A (decision) problem is in NP if it has a polynomially checkable proof (also called a certificate).

E.g. HAM (Hamiltonian circuit): Does a graph have a cycle visiting every vertex exactly once? Certificate: The Hamiltonian cycle

Certificate: The coloring

Fact: P \subseteq NP

Proof: Run poly-time algo to verify solution.

A problem A is said to be polynomially reducible to B (denoted A \leq B) if given an instance of problem A, we can produce an instance of problem B s.t. there is a polynomial time algorithm that can decide the instance given a decision on the instance of B.

\[ \text{Instance of } A \xrightarrow{\text{Poly}} \text{Instance of } B \xrightarrow{\text{Oracle}} \text{YES/NO for } B \xrightarrow{\text{Poly}} \text{YES/NO for } A \]

E.g., Bipartite Matching \leq Max Flow

\[ \text{Maxflow} \geq k \]

\[ \text{Matching} \geq k \]

We will use reductions to establish hardness. If A reduces to B (A \leq B), then B is at least as hard as A (upto a polynomial).

A problem is said to be NP-hard if all problems in NP reduce to the problem and NP-complete if it additionally belongs to NP.

Theorem (Cook-Levin): SAT \leq NP-complete.

\[ \text{SAT} : (x_1 \lor \neg x_2 \lor x_3) \land (x_2 \lor x_3) \land (x_4 \lor \neg x) \]
Is this formula (in CNF) satisfiable?

**Reductions**

If \( A \leq B \) and \( A \) is \( NP \)-hard, then so too is \( B \).

**Examples:**

1. \( SAT \leq \) \( 3\)-\( SAT \)

\[(x_1 \lor x_2 \lor x_3 \lor \ldots \lor x_k) \text{ is satisfied by an assignment if and only if there exists a setting of variables } x_1, x_2, \ldots \text{ such that}\]

\[(x_1 \lor x_2 \lor x_1) \land (x_1 \lor x_2 \lor x_2) \land (x_1 \lor x_2 \lor x_3) \land \ldots \land (x_{k-1} \lor x_k, x_k)\]

is satisfied.

2. \( SAT \rightarrow \) Integer programming

\[(x_1 \lor x_2 \lor x_3) \rightarrow x_1 + x_2 + (1-x_3) \geq 1\]

3. \( SAT \rightarrow \) Independent set

\[(x_1 \lor x_2 \lor x_3) \land (\neg x_1, \lor x_2)\]

\(\text{Independent set } \geq m \text{ (#of clauses)}\)

\(\uparrow\)

\(\text{SAT formula is satisfiable}\)

\[\text{Independent set } \geq m \text{ (#of clauses)}\]

\(\text{Independent set } \geq m \text{ (#of clauses)}\)

\(\uparrow\)

\(\text{SAT formula is satisfiable}\)

4. Independent set \(\rightarrow\) Clique

\(G\) has independent set of size \( k \) if and only if \( \overline{G} \) has clique of size \( k \).

5. Independent set \(\rightarrow\) Vertex cover

\(G\) has independent set of size \( k \) if and only if \( G\) has vertex cover of size \( N \)-\( k \).

\((C \text{ is an independent set } \iff V - C \text{ is a vertex cover)}\)

6. Vertex cover \(\rightarrow\) Dominating set

Place a vertex on every edge; size of dominating set in new graph equals the size of a vertex cover in original graph.

**Approximation Algorithms**

An \( \alpha \)-approximation algorithm for a minimization (resp. maximization)
An $\alpha$-approximation algorithm for a minimization (resp. maximization) problem is guaranteed to produce a solution $\text{ALGO}$ of value $\leq \alpha \cdot \text{OPT}$ (resp. $\geq \frac{\text{OPT}}{\alpha}$).

Examples:

1. **Vertex cover**: pick both ends of an edge, remove all incident edges, and repeat. 2-approx (best known?)

2. **Set cover** (generalizes vertex cover, hence $\text{NP}$-hard).

   - **Sets** $S \subset U$. Find min. collection of sets that covers all $U$.
   - **Greedy Alg**: generically think VC alg. pick set with max new
     elements covered.
   - **Analysis**: suppose $k$ elements left to be covered. Then cost
     per element is at most $\text{OPT}/k$.
     
     $$\Rightarrow H_k - \text{approx} = O(\log n) - \text{approx}.$$ 

3. **Max coverage**: given a budget of $k$ sets, how many elements
   can we cover?

   - **Greedy Alg**: pick set that maximizes # of new elements covered.
   - **Analysis**: if $\text{OPT} - \text{ALGO} = t_i$ currently, then $\text{Set}$ that covers $\frac{t}{k}$
     elements, i.e., $t_{i+1} \leq t_i (1 - \frac{1}{k})$

     Thus, $t_k \leq t_{k-1} (1 - \frac{1}{k}) \leq \ldots \leq t_0 (1 - \frac{1}{k})^k = \text{OPT} (1 - \frac{1}{k})^k$

     As $k \to \infty$, $t_k \leq \frac{1}{e} \cdot \text{OPT}$

     $$\Rightarrow \left(1 - \frac{1}{e}\right) - \text{approx}.$$ 

4. **Metric TSP**: Walk on spanning tree so that each edge is traversed
   at most twice - 2-approx