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

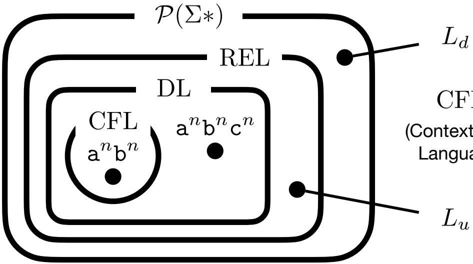
COSE215: Theory of Computation

Jihyeok Park

PLRG

2024 Spring

Recall

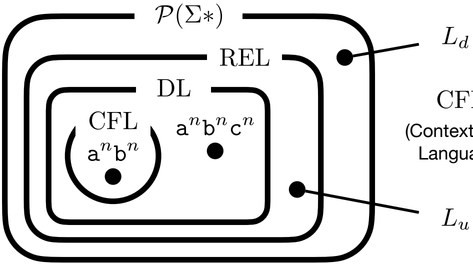


CFL (Context-Free Language) $\subset$ DL (Decidable Language) $\subset$ REL (Recursively Enumerable Language)

Definition (Decision Problem)

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

Recall



CFL (Context-Free Language) $\subset$ DL (Decidable Language) $\subset$ REL (Recursively Enumerable Language)

Definition (Decision Problem)

A **decision problem** π 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.

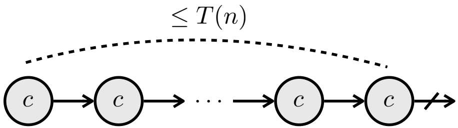
Contents

Contents

Time Complexity of TMs

Definition (Time Complexity of TMs)

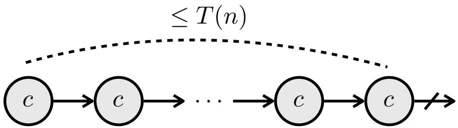
We say a **Turing machine (TM)** M has a **time complexity** $T : \mathbb{N} \to \mathbb{N}$ if M halts on w in at most $T(n)$ moves for all $w \in \Sigma^*$ whose length is n.



Time Complexity of TMs

Definition (Time Complexity of TMs)

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



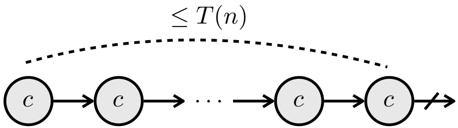
Definition (**DTIME**)

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

Time Complexity of TMs

Definition (Time Complexity of TMs)

We say a **Turing machine (TM)** M has a **time complexity** $T : \mathbb{N} \to \mathbb{N}$ 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 π 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:

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

Time Complexity of TMs – Example

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

Time Complexity of TMs – Example

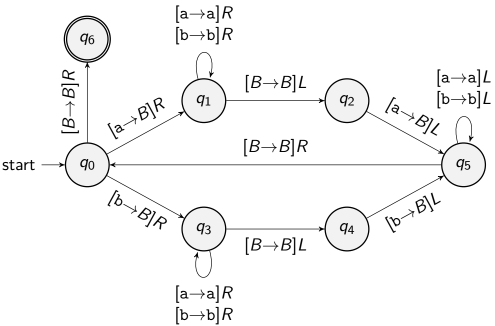
⟨**EvenPalin**⟩ – Is a word $w \in \{\mathtt{a}, \mathtt{b}\}^*$ an even-length palindrome?

$\cdots$	B	a	b	a	a	b	a	B	$\cdots$

Time Complexity of TMs – Example

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

···	B	a	b	a	a	b	a	B	···

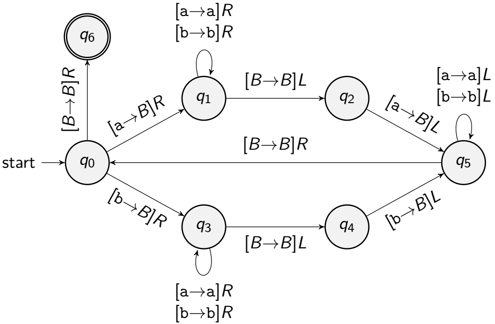
$M_{\langle \textbf{EvenPalin} \rangle} =$ 

Time Complexity of TMs – Example

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

$\cdots$	B	a	b	a	a	b	a	B	$\cdots$

$M_{\langle \textbf{EvenPalin} \rangle} =$



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

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

P – Polynomial Time Complexity

Definition (**P** – Polynomial Time Complexity)

A decision problem π 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 $k \geq 0$).

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

P – Polynomial Time Complexity

Definition (P – Polynomial Time Complexity)

A decision problem π 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 $\langle \textbf{EvenPalin} \rangle$ is in **P**.

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

P – Polynomial Time Complexity

PLRG

Definition (**P** – Polynomial Time Complexity)

A decision problem π 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)

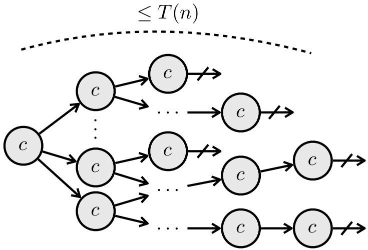
A problem π is called a **tractable problem** if it is a **P** problem.

Contents

Time Complexity of NTMs

Definition (Time Complexity of NTMs)

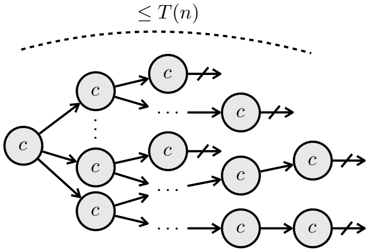
We say a **nondeterministic Turing machine (NTM)** M has a **time complexity** $T : \mathbb{N} \to \mathbb{N}$ if M halts on w in at most $T(n)$ moves for all $w \in \Sigma^*$ whose length is n.



Time Complexity of NTMs

Definition (Time Complexity of NTMs)

We say a **nondeterministic Turing machine (NTM)** M has a **time complexity** $T : \mathbb{N} \to \mathbb{N}$ 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 π is in **NTIME**$(T(n))$ if it is decidable by a NTM M whose time complexity is $T(n)$.

Time Complexity of NTMs – Example

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

Time Complexity of NTMs – Example

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

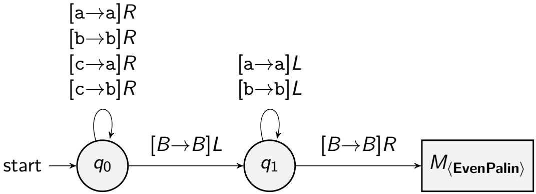
$\cdots$	B	a	c	b	c	c	a	B	$\cdots$

Time Complexity of NTMs – Example

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

⋯	B	a	c	b	c	c	a	B	⋯

$M_{\langle \textbf{MakeEvenPalin} \rangle} =$

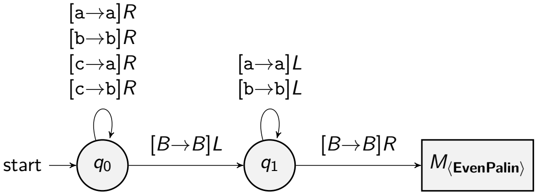


Time Complexity of NTMs – Example

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

$\cdots$	B	a	c	b	c	c	a	B	$\cdots$

$M_{\langle \textbf{MakeEvenPalin} \rangle} =$



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

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

Definition (NP – Nondeterministic Polynomial Time Complexity)

A decision problem π 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 $\langle$**MakeEvenPalin**$\rangle$ is in **NP**.

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

Search Problem

Definition (Search Problem)

A **search problem** π 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 the another decision problem π' as a **verification problem**.

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

Search Problem

Definition (Search Problem)

A **search problem** π 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 the another decision problem π' 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, $\langle$**MakeEvenPalin**$\rangle$ is a **search problem** with $\langle$**EvenPalin**$\rangle$ as a **verification problem**:

Search Problem

Definition (Search Problem)

A **search problem** π 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 the another decision problem π' 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, $\langle\textbf{MakeEvenPalin}\rangle$ is a **search problem** with $\langle\textbf{EvenPalin}\rangle$ as a **verification problem**:

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

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

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

Search Problem

Definition (Search Problem)

A **search problem** π 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 the another decision problem π' 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, ⟨**MakeEvenPalin**⟩ is a **search problem** with ⟨**EvenPalin**⟩ as a **verification problem**:

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

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

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

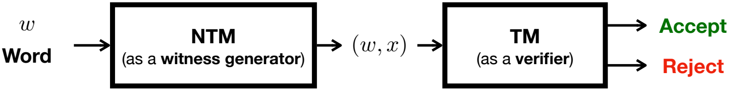
$$\text{e.g.,} \quad w = \texttt{acbcca} \quad S(w) = \left\{ \begin{array}{llll} \texttt{aabaaa}, & \texttt{aababa}, & \texttt{aabbaa}, & \texttt{aabbba}, \\ \texttt{abbaaa}, & \texttt{abbaba}, & \texttt{abbbaa}, & \texttt{abbbba} \end{array} \right\}$$

NP – Nondeterministic Polynomial Time Complexity

Definition (NP – Verifier-based Definition)

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

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



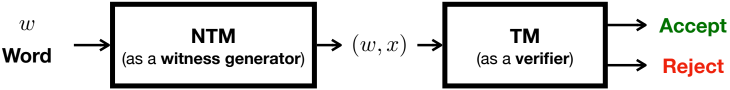
NP – Nondeterministic Polynomial Time Complexity

PLRG

Definition (**NP** – Verifier-based Definition)

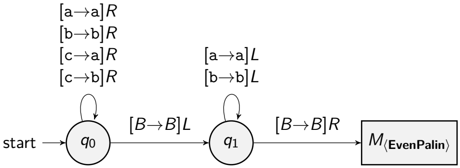
A search problem π defined with a verification problem π' is in **NP** if there is a polynomial time TM M as a **verifier** for π:

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



For example, ⟨**MakeEvenPalin**⟩ is a search problem in **NP**:

$M_{\langle \textbf{MakeEvenPalin} \rangle} =$



NP – Example: $\langle$**SAT**$\rangle$

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

NP – Example: ⟨**SAT**⟩

⟨**SAT**⟩ (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?

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

NP – Example: $\langle$SAT$\rangle$

$\langle$**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, $x_1 = \texttt{\#f}$, $x_2 = \texttt{\#f}$, and $x_3 = \texttt{\#t}$ is a satisfying assignment.

NP – Example: $\langle$**SAT**$\rangle$

$\langle$**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, $x_1 = \texttt{\#f}$, $x_2 = \texttt{\#f}$, and $x_3 = \texttt{\#t}$ is a satisfying assignment.

Is it $\langle$**SAT**$\rangle$ in **NP**?

NP – Example: ⟨**SAT**⟩

⟨**SAT**⟩ (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, $x_1 = \texttt{\#f}$, $x_2 = \texttt{\#f}$, and $x_3 = \texttt{\#t}$ is a satisfying assignment.

Is it ⟨**SAT**⟩ in **NP**? Yes!

We can construct a polynomial time TM as a **verifier** for ⟨**SAT**⟩, which takes 1) a **Boolean formula** and 1) an **assignment** of Boolean variables, and checks whether the assignment satisfies the formula.

NP – Example: ⟨**SAT**⟩

⟨**SAT**⟩ (Boolean **SAT**isfiability problem) – Is a given Boolean formula (consisting of Boolean variables, $\land$, $\lor$, and $\neg$) satisfiable?

For example, is the following Boolean formula satisfiable?

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

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

Is it ⟨**SAT**⟩ in **NP**? Yes!

We can construct a polynomial time TM as a **verifier** for ⟨**SAT**⟩, which takes 1) a **Boolean formula** and 1) an **assignment** of Boolean variables, and checks whether the assignment satisfies the formula.

In other words, we can construct a polynomial time NTM for ⟨**SAT**⟩ by 1) **generating all assignments** of Boolean variables and 2) **verifying** whether the assignment satisfies the formula using the verifier.

Contents

Polynomial Time Reduction ($\leq_P$)

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

A decision problem π_1 is **polynomial time reducible** to another decision problem π_2 (denoted by $\pi_1 \leq_P \pi_2$) if there exists a polynomial time computable function $f : \Sigma^* \to \Sigma^*$ such that:

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

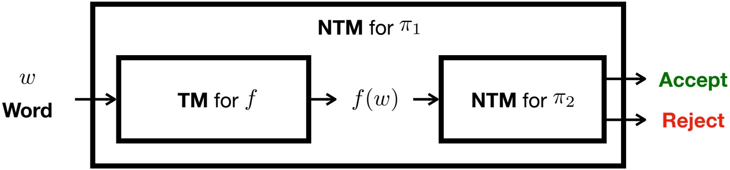
Polynomial Time Reduction ($\leq_P$)

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

A decision problem π_1 is **polynomial time reducible** to another decision problem π_2 (denoted by $\pi_1 \leq_P \pi_2$) if there exists a polynomial time computable function $f : \Sigma^* \to \Sigma^*$ such that:

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

We say that π_2 is **harder** than π_1 if $\pi_1 \leq_P \pi_2$ because we can solve π_1 in polynomial time if we can solve π_2 in polynomial time.



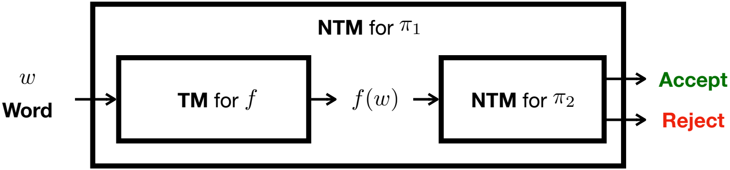
Polynomial Time Reduction ($\leq_P$)

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

A decision problem π_1 is **polynomial time reducible** to another decision problem π_2 (denoted by $\pi_1 \leq_P \pi_2$) if there exists a polynomial time computable function $f : \Sigma^* \to \Sigma^*$ such that:

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

We say that π_2 is **harder** than π_1 if $\pi_1 \leq_P \pi_2$ because we can solve π_1 in polynomial time if we can solve π_2 in polynomial time.



If a decision problem π_2 is in **NP** and $\pi_1 \leq_P \pi_2$, then π_1 is in **NP**.

Polynomial Time Reduction ($\leq_P$) – Example

Consider the following two decision problems:

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

Polynomial Time Reduction ($\leq_P$) – Example

PLRG

Consider the following two decision problems:

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

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

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

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

Polynomial Time Reduction ($\leq_P$) – Example

Consider the following two decision problems:

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

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

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

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

For example,

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

Polynomial Time Reduction ($\leq_P$) – Example

PLRG

Consider the following two decision problems:

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

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

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

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

For example,

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

Thus, we can solve $\langle$**MakeEvenPalin**$\rangle$ using a machine for $\langle$**SAT**$\rangle$, and $\langle$**SAT**$\rangle$ is harder problem than $\langle$**MakeEvenPalin**$\rangle$.

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

PLRG

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

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

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

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

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

In other word, π is in **NP-hard** if π is **harder than all problems in NP.**

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

PLRG

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

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

In other word, π is in **NP-hard** if π is **harder than all problems in NP.**

Definition (NP-complete – Hardest Problems in NP)

A decision problem π is in **NP-complete** if

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

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

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

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

In other word, π is in **NP-hard** if π is **harder than all problems in NP.**

Definition (NP-complete – Hardest Problems in NP)

A decision problem π is in **NP-complete** if

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

In other word, π is in **NP-complete** if π is the **hardest problem in NP**.

$\langle$**SAT**$\rangle$ – The First **NP-complete** Problem

Theorem (Cook–Levin theorem)

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

[1] https://en.wikipedia.org/wiki/Cook-Levin_theorem

$\langle$**SAT**$\rangle$ – The First **NP-complete** Problem

Theorem (Cook–Levin theorem)

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

We need to show that

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

[1] https://en.wikipedia.org/wiki/Cook-Levin_theorem

⟨**SAT**⟩ – The First **NP-complete** Problem

Theorem (Cook–Levin theorem)

⟨**SAT**⟩ *is in* **NP-complete**.

We need to show that

1. ⟨**SAT**⟩ is in **NP**, and
2. ⟨**SAT**⟩ is in **NP-hard**.

For ①, we already know that ⟨**SAT**⟩ is in **NP**.

[1] https://en.wikipedia.org/wiki/Cook-Levin_theorem

$\langle$**SAT**$\rangle$ – The First **NP-complete** Problem

Theorem (Cook–Levin theorem)

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

We need to show that

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

For (1), we already know that $\langle$**SAT**$\rangle$ is in **NP**.

For (2), we need to show that $\forall \pi \in$ **NP**, $\pi \leq_P \langle$**SAT**$\rangle$.

[1] https://en.wikipedia.org/wiki/Cook-Levin_theorem

⟨SAT⟩ – The First NP-complete Problem

Theorem (Cook–Levin theorem)

⟨**SAT**⟩ *is in* **NP-complete**.

We need to show that

1. ⟨**SAT**⟩ is in **NP**, and
2. ⟨**SAT**⟩ is in **NP-hard**.

For ①, we already know that ⟨**SAT**⟩ is in **NP**.

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

The core idea is to simulate an NTM M for π using a Boolean formula ϕ such that ϕ is satisfiable if and only if M accepts w. But, we skip the details of the proof. Please refer to the link[1] for the details.

[1] https://en.wikipedia.org/wiki/Cook-Levin_theorem

Other **NP-complete** Problems

Theorem (Lemma)

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

[2] https://en.wikipedia.org/wiki/List_of_NP-complete_problems

Other **NP-complete** Problems

PLRG

Theorem (Lemma)

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

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

[2] https://en.wikipedia.org/wiki/List_of_NP-complete_problems

Other **NP-complete** Problems

Theorem (Lemma)

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

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

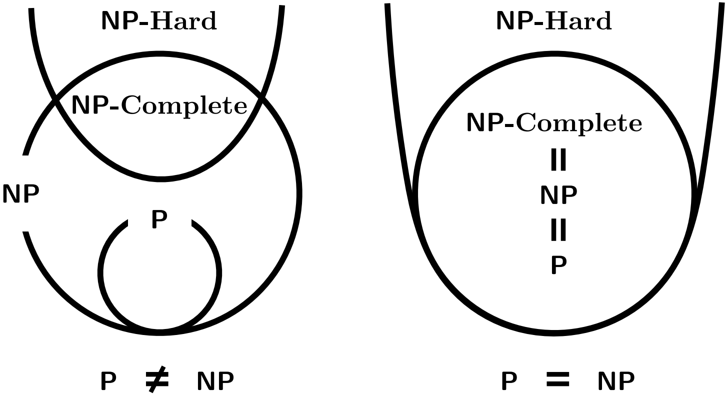
We can show that all of the following decision problems are in **NP-complete** by using this lemma:[2]

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

[2] https://en.wikipedia.org/wiki/List_of_NP-complete_problems

Contents

Major Unsolved Problem: **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*

Summary

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**

   $\langle$**SAT**$\rangle$ – The First **NP-complete** Problem

   Other **NP-complete** Problems

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

Next Lecture

PLRG

- Course Review

Jihyeok Park
jihyeok_park@korea.ac.kr
https://plrg.korea.ac.kr