CS310 Automata Theory – 2016-2017

Nutan Limaye

Indian Institute of Technology, Bombay
nutan@cse.iitb.ac.in

Lecture 34: Effective computation
April 09, 2017

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class $\mathrm{TIME}(t(n))$

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class $\mathrm{TIME}(t(n))$ if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class $\mathrm{TIME}(t(n))$ if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class $\mathrm{TIME}(t(n))$ if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

- M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x.

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class $\mathrm{TIME}(t(n))$ if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

- M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x.
- if $x \notin L$ then M rejects x.

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class $\mathrm{TIME}(t(n))$ if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

- M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x.
- if $x \notin L$ then M rejects x.

$$P = \bigcup_k \mathrm{TIME}(n^k)$$

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class $\mathrm{TIME}(t(n))$ if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

- M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x.
- if $x \notin L$ then M rejects x.

$$P = \bigcup_k \mathrm{TIME}(n^k)$$

$$EXP = \bigcup_k \mathrm{TIME}(2^{n^k})$$

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class $\mathrm{NTIME}(t(n))$

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class $\mathrm{NTIME}(t(n))$ if there exists a non-deterministic Turing machine M such that $\forall x \in \Sigma^*$,

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class $\mathrm{NTIME}(t(n))$ if there exists a non-deterministic Turing machine M such that $\forall x \in \Sigma^*$,

> each run of M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class NTIME$(t(n))$ if there exists a non-deterministic Turing machine M such that $\forall x \in \Sigma^*$,

each run of M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.

if $x \in L$ then

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class NTIME$(t(n))$ if there exists a non-deterministic Turing machine M such that $\forall x \in \Sigma^*$,

- each run of M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x on at least one run.

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class NTIME$(t(n))$ if there exists a non-deterministic Turing machine M such that $\forall x \in \Sigma^*$,

- each run of M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x on at least one run.
- if $x \notin L$ then

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class NTIME$(t(n))$ if there exists a non-deterministic Turing machine M such that $\forall x \in \Sigma^*$,

- each run of M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x on at least one run.
- if $x \notin L$ then M rejects x on all runs.

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class NTIME($t(n)$) if there exists a non-deterministic Turing machine M such that $\forall x \in \Sigma^*$,

- each run of M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x on at least one run.
- if $x \notin L$ then M rejects x on all runs.

$$NP = \bigcup_k \mathrm{NTIME}(n^k)$$

Time complexity and complexity classes

Let $t : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class $\mathrm{NTIME}(t(n))$ if there exists a non-deterministic Turing machine M such that $\forall x \in \Sigma^*$,

- each run of M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x on at least one run.
- if $x \notin L$ then M rejects x on all runs.

$$NP = \bigcup_k \mathrm{NTIME}(n^k)$$

$$NEXP = \bigcup_k \mathrm{NTIME}(2^{n^k})$$

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$.

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time $O((t(n))^2)$.

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time $O((t(n))^2)$.

Proof idea:

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time $O((t(n))^2)$.

Proof idea:

East step of multitape machine can be executed on a single tape machine in time $O\left(t(n)\right)$.

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time $O((t(n))^2)$.

Proof idea:

> East step of multitape machine can be executed on a single tape machine in time $O(t(n))$.

Lemma

Let $t(n) > n$. Let L be a language decided by a non-deterministic TM in time $t(n)$.

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time $O((t(n))^2)$.

Proof idea:

> East step of multitape machine can be executed on a single tape machine in time $O\left(t(n)\right)$.

Lemma

Let $t(n) > n$. Let L be a language decided by a non-deterministic TM in time $t(n)$. Then there is a deterministic TM that decides L in time

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time $O((t(n))^2)$.

Proof idea:

> East step of multitape machine can be executed on a single tape machine in time $O(t(n))$.

Lemma

Let $t(n) > n$. Let L be a language decided by a non-deterministic TM in time $t(n)$. Then there is a deterministic TM that decides L in time $2^{O(t(n))}$.

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time $O((t(n))^2)$.

Proof idea:

> East step of multitape machine can be executed on a single tape machine in time $O(t(n))$.

Lemma

Let $t(n) > n$. Let L be a language decided by a non-deterministic TM in time $t(n)$. Then there is a deterministic TM that decides L in time $2^{O(t(n))}$.

Proof idea:

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time $O((t(n))^2)$.

Proof idea:

> East step of multitape machine can be executed on a single tape machine in time $O\left(t(n)\right)$.

Lemma

Let $t(n) > n$. Let L be a language decided by a non-deterministic TM in time $t(n)$. Then there is a deterministic TM that decides L in time $2^{O(t(n))}$.

Proof idea: Analyze the construction of Problem 3 from Tutorial 9.

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

P $\subseteq$ NP by definition.

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

P $\subseteq$ NP by definition.

P $\subseteq$ EXP again by definition.

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

P $\subseteq$ NP by definition.

P $\subseteq$ EXP again by definition.

Similarly, NP $\subseteq$ NEXP by definition.

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

- $P \subseteq NP$ by definition.
- $P \subseteq EXP$ again by definition.
- Similarly, $NP \subseteq NEXP$ by definition.
- Finally, $NP \subseteq EXP$ due to the previous lemma.

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

P ⊆ NP by definition.

P ⊆ EXP again by definition.

Similarly, NP ⊆ NEXP by definition.

Finally, NP ⊆ EXP due to the previous lemma.

$$\begin{tikzcd}
\mathrm{P} \arrow[r] & \mathrm{NP} \\
\mathrm{EXP} & \mathrm{NEXP}
\end{tikzcd}$$

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

P $\subseteq$ NP by definition.

P $\subseteq$ EXP again by definition.

Similarly, NP $\subseteq$ NEXP by definition.

Finally, NP $\subseteq$ EXP due to the previous lemma.

$$\begin{tikzcd}
\mathrm{P} \arrow[r] \arrow[d] & \mathrm{NP} \\
\mathrm{EXP} & \mathrm{NEXP}
\end{tikzcd}$$

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

- P $\subseteq$ NP by definition.
- P $\subseteq$ EXP again by definition.
- Similarly, NP $\subseteq$ NEXP by definition.
- Finally, NP $\subseteq$ EXP due to the previous lemma.

$$\begin{tikzcd}
\mathsf{P} \arrow[r] \arrow[d] & \mathsf{NP} \\
\mathsf{EXP} \arrow[r] & \mathsf{NEXP}
\end{tikzcd}$$

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

- P $\subseteq$ NP by definition.
- P $\subseteq$ EXP again by definition.
- Similarly, NP $\subseteq$ NEXP by definition.
- Finally, NP $\subseteq$ EXP due to the previous lemma.

$$\begin{tikzcd}
\mathsf{P} \arrow[r] \arrow[d] & \mathsf{NP} \arrow[d] \\
\mathsf{EXP} \arrow[r] & \mathsf{NEXP}
\end{tikzcd}$$

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

P ⊆ NP by definition.

P ⊆ EXP again by definition.

Similarly, NP ⊆ NEXP by definition.

Finally, NP ⊆ EXP due to the previous lemma.

$$\begin{tikzcd}
\mathrm{P} \arrow[r] \arrow[d] & \mathrm{NP} \arrow[d] \arrow[dl, red] \\
\mathrm{EXP} \arrow[r] & \mathrm{NEXP}
\end{tikzcd}$$

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

- P $\subseteq$ NP by definition.
- P $\subseteq$ EXP again by definition.
- Similarly, NP $\subseteq$ NEXP by definition.
- Finally, NP $\subseteq$ EXP due to the previous lemma.

$$
\begin{tikzcd}
\mathrm{P} \arrow[r] \arrow[d] & \mathrm{NP} \arrow[d] \arrow[dl, red] \\
\mathrm{EXP} \arrow[r] & \mathrm{NEXP}
\end{tikzcd}
$$

P vs. NP

P vs. NP

P the class of languages where membership can be decided quickly.

P vs. NP

P	the class of languages where membership can be decided quickly.
NP	the class of languages where membership can be verified quickly.

Time heirarchy theorem

How do we separate NP from P?

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	pumping lemma or CFLs

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	pumping lemma or CFLs
not recognizable	

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	pumping lemma or CFLs
not recognizable	diagonalization

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	pumping lemma or CFLs
not recognizable	diagonalization
not decidable	

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	pumping lemma or CFLs
not recognizable	diagonalization
not decidable	Rice's theorem

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	pumping lemma or CFLs
not recognizable	diagonalization
not decidable	Rice's theorem or diagonalization and reductions

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	pumping lemma or CFLs
not recognizable	diagonalization
not decidable	Rice's theorem or diagonalization and reductions
not in P	

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	pumping lemma or CFLs
not recognizable	diagonalization
not decidable	Rice's theorem or diagonalization and reductions
not in P	???

Finer structure inside P

Definition

A function $t : \mathbb{N} \rightarrow \mathbb{N}$ is said to be time constructible

Finer structure inside P

Definition

A function $t : \mathbb{N} \rightarrow \mathbb{N}$ is said to be time constructible if the there exists a TM that on input 1^n, it outputs $t(n)$ in time $O(t(n))$.

Finer structure inside P

Definition

A function $t : \mathbb{N} \to \mathbb{N}$ is said to be time constructible if the there exists a TM that on input 1^n, it outputs $t(n)$ in time $O(t(n))$.

Examples

Finer structure inside P

Definition

A function $t : \mathbb{N} \to \mathbb{N}$ is said to be time constructible if the there exists a TM that on input 1^n, it outputs $t(n)$ in time $O(t(n))$.

Examples

n^2

Finer structure inside P

Definition

A function $t : \mathbb{N} \rightarrow \mathbb{N}$ is said to be time constructible if the there exists a TM that on input 1^n, it outputs $t(n)$ in time $O(t(n))$.

Examples

n^2, $n \log n$.

Finer structure inside P

Definition

A function $t : \mathbb{N} \to \mathbb{N}$ is said to be time constructible if the there exists a TM that on input 1^n, it outputs $t(n)$ in time $O(t(n))$.

Examples

$$n^2, \; n \log n.$$

Theorem

Let $t : \mathbb{N} \to \mathbb{N}$ be a time constructible function. There exists a language L such that $L \in TIME(t(n)^2)$, but $L \notin TIME(o(t(n)))$.

Finer structure inside P

Definition

A function $t : \mathbb{N} \to \mathbb{N}$ is said to be time constructible if the there exists a TM that on input 1^n, it outputs $t(n)$ in time $O(t(n))$.

Examples

n^2, $n \log n$.

Theorem

Let $t : \mathbb{N} \to \mathbb{N}$ be a time constructible function. There exists a language L such that $L \in TIME(t(n)^2)$, but $L \notin TIME(o(t(n)))$.

As a result of the theorem we have

Finer structure inside P

Definition

A function $t : \mathbb{N} \to \mathbb{N}$ is said to be time constructible if the there exists a TM that on input 1^n, it outputs $t(n)$ in time $O(t(n))$.

Examples

n^2, $n \log n$.

Theorem

Let $t : \mathbb{N} \to \mathbb{N}$ *be a time constructible function. There exists a language* L *such that* $L \in TIME(t(n)^2)$, *but* $L \notin TIME(o(t(n)))$.

As a result of the theorem we have

For any $i \geq 2$ and $1 > \epsilon > 0$, $TIME(n^i) \subsetneq TIME(n^{2i+\epsilon})$.

Finer structure inside P

Definition

A function $t : \mathbb{N} \to \mathbb{N}$ is said to be time constructible if the there exists a TM that on input 1^n, it outputs $t(n)$ in time $O(t(n))$.

Examples

n^2, $n \log n$.

Theorem

Let $t : \mathbb{N} \to \mathbb{N}$ be a time constructible function. There exists a language L such that $L \in TIME(t(n)^2)$, but $L \notin TIME(o(t(n)))$.

As a result of the theorem we have

For any $i \geq 2$ and $1 > \epsilon > 0$, $TIME(n^i) \subsetneq TIME(n^{2i+\epsilon})$.

Polynomial time reductions and NP-hardness

Definition

A function $f : \Sigma^* \to \Sigma^*$ is polynomial time computable if there is a polynomial time Turing machine TM, say M, such that on any input $w \in \Sigma^*$, M stops with only $f(w)$ on its tape.

Polynomial time reductions and NP-hardness

Definition

A function $f : \Sigma^* \to \Sigma^*$ is polynomial time computable if there is a polynomial time Turing machine TM, say M, such that on any input $w \in \Sigma^*$, M stops with only $f(w)$ on its tape.

Polynomial time reductions and NP-hardness

Definition

A language L_1 is said to be polynomial time reducible to another language L_2

Polynomial time reductions and NP-hardness

Definition

A language L_1 is said to be polynomial time reducible to another language L_2, denoted as $L_1 \leq_m L_2$

Polynomial time reductions and NP-hardness

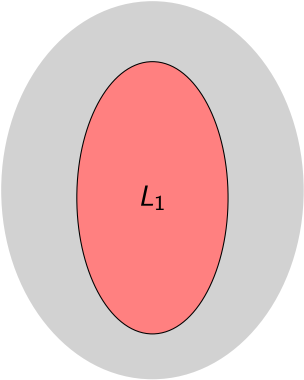
Definition

A language L_1 is said to be polynomial time reducible to another language L_2, denoted as $L_1 \leq_m L_2$, if there exists a polynomial time computable function f such that for all $w \in \Sigma^*$, $w \in L_1 \Leftrightarrow f(w) \in L_2$.

Polynomial time reductions and NP-hardness

Definition

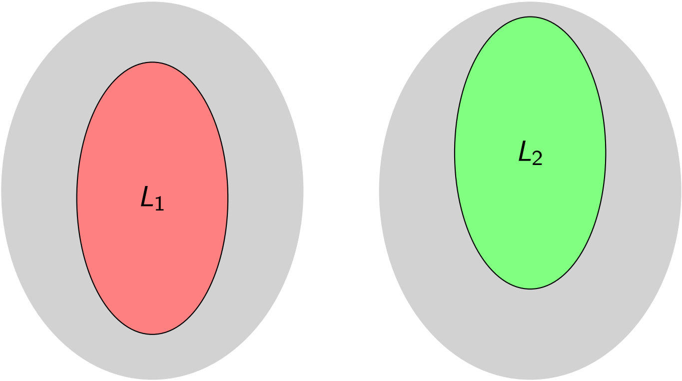
A language L_1 is said to be polynomial time reducible to another language L_2, denoted as $L_1 \leq_m L_2$, if there exists a polynomial time computable function f such that for all $w \in \Sigma^*$, $w \in L_1 \Leftrightarrow f(w) \in L_2$.



Polynomial time reductions and NP-hardness

Definition

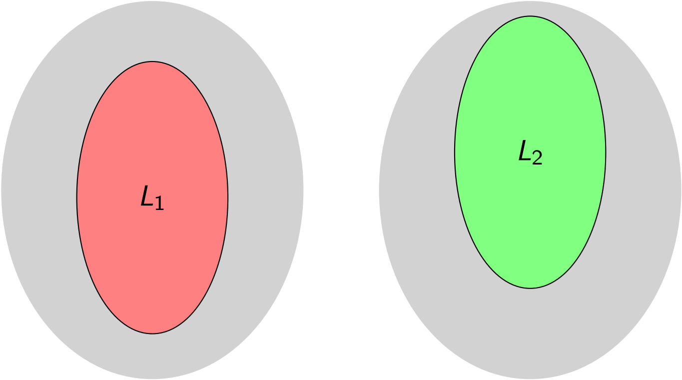
A language L_1 is said to be polynomial time reducible to another language L_2, denoted as $L_1 \leq_m L_2$, if there exists a polynomial time computable function f such that for all $w \in \Sigma^*$, $w \in L_1 \Leftrightarrow f(w) \in L_2$.



Polynomial time reductions and NP-hardness

Definition

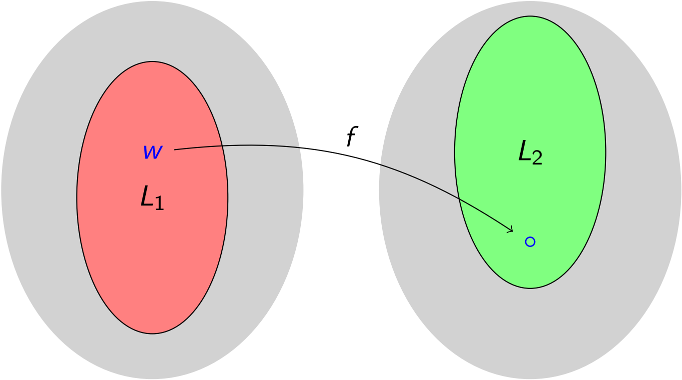
A language L_1 is said to be polynomial time reducible to another language L_2, denoted as $L_1 \leq_m L_2$, if there exists a polynomial time computable function f such that for all $w \in \Sigma^*$, $w \in L_1 \Leftrightarrow f(w) \in L_2$.



Polynomial time reductions and NP-hardness

Definition

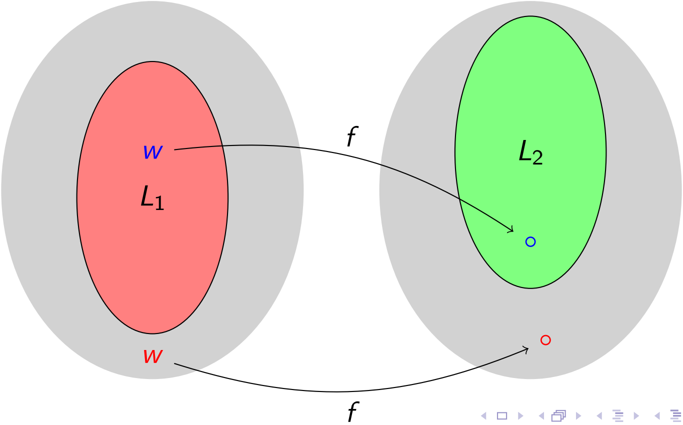
A language L_1 is said to be polynomial time reducible to another language L_2, denoted as $L_1 \leq_m L_2$, if there exists a polynomial time computable function f such that for all $w \in \Sigma^*$, $w \in L_1 \Leftrightarrow f(w) \in L_2$.



Polynomial time reductions and NP-hardness

Definition

A language L_1 is said to be polynomial time reducible to another language L_2, denoted as $L_1 \leq_m L_2$, if there exists a polynomial time computable function f such that for all $w \in \Sigma^*$, $w \in L_1 \Leftrightarrow f(w) \in L_2$.



Polynomial time reductions and NP-hardness

Definition

A language L is said to be NP-hard if for every language $L' \in \text{NP}$, there is a polynomial time reduction such that $L' \leq_m L$.

Polynomial time reductions and NP-hardness

Definition

A language L is said to be NP-hard if for every language $L' \in$ NP, there is a polynomial time reduction such that $L' \leq_m L$.

Definition

A language L is said to be NP-complete if the following two conditions hold:

Polynomial time reductions and NP-hardness

Definition

A language L is said to be NP-hard if for every language $L' \in$ NP, there is a polynomial time reduction such that $L' \leq_m L$.

Definition

A language L is said to be NP-complete if the following two conditions hold:

L is in NP.

Polynomial time reductions and NP-hardness

Definition

A language L is said to be NP-hard if for every language $L' \in$ NP, there is a polynomial time reduction such that $L' \leq_m L$.

Definition

A language L is said to be NP-complete if the following two conditions hold:

- L is in NP.
- L is NP-hard.

Theorem ([Cook-Levin, 1970])

SAT is NP-complete.

Space bounded Turing Machines

The Turing Machine model with space bounds

Space bounded Turing Machines

The Turing Machine model with space bounds

- The input tape is assumed to be read-only.

Space bounded Turing Machines

The Turing Machine model with space bounds

- The input tape is assumed to be read-only.

- The space required to write down the input is not counted towards the space of the machine.

Space bounded Turing Machines

The Turing Machine model with space bounds

- The input tape is assumed to be read-only.
- The space required to write down the input is not counted towards the space of the machine.
- The output tape assumed to be write-only.

Space bounded Turing Machines

The Turing Machine model with space bounds

The input tape is assumed to be read-only.

The space required to write down the input is not counted towards the space of the machine.

The output tape assumed to be write-only.

The space required to write down the output is not counted towards the space of the machine.

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class SPACE($s(n)$)

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class $\mathrm{SPACE}(s(n))$ if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class SPACE($s(n)$) if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

M halts on x using at most space $O(s(|x|))$

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class $\mathrm{SPACE}(s(n))$ if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

M halts on x using at most space $O(s(|x|))$,

where $|x|$ indicates the length of x.

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class SPACE$(s(n))$ if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

M halts on x using at most space $O(s(|x|))$,

where $|x|$ indicates the length of x.

if $x \in L$ then M accepts x.

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class SPACE$(s(n))$ if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

M halts on x using at most space $O(s(|x|))$,

where $|x|$ indicates the length of x.

if $x \in L$ then M accepts x.

if $x \notin L$ then M rejects x.

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class SPACE($s(n)$) if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

M halts on x using at most space $O(s(|x|))$,

where $|x|$ indicates the length of x.

if $x \in L$ then M accepts x.

if $x \notin L$ then M rejects x.

$$L = \mathrm{SPACE}(\log n)$$

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class SPACE($s(n)$) if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

- M halts on x using at most space $O(s(|x|))$,
- where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x.
- if $x \notin L$ then M rejects x.

$$L = \mathrm{SPACE}(\log n)$$

$$PSPACE = \bigcup_k \mathrm{SPACE}(n^k)$$

Examples

Min $= \{(w_1, w_2, \ldots, w_n, i) \mid w_i$ is the minimum among $w_1 \ldots w_n\}$.

Examples

Min $= \{(w_1, w_2, \ldots, w_n, i) \mid w_i$ is the minimum among $w_1 \ldots w_n\}$.

Deg $= \{(G = (V, E), d, i) \mid v_i$ has degree $d\}$.

Examples

Min = $\{(w_1, w_2, \ldots, w_n, i) \mid w_i$ is the minimum among $w_1 \ldots w_n\}$.

Deg = $\{(G = (V, E), d, i) \mid v_i$ has degree $d\}$.

ADD = $\{(u, v, i) \mid i$th bit of $u + v$ is $1\}$.

Examples

Min $= \{(w_1, w_2, \ldots, w_n, i) \mid w_i$ is the minimum among $w_1 \ldots w_n\}$.

Deg $= \{(G = (V, E), d, i) \mid v_i$ has degree $d\}$.

ADD $= \{(u, v, i) \mid i$th bit of $u + v$ is $1\}$.

Verify-SAT

$= \{(\phi, a) \mid a = a_1, a_2, \ldots, a_n$ is an assignment satisfying $\phi\}$.

Examples

Min $= \{(w_1, w_2, \ldots, w_n, i) \mid w_i$ is the minimum among $w_1 \ldots w_n\}$.

Deg $= \{(G = (V, E), d, i) \mid v_i$ has degree $d\}$.

ADD $= \{(u, v, i) \mid i$th bit of $u + v$ is $1\}$.

Verify-SAT

$= \{(\phi, a) \mid a = a_1, a_2, \ldots, a_n$ is an assignment satisfying $\phi\}$.