and check if it is a prime.If not, keep going.... but when can we stop?
- The real question here is then

Question

For n large, what is the probability for a number of size around n to be a prime?

How to get large primes?

(Guessed by Gauss, essentially due to Riemann, but proved by Hadamard-de la Poussin)

How to get large primes?

(Guessed by Gauss, essentially due to Riemann, but proved by Hadamard-de la Poussin)

Theorem (Prime Number Theorem)

If $\pi(x)$ is the number of primes up to x, then

$$\pi(x) \sim \frac{x}{\log x} \text{ as } x \to \infty$$

How to get large primes?

(Guessed by Gauss, essentially due to Riemann, but proved by Hadamard-de la Poussin)

Theorem (Prime Number Theorem)

If $\pi(x)$ is the number of primes up to x, then

$$\pi(x) \sim \frac{x}{\log x} \text{ as } x \to \infty$$

In fact Riemann said more

Theorem (Prime Number Theorem)

If $\pi(x)$ is the number of primes up to x, then

$$\pi(x) \sim \int_2^x \frac{1}{\log x} dx \text{ as } x \to \infty$$

How to get large primes?

Riemann also suggested the

Riemann Hypothesis

If $\pi(x)$ is the number of primes up to x, then

$$\pi(x) = \int_2^x \frac{1}{\log x} dx + O\left(x^{1/2+\epsilon}\right) \text{ as } x \to \infty$$

Solving Discrete Log Problem

- Recall that Diffie-Hellman rests on the Diffie-Hellman Problem (DHP), which is believed to be hard.
- No one knows how to approach DHP without first solving the discrete log problem. (DLP)

Solving Discrete Log Problem

- Recall that Diffie-Hellman rests on the Diffie-Hellman Problem (DHP), which is believed to be hard.
- No one knows how to approach DHP without first solving the discrete log problem. (DLP)
- Recall that DLP is

Discrete Log Problem

Given prime p, primitive root g and $g^a \mod p$ for some unknown a, it is hard to know what a is.

Solving Discrete Log Problem

Naive way:

Solving Discrete Log Problem

Naive way:

- Compute $g^1, g^2, \cdots, g^{p-1} \mod p$ and compare it with the public key we see.

Solving Discrete Log Problem

Naive way:

- Compute $g^1, g^2, \cdots, g^{p-1} \bmod p$ and compare it with the public key we see.
- In the worst case, number of multiplications needed?

Solving Discrete Log Problem

Naive way:

- Compute $g^1, g^2, \cdots, g^{p-1} \mod p$ and compare it with the public key we see.
- In the worst case, number of multiplications needed?
- This is an *exponential time* algorithm!

Solving Discrete Log Problem

Naive way:

- Compute $g^1, g^2, \cdots, g^{p-1} \mod p$ and compare it with the public key we see.
- In the worst case, number of multiplications needed?
- This is an *exponential time* algorithm!

Can we do better?

Solving Discrete Log Problem

Naive way:

- Compute $g^1, g^2, \cdots, g^{p-1} \bmod p$ and compare it with the public key we see.
- In the worst case, number of multiplications needed?
- This is an *exponential time* algorithm!

Can we do better?

We will describe two quicker algorithms - Shanks' baby-step-Giant-step algorithm, and Pohlig-Hellman algorithm.

Baby-Step-Giant-Step

We want to find the discrete logarithm of $h = g^k \mod p$. Instead of computing $g^1, g^2, \cdots, g^{p-1}$ one by one, we will

Baby-Step-Giant-Step

We want to find the discrete logarithm of $h = g^k \mod p$. Instead of computing $g^1, g^2, \cdots, g^{p-1}$ one by one, we will

Baby-step-Giant-step algorithm

- compute $1, g^1, \cdots, g^{[\sqrt{p}]}$, and store it in a list. (This is the baby-step)

Baby-Step-Giant-Step

We want to find the discrete logarithm of $h = g^k \mod p$. Instead of computing $g^1, g^2, \cdots, g^{p-1}$ one by one, we will

Baby-step-Giant-step algorithm

- compute $1, g^1, \cdots, g^{[\sqrt{p}]}$, and store it in a list. (This is the baby-step)
- compute $hg^{-i[\sqrt{p}]}$ for $i = 0, \cdots, [\sqrt{p}] + 1$. Check against the list if you find a match. (This is the giant-step)

Baby-Step-Giant-Step

We want to find the discrete logarithm of $h = g^k \mod p$. Instead of computing $g^1, g^2, \cdots, g^{p-1}$ one by one, we will

Baby-step-Giant-step algorithm

- compute $1, g^1, \cdots, g^{[\sqrt{p}]}$, and store it in a list. (This is the baby-step)
- compute $hg^{-i[\sqrt{p}]}$ for $i = 0, \cdots, [\sqrt{p}] + 1$. Check against the list if you find a match. (This is the giant-step)

When you find a match, you have found i, j so that $hg^{-in} = g^j$, so $h = g^{in+j}$.

Baby-Step-Giant-Step

Example

Suppose we want to find the discrete logarithm of $h = 93$, for primitive root $g = 41$ modulo $p = 317$.

Baby-Step-Giant-Step

Example

Suppose we want to find the discrete logarithm of $h = 93$, for primitive root $g = 41$ modulo $p = 317$. First compute $[\sqrt{317}] = 17$.

Baby-Step-Giant-Step

Example

Suppose we want to find the discrete logarithm of $h = 93$, for primitive root $g = 41$ modulo $p = 317$. First compute $[\sqrt{317}] = 17$.

- The baby step: we need to compute the list $1, g^1, \cdots, g^{17}$. We obtain the following numbers when doing this,

$$1, 41, 96, 132, 23, 309, 306, 183, 212, 133,$$

$$64, 88, 121, 206, 204, 122, 247, 300.$$

Example

Suppose we want to find the discrete logarithm of $h = 93$, for primitive root $g = 41$ modulo $p = 317$. First compute $[\sqrt{317}] = 17$.

- The baby step: we need to compute the list $1, g^1, \cdots, g^{17}$. We obtain the following numbers when doing this,

$$1, 41, 96, 132, 23, 309, 306, 183, 212, 133,$$

$$64, 88, 121, 206, 204, 122, 247, 300.$$

We would then sort this list for the ease of searching in the giant step - for simplicity we are skipping this step for now.

Baby-Step-Giant-Step

Example

Suppose we want to find the discrete logarithm of $h = 93$, for primitive root $g = 41$ modulo $p = 317$. First compute $[\sqrt{317}] = 17$.

- The baby step: we need to compute the list $1, g^1, \cdots, g^{17}$. We obtain the following numbers when doing this,

$$1, 41, 96, 132, 23, 309, 306, 183, 212, 133,$$

$$64, 88, 121, 206, 204, 122, 247, 300.$$

  We would then sort this list for the ease of searching in the giant step - for simplicity we are skipping this step for now.

- The giant step: we need to see if hg^{-17i} lies in the above list, starting from $i = 0$.

Baby-Step-Giant-Step

Example

Suppose we want to find the discrete logarithm of $h = 93$, for primitive root $g = 41$ modulo $p = 317$. First compute $[\sqrt{317}] = 17$.

- The baby step: we need to compute the list $1, g^1, \cdots, g^{17}$. We obtain the following numbers when doing this,

$$1, 41, 96, 132, 23, 309, 306, 183, 212, 133,$$

$$64, 88, 121, 206, 204, 122, 247, 300.$$

  We would then sort this list for the ease of searching in the giant step - for simplicity we are skipping this step for now.

- The giant step: we need to see if hg^{-17i} lies in the above list, starting from $i = 0$.
  - When $i = 0$, $hg^{-17 \cdot 0} = h = 93$

Baby-Step-Giant-Step

Example

Suppose we want to find the discrete logarithm of $h = 93$, for primitive root $g = 41$ modulo $p = 317$. First compute $[\sqrt{317}] = 17$.

- The baby step: we need to compute the list $1, g^1, \cdots, g^{17}$. We obtain the following numbers when doing this,

$$1, 41, 96, 132, 23, 309, 306, 183, 212, 133,$$

$$64, 88, 121, 206, 204, 122, 247, 300.$$

  We would then sort this list for the ease of searching in the giant step - for simplicity we are skipping this step for now.
- The giant step: we need to see if hg^{-17i} lies in the above list, starting from $i = 0$.
  - When $i = 0$, $hg^{-17 \cdot 0} = h = 93$, and it is not on the list.

Baby-Step-Giant-Step

Example

Suppose we want to find the discrete logarithm of $h = 93$, for primitive root $g = 41$ modulo $p = 317$. First compute $[\sqrt{317}] = 17$.

- The baby step: we need to compute the list $1, g^1, \cdots, g^{17}$. We obtain the following numbers when doing this,

$$1, 41, 96, 132, 23, 309, 306, 183, 212, 133,$$

$$64, 88, 121, 206, 204, 122, 247, 300.$$

  We would then sort this list for the ease of searching in the giant step - for simplicity we are skipping this step for now.
- The giant step: we need to see if hg^{-17i} lies in the above list, starting from $i = 0$.
  - When $i = 0$, $hg^{-17\cdot 0} = h = 93$, and it is not on the list.
  - When $i = 1$, $hg^{-17\cdot 1} = h = 181$

Baby-Step-Giant-Step

Example

Suppose we want to find the discrete logarithm of $h = 93$, for primitive root $g = 41$ modulo $p = 317$. First compute $[\sqrt{317}] = 17$.

- The baby step: we need to compute the list $1, g^1, \cdots, g^{17}$. We obtain the following numbers when doing this,

  $$1, 41, 96, 132, 23, 309, 306, 183, 212, 133,$$

  $$64, 88, 121, 206, 204, 122, 247, 300.$$

  We would then sort this list for the ease of searching in the giant step - for simplicity we are skipping this step for now.
- The giant step: we need to see if hg^{-17i} lies in the above list, starting from $i = 0$.
  - When $i = 0$, $hg^{-17\cdot 0} = h = 93$, and it is not on the list.
  - When $i = 1$, $hg^{-17\cdot 1} = h = 181$, also not on the list.

Baby-Step-Giant-Step

Example

(Giant step continued)

Baby-Step-Giant-Step

Example

(Giant step continued)

- Keep doing this. Eventually we found that when $i = 11$, $hg^{-17 \cdot 11} = 64$, which is on the list, corresponding to g^{10}.

Example

(Giant step continued)

- Keep doing this. Eventually we found that when $i = 11$, $hg^{-17\cdot 11} = 64$, which is on the list, corresponding to g^{10}.This means that

$$hg^{-17\cdot 11} \equiv g^{10} \mod p \Rightarrow h \equiv g^{11\cdot 17+10} \mod p \equiv g^{197} \mod p$$

Baby-Step-Giant-Step

Example

(Giant step continued)

- Keep doing this. Eventually we found that when $i = 11$, $hg^{-17\cdot 11} = 64$, which is on the list, corresponding to g^{10}.This means that

$$hg^{-17\cdot 11} \equiv g^{10} \mod p \Rightarrow h \equiv g^{11\cdot 17+10} \mod p \equiv g^{197} \mod p$$

Thus, the discrete logarithm of h in this case is 197.

Baby-Step-Giant-Step

Why does it work?

Baby-Step-Giant-Step

Why does it work?

- In the initial list, we check if h is one of the following,

$$1, g^1, \cdots, g^{[\sqrt{p}]}$$

Baby-Step-Giant-Step

Why does it work?

- In the initial list, we check if h is one of the following,

$$1, g^1, \cdots, g^{[\sqrt{p}]}$$

- The second step is to check if $hg^{-[\sqrt{p}]}$ shows up in the above list,

Baby-Step-Giant-Step

Why does it work?

- In the initial list, we check if h is one of the following,

$$1, g^1, \cdots, g^{[\sqrt{p}]}$$

- The second step is to check if $hg^{-[\sqrt{p}]}$ shows up in the above list,i.e. whether h shows up in

$$g^{[\sqrt{p}]}, g^{[\sqrt{p}]+1}, \cdots, g^{2[\sqrt{p}]}$$

Baby-Step-Giant-Step

Why does it work?

- In the initial list, we check if h is one of the following,

$$1, g^1, \cdots, g^{[\sqrt{p}]}$$

- The second step is to check if $hg^{-[\sqrt{p}]}$ shows up in the above list,i.e. whether h shows up in

$$g^{[\sqrt{p}]}, g^{[\sqrt{p}]+1}, \cdots, g^{2[\sqrt{p}]}$$

- We keep going, all the way until checking whether h shows up in

$$g^{[\sqrt{p}]^2}, g^{[\sqrt{p}]([\sqrt{p}]+1)+1}, \cdots, g^{[\sqrt{p}]([\sqrt{p}]+2)}$$

Baby-Step-Giant-Step

Why does it work?

- In the initial list, we check if h is one of the following,

$$1, g^1, \cdots, g^{[\sqrt{p}]}$$

- The second step is to check if $hg^{-[\sqrt{p}]}$ shows up in the above list,i.e. whether h shows up in

$$g^{[\sqrt{p}]}, g^{[\sqrt{p}]+1}, \cdots, g^{2[\sqrt{p}]}$$

- We keep going, all the way until checking whether h shows up in

$$g^{[\sqrt{p}]^2}, g^{[\sqrt{p}]([\sqrt{p}]+1)+1}, \cdots, g^{[\sqrt{p}]([\sqrt{p}]+2)}$$

- So as long as $[\sqrt{p}]([\sqrt{p}]+2) \geq p-1$, then we finish listing all the elements, so we win.

Baby-Step-Giant-Step

Why does it work?

- In the initial list, we check if h is one of the following,

$$1, g^1, \cdots, g^{[\sqrt{p}]}$$

- The second step is to check if $hg^{-[\sqrt{p}]}$ shows up in the above list,i.e. whether h shows up in

$$g^{[\sqrt{p}]}, g^{[\sqrt{p}]+1}, \cdots, g^{2[\sqrt{p}]}$$

- We keep going, all the way until checking whether h shows up in

$$g^{[\sqrt{p}]^2}, g^{[\sqrt{p}]([\sqrt{p}]+1)+1}, \cdots, g^{[\sqrt{p}]([\sqrt{p}]+2)}$$

- So as long as $[\sqrt{p}]([\sqrt{p}]+2) \geq p-1$, then we finish listing all the elements, so we win.But $[\sqrt{p}] > \sqrt{p}-1$, so $[\sqrt{p}]+2 > \sqrt{p}+1$, and

$$[\sqrt{p}]([\sqrt{p}]+2) > (\sqrt{p}-1)(\sqrt{p}+1) = p-1$$

Running time?
Initially, one may feel like the running time is again $O(p)$

Baby-Step-Giant-Step

Running time?
Initially, one may feel like the running time is again $O(p)$, because

- each baby step naively takes $O(\sqrt{p})$ time.

Baby-Step-Giant-Step

Running time?

Initially, one may feel like the running time is again $O(p)$, because

- each baby step naively takes $O(\sqrt{p})$ time.
- At each giant step, we need to check whether an element lies in a list of length $O(\sqrt{p})$.

Baby-Step-Giant-Step

Running time?
Initially, one may feel like the running time is again $O(p)$, because

- each baby step naively takes $O(\sqrt{p})$ time.
- At each giant step, we need to check whether an element lies in a list of length $O(\sqrt{p})$.Element-by-element comparison takes $O(\sqrt{p})$ time.

Baby-Step-Giant-Step

Running time?

Initially, one may feel like the running time is again $O(p)$, because

- each baby step naively takes $O(\sqrt{p})$ time.
- At each giant step, we need to check whether an element lies in a list of length $O(\sqrt{p})$.Element-by-element comparison takes $O(\sqrt{p})$ time.Since there are $O(\sqrt{p})$ giant steps, it will take

Baby-Step-Giant-Step

Running time?

Initially, one may feel like the running time is again $O(p)$, because

- each baby step naively takes $O(\sqrt{p})$ time.
- At each giant step, we need to check whether an element lies in a list of length $O(\sqrt{p})$.Element-by-element comparison takes $O(\sqrt{p})$ time.Since there are $O(\sqrt{p})$ giant steps, it will take$O(\sqrt{p}) \cdot O(\sqrt{p}) = O(p)$ time in total.

Baby-Step-Giant-Step

Running time?

Initially, one may feel like the running time is again $O(p)$, because

- each baby step naively takes $O(\sqrt{p})$ time.
- At each giant step, we need to check whether an element lies in a list of length $O(\sqrt{p})$.Element-by-element comparison takes $O(\sqrt{p})$ time.Since there are $O(\sqrt{p})$ giant steps, it will take$O(\sqrt{p}) \cdot O(\sqrt{p}) = O(p)$ time in total.

Can we improve this?

But searching actually takes a lot less time!

Baby-Step-Giant-Step

But searching actually takes a lot less time!

- One way is to first sort the list. It is doable to sort a list of length n in $O(n \log n)$ time. Then searching can be done in $O(\log n)$ time once the list is sorted.

Baby-Step-Giant-Step

But searching actually takes a lot less time!

- One way is to first sort the list. It is doable to sort a list of length n in $O(n \log n)$ time. Then searching can be done in $O(\log n)$ time once the list is sorted.
- In our case, it means that sorting a list of length $O(\sqrt{p})$ takes $O(p^{1/2} \log p)$ time, something we only need to do once. (Pre-processing of baby-step)

Baby-Step-Giant-Step

But searching actually takes a lot less time!

- One way is to first sort the list. It is doable to sort a list of length n in $O(n \log n)$ time. Then searching can be done in $O(\log n)$ time once the list is sorted.
- In our case, it means that sorting a list of length $O(\sqrt{p})$ takes $O(p^{1/2} \log p)$ time, something we only need to do once. (Pre-processing of baby-step)
- Then each giant step takes only $O(\log p)$ time, meaning that giant steps contribute $O(p^{1/2} \log p)$ time in total.

Baby-Step-Giant-Step

But searching actually takes a lot less time!

- One way is to first sort the list. It is doable to sort a list of length n in $O(n \log n)$ time. Then searching can be done in $O(\log n)$ time once the list is sorted.
- In our case, it means that sorting a list of length $O(\sqrt{p})$ takes $O(p^{1/2} \log p)$ time, something we only need to do once. (Pre-processing of baby-step)
- Then each giant step takes only $O(\log p)$ time, meaning that giant steps contribute $O(p^{1/2} \log p)$ time in total.
- Thus the total running time is $O(p^{1/2+\epsilon})$, a significant improvement of the naive algorithm.

Pohlig-Hellman

- Sometimes special features of p would help.

Pohlig-Hellman

- Sometimes special features of p would help.
- One thing that often helps is the *factorization of* $p - 1$.

Pohlig-Hellman

- Sometimes special features of p would help.
- One thing that often helps is the *factorization of* $p - 1$.
- Why would it help?

Pohlig-Hellman

- Sometimes special features of p would help.
- One thing that often helps is the *factorization of* $p - 1$.
- Why would it help?The discrete log problem is really a problem modulo $p - 1$.

Pohlig-Hellman

- Sometimes special features of p would help.
- One thing that often helps is the *factorization of* $p - 1$.
- Why would it help?The discrete log problem is really a problem modulo $p - 1$.
- If we know the prime factorization of $p - 1$, then by Chinese remainder theorem, we only need to understand the exponent mod each prime power factor of $p - 1$.

Pohlig-Hellman

Let us see an example:

Example

Take $p = 11251$, $g = 23$ (a primitive root), and $h = 9689$. We wish to find k such that

$$23^k \equiv 9689 \bmod 11251$$

Remember that k is defined modulo $p - 1$.

Pohlig-Hellman

Let us see an example:

Example

Take $p = 11251$, $g = 23$ (a primitive root), and $h = 9689$. We wish to find k such that

$$23^k \equiv 9689 \mod 11251$$

Remember that k is defined modulo $p - 1$.

Solution

- *Observe that* $p - 1 = 2 \cdot 3^2 \cdot 5^4$

Pohlig-Hellman

Let us see an example:

Example

Take $p = 11251$, $g = 23$ (a primitive root), and $h = 9689$. We wish to find k such that

$$23^k \equiv 9689 \bmod 11251$$

Remember that k is defined modulo $p - 1$.

Solution

- *Observe that* $p - 1 = 2 \cdot 3^2 \cdot 5^4$*, so by Chinese Remainder Theorem, it suffices to understand* $k \bmod 2, k \bmod 3^2$ *and* $k \bmod 5^4$*.*

Pohlig-Hellman

Let us see an example:

Example

Take $p = 11251$, $g = 23$ (a primitive root), and $h = 9689$. We wish to find k such that

$$23^k \equiv 9689 \bmod 11251$$

Remember that k is defined modulo $p - 1$.

Solution

- *Observe that* $p - 1 = 2 \cdot 3^2 \cdot 5^4$*, so by Chinese Remainder Theorem, it suffices to understand* $k \bmod 2, k \bmod 3^2$ *and* $k \bmod 5^4$.
- *How to figure out* $k \bmod 2$*?*

Pohlig-Hellman

Let us see an example:

Example

Take $p = 11251$, $g = 23$ (a primitive root), and $h = 9689$. We wish to find k such that

$$23^k \equiv 9689 \bmod 11251$$

Remember that k is defined modulo $p - 1$.

Solution

- *Observe that* $p - 1 = 2 \cdot 3^2 \cdot 5^4$, *so by Chinese Remainder Theorem, it suffices to understand* $k \bmod 2, k \bmod 3^2$ *and* $k \bmod 5^4$.
- *How to figure out* $k \bmod 2$*?*
- *We can use the square test!*

Pohlig-Hellman

Let us see an example:

Example

Take $p = 11251$, $g = 23$ (a primitive root), and $h = 9689$. We wish to find k such that

$$23^k \equiv 9689 \mod 11251$$

Remember that k is defined modulo $p - 1$.

Solution

- *Observe that* $p - 1 = 2 \cdot 3^2 \cdot 5^4$, *so by Chinese Remainder Theorem, it suffices to understand* $k \mod 2, k \mod 3^2$ *and* $k \mod 5^4$.
- *How to figure out* $k \mod 2$*?*
- *We can use the square test!For any* $a \in \mathbb{F}_p^*$,

$$a^{\frac{p-1}{2}} \equiv \begin{cases} 1 \mod p & (\textit{if } a \textit{ is a square in } \mathbb{F}_p^*) \\ -1 \mod p & (\textit{if } a \textit{ is not a square in } \mathbb{F}_p^*) \end{cases}$$

Pohlig-Hellman

Solution

- *In our case,*

Pohlig-Hellman

Solution

- *In our case,*
  - *if* k *is even, then* 9689 *is a square, so* $9689^{\frac{p-1}{2}} \equiv 1 \mod p$
  - *if* k *is odd, then* 9689 *is not a square, so* $9689^{\frac{p-1}{2}} \equiv -1 \mod p$

Pohlig-Hellman

Solution

- *In our case,*
  - *if k is even, then 9689 is a square, so* $9689^{\frac{p-1}{2}} \equiv 1 \mod p$
  - *if k is odd, then 9689 is not a square, so* $9689^{\frac{p-1}{2}} \equiv -1 \mod p$
- *Thus it suffices to check* $9689^{\frac{p-1}{2}}$ *to see if it is a square.*

Pohlig-Hellman

Solution

- *In our case,*
  - *if k is even, then 9689 is a square, so* $9689^{\frac{p-1}{2}} \equiv 1 \bmod p$
  - *if k is odd, then 9689 is not a square, so* $9689^{\frac{p-1}{2}} \equiv -1 \bmod p$
- *Thus it suffices to check* $9689^{\frac{p-1}{2}}$ *to see if it is a square.By repeated squaring, one can check that*

$$9689^{\frac{p-1}{2}} \equiv -1 \bmod 11251,$$

Pohlig-Hellman

Solution

- *In our case,*
  - *if k is even, then 9689 is a square, so* $9689^{\frac{p-1}{2}} \equiv 1 \mod p$
  - *if k is odd, then 9689 is not a square, so* $9689^{\frac{p-1}{2}} \equiv -1 \mod p$
- *Thus it suffices to check* $9689^{\frac{p-1}{2}}$ *to see if it is a square.By repeated squaring, one can check that*

$$9689^{\frac{p-1}{2}} \equiv -1 \mod 11251,$$

so k is odd, i.e. $k \equiv 1 \mod 2$.

Solution

- *In our case,*
  - *if* k *is even, then* 9689 *is a square, so* $9689^{\frac{p-1}{2}} \equiv 1 \mod p$
  - *if* k *is odd, then* 9689 *is not a square, so* $9689^{\frac{p-1}{2}} \equiv -1 \mod p$
- *Thus it suffices to check* $9689^{\frac{p-1}{2}}$ *to see if it is a square.By repeated squaring, one can check that*

$$9689^{\frac{p-1}{2}} \equiv -1 \mod 11251,$$

  so k *is odd, i.e.* $k \equiv 1 \mod 2$.
- *What about* $k \mod 3^2$*?*

Pohlig-Hellman

Solution

- *In our case,*
  - *if k is even, then 9689 is a square, so* $9689^{\frac{p-1}{2}} \equiv 1 \mod p$
  - *if k is odd, then 9689 is not a square, so* $9689^{\frac{p-1}{2}} \equiv -1 \mod p$
- *Thus it suffices to check* $9689^{\frac{p-1}{2}}$ *to see if it is a square.By repeated squaring, one can check that*
$$9689^{\frac{p-1}{2}} \equiv -1 \mod 11251,$$
so k is odd, i.e. $k \equiv 1 \mod 2$.
- *What about* $k \mod 3^2$?
- *We will first work out the baby case* $k \mod 3$.

Pohlig-Hellman

Solution

(Refined) Cube test

Let p be a prime such that $3|p-1$, g be a primitive root $\bmod p$ *and* $a = g^k \in \mathbb{F}_p^*$,

$$a^{\frac{p-1}{3}} \equiv \begin{cases} 1 \bmod\ p \equiv \left(g^{\frac{p-1}{3}}\right)^0 \bmod\ p & \textit{(if } k \equiv 0 \bmod\ 3\textit{)} \\ g^{\frac{p-1}{3}} \bmod\ p \equiv \left(g^{\frac{p-1}{3}}\right)^1 \bmod\ p & \textit{(if } k \equiv 1 \bmod\ 3\textit{)} \\ g^{\frac{2(p-1)}{3}} \bmod\ p = \left(g^{\frac{p-1}{3}}\right)^2 \bmod\ p & \textit{(if } k \equiv 2 \bmod\ 3\textit{)} \end{cases}$$

Pohlig-Hellman

Solution

(Refined) Cube test

Let p be a prime such that $3|p-1$, g be a primitive root mod p *and* $a = g^k \in \mathbb{F}_p^*$,

$$a^{\frac{p-1}{3}} \equiv \begin{cases} 1 \bmod p \equiv \left(g^{\frac{p-1}{3}}\right)^0 \bmod p & \textit{(if } k \equiv 0 \bmod 3\textit{)} \\ g^{\frac{p-1}{3}} \bmod p \equiv \left(g^{\frac{p-1}{3}}\right)^1 \bmod p & \textit{(if } k \equiv 1 \bmod 3\textit{)} \\ g^{\frac{2(p-1)}{3}} \bmod p = \left(g^{\frac{p-1}{3}}\right)^2 \bmod p & \textit{(if } k \equiv 2 \bmod 3\textit{)} \end{cases}$$

- *Thus,* k mod 3 *can be figured out by checking*
$$9689^{\frac{p-1}{3}} \equiv (g^{\frac{p-1}{3}})^k \bmod p.$$

Pohlig-Hellman

Solution

(Refined) Cube test

Let p be a prime such that $3|p-1$, g be a primitive root $\bmod p$ *and* $a = g^k \in \mathbb{F}_p^*$,

$$a^{\frac{p-1}{3}} \equiv \begin{cases} 1 \bmod p \equiv \left(g^{\frac{p-1}{3}}\right)^0 \bmod p & \textit{(if } k \equiv 0 \bmod 3\textit{)} \\ g^{\frac{p-1}{3}} \bmod p \equiv \left(g^{\frac{p-1}{3}}\right)^1 \bmod p & \textit{(if } k \equiv 1 \bmod 3\textit{)} \\ g^{\frac{2(p-1)}{3}} \bmod p = \left(g^{\frac{p-1}{3}}\right)^2 \bmod p & \textit{(if } k \equiv 2 \bmod 3\textit{)} \end{cases}$$

- *Thus,* $k \bmod 3$ *can be figured out by checking*
$$9689^{\frac{p-1}{3}} \equiv (g^{\frac{p-1}{3}})^k \bmod p.$$

Repeated squaring gives $k \equiv 1 \bmod 3$.

Pohlig-Hellman

Solution

(Refined) Cube test

Let p be a prime such that $3|p-1$, g be a primitive root mod p *and* $a = g^k \in \mathbb{F}_p^*$,

$$a^{\frac{p-1}{3}} \equiv \begin{cases} 1 \bmod p \equiv \left(g^{\frac{p-1}{3}}\right)^0 \bmod p & \text{(if } k \equiv 0 \bmod 3\text{)} \\ g^{\frac{p-1}{3}} \bmod p \equiv \left(g^{\frac{p-1}{3}}\right)^1 \bmod p & \text{(if } k \equiv 1 \bmod 3\text{)} \\ g^{\frac{2(p-1)}{3}} \bmod p = \left(g^{\frac{p-1}{3}}\right)^2 \bmod p & \text{(if } k \equiv 2 \bmod 3\text{)} \end{cases}$$

- *Thus,* $k \bmod 3$ *can be figured out by checking*

$$9689^{\frac{p-1}{3}} \equiv (g^{\frac{p-1}{3}})^k \bmod p.$$

  Repeated squaring gives $k \equiv 1 \bmod 3$.
- *Note: we are doing a simpler discrete log problem!*

Pohlig-Hellman

Solution

- *To find* $k \bmod 3^2$, *develop the* 3^2*th-power test.*

Pohlig-Hellman

Solution

- *To find* $k \mod 3^2$*, develop the* 3^2*th-power test.*
- *In other words,* $k \mod 3^2$ *can be figured out by checking*

$$9689^{\frac{p-1}{3^2}} \equiv (g^{\frac{p-1}{3^2}})^k \mod p$$

Pohlig-Hellman

Solution

- *To find* $k \bmod 3^2$, *develop the* 3^2*th-power test.*
- *In other words,* $k \bmod 3^2$ *can be figured out by checking*

$$9689^{\frac{p-1}{3^2}} \equiv (g^{\frac{p-1}{3^2}})^k \mod p$$

We can find that $k \equiv 4 \mod 3^2$.

Pohlig-Hellman

Solution

- *To find* $k \bmod 3^2$, *develop the* 3^2*th-power test.*
- *In other words,* $k \bmod 3^2$ *can be figured out by checking*

$$9689^{\frac{p-1}{3^2}} \equiv (g^{\frac{p-1}{3^2}})^k \mod p$$

  We can find that $k \equiv 4 \mod 3^2$.
- *Similarly, one sees that* $k \equiv 511 \mod 5^4$.

Pohlig-Hellman

Solution

- *To find* $k \bmod 3^2$*, develop the* 3^2*th-power test.*
- *In other words,* $k \bmod 3^2$ *can be figured out by checking*

$$9689^{\frac{p-1}{3^2}} \equiv (g^{\frac{p-1}{3^2}})^k \bmod p$$

  We can find that $k \equiv 4 \bmod 3^2$*.*

- *Similarly, one sees that* $k \equiv 511 \bmod 5^4$*.*
- *Now solve* k *by Chinese remainder theorem for the system*

$$\begin{cases} k \equiv 1 \bmod 2 \\ k \equiv 4 \bmod 3^2 \\ k \equiv 511 \bmod 5^4 \end{cases}$$

Pohlig-Hellman

Solution

- *To find* $k \mod 3^2$*, develop the* 3^2*th-power test.*
- *In other words,* $k \mod 3^2$ *can be figured out by checking*
$$9689^{\frac{p-1}{3^2}} \equiv (g^{\frac{p-1}{3^2}})^k \mod p$$

  We can find that $k \equiv 4 \mod 3^2$*.*
- *Similarly, one sees that* $k \equiv 511 \mod 5^4$*.*
- *Now solve* k *by Chinese remainder theorem for the system*
$$\begin{cases} k \equiv 1 \mod 2 \\ k \equiv 4 \mod 3^2 \\ k \equiv 511 \mod 5^4 \end{cases}$$
- *The smallest solution would be* $4261 \mod 11250$*. Therefore, the discrete log in this case is 4261, i.e.*
$$23^{4261} \equiv 9689 \mod 11251.$$

Pohlig-Hellman

What if $p - 1$ is has a large prime power?

Pohlig-Hellman

What if $p-1$ is has a large prime power?We can refine the process for 3^2 as follows.

- We want to find $k \mod 3^2$.

Pohlig-Hellman

What if $p - 1$ is has a large prime power?We can refine the process for 3^2 as follows.

- We want to find $k \mod 3^2$.First find $k \mod 3$ as before - we got $k \equiv 1 \mod 3$.

Pohlig-Hellman

What if $p - 1$ is has a large prime power?We can refine the process for 3^2 as follows.

- We want to find $k \bmod 3^2$.First find $k \bmod 3$ as before - we got $k \equiv 1 \bmod 3$.
- Write $k = 1 + 3k'$, where k' is 0,1 or 2 mod 3.

Pohlig-Hellman

What if $p-1$ is has a large prime power?We can refine the process for 3^2 as follows.

- We want to find $k \bmod 3^2$.First find $k \bmod 3$ as before - we got $k \equiv 1 \bmod 3$.
- Write $k = 1 + 3k'$, where k' is 0,1 or 2 mod 3.Note that

$$9689^{\frac{p-1}{3^2}} \equiv (g^{\frac{p-1}{3^2}})^k \bmod p$$

Pohlig-Hellman

What if $p-1$ is has a large prime power?We can refine the process for 3^2 as follows.

- We want to find $k \bmod 3^2$.First find $k \bmod 3$ as before - we got $k \equiv 1 \bmod 3$.
- Write $k = 1 + 3k'$, where k' is 0,1 or 2 mod 3.Note that

$$9689^{\frac{p-1}{3^2}} \equiv (g^{\frac{p-1}{3^2}})^k \bmod p \equiv (g^{\frac{p-1}{3^2}})(g^{\frac{p-1}{3}})^{k'} \bmod p.$$

Pohlig-Hellman

What if $p-1$ is has a large prime power?We can refine the process for 3^2 as follows.

- We want to find $k \bmod 3^2$.First find $k \bmod 3$ as before - we got $k \equiv 1 \bmod 3$.
- Write $k = 1 + 3k'$, where k' is 0,1 or 2 mod 3.Note that

$$9689^{\frac{p-1}{3^2}} \equiv (g^{\frac{p-1}{3^2}})^k \bmod p \equiv (g^{\frac{p-1}{3^2}})(g^{\frac{p-1}{3}})^{k'} \bmod p.$$

- Discrete log problem with the SAME base $g^{\frac{p-1}{3}}$ - so we can just use previous calculations.

Pohlig-Hellman

What if $p-1$ is has a large prime power?We can refine the process for 3^2 as follows.

- We want to find $k \bmod 3^2$.First find $k \bmod 3$ as before - we got $k \equiv 1 \bmod 3$.
- Write $k = 1 + 3k'$, where k' is 0,1 or 2 mod 3.Note that

$$9689^{\frac{p-1}{3^2}} \equiv (g^{\frac{p-1}{3^2}})^k \bmod p \equiv (g^{\frac{p-1}{3^2}})(g^{\frac{p-1}{3}})^{k'} \bmod p.$$

- Discrete log problem with the SAME base $g^{\frac{p-1}{3}}$ - so we can just use previous calculations.
- One can check that $k' = 1$ in this case.

Pohlig-Hellman

What if $p-1$ is has a large prime power?We can refine the process for 3^2 as follows.

- We want to find $k \mod 3^2$.First find $k \mod 3$ as before - we got $k \equiv 1 \mod 3$.
- Write $k = 1 + 3k'$, where k' is 0,1 or 2 mod 3.Note that

$$9689^{\frac{p-1}{3^2}} \equiv (g^{\frac{p-1}{3^2}})^k \mod p \equiv (g^{\frac{p-1}{3^2}})(g^{\frac{p-1}{3}})^{k'} \mod p.$$

- Discrete log problem with the SAME base $g^{\frac{p-1}{3}}$ - so we can just use previous calculations.
- One can check that $k' = 1$ in this case.Therefore $k \equiv 1 + 3k' \mod 3^2 \equiv 4 \mod 3^2$.

Pohlig-Hellman

We will skip the run time analysis for Pohlig-Hellman, but to summarize,

Pohlig-Hellman

We will skip the run time analysis for Pohlig-Hellman, but to summarize,

- Baby-step-giant-step is a general approach that works for ANY prime p, and is a $O(p^{1/2+\epsilon})$ algorithm. (Subexponential, but not polynomial)

Pohlig-Hellman

We will skip the run time analysis for Pohlig-Hellman, but to summarize,

- Baby-step-giant-step is a general approach that works for ANY prime p, and is a $O(p^{1/2+\epsilon})$ algorithm. (Subexponential, but not polynomial)
- Pohlig-Hellman is an approach that depends on the factorization of $p-1$.

Pohlig-Hellman

We will skip the run time analysis for Pohlig-Hellman, but to summarize,

- Baby-step-giant-step is a general approach that works for ANY prime p, and is a $O(p^{1/2+\epsilon})$ algorithm. (Subexponential, but not polynomial)
- Pohlig-Hellman is an approach that depends on the factorization of $p-1$.
- In particular, if $p-1$ has many small prime factors, Pohlig-Hellman would be much better.

Pohlig-Hellman

We will skip the run time analysis for Pohlig-Hellman, but to summarize,

- Baby-step-giant-step is a general approach that works for ANY prime p, and is a $O(p^{1/2+\epsilon})$ algorithm. (Subexponential, but not polynomial)
- Pohlig-Hellman is an approach that depends on the factorization of $p-1$.
- In particular, if $p-1$ has many small prime factors, Pohlig-Hellman would be much better.
- For example, if $p-1=2^k$ for some k (Fermat prime), then the running time for Pohlig-Hellman would be $O(k)=O(\log p)$, i.e. polynomial time!

Pohlig-Hellman

We will skip the run time analysis for Pohlig-Hellman, but to summarize,

- Baby-step-giant-step is a general approach that works for ANY prime p, and is a $O(p^{1/2+\epsilon})$ algorithm. (Subexponential, but not polynomial)
- Pohlig-Hellman is an approach that depends on the factorization of $p-1$.
- In particular, if $p-1$ has many small prime factors, Pohlig-Hellman would be much better.
- For example, if $p-1=2^k$ for some k (Fermat prime), then the running time for Pohlig-Hellman would be $O(k)=O(\log p)$, i.e. polynomial time!This shows that one has to be careful in choosing the prime p.