Time Hierarchy Theorem

**Theorem:** for every proper complexity function $f(n) \geq n$:

$\text{TIME}(f(n)) \not \subseteq \text{TIME}(f(2n)^3)$.

- **Proof idea:**
  - use diagonalization to construct a language that is not in $\text{TIME}(f(n))$.
  - constructed language comes with a TM that decides it and runs in time $f(2n)^3$.

Recall proof for Halting Problem

- **Proof:**
  - $\text{SIM}$ is TM deciding language
    \[ \{ <M, x> : M \text{ accepts } x \text{ in } \leq f(|x|) \text{ steps} \} \]
  - Claim: $\text{SIM}$ runs in time $g(n) = f(n)^3$.
  - define new TM $D$: on input $<M>$
    - if SIM accepts $<M>$, reject
    - if SIM rejects $<M>$, accept
  - $D$ runs in time $g(2n)$
Proof of Time Hierarchy Theorem

• Proof (continued):
  – suppose M in \textsc{Time}(f(n)) decides L(D)
  - M(<M>) = SIM(<M, <M>>) ≠ D(<M>)
  - but M(<M>) = D(<M>)
  - contradiction.

Proof of Time Hierarchy Theorem

• Claim: there is a TM SIM that decides 
  \{<M, x> : M accepts x in \leq f(|x|) steps\}
  and runs in time \(g(n) = f(n)^3\).
• Proof sketch: SIM has 4 work tapes
  - contents and "virtual head" positions for M’s tapes
  - M’s transition function and state
  - \(f(|x|)^+\)’s used as a clock
  - scratch space

Proof of Time Hierarchy Theorem

• Proof sketch (continued): 4 work tapes
  - contents and “virtual head” positions for M’s tapes
  - M’s transition function and state
  - \(f(|x|)^+\)’s used as a clock
  - scratch space
  – initialize tapes
  – simulate step of M, advance head on tape 3; repeat.
  – can check running time is as claimed.

So far…

• We have defined the complexity classes P (polynomial time), EXP (exponential time)

Poly-time reductions

• Type of reduction we will use:
  – “many-one” poly-time reduction (commonly)
  – “mapping” poly-time reduction (book)

Poly-time reductions

• function f should be poly-time computable

\textbf{Definition}: \(f : \Sigma^* \rightarrow \Sigma^*\) is poly-time computable if for some \(g(n) = n^O(1)\) there exists a \(g(n)\)-time TM \(M_f\) such that on every \(w \in \Sigma^*\), \(M_f\) halts with \(f(w)\) on its tape.
**Poly-time reductions**

**Definition:** A \(\leq_p B\) (“A reduces to B”) if there is a poly-time computable function \(f\) such that for all \(w\)
\[
w \in A \iff f(w) \in B
\]
- as before, condition equivalent to:
  - YES maps to YES and NO maps to NO
- as before, meaning is:
  - B is at least as “hard” (or expressive) as A

**Theorem:** if \(A \leq_p B\) and \(B \in P\) then \(A \in P\).

**Proof:**
- a poly-time algorithm for deciding A:
  - on input \(w\), compute \(f(w)\) in poly-time.
  - run poly-time algorithm to decide if \(f(w) \in B\)
- if it says “yes”, output “yes”
- if it says “no”, output “no”

**Example**
- \(2SAT = \{\text{CNF formulas with 2 literals per clause for which there exists a satisfying truth assignment}\}\)
- \(L = \{\text{directed graph } G, \text{ and list of pairs of vertices} (u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k), \text{ such that there is no } i \text{ for which } [u_i \text{ is reachable from } v_i \text{ in } G \text{ and } v_i \text{ is reachable from } u_i \text{ in } G]\}\)
- We gave a poly-time reduction from \(2SAT\) to \(L\).
- determined that \(2SAT \in P\) from fact that \(L \in P\)

**Hardness and completeness**
- Reasonable that can efficiently transform one problem into another.
- Surprising:
  - can often find a special language \(L\) so that every language in a given complexity class reduces to \(L\)!
  - powerful tool

**Definition:** a language \(L\) is **C-hard** if for every language \(A \in C\), A poly-time reduces to \(L\); i.e., \(A \leq_p L\).

**Definition:** a language \(L\) is **C-complete** if \(L\) is C-hard and \(L \in C\)

meaning: \(L\) is a “hardest” problem in \(C\)
An EXP-complete problem

• Version of $A_{TM}$ with a time bound:
  $A_{TM}^B = \{<M, x, m> : M \text{ is a TM that accepts } x \text{ within at most } m \text{ steps}\}$

**Theorem:** $A_{TM}^B$ is EXP-complete.

**Proof:**
– what do we need to show?

An EXP-complete problem

• $A_{TM}^B = \{<M, x, m> : M \text{ is a TM that accepts } x \text{ within at most } m \text{ steps}\}$
• Proof that $A_{TM}^B$ is EXP-complete:
  – Part 1. Need to show $A_{TM}^B \in \text{EXP}$.
    • simulate $M$ on $x$ for $m$ steps; accept if simulation accepts; reject if simulation doesn’t accept.
    • running time $m^2(1)$.
  • $n = \text{length of input} \geq \log_2 m$
  • running time $\leq m^k = 2^{k\log m} \leq 2^{kn}$

An EXP-complete problem

• $A_{TM}^B = \{<M, x, m> : M \text{ is a TM that accepts } x \text{ within at most } m \text{ steps}\}$
• Proof that $A_{TM}^B$ is EXP-complete:
  – Part 2. For each language $A \in \text{EXP}$, need to give poly-time reduction from $A$ to $A_{TM}^B$.
  – for a given language $A \in \text{EXP}$, we know there is a TM $M_A$ that decides $A$ in time $g(n) \leq 2^{kn}$ for some $k$.
  – what should reduction $f(w)$ produce?

An EXP-complete problem

• A C-complete problem is a surrogate for the entire class C.
• For example: if you can find a poly-time algorithm for $A_{TM}^B$ then there is automatically a poly-time algorithm for every problem in EXP (i.e., EXP = P).

• Can you find a poly-time alg for $A_{TM}^B$?

An EXP-complete problem

• Can you find a poly-time alg for $A_{TM}^B$?
• NO! we showed that $P \subsetneq \text{EXP}$.
• $A_{TM}^B$ is not tractable (intractable).
Back to 3SAT

- Remember $3SAT \in EXP$
  
  $3SAT = \{\text{formulas in CNF with 3 literals per clause for which there exists a satisfying truth assignment}\}$
  
- It seems hard. Can we show it is intractable?
  - formally, can we show $3SAT$ is $EXP$-complete?

Back to 3SAT

- can we show $3SAT$ is $EXP$-complete?
- Don’t know how to. Believed unlikely.
- One reason: there is an important positive feature of $3SAT$ that doesn’t seem to hold for problems in $EXP$ (e.g. $ATM_B$):

> $3SAT$ is decidable in polynomial time by a nondeterministic TM

Nondeterministic TMs

- Recall: nondeterministic TM
- informally, TM with several possible next configurations at each step
- formally, A NTM is a 7-tuple
  
  $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$
  
  where:
  
  - everything is the same as a TM except the transition function:
  
  $\delta : Q \times \Gamma \rightarrow 2^{(Q \times \Gamma \times \{L, R\})}$

Nondeterministic TMs

visualize computation of a NTM $M$ as a tree

- nodes are configurations
- leaves are accept/reject configurations
- $M$ accepts if and only if there exists an accept leaf
- $M$ is a decider, so no paths go on forever
- running time is max. path length

The class NP

**Definition:** $\text{TIME}(t(n)) = \{L : \text{there exists a TM } M \text{ that decides } L \text{ in time } O(t(n))\}$

$P = \bigcup_{k \geq 1} \text{TIME}(n^k)$

**Definition:** $\text{NTIME}(t(n)) = \{L : \text{there exists a NTM } M \text{ that decides } L \text{ in time } O(t(n))\}$

$NP = \bigcup_{k \geq 1} \text{NTIME}(n^k)$

NP in relation to P and EXP

- $P \subseteq NP$ (poly-time TM is a poly-time NTM)
- $NP \subseteq EXP$
  
  - configuration tree of $n^k$-time NTM has $\leq b^k$ nodes
  - can traverse entire tree in $O(b^k)$ time

we do not know if either inclusion is proper
Poly-time verifiers

- **NP** = \{L : L decided by poly-time NTM\}

- Very useful alternate definition of NP:
  
  **Theorem**: language L is in NP if and only if it is expressible as:
  
  \[ L = \{ x \mid \exists y, \ |y| \leq |x|^k, (x, y) \in R \} \]
  
  where R is a language in P.

- poly-time TM \( M_R \) deciding R is a "verifier" or "certificate" efficiently verifiable

---

**Example: 3SAT** expressible as

\[ \text{3SAT} = \{ \phi : \phi \text{ is a 3-CNF formula for which } \exists \text{ assignment } A \text{ for which } (\phi, A) \in R \} \]

- satisfying assignment A is a "witness" of the satisfiability of \( \phi \) (it "certifies" satisfiability of \( \phi \))
- \( R \) is decidable in poly-time

---

**Proof**:

(\( \Rightarrow \)) given \( L \in \text{NP} \), describe \( L \) as:

\[ L = \{ x \mid \exists y, |y| \leq |x|^k, (x, y) \in R \} \]

- \( L \) is decided by NTM \( M \) running in time \( n^k \)
- define the language
  
  \[ R = \{ (x, y) : y \text{ is an accepting computation history of } M \text{ on input } x \} \]

- check: accepting history has length \( \leq |x|^k \)
- check: \( M \) accepts \( x \) iff \( \exists y, |y| \leq |x|^k, (x, y) \in R \)

---

**Cook-Levin Theorem**

- Gateway to proving lots of natural, important problems NP-complete is:

  **Theorem** (Cook, Levin): 3SAT is NP-complete.

- Recall: 3SAT = \{ \( \phi : \phi \text{ is a CNF formula with 3 literals per clause for which there exists a satisfying truth assignment} \) \}

---

**Proof outline**

- show CIRCUIT-SAT is NP-complete
  
  CIRCUIT-SAT = \{ C : C is a Boolean circuit for which there exists a satisfying truth assignment \}

- show 3SAT is NP-complete (reduce from CIRCUIT SAT)
Boolean Circuits

- Boolean circuit C
  - directed acyclic graph
  - nodes: AND (∧); OR (∨); NOT (¬); variables $x_i$

- C computes function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ in natural way
  - identify C with function f it computes

- size = # nodes

- $\lor \land x_1 x_2 \land \lor \neg x_3 \ldots x_n$

Boolean Circuits

- every function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ computable by a circuit of size at most $O(n2^n)$
  - AND of $n$ literals for each $x$ such that $f(x) = 1$
  - OR of up to $2^n$ such terms

CIRCUIT-SAT is NP-complete

**Theorem:** CIRCUIT-SAT is NP-complete

CIRCUIT-SAT = \{ $C$ : $C$ is a Boolean circuit for which there exists a satisfying truth assignment\}

Proof:
- Part 1: need to show CIRCUIT-SAT $\in$ NP.
  - can express CIRCUIT-SAT as:
    CIRCUIT-SAT = \{ $C$ : $C$ is a Boolean circuit for which $\exists x$ such that $(C, x) \in R$ \}

  $R = \{(C, x) : C$ is a Boolean circuit and $C(x) = 1\}$

- Part 2: for each language $A \in$ NP, need to give poly-time reduction from $A$ to CIRCUIT-SAT
  - for a given language $A \in$ NP, we know
    $A = \{x | \exists y, |y| \leq |x|^k, (x, y) \in R \}$
  - and there is a (deterministic) TM $M_R$ that decides $R$ in time $g(n) \leq n^c$ for some $c$.

CIRCUIT-SAT is NP-complete

- Tableau (configurations written in an array) for machine $M_R$ on input $w = (x, y)$:
  - height = time taken = $|w|^c$
  - width = space used $\leq |w|^c$

  \[
  \begin{array}{cccc}
  w_1/q_1 & w_2 & \ldots & w_n \\
  w_1 & w_2/q_1 & \ldots & w_n \\
  w_1 & a & \ldots & w_n \\
  \vdots & \vdots & \ddots & \vdots \\
  \_ & \_ & \ldots & \_ \\
  \end{array}
  \]

  \[
  \begin{array}{cccc}
  a/q_1 & b & a \\
  a & b/q_1 & a \\
  a & b & a \\
  \end{array}
  \]
CIRCUIT-SAT is NP-complete

- Can build Boolean circuit STEP
  - input (binary encoding of) 3 cells
  - output (binary encoding of) 1 cell
  - each output bit is some function of inputs
  - can build circuit for each
  - size is independent of size of tableau

Tableau for $M_R$ on input $w = (x, y)$

- $|w|^c$ copies of STEP compute row $i$ from $i-1$

Circuit $C_{M_R, w}$ has inputs $w_1, w_2, \ldots, w_n$

- hardwire input $x$
- leave $y$ as variables

Size = $O(|w|^{2c})$

Cook-Levin Theorem

- Gateway to proving lots of natural, important problems NP-complete is:

  **Theorem** (Cook, Levin): 3SAT is NP-complete.

  **Recall:** 3SAT = \{ $\phi$ : $\phi$ is a CNF formula with 3 literals per clause for which there exists a satisfying truth assignment \}
Cook-Levin Theorem

- Proof outline
  - show CIRCUIT-SAT is NP-complete
    CIRCUIT-SAT = \{ C : C is a Boolean circuit for which there exists a satisfying truth assignment \}

- show 3SAT is NP-complete (reduce from CIRCUIT SAT)

CIRCUIT-SAT is NP-complete

Theorem: CIRCUIT-SAT is NP-complete

CIRCUIT-SAT = \{ C : C is a Boolean circuit for which there exists a satisfying truth assignment \}

3SAT is NP-complete

Theorem: 3SAT is NP-complete

3SAT = \{ \varphi : \varphi is a 3-CNF formula for which there exists a satisfying truth assignment \}

Proof:
- Part 1: need to show 3-SAT \in NP
  • already done

- Part 2: need to show 3-SAT is NP-hard
  • we will give a poly-time reduction from CIRCUIT-SAT to 3-SAT

3SAT is NP-complete

- given a circuit C
  • variables \( x_1, x_2, \ldots, x_n \)
  • AND (\land), OR (\lor), NOT (\neg) gates \( g_1, g_2, \ldots, g_m \)

  reduction \( f(C) \) produces these clauses for \( \varphi \) on variables \( x_1, x_2, \ldots, x_n, g_1, g_2, \ldots, g_m \):

  \[
  \neg g_i \quad (g_i \lor z) \quad (\neg z \lor \neg g_i)
  \]

  \( z \iff \neg g_i \)

3SAT is NP-complete

- given a circuit C
  • variables \( x_1, x_2, \ldots, x_n \)
  • AND (\land), OR (\lor), NOT (\neg) gates \( g_1, g_2, \ldots, g_m \)

  reduction \( f(C) \) produces these clauses for \( \varphi \) on variables \( x_1, x_2, \ldots, x_n, g_1, g_2, \ldots, g_m \):

  \[
  \begin{align*}
  (z_1 \land z_2) & \iff \neg g_i \\
  \neg g_i \land (z_1 \lor z_2) & \\
  (\neg z_1 \lor z_2) \land (\neg g_i \lor z_1) & \\
  \end{align*}
  \]
3SAT is NP-complete

– finally, reduction f(C) produces single clause (g_m) where g_m is the output gate.
– f(C) computable in poly-time?
  • yes, simple transformation
– YES maps to YES?
  • if C(x) = 1, then assigning x-values to x-variables of φ and gate values of C when evaluating x to the g-variables of φ gives satisfying assignment.

3SAT is NP-complete

– NO maps to NO?
  • show that φ satisfiable implies C satisfiable
  • satisfying assignment to φ assigns values to x-variables and g-variables
  • output gate g_m must be assigned 1
  • every other gate must be assigned value it would take given values of its inputs.
  • the assignment to the x-variables must be a satisfying assignment for C.

Search vs. Decision

• Definition: given a graph G = (V, E), an independent set in G is a subset V' ⊆ V such that for all u,w ∈ V' (u,w) ∉ E
• A problem: given G, find the largest independent set
• This is called a search problem
  – searching for optimal object of some type
  – comes up frequently

Search vs. Decision

• We want to talk about languages (or decision problems)
• Most search problems have a natural, related decision problem by adding a bound “k”; for example:
  – search problem: given G, find the largest independent set
  – decision problem: given (G, k), is there an independent set of size at least k

Ind. Set is NP-complete

**Theorem:** the following language is NP-complete:

Ind. Set is NP-complete

• We are reducing from the language:

  3SAT = \{ φ : φ is a 3-CNF formula that has a satisfying assignment \}

  to the language:

  IS = \{ (G, k) : G has an IS of size ≥ k \}.
Ind. Set is NP-complete

The reduction \( f \): given
\[ \varphi = (x \lor y \lor \neg z) \land (\neg x \lor w \lor z) \land \ldots \land (\ldots) \]
we produce graph \( G_\varphi \):

- one triangle for each of \( m \) clauses
- edge between every pair of contradictory literals
- set \( k = m \)

\[ \begin{array}{c}
\text{x} \\
\text{y} \\
\text{\neg z} \\
\text{w} \\
\text{z} \\
\end{array} \]

- Is \( f \) poly-time computable?
- YES maps to YES?
  - 1 true literal per clause in satisfying assign. \( A \)
  - choose corresponding vertices (1 per triangle)
  - IS, since no contradictory literals in \( A \)

\[ f(\varphi) = (G, \# \text{ clauses}) \]

\[ \begin{array}{c}
\text{x} \\
\text{\neg z} \\
\text{w} \\
\text{z} \\
\end{array} \]

Ind. Set is NP-complete

\[ \varphi = (x \lor y \lor \neg z) \land (\neg x \lor w \lor z) \land \ldots \land (\ldots) \]

- NO maps to NO?
  - IS can have at most 1 vertex per triangle
  - IS of size \( \geq \# \) clauses must have exactly 1 per
  - since IS, no contradictory vertices
  - can produce satisfying assignment by setting
  - these literals to true

\[ \begin{array}{c}
\text{x} \\
\text{y} \\
\text{\neg z} \\
\text{w} \\
\text{z} \\
\end{array} \]