

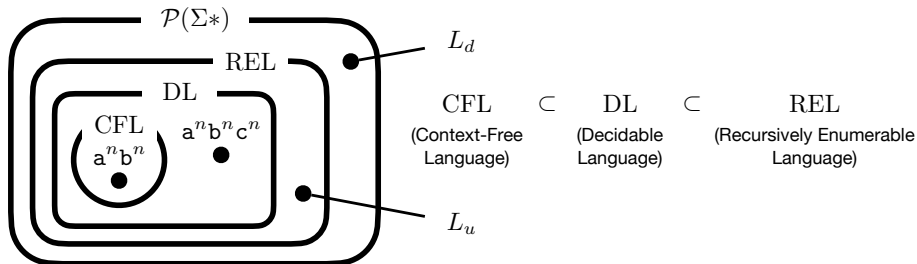
# Lecture 26 – P, NP, and NP-Complete Problems

## COSE215: Theory of Computation

Jihyeok Park



2026 Spring



## Definition (Decision Problem)

A **decision problem**  $\pi$  is a computational problem whose answer is either **yes** or **no** for a given input.

In this lecture, we will **classify** decision problems based on the **time complexity** of possible TMs (or NTMs) that solve the problems.

## 1. **P**

Time Complexity of TMs

**P** – Polynomial Time Complexity (Tractable Problems)

## 2. **NP**

Time Complexity of NTMs

**NP** – Nondeterministic Polynomial Time Complexity

**NP** – Verifier-based Definition

## 3. **NP-complete**

Polynomial Time Reduction ( $\leq_P$ )

**NP-complete** – Hardest Problems in **NP**

**<SAT>** – The First **NP-complete** Problem

Other **NP-complete** Problems

## 4. Major Unsolved Problem: **P = NP?**

## 1. **P**

Time Complexity of TMs

**P** – Polynomial Time Complexity (Tractable Problems)

## 2. **NP**

Time Complexity of NTMs

**NP** – Nondeterministic Polynomial Time Complexity

**NP** – Verifier-based Definition

## 3. **NP-complete**

Polynomial Time Reduction ( $\leq_P$ )

**NP-complete** – Hardest Problems in **NP**

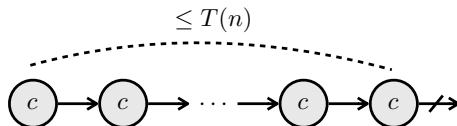
**<SAT>** – The First **NP-complete** Problem

Other **NP-complete** Problems

## 4. Major Unsolved Problem: **P = NP?**

## Definition (Time Complexity of TMs)

We say a **Turing machine (TM)**  $M$  has a **time complexity**  $T : \mathbb{N} \rightarrow \mathbb{N}$  if  $M$  halts on  $w$  in at most  $T(n)$  moves for all  $w \in \Sigma^*$  whose length is  $n$ .



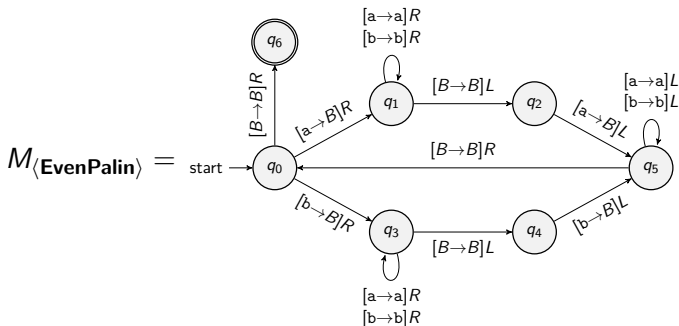
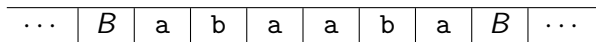
## Definition (DTIME)

A decision problem  $\pi$  is in **DTIME**( $T(n)$ ) if it is decidable by a TM  $M$  whose time complexity is  $T(n)$ .

We often use a **big O notation** to describe the time complexity of a TM:

$$f(n) = O(g(n)) \iff \exists k \in \mathbb{N}, n_0 \in \mathbb{N}. \forall n \geq n_0. f(n) \leq k \cdot g(n)$$

$\langle \mathbf{EvenPalin} \rangle$  – Is a word  $w \in \{a, b\}^*$  an even-length palindrome?



The decision problem  $\langle \mathbf{EvenPalin} \rangle$  is decidable by the above TM whose time complexity is  $T(n) = (n + 1)(n + 2)/2 = O(n^2)$ .

$$\langle \mathbf{EvenPalin} \rangle \in \mathbf{DTIME}(O(n^2))$$

## Definition (P – Polynomial Time Complexity)

A decision problem  $\pi$  is in **P** if it is decidable by a TM  $M$  whose time complexity is a **polynomial function** (i.e.,  $T(n) = O(n^k)$  for some  $k \geq 0$ ).

$$\mathbf{P} = \bigcup_{k \geq 0} \mathbf{DTIME}(O(n^k))$$

For example, the decision problem  $\langle \mathbf{EvenPalin} \rangle$  is in **P**.

$$\langle \mathbf{EvenPalin} \rangle \in \mathbf{DTIME}(O(n^2)) \subseteq \mathbf{P}$$

## Definition (Tractable Problems)

A problem  $\pi$  is called a **tractable problem** if it is a **P** problem.

## 1. **P**

Time Complexity of TMs

**P** – Polynomial Time Complexity (Tractable Problems)

## 2. **NP**

Time Complexity of NTMs

**NP** – Nondeterministic Polynomial Time Complexity

**NP** – Verifier-based Definition

## 3. **NP-complete**

Polynomial Time Reduction ( $\leq_P$ )

**NP-complete** – Hardest Problems in **NP**

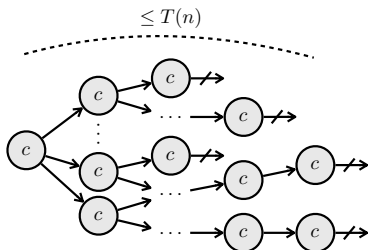
**<SAT>** – The First **NP-complete** Problem

Other **NP-complete** Problems

## 4. Major Unsolved Problem: **P = NP?**

## Definition (Time Complexity of NTMs)

We say a **nondeterministic Turing machine (NTM)**  $M$  has a **time complexity**  $T : \mathbb{N} \rightarrow \mathbb{N}$  if  $M$  halts on  $w$  in at most  $T(n)$  moves for all  $w \in \Sigma^*$  whose length is  $n$ .

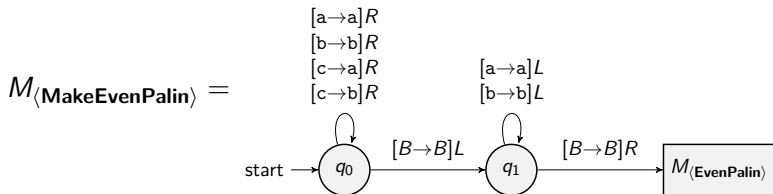
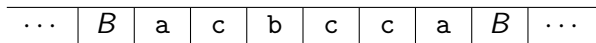


## Definition (NTIME)

A decision problem  $\pi$  is in **NTIME**( $T(n)$ ) if it is decidable by a NTM  $M$  whose time complexity is  $T(n)$ .

# Time Complexity of NTMs – Example

$\langle \mathbf{MakeEvenPalin} \rangle$  – Is a word  $w \in \{a, b, c\}^*$  convertible to an even-length palindrome by replacing all c's with a's or b's?



The decision problem  $\langle \mathbf{MakeEvenPalin} \rangle$  is decidable by the above NTM whose time complexity is  $T(n) = 2(n + 1) + O(n^2) = O(n^2)$ .

$$\langle \mathbf{MakeEvenPalin} \rangle \in \mathbf{NTIME}(O(n^2))$$

**Definition (NP – Nondeterministic Polynomial Time Complexity)**

A decision problem  $\pi$  is in **NP** if it is decidable by an NTM  $M$  whose time complexity is a **polynomial function** (i.e.,  $T(n) = O(n^k)$  for some  $k \geq 0$ ).

$$\mathbf{NP} = \bigcup_{k \geq 0} \mathbf{NTIME}(O(n^k))$$

For example, the decision problem  $\langle \mathbf{MakeEvenPalin} \rangle$  is in **NP**.

$$\langle \mathbf{MakeEvenPalin} \rangle \in \mathbf{NTIME}(O(n^2)) \subseteq \mathbf{NP}$$

## Definition (Search Problem)

A **search problem**  $\pi$  is a decision problem that asks for the existence of a **witness**  $x$  (i.e., a solution) in the search space  $S(w)$  for a given input  $w$ , satisfying another decision problem  $\pi'$  as a **verification problem**.

$$\forall w \in \Sigma^*. \pi(w) = \text{yes} \iff \exists x \in S(w). \pi'(w, x) = \text{yes}$$

For example,  $\langle \mathbf{MakeEvenPalin} \rangle$  is a **search problem** with  $\langle \mathbf{EvenPalin} \rangle$  as a **verification problem**:

$$\langle \mathbf{MakeEvenPalin} \rangle(w) = \text{yes} \iff \exists x \in S(w). \langle \mathbf{EvenPalin} \rangle(x) = \text{yes}$$

where the search space  $S(w)$  of an input  $w$  is defined as follows:

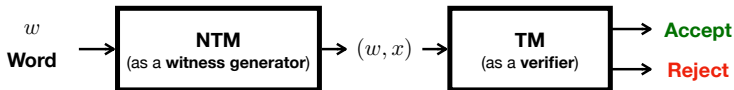
$$S(w) = \{x \mid x = (\text{a possible replacement of all } c\text{'s in } w \text{ with } a\text{'s or } b\text{'s})\}$$

e.g.,  $w = acbccca$   $S(w) = \left\{ \begin{array}{cccc} aabaaa, & aababa, & aabbaa, & aabbba, \\ abbaaa, & abbaba, & abbbaa, & abbbba \end{array} \right\}$

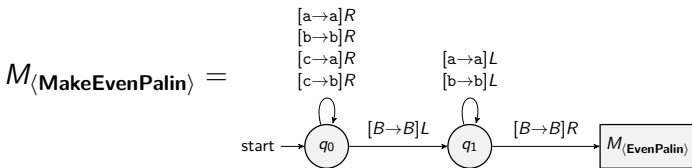
## Definition (NP – Verifier-based Definition)

A search problem  $\pi$  defined with a verification problem  $\pi'$  is in **NP** if there is a polynomial time TM  $M$  as a **verifier** for  $\pi$ :

$$\forall w \in \Sigma^*. \forall x \in S(w). \pi'(w, x) = \text{yes} \iff (w, x) \in L(M)$$



For example,  $\langle \text{MakeEvenPalin} \rangle$  is a search problem in **NP**:



$\langle \text{SAT} \rangle$  (Boolean **SAT**isfiability problem) – Is a given Boolean formula (consisting of Boolean variables,  $\wedge$ ,  $\vee$ , and  $\neg$ ) satisfiable?

For example, is the following Boolean formula satisfiable?

$$(x_1 \vee \neg x_2) \wedge (x_1 \vee x_2 \vee x_3) \wedge \neg x_1$$

Yes! For example,  $x_1 = \#f$ ,  $x_2 = \#f$ , and  $x_3 = \#t$  is a satisfying assignment.

Is it  $\langle \text{SAT} \rangle$  in **NP**? Yes!

We can construct a polynomial time TM as a **verifier** for  $\langle \text{SAT} \rangle$ , which takes 1) a **Boolean formula** and 2) an **assignment** of Boolean variables, and checks whether the assignment satisfies the formula.

In other words, we can construct a polynomial time NTM for  $\langle \text{SAT} \rangle$  by 1) **generating all assignments** of Boolean variables and 2) **verifying** whether the assignment satisfies the formula using the verifier.

## 1. P

Time Complexity of TMs

**P** – Polynomial Time Complexity (Tractable Problems)

## 2. NP

Time Complexity of NTMs

**NP** – Nondeterministic Polynomial Time Complexity

**NP** – Verifier-based Definition

## 3. NP-complete

Polynomial Time Reduction ( $\leq_P$ )

**NP-complete** – Hardest Problems in **NP**

**<SAT>** – The First **NP-complete** Problem

Other **NP-complete** Problems

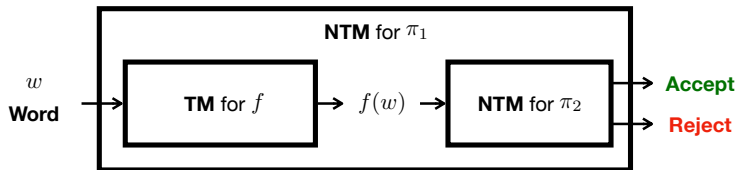
## 4. Major Unsolved Problem: **P = NP?**

## Definition (Polynomial Time Reduction ( $\leq_P$ ))

A decision problem  $\pi_1$  is **polynomial time reducible** to another decision problem  $\pi_2$  (denoted by  $\pi_1 \leq_P \pi_2$ ) if there exists a polynomial time computable function  $f : \Sigma^* \rightarrow \Sigma^*$  such that:

$$\forall w \in \Sigma^*. \pi_1(w) = \text{yes} \iff \pi_2(f(w)) = \text{yes}$$

We say that  $\pi_2$  is **harder** than  $\pi_1$  if  $\pi_1 \leq_P \pi_2$  because we can solve  $\pi_1$  in polynomial time if we can solve  $\pi_2$  in polynomial time.



If a decision problem  $\pi_2$  is in **NP** and  $\pi_1 \leq_P \pi_2$ , then  $\pi_1$  is in **NP**.

Consider the following two decision problems:

- **⟨MakeEvenPalin⟩** – Is a word  $w \in \{a, b, c\}^*$  convertible to an even-length palindrome by replacing all  $c$ 's with  $a$ 's or  $b$ 's?
- **⟨SAT⟩** – Is a given Boolean formula satisfiable?

We can show that **⟨MakeEvenPalin⟩**  $\leq_P$  **⟨SAT⟩** by the following polynomial time computable function  $f$ :

$$f(a_1 a_2 \cdots a_n) = \bigwedge_{i=1}^n ((x_i \wedge x_{n+1-i}) \vee (\neg x_i \wedge \neg x_{n+1-i})) \\ \wedge \bigwedge \{x_i \mid a_i = a\} \wedge \bigwedge \{\neg x_i \mid a_i = b\}$$

( $x_i = \#t$  means that  $a_i = a$  and  $x_i = \#f$  means that  $a_i = b$ )

For example,

$$f(acba) = ((x_1 \wedge x_4) \vee (\neg x_1 \wedge \neg x_4)) \wedge ((x_2 \wedge x_3) \vee (\neg x_2 \wedge \neg x_3)) \wedge x_1 \wedge \neg x_3 \wedge x_4$$

Thus, we can solve **⟨MakeEvenPalin⟩** using a machine for **⟨SAT⟩**, and **⟨SAT⟩** is harder problem than **⟨MakeEvenPalin⟩**.

## Definition (NP-hard – Harder Problems Than All NP)

A decision problem  $\pi$  is in **NP-hard** if  $\forall \pi' \in \mathbf{NP}, \pi' \leq_P \pi$ .

In other words,  $\pi$  is in **NP-hard** if  $\pi$  is **harder than all problems in NP**.

## Definition (NP-complete – Hardest Problems in NP)

A decision problem  $\pi$  is in **NP-complete** if

- 1  $\pi$  is in **NP**, and
- 2  $\pi$  is in **NP-hard** (i.e.,  $\forall \pi' \in \mathbf{NP}, \pi' \leq_P \pi$ ).

In other words,  $\pi$  is in **NP-complete** if  $\pi$  is the **hardest problem in NP**.

## Theorem (Cook–Levin theorem)

$\langle \text{SAT} \rangle$  is in **NP-complete**.

We need to show that

- ①  $\langle \text{SAT} \rangle$  is in **NP**, and
- ②  $\langle \text{SAT} \rangle$  is in **NP-hard**.

For ①, we already know that  $\langle \text{SAT} \rangle$  is in **NP**.

For ②, we need to show that  $\forall \pi \in \mathbf{NP}, \pi \leq_P \langle \text{SAT} \rangle$ .

The core idea is to simulate an NTM  $M$  for  $\pi$  using a Boolean formula  $\phi$  such that  $\phi$  is satisfiable if and only if  $M$  accepts  $w$ . But, we skip the details of the proof. Please refer to the link<sup>1</sup> for the details.

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Cook-Levin\\_theorem](https://en.wikipedia.org/wiki/Cook-Levin_theorem)

## Theorem (Lemma)

A decision problem  $\pi$  is in **NP-hard** if  $\langle \mathbf{SAT} \rangle \leq_P \pi$

This lemma is very useful to show that a decision problem  $\pi$  is in **NP-complete** by showing that 1)  $\pi$  is in **NP** and 2)  $\langle \mathbf{SAT} \rangle \leq_P \pi$ .

We can show that all of the following decision problems are in **NP-complete** by using this lemma:<sup>2</sup>

- $\langle \mathbf{SubsetSum} \rangle$  – Given a set of integers  $S$  and an integer  $t$ , is there a subset  $S' \subseteq S$  such that  $\sum S' = t$ ?
- $\langle \mathbf{Clique} \rangle$  – Given a graph  $G$  and an integer  $k$ , is there a clique of size  $k$  in  $G$ ?
- $\langle \mathbf{VertexCover} \rangle$  – Given a graph  $G$  and an integer  $k$ , is there a vertex cover of size  $k$  in  $G$ ?
- ...

<sup>2</sup>[https://en.wikipedia.org/wiki/List\\_of\\_NP-complete\\_problems](https://en.wikipedia.org/wiki/List_of_NP-complete_problems)

## 1. P

Time Complexity of TMs

**P** – Polynomial Time Complexity (Tractable Problems)

## 2. NP

Time Complexity of NTMs

**NP** – Nondeterministic Polynomial Time Complexity

**NP** – Verifier-based Definition

## 3. NP-complete

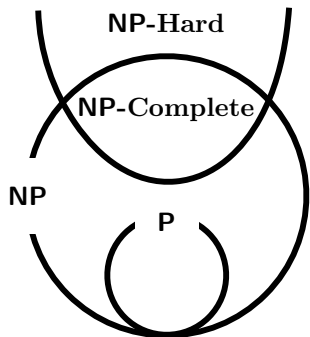
Polynomial Time Reduction ( $\leq_P$ )

**NP-complete** – Hardest Problems in **NP**

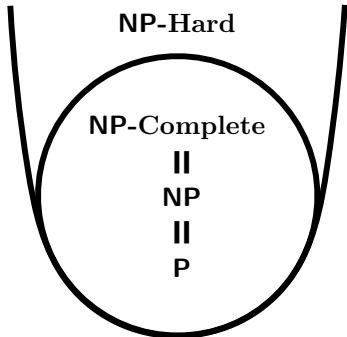
**<SAT>** – The First **NP-complete** Problem

Other **NP-complete** Problems

## 4. Major Unsolved Problem: **P = NP?**



$P \neq NP$



$P = NP$

*“If  $P = NP$ , then the world would be a profoundly different place than we usually assume it to be. There would be no special value in creative leaps, no fundamental gap between solving a problem and recognizing the solution once it’s found.”*

— Scott Aaronson, *UT Austin*

## 1. **P**

Time Complexity of TMs

**P** – Polynomial Time Complexity (Tractable Problems)

## 2. **NP**

Time Complexity of NTMs

**NP** – Nondeterministic Polynomial Time Complexity

**NP** – Verifier-based Definition

## 3. **NP-complete**

Polynomial Time Reduction ( $\leq_P$ )

**NP-complete** – Hardest Problems in **NP**

**<SAT>** – The First **NP-complete** Problem

Other **NP-complete** Problems

## 4. Major Unsolved Problem: **P = NP?**

- The final exam will be given in class.
- **Date:** 13:30-14:45 (1 hour 15 minutes), June 17 (Wed.).
- **Location:**
  - **IT-Education Hall B102:** Students with Student ID 2025XXXX
  - **IT-Education Hall 611:** All other students
- **Coverage:** Lectures 14 – 26
- **Format:** 7–9 questions with closed book and closed notes
  - Yes/No questions about concepts in the theory of computation.
  - Construction of automata or grammars for given languages.
  - Proofs of given statements related to automata or grammars.
  - etc.
- Note that there is **no class** on **June 22 (Mon.)**.
- Please refer to the **previous exams** in the course website:

<https://plrg.korea.ac.kr/courses/cose215/>

- Course Review

Jihyeok Park  
jihyeok\_park@korea.ac.kr  
<https://plrg.korea.ac.kr>