Bob needs to:
- find 2 large primes \( p, q \leftarrow ? \)
- find \( e \) s.t. \( \gcd(e, \phi(pq)) = 1 \)

Good news:
- primes are fairly common:
  - there are about \( \frac{N}{\ln N} \) primes \( \leq N \)

Exercise:
If looking for a 512-bit prime, how many randomly generated numbers need to try?

\[
\text{#primes} = \frac{2^{512}}{\ln 2^{512}}
\]

Fraction of primes:
\[
\frac{1}{\ln 2^{512}} = \frac{1}{512 \cdot \ln 2}
\]

Need to try:
\[
512 \cdot \ln 2 \approx 355
\]
RSA Parameter Generation

We need to decide:
Given a number \( x \), how to determine if \( x \) is a prime?

```
for i = 2 to \( \sqrt{x} \):
    if i divides \( x \), then output NOT PRIME and exit
output PRIME
```

What is the running time?

\[
O(\sqrt{x}) = O\left((2^2)^\frac{1}{2}x\right)
\]

Suppose \( x \) is 512 bits

\[
x < 2^{512}
\]

\[
\sqrt{x} < 2^{256}
\]

Do not have the time to execute
Until recently, no (deterministic) poly-time algorithm for primality testing.

In 2002, Agrawal, Kayal, and Saxena: Primality testing is in P !!!
Good news: there is a faster approach using randomization.

First, some terminology:

A **yes-biased Monte Carlo algorithm** is a randomized algorithm that:
- if the algo says YES, then the answer is correct
- if the algo says NO, then the answer might be incorrect, but this happens with a small probability

More precisely, there is a (small) error probability $\epsilon > 0$ s.t. for any “yes” instance, the algo says NO with probability $\leq \epsilon$ (considering all possible random choices of the algo).
Good news: there is a faster approach using randomization (yes-biased Monte Carlo algorithm to determine if an input number is composite)

First, some terminology:

A yes-biased Monte Carlo algorithm is a randomized algorithm that:
- if the algo says YES, then the answer is correct
- if the algo says NO, then the answer might be incorrect, but this happens with a small probability

More precisely, there is a (small) error probability $\epsilon > 0$ s.t. for any “yes” instance, the algo says NO with probability $\leq \epsilon$ (considering all possible random choices of the algo).
Fermat's Little Theorem (pg 79):
If $p$ is a prime, then $a^{p-1} \equiv 1 \pmod{p}$ for all $a \in \mathbb{Z}_p \backslash \{0\}$

PseudoPrime($x$):
1. Choose random $a$, $1 \leq a \leq x - 1$.
2. if $a^{x-1} \equiv 1 \pmod{x}$:
3. return prime
4. else
5. return composite

Is this a yes-biased Monte Carlo algorithm? ✓
For primes? For **composites**?
Polynomial-time? ✓
Primality Testing – randomized attempt 1

Problem:
There are composite numbers for which the Fermat’s Little Theorem holds.
(A composite number $x$ is a **Carmichael number** if $a^{x-1} \equiv 1 \pmod{x}$, for every $a \in \mathbb{Z}_x-{0}$)

Good news:
Carmichael numbers are very rare: only 255 Carmichael numbers smaller than $10^9$ (the first three are 561, 1105, and 1729).

Bad news:
What is $\epsilon$ for our algo from the previous slide?
Miller-Rabin(x):
1. Find k, m such that \( x-1 = 2^k m \), where m is odd
2. Choose random a, \( 1 \leq a \leq x-1 \)
3. Let \( b = a^m \mod x \)
4. if \( b \equiv 1 \pmod{x} \): return prime
5. for i=0 to k-1:
   6. if \( b \equiv -1 \pmod{x} \): return prime
   7. else: \( b = b^2 \mod x \)
8. return composite

This is a polynomial-time yes-biased Monte Carlo algorithm that tests whether \( x \) is composite. Why?

Note: \( \epsilon \leq \frac{1}{4} \) (we will not prove this)
Miller-Rabin

Miller-Rabin(x):
1. Find k,m such that x−1 = 2^km, where m is odd
2. Choose random a, 1 ≤ a ≤ x−1
3. Let b = a^m mod x
4. if b ≡ 1 (mod x): return prime
5. for i=0 to k-1:
6. if b ≡ -1 (mod x): return prime
7. else: b = b^2 mod x
8. return composite

This is a polynomial-time yes-biased Monte Carlo algorithm that tests whether x is composite. Why?

Note: ε ≤ \frac{1}{4} (we will not prove this)
RSA Questions

- Eve can compute the e-th root modulo n to decrypt... The catch: computing roots mod n as hard as factoring!

- If Bob chooses p,q but one of them will not be a prime, will RSA still work?

- Can Eve precompute all products of 512-bit primes, to have a table (and factorization) of all possible n?