Computational Complexity Theory

Fall 2025

Time complexity and Hierarchy theorems
September 2 and 4, 2025

Lijie Chen

University of California, Berkeley
lijiechen@berkeley.edu

Some Logistics

- **Office Hours:** 2:00 - 3:00 PM, SODA 627, Tuesday

Some Logistics

- **Office Hours:** 2:00 - 3:00 PM, SODA 627, Tuesday
- **Additional Office Hours?:** 11:30 AM - 12:30 PM, SODA 627, Thursday

Some Logistics

- **Office Hours:** 2:00 - 3:00 PM, SODA 627, Tuesday
- **Additional Office Hours?:** 11:30 AM - 12:30 PM, SODA 627, Thursday
- **First HW out: Sept 4**

Some Logistics

- **Office Hours:** 2:00 - 3:00 PM, SODA 627, Tuesday
- **Additional Office Hours?:** 11:30 AM - 12:30 PM, SODA 627, Thursday
- **First HW out: Sept 4**
- **Suggested projects list out: Sept 4**

Some Logistics

- **Office Hours:** 2:00 - 3:00 PM, SODA 627, Tuesday
- **Additional Office Hours?:** 11:30 AM - 12:30 PM, SODA 627, Thursday
- **First HW out: Sept 4**
- **Suggested projects list out: Sept 4**
- **Course website:** https://chen-lijie.github.io/cs278-complexity.html

Recap:Definition of Multi-tape Turing machine

- A multi-tape Turing machine M consists of:

Recap:Definition of Multi-tape Turing machine

- A multi-tape Turing machine M consists of:
  - A finite set of states Q

Recap:Definition of Multi-tape Turing machine

- A multi-tape Turing machine M consists of:
  - A finite set of states Q
  - An input alphabet Σ and tape alphabet Γ (where $\Sigma \subseteq \Gamma$)

Recap:Definition of Multi-tape Turing machine

- A multi-tape Turing machine M consists of:
  - A finite set of states Q
  - An input alphabet Σ and tape alphabet Γ (where $\Sigma \subseteq \Gamma$)
  - k tapes, each infinite in both directions

Recap:Definition of Multi-tape Turing machine

- A multi-tape Turing machine M consists of:
  - A finite set of states Q
  - An input alphabet Σ and tape alphabet Γ (where $\Sigma \subseteq \Gamma$)
  - k tapes, each infinite in both directions
  - k read/write heads, one for each tape

Recap:Definition of Multi-tape Turing machine

- A multi-tape Turing machine M consists of:
  - A finite set of states Q
  - An input alphabet Σ and tape alphabet Γ (where $\Sigma \subseteq \Gamma$)
  - k tapes, each infinite in both directions
  - k read/write heads, one for each tape
  - A transition function $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$

Recap:Definition of Multi-tape Turing machine

- A multi-tape Turing machine M consists of:
  - A finite set of states Q
  - An input alphabet Σ and tape alphabet Γ (where $\Sigma \subseteq \Gamma$)
  - k tapes, each infinite in both directions
  - k read/write heads, one for each tape
  - A transition function $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$
  - A start state $q_0 \in Q$

Recap:Definition of Multi-tape Turing machine

- A multi-tape Turing machine M consists of:
  - A finite set of states Q
  - An input alphabet Σ and tape alphabet Γ (where $\Sigma \subseteq \Gamma$)
  - k tapes, each infinite in both directions
  - k read/write heads, one for each tape
  - A transition function $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$
  - A start state $q_0 \in Q$
  - Accept and reject states $q_{accept}, q_{reject} \in Q$

Recap:Definition of Multi-tape Turing machine

- A multi-tape Turing machine M consists of:
  - A finite set of states Q
  - An input alphabet Σ and tape alphabet Γ (where $\Sigma \subseteq \Gamma$)
  - k tapes, each infinite in both directions
  - k read/write heads, one for each tape
  - A transition function $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$
  - A start state $q_0 \in Q$
  - Accept and reject states $q_{accept}, q_{reject} \in Q$
- Initially: input x is on tape 1, all other tapes are blank, all heads start at position 0

Recap:Definition of Multi-tape Turing machine

- A multi-tape Turing machine M consists of:
  - A finite set of states Q
  - An input alphabet Σ and tape alphabet Γ (where $\Sigma \subseteq \Gamma$)
  - k tapes, each infinite in both directions
  - k read/write heads, one for each tape
  - A transition function $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$
  - A start state $q_0 \in Q$
  - Accept and reject states $q_{accept}, q_{reject} \in Q$
- Initially: input x is on tape 1, all other tapes are blank, all heads start at position 0
- At each step: reads symbols under all heads, writes new symbols, moves heads, changes state

Recap:Definition of Multi-tape Turing machine

- A multi-tape Turing machine M consists of:
    - A finite set of states Q
    - An input alphabet Σ and tape alphabet Γ (where $\Sigma \subseteq \Gamma$)
    - k tapes, each infinite in both directions
    - k read/write heads, one for each tape
    - A transition function $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$
    - A start state $q_0 \in Q$
    - Accept and reject states $q_{accept}, q_{reject} \in Q$
- Initially: input x is on tape 1, all other tapes are blank, all heads start at position 0
- At each step: reads symbols under all heads, writes new symbols, moves heads, changes state
- Computation ends when machine reaches q_{accept} or q_{reject}

Why study multi-tape TM?

- From *computability* perspective, single-tape TM and multi-tape TM are equivalent.

Why study multi-tape TM?

- From *computability* perspective, single-tape TM and multi-tape TM are equivalent.
- But from *complexity* perspective, multi-tape TM is more powerful than single-tape TM.

Why study multi-tape TM?

- From *computability* perspective, single-tape TM and multi-tape TM are equivalent.
- But from *complexity* perspective, multi-tape TM is more powerful than single-tape TM.
    - There can be a quadratic separation between single-tape TM running time and multi-tape TM running time.

Why study multi-tape TM?

- From *computability* perspective, single-tape TM and multi-tape TM are equivalent.
- But from *complexity* perspective, multi-tape TM is more powerful than single-tape TM.
  - There can be a quadratic separation between single-tape TM running time and multi-tape TM running time.
  - **Example**: Checking if a string is a palindrome is in $O(n)$ time on multi-tape TM, but in $O(n^2)$ time on single-tape TM.

Why study multi-tape TM?

- From *computability* perspective, single-tape TM and multi-tape TM are equivalent.
- But from *complexity* perspective, multi-tape TM is more powerful than single-tape TM.
  - There can be a quadratic separation between single-tape TM running time and multi-tape TM running time.
  - **Example**: Checking if a string is a palindrome is in $O(n)$ time on multi-tape TM, but in $O(n^2)$ time on single-tape TM.
  - You can **sort** and evaluate a **circuit** in time $O(n \log n)$ on multi-tape TM, so they are indeed quite powerful!

Why study multi-tape TM?

- From *computability* perspective, single-tape TM and multi-tape TM are equivalent.
- But from *complexity* perspective, multi-tape TM is more powerful than single-tape TM.
  - There can be a quadratic separation between single-tape TM running time and multi-tape TM running time.
  - **Example**: Checking if a string is a palindrome is in $O(n)$ time on multi-tape TM, but in $O(n^2)$ time on single-tape TM.
  - You can **sort** and evaluate a **circuit** in time $O(n \log n)$ on multi-tape TM, so they are indeed quite powerful!
  - Later in this course, we will study Ryan Williams' breakthrough results on T-time in $\sqrt{T}$-space, which holds for multi-tape TM.

Tangent: Ohter variants of Turing machines

- In multi-tape Turing machines, moving tape heads is still slow

Tangent: Ohter variants of Turing machines

- In multi-tape Turing machines, moving tape heads is still slow
- Random access machines (RAMs) are a model of computation that allow for random access to memory. This is the model underlying modern CPU architectures.

Tangent: Ohter variants of Turing machines

- In multi-tape Turing machines, moving tape heads is still slow
- Random access machines (RAMs) are a model of computation that allow for random access to memory. This is the model underlying modern CPU architectures.
- RAMs are (believed to be) more powerful than multi-tape Turing machines.

Tangent: Ohter variants of Turing machines

- In multi-tape Turing machines, moving tape heads is still slow
- Random access machines (RAMs) are a model of computation that allow for random access to memory. This is the model underlying modern CPU architectures.
- RAMs are (believed to be) more powerful than multi-tape Turing machines.
- d-dimensional multi-/single-tape Turing machines have d-dimensional memory, and can move the tape heads to any neighboring cell in one step.

Tangent: Ohter variants of Turing machines

- In multi-tape Turing machines, moving tape heads is still slow
- Random access machines (RAMs) are a model of computation that allow for random access to memory. This is the model underlying modern CPU architectures.
- RAMs are (believed to be) more powerful than multi-tape Turing machines.
- d-dimensional multi-/single-tape Turing machines have d-dimensional memory, and can move the tape heads to any neighboring cell in one step.
- Pointer machines allow the machine to maintain “pointers” to arbitrary cells in the memory (instead of directly accessing the memory as the RAMs).

Universal Turing machine

Theorem (Universal Multi-tape TM)

There exists a universal multi-tape Turing machine U such that for any multi-tape Turing machine M and input x:

- *U takes as input $\langle M, x \rangle$ (an encoding of M and x)*

Universal Turing machine

Theorem (Universal Multi-tape TM)

There exists a universal multi-tape Turing machine U such that for any multi-tape Turing machine M and input x:

- *U takes as input $\langle M, x \rangle$ (an encoding of M and x)*
- *U accepts $\langle M, x \rangle$ if and only if M accepts x*

Universal Turing machine

Theorem (Universal Multi-tape TM)

There exists a universal multi-tape Turing machine U such that for any multi-tape Turing machine M and input x:

- *U takes as input $\langle M, x\rangle$ (an encoding of M and x)*
- *U accepts $\langle M, x\rangle$ if and only if M accepts x*
- *If M runs in time $T(n)$ on input x of length n, then U runs in time $O(T(n) \log T(n))$ on input $\langle M, x\rangle$*

Universal Turing machine

Theorem (Universal Multi-tape TM)

There exists a universal multi-tape Turing machine U such that for any multi-tape Turing machine M and input x:

- *U takes as input $\langle M, x\rangle$ (an encoding of M and x)*
- *U accepts $\langle M, x\rangle$ if and only if M accepts x*
- *If M runs in time $T(n)$ on input x of length n, then U runs in time $O(T(n) \log T(n))$ on input $\langle M, x\rangle$*

- The same is true for other variants of Turing machines.

Universal Turing machine

Theorem (Universal Multi-tape TM)

There exists a universal multi-tape Turing machine U such that for any multi-tape Turing machine M and input x:

- *U takes as input* $\langle M, x \rangle$ *(an encoding of M and x)*
- *U accepts* $\langle M, x \rangle$ *if and only if M accepts x*
- *If M runs in time* $T(n)$ *on input x of length n, then U runs in time* $O(T(n) \log T(n))$ *on input* $\langle M, x \rangle$

- The same is true for other variants of Turing machines.
- The proof is technical (and won't be really needed for this course), can be found in Chapter 1 of Arora-Barak.

Universal Turing machine

Theorem (Universal multi-tape TM with time bound T)

There exists a universal multi-tape Turing machine U_{clock} such that for any multi-tape Turing machine M, input x, and time bound T:

- *U_{clock} takes as input $\langle M, x, T\rangle$ (an encoding of M, x, and T)*

Universal Turing machine

Theorem (Universal multi-tape TM with time bound T)

There exists a universal multi-tape Turing machine U_{clock} such that for any multi-tape Turing machine M, input x, and time bound T:

- *U_{clock} takes as input $\langle M, x, T\rangle$ (an encoding of M, x, and T)*
- *U_{clock} accepts $\langle M, x, T\rangle$ if and only if M accepts x within time T*

Universal Turing machine

Theorem (Universal multi-tape TM with time bound T)

There exists a universal multi-tape Turing machine U_{clock} such that for any multi-tape Turing machine M, input x, and time bound T:

- *U_{clock} takes as input $\langle M, x, T\rangle$ (an encoding of M, x, and T)*
- *U_{clock} accepts $\langle M, x, T\rangle$ if and only if M accepts x within time T*
- *U_{clock} runs in time $O(T(n) \log T(n))$ on input $\langle M, x, T\rangle$*

Universal Turing machine

Theorem (Universal multi-tape TM with time bound T)

There exists a universal multi-tape Turing machine U_{clock} such that for any multi-tape Turing machine M, input x, and time bound T:

- *U_{clock} takes as input $\langle M, x, T\rangle$ (an encoding of M, x, and T)*
- *U_{clock} accepts $\langle M, x, T\rangle$ if and only if M accepts x within time T*
- *U_{clock} runs in time $O(T(n) \log T(n))$ on input $\langle M, x, T\rangle$*

- The same is true for other variants of Turing machines.

Time hierarchy theorem for deterministic time

Definition (DTIME)

- For a function $T : \mathbb{N} \to \mathbb{N}$, we define $\mathrm{DTIME}[T(n)]$ to be the class of languages that can be decided by a deterministic multi-tape Turing machine in time $O(T(n))$.

Remark

Time hierarchy theorem for deterministic time

Definition (DTIME)

- For a function $T : \mathbb{N} \to \mathbb{N}$, we define $\mathrm{DTIME}[T(n)]$ to be the class of languages that can be decided by a deterministic multi-tape Turing machine in time $O(T(n))$.
- That is, $L \in \mathrm{DTIME}[T(n)]$ if there exists a deterministic multi-tape Turing machine M and a constant c such that:

Remark

Time hierarchy theorem for deterministic time

Definition (DTIME)

- For a function $T : \mathbb{N} \to \mathbb{N}$, we define $\mathrm{DTIME}[T(n)]$ to be the class of languages that can be decided by a deterministic multi-tape Turing machine in time $O(T(n))$.

- That is, $L \in \mathrm{DTIME}[T(n)]$ if there exists a deterministic multi-tape Turing machine M and a constant c such that:
  - M decides L (i.e., M accepts x if and only if $x \in L$)

Remark

Time hierarchy theorem for deterministic time

Definition (DTIME)

- For a function $T : \mathbb{N} \to \mathbb{N}$, we define $\mathrm{DTIME}[T(n)]$ to be the class of languages that can be decided by a deterministic multi-tape Turing machine in time $O(T(n))$.

- That is, $L \in \mathrm{DTIME}[T(n)]$ if there exists a deterministic multi-tape Turing machine M and a constant c such that:
    - M decides L (i.e., M accepts x if and only if $x \in L$)
    - For all inputs x of length n, M halts within $c \cdot T(n)$ steps

Remark

Time hierarchy theorem for deterministic time

Definition (DTIME)

- For a function $T : \mathbb{N} \to \mathbb{N}$, we define $\mathrm{DTIME}[T(n)]$ to be the class of languages that can be decided by a deterministic multi-tape Turing machine in time $O(T(n))$.

- That is, $L \in \mathrm{DTIME}[T(n)]$ if there exists a deterministic multi-tape Turing machine M and a constant c such that:
  - M decides L (i.e., M accepts x if and only if $x \in L$)
  - For all inputs x of length n, M halts within $c \cdot T(n)$ steps

Remark

- *In many situations, you want $T : \mathbb{N} \to \mathbb{N}$ to be time-constructible, i.e., there exists a deterministic multi-tape Turing machine that can compute $T(n)$ given (binary encoded input) n in time $O(T(n))$.*

Time hierarchy theorem for deterministic time

Theorem

- *Let $T : \mathbb{N} \to \mathbb{N}$ be a time-constructible function. There is a language $L \in DTIME[T(n) \log^2 T(n)]$ but $L \notin DTIME[T(n)]$.*

Proof

□

Time hierarchy theorem for deterministic time

Theorem

- *Let* $T : \mathbb{N} \to \mathbb{N}$ *be a time-constructible function. There is a language* $L \in DTIME[T(n) \log^2 T(n)]$ *but* $L \notin DTIME[T(n)]$.

Proof

- Let $\widetilde{T}(n) := T(n) \cdot \log \log T(n)$.

□

Time hierarchy theorem for deterministic time

Theorem

- *Let* $T : \mathbb{N} \to \mathbb{N}$ *be a time-constructible function. There is a language* $L \in DTIME[T(n) \log^2 T(n)]$ *but* $L \notin DTIME[T(n)]$.

Proof

- Let $\widetilde{T}(n) := T(n) \cdot \log\log T(n)$.
- Let $H(M) := \neg U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle)$. ($\neg$ means negation.)

□

Time hierarchy theorem for deterministic time

Theorem

- *Let $T : \mathbb{N} \to \mathbb{N}$ be a time-constructible function. There is a language $L \in DTIME[T(n) \log^2 T(n)]$ but $L \notin DTIME[T(n)]$.*

Proof

- Let $\widetilde{T}(n) := T(n) \cdot \log\log T(n)$.
- Let $H(M) := \neg U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle)$. ($\neg$ means negation.)
- Let L be the language decides by H. I.e.,

$$L = \{M \mid U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle) \text{ rejects}\}$$

□

Time hierarchy theorem for deterministic time

Theorem

- *Let* $T : \mathbb{N} \to \mathbb{N}$ *be a time-constructible function. There is a language* $L \in DTIME[T(n) \log^2 T(n)]$ *but* $L \notin DTIME[T(n)]$.

Proof

- Let $\widetilde{T}(n) := T(n) \cdot \log\log T(n)$.
- Let $H(M) := \neg U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle)$. ($\neg$ means negation.)
- Let L be the language decides by H. I.e.,

$$L = \{M \mid U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle) \text{ rejects}\}$$

- By construction, $L \in \text{DTIME}[T(n) \log^2 T(n)]$.

□

Time hierarchy theorem for deterministic time

Theorem

- *Let* $T : \mathbb{N} \to \mathbb{N}$ *be a time-constructible function. There is a language* $L \in DTIME[T(n) \log^2 T(n)]$ *but* $L \notin DTIME[T(n)]$.

Proof

- Let $\widetilde{T}(n) := T(n) \cdot \log \log T(n)$.
- Let $H(M) := \neg U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle)$. ($\neg$ means negation.)
- Let L be the language decides by H. I.e.,

$$L = \{M \mid U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle) \text{ rejects}\}$$

- By construction, $L \in \text{DTIME}[T(n) \log^2 T(n)]$.
- **Have to prove**: $L \notin \text{DTIME}[T(n)]$.

□

Time hierarchy theorem for deterministic time

Proof

- Let $\widetilde{T}(n) := T(n) \cdot \log\log T(n)$.

□

Time hierarchy theorem for deterministic time

Proof

- Let $\widetilde{T}(n) := T(n) \cdot \log \log T(n)$.
- Let $H(M) := \neg U_{\text{clock}}(\langle M, M, \widetilde{T}(n) \rangle)$.

Time hierarchy theorem for deterministic time

Proof

- Let $\widetilde{T}(n) := T(n) \cdot \log\log T(n)$.
- Let $H(M) := \neg U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle)$.
- Let L be the language decides by H. I.e.,

$$L = \{M \mid U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle) \text{ does not accept}\}$$

□

Time hierarchy theorem for deterministic time

Proof

- Let $\widetilde{T}(n) := T(n) \cdot \log\log T(n)$.
- Let $H(M) := \neg U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle)$.
- Let L be the language decides by H. I.e.,

$$L = \{M \mid U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle) \text{ does not accept}\}$$

- **For the sake of contradiction**, assume $L \in \mathrm{DTIME}[T(n)]$ and M decides L in $O(T(n))$ time.

□

Time hierarchy theorem for deterministic time

Proof

- Let $\widetilde{T}(n) := T(n) \cdot \log\log T(n)$.
- Let $H(M) := \neg U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle)$.
- Let L be the language decides by H. I.e.,

$$L = \{M \mid U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle) \text{ does not accept}\}$$

- **For the sake of contradiction**, assume $L \in \mathrm{DTIME}[T(n)]$ and M decides L in $O(T(n))$ time.
- Then $L(M) = M(M) = \neg U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle) = \neg M(M)$, contradiction!

□

Time hierarchy theorem for deterministic time

Proof

- Let $\widetilde{T}(n) := T(n) \cdot \log\log T(n)$.
- Let $H(M) := \neg U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle)$.
- Let L be the language decides by H. I.e.,

$$L = \{M \mid U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle) \text{ does not accept}\}$$

- **For the sake of contradiction**, assume $L \in \text{DTIME}[T(n)]$ and M decides L in $O(T(n))$ time.
- Then $L(M) = M(M) = \neg U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle) = \neg M(M)$, contradiction!
- Essentially the same proof for the hardness of the halting problem.

□

Recap: Multi-tape non-deterministic TM

Definition (Multi-tape non-deterministic Turing machine)

- A *multi-tape non-deterministic Turing machine* is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ where:

Recap: Multi-tape non-deterministic TM

Definition (Multi-tape non-deterministic Turing machine)

- A *multi-tape non-deterministic Turing machine* is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ where:
  - Q is a finite set of states

Recap: Multi-tape non-deterministic TM

Definition (Multi-tape non-deterministic Turing machine)

- A *multi-tape non-deterministic Turing machine* is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ where:
  - Q is a finite set of states
  - Σ is the input alphabet

Recap: Multi-tape non-deterministic TM

Definition (Multi-tape non-deterministic Turing machine)

- A *multi-tape non-deterministic Turing machine* is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ where:
  - Q is a finite set of states
  - Σ is the input alphabet
  - Γ is the tape alphabet (with $\Sigma \subseteq \Gamma$)

Recap: Multi-tape non-deterministic TM

Definition (Multi-tape non-deterministic Turing machine)

- A *multi-tape non-deterministic Turing machine* is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ where:
  - Q is a finite set of states
  - Σ is the input alphabet
  - Γ is the tape alphabet (with $\Sigma \subseteq \Gamma$)
  - $\delta : Q \times \Gamma^k \to \mathcal{P}(Q \times \Gamma^k \times \{L, R, S\}^k)$ is the transition function

Recap: Multi-tape non-deterministic TM

Definition (Multi-tape non-deterministic Turing machine)

- A *multi-tape non-deterministic Turing machine* is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ where:
  - Q is a finite set of states
  - Σ is the input alphabet
  - Γ is the tape alphabet (with $\Sigma \subseteq \Gamma$)
  - $\delta : Q \times \Gamma^k \to \mathcal{P}(Q \times \Gamma^k \times \{L, R, S\}^k)$ is the transition function
  - $q_0 \in Q$ is the initial state

Recap: Multi-tape non-deterministic TM

Definition (Multi-tape non-deterministic Turing machine)

- A *multi-tape non-deterministic Turing machine* is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ where:
  - Q is a finite set of states
  - Σ is the input alphabet
  - Γ is the tape alphabet (with $\Sigma \subseteq \Gamma$)
  - $\delta : Q \times \Gamma^k \to \mathcal{P}(Q \times \Gamma^k \times \{L, R, S\}^k)$ is the transition function
  - $q_0 \in Q$ is the initial state
  - $q_{acc}, q_{rej} \in Q$ are the accepting and rejecting states

Recap: Multi-tape non-deterministic TM

Definition (Multi-tape non-deterministic Turing machine)

- A *multi-tape non-deterministic Turing machine* is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ where:
  - Q is a finite set of states
  - Σ is the input alphabet
  - Γ is the tape alphabet (with $\Sigma \subseteq \Gamma$)
  - $\delta : Q \times \Gamma^k \to \mathcal{P}(Q \times \Gamma^k \times \{L, R, S\}^k)$ is the transition function
  - $q_0 \in Q$ is the initial state
  - $q_{acc}, q_{rej} \in Q$ are the accepting and rejecting states

- M accepts x if and only if there is a sequence of transitions that leads to q_{acc}.

Recap: Multi-tape non-deterministic Turing machine

Definition (NTIME)

- For a function $T : \mathbb{N} \to \mathbb{N}$, we define $\mathrm{NTIME}[T(n)]$ to be the class of languages that can be decided by a non-deterministic multi-tape Turing machine in time $O(T(n))$.

Recap: Multi-tape non-deterministic Turing machine

Definition (NTIME)

- For a function $T : \mathbb{N} \to \mathbb{N}$, we define $\mathrm{NTIME}[T(n)]$ to be the class of languages that can be decided by a non-deterministic multi-tape Turing machine in time $O(T(n))$.
- That is, $L \in \mathrm{NTIME}[T(n)]$ if there exists a non-deterministic multi-tape Turing machine M and a constant c such that:

Recap: Multi-tape non-deterministic Turing machine

Definition (NTIME)

- For a function $T : \mathbb{N} \to \mathbb{N}$, we define $\mathrm{NTIME}[T(n)]$ to be the class of languages that can be decided by a non-deterministic multi-tape Turing machine in time $O(T(n))$.

- That is, $L \in \mathrm{NTIME}[T(n)]$ if there exists a non-deterministic multi-tape Turing machine M and a constant c such that:
  - M decides L (i.e., M accepts x if and only if $x \in L$)

Recap: Multi-tape non-deterministic Turing machine

Definition (NTIME)

- For a function $T : \mathbb{N} \to \mathbb{N}$, we define $\mathrm{NTIME}[T(n)]$ to be the class of languages that can be decided by a non-deterministic multi-tape Turing machine in time $O(T(n))$.

- That is, $L \in \mathrm{NTIME}[T(n)]$ if there exists a non-deterministic multi-tape Turing machine M and a constant c such that:
  - M decides L (i.e., M accepts x if and only if $x \in L$)
  - For all inputs x of length n, M halts within $c \cdot T(n)$ steps

Universal non-deterministic TM

Theorem (Universal non-deterministic TM with time bound T***)***

- *There exists a universal non-deterministic multi-tape Turing machine* U *such that for any non-deterministic multi-tape Turing machine* M *and input* x*:*

Universal non-deterministic TM

Theorem (Universal non-deterministic TM with time bound T)

- *There exists a universal non-deterministic multi-tape Turing machine U such that for any non-deterministic multi-tape Turing machine M and input x:*
  - *U takes as input $\langle M, x, T\rangle$ (an encoding of M, x, and T)*

Universal non-deterministic TM

Theorem (Universal non-deterministic TM with time bound T)

- *There exists a universal non-deterministic multi-tape Turing machine U such that for any non-deterministic multi-tape Turing machine M and input x:*
  - *U takes as input $\langle M, x, T\rangle$ (an encoding of M, x, and T)*
  - *U accepts $\langle M, x, T\rangle$ if and only if M accepts x within time T.*

Universal non-deterministic TM

Theorem (Universal non-deterministic TM with time bound** T**)

- *There exists a universal non-deterministic multi-tape Turing machine* U *such that for any non-deterministic multi-tape Turing machine* M *and input* x*:*
  - U *takes as input* $\langle M, x, T\rangle$ *(an encoding of* M, x, *and* T*)*
  - U *accepts* $\langle M, x, T\rangle$ *if and only if* M *accepts* x *within time* T.
  - U *runs in time* $O(T(n))$.

Time hierarchy theorem for non-deterministic time

- Before stating and prove the non-deterministic time hierarchy theorem,

Time hierarchy theorem for non-deterministic time

- Before stating and prove the non-deterministic time hierarchy theorem,
- **Question**: Can you think about why we cannot use the same proof as the deterministic time hierarchy theorem?

Time hierarchy theorem for non-deterministic time

- Before stating and prove the non-deterministic time hierarchy theorem,
- **Question**: Can you think about why we cannot use the same proof as the deterministic time hierarchy theorem?
- It seems for deterministic time hierarchy theorem, we only used the existence of universal Turing machine? And such machine exists for non-deterministic time as well?

Time hierarchy theorem for non-deterministic time

- Before stating and prove the non-deterministic time hierarchy theorem,
- **Question**: Can you think about why we cannot use the same proof as the deterministic time hierarchy theorem?
- It seems for deterministic time hierarchy theorem, we only used the existence of universal Turing machine? And such machine exists for non-deterministic time as well?
- **Answer**: "Let $H(M) := \neg U_{\text{clock}}(\langle M, M, \widetilde{T}(n)\rangle)$." This $\neg$ cannot be done for non-deterministic Turing machine!

Time hierarchy theorem for non-deterministic time

Theorem (Non-deterministic Time Hierarchy Theorem)

- *Let $f, g : \mathbb{N} \to \mathbb{N}$ be time-constructible functions such that $f(n+1) = o(g(n))$. Then*

$$NTIME[f(n)] \subsetneq NTIME[g(n)]$$

Time hierarchy theorem for non-deterministic time

Theorem (Non-deterministic Time Hierarchy Theorem)

- *Let $f, g : \mathbb{N} \to \mathbb{N}$ be time-constructible functions such that $f(n+1) = o(g(n))$. Then*

$$NTIME[f(n)] \subsetneq NTIME[g(n)]$$

- **Proof:** see the white board!

infinitey often and almost everywhere separation

- After studying the non-deterministic time hierarchy theorem, can you name a serious issue with it?

infinitey often and almost everywhere separation

- After studying the non-deterministic time hierarchy theorem, can you name a serious issue with it?

- **Answer**: It only shows for a very few input lengths, the hard language is hard.

infinitey often and almost everywhere separation

- After studying the non-deterministic time hierarchy theorem, can you name a serious issue with it?

- **Answer**: It only shows for a very few input lengths, the hard language is hard.

- **Infinite often separation (default)**: The language L is not in $\mathrm{NTIME}[T(n)]$ if and only if $L' \in \mathrm{NTIME}[T(n)]$, for **infinitely many** input lengths n, $L_n \neq L'_n$. (here, L_n is the language L on input length n.)

infinitey often and almost everywhere separation

- After studying the non-deterministic time hierarchy theorem, can you name a serious issue with it?

- **Answer**: It only shows for a very few input lengths, the hard language is hard.

- **Infinite often separation (default)**: The language L is not in $\text{NTIME}[T(n)]$ if and only if $L' \in \text{NTIME}[T(n)]$, for **infinitely many** input lengths n, $L_n \neq L'_n$. (here, L_n is the language L on input length n.)

- **Almost everywhere separation**: For every $L' \in \text{NTIME}[T(n)]$, for **all except finitely many** n, $L_n \neq L'_n$.

infinitey often and almost everywhere separation

- After studying the non-deterministic time hierarchy theorem, can you name a serious issue with it?

- **Answer**: It only shows for a very few input lengths, the hard language is hard.

- **Infinite often separation (default)**: The language L is not in $\mathrm{NTIME}[T(n)]$ if and only if $L' \in \mathrm{NTIME}[T(n)]$, for **infinitely many** input lengths n, $L_n \neq L'_n$. (here, L_n is the language L on input length n.)

- **Almost everywhere separation**: For every $L' \in \mathrm{NTIME}[T(n)]$, for **all except finitely many** n, $L_n \neq L'_n$.

- **Big open question**: prove an almost everywhere separation between $\mathrm{NTIME}[n^2]$ and $\mathrm{NTIME}[2^n]$.

A weaker a.e. ntime hierarchy theorem

Definition (Non-deterministic time with bounded guess)

- For a function $T, G : \mathbb{N} \to \mathbb{N}$, we define $\mathrm{NTIMEGUESS}[T(n), G(n)]$ to be the class of languages that can be decided by a non-deterministic multi-tape Turing machine in time $O(T(n))$ with making at most $G(n)$ non-deterministic guesses.

A weaker a.e. ntime hierarchy theorem

Definition (Non-deterministic time with bounded guess)

- For a function $T, G : \mathbb{N} \to \mathbb{N}$, we define NTIMEGUESS$[T(n), G(n)]$ to be the class of languages that can be decided by a non-deterministic multi-tape Turing machine in time $O(T(n))$ with making at most $G(n)$ non-deterministic guesses.

- That is, $L \in$ NTIMEGUESS$[T(n), G(n)]$ if there exists a non-deterministic multi-tape Turing machine M and a constant c such that:

A weaker a.e. ntime hierarchy theorem

Definition (Non-deterministic time with bounded guess)

- For a function $T, G : \mathbb{N} \to \mathbb{N}$, we define $\mathrm{NTIMEGUESS}[T(n), G(n)]$ to be the class of languages that can be decided by a non-deterministic multi-tape Turing machine in time $O(T(n))$ with making at most $G(n)$ non-deterministic guesses.

- That is, $L \in \mathrm{NTIMEGUESS}[T(n), G(n)]$ if there exists a non-deterministic multi-tape Turing machine M and a constant c such that:
  - M decides L (i.e., M accepts x if and only if $x \in L$)

A weaker a.e. ntime hierarchy theorem

Definition (Non-deterministic time with bounded guess)

- For a function $T, G : \mathbb{N} \to \mathbb{N}$, we define NTIMEGUESS$[T(n), G(n)]$ to be the class of languages that can be decided by a non-deterministic multi-tape Turing machine in time $O(T(n))$ with making at most $G(n)$ non-deterministic guesses.

- That is, $L \in$ NTIMEGUESS$[T(n), G(n)]$ if there exists a non-deterministic multi-tape Turing machine M and a constant c such that:
    - M decides L (i.e., M accepts x if and only if $x \in L$)
    - For all inputs x of length n, M halts within $c \cdot T(n)$ steps and makes at most $G(n)$ non-deterministic guesses.

A weaker a.e. ntime hierarchy theorem

Theorem (Non-deterministic time hierarchy theorem with bounded guess)

- *Let* $T, G, W \colon \mathbb{N} \to \mathbb{N}$ *be time-constructible functions such that* $G(n) = o(T(n))$ *and* $W(n) = o(n)$*. Then there is a language* $L \in NTIME[T(n)]$ *but* L *is almost-everywhere separated from* $NTIMEGUESS[G(n), W(n)]$.

A weaker a.e. ntime hierarchy theorem

Theorem (Non-deterministic time hierarchy theorem with bounded guess)

- *Let $T, G, W : \mathbb{N} \to \mathbb{N}$ be time-constructible functions such that $G(n) = o(T(n))$ and $W(n) = o(n)$. Then there is a language $L \in NTIME[T(n)]$ but L is almost-everywhere separated from $NTIMEGUESS[G(n), W(n)]$.*

- **Proof:** see the white board!