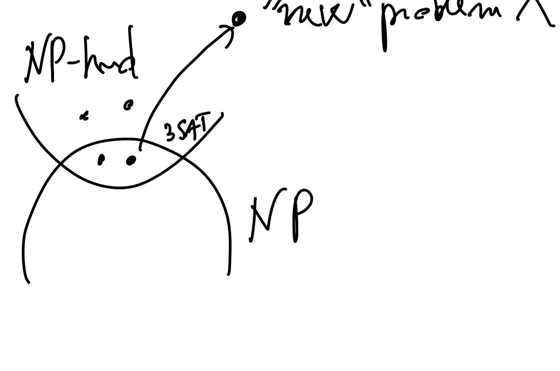
Definition : a problem A <u>reduces in polynomial time</u> to problem B $(A <_p B)$ if $\exists$ a polynomial-time algorithm that transforms an arbitrary input instance a of A into an input instance b of B such that

1) a is a YES instance of A $\Rightarrow$ b is a YES instance of B
2) b is a YES instance of B $\Rightarrow$ a is a YES instance of A

Property: $A <_p B$ and $B <_p C \Rightarrow A <_p C$

NP-hardness (formal definition) : a problem is <u>NP-hard</u> if every problem in NP reduces in poly. time to it.

Then, to prove that a problem X is NP-hard reduce a known NP-hard problem Y to X



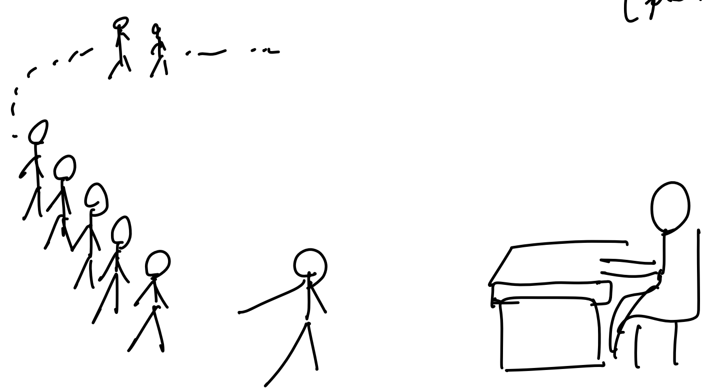
Let's emphasize: the reduction is FROM $Y \to$ I already know it's NP-hard

TO $X \to$ the "new" problem

rookie mistake: do a reduction in the wrong direction

Again, NP-hardness <u>doesn't</u> mean the problem is not in P, but it <u>does</u> provide strong evidence for that:

(from Garey & Johnson)



"I can't find an efficient algorithm, but neither can all these famous people."

Library of NP-hard problems

$\boxed{\text{3SAT}}$ $\leftarrow$ Hamiltonian Circuit

Karp '72

21 NP-hard problems

our first NP-hardness proof:

Theorem: TSP is NP-hard

Proof: reduction from Hamiltonian Circuit to TSP

$(\text{Ham.} <_p \text{TSP})$

but TSP is not a decision problem! No problem:

TSP: input: $G=(V,E)$ complete, undirected, weighted

$K \in \mathbb{R}$

output: $\exists$ in G a Hamiltonian Circuit of cost $\leq k$?

pick an arbitrary input instance for Ham.; create the following input for TSP: $\longrightarrow G=(V,E)$

$G' = (V, E')$ complete, undir., weighted

with

$$W(e \in E') = \begin{cases} 1 & \text{if } e \in E \\ \infty & \text{otherwise} \end{cases}$$

$$K = n$$

this red. takes poly-time $(O(n^2))$

Then

1) if G has a Ham. circuit, then the TSP algorithm run on G' returns a Ham. circuit of cost n

2) if G doesn't have a Ham. circuit, then any Ham. circuit in G' must have ≥ 1 edge not in G, hence of weight ∞. Hence in this case a TSP alg. run on G' returns a Ham. circuit of cost $> n$

More problems:

Independent Set: given a graph $G = (V, E)$, an ind. set in G is a subset $I \subseteq V$ with no edges between them.

Maximum Indepent Set: compute an ind. set of maximum size

Theorem: Maximum Ind. Set is NP-hard

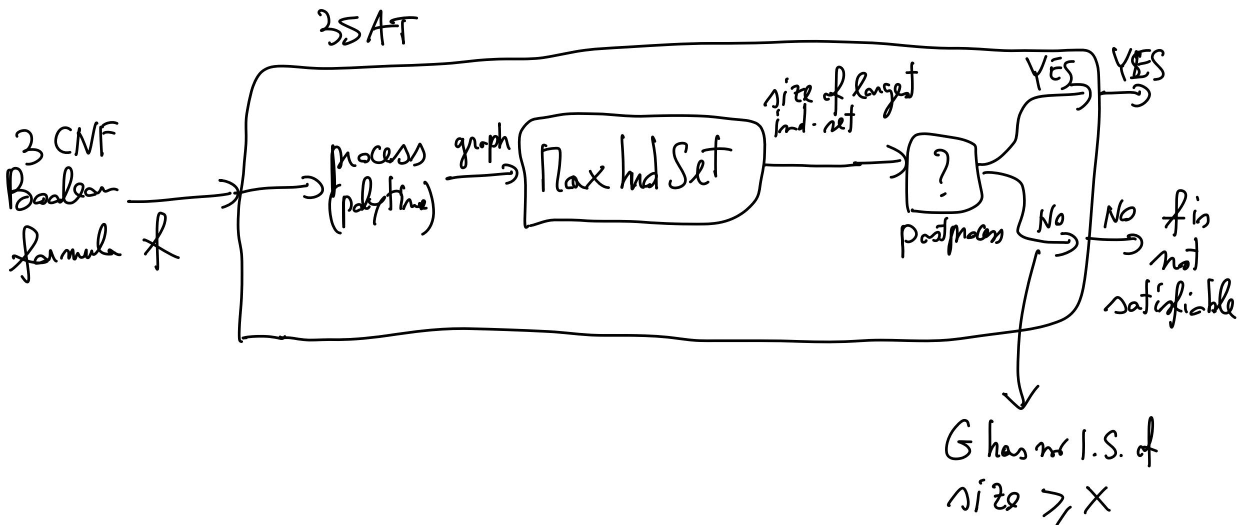
Proof: reduction from 3SAT to Max. Ind Set

they seem totally unrelated problems

logic

graphs

let's see what we have to do:

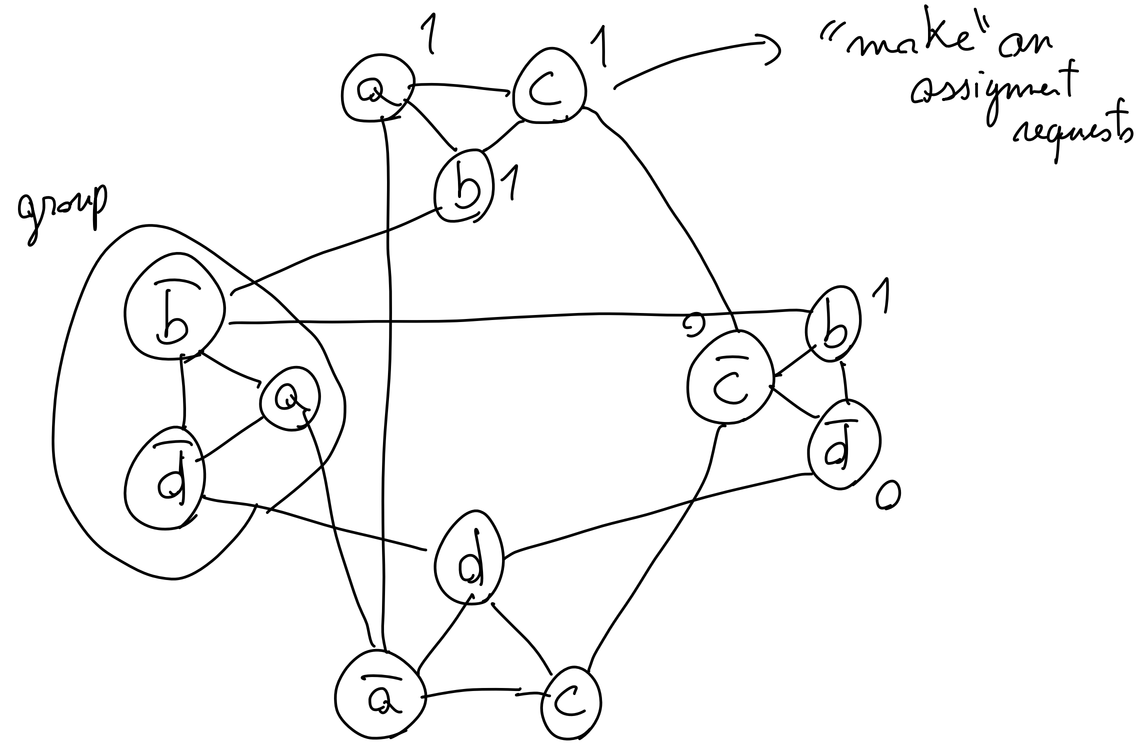


let's see the main ideas:

- pick an arbitrary 3 CNF Boolean formula f with k clauses

$$(a \vee b \vee c) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b} \vee \bar{d})$$

- vertices: each vertex represents one literal in f



- edges:

1) idea: ind. set represents conflicts $\Rightarrow$ add an edge between every pair of vertices making requests that are inconsistent (asking for opposite assignments to the same variable)

   obs.: an ind. set with ≥ 1 vertex in each group gives a satisfying truth assignment $\rightarrow$ should look for ind. sets of size $\geq k$ to say "YES, f is satisfiable"

   Issue: an ind. set now is free to recruit multiple vertices from a group, so I might output "YES, f is satisfiable" even if this is not true!

   $\Rightarrow$ idea: force the recruitment of one vertex per group

2) add one edge between every pair of vertices that are in the same group.

End of the intuition

Claim: G contains an ind. set of size exactly k
$\iff$ the formula f is satisfiable