Interactive Proofs

NP - 1 round of interaction between

A poly-time verifier V

&

an all powerful prover that wants

V to accept.

NP = languages that can be decided

correctly with such a protocol.

What if we allow interaction?

poly(n) many rounds

$$\text{Verifier} \quad \begin{array}{c} \longrightarrow \\ \longleftarrow \\ \longrightarrow \\ \longleftarrow \\ \vdots \end{array} \quad \text{Prover}$$

At the end, V says 0 or 1.

Observation: the class of problems is still NP!

Reason: Given an interactive protocol, we can simulate it with a 1-round protocol where the prover just anticipates the messages sent by the Verifier & sends all the replies at once!

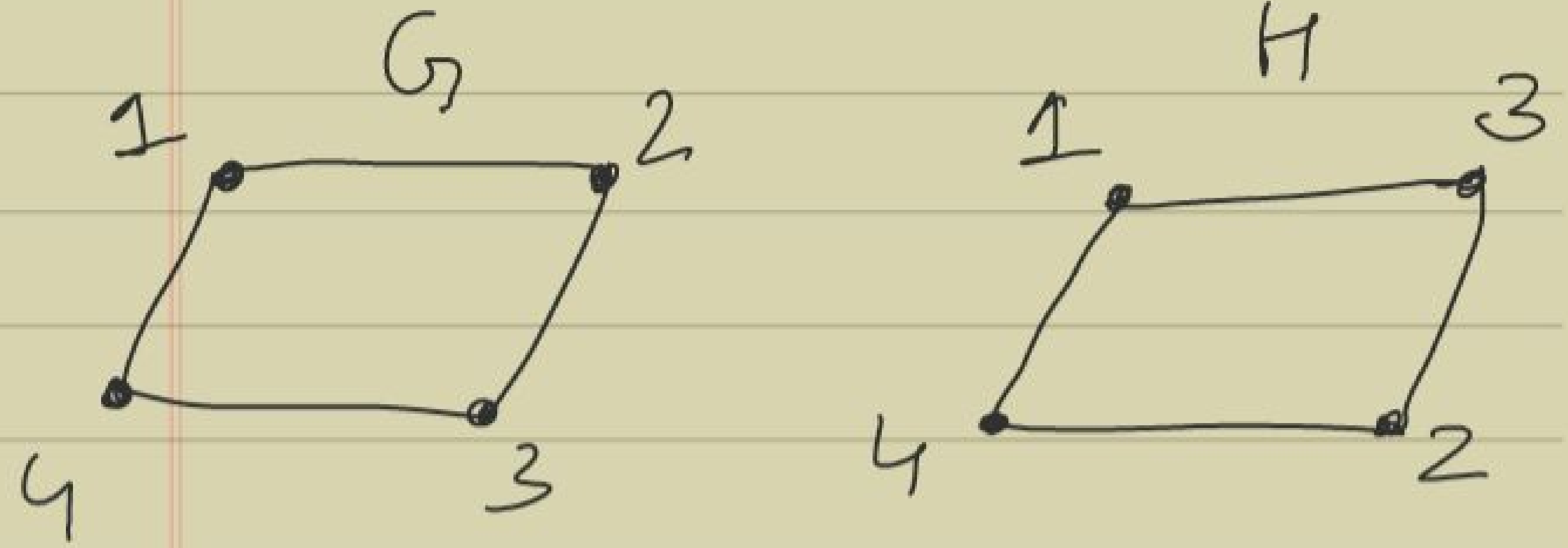
So "Interactive" NP = NP.

What if we now allow V to be a randomized poly-time algorithm?

Exponentially many choices for V's messages! P can't send replies to all (in poly(n) sized messages).

Additionally, the "correct" answer may be hidden by the random choices. (See example below.)

Graph Isomorphism (GI)



G & H are essentially "the same" graph

i.e there is a way to rename vertices of G so that it becomes H.

Formally, we say two graphs G & H are isomorphic if there is a bijection $\pi : V(G) \to V(H)$ s.t.

$$\{u,v\} \in E(G) \iff \{\pi(u), \pi(v)\} \in E(H)$$

$GI = \{ (G,H) \mid G \;\&\; H \text{ isomorphic}\}$

Clear: $GI \in NP$

Certificate: an isomorphism π

What about

$$GNI = \{ (G,H) \mid G \,\&\, H \text{ not isomorphic} \}?$$

No clear way to certify two graphs are not isomorphic!

But we will see a simple interactive protocol (with randomized verifier).

Notation: $G \cong H$ (isomorphic)

$G \not\cong H$ (not isomorphic)

GNI protcol

Input: Graphs (G, H) on the vertex set $\{1, \dots, n\}$.

Verifier:

1. Choose $b \in_R \{0,1\}$
2. If $b=0$, randomly permute the vertices of G [i.e choose a random bijection $\pi : \{1,\dots,n\} \to \{1,\dots,n\}$ & rename i by $\pi(i)$.] & call this graph R
3. If $b=1$, get R similarly from H
4. Send R to Prover & ask for b.
5. Accept if & only if Prover guesses b correctly.

Analysis:

Let $\mathcal{G}_0$ = possible graphs R if $b=0$

$\mathcal{G}_1$ = " if $b=1$

Obs: If $G \not\cong H$, then $\mathcal{G}_0$ & $\mathcal{G}_1$ are disjoint!

Reason: If G can be permuted to R & H can be permuted to R then G can be permuted to H!

So in this case, the prover can always compute b based on graph R.

Obs: If $G \cong H$, then $\mathcal{G}_0 = \mathcal{G}_1 = \mathcal{G}$ & R is uniform over $\mathcal{G}$.

Reason: Think of the following alternate experiment to produce R from G or H.

→ "Erase" the names of all the vertices.

→ Fill in random names from $\{1, \dots, n\}$ for the vertices.

From this view, it should be clear that no matter which of G & H we started with, R has the same distribution!

Thus, in this case, the best the prover can do is guess b. This is correct with prob. $1/2$.

So:

$G \not\cong H \Rightarrow$ The prover can answer in a way that the verifier accepts with probability 1.

$G \cong H \Rightarrow$ No matter what the prover says the verifier accepts with probability at most $1/2$.

Can repeat the protocol k times to reduce $1/2$ to $1/2^k$.

Interactive Proofs with k rounds

IP[k]

Class of languages L for which there is a protocol with at most k rounds of polynomially long messages between a probabilistic poly time TM V & an unbounded prover P s.t.

$x \in L \Rightarrow \exists P: \Pr[V \text{ accepts}] \geq 2/3$

$x \notin L \Rightarrow \forall P: \Pr[V \text{ accepts}] \leq 1/3$.

Called Completeness & Soundness conditions respectively

$$\boxed{IP = \bigcup_c IP[n^c]}$$

poly(n) many rounds of messages.

Note: (1) Probability is over the choice of V's random bits

(2) We can replace $2/3$ by $1/2 + 1/poly(n)$ & $1/3$ by $1/2 - 1/poly(n)$ in completeness & soundness conditions without changing the class.

Clearly, $IP \supseteq NP$.

Is it more powerful? For a while, researchers thought not, until...

Thm: $IP = PSPACE$ (!).

Landmark result in Complexity theory with reverberations in many other fields.

We will see: $IP \subseteq PSPACE$ (relatively straightforward)

& $\text{co-}NP \subseteq IP$

(highly non-trivial & strong evidence that $IP \neq NP$).

$IP \subseteq PSPACE$

$L \in IP[n^c]$ & V the probabilistic TM running time $\leq n^c$.

Given x of length n, want to decide if $x \in L$ or not using poly(n) space.

We compute something more general:

Assume that $k \leq n^c$ rounds have already passed with prover sending messages

$y_1, y_3, y_5, \ldots$ of length $\leq n^c$

& verifier sending (random) messages $z_2, z_4, \ldots$ of length $\leq n^c$

[i.e. Prover sending messages in the odd rounds.]

y_k if k is odd & z_k if k is even

$P_k(x, y_1, z_2, y_3, z_4, \ldots, u_k) =$

probability that V accepts from now on if prover "behaves optimally" (ie is the best possible way to make V accept)

Obs ① $x \in L \iff P_0(x) \geq 2/3$.

② If k even

$$P_k(x, y_1, z_2, \ldots, z_k) = \max_{y_{k+1} \in \{0,1\}^{n^c}} P_{k+1}(x, y_1, z_2, \ldots, y_{k+1})$$

③ If k odd

Can compute in space n^c

$$P_k(x, y_1, z_2, \ldots, y_k) = \sum_{z_{k+1} \in \{0,1\}^{n^c}} \Pr[\text{next message} = z_{k+1}] \cdot P_{k+1}(x, y_1, z_2, \ldots, z_{k+1})$$

This gives a poly(n)-space algorithm to compute P_x using a recursive call to P_{x+1}. [Ex.]

co-NP $\subseteq$ IP or equivalently UNSAT $\in$ IP

Input: φ a 3-CNF

Want to know: φ unsatisfiable?

Equivalently

$$\sum_{\substack{b_1,\dots,b_n \\ \in\{0,1\}}} \underbrace{\varphi(b_1,b_2,\dots,b_n)} = 0$$

1 if φ satisfied by $(b_1,\dots,b_n)$

& 0 otherwise.

We will develop a protocol to verify statements of this form.

Step 1: Arithmetization.

Turn φ into a polynomial

$$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_7)$$

$$\downarrow$$

poly. P_φ that behaves like φ on Boolean inputs

$$P_\varphi = \underbrace{P_1}_{\text{from clause 1}} \cdot \underbrace{P_2}_{\text{from clause 2}}$$

(AND is the same as product for Boolean inputs)

For a clause with ORs, use De Morgan's laws to turn OR into AND.

$$x_1 \vee \neg x_2 \vee x_3 \equiv \neg(\neg x_1 \wedge x_2 \wedge \neg x_3)$$

$$\downarrow$$

$$\left(1 - (1-x_1)\cdot x_2 \cdot (1-x_3)\right)$$

[NOT(b) is the same as $1-b$]

In general, if φ has m clauses

$$C_1 \wedge C_2 \wedge \dots \wedge C_m$$

$$P_\varphi = P_1 \cdot P_2 \cdot P_3 \cdots P_m$$

where each P_i looks like the example above.

Obs:

P_φ is a small circuit made up of $O(m)$ additions & multiplications computing a polynomial of degree $\leq 3m$.

Thus, we can evaluate P_φ at any point efficiently.

But we want to know:

Is $\sum_{b_1,\dots,b_n} P(b_1,\dots,b_n) = 0$?

Seems to need 2^n evaluations!

Or does it?

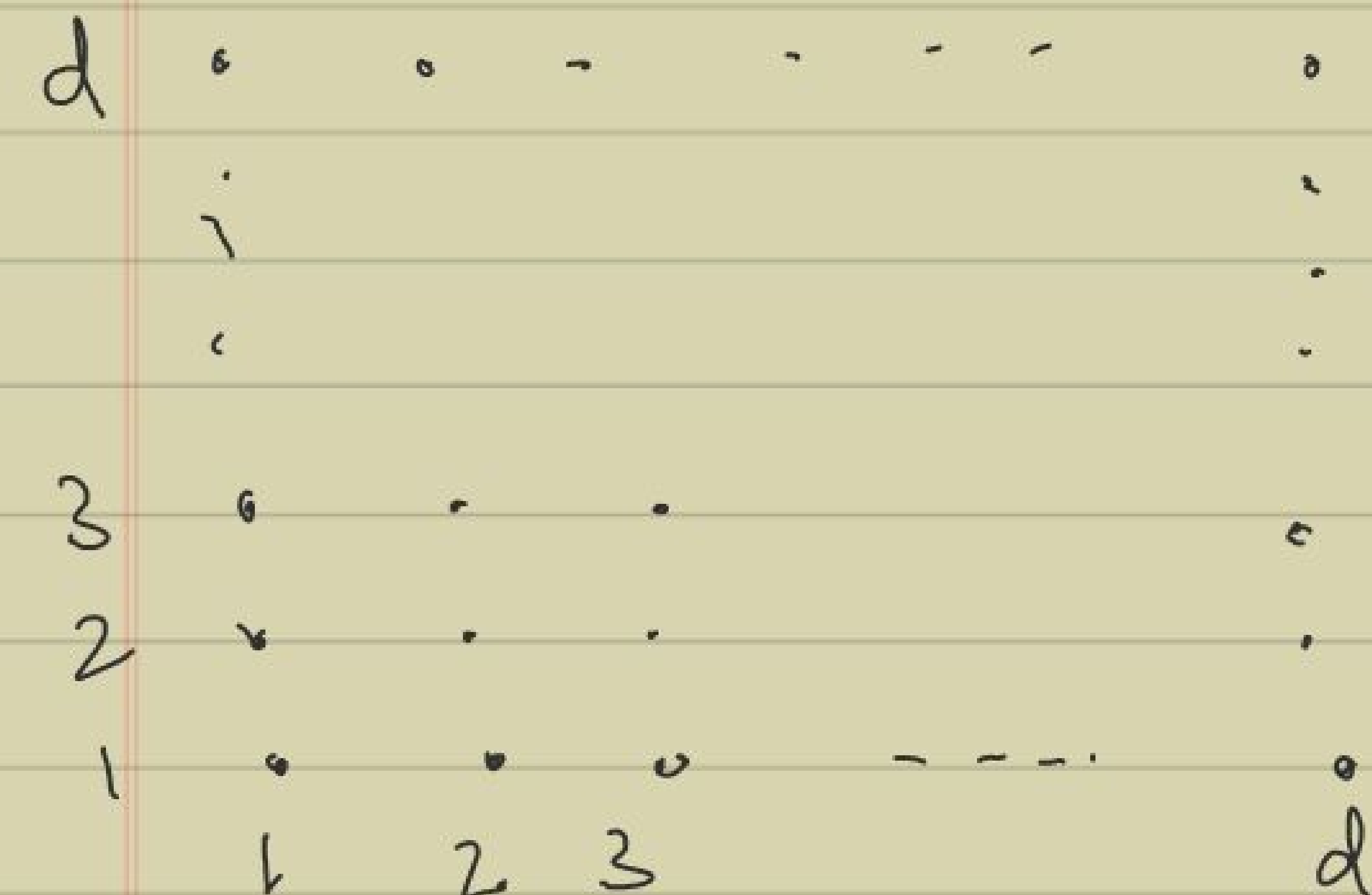
New idea: Can evaluate P at non-Boolean points!

Can that help? YES!

Toy example:

Polynomial $P(X,Y)$ in two variables of degree d & Verifier V wants to check that

$$\sum_{a \in \{1,\dots,d\}} \sum_{b \in \{1,\dots,d\}} P(a,b) = 0$$



Want to sum P at all points in this grid & check if sum $= 0$.

Seems to need d^2 evaluations of P.
Can we do it in fewer?

<u>Protocol:</u>

(1) Prover sends $Q'(x)$ where Q' is a polynomial of degree d & <u>supposedly</u> Q' is the same as $Q(x) = \sum_{b \in \{1,..,d\}} P(x,b)$

(2) Verifier checks:

$$Q'(1) + Q'(2) + \cdots + Q'(d) = 0$$

If not true, Verifier rejects.

(3) If the above check passes, V chooses a random $a' \in \{1,..,10d\}$ & checks

$$Q'(a') \doteq \sum_{b \in \{1,..,d\}} P(a',b).$$

Analysis:

$\boxed{\text{Note that Verifier makes } d \text{ evaluations of } Q' \text{ in step (2) \& } d \text{ evaluations of } P \text{ along with 1 evaluation of } Q' \text{ in step (3). So this requires } O(d) \text{ polynomial evaluations overall.}}$

Correctness: Requires analyzing completeness & soundness.

Completeness: Here we assume that what the prover is trying to show is true. i.e

$$\sum_{a \in \{1,..,d\}} \sum_{b \in \{1,..,d\}} P(a,b) = 0$$

→ Now if the prover indeed sent us the "correct" polynomial. i.e.

if $Q'(x) = Q(x) = \sum_{b \in \{1,\dots,d\}} P(x,b)$

Then $Q(1) + Q(2) + \dots + Q(d)$

$$= \sum_{a \in \{1,\dots,d\}} \sum_{b \in \{1,\dots,d\}} P(a,b) = 0.$$

So the check in step ② succeeds.

→ Moreover, in step ③

$$Q'(a') = Q(a') = \sum_{b \in \{1,\dots,d\}} P(a',b)$$

So this check succeeds as well.
Overall, the verifier accepts with prob 1.

<u>Soundness</u>: Now we assume that

$$\sum_{a \in \{1,..,d\}} \sum_{b \in \{1,..,d\}} P(a,b) \neq 0.$$

→ If the prover sends the "correct" $Q'(x)$

i.e. if $Q'(x) = Q(x)$, then the check in step ② immediately fails & the Verifier <u>rejects</u>.

→ If the check in step ② succeeds, then the prover <u>must</u> have sent

$$Q'(x) \neq Q(x).$$

In this case, by the Schwartz-Zippel lemma with probability 9/10,

$$Q'(a') \neq Q(a') = \sum_{b \in \{1,..,d\}} P(a',b)$$

& the Verifier rejects in step ③.