CS265 Class 3

Primality-Testing and the Miller-Rabin Test

If you don’t have a nametag or forgot to bring yours back, please make a new one!
Markers and paper are up front.

Also, if you want a paper agenda, come grab one from the front of the room!

1 Warm-Up

Let $S_r = \{x \in \mathbb{Z}_n^* : x^r = \pm 1 \mod n\}$. Is S_r a group?

If you don’t have a nametag or forgot to bring yours back, please make a new one!
Markers and paper are up front.

Also, if you want a paper agenda, come grab one from the front of the room!

Announcements

- HW1 Due Friday!
- Check Ed for some resources on big-Oh notation
- Also check Ed for a quantitatively betters solution to Wednesday's group work!
  - Thanks Shurui and Ruochuan!

Recall: Primality testing

- **Goal:** Given n, is it prime?

- Today we'll see a pretty simple randomized algorithm.
- It was open for a long time to give a deterministic algorithm.
  - Finally solved in 2002 by Agrawal, Kayal, and Saxena.
  - We won't talk about the deterministic algorithm in class: see links on website if you want to learn about it on your own.

When we last left our heroes…

- We saw **Fermat's test**.
  - Check if $x^{n-1} = 1$ for a random x
    - If so, say n is prime; if not, say it's not.
- We proved that it mostly worked:
  - For lots of numbers n (all those that are not Carmichael numbers):
    - If n is prime, Fermat's test will always say so.
    - If n is not prime, Fermat's test will say "not prime" with probability at least $\frac{1}{2}$.

<u>Goal for today:</u>

- Can we come up with a test that gives this guarantee for *all* n?

But first, questions?

1 Warm-Up

Let $S_r = \{x \in \mathbb{Z}_n^* : x^r = \pm 1 \mod n\}$. Is S_r a group?

- Warm-up, quiz, mini-lectures?

- Also, can you see the solutions to the quiz now?
  - Both last week and today?

Warm-up

1 Warm-Up

Let $S_r = \{x \in \mathbb{Z}_n^* : x^r = \pm 1 \mod n\}$. Is S_r a group?

- Yes, this is a group!
  - Since $\mathbb{Z}_n^*$ is a group (and $1 \in S_r$), we have identity and associativity.
  - We just need to establish that it's closed under multiplication and inverses.
- Closure under multiplication:
  - if $x^r = \pm 1$ and $y^r = \pm 1$, then $(xy)^r = x^r y^r = (\pm 1)(\pm 1) = \pm 1$
- Inverses:
  - Let $x \in S_r$ and let $y = x^{-1} \in \mathbb{Z}_n^*$. Want to show $y \in S_r$.
  - $xy = 1$
  - $x^r y^r = 1$
  - $(\pm 1) y^r = 1$
  - $y^r = \pm 1$
  - So $y \in S_r$ ☺

Plan for today

- We will develop the Miller-Rabin algorithm for primality testing.
- We will mostly* prove that it works.

Compare to Fermat's test, where we proved that it mostly works.

*We will skip over stuff that's interesting from an algebra perspective but not so interesting from a "randomized algorithms and probabilistic analysis" perspective. Details in lecture notes if you are curious.

How many square roots of 1 in $\mathbb{Z}_n^*$?

- That is, how many $x \in \mathbb{Z}_n^*$ have $x^2 = 1$?

Why are we only looking at the numbers 1,2,4,7,8,11,13,14?

- Example: $n = 7$

$1^2 = 1$
$2^2 = 4$
$3^2 = 2$
$4^2 = 2$
$5^2 = 4$
$6^2 = 1$

$1^2 = 1$

$(-1)^2 = 1$

± 1 are the only square roots of 1

- Example: $n = 15$

$1^2 = 1$
$2^2 = 4$
$4^2 = 1$
$7^2 = 4$
$8^2 = 4$
$11^2 = 1$
$13^2 = 4$
$14^2 = 1$

There are more than two square roots of 1

Idea of the Miller-Rabin Algorithm

- If n is prime, then there are exactly two square roots of 1 in $\mathbb{Z}_n^*$, $+1$ and -1.
  - Assume that this is true for now!
  - We may come back to it later if time
  - (If we don't have time, it's in the lecture notes, or try yourself!)
- If n is an odd composite number that is not the power of a prime, then there are **more** than two square roots of 1 in $\mathbb{Z}_n^*$.
- The Miller-Rabin Algorithm gives a clever way to find some $x \neq \pm 1$ so that $x^2 = 1$ when one exists.

Here is a way to generate a list of numbers

Why? We'll see soon…

- Suppose that n is odd, and not a power of a prime.
- Find k and m so that $n - 1 = 2^k m$ where m is odd.
- Choose $x \in \{1, \dots, n-1\}$ uniformly at random, and compute the list:

$$x^m, x^{2m}, x^{2^2 m}, \dots, x^{2^k m} \mod n$$

Here is a way to generate a list of numbers

Why? We'll see soon…

- Suppose that n is odd, and not a power of a prime.
- Find k and m so that $n-1 = 2^k m$ where m is odd.
- Choose $x \in \{1, \dots, n-1\}$ uniformly at random, and compute the list:

$$x^m, x^{2m}, x^{2^2 m}, \dots, x^{2^k m} \mod n$$

Example

- Say $n = 21$
- Then $n - 1 = 2^2 \cdot 5$, so $k = 2, m = 5$
- Choose $x = 3$.
- The list is:
  - $3^5 = 12 \mod 21$,
  - $3^{10} = 18 \mod 21$,
  - $3^{20} = 9 \mod 21$.

Group work

GenerateSomeNumbers:

- Write $n - 1 = 2^k \cdot m$ where m is odd.
- Choose $x \in \{1, \ldots, n-1\}$ uniformly at random.
- Consider $x^m, x^{2m}, x^{2^2 m}, \ldots, x^{2^k m} \mod n$.

1. Mess around with web.stanford.edu/~marykw/CS265Class3.html
2. Say n is prime. What is the last number generated by this list?
3. Say n is prime. What are the options for the second-to-last number?
4. Say n=561 (the first Carmichael number). Consider x=23,13,63,458. What list of numbers to do you get for each? Does this list prove to you that n is not prime?
5. If time, come up with (and analyze) an algorithm for primality testing.

Solutions to group work

- If n is prime, the last number generated should be 1.
- If n is prime, the second-to-last number should be ± 1.
- If n=561,
  - x=23 → [386, 331, 166, 67, 1]. Not prime! $(67)^2 = 1$
  - x=13 → [208, 67, 1, 1, 1]. Not prime! $(67)^2 = 1$
  - x=63 → [351, 342, 276, 441, 375]. Not prime! $(63)^{n-1} = 375 \neq 1$
  - x=458 → [560, 1, 1, 1, 1]. Could be prime…

Miller-Rabin Algorithm

How do we check for these efficiently?

- Given an input n:
  - If n is even, or the power of a prime, output **Composite!**
  - Choose a random x and find the special list of numbers.
  - Make sure it looks like [*,*,*,-1,1,1,1,1] or [1,1,1,1,1,1,1,1]:
    - If the last thing isn't 1, output **Composite!** n fails Fermat's test!
    - If there's ever (not ± 1) followed by 1, output **Composite!**

There's some number other than ± 1 that squares to 1 mod n

  - Output **Prime!**

Theorem

- If n is prime, the Miller-Rabin Test will always output “Prime!”
- If n is composite, the Miller-Rabin Test will output “Composite!” with probability at least ½.
- The running time is polynomial in $\log n$

When n is composite

and odd and not a prime power

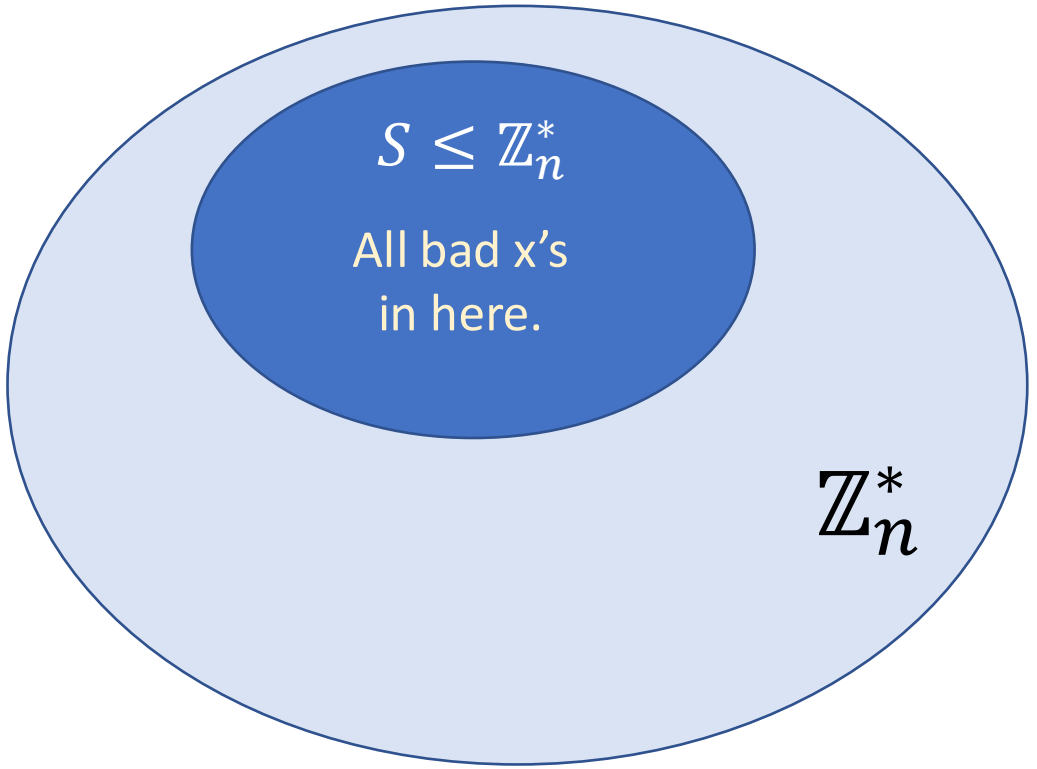
Good x:

x → Algorithm running on n → **Composite!**

Bad x:

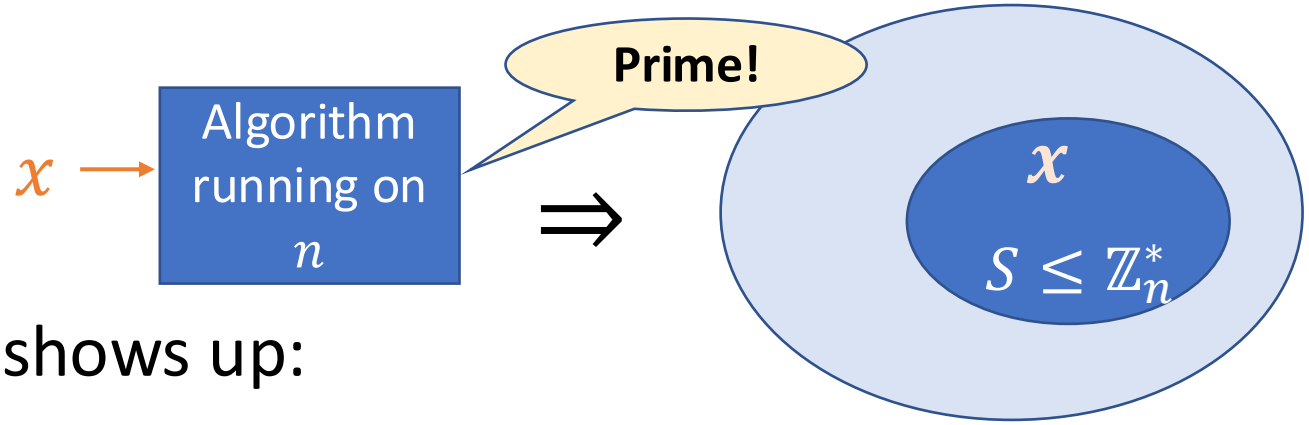
x → Algorithm running on n → **Prime!**

- We will define a proper subgroup $S \leq \mathbb{Z}_n^*$ that contains all of the bad x's.

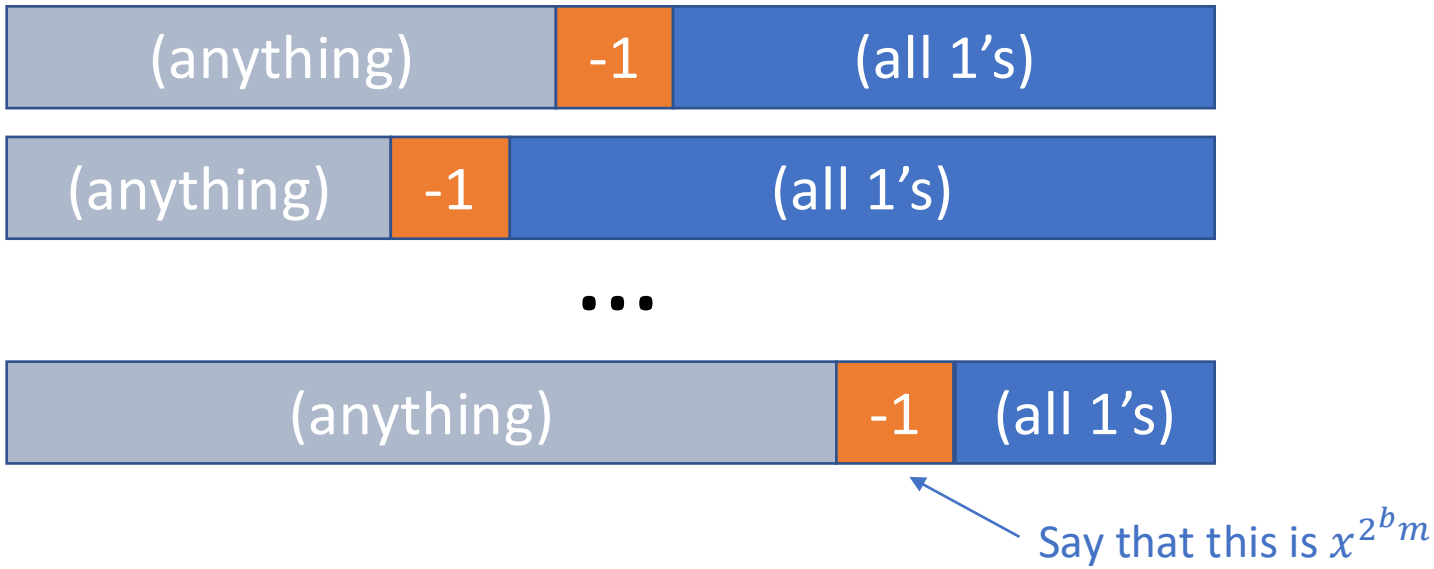


- By Lagrange's theorem, $|S| \leq \frac{1}{2}|\mathbb{Z}_n^*|$
- The algorithm succeeds with probability $\geq \frac{1}{2}$.

Defining S



- Consider all the $x \in \mathbb{Z}_n^*$ where -1 shows up:



- Let b be the **largest** value of i less than k so that there exists an $x \in \mathbb{Z}_n^*$ so that $x^{2^i m} = -1 \bmod n$.
- Define $S = \left\{ y \in \mathbb{Z}_n^* : y^{2^b m} = \pm 1 \right\}$

Claim 1

- For any x so that the algorithm says "Prime!", $x \in S$

If $x \in \mathbb{Z}_n^*$, then…

✓ $x \in S$, by definition of b and the fact that all such x's look like this →

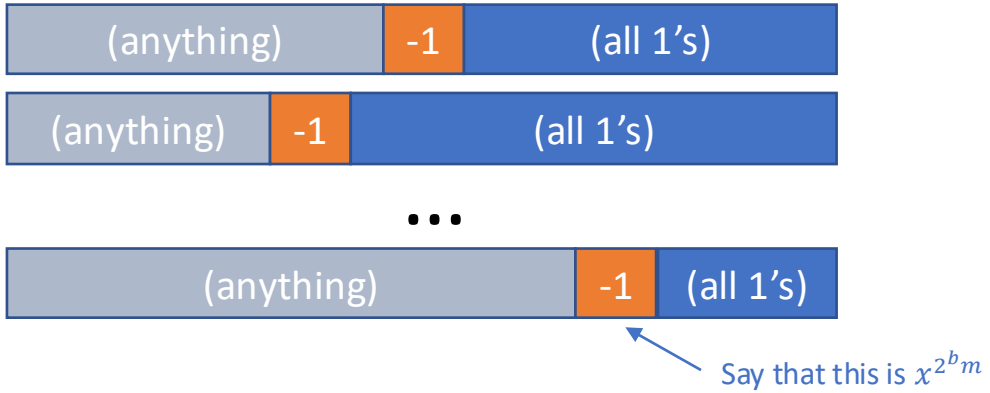
If $x \notin \mathbb{Z}_n^*$, then…

…the algorithm won't say "Prime"

$x^{n-1} \neq 1$, so fails Fermat's test.

Otherwise $x^{n-2} \cdot x = 1$, but supposedly x doesn't have an inverse

- Consider all the $x \in \mathbb{Z}_n^*$ where -1 shows up:



- Let b be the **largest** value of i less than k so that there exists an $x \in \mathbb{Z}_n^*$ so that $x^{2^i m} = -1 \bmod n$.
- Define $S = \left\{ y \in \mathbb{Z}_n^* : y^{2^b m} = \pm 1 \right\}$

Claim 2

- S is a subgroup of $\mathbb{Z}_n^*$.

This was the warm-up!

Claim 3

- If n is odd, composite, and not a prime power, then $S \neq \mathbb{Z}_n^*$

Fun exercise if you have time!

$$S = \left\{ y \in \mathbb{Z}_n^* : y^{2^b m} = \pm 1 \right\}$$

Three claims:

- Claim 1: For any x so that the algorithm says "Prime!", $x \in S$
- Claim 2: S is a subgroup of $\mathbb{Z}_n^*$.
- Claim 3: If n is odd, composite, and not a prime power, $S \neq \mathbb{Z}_n^*$

Group work:

Claims 1, 2 and 3 imply that the algorithm works with probability at least ½.

- Claim 1: For any x so that the algorithm says "Prime!", $x \in S$
- Claim 2: S is a subgroup of $\mathbb{Z}_n^*$.
- Claim 3: If n is odd, composite, and not a prime power, $S \neq \mathbb{Z}_n^*$

- If n is odd, composite, not a prime power, then by Claims 2 and 3, $\mathrm{S} < \mathbb{Z}_n^*$ is a proper subgroup.
- That means that $|S| \leq \frac{|\mathbb{Z}_n^*|}{2}$ by Lagrange's theorem.
- $\Pr[\text{test says } n \text{ is prime}] \leq \Pr[\text{test picks } x \in S] \leq \frac{1}{2}$

By Claim 1

By the above

Theorem: for <u>any</u> n,

- If n is prime, the Miller-Rabin Test will always output “Prime!”
- If n is composite, the Miller-Rabin Test will output “Composite!” with probability at least ½.
- The running time is polynomial in $\log n$

Modulo the missing proposition…

Modulo proof of Claim 3…

$n - 1 = 2^k m$ where m is odd

Running time

- If n is even, or the power of a prime, output **Composite!**
- Choose random x and find
  $x^m, x^{2m}, \ldots x^{2^k m}$
- Make sure it looks like [*,*,*,-1,1,1,1,1] or [1,1,1,1,1,1,1,1]:
  - If the last thing isn't 1, output **Composite!**
  - If there's ever (not ± 1) followed by 1, output **Composite!**
- Output **Prime!**

Time $\text{poly}(\log(n))$

To see if $n = a^k$ for a fixed k, binary search for an appropriate a between 1 and n. Then try all $k = 2, \ldots, \log n$ to see if n is a perfect kth power. (If it is, we don't really care if it's a prime power or not…)

- Compute x^m by repeated squaring
  - $O(\log m) = O(\log n)$ multiplications
- Square x^m $k = O(\log n)$ times

- Need to check $k = O(\log n)$ things

Total time $\text{poly}(\log n)$

Theorem: for <u>any</u> n,

- If n is prime, the Miller-Rabin Test will always output “Prime!”
- If n is composite, the Miller-Rabin Test will output “Composite!” with probability at least ½.
- The running time is polynomial in $\log n$

Modulo the missing proposition…

Modulo proof of Claim 3…

Recap

The Miller-Rabin Primality Test

- Cleverly generate a list of numbers for a random x.
- If n isn't prime, probably these numbers will give it away.

- The point: There is a fast and not-too-hard randomized algorithm for primality testing!

- It was open for a long time to get a deterministic algorithm
  - (And it isn't that simple).

Next time! Markov and Chebyshev!

- Before next time:
  - Watch the videos/read the lecture notes
  - Do the quiz

Missing piece 1:

Square Roots of 1 in $\mathbb{Z}_n^*$ when n is prime

- Suppose that $x^2 \equiv 1$ mod n.
- Then $(x-1)(x+1) \equiv 0$ mod n
- That means that n divides $(x-1)(x+1)$
- Since n is prime, n must divide either $x-1$ or $x+1$
  - Example:
    - 3 (prime) divides 6×4, and that implies that 3 divides one of 6 or 4.
    - OTOH 6 (not prime) divides 3×8, but it doesn't divide either 3 or 8.
- But then either $x \equiv 1$ mod n or $x \equiv -1$ mod n.

- Let b be the **largest** value of i less than k so that there exists an x with $x^{2^i m} = -1 \bmod n$.
- $S = \left\{ y \in \mathbb{Z}_n^* : y^{2^b m} = \pm 1 \right\}$

Missing piece 2:

Proof of Claim 3

If n is odd, composite, and not a prime power, $S \neq \mathbb{Z}_n^*$

- Let $r = 2^b m$ for convenience.
- Write $n = s \cdot t$ where $s, t > 1$ are relatively prime.

We can do this since n is composite, and not a prime power.

- Suppose $x \in \mathbb{Z}_n^*$ is such that $x^r = -1$.

This x exists by the definition of r and b

- Claim: There exists a $y \in \mathbb{Z}_n$ so that $y = x \bmod s$, and $y = 1 \bmod t$

This follows from the Chinese Remainder Theorem

- Claim: $y \in \mathbb{Z}_n^*$

By def of $\mathbb{Z}_n^*$, $\gcd(x, n) = 1$
Then $\gcd(x, s) = 1$, since $s|n$.
But then we also have $\gcd(y, s) = 1$, since $y \equiv x \bmod s$.

- Claim: $y \notin S_r$

$y^r = 1 \bmod t$, and $y^r = x^r = -1 \bmod s$.
Case 1: $y^r = -1 \bmod n \Rightarrow y^r = -1 \bmod t$. (Since $t|n$)
Case 2: $y^r = 1 \bmod n \Rightarrow y^r = 1 \bmod s$. (Since $s|n$)
Either way it's a contradiction.