COL 352 Introduction to Automata and Theory of Computation

Nikhil Balaji

Bharti 420
Indian Institute of Technology, Delhi
nbalaji@cse.iitd.ac.in

April 20, 2023

Lecture 32: Computational Complexity Theory (Part 1)

Effective computation

Turing machines with resource constraints.

Effective computation

Turing machines with resource constraints.

Resources for computation.

Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time

Effective computation

Turing machines with resource constraints.

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

Effective computation

Turing machines with resource constraints.

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

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

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

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

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
  - $\vdots$

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 resources?
  - Viewing TM as algorithms.

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 resources?
  - Viewing TM as algorithms.
  - TM to help in computation of important problems.

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

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

- How should we bound the resources?

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 resources?
  - 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. ...

Measuring Time

Measuring Time

$$L = \{0^k 1^k \mid k \in \mathbb{N}\}$$

Measuring Time

$$L = \{0^k 1^k \mid k \in \mathbb{N}\}$$

On input string w:

1. Scan across the tape and reject if a 0 is found to the right of a 1.
2. Repeat if both 0s and 1s remain on the tape:
3. Scan across the tape, crossing off a single 0 and a single 1.
4. If 0s still remain after all the 1s have been crossed off, or if 1s still remain after all the 0s have been crossed off, reject. Otherwise, if neither 0s nor 1s remain on the tape, accept.

Measuring Time

$$L = \{0^k 1^k \mid k \in \mathbb{N}\}$$

On input string w:

1. Scan across the tape and reject if a 0 is found to the right of a 1.
2. Repeat if both 0s and 1s remain on the tape:
3. Scan across the tape, crossing off a single 0 and a single 1.
4. If 0s still remain after all the 1s have been crossed off, or if 1s still remain after all the 0s have been crossed off, reject. Otherwise, if neither 0s nor 1s remain on the tape, accept.

$$O(n) + \frac{n}{2} O(n) = O(n^2)$$

Time complexity

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

Time complexity

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

Definition

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

Time complexity

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 $\forall x \in \Sigma^*$,

Time complexity

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

Definition

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

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.

Time complexity

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

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.

Revisiting Example

$$L = \{0^k 1^k \mid k \in \mathbb{N}\}$$

Revisiting Example

$$L = \{0^k 1^k \mid k \in \mathbb{N}\}$$

On input string w:

1. Scan across the tape and reject if a 0 is found to the right of a 1.
2. Repeat as long as some 0s and some 1s remain on the tape:

Revisiting Example

$$L = \{0^k 1^k \mid k \in \mathbb{N}\}$$

On input string w:

1. Scan across the tape and reject if a 0 is found to the right of a 1.
2. Repeat as long as some 0s and some 1s remain on the tape:
3. Scan across the tape, checking whether the total number of 0s and 1s remaining is even or odd. If it is odd, reject .
4. Scan again across the tape, crossing off every other 0 starting with the first 0, and then crossing off every other 1 starting with the first 1.
5. If no 0s and no 1s remain on the tape, accept. Otherwise, reject.

Revisiting Example

$$L = \{0^k 1^k \mid k \in \mathbb{N}\}$$

On input string w:

1. Scan across the tape and reject if a 0 is found to the right of a 1.
2. Repeat as long as some 0s and some 1s remain on the tape:
3. Scan across the tape, checking whether the total number of 0s and 1s remaining is even or odd. If it is odd, reject .
4. Scan again across the tape, crossing off every other 0 starting with the first 0, and then crossing off every other 1 starting with the first 1.
5. If no 0s and no 1s remain on the tape, accept. Otherwise, reject.

$$L \in \mathsf{TIME}(n \log n)$$

Re-Revisiting Example

$$L = \{0^k 1^k \mid k \in \mathbb{N}\}$$

Re-Revisiting Example

$$L = \{0^k 1^k \mid k \in \mathbb{N}\}$$

On input string w:

1. Scan across the tape and reject if a 0 is found to the right of a 1.

Re-Revisiting Example

$$L = \{0^k 1^k \mid k \in \mathbb{N}\}$$

On input string w:

1. Scan across the tape and reject if a 0 is found to the right of a 1.
2. Scan across the 0s on tape 1 until the first 1. At the same time, copy the 0s onto tape 2.
3. Scan across the 1s on tape 1 until the end of the input. For each 1 read on tape 1, cross off a 0 on tape 2. If all 0s are crossed off before all the 1s are read, reject.
4. If all the 0s have now been crossed off, accept . If any 0s remain, reject.

Re-Revisiting Example

$$L = \{0^k 1^k \mid k \in \mathbb{N}\}$$

On input string w:

1. Scan across the tape and reject if a 0 is found to the right of a 1.
2. Scan across the 0s on tape 1 until the first 1. At the same time, copy the 0s onto tape 2.
3. Scan across the 1s on tape 1 until the end of the input. For each 1 read on tape 1, cross off a 0 on tape 2. If all 0s are crossed off before all the 1s are read, reject.
4. If all the 0s have now been crossed off, accept . If any 0s remain, reject.

$$L \in \mathsf{TIME}(n)$$

Exercise: There is no single-tape TM solving L in $O(n)$ time.

Relationships between models

Lemma

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

Relationships between models

Lemma

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

Relationships between models

Lemma

Let $t(n) \geq n$. 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 $t(n) \geq n$. 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 $t(n) \geq n$. 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 $t(n) \geq n$. 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 $t(n) \geq n$. 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 $t(n) \geq n$. 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 $t(n) \geq n$. 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 $t(n) \geq n$. 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: DFS or BFS.

Time complexity and complexity classes

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

Definition

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

Time complexity and complexity classes

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

Definition

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

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

Time complexity and complexity classes

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

Definition

A language $L \subseteq \Sigma^*$ is said to be in class 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.

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

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

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 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 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 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 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.

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

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

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

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

Relationships between complexity classes

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

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

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

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.

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

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.

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

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.

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

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.

Definition

A verifier for a language A is an algorithm V, where

$$A = \{w \mid V \text{ accepts } (w, c) \text{ for some string } c\}$$

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.

Definition

A verifier for a language A is an algorithm V, where

$$A = \{w \mid V \text{ accepts } (w, c) \text{ for some string } c\}$$

NP is the class of languages that have polynomial time verifiers. c is the "certificate" or "witness" or "proof" that $w \in A$.