P and NP.

recall defn

Definition

P = {decision problems solvable in poly. time}

We will study what is / is not in this class.

Careful of
- machine model – log cost RAM
- input size – # bits

Recall from last day:

$A \leq_p B$ for problems A, B "A reduces to B"

means: we can use a poly. time algorithm for B to give a poly. time algorithm for A.

Example.

Hamiltonian cycle / path = cycle / path that visits every vertex exactly once

this graph has a Hamiltonian path but not a Hamiltonian cycle.

Lemma Hamiltonian path $\leq_p$ Hamiltonian cycle.

Pf. Suppose we have a poly. time algorithm for Hamiltonian cycle.

We want to design a poly. time algorithm for Hamiltonian path.

Input: graph G.
Algorithm:
- construct graph G' by adding one new vertex adjacent to all vertices of G

- send G' to the algorithm to test for Ham. cycle
- return the YES/NO answer

This alg. runs in poly. time.
Correctness: must prove

Claim G has a Ham. path iff G' has a Ham. cycle
Pf $\Rightarrow$. Suppose G has a Ham. path x to y. Adding v and edges (x,v), (v,y) gives a Ham. cycle in G'.
$\Leftarrow$ Suppose G' has a Ham. cycle. Removing v gives a Ham. path in G.

Lemma Ham. cycle $\leq_P$ Ham. path

EX. prove this.

FACT: no one knows a poly. time alg. for either problem.

There is a large class of decision problems not known to be in P but all <u>equivalent</u> in the sense that $A \le_P B$ for all A, B in the class. (recall defn of $\le_P$)
i.e. poly. time alg. for one yields poly. time alg. for all.

A few problems in the class:

Hamiltonian path / cycle.

TSP – given edge weighted graph, number k, is there a TSP tour of weight $\le k$?

IND. SET – given graph, number k, is there an ind. set of size $\ge k$?

Common feature: if the answer is YES there is some $\underbrace{\text{succinct info.}}_{\text{"certificate"}}$ to <u>verify</u> it.

(in particular, the TSP tour, the ind. set)
Contrast this with NO answer.

A verification alg. takes input + certificate and checks it.

Definition Alg. A is a verification alg. for problem the decision problem X if
- A takes two inputs x, y and outputs YES or NO
- for every input x for problem X, x is a YES for X iff there exists a y "certificate" s.t. $A(x,y)$ outputs YES.

Furthermore, A is a polynomial time verification alg. if
- A runs in poly. time
- there is a polynomial bound on the size of the certificate, i.e.

  $\forall x$, x is a YES input for X iff

  $\exists y$ with $\text{size}(y) \le (\text{size}(x))^k$, k const.

  s.t. $A(x,y)$ outputs YES

NP = { decision problems that can be verified in polynomial time }
i.e. have poly. time verification algorithms

Example Subset Sum $\in$ NP

Given numbers $w_1 \cdots w_n$ and W
is there a subset $S \subseteq \{1 \cdots n\}$ s.t. $\sum_{i \in S} w_i = W$

certificate is S

verification alg.: check that $\sum_{i \in S} w_i = W$
poly. time

? Is there a poly. time verification alg. for NO answers?
What could you give to verify that no subset has sum W? OPEN.

<u>Example</u> TSP (decision version) $\in$ NP

Given graph G, weights on edges, number k, does G have a TSP tour of length $\leq k$?

certificate: the tour, i.e. permutation of vertices

poly. time verification alg:

- check it's a permutation
- check that edges exist
- check that $\sum$ edge weights in tour $\leq k$

---

coNP = {decision problems where the NO instances can be verified in poly. time}

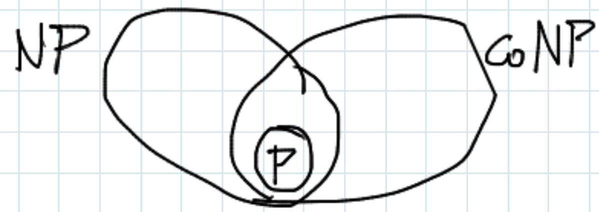
e.g. Primes: given number n, is it prime?

Primes $\in$ coNP

easy: to verify n is <u>not</u> prime, show natural numbers $a, b \geq 2$ s.t. $a \cdot b = n$

OPEN QUESTIONS

1. $P =? NP$
2. $NP =? coNP$
3. $P =? NP \cap coNP$



Properties

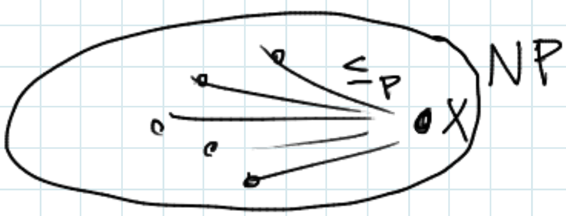
- $P \subseteq NP$, $P \subseteq coNP$
- any problem in NP can be solved in time $O(2^{n^k})$ by trying all certificates one by one

Definition A decision problem X is NP-complete if
- $X \in NP$
- for every $Y \in NP$, $Y \le_p X$

i.e. X is [one of] the hardest problems in NP.

Two important implications of X being NP-complete
- if $X \in P$ then $P = NP$
- if X cannot be solved in poly. time then no NP-complete problem can be solved in poly. time
- if $X \in coNP$ then $NP = coNP$ (this needs proof)

The first NP-completeness proof is difficult

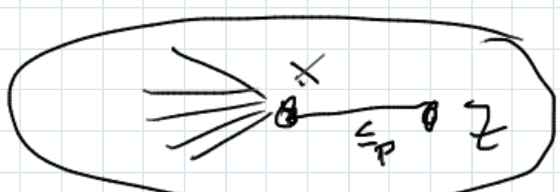


Must show that <u>every</u> problem $Y \in NP$ reduces to X.

Subsequent NP-completeness pfs are easier because $\le_p$ is transitive

$Y \le_p X$ and $X \le_p Z$ implies $Y \le_p Z$

So to prove Z is NP-complete we just need to prove $X \le_p Z$ where X is a known NP-complete problem.



Summary To prove Z is NP-complete
1. prove $Z \in NP$
2. prove $X \leq_p Z$ for known NP-complete problem X

Our first NP-complete problem: Circuit Satisfiability
[proof later – also definition]
2nd NP complete problem: Satisfiability
[proof later, but will define the problem now]

Satisfiability
Input: a Boolean formula made of Boolean variables, $\wedge$ "and", $\vee$ "or", $\neg$ "not",
e.g. $\neg(x_1 \wedge x_2) \vee (x_3 \wedge (x_5 \vee \neg x_4))$
Question: Is there an assignment of True/False to the variables to make the formula True.

Ex. Satisfiability $\in$ NP.
Sat is NP-complete, even the special form from Assign 4, "CNF" – Conjunctive Normal Form

e.g. $\underbrace{\underbrace{(x_1 \vee \neg x_2 \vee x_3)}_{\substack{\text{clause} \\ \text{is } \vee \text{ of literals}}} \wedge (\underbrace{\neg x_1}_{\text{literal}} \vee \underbrace{x_4}) \wedge (x_3 \vee x_4 \vee \neg x_5)}_{\text{formula is } \wedge \text{ of clauses}}$

In fact it's still NP-complete when all clauses have 3 literals — 3-SAT
but 2-SAT is in P

3-SAT
Input: A Boolean formula that is an $\wedge$ of clauses, each clause an $\vee$ of 3 literals, each literal a variable or negation of variable.
Question: Is there an assignment of True/False to variables that makes the formula True.

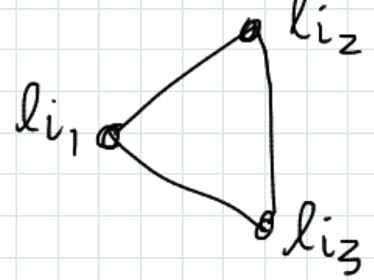
<u>Thm</u> 3-SAT is NP-complete [pf. later].

Ind. Set
Input: Graph $G=(V,E)$, number k
Q: Does G have an $\underbrace{\text{independent set}}$ of size $\geq k$
i.e. a set $S \subseteq V$ s.t.
there is no edge (u,v) with $u,v \in S$

<u>Thm</u> Ind. Set is NP-complete
<u>Pf</u>. 1. Ind. Set $\in$ NP — we saw this already
2. 3-SAT $\leq_P$ Ind. Set.
Suppose we have a (black box) poly.-time alg. for Ind. Set.
Give a poly. time alg. for 3-SAT
Input: 3-SAT formula F
clauses $C_1 \cdots C_m$, variables $x_1 \cdots x_n$

$C_i = (l_{i_1} \vee l_{i_2} \vee l_{i_3})$

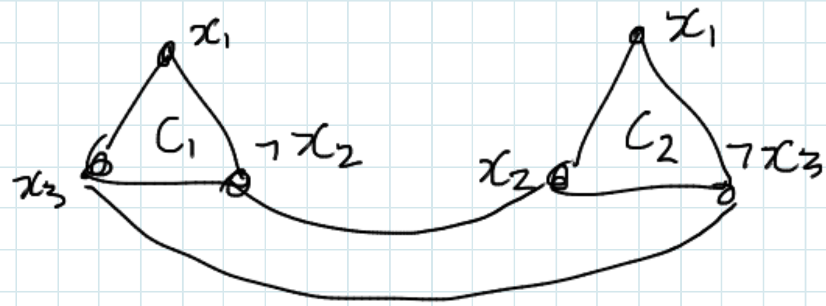
Create a graph G on vertices l_{ij} $i = 1 \cdots m$ $j = 1, 2, 3$



join literals in a clause
(ind. set will "pick" one)

join literals that are negation of each other

e.g. $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3)$



G has poly. size — $3m$ vertices

<u>Claim</u> G has an Ind. Set of size $\geq m$ iff F is satisfiable

Thus our 3-SAT alg. is

- construct G
- run Ind. Set alg. on G, m
- output the YES/NO answer.

This alg. runs in poly.-time.

<u>Proof of Claim</u>

$\Leftarrow$ Suppose F is satisfiable

Pick one vertex from each $\triangle$ corresponding to a True literal. Gives ind. set of size m

$\Rightarrow$ Suppose G has ind. set S of size m.
S can only have one vertex from each Δ.
S cannot use x and $\neg x$
Thus we can set all literals in S True
and this satisfies the formula. (If a variable isn't set by S (i.e. neither y nor $\neg y$ in S), then can set it arbitrarily.

Ex. Carry out this construction on an example

X