CS 154

Time Hierarchy, P and NP

An Efficient Universal TM

Theorem: There is a (one-tape) Turing machine U which takes as input:

- the code of an arbitrary TM M
- an input string w
- and a string of t 1s, $t > |w|$

such that $U(M, w, 1^t)$ halts in $O(|M|^2 t^2)$ steps and U accepts $(M, w, 1^t)$ $\Leftrightarrow$ M accepts w in t steps

The Universal TM with a Clock

Idea: Make a multi-tape TM U' that does the above, and runs in $O(|M|\ t)$ steps

The Time Hierarchy Theorem

Intuition: If you get more time to compute, then you can solve strictly more problems.

Theorem: For all "reasonable" f, g : $\mathbb{N} \to \mathbb{N}$ where for all n, $g(n) > n^2 f(n)^2$, $\text{TIME}(f(n)) \subsetneq \text{TIME}(g(n))$

Proof Idea: Diagonalization with a clock.

Make a TM N that on input M, simulates the TM M on input M for f(|M|) steps, then flips the answer.

Then, L(N) cannot have time complexity f(n)

The Time Hierarchy Theorem

Theorem: For "reasonable" f, g where $g(n) > n^2 f(n)^2$,

$$TIME(f(n)) \subsetneq TIME(g(n))$$

Proof Sketch: Define a TM N as follows:

N(M) = Compute $t = f(|M|)$

Run $U(M, M, 1^t)$ and output the opposite answer.

Claim: L(N) does not have time complexity f(n).

Proof: Assume N' runs in f(n) time, and L(N') = L(N).

By assumption, N'(N') runs in f(|N'|) time and

outputs the *opposite* answer of $U(N', N', 1^{f(|N'|)})$

But by definition of U, $U(N', N', 1^{f(|N'|)})$ accepts

$\Leftrightarrow$ N'(N') accepts in f(|N'|) steps.

This is a contradiction!

The Time Hierarchy Theorem

Theorem: For "reasonable" f, g where $g(n) > n^2 f(n)^2$,

$$\mathrm{TIME}(f(n)) \subsetneq \mathrm{TIME}(g(n))$$

Proof Sketch: Define a TM N as follows:

N(M) = Compute $t = f(|M|)$

Run $U(M, M, 1^t)$ and output the opposite answer.

So, L(N) does *not* have time complexity f(n).

What do we need in order for N to run in O(g(n)) time?

1. Compute $f(|M|)$ in $O(g(|M|))$ time ["reasonable"]
2. Simulate $U(M, M, 1^t)$ in $O(g(|M|))$ time

Recall: $U(M, w, 1^t)$ halts in $O(|M|^2 t^2)$ steps

Set g(n) so that $g(|M|) > |M|^2 f(|M|)^2$ for all n. QED

Remark: Time hierarchy also holds for multitape TMs!

A Better Time Hierarchy Theorem

Theorem: For "reasonable" f, g where $g(n) > f(n) \log^2 f(n)$, $\mathrm{TIME}(f(n)) \subsetneq \mathrm{TIME}(g(n))$

Corollary: $\mathrm{TIME}(n) \subsetneq \mathrm{TIME}(n^2) \subsetneq \mathrm{TIME}(n^3) \subsetneq \ldots$

There is an infinite hierarchy of
increasingly more time-consuming problems

Question: Are there important everyday problems that are high up in this time hierarchy?

A *natural* problem that *needs exactly* n^{10} time?

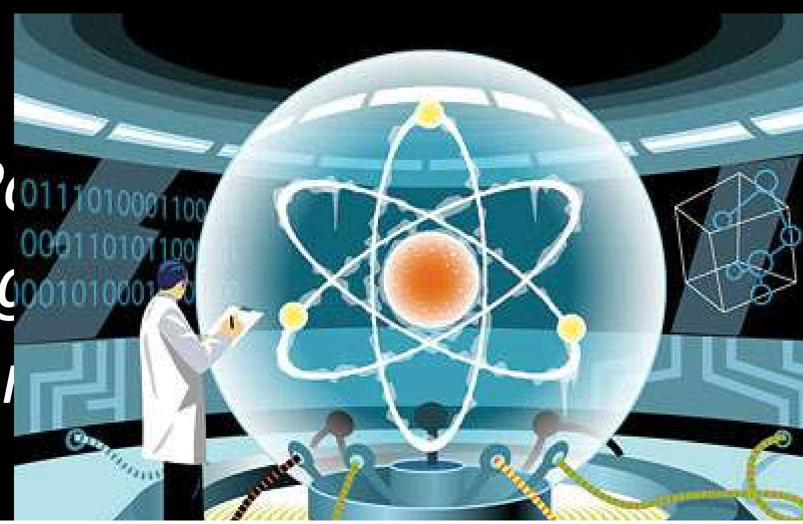
THIS IS AN OPEN QUESTION!

$$\mathbf{P} = \bigcup_{k \in \mathbf{N}} \mathrm{TIME}(n^k)$$

Polynomial Time

The EXTENDED Church-Turing Thesis

Everyone's Intuitive Notion of Efficient Algorithms = Polynomial-Time Turing Machines



A controversial thesis! *P*

include n^{100} time alg

algorithms, quantu

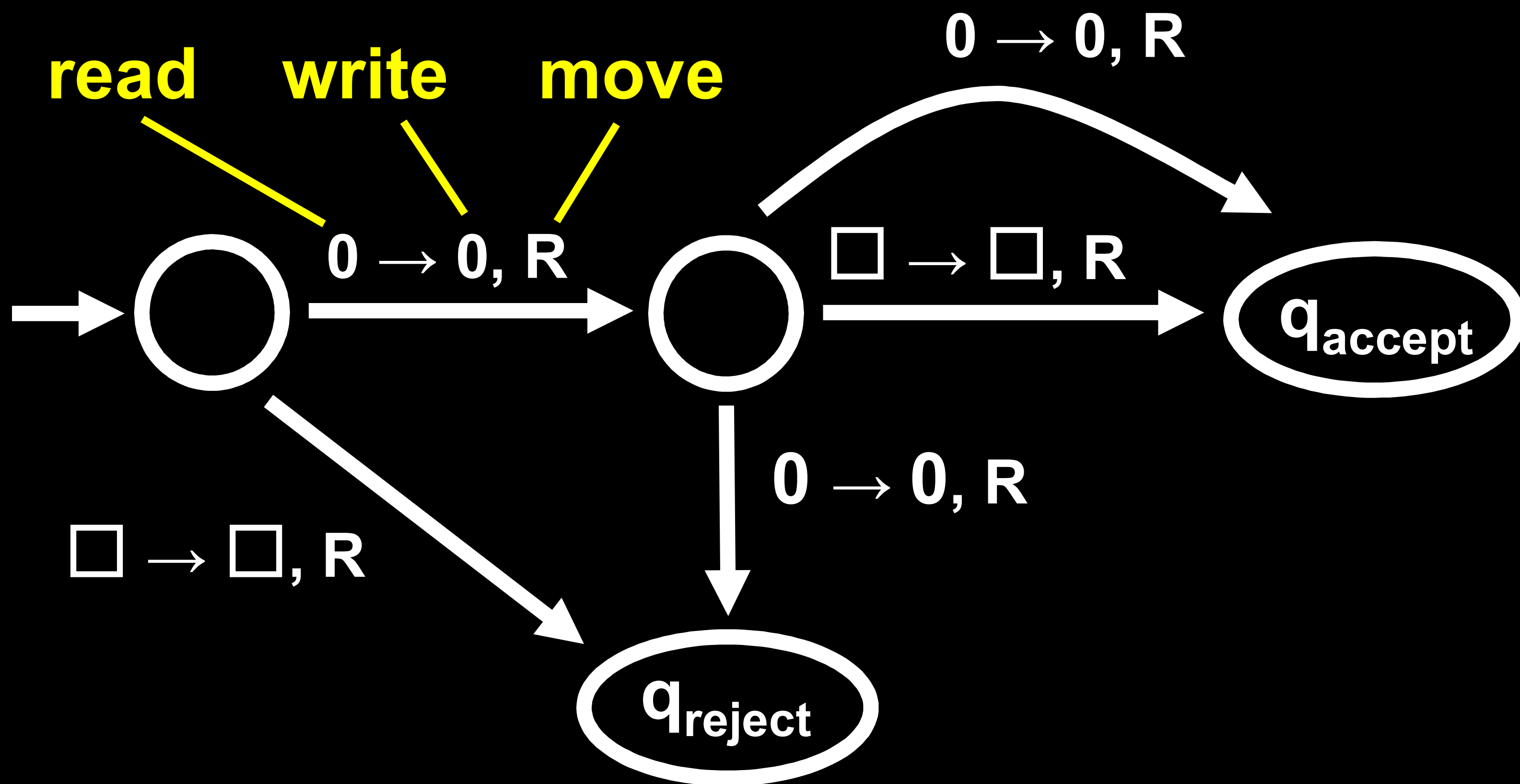
Nondeterminism and NP

Nondeterministic Turing Machines

…are just like standard TMs, except:

1. **The machine may proceed according to several possible transitions (like an NFA)**

2. **The machine *accepts* an input string if there *exists* an accepting computation history for the machine on the string**


read
write
move
0 → 0, R
0 → 0, R
□ → □, R
q_accept
0 → 0, R
□ → □, R
q_reject

Definition: A **nondeterministic** TM is a 7-tuple

$T = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where:

Q is a finite set of states

Σ is the input alphabet, where $\square \notin \Sigma$

Γ is the tape alphabet, where $\square \in \Gamma$ and $\Sigma \subseteq \Gamma$

$\delta : Q \times \Gamma \rightarrow 2^{(Q \times \Gamma \times \{L,R\})}$

$q_0 \in Q$ is the start state

$q_{accept} \in Q$ is the accept state

$q_{reject} \in Q$ is the reject state, and $q_{reject} \neq q_{accept}$

Defining Acceptance for NTMs

Let N be a nondeterministic Turing machine

An **accepting computation history for N on w** is a sequence of configurations $C_0, C_1, \ldots, C_t$ where

1. C_0 is the start configuration q_0w,
2. C_t is an accepting configuration,
3. Each configuration C_i yields C_{i+1}

Def. N(w) accepts in t time ⇔ Such a history exists

N has time complexity T(n) if for all n, for all inputs of length n and for all histories, N halts in T(n) time

Definition: NTIME(t(n)) =

{ L | L is decided by a O(t(n)) time nondeterministic Turing machine }

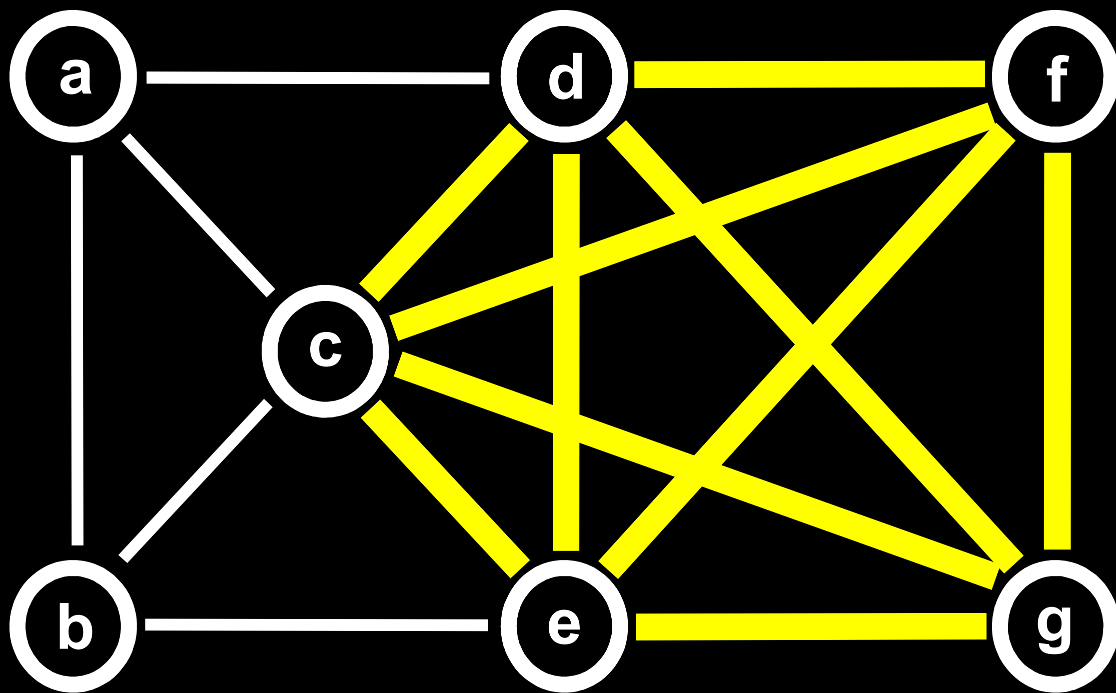
$\text{TIME}(t(n)) \subseteq \text{NTIME}(t(n))$

Is TIME(t(n)) = NTIME(t(n)) for all t(n)?

THIS IS AN OPEN QUESTION!

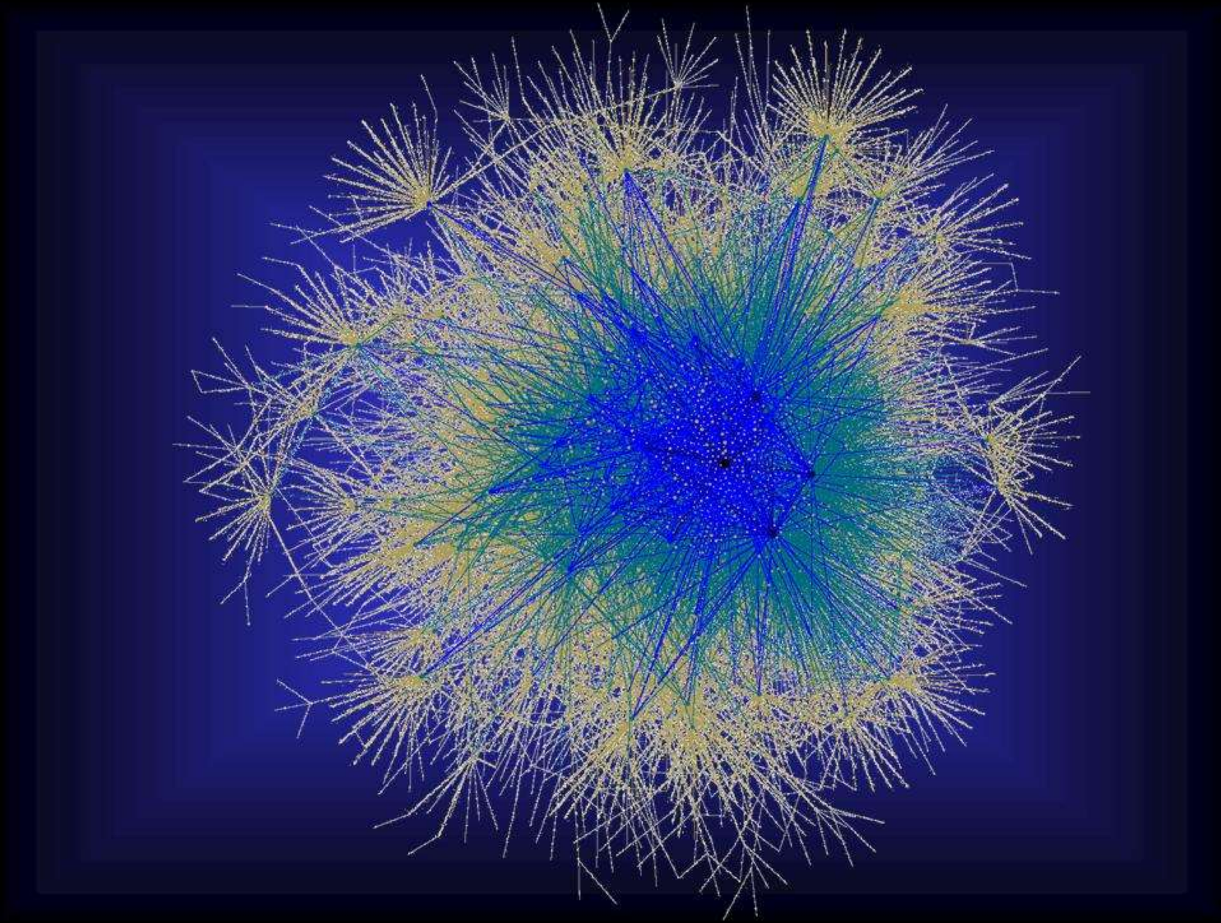
What problems can we efficiently solve nondeterministically, but not deterministically?

The Clique Problem



k-clique = complete subgraph on k nodes

The Clique Problem



Find a clique of 1 million nodes?

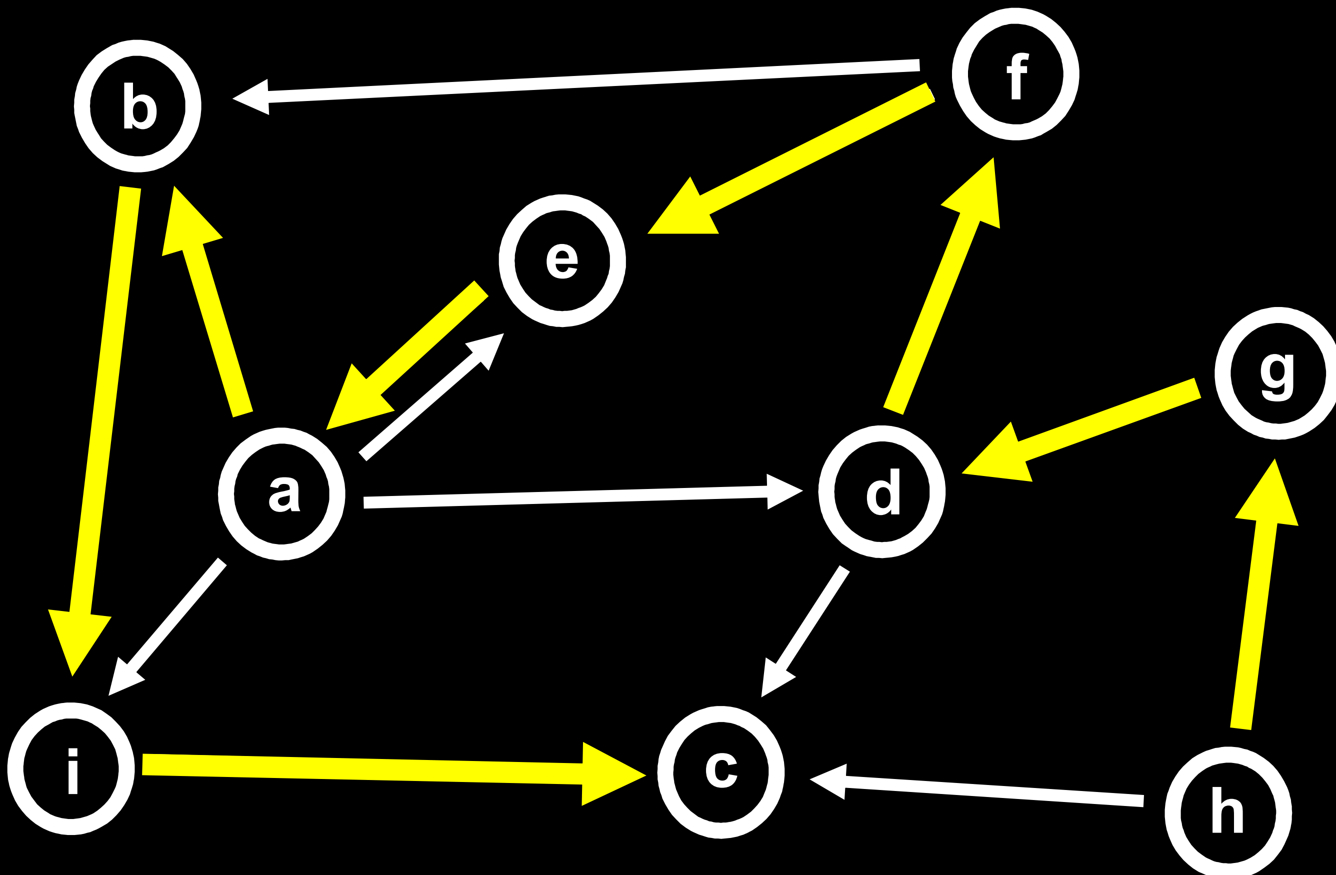
Assume a reasonable encoding of graphs (example: the adjacency matrix is reasonable)

CLIQUE = { (G,k) | G is an undirected graph with a k-clique }

Theorem: **CLIQUE $\in$ NTIME(n^c) for some c > 1**

N((V,E),k): **Nondeterministically guess**
a subset S of V with |S| = k
For all u, v in S,
if (u,v) is not in E then *reject*
Accept

The Hamiltonian Path Problem



A Hamiltonian path traverses through each node exactly once

HAMPATH = { (G,s,t) | G is a directed graph with a Hamiltonian path from s to t }

Theorem: HAMPATH $\in$ NTIME(n^c) for some c > 1

N((V,E),s,t): Nondeterministically guess
a sequence $v_1, \ldots, v_{|V|}$ of vertices
If $v_i = v_j$ for some $i \neq j$, *reject*
For all i = 1,…,|V|-1,
if (v_i,v_{i+1}) is not in E then *reject*
If (v_1 = s & v_n = t) then *accept* else *reject*

$$\mathbf{NP} = \bigcup_{k \in \mathbf{N}} \mathbf{NTIME}(n^k)$$

Nondeterministic Polynomial Time

Theorem: $L \in NP \Leftrightarrow$ There is a constant k and polynomial-time TM V such that

$$L = \{ x \mid \exists\, y \in \Sigma^* \; [\,|y| \leq |x|^k \text{ and } V(x,y) \text{ accepts} \,] \}$$

Proof: (1) If $L = \{ x \mid \exists y \; |y| \leq |x|^k \text{ and } V(x,y) \text{ accepts} \}$ then $L \in NP$

Define the NTM $N(x)$: Guess y of length at most $|x|^k$
Run $V(x,y)$ and output answer

Then, $L(N)$ is the set of x s.t. $[\,|y| \leq |x|^k$ & $V(x,y)$ accepts$]$

(2) If $L \in NP$ then

$$L = \{ x \mid \exists y \; |y| \leq |x|^k \text{ and } V(x,y) \text{ accepts} \}$$

Suppose N is a poly-time NTM that decides L.

Define $V(x,y)$ to accept iff y encodes an accepting computation history of N on x

A language L is in NP
if and only if
there are polynomial-length *proofs*
for membership in L

CLIQUE = { (G,k) | $\exists$ subset of nodes S such that S is a k-clique in G }

HAMPATH = { (G,s,t) | $\exists$ Hamiltonian path in graph G from node s to node t }

Boolean Formula Satisfiability

logical operations

parentheses

$$\phi = (\neg x \wedge y) \vee z$$

Boolean variables (0 or 1)

Boolean Formula Satisfiability

$$\phi = (\neg x \wedge y) \vee z$$

A satisfying assignment is a setting of the variables that makes the formula true

x = 1, y = 1, z = 1 is a satisfying assignment for ϕ
(in fact, any assignment with z = 1 is satisfying)

$$\phi = \neg(\underset{0}{x} \vee \underset{0}{y}) \wedge (\underset{1}{z} \wedge \neg \underset{0}{x})$$

A Boolean formula is satisfiable if there is a true/false setting to the variables that makes the formula true

SAT = { ϕ | ϕ is a satisfiable Boolean formula }

A **3cnf-formula** has the form:

$$(x_1 \lor \neg x_2 \lor x_3) \land (x_4 \lor x_2 \lor x_5) \land (x_3 \lor \neg x_2 \lor \neg x_1)$$

literals

clauses

3SAT = { ϕ | ϕ is a satisfiable 3cnf-formula }

3SAT = { ϕ | ϕ is a satisfiable 3cnf-formula }

Theorem: 3SAT $\in$ NP

We can express 3SAT as

3SAT = { ϕ | $\exists$ string y such that ϕ is in 3cnf and y encodes a satisfying assignment to ϕ }

The number of variables of ϕ is at most $|\phi|$, so $|y| \leq |\phi|$.

Then, argue that the language

3SAT-CHECK = {(ϕ,y) | ϕ is in 3cnf and y is a satisfying assignment to ϕ}

is in P.

(Similarly, SAT $\in$ NP)

NP = Problems with the property that, once you *have* the solution, it is "easy" to verify the solution

When $\phi \in$ SAT,
or (G, k) $\in$ CLIQUE,
or (G,s,t) $\in$ HAMPATH,
I can prove that fact to you
with a short proof that you can easily verify

What if $\phi \notin$ SAT? (G, k) $\notin$ CLIQUE?
Or (G,s,t) $\notin$ HAMPATH?

P = the problems that can be efficiently solved

NP = the problems where *proposed solutions can be efficiently verified*

Is P = NP?

can problem solving be automated?

$$$

P = NP?

$$$

If P = NP…

Mathematicians may be out of a job

Cryptography as we know it may be impossible

In principle, every aspect of life could be efficiently and globally optimized…
… life as we know it would be different!

Conjecture: $P \neq NP$

Polynomial Time Reducibility

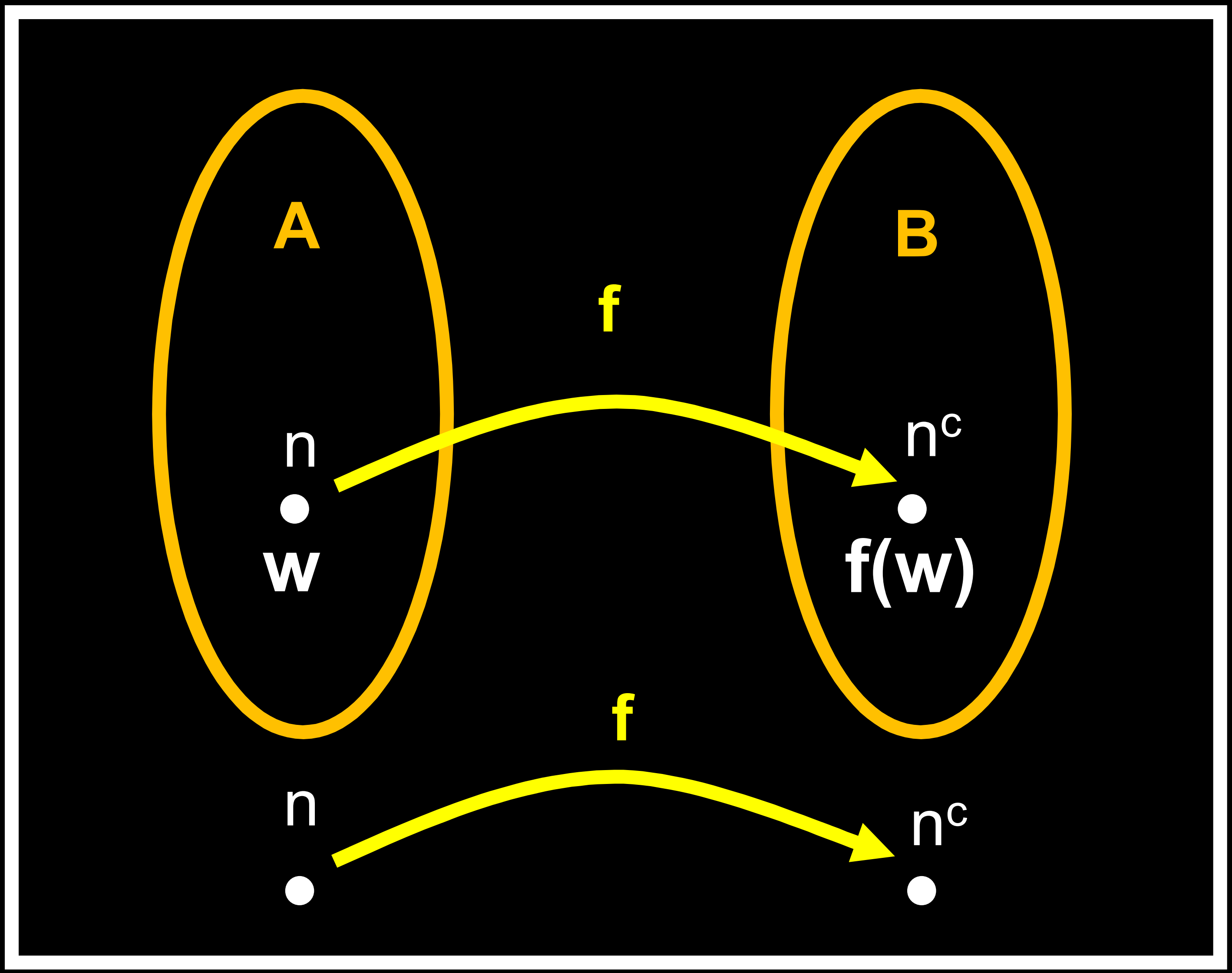
$f : \Sigma^* \to \Sigma^*$ is a **polynomial time computable function** if there is a poly-time Turing machine M that on every input w, halts with just f(w) on its tape

Language A is poly-time reducible to language B, written as $A \leq_P B$, if there is a poly-time computable $f : \Sigma^* \to \Sigma^*$ so that:

$$w \in A \Leftrightarrow f(w) \in B$$

f is a polynomial time reduction from A to B

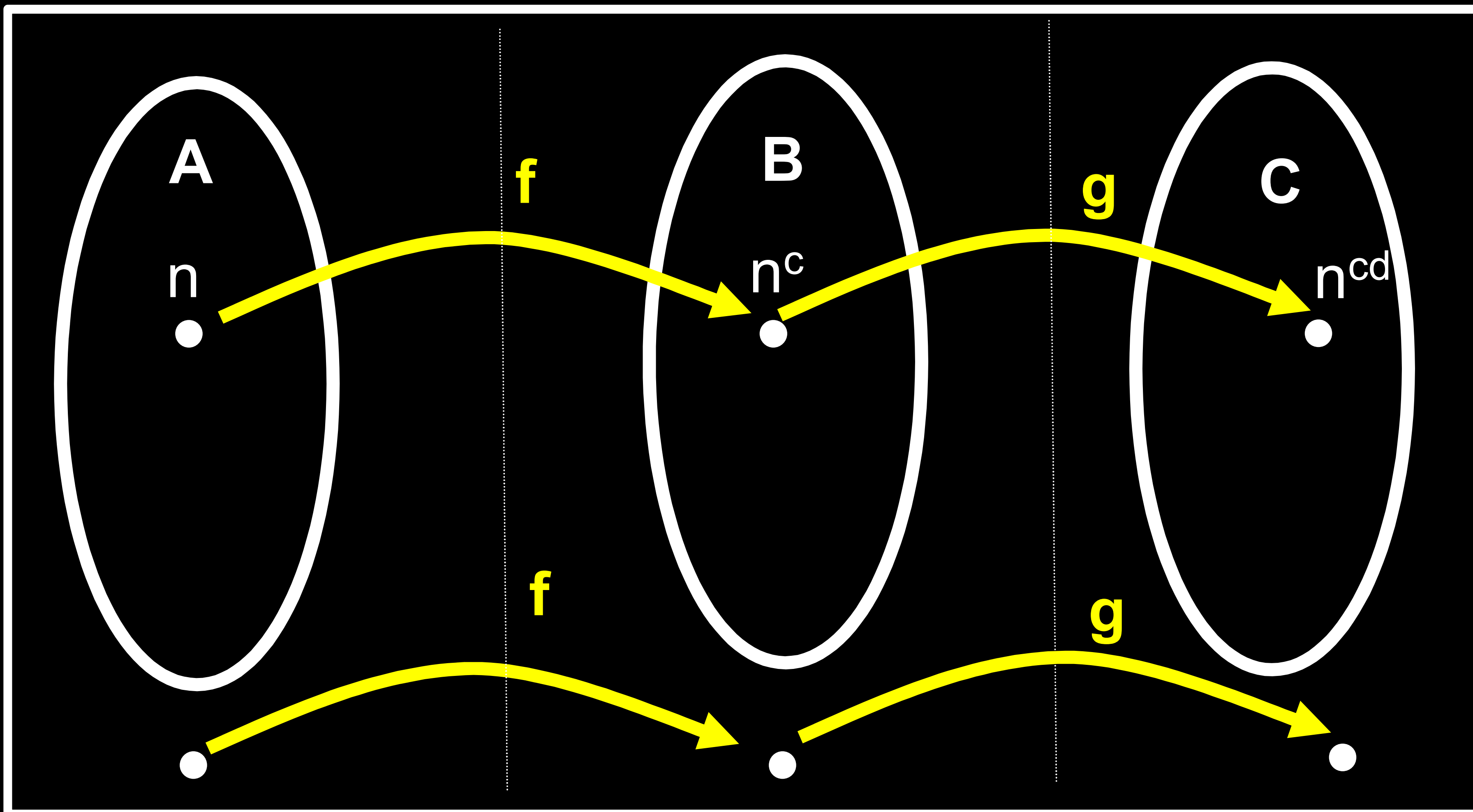
Note there is a k such that for all w, $|f(w)| \leq |w|^k$



f converts any string w into a string f(w) such that

$w \in A \Leftrightarrow f(w) \in B$

Theorem: If $A \leq_P B$ and $B \leq_P C$, then $A \leq_P C$



Theorem: If $A \leq_P B$ and $B \in P$, then $A \in P$

Proof: Let M_B be a poly-time TM that decides B. Let f be a poly-time reduction from A to B.

We build a machine M_A that decides A as follows:

M_A = On input w,

1. Compute f(w)
2. Run M_B on f(w), output its answer

$$w \in A \Leftrightarrow f(w) \in B$$