Harder example Show FORMULA_SAT $\leq_p$ CLIQUE.

actually show 3-SAT $\leq_p$ CLIQUE

3-SAT = { "φ" | φ is satisfiable and in 3-CNF }

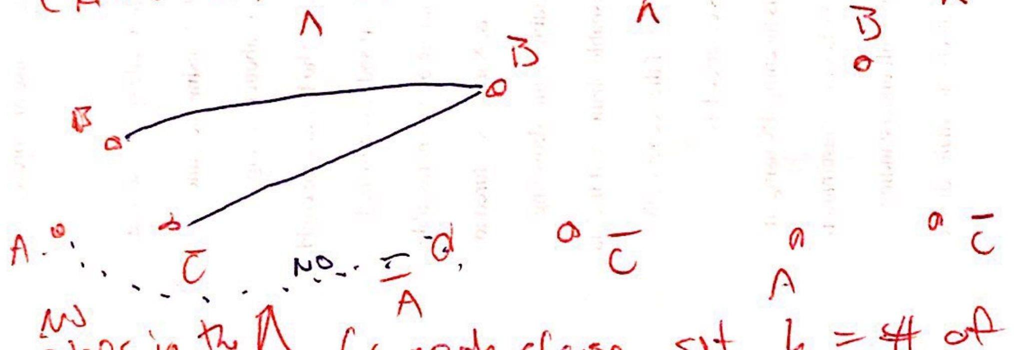
A formula is in 3-CNF (3-conjunctive normal form) iff it is a conjunction (AND) of "3-clauses" (arbitrarily many) $(A \vee \bar{B} \vee C)$

A 3-clause is a disjunction (OR) of exactly 3 literals

A literal is a variable or a negated variable. A $\bar{A}$, B $\bar{B}$

e.g. $(A \vee B \vee \bar{C}) \wedge (\bar{A} \vee B \vee \bar{C}) \wedge (A \vee B \vee \bar{C}) \wedge \dots$

Edges to pick: all edges between consistent literals in different triangles



no edges in the Δ for each clause, set k = # of clauses

Then prove

$\forall \varphi$ a 3CNF formula, $T(\varphi) = G, k$ s.t.

$\varphi \in$ 3-SAT iff $T(\varphi) \in$ CLIQUE

i.e. argue that if φ was satisfiable then $T(\varphi)$ has a k-clique

— but if φ satisfiable, $\exists$ a truth assignment picking a true literal in each clause; this collection of literals is clique in the graph

AND converse: if $T(\varphi)$ has a CLIQUE Then φ is satisfiable

Therefore 3-SAT $\leq_p$ CLIQUE, implying that 3-SAT is "no harder to ~~sta~~ solve" than CLIQUE, i.e. CLIQUE is at least as hard as 3-SAT

Is there a hardest problem in NP?
i.e. a problem that all others reduce to?

Def. A language L is NP-hard if for every language $L' \in NP$, $L' \leq_p L$.

Def. L is NP-complete if $L \in NP$ and L is NP-hard.

not too hard   hard

Any such L essentially captures the hardness of NP, i.e. search for a scalable size certificate that is scalably recognizable

Note: $\leq_p$ is transitive $L_1 \leq_p L_2 \wedge L_2 \leq_p L_3 \rightarrow L_1 \leq_p L_3$
by composing the reductions.

Implies: If L is NP-hard and $L \leq_p L'$ then L' is NP-hard.

$\rightarrow L \longrightarrow L'$ so we just need a first NP-hard problem and then reductions.

A "first" NP-hard problem

<u>Cook's Theorem</u> 3-SAT is NP-hard.

<u>Proof.</u> Given arbitrary $L \in NP$, show there exists a polynomial reduction from L to 3-SAT. We know L can be solved (i.e. $w \in L$ answered) by searching for a certificate C so that $T(w,C)$ is true for a scalable computation $T(w,C)$.

can be implemented as a circuit ??

We will show that finding a certificate that makes $T(w,c)$ true is exactly searching for a certificate to <u>satisfy the circuit for $T(w,C)$</u>. i.e. satisfiability.