Puzzle:

HAMPATH = $\{\langle G\rangle \mid$ G encodes a graph with a "Hamiltonian cycle", a path that visits each vertex exactly once.$\}$

3-COLOR = $\{\langle G\rangle \mid$ G encodes a graph that can be "3-colored": we can give each vertex a color in $\{R, G, B\}$ s.t. no two adjacent vertices have the same color$\}$

Given G, what info do we (a verifier) need to check if $\langle G\rangle \in$ HAMPATH, or $\langle G\rangle \in$ 3-COLOR?

If this information is encoded as a string c, what's the runtime of $V_{HAMPATH}(\langle G, c\rangle)$, $V_{3COL}(\langle G, c\rangle)$?

(Say V has n vertices).

Certificate for HAMPATH: a path.
Certificate for 3-COLOR: a coloring.

Show in NP:

$V_{HAMPATH}(\langle G, c\rangle)$:

Check $O(n)$ edges of the path c are in the graph G, and form a Hamiltonian cycle. (plus time to scan back and forth)

$V_{3COLOR}(\langle G, c\rangle)$: Check $O(n^2)$ pairs of adjacent vertices $(\bullet\!-\!\bullet)$, make sure our coloring c never gives an adjacent pair the same coloring.

Today:

1. NP-completeness: Cook-Levin Theorem.
2. NP-completeness reductions.

transition to review session mode.

— Tim will be in class to 4:45
— —— on Zoom ———— 5-7.

First: Course Evals.

P $\approx$ "efficiently" decidable languages
(TM can decide in poly(n)).

NP $\approx$ "efficiently verifiable" languages
(verifiable by a TM, with the right proof string, in time poly(n))
(NTM decides in poly(n)).

$P \stackrel{?}{=} NP$.

What would it take to prove P=NP?

To show $NP \subseteq P$:
- simulate an NTM with a TM in time poly(n)?

Cook-Levin: If we can ~~solve~~ decide one special problem in NP (SAT), we can solve every problem in NP in time poly(n).

in poly(n)

"To show P=NP, solve SAT in time poly(n)."

SAT = "Boolean formula satisfiability"

Boolean formula: n Boolean variables, connected by $\wedge, \vee, \overline{}$.

$$\phi = (\overset{T}{x_1} \wedge \overset{T}{x_2}) \vee \left((\overset{F}{\overline{x_1}} \wedge x_4) \wedge (x_3 \vee x_2) \right)$$

Def. A Boolean formula ϕ is satisfiable if there is some assignment of T/F values to the variables that makes the statement true.

$(x_1 \wedge \overline{x_1})$ is unsatisfiable.

$$(\underset{F}{x_1} \vee \underset{T}{x_2}) \wedge (\underset{T}{\overline{x_1}} \vee \underset{F}{\overline{x_2}}) \wedge (\underset{T}{\overline{x_1}} \vee \underset{T}{x_2}) \qquad \begin{array}{l} x_2 = T \\ x_1 = F \end{array}$$

SAT = $\{\langle \phi \rangle \mid \phi$ is a satisfiable Boolean formula (on n variables) $\}$

SAT $\in$ NP. (NTM can guess a satisfying assignment.) $n \geq 1$

Cook-Levin: Decide SAT in time poly(n)
$\Longrightarrow$ decide any language in NP in time poly(n).

Proof sketch. (Sipser p. 304-311).
Start with any language $L \in NP$, decided by the poly(n)-time NTM N_L. Will show that if we could decide SAT in time poly(n), we could decide L in time poly(n).

Idea: Write the statement
"N_L accepts w" as a Boolean formula ϕ_w such that
$N_L(w)$ accepts $\Longleftrightarrow$ $\phi_w \in$ SAT (ϕ_w satisfiable).

(If we have this, can use a decider for SAT on $\langle \phi_w \rangle$ to determine if $N_L(w)$ accepts.)

To do's: (1) What is ϕ_w?

(2) $\langle \phi_w \rangle \in SAT \iff w \in L$

(3) $|\langle \phi_w \rangle| = poly(n)$, and ϕ_w can be built in time $poly(n)$.

(We'll ignore (2) and (3), focus on (1).)

Recall: we can summarize an (N)TM with a configuration string, which lists the state, the tape contents, and the tape head position.

Input: $w = w_1 w_2 \cdots w_m$

$c = q_0 \overset{\downarrow}{w_1} w_2 \cdots w_n$

$c' = q_r w_1 \overset{\downarrow}{w_2} \cdots w_n$

"N_L accepts w" = "There exits a sequence $c_1, c_2, \cdots c_\ell$ of NTM configs"

$\wedge$ "$c_1 = q_0 w$, the start configuration $N_L(w)$"

$\wedge$ (2) "c_{i+1} follows from c_i according to N_L's transition function, for all $1 \le i \le \ell$."

$\wedge$ "c_ℓ is an accepting configuration."

= "c_1, character 1, is q_0" $\wedge$ "c_1, char 2, is w_1"

$\wedge$ "c_1, char 3, is w_2" $\wedge$ $\cdots$ $\wedge$ "c_1, char $n+1$, is w_n"

= Boolean variable $x_{1,1,q_0}$

Connect those variables to make (2) true.

"$c_2[i] = a \iff c_1[i] = a$" "config 1, char $i = a$"

$(x_{1,i,a} \wedge x_{2,i,a}) \vee (\overline{x_{1,i,a}} \wedge \overline{x_{2,i,a}})$.

Remainder of proof: carefully building up ϕ_w s.t.

$\langle \phi_w \rangle \in SAT \iff w \in L$. $\square$

Takeaway: Decide SAT in $poly(n)$ $\implies$ Decide any $L \in NP$ in $poly(n)$.

(1) SAT $\in$ NP, (2) If we solve SAT in poly(n), we can solve any problem in NP in poly(n).

= SAT is NP-complete.

Back @ 2:35

"SAT is  of NP"

3SAT = $\{\langle\phi\rangle \mid \phi$ is a satisfiable Boolean formula in "3-CNF," or "3-conjunctive normal form"$\}$

= "a big $\wedge$ of 3-variable $\vee$ clauses."

$(x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_4 \vee x_3) \wedge (x_4 \vee \overline{x_5} \vee x_6) \cdots$

Fact. If you can decide 3SAT in time poly(n)

$\Longrightarrow$ can decide SAT in time poly(n).

3SAT fast $\Longrightarrow$ SAT fast $\Longrightarrow$ any $L \in$ NP fast.

*

SUDOKU fast
HAMPATH
3-COLOR
VX COVER
SUBSETSUM

all $\in$ NP = NP-complete.

all allow you to decide any $L \in$ NP in time poly(n), if you have an efficient alg for them

If we could solve any of many efficiently verifiable problems quickly, we could solve them all.

$$P \stackrel{?}{=} NP.$$

NP-completeness reduction.

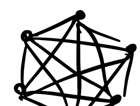
Def. A decision problem (language) L_1 is NP-complete if

*(1) $L_1 \in NP$

(2) Solving/deciding L_1 in time poly(n) implies deciding any $L \in NP$ in time poly(n).

CLIQUE = $\{\langle G, k\rangle \mid$ G has a k-clique: k vertices, all mutually adjacent. $\}$

Theorem. CLIQUE is NP-complete.



We'll show that if we could solve CLIQUE in time poly(n), we could solve 3SAT in time poly(n).

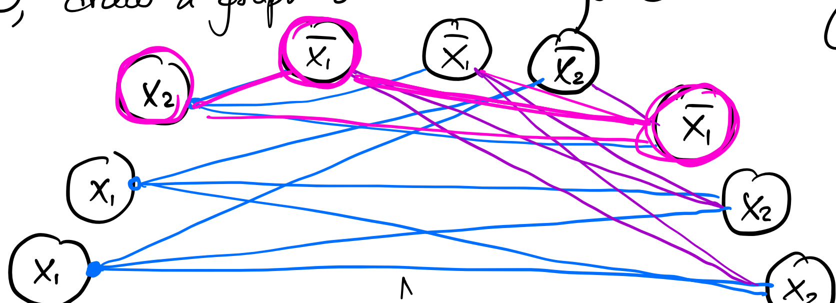
*(1) CLIQUE $\in$ NP. (Given a certificate clique, a verifier can quickly determine if the graph contains it.)

(2) Start with a 3-SAT formula Φ with k clauses.

Example: $(x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$.

Given Φ, draw a graph with a vertex for each variable appearance (literal).

G



Connect every pair of vertices (u,v) unless (1) u,v are in the same clause
(2) $u = \overline{v}$.

Claim: Graph G has a k-clique $\Longleftrightarrow$ k-clause formula Φ satisfiable

$\Rightarrow$. G has a k-clique.

- It must have 1 vertex from each clause (clauses unconnected within themselves, k clauses).
- It doesn't include vertices x, $\overline{x}$, as these are never connected.

So: assigning variables according to my k-clique satisfies every clause.

$\Leftarrow$. ϕ satisfiable. In this case, there is an assignment that satisfies every clause. Choose k vertices in G corresponding to one satisfied variable in each clause.

These vertices form a clique: no $x, \bar{x}$ pairs,
no vertices in the same clause.

Given any 3SAT formula ϕ, we can convert it to a graph G such that $\langle G, k\rangle \in$ CLIQUE if and only if the k-clause formula $\langle\phi\rangle \in$ 3SAT.

Thus, solving CLIQUE gives us an algorithm for 3SAT. $\square$

Back at 3:18