NP and NP-complete

So far we have been focusing on algorithms. What about hardness?

Example: Any comparison-based sorting algorithm must take $\Omega(n \log n)$ time.

However, there are many problems in a gray area:

- we do not know poly-time algorithms.
- we cannot prove there are no poly-time algorithms.

A set of problems that are "equivalent":

if there exists poly-time algorithm for one of them,
then there exist poly-time algorithms for all of them.

Decision vs. optimization:

Example: What is the length of the shortest path from $s$ to $t$? vs.
Is there a path from $s$ to $t$ with length $\leq k$?

we are going to work with the decision version.

Polynomial-Time Reductions.


Poly-time reduction: For two problems $X$ and $Y$, we say $X \leq_p Y$

read as "$X$ is polynomial-time reducible to $Y"
"$X$ is at most as hard as $Y$" (w.r.t. poly-time).

iff there is a poly-time algorithm that can transform instances of problem $X$
into instances of problem $Y$ such that the answers to both instances are always the same.

Corollary: If $X$ can be solved in poly-time, then so can $Y$. 
Example of \( \leq_p \):

**Vertex Cover**: given a graph \( G \) and an integer \( k \), decide if \( G \) has a vertex cover of size \( \leq k \).

**Definition (vertex cover)**: a set \( S \) of vertices such that every edge has at least one endpoint in \( S \).

**Independent Set**: given a graph \( H \) and an integer \( \ell \), decide if \( H \) has an independent set of size \( \geq \ell \).

**Definition (independent set)**: a set \( S \) of vertices such that there are no edges between vertices in \( S \).

**Theorem**: \( \text{Vertex Cover}(VC) \leq_p \text{Independent Set (IS)} \)

\[
G =
\begin{array}{ccccc}
1 & 2 & 3 & 4 & 5 \\
\end{array}
\]

**Proof**: we show that \( S \) is a vertex cover if and only if \( V-S \) is an independent set.

\( S : VC \) is a VC
\( V-S : IS \) is an IS

- every edge must have \( \geq 1 \) endpoint in \( S \)
- no edge has both endpoints in \( V-S \)

\( VC \leq_p IS \):

we are given a VC instance \( (G, K) \) -> a reduction

\[ H \leftarrow G \]
\[ \ell \leftarrow n-k \]

an IS instance \((H, \ell)\) that always has the same answer.
P vs. NP.

Both are complexity classes.

P: all decision problems that can be solved (deterministically) in polynomial time.

NP: all decision problems for which the "yes" instances have proofs that be verified in (deterministic) polynomial time.

Example: (the decision version of)

- MST
- Shortest path
- MaxFlow
- Sorting

E P.

( the decision version of )

- Vertex cover
- Independent set

E NP.

Theorem: Vertex Cover E NP.

Proof: If Peggy wants to convince Victor the answer to VCG, k) is "yes" Peggy can simply show a vertex cover of size \( \leq k \).

proof: \( S \subseteq V \) with \( |S| \leq k \)

verification: Victor can check every edge \( e \in E \) to see if \( e \) has at least one endpoint in \( S \).

Theorem: \( P \subseteq NP \).

Proof: proof = \( \emptyset \).

verification: Solve the instance and verify the answer is yes.

Open Question: \( P = NP \)?
NP-Complete (NPC) All decision problems $X$ such that:

1. $X \in \text{NP}$
2. $\forall Y \in \text{NP}, Y \leq_p X$

Intuitively, NPC contains the hardest problems in NP.

Lemma. If $X \in \text{NPC}$, $Y \in \text{NP}$, and $X \leq_p Y$, then $Y \in \text{NPC}$.

Proof. ① $Y \in \text{NP}$.
② $\forall Z \in \text{NP}, Z \leq_p X$ (because $X$ is NPC), then $Z \leq_p Y$ (if $Z \leq_p X$ and $X \leq_p Y$, then $Z \leq_p Y$)

An example of NP-complete problems: 3-SAT.

3-SAT. Input: $n$ boolean variables $X_1, \ldots, X_n$, $m$ clauses $C_1, \ldots, C_m$.

Output: "YES" iff there is an assignment $X_i \rightarrow \{0,1\}$ that satisfies all clauses simultaneously.

(Each clause is the OR "$\lor$"

of three literals.

Literal: $X_i$, $\overline{X_i}$ = "not $X_i$"

Example: $n=4$, $m=3$

$C_1 = X_1 \lor \overline{X_2} \lor X_3$
$C_2 = \overline{X_1} \lor X_3 \lor X_4$
$C_3 = \overline{X_1} \lor \overline{X_3} \lor X_4$

Answer = "YES", one solution is $X_1 = 1$, $X_3 = 0$

Problems harder than NP?

Generalized Chess & NP.

Input: An $n \times n$ chess board position (with natural generalized chess rules)

Output: Black moves first. Output "YES" iff Black can win.