CS310 : Automata Theory 2019

Lecture 36: Efficiency in computation

Instructor: S. Akshay

IITB, India

04-04-2019

Recap

Turing machines and computability

1. Turing machines
   (i) Definition
   (ii) Variants
   (iii) Decidable and Turing recognizable languages
   (iv) Church-Turing Hypothesis

Recap

Turing machines and computability

1. Turing machines
   (i) Definition
   (ii) Variants
   (iii) Decidable and Turing recognizable languages
   (iv) Church-Turing Hypothesis
2. Undecidability
   (i) A proof technique by diagonalization
   (ii) Via reductions
   (iii) Rice's theorem

Recap

Turing machines and computability

1. Turing machines
   (i) Definition
   (ii) Variants
   (iii) Decidable and Turing recognizable languages
   (iv) Church-Turing Hypothesis
2. Undecidability
   (i) A proof technique by diagonalization
   (ii) Via reductions
   (iii) Rice's theorem
3. Applications: showing (un)decidability of other problems
   (i) A string matching problem: Post's Correspondance Problem
   (ii) A problem for compilers: Unambiguity of Context-free languages
   (iii) Between TM and PDA: Linear Bounded Automata

Recap

Turing machines and computability

1. Turing machines
   (i) Definition
   (ii) Variants
   (iii) Decidable and Turing recognizable languages
   (iv) Church-Turing Hypothesis
2. Undecidability
   (i) A proof technique by diagonalization
   (ii) Via reductions
   (iii) Rice's theorem
3. Applications: showing (un)decidability of other problems
   (i) A string matching problem: Post's Correspondance Problem
   (ii) A problem for compilers: Unambiguity of Context-free languages
   (iii) Between TM and PDA: Linear Bounded Automata
4. Efficiency in computation: run-time complexity.

Running Time Complexity

Given M a halting TM, running time of M is the function $f(n) : \mathbb{N} \to \mathbb{N}$, which counts the maximum number of steps that M uses on any input of length n.

Running Time Complexity

Given M a halting TM, running time of M is the function $f(n) : \mathbb{N} \to \mathbb{N}$, which counts the maximum number of steps that M uses on any input of length n.

Is this the only notion possible? Any others?

Running Time Complexity

Given M a halting TM, running time of M is the function $f(n): \mathbb{N} \to \mathbb{N}$, which counts the maximum number of steps that M uses on any input of length n.

- Worst-case complexity - longest running time of all inputs of length n (in this course, we consider this)
- Average-case complexity - average running time over all inputs of length n.

Running Time Complexity

Let $t : \mathbb{N} \to \mathbb{R}^+$. A language $L \subseteq \Sigma^*$ is said to be in $TIME(t(n))$ if there exists a deterministic (halting) Turing machine M such that $\forall x \in \Sigma^*$ of length n, M halts on x within time $O(t(n))$.

Running Time Complexity

Let $t : \mathbb{N} \to \mathbb{R}^+$. A language $L \subseteq \Sigma^*$ is said to be in $TIME(t(n))$ if there exists a deterministic (halting) Turing machine M such that $\forall x \in \Sigma^*$ of length n, M halts on x within time $O(t(n))$.

$TIME(t(n))$ is set of all languages decidable by a $O(t(n))$ TM

Running Time Complexity

Let $t : \mathbb{N} \to \mathbb{R}^+$. A language $L \subseteq \Sigma^*$ is said to be in $TIME(t(n))$ if there exists a deterministic (halting) Turing machine M such that $\forall x \in \Sigma^*$ of length n, M halts on x within time $O(t(n))$.

$TIME(t(n))$ is set of all languages decidable by a $O(t(n))$ TM

Exercise: Is $A = \{0^k 1^k \mid k \geq 0\}$ in $O(n^2)$?

Running Time Complexity

Let $t : \mathbb{N} \to \mathbb{R}^+$. A language $L \subseteq \Sigma^*$ is said to be in $TIME(t(n))$ if there exists a deterministic (halting) Turing machine M such that $\forall x \in \Sigma^*$ of length n, M halts on x within time $O(t(n))$.

$TIME(t(n))$ is set of all languages decidable by a $O(t(n))$ TM

Exercise: Is $A = \{0^k 1^k \mid k \geq 0\}$ in $O(n^2)$?

1. scan tape once to check if input is of form 0^*1^*. else reject.
2. repeat until there are no 0's:
3. scan tape to cross a single 0 and single 1 (if you cant find 1 to cross off, reject)
4. at end if there are 1's remaining reject, otherwise, accept.

Running Time Complexity

Let $t : \mathbb{N} \to \mathbb{R}^+$. A language $L \subseteq \Sigma^*$ is said to be in $TIME(t(n))$ if there exists a deterministic (halting) Turing machine M such that $\forall x \in \Sigma^*$ of length n, M halts on x within time $O(t(n))$.

$TIME(t(n))$ is set of all languages decidable by a $O(t(n))$ TM

Exercise: Is $A = \{0^k 1^k \mid k \geq 0\}$ in $O(n^2)$?

1. scan tape once to check if input is of form 0^*1^*. else reject. $O(n)$ steps
2. repeat until there are no 0's:
3. scan tape to cross a single 0 and single 1 (if you cant find 1 to cross off, reject)
4. at end if there are 1's remaining reject, otherwise, accept.

Running Time Complexity

Let $t : \mathbb{N} \to \mathbb{R}^+$. A language $L \subseteq \Sigma^*$ is said to be in $TIME(t(n))$ if there exists a deterministic (halting) Turing machine M such that $\forall x \in \Sigma^*$ of length n, M halts on x within time $O(t(n))$.

$TIME(t(n))$ is set of all languages decidable by a $O(t(n))$ TM

Exercise: Is $A = \{0^k 1^k \mid k \geq 0\}$ in $O(n^2)$?

1. scan tape once to check if input is of form 0^*1^*. else reject. $O(n)$ steps
2. repeat until there are no 0's:
3. scan tape to cross a single 0 and single 1 (if you cant find 1 to cross off, reject) $O(n)$ steps
4. at end if there are 1's remaining reject, otherwise, accept.

Running Time Complexity

Let $t : \mathbb{N} \to \mathbb{R}^+$. A language $L \subseteq \Sigma^*$ is said to be in $TIME(t(n))$ if there exists a deterministic (halting) Turing machine M such that $\forall x \in \Sigma^*$ of length n, M halts on x within time $O(t(n))$.

$TIME(t(n))$ is set of all languages decidable by a $O(t(n))$ TM

Exercise: Is $A = \{0^k 1^k \mid k \geq 0\}$ in $O(n^2)$?

1. scan tape once to check if input is of form 0^*1^*. else reject. $O(n)$ steps
2. repeat until there are no 0's: $n/2$ repetitions
3. scan tape to cross a single 0 and single 1 (if you cant find 1 to cross off, reject) $O(n)$ steps
4. at end if there are 1's remaining reject, otherwise, accept.

Running Time Complexity

Let $t : \mathbb{N} \to \mathbb{R}^+$. A language $L \subseteq \Sigma^*$ is said to be in $TIME(t(n))$ if there exists a deterministic (halting) Turing machine M such that $\forall x \in \Sigma^*$ of length n, M halts on x within time $O(t(n))$.

$TIME(t(n))$ is set of all languages decidable by a $O(t(n))$ TM

Exercise: Is $A = \{0^k 1^k \mid k \geq 0\}$ in $O(n^2)$?

1. scan tape once to check if input is of form 0^*1^*. else reject. $O(n)$ steps
2. repeat until there are no 0's: $n/2$ repetitions
3. scan tape to cross a single 0 and single 1 (if you cant find 1 to cross off, reject) $O(n)$ steps
4. at end if there are 1's remaining reject, otherwise, accept. $O(n)$ steps

Running Time Complexity

Let $t : \mathbb{N} \to \mathbb{R}^+$. A language $L \subseteq \Sigma^*$ is said to be in $TIME(t(n))$ if there exists a deterministic (halting) Turing machine M such that $\forall x \in \Sigma^*$ of length n, M halts on x within time $O(t(n))$.

$TIME(t(n))$ is set of all languages decidable by a $O(t(n))$ TM

Exercise: Is $A = \{0^k 1^k \mid k \geq 0\}$ in $O(n^2)$?

1. scan tape once to check if input is of form 0^*1^*. else reject. $O(n)$ steps
2. repeat until there are no 0's: $n/2$ repetitions
3. scan tape to cross a single 0 and single 1 (if you cant find 1 to cross off, reject) $O(n)$ steps
4. at end if there are 1's remaining reject, otherwise, accept. $O(n)$ steps

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

Running Time Complexity (Contd.)

$TIME(t(n))$ is set of all languages decidable by a $O(t(n))$ Turing machine

Questions

- Show that $A = \{0^k 1^k \mid k \geq 0\}$ in $O(n \log(n))$?
- Is A in $O(n)$ No. why?

Running Time Complexity (Contd.)

$TIME(t(n))$ is set of all languages decidable by a $O(t(n))$ Turing machine

Questions

- Show that $A = \{0^k 1^k \mid k \geq 0\}$ in $O(n\log(n))$?
- Is A in $O(n)$ No. why?

Does crossing two 0s and 1s on every scan instead of just one help?

Running Time Complexity (Contd.)

$TIME(t(n))$ is set of all languages decidable by a $O(t(n))$ Turing machine

Questions

- Show that $A = \{0^k 1^k \mid k \geq 0\}$ in $O(n \log(n))$?
- Is A in $O(n)$ No. why?

- Scan tape and reject, if 0 is to right of 1.
- Scan the 0s and copy them to another tape, until first 1.
- Scan 1s in first tape together with 0s in second tape, if 0's are crossed before 1s are read, reject
- if all 0s are crossed off at the end, accept, else reject.

Running Time Complexity (Contd.)

$TIME(t(n))$ is set of all languages decidable by a $O(t(n))$ Turing machine

Questions

- Show that $A = \{0^k 1^k \mid k \geq 0\}$ in $O(n \log(n))$?
- Is A in $O(n)$ No. why?

- Scan tape and reject, if 0 is to right of 1.
- Scan the 0s and copy them to another tape, until first 1.
- Scan 1s in first tape together with 0s in second tape, if 0's are crossed before 1s are read, reject
- if all 0s are crossed off at the end, accept, else reject.

So, with 2-tape can improve "complexity".

Running Time Complexity (Contd.)

$TIME(t(n))$ is set of all languages decidable by a $O(t(n))$ 1-tape Turing machine

Questions

- (HW) Show that $A = \{0^k 1^k \mid k \geq 0\}$ in $O(n\log(n))$?
- Is A in $O(n)$ No. why?

- Scan tape and reject, if 0 is to right of 1.
- Scan the 0s and copy them to another tape, until first 1.
- Scan 1s in first tape together with 0s in second tape, if 0's are crossed before 1s are read, reject
- if all 0s are crossed off at the end, accept, else reject.

So, with 2-tape can improve "complexity". Can we improve with 1-tape?

Running Time Complexity (Contd.)

$TIME(t(n))$ is set of all languages decidable by a $O(t(n))$ 1-tape Turing machine

Questions

- (HW) Show that $A = \{0^k 1^k \mid k \geq 0\}$ in $O(n \log(n))$?
- (Challenge) Is A in $O(n)$ No. why?

So, with 2-tape can improve "complexity".

Running Time Complexity (Contd.)

$TIME(t(n))$ is set of all languages decidable by a $O(t(n))$ 1-tape Turing machine

Questions

- (HW) Show that $A = \{0^k 1^k \mid k \geq 0\}$ in $O(n \log(n))$?
- (Challenge) Is A in $O(n)$ No. why?

So, with 2-tape can improve "complexity".

Conclusions: change of model can change complexity, even if it does not change computability.

Complexity between models of computation

Computability vs Complexity

- Computability: (Church Turing Hypothesis) All reasonable models of computation are equivalent, i.e., they decide the same class of languages.
- Complexity: Choice of model affects running time.

Complexity between models of computation

Computability vs Complexity

- Computability: (Church Turing Hypothesis) All reasonable models of computation are equivalent, i.e., they decide the same class of languages.
- Complexity: Choice of model affects running time.

Our goal

Complexity between models of computation

Computability vs Complexity

- Computability: (Church Turing Hypothesis) All reasonable models of computation are equivalent, i.e., they decide the same class of languages.
- Complexity: Choice of model affects running time.

Our goal

- to measure or classify problems by their (running) time complexity

Complexity between models of computation

Computability vs Complexity

- Computability: (Church Turing Hypothesis) All reasonable models of computation are equivalent, i.e., they decide the same class of languages.
- Complexity: Choice of model affects running time.

Our goal

- to measure or classify problems by their (running) time complexity
- But if change of model changes complexity, then how can we measure or classify them?

Complexity between models of computation

Computability vs Complexity

- Computability: (Church Turing Hypothesis) All reasonable models of computation are equivalent, i.e., they decide the same class of languages.
- Complexity: Choice of model affects running time.

Our goal

- to measure or classify problems by their (running) time complexity
- But if change of model changes complexity, then how can we measure or classify them?
- Fortunately, it doesn't change by much, at least for deterministic models!

Complexity between models of computation

Computability vs Complexity

- Computability: (Church Turing Hypothesis) All reasonable models of computation are equivalent, i.e., they decide the same class of languages.
- Complexity: Choice of model affects running time.

Our goal

- to measure or classify problems by their (running) time complexity
- But if change of model changes complexity, then how can we measure or classify them?
- Fortunately, it doesn't change by much, at least for deterministic models!

Theorem

Let $t(n)$ be a function such that $t(n) \geq n$. Then every $t(n)$ time multitape det TM has an equivalent $O(t^2(n))$ time 1-tape det TM.

Multi-tape to single tape

Theorem

Let $t(n)$ be a function such that $t(n) \geq n$. Then every $t(n)$ time multitape det TM has an equivalent $O(t^2(n))$ time 1-tape det TM.

Multi-tape to single tape

Theorem

Let $t(n)$ be a function such that $t(n) \geq n$. Then every $t(n)$ time multitape det TM has an equivalent $O(t^2(n))$ time 1-tape det TM.

Proof: Given k-tape TM M running in $t(n)$ time, define 1-tape TM S:

Multi-tape to single tape

Theorem

Let $t(n)$ be a function such that $t(n) \geq n$. Then every $t(n)$ time multitape det TM has an equivalent $O(t^2(n))$ time 1-tape det TM.

Proof: Given k-tape TM M running in $t(n)$ time, define 1-tape TM S:

- Store k-tapes of M in 1-tape of S, with head positions marked.
- To simulate one-step of M,

Multi-tape to single tape

Theorem

Let $t(n)$ be a function such that $t(n) \geq n$. Then every $t(n)$ time multitape det TM has an equivalent $O(t^2(n))$ time 1-tape det TM.

Proof: Given k-tape TM M running in $t(n)$ time, define 1-tape TM S:

- Store k-tapes of M in 1-tape of S, with head positions marked.
- To simulate one-step of M,
  - S scans all info on its tape to check all head positions
  - then makes another pass over tape to update tape contents and head positions.
  - If some head moves rightward into previously unread portion of tape in M, then in S, space allocated for that tape is increased by a right-shift of all content to right.

Multi-tape to single tape

Theorem

Let $t(n)$ be a function such that $t(n) \geq n$. Then every $t(n)$ time multitape det TM has an equivalent $O(t^2(n))$ time 1-tape det TM.

Proof: Given k-tape TM M running in $t(n)$ time, define 1-tape TM S:

- Store k-tapes of M in 1-tape of S, with head positions marked. $O(n)$
- To simulate one-step of M,
  - S scans all info on its tape to check all head positions $O(t(n))$ steps
  - then makes another pass over tape to update tape contents and head positions. $O(t(n))$ steps
  - If some head moves rightward into previously unread portion of tape in M, then in S, space allocated for that tape is increased by a right-shift of all content to right.

Multi-tape to single tape

Theorem

Let $t(n)$ be a function such that $t(n) \geq n$. Then every $t(n)$ time multitape det TM has an equivalent $O(t^2(n))$ time 1-tape det TM.

Proof: Given k-tape TM M running in $t(n)$ time, define 1-tape TM S:

- Store k-tapes of M in 1-tape of S, with head positions marked. $O(n)$
- To simulate one-step of M,
  - S scans all info on its tape to check all head positions $O(t(n))$ steps
  - then makes another pass over tape to update tape contents and head positions. $O(t(n))$ steps
  - If some head moves rightward into previously unread portion of tape in M, then in S, space allocated for that tape is increased by a right-shift of all content to right. k tapes $=$ k heads$=k \times O(t(n))$ steps

Multi-tape to single tape

Theorem

Let $t(n)$ be a function such that $t(n) \geq n$. Then every $t(n)$ time multitape det TM has an equivalent $O(t^2(n))$ time 1-tape det TM.

Proof: Given k-tape TM M running in $t(n)$ time, define 1-tape TM S:

- Store k-tapes of M in 1-tape of S, with head positions marked. $O(n)$
- To simulate one-step of M, $O(t(n))$
  - S scans all info on its tape to check all head positions $O(t(n))$ steps
  - then makes another pass over tape to update tape contents and head positions. $O(t(n))$ steps
  - If some head moves rightward into previously unread portion of tape in M, then in S, space allocated for that tape is increased by a right-shift of all content to right. k tapes $=$ k heads$=k \times O(t(n))$ steps

Multi-tape to single tape

Theorem

Let $t(n)$ be a function such that $t(n) \geq n$. Then every $t(n)$ time multitape det TM has an equivalent $O(t^2(n))$ time 1-tape det TM.

Proof: Given k-tape TM M running in $t(n)$ time, define 1-tape TM S:

- Store k-tapes of M in 1-tape of S, with head positions marked. $O(n)$
- To simulate one-step of M, $O(t(n))$
  - S scans all info on its tape to check all head positions $O(t(n))$ steps
  - then makes another pass over tape to update tape contents and head positions. $O(t(n))$ steps
  - If some head moves rightward into previously unread portion of tape in M, then in S, space allocated for that tape is increased by a right-shift of all content to right. k tapes $= k$ heads$=k \times O(t(n))$ steps
- $t(n)$ steps of M implies $t(n) \times O(t(n)) = O(t^2(n))$ steps

Multi-tape to single tape

Theorem

Let $t(n)$ be a function such that $t(n) \geq n$. Then every $t(n)$ time multitape det TM has an equivalent $O(t^2(n))$ time 1-tape det TM.

Proof: Given k-tape TM M running in $t(n)$ time, define 1-tape TM S:

- Store k-tapes of M in 1-tape of S, with head positions marked. $O(n)$
- To simulate one-step of M, $O(t(n))$
  - S scans all info on its tape to check all head positions $O(t(n))$ steps
  - then makes another pass over tape to update tape contents and head positions. $O(t(n))$ steps
  - If some head moves rightward into previously unread portion of tape in M, then in S, space allocated for that tape is increased by a right-shift of all content to right. k tapes $=$ k heads$=k \times O(t(n))$ steps
- $t(n)$ steps of M implies $t(n) \times O(t(n)) = O(t^2(n))$ steps
- Overall: $O(n) + O(t^2(n)) = O(t^2(n))$ (since $t(n) \geq n$)

What about non-determinism?

Running time of a non-det halting TM

The running time of a non-det halting TM N is the function $f(n : \mathbb{N} \to \mathbb{N})$, where $f(n)$ is the max number of steps that N uses on any branch of its computation on any input of length n.

What about non-determinism?

Running time of a non-det halting TM

The running time of a non-det halting TM N is the function $f(n : \mathbb{N} \to \mathbb{N})$, where $f(n)$ is the max number of steps that N uses on any branch of its computation on any input of length n.

Theorem

Let $t(n)$ be a function such that $t(n) \geq n$. Then every $t(n)$ time non-det 1-tape TM N has an equivalent $2^{O(t(n))}$ time det 1-tape TM D.