Reductions

goal: relate runtimes of problems A and B.

To solve decision problem A:

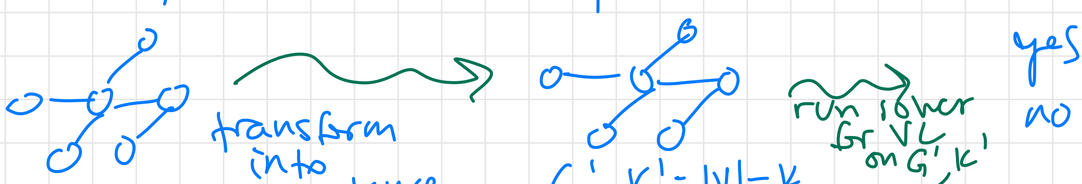
- transform an instance of A into an instance of B
- run solver (algorithm) for B on the instance
- return answer as answer to A

If transformation is polynomial time ($O(n^c)$) then A reduces to B

$$A \le_p B$$

- B can be solved no faster than A can be solved
- If we give a fast alg. for A, we have a fast alg. for B

eg Independent Set $\le_p$ Vertex Cover



G, k

$\exists$ IS of size k in G?

$G', k' = |V| - k$

$\exists$ VC of size k' in G'?

P: set of decision problems solvable in poly time

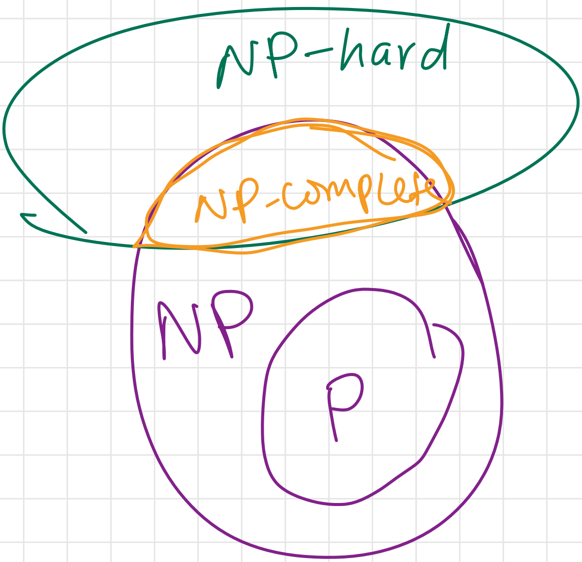
NP: set of decision problems verifiable in poly time

NP-hard: problem is at least as hard as every problem in NP.

$B \geq$ every prob in NP

B is NP-hard: for every $A \in NP$, $A \leq_p B$

NP-complete:
- NP-hard
- in NP



assuming $P \neq NP$

To show a problem C is NP-hard give a reduction from a known NP-hard problem to that problem

$B \leq_p C$

B: known NP-hard prob
C: your prob

SAT
IS
VC

NP-Complete probs speed dating

in pair, make sure your partner:

1. understands problem
   input
   desired output

2. understands how to verify a "yes"/TRUE in poly time

when both partners understand other's 1, 2, hold up hands and find new pairs.