Fermat Primality Test

for any natural number \( n \), we may ask:

Q1: Is \( n \) a prime?

Q2: How to factorize \( n \), if the answer to Q1 is no?

Q3: Is there an algorithm to answer Q1 and Q2 within a reasonable amount of time?

Naive Way:

Ex. 1: \( 2^{607} - 1 \) is \( \square \) a prime.

Suppose you have \( 10^6 \) CPUs, each run at clock rate 3 GHz. Also suppose 1 CPU clock cycle can do a

perform one divisibility test. Then, if you perform the

primality test in the naive way, then it would take you

\[
\sqrt{2^{607} - 1} \div (10^6 \times 3 \times 10^9) \approx 10^{73} \text{ Seconds}
\]

1 year \( \approx 3.15 \times 10^7 \) seconds \( \approx 10^{68} \) years.
Contrapositive of Fermat's Little Theorem.

If \( a^{n-1} \equiv 1 \pmod{n} \) for some \( a \) relatively prime to \( n \),
then \( n \) is not a prime.

However, some composite number will pretend to be a prime from the above point of view.

**Ex 2**

Recall that in HW #5, we have shown that

\[
341 = 11 \times 31 \quad \text{is not a prime}
\]

but

\[
340 \equiv 1 \pmod{341}
\]

Any \( a \) such that \( a^{n-1} \equiv 1 \pmod{n} \) when \( n \) is composite is known as a Fermat liar. If we do pick an a such that \( a^{n-1} \equiv 1 \pmod{n} \) then \( a \) is known as a Fermat witness for the compositeness of \( n \).

Ex 2 shows that \( 2 \) is a Fermat liar for \( 341 \).
Cryptography—an induction

From last lecture, we have learned that

1. It is very hard to factorize a random large number without knowing any priori information.

2. It is relatively easy to determine primality via various primality tests.

Based on 1, we may build a secure communication system.

Suppose that Alice wants to send a secured message to Bob.

\[\text{Alice} \xrightarrow{\text{encryption}} \text{cyphertext} \xrightarrow{\text{decryption}} \text{Bob}\]

\[\text{Recover original message}\]

- **Q.** Everyone can potentially read the cyphertext.
- **Q.** Our goal is to make sure that only Bob (the intended recipient) can decrypt the message.
The RSA Algorithm

Fix two large primes \( p \) and \( q \), and compute their product \( N = pq \). Choose a number \( e \) (known as the public exponent), such that

\[
1 \leq e < n, \quad \gcd(e, (p-1)(q-1)) = 1. \quad \Phi(pq)
\]

Next find the inverse of \( e \) modulo \((p-1)(q-1)\) with

\[
ed \equiv 1 \pmod{(p-1)(q-1)}
\]

\((n, e)\) — public key (accessible to anyone)

\((n, d)\) — private key (kept secret)

From this point on, \( p \) and \( q \) are no longer needed but need to be kept secret.

The security of the system is based on the fact that it is very difficult to compute \( d \) given \( e \) and \( n \), unless you know \( p \) and \( q \) (i.e., factorizing \( n \)). In general, knowing \( \Phi(pq) \) is insufficient to work out \((p-1)(q-1)\). A fast method to factorize large numbers would render this method insecure and would revolutionize the whole industry.
Suppose that Alice wants to send an integer \( m < n \) to Bob. Alice converts \( m \) into ciphertext \( C \) by computing
\[
C \equiv m^e \pmod{n}
\]
where the pair \( (n, e) \) is Bob's public key. She then sends the number \( C \) to Bob over an open channel.

When Bob receives the re-encrypted message \( C \), he uses his private key \( (n, d) \) to compute
\[
C^d \equiv (m^e)^d \pmod{n} \equiv m \pmod{n}
\]
(Euler's thm)

and \( de \equiv 1 \pmod{\phi(pq)} \)

* Anyone other than Bob will only have access to the public key \( (n, e) \). In order to recover \( m \), they would have to somehow compute \( d \), which involves factoring \( n \).
Digital signatures

Suppose now that Alice wants to send a message m to Bob in such a way that Bob is assured that the message has not been substituted by another message, and that the message comes from Alice and not from someone else. These problems are real because the public key allows anyone to generate messages that appear authentic. In order to do this, both Alice and Bob create public and private keys as above.

Alice creates a digital signature s by computing

\[ S_A = m^{d_A} \pmod{n_A} \]

where \((n_A, d_A)\) is Alice's private key. She sends both m and \(S_A\) (using the above system) to Bob.

To verify the signature, Bob checks that the message m is recovered by

\[ m = S_A^{e_A} \pmod{n_A}, \]

where \((n_A, e_A)\) is Alice's public key. Anyone else who doesn't have Alice's private key would not pass this verification process.

Ex. Explain what happens if either the encrypted message or the signature is tampered with.
Key facts about RSA algorithm

1. Public key is accessible to anyone and is used to identity each recipient.

2. Bob and Alice don't need to know each other's private key for secure communication. Only the intended recipient's public key is needed to generate the cyphertext which can only be decrypted by the intended recipient.