chang@umbc.edu

NP Completeness:

a machine model

$3COLOR \le_m^p SAT$

$SAT \le_m^p 3COLOR$

Recap: 3COLOR is equivalent to SAT.

sets.

Defn: Let A & B be decision problems.
We say A reduces to B (written $A \le_m^p B$)
if there exists a polynomial-time computable
function f s.t.

$$x \in A \iff f(x) \in B$$

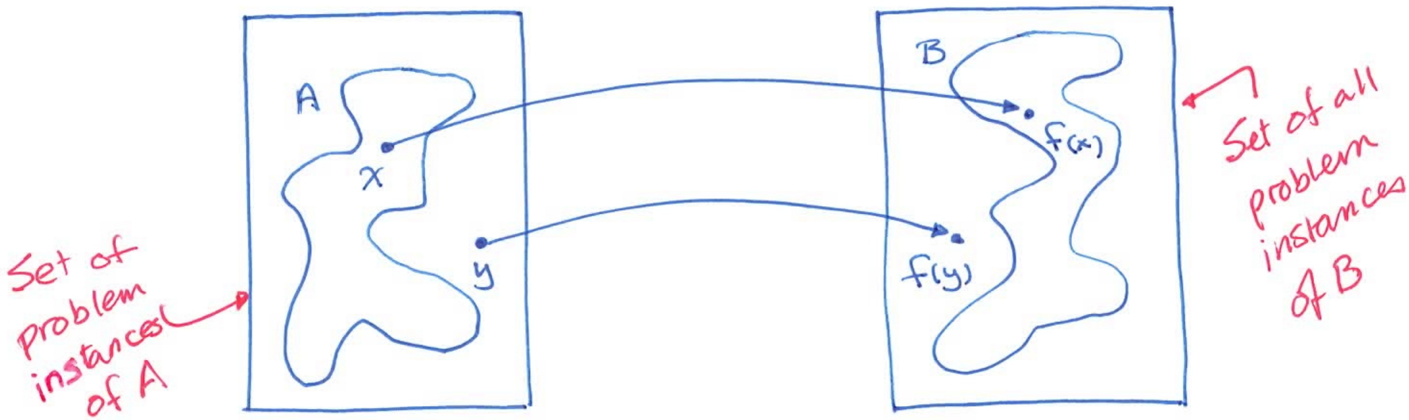
"polynomial-time computable" = algorithm for f runs in $O(n^k)$ time for some $k \in \mathbb{N}$.

constant

$A \le_m B$ + fast algorithm for B

$\Rightarrow$ "fast" algorithm for A

Suppose $A \leq_m^p B$.



$x \in A \Rightarrow f(x) \in B$

$y \notin A \Rightarrow f(y) \notin B$

will define later

Problems equivalent to SAT & 3COLOR are NP-complete.

equivalence is transitive

$A \leq_m^p B$ & $B \leq_m^p C \Rightarrow A \leq_m^p C$

Thousands of problems are NP-complete.

Capture many important optimization problems, eg:

- Clique
- Vertex Cover
- Traveling Salesman Problem
- Partition
- 3D Matching.

will give you the flavor of a range of NP-complete problems.

Clique:

Input: undirected graph $G = (V, E)$

number k

Question: Does G have a k-clique? as a subgraph

k-clique = k vertices in V s.t. any two vertices are connected by an edge.

3-Clique 4-Clique 5-Clique

Vertex Cover

Input: an undirected graph $G = (V, E)$
a number k

Question: does there exist a subset $V' \subseteq V$ s.t.
$|V'| \leq k$, and
for all edges $(u,v) \in E$, either $u \in V'$ or $v \in V'$?

similar to Hamiltonian Cycle

Traveling Salesman Problem

Input: an undirected graph $G=(V,E)$

a weight function $\omega: E \to \mathbb{R}^+$

a bound B

Question: Does there exist a simple cycle in G that visits every vertex exactly once s.t. the sum of the edge weights of the edges in the cycle is $\leq B$.

Partition

Input: n number $a_1, \ldots, a_n \in \mathbb{Z}^+$

Question: Does there exist a subset S of the numbers s.t

$$\sum_{i \in S} a_i = \sum_{i \notin S} a_i$$

I.e., pick a subset of the numbers s.t. the sum of the numbers is exactly half of the total sum.

3-Dimensional Matching

Input: disjoint sets W, X, Y s.t. $n = |W| = |X| = |Y|$

$$M \subseteq W \times X \times Y$$

$\hookrightarrow M = \{(w, x, y) \mid w, x \,\&\, y \text{ are "compatible"}\}$

Question: does there exist $M' \subseteq M$ s.t. $|M'| = n$ and no two elements of M' agree in any coordinate.

$$W = \{w \mid \exists x \in X \,\&\, \exists y \in Y \quad (w, x, y) \in M'\}$$
$$X = \{x \mid \exists w \in W \,\&\, \exists y \in Y \quad (w, x, y) \in M'\}$$
$$Y = \{y \mid \exists w \in W \ \exists x \in X \quad (w, x, y) \in M'\}$$

each $w, x \,\&\, y$ appears exactly once

P = decision problems that can be solved by some algorithm that runs in time $O(n^k)$ for some constant k.

NP = decision problems that can be verified in P.

= guessing.

= "nondeterministic" polynomial time.

Check this is true for Clique, VC, 3DM, partition, etc.

P = NP means there is a free lunch.

Working definition of NP

A decision problem $A \in NP$ if

① $\exists B \in P$

② $\exists k \in \mathbb{N}$

$$x \in A \iff \exists y,\ |y| \le |x|^k,\ \text{s.t. } (x,y) \in B.$$

otherwise $NP = coNP$

Note: some properties are difficult to verify.

$\{(G,k) \mid$ the largest clique in G has $\le k$ vertices $\}$

$=$ G does not have cliques $> k$

Defn: A decision problem X is NP-complete, if

1. $X \in NP$
2. for all $Y \in NP$, $Y \leq_m^p X$

Cook's Theorem [1971] : SAT is NP-complete.

How to show that a new problem Q is NP-complete.

1. Show $Q \in NP$
2. Show $SAT \leq_m^p Q$

or some other known NP-complete problem.

Example: Vertex Cover (VC)

$VC = \{ (G,k) \mid \exists V' \subseteq V,\ |V'| \leq k$ and for all $(u,v) \in E$ either $u \in V'$ or $v \in V'$. $\}$

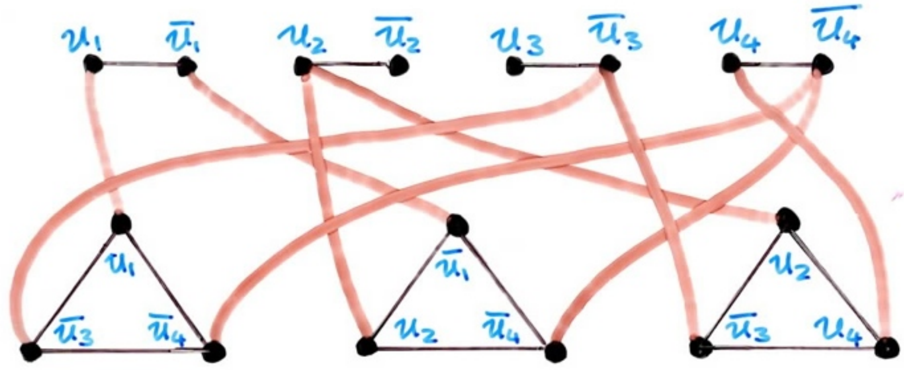
Show $VC \in NP$. Guess $V' \subseteq V$, check each edge in E.

much easier than showing $VC \leq_m^P E$.

$3SAT \le_m^P VC$

$\phi = (u_1 \vee \overline{u}_3 \vee \overline{u}_4) \wedge (\overline{u}_1 \vee u_2 \vee \overline{u}_4) \wedge (u_2 \vee \overline{u}_3 \vee u_4)$

G:



ϕ has n variables & m clauses

G has $2n+3m$ nodes & $n+6m$ edges

$k = n + 2m$

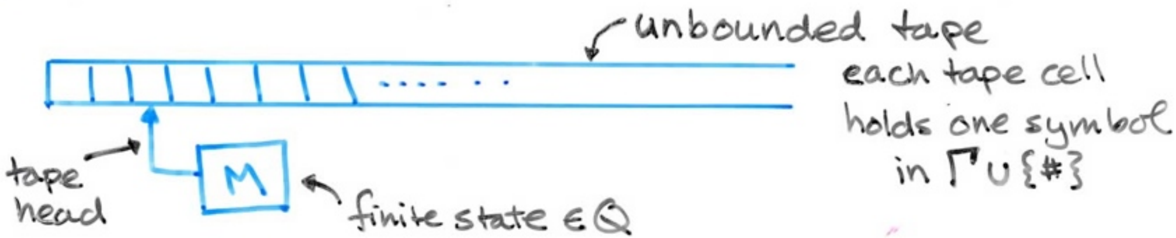
<u>Claim:</u>

$\phi \in 3SAT \iff G$ has a vertex cover w/ k vertices

Cook's Theorem in 20 minutes

Hand Wavy Part

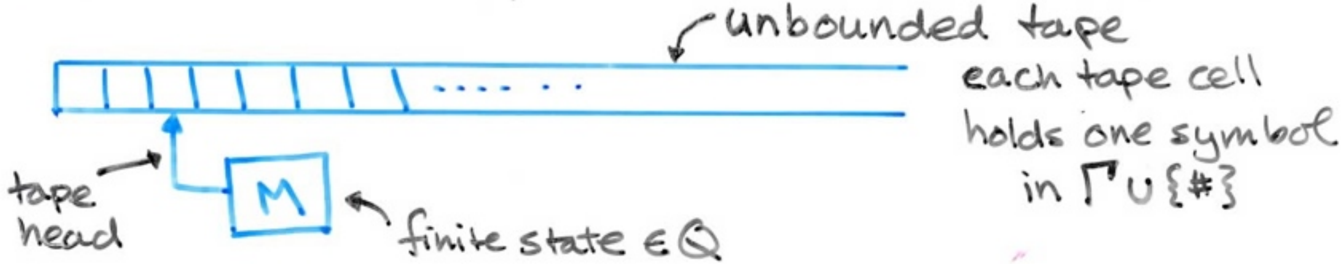
Turing machines



- In each step, a TM M can read one symbol of the tape under the tape head, enter a new state, replace the symbol underneath the tape head and move the tape head left or right.

skip

Turing machines



- transition function

$$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$$

- An input string x is accepted by a TM M if M starting in the start state & the tape head on the leftmost tape cell & x on the tape, enters a unique accepting state q_{acc} after a finite number of transitions.

- $x \in L(M)$ if x is accepted by TM M.

Church-Turing Thesis:

If A is a "computable" set, then $A = L(M)$ for some Turing machine M

Robustness of TM's

extra heads, tapes, ... do not add computational power to TM's.

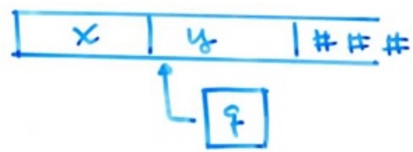
TM's & running time:

If $A \in P$ via the RAM model then $A = L(M)$ for some TM that makes a polynomial number of transitions.

Representing TM configurations

"instantaneous description" = ID

$x q y$ means



Tape holds xy. Tape head reading first symbol of y.
Machine M in state q.

NOT SO HAND WAVY PART

Tableau: visual aid for thinking about
a sequence of ID's

Working defn of NP:

$A \in NP$ if $\exists$ $B \in P$ and a polynomial $q(\ )$ s.t.

$x \in A \iff \exists\, y \in \Sigma^*,\ |y| \leq q(|x|)\ \&\ (x, y) \in B.$

Thus, $x \in A$ iff

$\exists$ a legal tableau

Starting with ID $q_0 x \# y$

s.t. M enters the accepting state q_{acc}

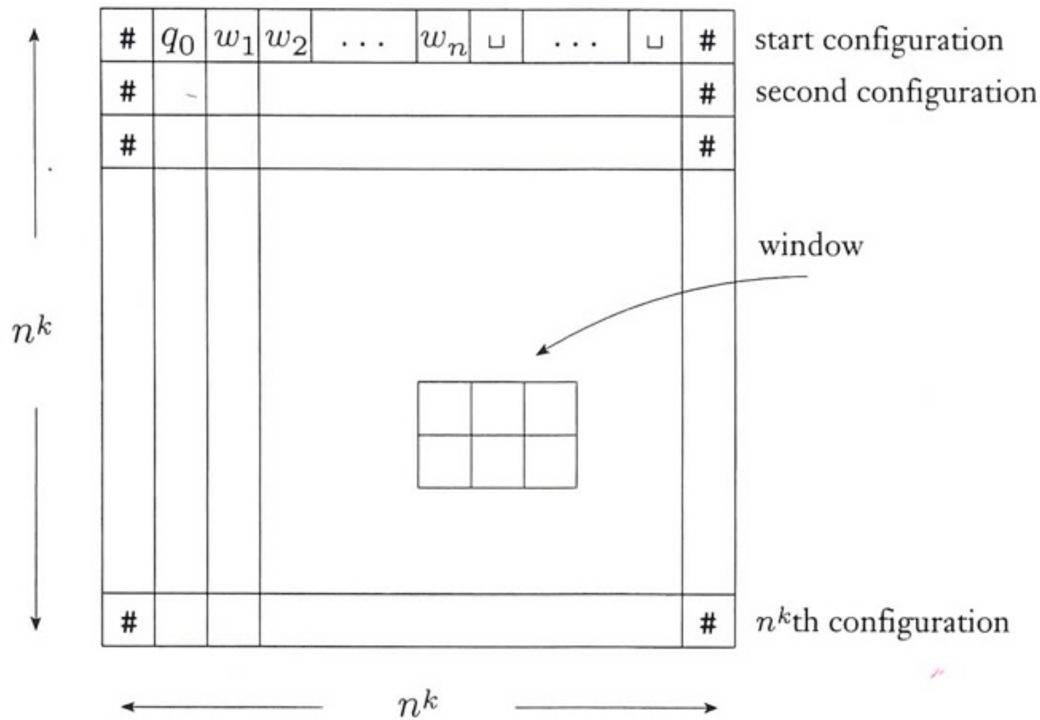


FIGURE 7.8
A tableau is an $n^k \times n^k$ table of configurations

[Sipser]

NOT HAND WAVY PART

Use Boolean formulas to check a tableau is legal

Each cell can hold a state, a tape symbol, or a #. Let $C = Q \cup \Gamma \cup \{\#\}$.

Each cell indexed by i, j, $1 \le i \le n^k$ & $1 \le j \le n^k$.

For each cell i,j and each symbol $s \in C$

$X_{i,j,s}$ is true "means" cell i,j holds symbol s.

Enforce that each cell has a symbol

$$\phi_{cell1} = \bigwedge_{i,j} \bigvee_{s \in C} x_{i,j,s}$$

Enforce that each cell has no more than one symbol

$$\phi_{cell2} = \bigwedge_{i,j} \bigwedge_{\substack{s,t \in C \\ s \neq t}} \left(\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}} \right)$$

Enforce that initial configuration is $q_0 x \# y$

$$\phi_{start} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,\_} \wedge x_{1,4,\_}$$

$$\wedge\ x_{1,n+2,\#} \wedge x_{1,m+2,\#} \wedge \cdots \wedge x_{1,n^k,\#}$$

where $n = |x|$, $m = |x\#y|$

This ensures the first line of the tableau is

$$\# \, q_0 \, x \, \# \, y \, \# \# \# \cdots \#$$

for <u>some</u> y.

Enforce that M entered the accepting state

$$\phi_{acc} = \bigvee_{i,j} X_{i,j,q_{acc}}$$

Enforce that line i+1 of the tableau follows from line i.

Observation: Only need to check that all 2×3 "windows" are legal.

LEGAL = set of all legal 2×3 windows $\subseteq C \times C \times C \times C \times C \times C$

Note: |LEGAL| is finite & constant.

$$\phi_{move} = \bigwedge_{i,j} \text{2×3 window at index } i,j \text{ is LEGAL}$$

$$= \bigwedge_{i,j} \bigwedge_{\substack{(a_1 \cdots a_6) \\ \notin \text{LEGAL}}} \left(\overline{X_{i-1,j,a_1}} \vee \overline{X_{i,j,a_2}} \vee \overline{X_{i+1,j,a_3}} \vee \overline{X_{i-1,j+1,a_4}} \vee \overline{X_{i,j+1,a_5}} \vee \overline{X_{i+1,j+1,a_6}} \right)$$

i.e., a legal window a i,j has one entry different from every illegal window.

(a)

a	q_1	b
q_2	a	c

(b)

a	q_1	b
a	a	q_2

(c)

a	a	q_1
a	a	b

(d)

#	b	a
#	b	a

(e)

a	b	a
a	b	q_2

(f)

b	b	b
c	b	b

FIGURE 7.9
Examples of legal windows

(a)

a	b	a
a	a	a

(b)

a	q_1	b
q_1	a	a

(c)

b	q_1	b
q_2	b	q_2

FIGURE 7.10
Examples of illegal windows

[Sipser]

Claim: $x \in A$

$\Longleftrightarrow$ $\exists$ a legal tableau starting with $q_0 x \# y$

$\Longleftrightarrow$ $\phi_{cell1} \wedge \phi_{cell2} \wedge \phi_{start} \wedge \phi_{acc} \wedge \phi_{move} = \phi$

is satisfiable.

Note: ϕ is in conjunctive normal form

What does it take to settle the P v.s. NP question?

$P = NP$: Give polynomial-time algorithm for SAT, VC, ...

- Favorite bogus proof: use linear programming to solve integer programming and magically round to integer solution.

$P \neq NP$: Argue that every polynomial-time algorithm fails to solve SAT (or any problem in NP).

- There are infinitely many algorithms in P
- Not enough to say the 3 I thought of doesn't work
- What if SAT can be solved in time $n^{1{,}000{,}000{,}000}$?

What if P=NP?

- Maybe there are fast algorithms for important optimization problems.
- n^{1000} not necessarily "fast", but still faster than currently known algorithms.
- Encryption not possible.

What if $P \neq NP$?

- No fast algorithms exist for important optimization problems, not even n^{1000} time algorithms.

- The world is a saner place.

or some other NP complete problem

Maybe the decision problem for Clique is easy, but finding the maximum clique is still hard.

not NP complete unless...

- Called decision v.s. search.

  Note: primality testing is easy, but factoring is still considered hard.

- NP-complete problems are self-reducible and can be used to find "actual" solution.

Example: Finding the largest Clique, given the decision problem for Clique as an oracle.

the proverbial black box

$(G,k)? \Rightarrow$  $\Rightarrow$ yes or no

① Use oracle to find size of largest clique by binary search.

② Throw out a vertex x of G and ask oracle if G still has a k-clique

repeat until found

Yes: throw out x permanently

No: put x back in G and mark it.
remove vertices not connected to x.

Coping with NP-completeness

1. use an approximation algorithm. vertex cover, TSP w/ Δ-inequality.

2. consider special cases — 2D Matching, planar graphs

3. keep your fingers crossed — heuristics.

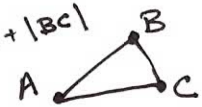
4. work on a different problem. eg., MST v.s. Steiner tree

Approximation Algorithms

Maybe we don't need the largest clique

- Vertex Cover: greedy algorithm can find a vertex cover V' s.t. $|V'|/|V^*| \le 2$. not necessarily a good bound.
  opt. solution

$|AC| \le |AB| + |BC|$



- TSP: if weight function satisfies the triangle inequality, a modified MST + matching algorithm can find a tour T in polynomial time s.t. $|T|/|T^*| \le 1.5$.

- Problems like Clique are NP-hard to approximate.

- Hard to find bounds when you can't compute opt. solution.

Special Cases:

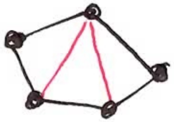
- Some cases are easy.

  2SAT $\in P$, 2-COLOR $\in P$, 2DM $\in P$

- Polynomial-time algorithms for Clique for:

  planar graphs

  chordal graphs → every cycle with 4 or more vertices has a chord



- Subgraph Isomorphism is NP-complete

  polynomial-time if we ask whether a tree T is isomorphic to a subgraph of a forest G.

Keeping your fingers crossed

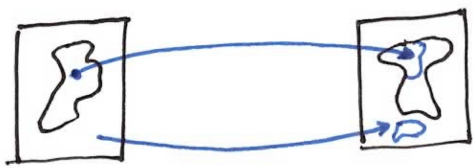
Use "heuristics" that seem to work on most cases.

- Often do not have running time or performance guarantees
- May be difficult to describe when the algorithm works.

It works, except when it doesn't

- OTOH, every NP-complete problem must have "easy" parts as well, because every problem in NP including the easy ones reduce to them.

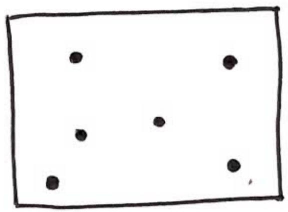
"easy" $A \in P$

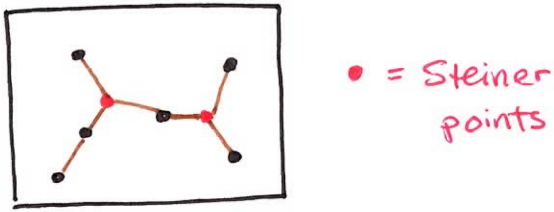


SAT

Work on a different problem.

Example: Steiner Tree is NP-complete.





connect locations
after adding Steiner pts

Maybe MST is
good enough