**NP-Completeness:**

**Def:** An algorithm runs in polynomial time if its (worst-case) running time is \( O(n^c) \) for some constant \( c \) (where \( n \) = size of input).

**Decision problem:** output is either YES or NO.

**Ex:** Max flow as decision problem

"Is there a feasible flow of value \( \geq k \)?"

**Def:** \( \mathcal{P} \) is the set of decision problems solvable in polynomial time.

**Different setting:** if we’re given a solution, we can easily check if it’s valid (feasible).
Max-flow: check if flow, cap is obeyed, value $\geq k$

3-coloring: Given undirected $G=(V,E)$.
- *YES* if $\exists f: V \rightarrow \{0,\ 0,\ 1\}$ s.t.
  \[
  f(-) \neq f(v) \quad \forall \{u,v\} \in E
  \]
- *NO* otherwise

given $f$, check if valid: check colors of endpoints

**Def:** Decision problem $Q$ is in $NP$ if
- exists polynomial time algorithm $VC(I,X)$ (verifier) s.t.
  1) If $I$ is a *YES*-instance of $Q$, then $\exists X$ with $|X|$ polynomial in $|I|$ s.t. $VC(I,X)=YES$
  2) If $I$ is a *NO*-instance of $Q$, then $VC(I,X)=NO \ \forall X$

**Examples:**
- 3-coloring: witness: coloring $f$
verifier: checks each edge

Max-flow: witness: flow

verifier: check flow conservation, capacity, value 2k

Factoring: given integers Mk. YES if M has a factor in \( \{2, \ldots, k\}\), NO otherwise

witness: factor in \( \{2, \ldots, k\}\)

verifier: compute \( M(f, \text{check if integer} \)

Important asymmetry: witness for YES instance, not NO instances

Thm: \( P \leq NP \).

pf: Let \( x \in P \). Ignore \( X \), solve for instance.
Q: P = NP?

**NP-Completeness and Reductions**: How to prove \( P = NP \)?

**Def**: Problem \( A \) is **polytime reducible to** \( B \) \( (A \leq_p B) \) if, given polytime algorithm for \( B \), we can use it to produce a polytime algorithm for \( A \).
Note: If $A \leq_p B$, then if $B$ is in $\mathcal{P}$, so is $A$.

$\implies$ $B$ is “at least as hard” as $A$.

**Def:** A many-one reduction from $A$ to $B$ is a function $f$ which takes arbitrary instances of $A$ and turns them into instances of $B$ s.t.

1) $x$ is YES of $A \implies f(x)$ YES of $B$
2) $x$ NO of $A \implies f(x)$ NO of $B$
3) $f$ can be computed in polytime

**Def:** Problem $Q$ is NP-hard if $Q_1 \leq_p Q$ & $Q_1 \in \text{NP}$

**Def:** Problem $Q$ is NP-complete if $Q$ is NP-hard and $Q \in \text{NP}$. 

\[ P \subset \text{NP} \subset \text{NP-complete} \]
Circuit - SAT:

Given boolean circuit with 1 output, no less/cycles, is there a way of setting inputs so output is 1?

Gates: AND

OR

NOT

Arbitrary fan-out
Then: Circuit-SAT is NP-complete.

In NP: Check if circuit outputs 1 on input X

NP hard:

Let A \in NP. Want to show A \leq P Circuit-SAT.

What do we know about A?

Has verifier algorithm V.

Algorithm: runs on a computer (or Turing machine).

Computer: Circuit + Memory!
Get rid of loops?

V is poly time!
Final reduction: given input I of A, construct circuit for V, hardware I.

If I YES: \exists x s.t. UC(I, x) = YES
  \Rightarrow circuit outputs 1
  \Rightarrow YES of Circuit-SAT

If circuit YES of Circuit-SAT \Rightarrow
  \exists x s.t. circuit outputs 1
  \Rightarrow UC(I, x) = YES \Rightarrow YES of A.