15-251: Great Theoretical Ideas in Computer Science

Lecture 24

(Interactive) Proofs

Proof. Define f_{ij} as in (5). As f is symmetric, we only need to consider f_{12}.

$$\begin{aligned}
\mathbf{E}\left[f_{12}^2\right] &= \mathbf{E}_{x_3\ldots x_n}\left[\frac{1}{4}\cdot\left(f_{12}^2(00x_3\ldots x_n)+f_{12}^2(01x_3\ldots x_n)+f_{12}^2(10x_3\ldots x_n)+f_{12}^2(11x_3\ldots x_n)\right)\right]\\
&= \frac{1}{4}\mathbf{E}_{x_3\ldots x_n}\left[\left(f(00x_3\ldots x_n)-f(11x_3\ldots x_n)\right)^2+\left(f(11x_3\ldots x_n)-f(00x_3\ldots x_n)\right)^2\right]\\
&\geq \frac{1}{2}\left(\binom{n-2}{r_0-1}\cdot 2^{-(n-2)}\cdot 4+\binom{n-2}{n-r_1-1}\cdot 2^{-(n-2)}\cdot 4\right)\\
&= 8\cdot\left(\frac{(n-r_0+1)(n-r_0)}{n(n-1)}\cdot\binom{n}{r_0-1}+\frac{(n-r_1+1)(n-r_1)}{n(n-1)}\cdot\binom{n}{r_1-1}\right)2^{-n}.
\end{aligned}$$

Inequality (6) follows by applying Lemma 2.2.

In order to establish inequality (7), we show a lower bound on the principal Fourier coefficient of f:

$$\widehat{f}(\emptyset)\geq 1-2\left(\sum_{s<r_0}\binom{n}{s}+\sum_{s>n-r_1}\binom{n}{s}\right)2^{-n},$$

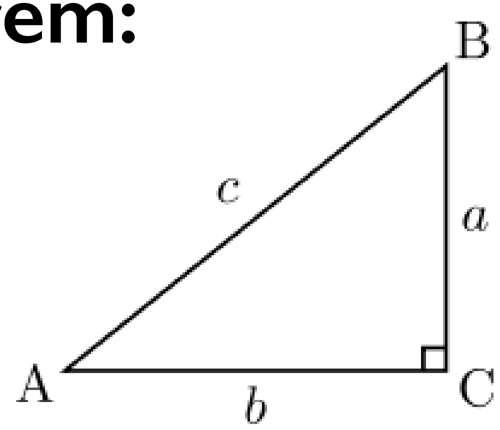
which implies that

$$\widehat{f}(\emptyset)^2\geq 1-4\cdot\left(\sum_{s<r_0}\binom{n}{s}+\sum_{s<r_1}\binom{n}{s}\right)2^{-n}.$$

□

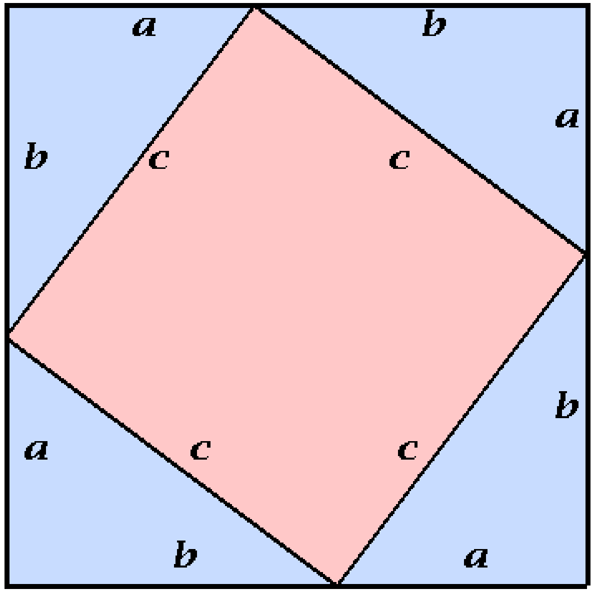
Proofs from 900 BCE until 1800s

Pythagoras's Theorem:



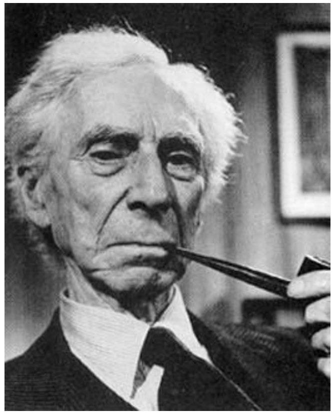
$$a^2 + b^2 = c^2$$

Proof:



$$(a+b)^2 = a^2 + 2ab + b^2$$

Then there was Russell



Russell and others worked on formalizing proofs.

Principia Mathematica Volume 2

$*110\cdot632.\ \vdash : \mu \in NC . \supset . \mu +_c 1 = \hat{\xi}\{(\exists y) . y \in \xi . \xi - \iota`y \in sm``\mu\}$

Dem.

$\vdash . *110\cdot631 . *51\cdot211\cdot22 . \supset$

$\vdash : Hp . \supset . \mu +_c 1 = \hat{\xi}\{(\exists\gamma, y) . \gamma \in sm``\mu . y \in \xi . \gamma = \xi - \iota`y\}$

$[*13\cdot195] \quad = \hat{\xi}\{(\exists y) . y \in \xi . \xi - \iota`y \in sm``\mu\} : \supset \vdash . Prop$

$*110\cdot64.\ \vdash . 0 +_c 0 = 0 \quad [*110\cdot62]$

$*110\cdot641.\ \vdash . 1 +_c 0 = 0 +_c 1 = 1 \quad [*110\cdot51\cdot61 . *101\cdot2]$

$*110\cdot642.\ \vdash . 2 +_c 0 = 0 +_c 2 = 2 \quad [*110\cdot51\cdot61 . *101\cdot31]$

$*110\cdot643.\ \vdash . 1 +_c 1 = 2$

Dem.

$\vdash . *110\cdot632 . *101\cdot21\cdot28 . \supset$

$\vdash . 1 +_c 1 = \hat{\xi}\{(\exists y) . y \in \xi . \xi - \iota`y \in 1\}$

$[*54\cdot3] \quad = 2 . \supset \vdash . Prop$

The above proposition is occasionally useful. It is used at least three times, in $*113\cdot66$ and $*120\cdot123\cdot472$.

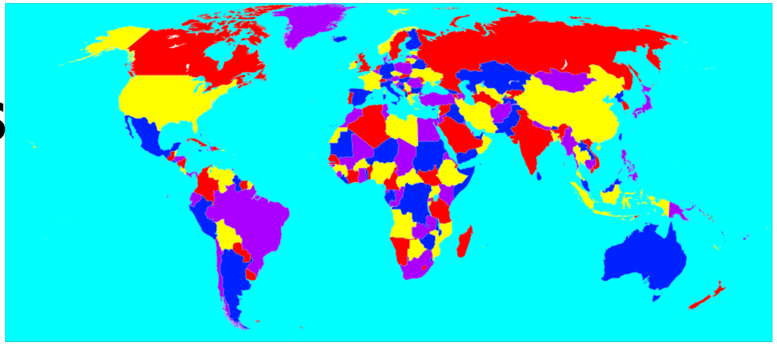
This meant proofs could be verified mechanically.

Proofs and Computers

All this played a key role in the birth of computer science.

Computers themselves can verify proofs.
(automated theorem provers)

Computers can help us find proofs
(e.g. 4-Color Theorem)



Are these really proofs?

TODAY: Proofs and Computer Science

A modern understanding of proofs in computer science includes proofs that are:

- randomized
- interactive
- zero-knowledge (proofs which don't explain anything)
- spot-checkable

This modern understanding of proofs has revolutionized much of theoretical computer science.

Review of NP

Definition:

A language A is in NP if

- there is a polynomial time TM V
- a polynomial p

such that for all x :

$$x \in A \iff \exists u \text{ with } |u| \le p(|x|) \text{ s.t. } V(x,u) = 1$$

"$x \in A$ iff there is a polynomial length proof u that is verifiable by a poly-time algorithm."

If $x \in A$, there is some proof that leads V to accept.

If $x \notin A$, every "proof" leads V to reject.

NP: A game between a Prover and a Verifier

Verifier | **Prover**

poly-time
skeptical

omniscient
untrustworthy

Given some string x.

Prover wants to convince Verifier $x \in A$.

Prover cooks up a proof string u and sends it to Verifier.

Verifier, in polynomial time, should be able to tell if the proof is legit.

NP: A game between a Prover and a Verifier

Verifier | **Prover**

poly-time
skeptical

omniscient
untrustworthy

"Completeness"

If $x \in A$, there must be some proof u that convinces the Verifier.

"Soundness"

If $x \notin A$, no matter what "proof" Prover gives, Verifier should detect the lie.

Limitations of NP

We know many languages are in NP.

SAT, 3SAT, CLIQUE, MAX-CUT, VERTEX-COVER, SUDOKU, THEOREM-PROVING, 3COL, ...

What about $\overline{\text{3COL}}$ or $\overline{\text{3SAT}}$?

i.e.

Given an unsatisfiable formula, is there a way for the Prover to convince the Verifier that it is unsatisfiable?

How can we generalize proofs?

The NP setting seems too weak for this purpose.

But, in real life, people use more general ways of convincing each other of the validity of statements.

- Make the protocol interactive.

  One can show interaction does not change the model.
  I.e., whatever you can do with interaction, you can do with the original setting.

- Make the verifier probabilistic.

  We do not think randomization by itself adds significant power.

But, magic happens when you combine the two.

Interaction + Randomization

Coke vs Pepsi Challenge

Your friend tells you he can taste the difference between Coke and Pepsi.

How can he convince you of this?

Coke vs Pepsi

Choose Coke or Pepsi
at random.

a challenge

Send it to your friend. → Your friend tastes it

← Coke

Gives an answer.

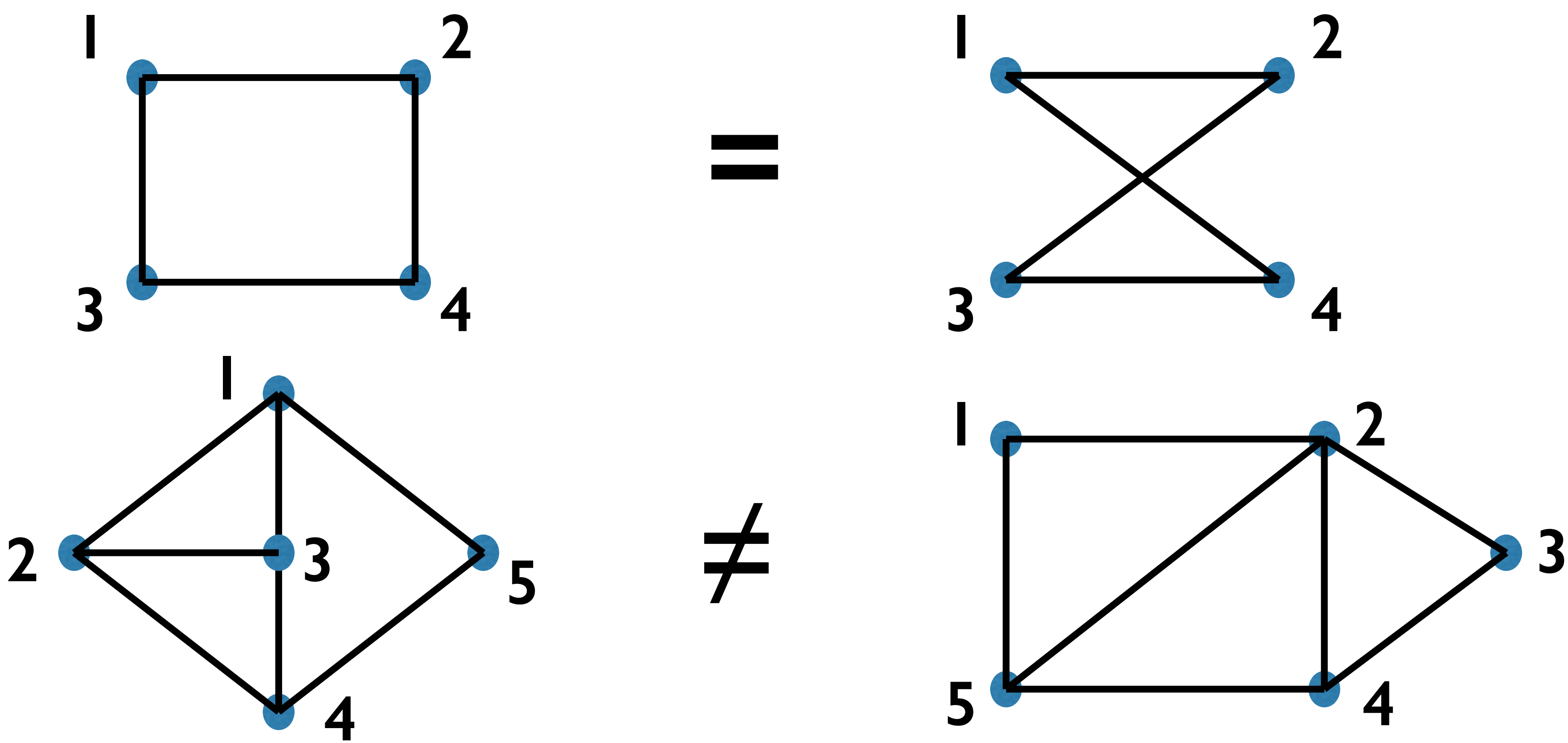
a response
to the challenge

Repeat

Graph Isomorphism Problem

Given two graphs G_1, G_2, are they isomorphic?
i.e., is there a permutation π of the vertices such that

$$\pi(G_1) = G_2$$



Graph Isomorphism Problem

Is Graph Isomorphism in NP?

Sure! A good proof is the permutation of the vertices.

Is Graph Non-isomorphism in NP?

No one knows!

But there is a simple randomized interactive proof.

Interactive Proof for Graph Non-isomorphism

$\langle G_1, G_2 \rangle$

Pick at random $i \in \{1, 2\}$

Choose a permutation π **of vertices at random.**

a challenge

$\pi(G_i)$ →

j ←

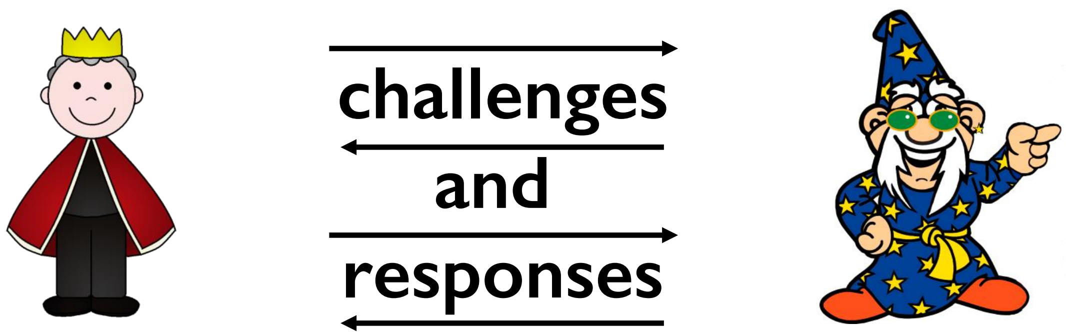
Accept if $i = j$

a response to the challenge

The complexity class IP

We say that a language A is in IP if:

- there is a probabilistic poly-time Verifier 
- there is a computationally unbounded Prover 



(poly rounds)

"Completeness"

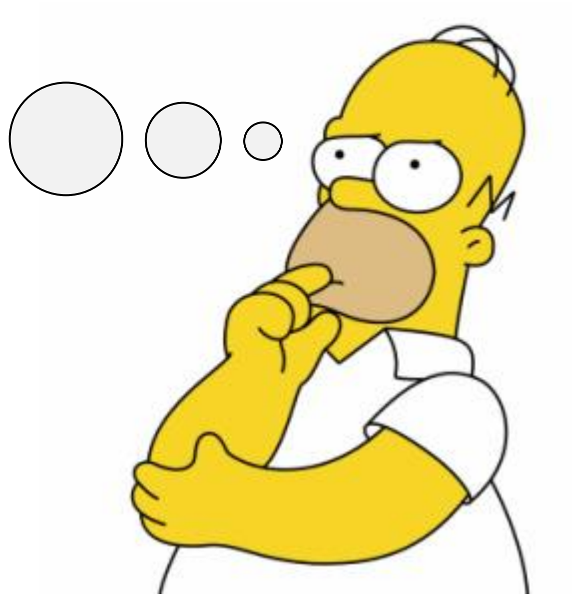
If $x \in A$, Verifier accepts.

"Soundness"

If $x \notin A$, Verifier rejects with prob. at least 1/2.

The complexity class IP

But being fooled with probability ½ is still pretty bad! What can we do about it?



Repeat: After 100 challenges the probability to be fooled is < 1/1000000000000000000000000000000

Poll 1: What is the power of IP

Poll 1: What is the relation between NP and IP?

1. $NP \subset IP$
2. $IP \subset NP$
3. $IP = NP$
4. They are incomparable

Poll 1: What is the power of IP

Poll 1: What is the relation between NP and IP?

1. $NP \subset IP$ ✓
2. $IP \subset NP$
3. $IP = NP$
4. They are incomparable

The power of IP

We showed that Graph Non-Isomorphism is in IP.

What about $\overline{\mathrm{3SAT}}$? Is it in IP?

Yes!

In fact, the complement of any language in NP is in IP.

Many more languages beyond this are in IP, too.

How powerful is IP?

So how powerful are interactive proofs?

How big is IP?

Theorem:

$$\mathrm{IP} = \mathrm{PSPACE}$$

Adi Shamir

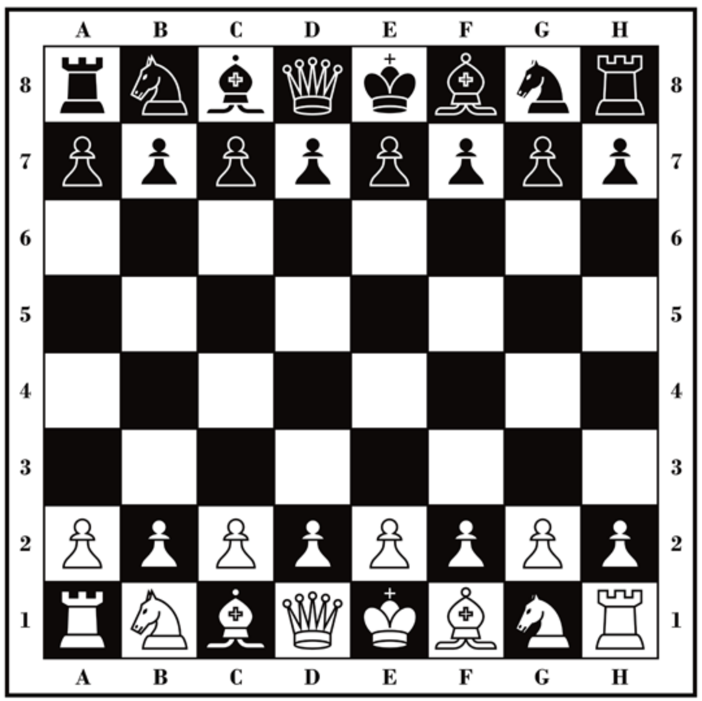
1990

(another application of polynomials)

Chess

An interesting corollary:

Suppose in chess, white can always win in ≤ 300 moves.



How can the wizard prove this to you?

Zero Knowledge Proofs

Zero-Knowledge Proofs

I found a truly marvelous proof of Riemann Hypothesis.

I want to convince you that I have a valid proof.

But I don't want you to learn anything about the proof.

Is this possible?

For what problems is there a zero-knowledge IP?

Back to Graph Non-isomorphism

$\langle G_1, G_2 \rangle$

Pick at random $i \in \{1, 2\}$

Choose a permutation π of vertices at random.

$\pi(G_i)$ →

← j

Accept if $i = j$

There is more to this protocol than meets the eye.

Back to Graph Non-isomorphism

Does the verifier gain any insight about why the graphs are not isomorphic?

$\langle G_1, G_2 \rangle$

Pick at random $i \in \{1, 2\}$

Choose a permutation π of vertices at random.

$\pi(G_i)$ →

j ←

Accept if $i = j$

There is more to this protocol than meets the eye.

Zero-Knowledge Proofs

The Verifier is convinced,
but he learns nothing about why the graphs are not isomorphic!

The Verifier could have produced the communication transcript by himself, with no help from the Prover.

A proof with 0 explanatory content!

Zero-Knowledge Proofs for NP


Goldreich


Micali


Wigderson

1986

Does every problem in NP have a zero-knowledge IP?

Yes! (under plausible cryptographic assumptions)

And the prover need not be a wizard.

He just needs to know the ordinary proof.

Zero-Knowledge Proofs for NP

Does every problem in NP have a zero-knowledge IP?

Yes! (under plausible cryptographic assumptions)

And the prover need not be a wizard.

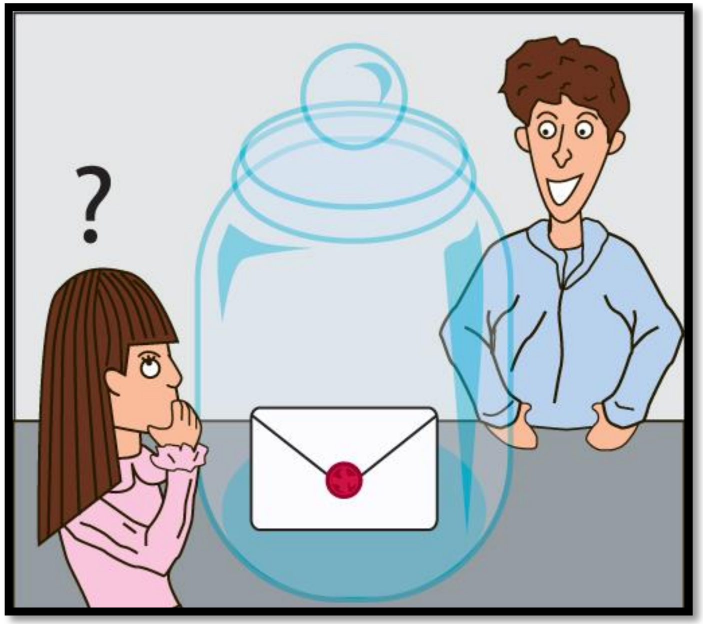
He just needs to know the ordinary proof.

It suffices to show this for your favorite NP-complete problem. (every problem in NP reduces to an NP-complete prob.)

We'll pick the 3-COLORING Problem.

Zero-Knowledge Proof for 3-Coloring

- We want to design an zero knowledge proof system for 3-COLORING
- We will rely on a cryptographic construction known as bit commitment
- Prover can put bits in envelopes and send them to Verifier
- Verifier can only open an envelope if Prover provides the key



Zero-Knowledge Proof for 3-Coloring

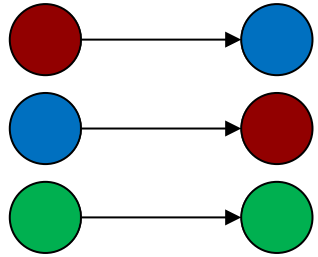
Selects random permutation π of $\{R, G, B\}$; commits to $\pi\big(\gamma(v)\big)$ for all $v \in V$

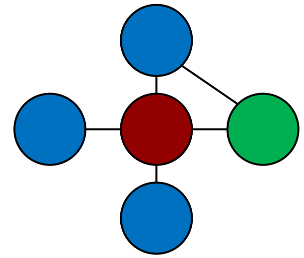
Selects an edge $(u, v) \in E$ uniformly at random

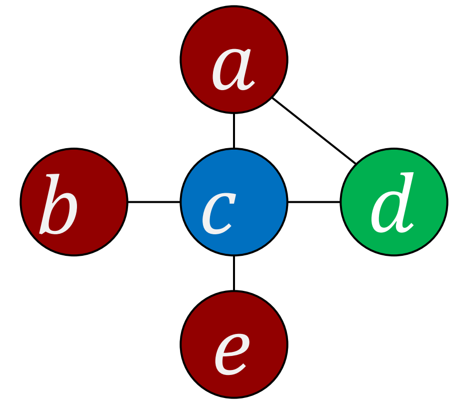
Reveals $a = \pi\big(\gamma(u)\big)$ and $b = \pi(\gamma(v))$

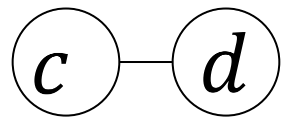
Accepts iff $a \neq b$

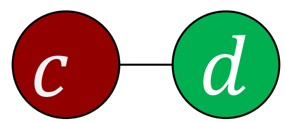
Zero-Knowledge Proof for 3-Coloring











$\gamma(G)$

Accept

Poll 2: Zero-Knowledge Proof for 3-Coloring

Selects random permutation π of $\{R, G, B\}$;
commits to $\pi(\gamma(v))$ for all $v \in V$

Selects an edge $(u, v) \in E$ uniformly at random

Reveals $a = \pi(\gamma(u))$ and $b = \pi(\gamma(v))$

Accepts iff $a \neq b$

Poll 2: If G has no 3-coloring, what is the worst-case prob. for Prover to convince Verifier?

$1 - \frac{1}{3!}$ $\qquad 1 - \frac{1}{|E|}$ $\qquad 1 - \frac{1}{2}$ $\qquad 1 - \frac{1}{n!}$

Poll 2: Zero-Knowledge Proof for 3-Coloring

Selects random permutation π of $\{R, G, B\}$; commits to $\pi(\gamma(v))$ for all $v \in V$

Selects an edge $(u, v) \in E$ uniformly at random

Reveals $a = \pi(\gamma(u))$ and $b = \pi(\gamma(v))$

Accepts iff $a \neq b$

Poll 2: **If G has no 3-coloring, what is the worst-case prob. for Prover to convince Verifier?**

$1 - \frac{1}{3!}$

$1 - \frac{1}{|E|}$ ✓

$1 - \frac{1}{2}$

$1 - \frac{1}{n!}$

Zero-Knowledge Proof for 3-Coloring

Selects random permutation π of $\{R, G, B\}$;
commits to $\pi(\gamma(v))$ for all $v \in V$

Selects an edge $(u, v) \in E$ uniformly
at random

Reveals $a = \pi(\gamma(u))$ and $b = \pi(\gamma(v))$

Accepts iff $a \neq b$

Completeness:

Follows from valid 3-coloring

Soundness:

Repeat $2|E|$ times to get ½ prob.

Zero knowledge:

Prover just reveals a pair of distinct random colors.

Zero-Knowledge for all?

This shows that every problem in NP has a zero knowledge IP.

In fact, every problem in IP = PSPACE has a zero-knowledge proof!

Ben-Or

Goldreich

Goldwasser

Håstad

Kilian

Micali

Rogaway

1990

"Everything provable is provable in zero-knowledge"

Statistical vs Computational Zero-Knowledge

There is a difference between
- zero-knowledge proof for Graph Non-isomorphism
- zero-knowledge proof for Hamiltonian Cycle

Statistical zero-knowledge:

Verifier wouldn't learn anything even if it was computationally unbounded.

Computational zero-knowledge:

Verifier wouldn't learn anything assuming it cannot unlock the locks in polynomial time.

Statistical vs Computational Zero-Knowledge

SZK = set of all problems with
statistically zero-knowledge proofs

CZK = set of all problems with
computationally zero-knowledge proofs

IP = PSPACE = CZK

SZK is believed to be much smaller.

In fact, it is believed that it does not contain NP-complete problems.

And now…

Modern computer science proofs can be:

- randomized
- interactive
- zero-knowledge
- spot-checkable

Spot-Checkable Proofs

Suppose I have a proof that is a few hundred pages long.

I give you the proof, and ask you to verify it.

It could be that there is some tiny mistake somewhere in the proof.

Trying to find it is super annoying!

Spot-Checkable Proofs

If only there was a way to just check a few random places of the proof, and be convinced that the proof is correct…

That's a dream too good to be true.

Or is it?

Let's go back to Graph Non-isomorphism.

Can we realize this dream for this problem?

Given two graphs G_0, G_1, is there a "spot-checkable" proof that they are non-isomorphic?

Spot-Checkable Proofs

Enumerate all possible n-vertex graphs:

$H_1, H_2, H_3, H_4, H_5, H_6, H_7, \ldots, H_N$ $\quad N = 2^{\binom{n}{2}}$

proof: 0 1 0 0 1 1 0 ... 1

Index i: if $H_i \approx G_0$, put 0.

if $H_i \approx G_1$, put 1.

if neither, put 0 or 1 (doesn't matter).

Verifier:

Pick at random $i \in \{0, 1\}$.

Choose a permutation π of vertices at random.

Figure out the index j corresponding to $\pi(G_i)$.

Check: is the bit at index j equal to i.

Spot-Checkable Proofs

OK, the proof is exponentially long.

Not so useful in that sense.

Is there a way to do something similar but with poly-length proof?

Spot-Checkable Proofs

Probabilistically Checkable Proofs (PCP) Theorem:

Every problem in NP admits "spot-checkable" proofs of polynomial length.
The verifier can be convinced with high probability by looking only at a *constant* number of bits in the proof.

old proof (poly-length) → new proof (poly-length)

tiny local error → *error almost everywhere*

"New shortcut found for long math proofs!"

Spot-Checkable Proofs

Probabilistically Checkable Proofs (PCP) Theorem:

Every problem in NP admits "spot-checkable" proofs of polynomial length.
The verifier can be convinced with high probability by looking only at a ***constant*** number of bits in the proof.

1998

Arora

Lund

Motwani

Safra

Sudan

Szegedy

Spot-Checkable Proofs

This theorem is <u>equivalent</u> to:

PCP Theorem (version 2):

There is some constant ϵ such that if there is a polynomial-time ϵ-approximation algorithm for MAX-3SAT then P=NP.

I.e., it is NP-hard to approximate MAX-3SAT within an ϵ factor.

This is called an "*hardness of approximation*" result.

They are hard to prove!

Spot-Checkable Proofs

PCP Theorem is one of the crowning achievements in CS theory!

Proof is a half a semester course.

Blends together:

- P/NP
- random walks
- expander graphs
- polynomials / finite fields
- error-correcting codes
- Fourier analysis

Summary

Computer science gives a whole new perspective on proofs:

- can be probabilistic
- can be interactive
- can be zero-knowledge
- can be spot-checkable

Summary

old-fashioned proof + deterministic verifier

problems whose solutions can be efficiently verifiable:

NP

randomization + interaction

problems whose solutions can be efficiently verifiable:

PSPACE

PSPACE = Computationally Zero-Knowledge (CZK)

"Everything provable is provable in zero-knowledge"

(some special problems are in SZK)

Summary

PCP Theorem

Old-fashioned proofs can be turned into spot-checkable.
(you only need to check constant number of bits!)

Equivalent to an hardness of approximation result.

Opens the door to many other hardness of approximation results.