Welcome to CS265/CME309!

Randomized Algorithms and Probabilistic Analysis

Please make a nametag while we wait for class to get started!

Welcome to CS265/CME309!

- Instructor:
  - Mary Wootters
- CAs:
  - Dorsa Fathollahi
  - Spencer Compton

Mary

Dorsa

Spencer

Who are you?

- Now’s a good time to try out PollEverywhere!

PollEv.com/marykw

Which best describes you?

0

(A) Freshman/Sophomore

0

(B) Junior/Senior

0

(C) MS student

0

(D) PhD student

0

(E) NDO or Other

0

Why are you taking this class?

0

I like probability theory

0

I like algorithms

0

I think it will be useful for my major/work/research

0

I have to take this class for my major/program

0

You tell me...I'm just shopping!

0

Other

0

CS265/CME309 Course Overview

Course Information

web.stanford.edu/class/cs265

Course Pitch

Randomized Algorithms and Probabilistic Analysis

- Randomness can help in computation!
  - There are many problems where the best solutions known are randomized.
  - There are many problems where the best solutions **possible** are randomized.

- In this class, we will:
  - Learn techniques for analyzing randomized processes and structures.
    - Concentration inequalities, Poissonization, Probabilistic Method, Markov Chains, Martingales, ...
  - Learn to design random structures and algorithms to solve problems.
    - Topics may include: Graph Algorithms, Scheduling, Routing, Dimension Reduction, Computational Geometry, Constraint Satisfaction, Sketching & Streaming, Counting & Sampling, …

Course Structure

The course website tells you what you are supposed to watch/read/do before each class.

- This is a **flipped class.**
- **Before class:** (except this one)
  - You watch mini-lecture videos and/or read lecture notes.
    - Videos are available either on YouTube or on Canvas
  - You take a short (open-book, untimed) quiz, due BEFORE each class.
    - **FIRST ONE IS DUE WEDNESDAY! Find it on Gradescope.**
- **During class:**
  - We'll answer any questions from the Before-Class material.
  - We'll practice/develop the material, mostly via group work.
  - (There might be a few short bits of lecture as needed).
- **Homework:**
  - Weekly homework, done in small teams.
    - Find teammates in class, on Ed, in OH, …
  - First one is due a week from Friday.
- **Exams:**
  - Midterm: Wednesday Feb 11, 9am-11:50am
  - Final: Monday March 16, 3:30pm-6:30pm
  - **If you cannot make an exam, email the course staff list ASAP!!!!**

This first class will have more lecture, since you didn't have time to watch videos beforehand!

A few policies

See course website for complete list

- **Grading policy**
  - See website for breakdown
  - Two schemes: one with some weight on attendance, one where that weight is shifted to exams.
    - We will automatically choose which one is best for your grade.
    - Please **do not come to class if you are sick**. You can skip 5 classes with no penalty; this skips are meant for illness and other emergencies.

- **Exam Conflicts**
  - Let us know ASAP if you have a conflict with either exam!! (**Before** the add/drop deadline, if possible!)

Logistics Recap

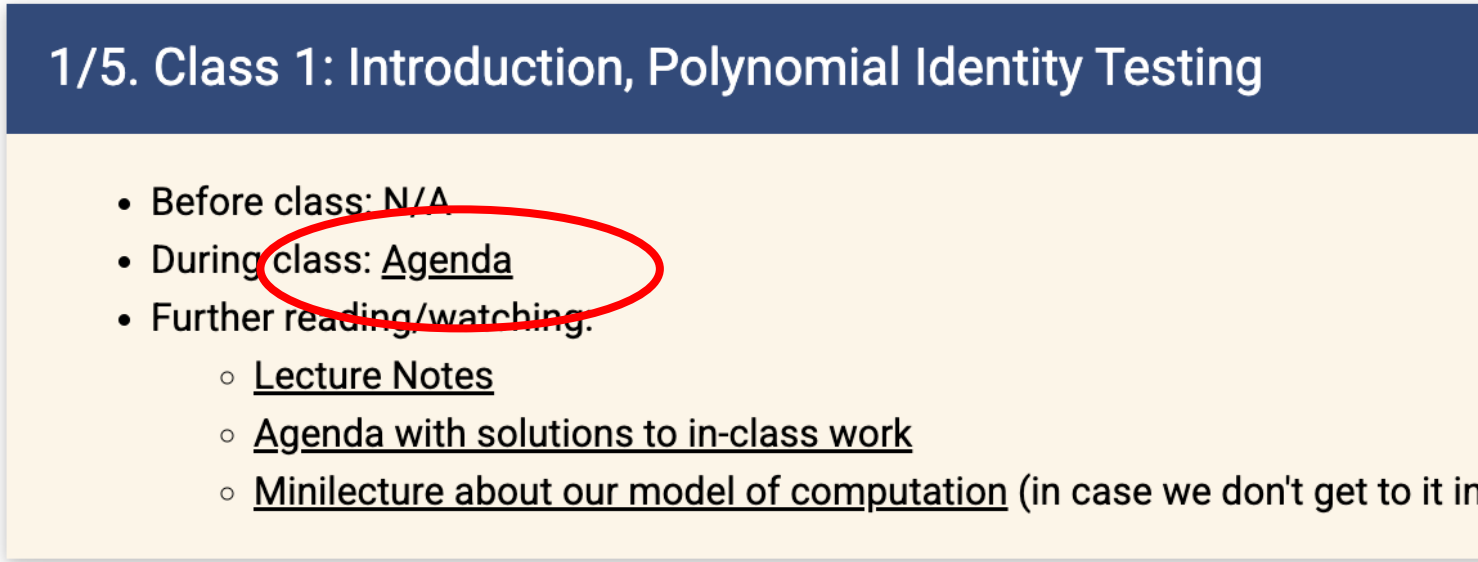
web.stanford.edu/class/cs265

Let’s get started!

web.stanford.edu/class/cs265

Either access the agenda on your preferred device now or else grab a print-out.

If you scroll down or click to “Class-by-class resources” …



Identically zero polynomials

- (Multivariate) polynomials:

$$\text{Eg., } f(x_1, x_2) = (x_1 + 1)(x_2 + 1) - 1 - x_1 x_2 - (x_1 + x_2)$$

- A polynomial $f(x_1, x_2, \ldots, x_n)$ is **identically zero** if when you simplify it, you get zero.
  - The one above is identically zero.

While you work, we'll wander around, check in, and also take attendance!

Group work time!

Start on all these questions!

2 Polynomial Identity Testing

2.1 Group work

Group Work

1. First, introduce yourselves to each other. What year/program are you in? What class/activity/etc are you most excited about for this quarter?
2. Read the following definition.

   A multivariate polynomial $f(x_1, \ldots, x_m)$ is *identically zero* if all its coefficients are zero. For example, the polynomial $f(x_1, x_2) = (x_1+1)(x_2+1) - 1 - x_1x_2 - (x_1+x_2)$ is identically zero because when you expand it out, all of the terms cancel.

   Now, work on the following questions with your group.
3. Which of the following two polynomials are identically zero?

$$f(x,y) = (x-y)^2 + 2xy + (x+y)^3 - y(3x^2 + y(3x+y+1)) - (x+1)x^2$$

$$g(x,y) = (x-2y)^2 + xy + (x+y)^3 - y(3x^2 + y(3x+y+1)) - (x+1)x^2$$

Anyone not register their attendance?

- Find Dorsa or Spencer before/at the end of class if we missed you

While you work, we'll wander around, check in, and also take attendance!

Group work time!

Start on all these questions!

2 Polynomial Identity Testing

2.1 Group work

Group Work

1. First, introduce yourselves to each other. What year/program are you in? What class/activity/etc are you most excited about for this quarter?
2. Read the following definition.

   A multivariate polynomial $f(x_1, \ldots, x_m)$ is *identically zero* if all its coefficients are zero. For example, the polynomial $f(x_1, x_2) = (x_1+1)(x_2+1) - 1 - x_1x_2 - (x_1+x_2)$ is identically zero because when you expand it out, all of the terms cancel.

   Now, work on the following questions with your group.
3. Which of the following two polynomials are identically zero?

$$f(x,y) = (x-y)^2 + 2xy + (x+y)^3 - y(3x^2 + y(3x+y+1)) - (x+1)x^2$$

$$g(x,y) = (x-2y)^2 + xy + (x+y)^3 - y(3x^2 + y(3x+y+1)) - (x+1)x^2$$

Which is identically zero?

f(x,y) 0%

g(x,y) 0%

Both 0%

Neither 0%

How long does the straightforward algorithm take, in the worst case?

- $O(n)$
- $\text{poly}(n)$
- $2^{\Omega(n)}$
- None of the above
- Not sure

How long does the straightforward algorithm take, in the worst case?

$O(n)$ 0%

$\text{poly}(n)$ 0%

$2^{\Omega(n)}$ 0%

None of the above 0%

Not sure 0%

How long does the straightforward algorithm take, in the worst case?

$O(n)$ 0%

$\text{poly}(n)$ 0%

$2^{\Omega(n)}$ 0%

None of the above 0%

Not sure 0%

Solutions (pre-challenge)

- Question 3:
  - f(x,y) is identically zero.
- Question 4:
  - This takes time $2^{\Omega(n)}$
  - There are $2^{\Omega(n)}$ possible monomials, so even writing them all down takes that much time.
- Questions 5,6,7…let's see how well your strategies work!

Challenge time!!

Which of the following is identically zero?

0

$$(x+y)^5 - x^3(3y+x^2) - y(5x-3)x^3 \\ -10y^2x^3 - y^3(10x^2 + y(5x+y))$$

(A)

$$xy(x+y)^3 - 3x^2(y^3 + y^2(x-2y) + 3y((x+y)^2 - x^2))$$

(B)

Which of the following is identically zero?

0

(A) $(x+y)^5 - x^3(3y+x^2) - y(5x-3)x^3$
$-10y^2x^3 - y^3(10x^2 + y(5x+y))$

0

(B) $xy(x+y)^3 - 3x^2(y^3 + y^2(x-2y) + 3y((x+y)^2 - x^2))$

0

Which of the following is identically zero?

0

(A) $(x+y)^5 - x^3(3y+x^2) - y(5x-3)x^3$
$-10y^2x^3 - y^3(10x^2 + y(5x+y))$

0

(B) $xy(x+y)^3 - 3x^2(y^3 + y^2(x-2y) + 3y((x+y)^2 - x^2))$

0

Give a short description of your algorithm.

0

Nobody has responded yet.

Hang tight! Responses are coming in.

Solutions (post-challenge)

- f(x,y) is identically zero, g(x,y) is not.
- There are many good strategies! The one we had in mind was:
  - Choose a random point (x,y). Check if f(x,y) = 0.

- Detail: how do we choose a random point? What distribution should we use? And does it matter if we only have finite precision?
  - Choose a uniformly random point from some pre-defined set.
  - How well does this work?

Polynomial Identity Testing

Polynomial Identity Testing (PIT)

- Given a polynomial $P(x_1, x_2, \ldots, x_n)$, is P identically zero?

We care about polynomial identity testing

- Applications to graph theory
  - Whether a graph has a perfect matching or a particular subgraph can be encoded as a particular polynomial being identically zero
  - See optional question on HW1!
- Applications in Error Correcting Codes and Cryptography
  - Quickly verifying algebraic constraints
- Applications in computational geometry
  - Quickly verifying relationships between algebraic curves
- …

Algorithm

- Given a polynomial $P(x_1, \dots, x_n)$:
  - Fix a finite set S
  - Choose $r_1, r_2, \dots, r_n \in S$ independently and uniformly at random.
  - If $P(r_1, \dots, r_n) = 0$, output “P is zero!”
  - Else output “P is not zero!”

Theorem

- If P is identically zero:
  - The algorithm always outputs "P is zero!"
- If P is not identically zero:
  - The algorithm outputs "P is not zero!" with probability at least $1 - \frac{\text{degree}(P)}{|S|}$

Easy!

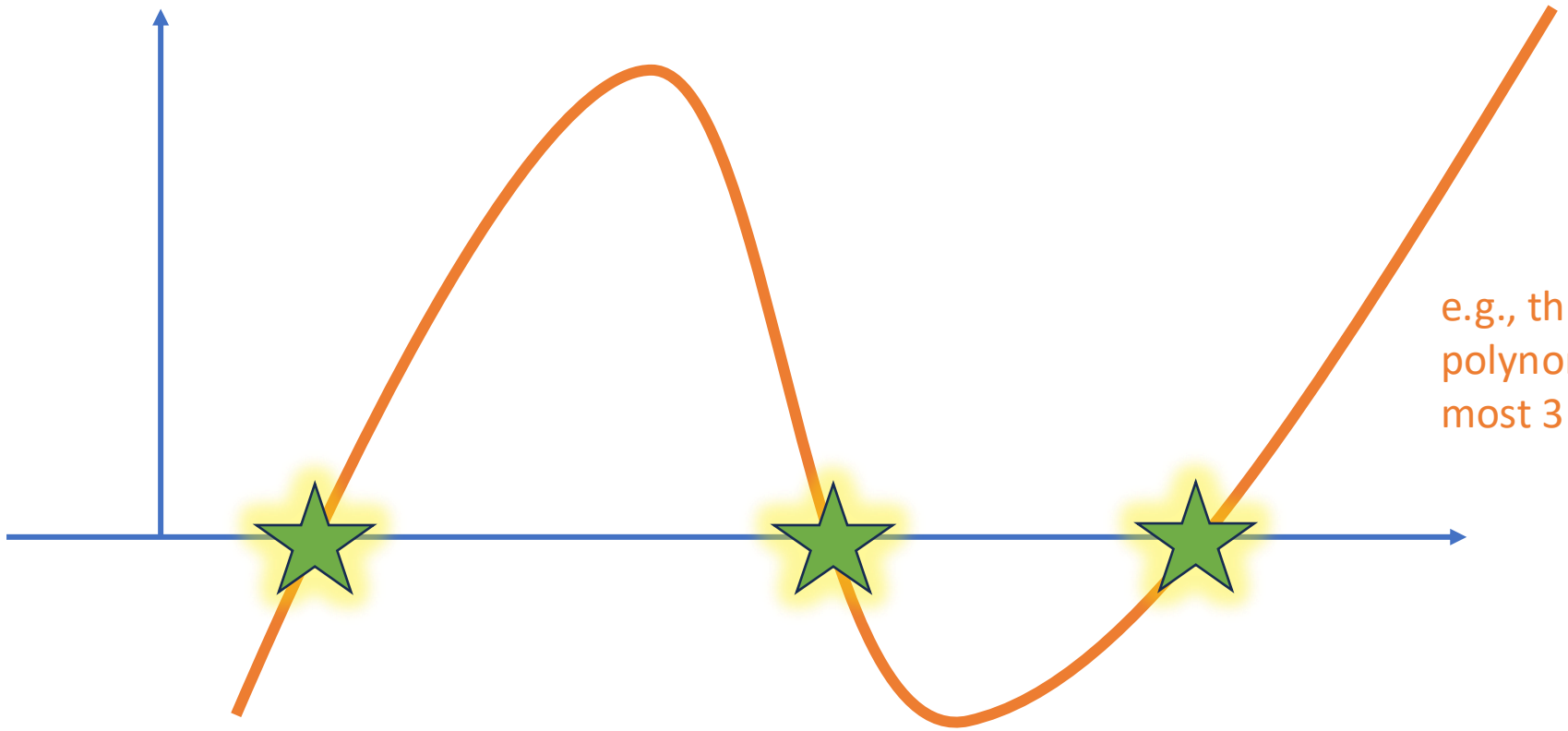
Focus on this part

Algorithm:
Given a polynomial $P(x_1, \dots, x_n)$:

- Fix a finite set S
- Choose $r_1, r_2, \dots, r_n \in S$ independently and uniformly at random.
- If $P(r_1, \dots, r_n) = 0$, output "P is zero!"
- Else output "P is not zero!"

Fact

- A nonzero univariate polynomial $P(x)$ of degree d has at most d roots.



e.g., this cubic polynomial* has at most 3 roots

*disclaimer: not actually a cubic polynomial, this is a Powerpoint drawing…

Total degree: For example, the polynomial $P(x_1, x_2) = x_1^2 x_2^3 + x_2^4$ has total degree 5.

Schwartz-Zippel Lemma

- Suppose $P(x_1, \dots, x_n)$ has *total degree* d and is not identically 0
- Fix any set S
- Draw $r_1, \dots, r_n \in S$ independently, uniformly at random
- Then $\Pr[P(r_1, \dots, r_n) = 0] \leq d/|S|$

This lemma implies our theorem!

- Aka, if $P \neq 0$, Algorithm is correct with probability $\geq 1 - d/|S|$

Algorithm:
Given a polynomial $P(x_1, \dots, x_n)$:

- Fix a finite set S
- Choose $r_1, r_2, \dots, r_n \in S$ independently and uniformly at random.
- If $P(r_1, \dots, r_n) = 0$, output "P is zero!"
- Else output "P is not zero!"

Total degree: For example, the polynomial $P(x_1, x_2) = x_1^2 x_2^3 + x_2^4$ has total degree 5.

Schwartz-Zippel Lemma

- Suppose $P(x_1, \dots, x_n)$ has *total degree* d and is not identically 0
- Fix any set S
- Draw $r_1, \dots, r_n \in S$ independently, uniformly at random
- Then $\Pr[P(r_1, \dots, r_n) = 0] \leq d/|S|$

The n=1 case follows from our Fact!

Fact

- A nonzero univariate polynomial $P(x)$ of degree d has at most d roots.

But for n>1, the fact is false!

So we can't just appeal to it in general…

E.g., $P(x, y) = x - y$ has total degree 1 but infinitely many roots.

Proof of Schwartz-Zippel

By induction on n

Say $P(x_1, \dots, x_n)$ has (total) degree d and is not identically 0
Fix any set S
Draw $r_1, \dots, r_n \in S$ independently, uniformly at random
Then $\Pr[P(r_1, \dots, r_n) = 0] \leq d/|S|$

- Base Case: statement is true for n=1.

Follows from **fact**!

Say $P(x_1, \dots, x_n)$ has (total) degree d and is not identically 0
Fix any set S
Draw $r_1, \dots, r_n \in S$ independently, uniformly at random
Then $\Pr[P(r_1, \dots, r_n) = 0] \leq d/|S|$

Proof of Schwartz-Zippel

By induction on n

- Inductive Step: suppose statement is true for $\leq n - 1$ variables.

  Write

$$P(x_1, x_2, \dots, x_n) = x_1^k Q(x_2, \dots, x_n) + T(x_1, \dots, x_n),$$

  where $Q \neq 0$ and the maximum degree of x_1 in T is $< k$, for some $k > 0$

Example: $P(x_1, x_2) = x_1^3 x_2 + x_1^3 x_2^4 + x_1^2 + x_1 x_2^7 + x_2$

- ...then the statement holds for n variables.

Say $P(x_1, \dots, x_n)$ has (total) degree d and is not identically 0
Fix any set S
Draw $r_1, \dots, r_n \in S$ independently, uniformly at random
Then $\Pr[P(r_1, \dots, r_n) = 0] \leq d/|S|$

Proof of Schwartz-Zippel

By induction on n

- Inductive Step: suppose statement is true for $\leq n - 1$ variables.

  Write

  $$P(x_1, x_2, \dots, x_n) = x_1^k Q(x_2, \dots, x_n) + T(x_1, \dots, x_n),$$

  where $Q \neq 0$ and the maximum degree of x_1 in T is $< k$, for some $k > 0$

Example: $P(x_1, x_2) = x_1^3 x_2 + x_1^3 x_2^4 + x_1^2 + x_1 x_2^7 + x_2$

$$x_1^3(x_2 + x_2^4) + (x_1^2 + x_1 x_2^7 + x_2)$$

$Q(x_2)$ $T(x_1, x_2)$

- ...then the statement holds for n variables.

Say $P(x_1, \dots, x_n)$ has (total) degree d and is not identically 0
Fix any set S
Draw $r_1, \dots, r_n \in S$ independently, uniformly at random
Then $\Pr[P(r_1, \dots, r_n) = 0] \leq d/|S|$

Proof of Schwartz-Zippel

By induction on n

- Inductive Step: suppose statement is true for $\leq n - 1$ variables.

Write

$$P(x_1, x_2, \dots, x_n) = x_1^k Q(x_2, \dots, x_n) + T(x_1, \dots, x_n),$$

where $Q \neq 0$ and the maximum degree of x_1 in T is $< k$, for some $k > 0$

$$\Pr_{r_2, \dots, r_n}[Q(r_2, \dots, r_n) \neq 0] \geq 1 - \frac{d-k}{|S|}$$

If this happens, then $P(x_1, r_2, \dots, r_n)$ is a nonzero univariate polynomial in x_1 of degree k

Assuming that happens…

$$\Pr_{r_1}[P(r_1, r_2, \dots, r_n) \neq 0] \geq 1 - \frac{k}{|S|}$$

⇒ The probability that $P(r_1, r_2, \dots, r_n) \neq 0$ is at least $\left(1 - \frac{d-k}{|S|}\right)\left(1 - \frac{k}{|S|}\right) \geq 1 - \frac{d}{|S|}$

- …then the statement holds for n variables. ✓

What have we learned?

- Our randomized algorithm works w.h.p. (provided $|S| \gg \deg(P)$).

This means "with high probability."
In this course, we'll use it in an informal way to mean "with probability close to 1"

This means "much bigger than."
Here we are also using it informally.

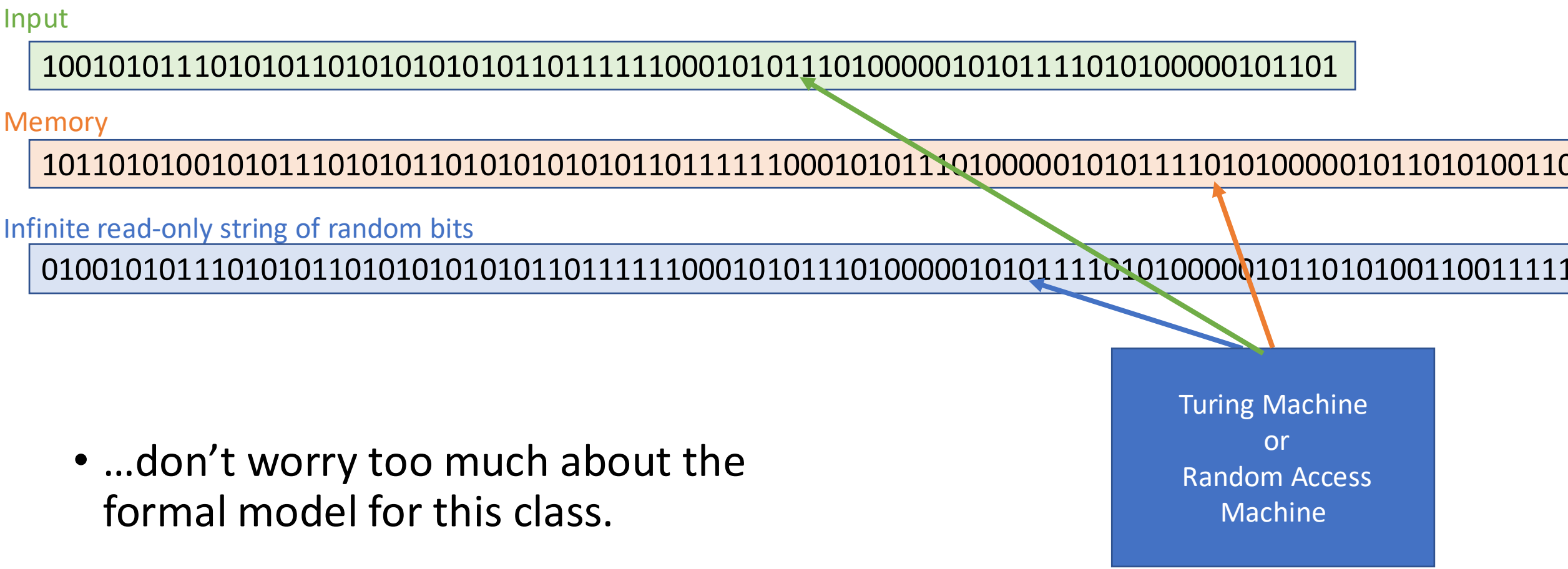
Is there an efficient deterministic algorithm?

- **No one knows!**
- There are algorithms for special cases.
- We* believe that there **are** polynomial-time deterministic algorithms
  - Otherwise there would be some weird complexity-theoretic consequences
- **But** if we could find one, it would solve a hard open problem in complexity theory
  - It would imply strong circuit lower bounds
- General theme in this class:
  - Simple randomized algorithms, often with simple (if clever) analyses.
  - Maybe there are deterministic algorithms, but they are probably pretty complicated.

*We = most theoretical computer scientists

What is a randomized algorithm?

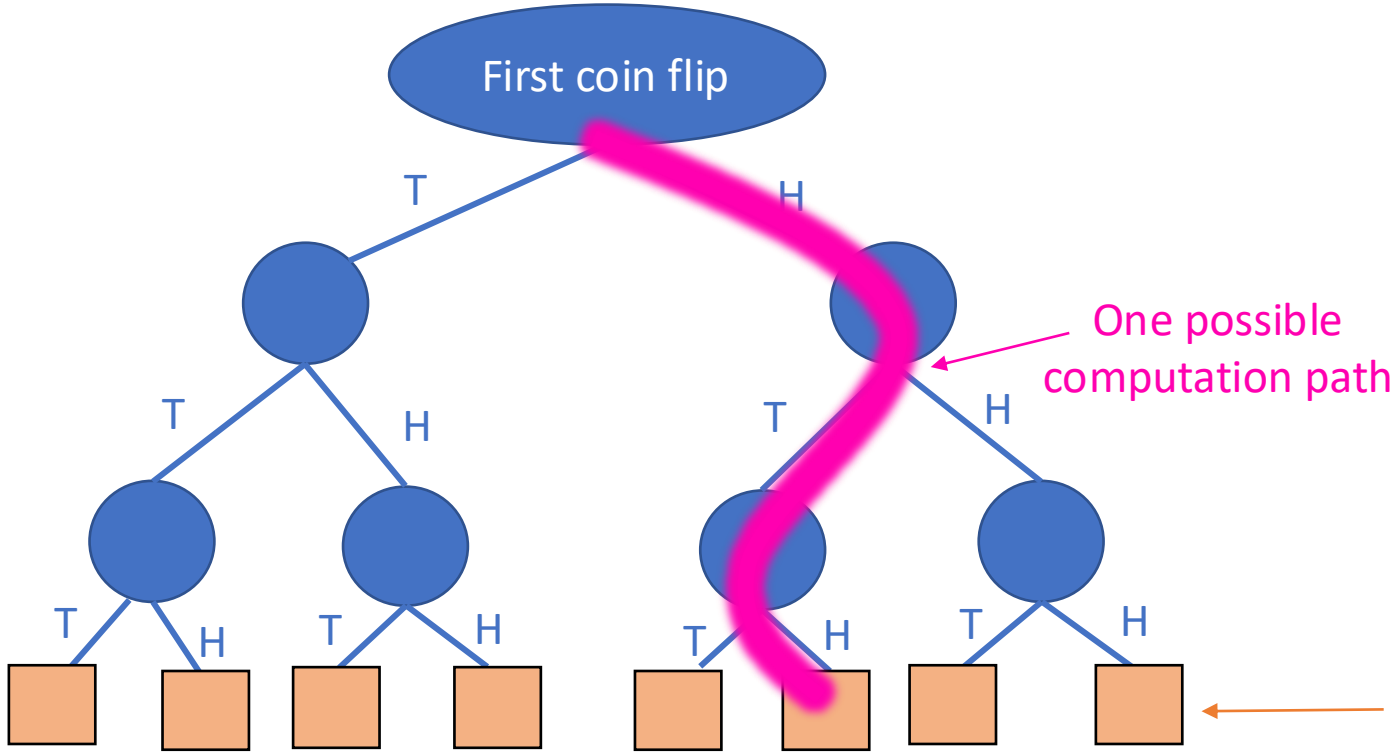
Computational Model



- …don’t worry too much about the formal model for this class.

Computational Model

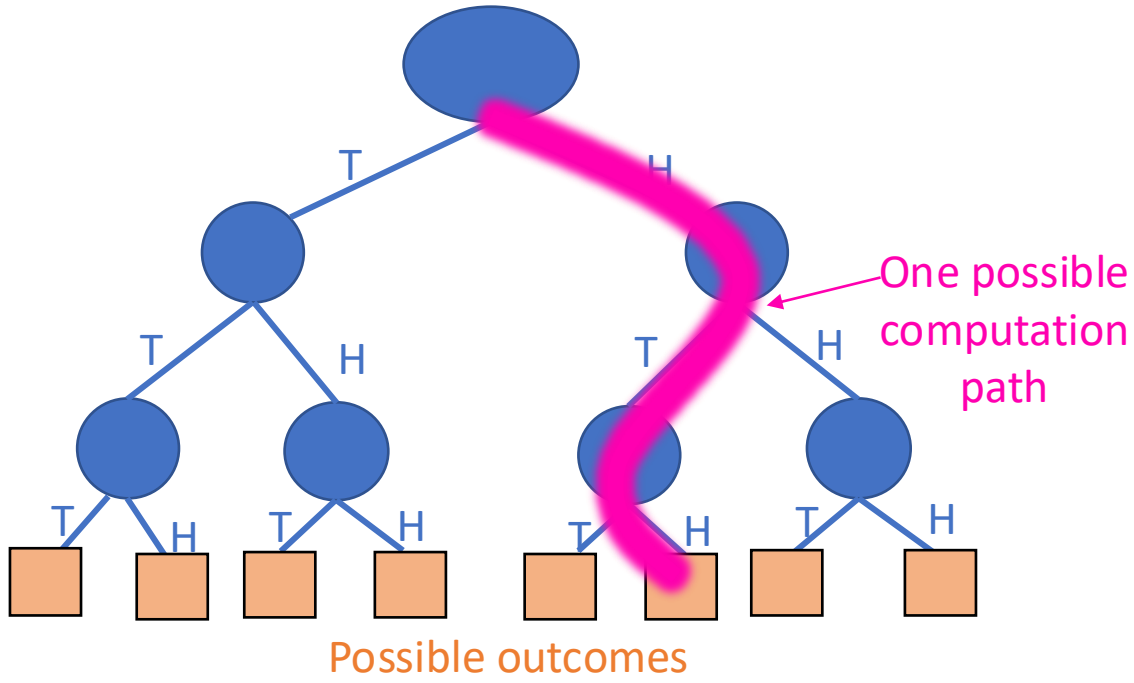
- Allow your algorithm to flip random coins.
- Computation path on a specific input:



Turing Machine
or
Random Access Machine

Observations

- The **output** of a randomized algorithm is a random variable.
- The **execution path** of a randomized algorithm is a random variable.



- Meaningful statements:
  - "For any input x, the algorithm is successful on x with high probability."
  - "For any input x, the running time of the algorithm on x is small with high probability."

Two types of randomized algorithms

- **Las Vegas Algorithms** always output the correct answer, but might be slow.
  - Output is deterministic.
  - Running time is a random variable. (We will demand $\mathbb{E}[\text{runtime}] < \infty$).

e.g., Quicksort

- **Monte Carlo Algorithms** are always fast, but might be wrong.
  - Output is a random variable.
  - Running time is bounded by something deterministic.

e.g., Karger's Algorithm

One-sided vs. two-sided error

- For Monte-Carlo algorithms that output yes/no:
  - One-sided error:
    - If the answer is “Yes,” the algorithm says “Yes” with probability 1.
    - If the answer is “No,” the algorithm says “No” with probability at least $\epsilon > 0$.
  - Two-sided error:
    - No matter what the answer is, the algorithm is correct with probability at least $\frac{1}{2} + \epsilon$, for some $\epsilon > 0$.

One-sided vs. two-sided error

- For Monte-Carlo algorithms that output yes/no:
  - One-sided error:
    - If the answer is "Yes," the algorithm says "Yes" with probability 1.
    - If the answer is "No," the algorithm says "No" with probability at least $\epsilon > 0$.

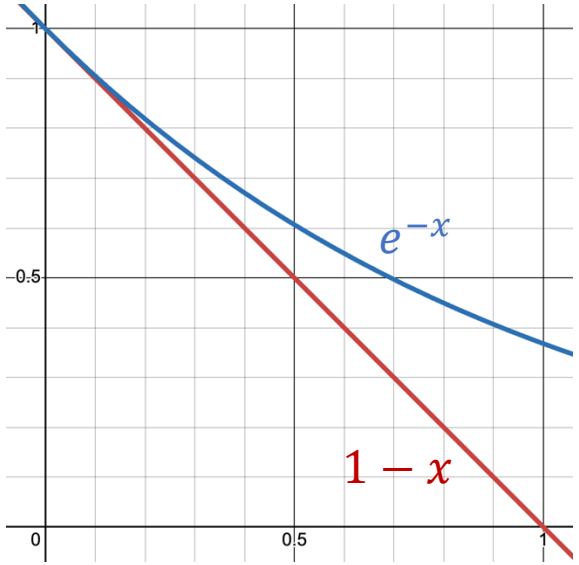
How can we turn this into an algorithm that is correct 90% of the time?

  - Two-sided error:
    - No matter what the answer is, the algorithm is correct with probability at least $\frac{1}{2} + \epsilon$, for some $\epsilon > 0$.

Repeat it a bunch of times

Very useful facts:

- $1 - x \leq e^{-x}$
- $1 - x \approx e^{-x}$ for small x



- For $i = 1, \ldots, t$:
  - Run one-sided algorithm.
  - If it says "No", return "No."
- Return "Yes".

- If the correct answer is "Yes", will always return "Yes."
- If the correct answer is "No",

$$\begin{aligned} \Pr[\text{incorrect}] &= \Pr[\text{each of } t \text{ trials says "}Yes\text{"}] \\ &\leq (1 - \epsilon)^t \\ &\leq e^{-\epsilon t} \end{aligned}$$

Can make this tiny by choosing $t \gg 1/\epsilon$ sufficiently large.

One-sided vs. two-sided error

- For Monte-Carlo algorithms that output yes/no:
  - One-sided error:
    - If the answer is “Yes,” the algorithm says “Yes” with probability 1.
    - If the answer is “No,” the algorithm says “No” with probability at least $\epsilon > 0$.
  - Two-sided error:
    - No matter what the answer is, the algorithm is correct with probability at least $\frac{1}{2} + \epsilon$, for some $\epsilon > 0$.

How can we turn this into an algorithm that is correct 90% of the time?

Repeat it a bunch of times

- For $i = 1, \ldots, t$:
  - Run two-sided algorithm.
- Return the majority answer

Claim: Pr[this gets the wrong answer] $\leq e^{-2t\epsilon^2}$

Proof: See lecture notes.

So if $t \gg \frac{1}{2\epsilon^2}$, this is tiny!

Wrap-up

What have we learned?

- Polynomial Identity Testing is a useful primitive.
  - We know an efficient randomized algorithm.
  - No efficient deterministic algorithm known!

- Randomized algorithms are algorithms that use randomness.
  - We won't stress the exact model too much.

- Going forward in this class, we're going to learn lots more about randomized algorithms and ways to analyze them!
  - Next time: Karger's algorithm!

Before next time

- Watch the video lectures for Wednesday **before** class, and do the corresponding quiz.

- Let us know if you have conflicts with the exams!