The EXTENDED Church-Turing Thesis

Everyone’s Intuitive Notion of Efficient Algorithms \( \subseteq \) Polynomial-Time Turing Machines

More generally: TM can simulate every “reasonable” model of computation with only polynomial increase in time

A controversial thesis! Potential counterexamples: quantum algorithms
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
Definition: A nondeterministic TM is a 7-tuple 
\( \Gamma = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}) \), where:

- \( Q \) is a finite set of states
- \( \Sigma \) is the input alphabet, where \( \square \notin \Sigma \)
- \( \Gamma \) 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
- \( q_{\text{accept}} \in Q \) is the accept state
- \( q_{\text{reject}} \in Q \) is the reject state, and \( q_{\text{reject}} \neq q_{\text{accept}} \)
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 $q_0 w$,
2. $C_t$ is an accepting configuration,
3. Each configuration $C_i$ yields $C_{i+1}$

Def. $N(w)$ accepts in $t$ time $\iff$ 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: \( \text{NTIME}(t(n)) = \{ L \mid \text{L is decided by a } O(t(n)) \text{ time nondeterministic Turing machine} \} \)

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

Is \( \text{TIME}(t(n)) = \text{NTIME}(t(n)) \) for all \( t(n) \)?
What problems can we efficiently solved 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)

\[ \text{CLIQUE} = \{ (G,k) \mid G \text{ is an undirected graph with a } k\text{-clique} \} \]

Theorem: \( \text{CLIQUE} \in \text{NTIME}(n^c) \) for some \( c > 1 \)

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

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 ∈ 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,\ldots,|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
Nondeterministic Polynomial Time

\[ \text{NP} = \bigcup_{k \in \mathbb{N}} \text{NTIME}(n^k) \]
Theorem: \( L \in \text{NP} \iff \) 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 \( L \in \text{NP} \)

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. \( |y| \leq |x|^k \& V(x,y) \text{ accepts} \)

(2) If \( L \in \text{NP} \) 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* (aka. *certificates* or *witnesses*) for membership in $L$.

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

$\text{HAMPATH} = \{ (G,s,t) \mid \exists \text{ Hamiltonian path in graph } G \text{ from node } s \text{ to node } t \}$
Boolean Formula Satisfiability

\[ \phi = (\neg x \land y) \lor z \]

logical operations
parentheses

\[ \neg \text{ recedes} \]
\[ \land \text{ precedes } \lor \]

Boolean variables (0 or 1)
Boolean Formula Satisfiability

\[ \phi = (\neg x \land y) \lor z \]

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 \( \phi \)
(in fact, any assignment with \( z = 1 \) is satisfying)

\[ \phi = (x \lor y) \land (z \land \neg x) \]

\[
\begin{array}{ccc}
0 & 0 & 1 \\
0 & 1 & 0
\end{array}
\]
A Boolean formula is **satisfiable** if there is a true/false setting to the variables that makes the formula true.

YES: \( a \land b \land c \land \neg d \)

NO: \( \neg(x \lor y) \land x \)

\[ SAT = \{ \phi \mid \phi \text{ is a satisfiable Boolean formula} \} \]
A 3cnf-formula has the form:

\[(x_1 \lor \neg x_2 \lor x_3) \land (x_4 \lor x_2 \lor x_5) \land (x_3 \lor \neg x_2 \lor \neg x_1)\]

3SAT = \{ \phi \mid \phi \text{ is a satisfiable 3cnf-formula} \}
Theorem: \( 3\text{SAT} \in \text{NP} \)

We can express \( 3\text{SAT} \) as

\[ 3\text{SAT} = \{ \phi \mid \phi \text{ is in 3cnf and } \exists \text{ string } y \text{ that encodes a satisfying assignment to } \phi \} \]

The number of variables of \( \phi \) is at most \(|\phi|\), so \(|y| \leq |\phi|\).

Then, argue that the language

\[ 3\text{SAT-CHECK} = \{(\phi,y) \mid \phi \text{ is in 3cnf and } y \text{ is a satisfying assignment to } \phi\} \]

is in \( \text{P} \).

(Similarly, \( \text{SAT} \in \text{NP} \))
NP = Problems with the property that, once you have the solution, it is “easy” to verify the solution

When $\phi \in \text{SAT}$, or $(G, k) \in \text{CLIQUE}$, or $(G, s, t) \in \text{HAMPATH}$,

Can prove that with a short proof that can easily been verified

What if $\phi \notin \text{SAT}$? $(G, k) \notin \text{CLIQUE}$? Or $(G, s, t) \notin \text{HAMPATH}$?
$P$ = 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?*

Clay Math Institute in the year 2000: “millennium problems”
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 ≠ NP
Polynomial Time Reducibility

$f : \Sigma^* \rightarrow \Sigma^*$ 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 $f : \Sigma^* \rightarrow \Sigma^*$ so that:

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

$f$ 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

\[ w \in A \iff f(w) \in B \]
Theorem: If $A \leq_B B$ and $B \leq_C C$, 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

$w \in A \iff f(w) \in B$
Theorem: If $A \leq_p B$ and $B \in \text{NP}$, then $A \in \text{NP}$

Proof: Analogous...
Theorem: If $A \leq_p B$ and $B \in P$, then $A \in P$

Theorem: If $A \leq_p B$ and $B \in NP$, then $A \in NP$

Corollary: If $A \leq_p B$ and $A \notin P$, then $B \notin P$
Definition: A language $B$ is NP-complete if:

1. $B \in \text{NP}$
2. Every $A$ in NP is poly-time reducible to $B$
   That is, $A \leq_{p} B$

When this is true, we say “$B$ is NP-hard”

On homework, you showed

A language $L$ is recognizable iff $L \leq_{m} A_{\text{TM}}$

$A_{\text{TM}}$ is “complete for recognizable languages”:

$A_{\text{TM}}$ is recognizable, and for all recognizable $L$, $L \leq_{m} A_{\text{TM}}$
Suppose $L$ is NP-Complete...

If $L \in P$, then $P = NP$

If $L \notin P$, then $P \neq NP$
Suppose $L$ is NP-Complete...

Then assuming the conjecture $P \neq NP$, $L$ is not decidable in $n^k$ time, for every $k$. 
The Cook-Levin Theorem: SAT and 3SAT are NP-complete

1. 3SAT ∈ NP
   A satisfying assignment is a “proof” that a 3cnf formula is satisfiable

2. 3SAT is NP-hard
   Every language in NP can be polynomial-time reduced to 3SAT (complex logical formula)

Corollary: 3SAT ∈ P if and only if P = NP