CS310 Automata Theory – 2016-2017

Nutan Limaye

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

Lecture 33: Effective computation
April 06, 2017

Module IV: Effective computation

Turing machines with resource constraints.

Module IV: Effective computation

Turing machines with resource constraints.

Resources for computation.

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes
  - The number of times an input bit can be read

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes
  - The number of times an input bit can be read
  - The amount of energy used

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes
  - The number of times an input bit can be read
  - The amount of energy used
  - ⋮

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes
  - The number of times an input bit can be read
  - The amount of energy used
  - ⋮
- Why bound resorces?
  - Viewing TM as algorithms.

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes
  - The number of times an input bit can be read
  - The amount of energy used
  - ⋮
- Why bound resorces?
  - Viewing TM as algorithms.
  - TM to help in computation of important problems.

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes
  - The number of times an input bit can be read
  - The amount of energy used
  - ⋮
- Why bound resorces?
  - Viewing TM as algorithms.
  - TM to help in computation of important problems.
  - Finer study of decidable languages.

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes
  - The number of times an input bit can be read
  - The amount of energy used
  - ⋮

- Why bound resorces?
  - Viewing TM as algorithms.
  - TM to help in computation of important problems.
  - Finer study of decidable languages.

- How should we bound the resources?

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes
  - The number of times an input bit can be read
  - The amount of energy used
  - ⋮

- Why bound resorces?
  - Viewing TM as algorithms.
  - TM to help in computation of important problems.
  - Finer study of decidable languages.

- How should we bound the resources?
  - Many different ways exist. ...

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 $\text{TIME}(t(n))$

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 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 $\text{NTIME}(t(n))$ 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 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 $\mathrm{NTIME}(t(n))$ 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

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 $\mathrm{NTIME}(t(n))$ 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.

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

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 $O(t(n))$.

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

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 $O(t(n))$.

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: DIY

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?

- $\mathrm{P} \subseteq \mathrm{NP}$ by definition.
- $\mathrm{P} \subseteq \mathrm{EXP}$ again by definition.
- Similarly, $\mathrm{NP} \subseteq \mathrm{NEXP}$ by definition.
- Finally, $\mathrm{NP} \subseteq \mathrm{EXP}$ 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.

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

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.

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

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.

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

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.

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

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.

Examples

SAT $= \{\phi \mid \phi$ is satisfiable$\}$.

Examples

SAT $= \{\phi \mid \phi$ is satisfiable$\}$. in NP (and not known to be in P)

Examples

SAT $= \{\phi \mid \phi$ is satisfiable$\}$. in NP (and not known to be in P)

Reach $= \{(G, s, t) \mid t$ is reachable from s in $G\}$.

Examples

SAT $= \{\phi \mid \phi$ is satisfiable$\}$. in NP (and not known to be in P)

Reach $= \{(G, s, t) \mid t$ is reachable from s in $G\}$. in P

Examples

SAT $= \{\phi \mid \phi$ is satisfiable$\}$. in NP (and not known to be in P)

Reach $= \{(G, s, t) \mid t$ is reachable from s in $G\}$. in P

3-SAT $= \{\phi \mid \phi$ is a 3-CNF and satisfiable$\}$.

Examples

SAT $= \{\phi \mid \phi$ is satisfiable$\}$. in NP (and not known to be in P)

Reach $= \{(G, s, t) \mid t$ is reachable from s in $G\}$. in P

3-SAT $= \{\phi \mid \phi$ is a 3-CNF and satisfiable$\}$. in NP (and not known to be in P)

Examples

SAT $= \{\phi \mid \phi$ is satisfiable$\}$. in NP (and not known to be in P)

Reach $= \{(G, s, t) \mid t$ is reachable from s in $G\}$. in P

3-SAT $= \{\phi \mid \phi$ is a 3-CNF and satisfiable$\}$. in NP (and not known to be in P)

2-SAT $= \{\phi \mid \phi$ is a 2-CNF and satisfiable$\}$.

Examples

SAT $= \{\phi \mid \phi$ is satisfiable$\}$. in NP (and not known to be in P)

Reach $= \{(G, s, t) \mid t$ is reachable from s in $G\}$. in P

3-SAT $= \{\phi \mid \phi$ is a 3-CNF and satisfiable$\}$. in NP (and not known to be in P)

2-SAT $= \{\phi \mid \phi$ is a 2-CNF and satisfiable$\}$. in P

Examples

SAT $= \{\phi \mid \phi$ is satisfiable$\}$. in NP (and not known to be in P)

Reach $= \{(G, s, t) \mid t$ is reachable from s in $G\}$. in P

3-SAT $= \{\phi \mid \phi$ is a 3-CNF and satisfiable$\}$. in NP (and not known to be in P)

2-SAT $= \{\phi \mid \phi$ is a 2-CNF and satisfiable$\}$. in P

Factoring $= \{(k, n) \mid n$ has a factor $\leq k\}$.

Examples

SAT $= \{\phi \mid \phi$ is satisfiable$\}$. in NP (and not known to be in P)

Reach $= \{(G, s, t) \mid t$ is reachable from s in $G\}$. in P

3-SAT $= \{\phi \mid \phi$ is a 3-CNF and satisfiable$\}$. in NP (and not known to be in P)

2-SAT $= \{\phi \mid \phi$ is a 2-CNF and satisfiable$\}$. in P

Factoring $= \{(k, n) \mid n$ has a factor $\leq k\}$. Google it!

Examples

SAT $= \{\phi \mid \phi$ is satisfiable$\}$. in NP (and not known to be in P)

Reach $= \{(G, s, t) \mid t$ is reachable from s in $G\}$. in P

3-SAT $= \{\phi \mid \phi$ is a 3-CNF and satisfiable$\}$. in NP (and not known to be in P)

2-SAT $= \{\phi \mid \phi$ is a 2-CNF and satisfiable$\}$. in P

Factoring $= \{(k, n) \mid n$ has a factor $\leq k\}$. Google it!

Clique $= \{(G, k) \mid G$ has a clique of size $\geq k\}$.

Examples

SAT $= \{\phi \mid \phi$ is satisfiable$\}$. in NP (and not known to be in P)

Reach $= \{(G, s, t) \mid t$ is reachable from s in $G\}$. in P

3-SAT $= \{\phi \mid \phi$ is a 3-CNF and satisfiable$\}$. in NP (and not known to be in P)

2-SAT $= \{\phi \mid \phi$ is a 2-CNF and satisfiable$\}$. in P

Factoring $= \{(k, n) \mid n$ has a factor $\leq k\}$. Google it!

Clique $= \{(G, k) \mid G$ has a clique of size $\geq k\}$. in NP (and not known to be in P)

Examples

SAT $= \{\phi \mid \phi$ is satisfiable$\}$. in NP (and not known to be in P)

Reach $= \{(G, s, t) \mid t$ is reachable from s in $G\}$. in P

3-SAT $= \{\phi \mid \phi$ is a 3-CNF and satisfiable$\}$. in NP (and not known to be in P)

2-SAT $= \{\phi \mid \phi$ is a 2-CNF and satisfiable$\}$. in P

Factoring $= \{(k, n) \mid n$ has a factor $\leq k\}$. Google it!

Clique $= \{(G, k) \mid G$ has a clique of size $\geq k\}$. in NP (and not known to be in P)

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 $t : \mathbb{N} \to \mathbb{N}$ 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 $t : \mathbb{N} \rightarrow \mathbb{N}$ 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

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

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

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 $L \notin TIME(o(t(n)))$.