TOPICS (APRIL 7th).

- Hardness Amplification
- Alternate Characterization of NP.
- Probabilistically Checkable Proofs

<u>Hardness Amplification</u>: Example of IND SET.

<u>Theorem</u>: If INDSET is hard to approximate to a factor of 1.001, then it is also hard to approximate to a factor 1000.

<u>Proof</u>: Tensored graph.

General Technique: "Parallel repetition".

<u>NP as a prover-verifier "game"</u>:

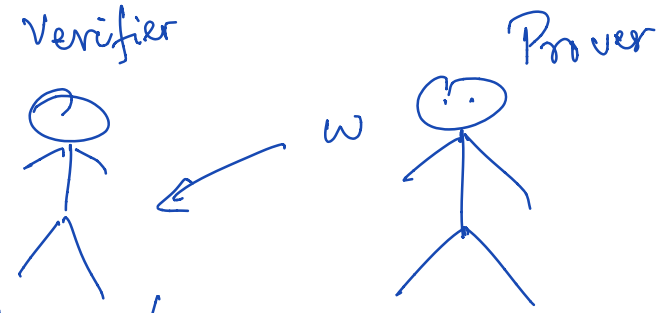
<u>Classic definition</u>: a language L is said to be in NP if $\exists$ an algorithm A s.t.

1) $\forall x \in L$, $\exists$ a witness w (of poly size) such that $A(x, w) = 1$; and

2) $\forall x \notin L$, there is no w s.t. $A(x, w) = 1$.

A is a "witness verifier.

E.g. for SAT, A could simply check if w is a satisfying assignment.



- has ϕ
- runs $A(\phi, w)$ and accepts if w satisfies all clauses..

<u>PROBABILISTIC CHECKING</u>: Verifier does not check all clauses. He simply checks one clause at random (lazy verifier)

KEY: Prover does not know which clause verifier will check!

Best strategy for prover: provide an assn. that satisfies as many clauses as possible..

$\Updownarrow$

Prover solves max-SAT!

Success probability $\equiv \frac{1}{m} \cdot OPT(\text{max-SAT}(\phi))$

m → total # of clauses.

Note: The Lazy prover above:

- Uses $O(\log m)$ "bits of randomness"
- Probes at most 3 bits of proof
- Accepts w.p. $\begin{cases} 1 & \text{if } \phi \text{ is SAT} \\ 1 - 1/m & \text{if } \phi \text{ is not SAT.} \end{cases}$

assuming the witness is correct

DEFINITION: An (r, q)-verifier is one that uses r bits of randomness, and queries q bits of the proof before accepting/rejecting.

[it could do an arbitrary polynomial amt of computation involving ϕ and these bits]

DEFN:

A language L has a PCP(r, q) if there exists an (r, q) verifier A s.t.:

- $\forall x \in L$, there exists a w s.t.
$$\Pr[A(x, w) = 1] = 1.$$
- $\forall x \notin L$, there is no 'w' s.t.
$$\Pr[A(x, w) = 1] \geq \frac{1}{2}.$$
(equivalently, $\forall w$, $\Pr[A(x, w)] < \frac{1}{2}$)

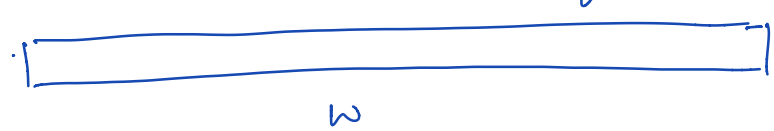
PCP Theorem: (ALMSS 92).

Every language $L \in NP$ has a PCP$(O(\log n), O(1))$.

"$\exists$ a way of writing proofs s.t. even a lazy prover "succeeds" w.p. $\geq \frac{1}{2}$" [Error correction "built-in"]

<u>We now show</u>: PCP Theorem $\Rightarrow$ 3-SAT is hard to approximate to a factor $1+\varepsilon$, for some const $\varepsilon > 0$ (depends on the $O(1)$ term in $PCP(\ldots, \ldots)$).

<u>Proof</u>: Let L be some NP-complete language, and consider an $(O(\log n), \underbrace{O(1)}_{q})$ verifier.



- The algorithm can possibly only query $2^{O(\log n)} \equiv n^{O(1)}$ bits of w [no matter what the random bits are].

- It makes its decision based on q of these bits. I.e., based on whether:

$$C_{\phi, R}(w_{i_1}, w_{i_2}, \ldots, w_{i_q}) = 1, \text{ for some } C$$

- We can encode $C_{\phi, R}(\ldots)$ as a SAT formula of size $2^q \rightsquigarrow$ 3SAT of size $\approx q \cdot 2^q \equiv O(1)$.

<u>Thus</u>: since we start with a good verifier, we get that

- If $x \in L$, "final" 3-SAT formula is satisfiable.
- If $x \notin L$, final 3-SAT formula has at least a $\underbrace{\frac{1}{2} \cdot \frac{1}{q \cdot 2^q}}_{\equiv\, \varepsilon}$ fraction of clauses being unsat.

[Completes the proof] ∎