<u>Def'n</u> Given functions $f, g: \mathbb{N} \to \mathbb{N}$, we say f is $O(g)$ ("on the order of g") if there is n_0 and C s.t. for all $n \geq n_0$, we have $f(n) \leq C g(n)$

- e.g. if $p(n)$ is a polynomial of one variable w/ coeffs in $\mathbb{N}$, then p is $O(n^d)$ for large enough d
  - → e.g. if $p(n) = 5n^3 + 2n^2 + 3$, p is $O(n^3)$ since $p(n) \leq 6n^3$ for large enough n

    C

<u>Def'n</u> Given a function $T: \mathbb{N} \to \mathbb{N}$ let TIME(T) denote the collection of all decision problems (A, P) that can be solved in $O(T)$ time (in the length of the problem instance)

More precisely: (A,P) belongs to TIME(T) if there is a function $t: \mathbb{N} \to \mathbb{N}$ which is $O(T)$ and a TM on some alphabet $B \supseteq A \cup \{Y, N\}$ s.t. for $w \in A^*$ we have:

① if $w \in P$, then on input w M halts in $\leq t(|w|)$ steps w/ output Y

② if $w \notin P$, then on input w M halts in $\leq t(|w|)$ steps w/ output N

- when should we consider a given problem (A,P) to be tractable? (i.e. efficiently decidable?)
- turns out: there is a major dividing line between problems that can be solved in polynomial time, and those that cannot

<u>Def'n</u> A decision problem (A,P) is in the class $\mathcal{P}$ if it is $TIME(n^d)$ for some $d \geq 0$

... i.e. if it can be decided in "polynomial time"

<u>Ex's</u> ① ELEMENTARY ALGEBRA is not in $\mathcal{P}$

In fact: it is in $TIME(2^{2^{cn}})$ for some $c > 0$ but not in $TIME(2^{dn})$ for any $d > 0$

... i.e. it can only be solved in general in super exponential time

② SATISFIABILITY may or may not be in $\mathcal{P}$, as best we know...

... but probably not!

Why? brings us to:

The class NP

- there is a class of problems that seem to be intractable (ie not in P) but no one has been able to prove this.
    - the NP problems

- intuitively: a decision problem (A, P) is in NP if, given a word $w \in A^*$ and a supposed "proof" or "guess" that $w \in P$, or "witness"

  we can check in polynomial time whether this guess works or not.

- e.g. SATISFIABILITY is in NP since:
    - if $\varphi(p_1, \ldots, p_n)$ is a prop'l fmla w/ vars among $p_1, \ldots, p_n$ (e.g. $(p_1 \wedge \neg p_2) \vee (\neg p_1 \wedge p_2)$)
    - and τ is a truth assignment for these variables (our "guess") (e.g. $\tau(p_1) = 0$, $\tau(p_2) = 1$)
    - we can efficiently check (i.e. in poly time) whether $\tau \models \varphi$ (in this case: Yes)

↳ this is different than checking if φ is satisfiable from scratch, which in principle involves checking all 2^n possible truth assignments to see if any work.
↳ here we need only check that the given assignment works

- Also: TRAVELING SALESMAN is NP: given a tour of the cities we can quickly check (adding the pairwise distances) if this tour has total distance $\leq B$
↳ different than checking if there is such a tour from scratch, which in principle would involve checking all possible tours (factorial many in the # of cities)

- more precisely, have following defn:

Def'n: A decision problem (A, P) is in NP if there is a TM M on an alphabet $B \supseteq A \cup \{Y, N\}$ and a polynomial $p(n)$ s.t. for any word $w \in A^*$ we have:

→ over

our "given"

$w \in P$ iff there is $v \in B^*$ s.t. $|v| \leq p(|w|)$ and on input $w*v$ M stops w/ output Y after at most $p(n)$ steps.

can check given is correct in poly time

↳ using this def'n we could (if we liked) formalize the sketched arguments above to show SAT and TRAV SAL are in NP

- observe: we clearly have $P \subseteq NP$: if we can already determine in poly time if a given word $w \in A^*$ belongs to P, then we don't need the help of any "given" v to return Y/N in poly time.

- following is one of the most famous open problems in math:

<u>Problem</u>. is $P = NP$?

- widely answered answer is no, i.e. problems like SAT for which there is no <u>apparent</u> poly time algorithm actually have no such algorithm

NP-complete problems

- can be shown: certain NP problems are "as hard as possible" among all NP problems
- these are the "NP-complete" probs
- in some sense, solving an NP-complete problem allows you to solve every other NP problem
- to formalize this, need concept of a reduction.

Def'n sps (A,P) and (B,Q) are decision problems.

A function $f: A^* \to B^*$ is called a reduction of P to Q if for every $w \in A^*$ we have:

$$w \in P \text{ iff } f(w) \in Q$$

- idea of a reduction is: if we have a decision algorithm M for Q then, using f, can get a decision algorithm M' for P
  - ↳ M' is: given $w \in A^*$, compute $f(w)$, apply M to check if $f(w) \in Q$ (true iff $w \in P$)

- for a reduction f to be useful, need the time it takes to compute $f(w)$ to be on same order of time it takes M to determine if $f(w) \in Q$

<u>Def'n</u> Sps A, B are finite alphabets, a function $f: A^* \to B^*$ is <u>polynomial time computable</u> if there is a TM M on a finite $C \supseteq A \cup B$ and a polynomial p s.t. for every word $w \in A^*$, M terminates on input w in at most $p(|w|)$ many steps w/ output $f(w)$

"f can be computed (by a TM) in polynomial time on the length of the input"

<u>Def'n</u> A decision problem (B, Q) is <u>NP-complete</u> if it is NP and for any NP problem (A, P) there is a polynomial time computable $f: A^* \to B^*$ ~~from~~ P to Q

ie $w \in P$ iff $f(w) \in Q$

- NP-complete are "universal" or "as hard as possible" among NP problems unlikely!

- to show $P = NP$, is sufficient to show that a single NP-complete problem (B, Q) is in P

assuming such a prob exists!
(more in a moment)

- then: since any other NP-problem (A, P) can be reduced to (B, Q) (in poly time) and we can solve Q in poly time, we can solve P in poly time, giving $NP = P$.

- and it turns out: there are NP-complete problems!

<u>Theorem</u> SATISFIABILITY is NP-complete.

(It can also be shown TRAV SAL is NP-complete; and there are many other natural problems which are NP-complete)

<u>Pf (Rough sketch)</u>: - Sps (A, P) is in NP

- need to find a poly time reduction f of P to SAT
- let M be TM associated to (A, P)

(M checks in poly time, given $w \in A^*$ and a "given" v whether v shows $w \in P$)

- we find a poly time reduction $f: A^* \to B^*$ of P to SAT

  i.e. $w \in P$ iff $f(w)$ is satisfiable

  a prop'l fmla

long we use to write prop'l fmlas.

- the idea: $f(w)$ will be a prop'l fmla into which we have "programmed" M.

- how to find such a fmla?
- fix $w = a_1 \ldots a_n \in A^*$
- introduce prop'l variables

  $R_{i,k}$

  $H_{i,j}$

  $T_{i,j,l}$

- intended meanings are

  $R_{i,k}$ true iff at time i, M in state Q_k

  $H_{i,j}$ true iff at time i, M's head over jth square

  $T_{i,j,l}$ true iff at time i, tape contains lth symbol in square j

- since M only needs $p(n)$ many steps to complete its computation on w by assumption, need only have such variables for $i \leq p(n)$ and $-p(n) \leq j \leq p(n)$

- we can program M's table into prop'l clauses
- e.g. if $(Q_k, \sigma_e, \sigma_{e'}, +1, Q_{k'})$ is in the table, can write, for all $i \leq p(n)$, clauses
$$(R_{i,k} \wedge H_{i,j} \wedge T_{i,j,e} \Rightarrow R_{i+1,k'} \wedge H_{i+1,j+1} \wedge T_{i+1,j,e'})$$

- intuitively asserts: if at time i M is in state Q_k, head is over square j and σ_e is written, then at time $i+1$, square j contains $\sigma_{e'}$, head is at square $i+1$

- there are details to fill in and more clauses we need to write to make everything work but at the end if we take the conjunction of all these clauses, we get a fmla that is satisfiable (by a truth assignment corresponding to a computation

of M on input $w * v$)
iff $w \in P$ (as checked by inputting $w * v$ in M)

- since this fmla, call it $f(w)$, can "clearly" be constructed in poly time (in $|w|$) we have the desired reduction ✓