Lecture 20

Announcements:

HW4 returned later today

Note: regrade requests could raise or lower your score

HW5 → out today (Due Date Now Nov 30 11am)

The Class NP

Defn 1 NP = { L | L is a language decided by a nondeterministic TM running in polynomial time }

Runtime of a nondeterministic TM M on input w:
= Max. runtime over all nondeterministic computation paths on input w.

f(n) runtime : $\forall w \in \{0,1\}^n$: runtime of TM on on w is $\leq f(n)$

polynomial time = poly(n) ($O(n^k)$ for some $k \geq 0$)

The Class NP

Defn 1 $NP = \{ L \mid L$ is a language decided by a nondeterministic TM running in polynomial time $\}$

Equivalent Defn of NP

A verifier for Language $L \subseteq \{0,1\}^*$ is an algorithm (a deterministic algorithm) $V(w,c)$ where $L = \{w \mid V(w,c) \text{ accepts}\}$ where c is an additional string that we call a certificate or proof

A verifier is polynomial-time if it runs in time polynomial in $|w|$.

* Note that if A is a polytime verifier then $|c|$ must also be polynomial in $|w|$.

Defn 2 $NP = \{ L \mid L$ has a polytime verifier$\}$

A verifier $\underbrace{V(w, c)}$ accepts a language L

det. TM runs
in time poly(|w|)

if : $\forall w \in L$: $\exists c$ st. $V(w,c)$ accepts

$\forall w \notin L$: $\forall c$ $V(w,c)$ halts + rejects

Equivalence Between Defns 1 and 2

$\mathcal{L}_1 = \{ L \mid L$ is accepted by a nondeterministic polytime algorithm$\}$

$\mathcal{L}_2 = \{ L \mid L$ has a polytime verifier$\}$

① $L \in \mathcal{L}_2 \Rightarrow L \in \mathcal{L}_1$:

Let Algorithm V be verifier for L running in time n^k

Nondet TM N: on input w, $|w|=n$

Nondeterministically select $c \in \{0,1\}^{n^k}$

Run V on (w,c)

If V accepts (w,c), accept, otherwise reject

$A(w,c)$

Equivalence Between Defns 1 and 2

$\mathcal{L}_1 = \{ L \mid L$ is accepted by a nondeterministic polytime algorithm$\}$

$\mathcal{L}_2 = \{ L \mid L$ has a polytime verifier$\}$

② $L \in \mathcal{L}_1 \Rightarrow L \in \mathcal{L}_2$:

Let N be a nondeterministic TM accepting L and running in time n^k

Verifier A on $\langle w, c \rangle$:

Simulate N on w, where c is a description of the nondeterministic choices to make at each step

If this computation path (described by c) accepts then accept $\langle w, c \rangle$; otherwise reject

← c is choices for nondet. moves

Example

SAT: Input is a CNF formula $f = C_1 \wedge C_2 \wedge \dots \wedge C_m$

over Boolean variables $x_1, \dots, x_n$

Each C_i is a disjunction of ≤ 3 literals

$f \in$ 3SAT iff f is satisfiable.

$x_1=0\ x_2=0\ x_3=1$:

$(x_1 \vee x_2 \vee \bar{x}_3)$

$0\ \ 0\ \ 0$

Example $f = \underbrace{(x_1 \vee x_2 \vee \bar{x}_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee x_4)}_{C_2} \wedge \underbrace{(\bar{x}_1 \vee x_4 \vee \bar{x}_5)}_{C_3} \wedge \underbrace{(x_2 \vee x_3 \vee \bar{x}_1 \vee x_5)}_{C_4}$

f is satisfiable iff there exists an assignment

$\alpha \in \{0,1\}^n$ to $x_1, \dots, x_n$ such that $f(\alpha) = 1$

Example $f = \underbrace{(x_1 \vee x_2 \vee \bar{x}_3)}_{c_1} \wedge \underbrace{(\bar{x}_2 \vee x_4)}_{c_2} \wedge \underbrace{(\bar{x}_1 \vee x_4 \vee \bar{x}_3)}_{c_3} \wedge \underbrace{(x_2 \vee x_3 \vee x_4 \vee \bar{x}_1)}_{c_4}$

α : 0 0 1 1
$x_1\ x_2\ x_3\ x_4$

$f(\alpha) = (0 \vee 0 \vee \bar{1}) \wedge (\bar{0} \vee 1) \wedge (\bar{0} \vee 1 \vee \bar{1}) \wedge (0 \vee 1)$

$= (0) \wedge (1) \wedge (1) \wedge (1)$

$= 0$

$\alpha = 1101 \qquad f(\alpha) = 1$

$\neg x_i \equiv \bar{x}_i$

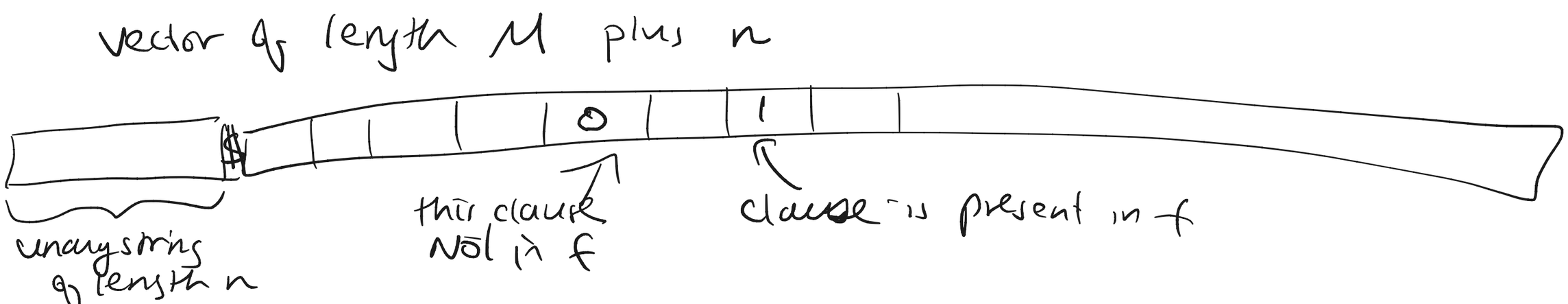
3SAT Same as SAT but now all clauses have size (= # literals in clause) ≤ 3

Encoding SAT / 3SAT :

3SAT encoding :

The total # of distinct clauses of size ≤ 3 is poly(n).

$$M = \binom{n}{0} + \binom{n}{1}2 + \underbrace{\binom{n}{2}\cdot 4}_{\text{clauses of size 2}} + \binom{n}{3}\cdot 2^3 = \text{poly}(n)$$



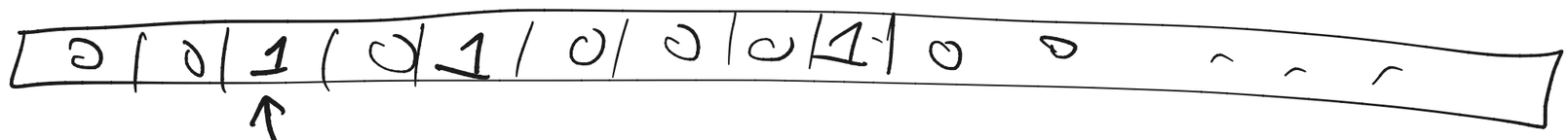
3SAT encoding. say $n = 3$ $\quad x_1\ x_2\ x_3$

all possible 3-clause (clause of size ≤ 3)

ϕ, (x_1), $(\bar{x}_1)$, (x_2), $(\bar{x}_2)$, (x_3), $(\bar{x}_3)$, $(x_1 \vee x_2)$, $(\bar{x}_1 \vee x_2)$, $(x_1 \vee \bar{x}_2)$, $(\bar{x}_1 \vee \bar{x}_2)$

$(x_2 \vee x_3)$ $\quad$ $(x_1 \vee x_3)$ $\quad$ $x_1 \vee x_2 \vee x_3$

$1 + 6 + 12 + 8 = 27$

M 

$(\bar{x}_1)$ is a clause in f

$(\bar{x}_2)$ " " " " "

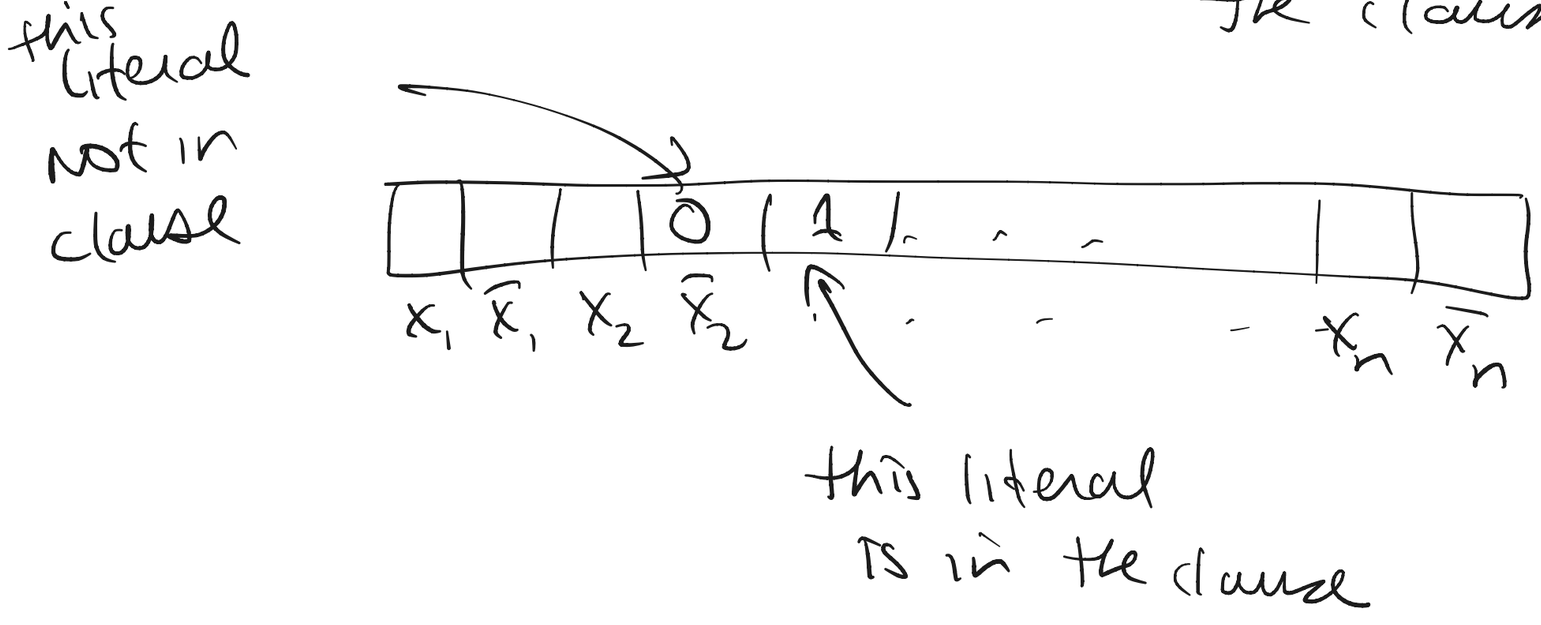
$(\bar{x}_1 \vee x_2)$

Encoding SAT: $f = C_1 \wedge \dots \wedge C_m$ m clauses

No restriction on size of the clauses.

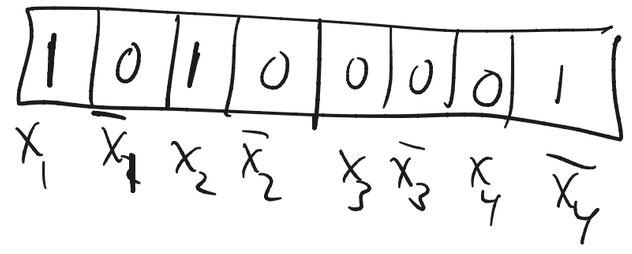
List of clauses $f = (x_1 \vee x_2 \vee \bar{x}_4) \wedge (x_3 \vee x_7) \wedge \dots$

Clause 1 : $\leq 2n$ bits to describe the clause



Ex $x_1 x_2 x_3 x_4$

$C_1 = (x_1 \vee x_2 \vee \bar{x}_4)$



m clauses $\longrightarrow$ $O(m \cdot n)$ length of encoding

$$3SAT = \{ \langle f \rangle \mid \langle f \rangle \text{ encodes a } \underline{\text{satisfiable}} \text{ 3CNF } f \}$$

$$SAT = \{ \langle f \rangle \mid \langle f \rangle \text{ encodes a satisfiable CNF, } f \}$$

Brute force alg:

try all 2^n possible Boolean assignments to underlying variables $x_1, \ldots, x_n$

$V(\langle f \rangle, c)$ ← c is a Boolean ass. $\alpha \in \{0,1\}^n$

SAT & 3SAT in NP

Defn 1 NTM N for 3SAT:

On input $f = C_1 \wedge \cdots \wedge C_m$:

guess an assignment α to the underlying variables } Nondeterministic choices

Evaluate $f(\alpha)$

If $f(\alpha) = 1$ halt & accept

else halt & reject

N runs in time: $\text{poly}(\text{size}(f)) + \text{poly}(n, m)$

$f = C_1 \wedge \cdots \wedge C_m$

size(f): $m(3 \log n) \leq O(n^3 \cdot 3 \log n)$

$= \text{poly}(n)$

3SAT in NP

Defn 1 NTM N for 3SAT:

On input $f = c_1 \wedge \cdots \wedge c_m$:

guess an assignment α to the underlying variables } Nondeterministic choices

Evaluate $f(\alpha)$

If $f(\alpha) = 1$ halt & accept

else halt & reject

If $f \in$ 3SAT: there is an assignment α s.t. $f(\alpha) = 1$

$\therefore$ on this nondet. choice, N accepts

If $f \notin$ 3SAT: $\forall \alpha \in \{0,1\}^n \quad f(\alpha) = 0$

therefore N will never accept on any computation path

3SAT in NP

Defn 2

$V(\langle f \rangle, c)$

- See if $\langle f \rangle$ is a legal encoding of a 3CNF formula. If not → halt & reject
- If so, Let n = # underlying variables
- Check if $c \in \{0,1\}^n$. If not halt & reject

Check if c satisfies f

if yes ⟶ halt & accept
if no ⟶ halt & reject

Correctness

(1) $V(\langle f \rangle, c)$ runs in time $poly(|\langle f \rangle|)$

$poly(n)$

(2) show If $\langle f \rangle \in 3SAT$ then $\exists c$ s.t. $V(\langle f \rangle, c)$ accepts

If $\langle f \rangle \in 3SAT$ by defn $\exists$ an assignment $c \in \{0,1\}^n$ such that $f(c) = 1$

$\therefore$ on this c, $V(\langle f \rangle, c)$ halts + accepts.

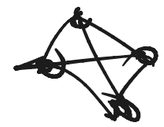
(3) show if $\langle f \rangle \notin 3SAT$ then $\forall c$, $V(\langle f \rangle, c)$ halts + rejects

By defn $\langle f \rangle \notin 3SAT \Rightarrow \forall c \in \{0,1\}^n \quad f(c) = 0$.

$\therefore$ verifier will never accept.

Examples of Languages in NP

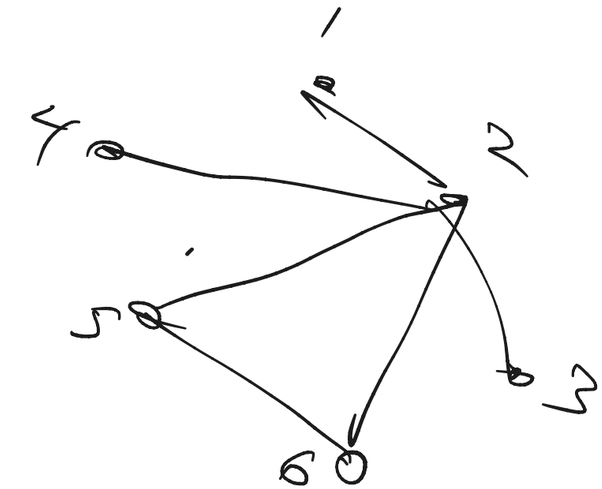
K=4



① Any $L \in P$ is also in NP

Verifier V on input (w, c) : ignore c and just run polytime alg for L on input w.

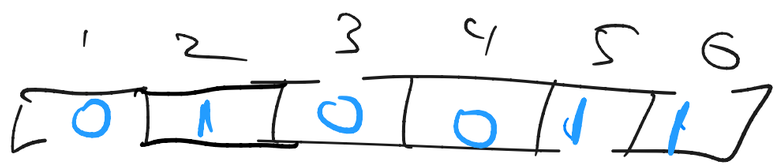
② CLIQUE (g, k). $g = (V, E)$ $|V| = n$



Verifier V on input $(w = (g, k), c)$:

- check that c encodes a subset $V' \subseteq V$ of k vertices
- For all pairs of vertices $i, j \in V'$ check if (i, j) is an edge in E (i.e., $(i, j) \in E$)

c:



Examples of Languages in NP

① Any $L \in P$ is also in NP

Verifier V on input (w, c) : ignore c and just run polytime alg for L on input w.

② CLIQUE $= \{ (g, k) \mid g$ is an undirected graph containing a size-k clique$\}$

Verifier V on input $((g,k), s)$:

- check that S encodes a subset $S \subseteq V$ of k vertices
- For all pairs of vertices $i, j \in V'$ check if (i,j) is an edge in E (i.e., $(i,j) \in E$)

③ k-SAT

Examples of Languages in NP

(3) $K\text{-SAT} = \{ \phi \mid \phi \text{ is a satisfiable } K\text{-CNF formula} \}$

Input is a Boolean formula over $x_1 \dots x_n$ in kCNF form.

kCNF form: $C_1 \wedge C_2 \wedge \dots \wedge C_m$

where each C_i is an OR of $\leq k$ literals

Example: $\phi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_1) \wedge (\bar{x}_2 \vee x_3)$

ϕ is satisfiable if there is a 0/1 assignment $\alpha \in \{0,1\}^n$ the variables of ϕ such that $\phi(\alpha) = 1$

$x_1 = 0$ $x_2 = 0$ $x_3 = 1$ $x_4 = 0$ satisfies ϕ

Examples of Languages in NP

(3) $K\text{-SAT} = \{ \phi \mid \phi \text{ is a satisfiable } K\text{-CNF formula} \}$

Input is a Boolean formula over $x_1 \dots x_n$ in KCNF form.

KCNF form: $C_1 \wedge C_2 \wedge \dots \wedge C_m$

Example: $\phi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_1) \wedge (\bar{x}_2 \vee x_3)$

ϕ is satisfiable if there is a 0/1 assignment $\alpha \in \{0,1\}^n$ to the variables of ϕ such that $\phi(\alpha) = 1$

Let $\alpha = x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1$. ϕ is satisfiable since $\phi(\alpha) = 1$

Example 2

$\phi' = (x_1 \vee x_2)(\bar{x}_1 \vee x_2)(\bar{x}_3 \vee x_4 \vee \bar{x}_2)(x_3)(\bar{x}_4 \vee \bar{x}_2)$

this is unsatisfiable. (check all 2^4 assignments)

$x_3 = 1 \quad x_2 = 1 \quad x_4 = 0$

Examples of Languages in NP

(3) $K\text{-SAT} = \{ \phi \mid \phi \text{ is a satisfiable } K\text{-CNF formula} \}$

Input is a Boolean formula over $x_1 \dots x_n$ in KCNF form.

KCNF form: $C_1 \wedge C_2 \wedge \dots \wedge C_m$

Example: $\phi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_1) \wedge (\bar{x}_2 \vee x_3)$

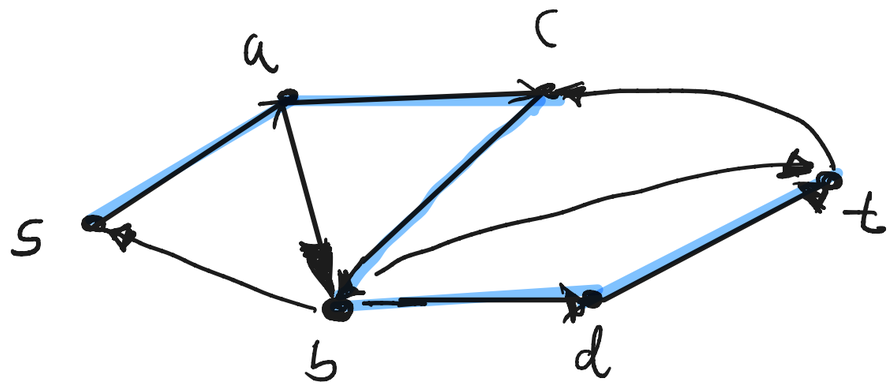
Verifier V on input (ϕ, α):

check that α is a Boolean satisfying assignment for ϕ. If yes $\rightarrow$ accept, otherwise $\rightarrow$ reject

Examples of Languages in NP

④ HAMPATH = $\{(g,s,t) \mid g$ is a directed graph containing a Hamilton path (visits all vertices once) from s to $t\}$

Example



p: s a c b d t

p: s b d c a t

Verifier V on input $((g,s,t), p)$:

Check if p encodes a Hamiltonian path from s to t.
If yes → accept; otherwise → reject

The Ubiquity of NP

→ It turns out there are thousands of problems in NP!

→ Many NP problems are fundamental in their respective areas of study

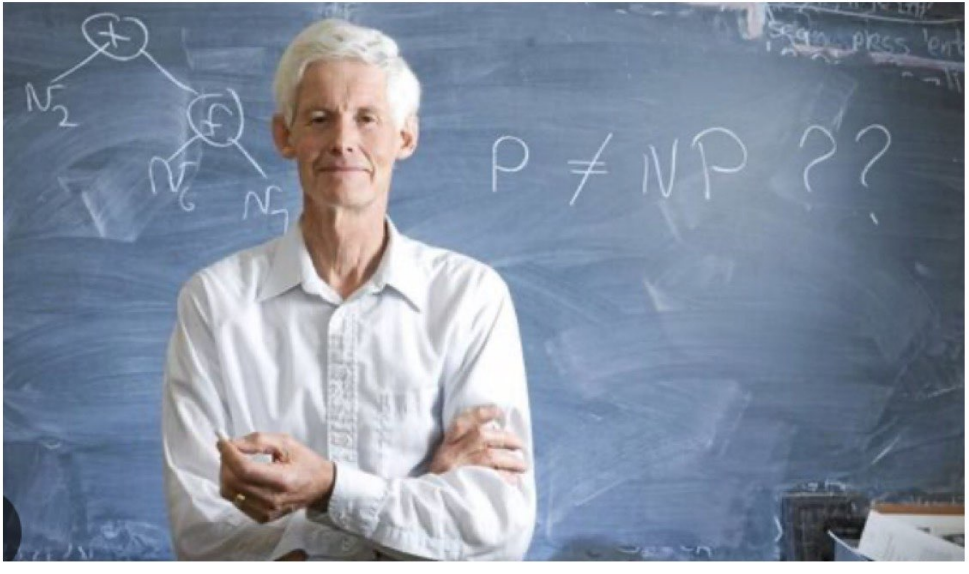
→ BIG QUESTION: $P \stackrel{?}{=} NP$


P=NP
$e^{\pi i} = -1$

The $1M question

The Clay Mathematics Institute
Millennium Prize Problems

1. Birch and Swinnerton-Dyer Conjecture
2. Hodge Conjecture
3. Navier-Stokes Equations
4. P vs NP
5. Poincaré Conjecture
6. Riemann Hypothesis
7. Yang-Mills Theory


P ≠ NP ??

NP-completeness

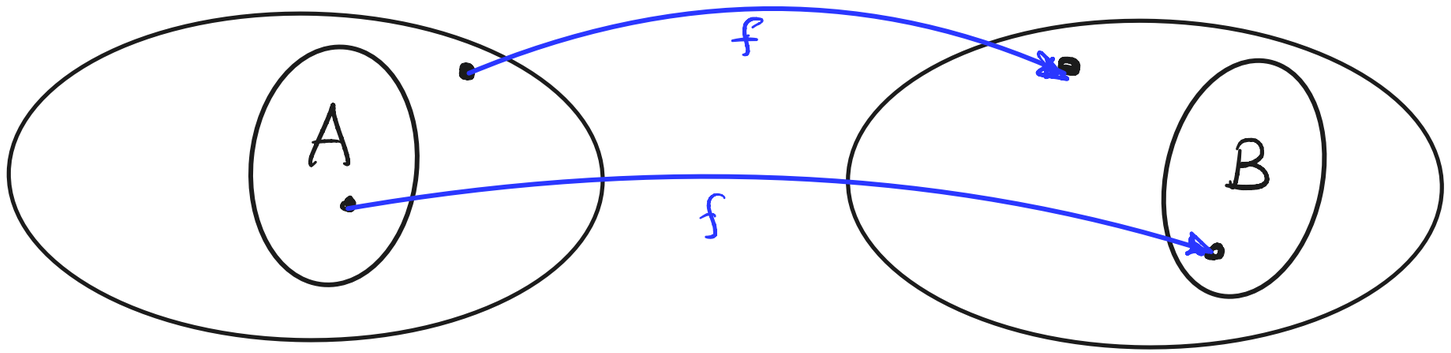
Cook (my advisor) and independently Levin established in 1970's that certain problems in NP (called NP-complete languages) whose individual complexity as the entire class of all NP problems!
For Example 3-SAT is NP-complete which implies that if there is a polytime algorithm for 3-SAT then all languages in NP are in P.

To formalize NP-completeness we need the notion of a polynomial-time reduction. This is just like the reductions we defined in previous section on computability, but now we require that the reduction is polynomial-time computable.

NP completeness idea: There are natural languages in NP (like SAT + 3SAT) such that ~~if~~ solving this language in polytime implies all NP languages solvable in polytime.

NP - completeness

Definition Language A is polynomial-time (mapping) reducible to B (written $A \leq_p B$) if there is a polynomial-time computable function $f : \Sigma^* \to \Sigma^*$ such that $w \in A \iff f(w) \in B$



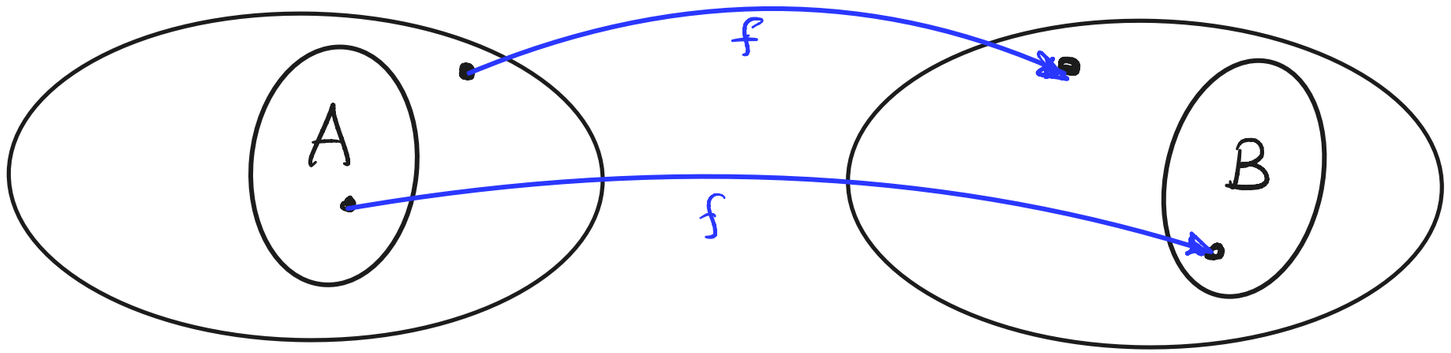
Definition

- A language $B \subseteq \{0,1\}^*$ is NP-hard if for every $A \in NP$ there is a polynomial time reduction from A to B ($A \leq_p B$)

$$A \leq_p B$$

NP - completeness

Definition Language A is polynomial-time (mapping) reducible to B (written $A \leq_p B$) if there is a polynomial-time computable function $f : \Sigma^* \to \Sigma^*$ such that $w \in A \iff f(w) \in B$



Definition

- A language $B \subseteq \{0,1\}^*$ is NP-hard if for every $A \in NP$ there is a polynomial time reduction from A to B ($A \leq_p B$)
- $B \subseteq \{0,1\}^*$ is NP-complete if: (i) B is in NP and (ii) B is NP-hard

- We will prove Cook-Levin Theorem Next week.

- We currently cannot show $P \neq NP$, and therefore we don't know if 3-SAT is in P or not.

- Best evidence that a problem in NP is computationally infeasible (not in P) is by showing it is NP-complete.

- Next: Prove other Languages are NP-complete via <u>reductions</u>.

(Analogous to: Proving other Languages not decidable, once we have one undecidable language)