P and NP.

**Definition**

\[ P = \{ \text{decision problems solvable in poly. time}\} \]

We will study what is/is not in this class.

Careful of

- machine model - log cost RAM
- input size - # bits

Recall from last day:

\[ A \leq_p B \text{ for problems } A, B \quad \text{“} A \text{ reduces to } B \text{”} \]

means: we can use a poly. time algorithm for B
to give a poly. time algorithm for A.

Example.

Hamiltonian cycle/path = cycle/path
that visits every vertex exactly once

This graph has a Hamiltonian path
but not a Hamiltonian cycle.

**Lemma** Hamiltonian path \( \leq_p \) Hamiltonian cycle.

**Pf.** Suppose we have a poly. time algorithm for Hamiltonian cycle.

We want to design a poly. time algorithm for Hamiltonian path.
Input: graph $G$.
Algorithm:
- Construct graph $G'$ by adding one new vertex adjacent to all vertices of $G$.
  \[ G' \] \[ \hat{G} \] \[ v \]
- Send $G'$ to the algorithm to test for Hamiltonian cycle.
- Return the YES/NO answer.

This algorithm runs in polynomial time.

Correctness: must prove

Claim: $G$ has a Hamiltonian path iff $G'$ has a Hamiltonian cycle.

Proof ($\Rightarrow$): Suppose $G$ has a Hamiltonian path $x$ to $y$.
Adding $v$ and edges $(x,v)$, $(v,y)$ gives a Hamiltonian cycle in $G'$.

($\Leftarrow$): Suppose $G'$ has a Hamiltonian cycle. Removing $v$ gives a Hamiltonian path in $G$.

**Lemma** Hamiltonian cycle $\leq_p$ Hamiltonian path

Ex. prove this.

**Fact**: no one knows a polynomial time alg. for either problem.
There is a large class of decision problems not known to be in \( P \) but all equivalent in the sense that 
\( A \leq_p B \) for all \( A, B \) in the class. (recall defn of \( \leq_p \))
i.e. poly-time algo. for one yields poly-time algo. for all.

A few problems in the class:

- Hamiltonian path/cycle
- TSP - given edge weighted graph, number \( k \),
  is there a TSP tour of weight \( \leq k \)?
- IND. SET - given graph, number \( k \),
  is there an ind. set of size \( \geq k \)?

Common feature: if the answer is YES there is some succinct info. to verify it.
"certificate"

(in particular, the TSP tour, the ind. set)

Contrast this with NO answer.
A verification alg. takes input + certificate and checks it.

Definition Alg. $A$ is a verification alg. for problem the decision problem $X$ if

- $A$ takes two inputs $x$, $y$ and outputs YES or NO
- for every input $x$ for problem $X$, $x$ is a YES for $X$ iff there exists a $y$ “certificate” s.t. $A(x, y)$ outputs YES.

Furthermore, $A$ is a polynomial time verification alg. if

- $A$ runs in poly. time
- there is a polynomial bound on the size of the certificate, i.e.,
  \[ \forall x, x \text{ is a YES input for } X \text{ iff } \exists y \text{ with size}(y) \leq (\text{size}(x))^k, \text{ k const.} \text{ s.t. } A(x, y) \text{ outputs YES} \]

$NP = \{ \text{decision problems that can be verified in polynomial time}\}$

Example Subset Sum $\in NP$

Given numbers $w_1, \ldots, w_n$ and $W$

is there a subset $S \subseteq \{1, \ldots, n\}$ s.t. $\sum_{i \in S} w_i = W$
certificate is $S$
 verification alg: check that $\sum w_i = N$
i.e.$ S$
poly. time

Is there a poly. time verification alg. for NO answers?
What could you give to verify that no subset has
sum $W$?  OPEN

Example  TSP (decision version) $\in$ NP
Given graph $G$, weights on edges, number $k$,
does $G$ have a TSP tour of length $\leq k$?
certificate: the tour, i.e. permutation of vertices
poly. time verification alg:
- check it's a permutation
- check that edges exist
- check that $\sum$ edge weights in tour $\leq k$

coNP = $\{\text{decision problems where the NO instances can be verified in poly. time}\}$

e.g. Primes: given number $n$, is it prime?
Primes $\in$ coNP

easy: to verify $n$ is not prime, show
natural numbers $a, b \geq 2$ s.t. $a \cdot b = n$
OPEN QUESTIONS
1. $P = ? \ \text{NP}$
2. $\text{NP} = ? \ \text{coNP}$
3. $P = ? \ \text{NP} \cap \text{coNP}$

Properties
- $P \subseteq \text{NP}, \quad P \subseteq \text{coNP}$
- any problem in $\text{NP}$ can be solved in time $O(2^n)$ by trying all certificates one by one
Definition: A decision problem $X$ is NP-complete if

- $X \in \text{NP}$
- for every $Y \in \text{NP}$, $Y \leq_p X$

i.e., $X$ is [one of] the hardest problems in NP.

Two important implications of $X$ being NP-complete:

- if $X \in \text{P}$ then $\text{P} = \text{NP}$
- if $X$ cannot be solved in poly-time then no NP-complete problem can be solved in poly-time
- if $X \in \text{coNP}$ then $\text{NP} = \text{coNP}$ (this needs proof)

The first NP-completeness proof is difficult.

Subsequent NP-completeness proofs are easier because $\leq_p$ is transitive.

$Y \leq_p X$ and $X \leq_p Z$ implies $Y \leq_p Z$.

So to prove $Z$ is NP-complete we just need to prove $X \leq_p Z$ where $X$ is a known NP-complete problem.
Summary: To prove \( \mathcal{L} \) is \( \text{NP-complete} \)
1. prove \( \mathcal{L} \in \text{NP} \)
2. prove \( X \leq_P \mathcal{L} \) for known \( \text{NP-complete} \) problem \( X \)

Our first \( \text{NP-complete} \) problem: Circuit Satisfiability
(proof later - also definition)

2nd \( \text{NP-complete} \) problem: Satisfiability
(proof later, but will define the problem now)

Satisfiability
Input: a \text{Boolean formula} made of \text{Boolean variables}, \( \land "\text{and}" \), \( \lor "\text{or}" \), \( \neg "\text{not}" \),
e.g. \( \neg(x_1 \land x_2) \lor (x_3 \land (x_5 \lor \neg x_4)) \)

Question: Is there an assignment of True/False to the variables to make the formula True?

Ex. Satisfiability \( \in \text{NP} \).
Sat is \( \text{NP-complete} \), even the special form from
Assign 4, "CNF" - Conjunctive Normal Form

e.g. \( (x_1 \lor \neg x_2 \lor x_3) \land (x_3 \lor x_4) \land (x_3 \lor x_4 \lor \neg x_5) \)

- clause
is \( \lor \) of literals

- formula is \( \land \) of clauses
In fact it’s still NP-complete when all clauses have 3 literals — 3-SAT
but 2-SAT is in P

3-SAT
Input: A Boolean formula that is an \( \land \) of clauses,
each clause an \( \lor \) of 3 literals, each literal
a variable or negation of variable.
Question: Is there an assignment of True/False to
variables that makes the formula True?

\[ \text{III} \] 3-SAT is NP-complete [pf. later].

Ind. Set
Input: Graph \( G = (V, \varepsilon) \), number \( k \)
Q: Does \( G \) have an independent set of size \( \geq k \)
i.e. a set \( S \subseteq V \) s.t.
there is no edge \((u,v)\) with \( u,v \in S \)

\[ \text{Thm} \] Ind. Set is NP-complete
pf. 1. Ind. Set \( \in \text{NP} \) — we saw this already
2. 3-SAT \( \leq_p \) Ind. Set.
Suppose we have a (black box) poly-time alg. for Ind. Set,
give a poly-time alg. for 3-SAT
Input: 3-SAT formula \( F \)
clauses \( C_1, \ldots, C_m \), variables \( x_1, \ldots, x_n \)
$C_i = (l_{i_1} \lor l_{i_2} \lor l_{i_3})$

Create a graph $G$ on vertices $l_{ij}$ $i=1\ldots m$ $j=1,2,3$

Join literals in a clause

Join literals that are negation of each other e.g. $(x_1 \lor \neg x_2 \lor x_3) \land (x_1 \lor x_2 \lor \neg x_3)$

$G$ has poly. size — $3m$ vertices

Claim $G$ has an Ind. Set of size $\geq m$ if $F$ is satisfiable

Thus our 3-SAT alg. is

- construct $G$
- run Ind. Set alg. on $G$, $m$
- output the YES/NO answer.

This alg. runs in poly. time.

Proof of Claim

$\Leftarrow$ Suppose $F$ is satisfiable

Pick one vertex from each $\Delta$ corresponding to a True literal. Gives Ind. set of size $m$
Suppose $G$ has ind. set $S$ of size $m$. $S$ can only have one vertex from each $\Delta$. $S$ cannot use $x$ and $\neg x$.

Thus we can set all literals in $S$ True and this satisfies the formula. (If a variable isn't set by $S$ (i.e. neither $y$ nor $\neg y$ in $S$), then can set it arbitrarily.

Ex. Carry out this construction on an example.