2 Apr 2021

Announcement

CS research night Apr. 8 5:30-7:30 over Discord.

https://discord.com/invite/cPGHKGqD

Learn about research in CS at Cornell, opportunities to get involved.

---

Def: A decision problem $A$ belongs to $NP$ if $A \leq_p 3SAT$.

$A$ is $NP$-hard if $3SAT \leq_p A$.

$A$ is $NP$-complete if both $NP$ and $NP$-hard.

$$A \leq_p 3SAT \leq_p A$$

$A \equiv_p 3SAT$

---

$NP$ and efficient verification

Problems like 3SAT and (indep. set) have the property that (we believe) it’s hard
to find a solution but easy to verify a solution is correct.

Ex. in 3SAT if someone tells you a truth assignment of \( x_1, \ldots, x_n \)
it's easy to check every clause is satisfied.

In Ind set if someone gives you a set of vertices in a graph, it's easy to check there are \( k \) of them all distinct, no edges between them.

**Def.** Decision problem \( A \) belongs to \( NP \) if and only if there exists a poly-time algorithm ("verifier") \( V \) with two inputs \( x, y \) ("problem" & "solution") such that

\[
A(x) = \text{true} \iff \exists y \text{ s.t. } |y| \leq \text{poly}(|x|) \quad \text{and} \quad V(x, y) = \text{true}
\]

**Theorem.** (Cook-Levin) The two definitions of \( NP \) given above are equivalent. \( A \) has a poly-time verifier if and only if \( A \leq_p 3\text{SAT} \).
Observation. Problem $A$ is NP-Hard if and only if there is an NP-Hard $B$ s.t. $B \leq_{p} A$.

Proof. Def of NP-Hard is $3\text{SAT} \leq_{p} A$.

So if there is an NP-Hard $B$ s.t. $B \leq_{p} A$ then $3\text{SAT} \leq_{p} B \leq_{p} A$, by transitivity $3\text{SAT} \leq_{p} A$.

$\therefore$ $A$ is NP-Hard. Conversely if $A$ is NP-Hard then $A \leq_{p} A$ so

$\exists$ an NP-Hard $B$, namely $B = A$, satisfying $B \leq_{p} A$.

Practical advice. When attempting to show $A$ is NP-Hard, it helps to pick a known NP-Hard $B$ that has some similarity to $A$ and try reducing from $B$ to $A$.

Ex. If $A$ says "Does there exist a structure that satisfies a list of constraints?" try $3\text{SAT} \leq_{p} A$.

If $A$ says "Does there exist a set of at least $k$ things that satisfies some constraints?" try $\text{IND SET} \leq_{p} A$.
Similarly “at most k” means try
\[ V_{\text{TX COVER}} \leq P \leq A. \]

Known NP-Hard problems at this point in 4520:

- 3SAT
- INDEP SET
- CLIQUE
- VTX COVER

Some new problems coming from set theory.

Given a set \( U \) with \( n \) elements and a collection of subsets \( S_1, S_2, \ldots, S_m \subseteq U \) and a positive integer \( k \)...

**SET COVER:** Can you find \( k \) of the sets whose union is \( U \)?

**SET PACKING:** Can you find \( k \) of the sets that are all disjoint from each other?
Reducing \textsc{Indep Set} to \textsc{Set Packing}:

- Given $G = (V, E)$ and $k > 0$ (an instance of \textsc{Indep Set})
- Construct $U = E$.
- For each $v \in V$ create set $S_v$ consisting of all edges that have $v$ as an endpoint.
- Use the same $k$.

Does this work?

$(\Rightarrow)$ if $G$ has a $k$-element Indep set, $I$,
then there are $k$ mutually disjoint subsets in our \textsc{Set Packing} instance.
Yes! \{ $S_v \mid v \in I$ \}.

$(\Leftarrow)$ if \textsc{Set Packing} has $k$ mutually disjoint sets $S_{v_1}, S_{v_2}, \ldots, S_{v_k}$
does $G$ have a $k$-element Indep set?
Yes! \{ $v_1, \ldots, v_k$ \}.

The same reduction reduces \textsc{Vertex Cover} to \textsc{Set Cover}. 
Don't reduce in the wrong direction!

E.g., when trying to show SET COVER NP-Hard, don't start by reasoning "Let's say I had a set cover problem. How would I transform it into a graph?"

Instead say "Let's say I had a graph representing a VTR COVER problem. How would I transform it into SET COVER?"