Algorithm Analysis and Design (CS1.301)

Monsoon 2021, IIIT Hyderabad
16 October, Saturday (Lecture 15)

Polynomial-Time Reductions

The Class NP

The class **NP** consists of polynomial-time verifiable languages. A *verifier* for a language A is an algorithm V where

$$A = \{w \mid V \text{ accepts } \langle w, c \rangle \text{ for some string } c\}.$$

Such a string c is called a *certificate*.
A polynomial-time verifier therefore runs in polynomial time in the length of w.

For example, the language CLIQUE $= \{\langle G, k \rangle \mid G$ is an undirected graph with a $k-$clique$\}$ has a polynomial-time verifier – the certificate would be the clique itself.

The class **coNP** consists of languages whose *complements* are polynomial-time verifiable.

Polynomial-Time Reducibility

A function $f : \Sigma^* \to \Sigma^*$ is a polynomial-time computable function if a TM can compute it in polynomial-time.

A language A is polynomial-time mapping reducible to a language B (written $A \leq_P B$) if a polynomial-time computable function f exists such that $\forall w, w \in A \iff f(w) \in B$.
f is called the polynomial-time reduction of A to B.
Note that if $A \leq_P B$ and $B \in P$, then $A \in P$.

For example, 3SAT $= \{\langle \phi \rangle \mid \phi$ is a satisfiable 3cnf formula$\}$ is polynomial-time reducible to CLIQUE.

To prove this, let $\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$. We need a reduction f that generates $\langle G, k \rangle$, where G is an undirected graph.
The nodes of G are organised into k triples, each of which corresponds to a

clause in ϕ. Each node then corresponds to a literal in the clause.
No edge connects nodes of the same triple, and no edge connects two nodes which are complements of each other. All other pairs of edges are connected by edges.

If ϕ has a satisfiable, each clause has a true literal. Select a node corresponding to a true literal from *each* triple. This set of nodes forms a clique.

NP-Completeness

A language B is NP-complete if it is in **NP** and every $A \in \mathbf{NP}$ is polynomial-time reducible to B. paid

We can see that if B is NP-complete and $B \leq_P C$ for $C \in \mathbf{NP}$, then C is NP-complete.

According to the Cook-Levin Theorem, 3SAT is NP-complete.