CS310 : Automata Theory 2019

Lecture 39: Efficiency in computation
Classifying problems by their complexity

Instructor: S. Akshay

IITB, India

11-04-2019

Recap

Turing machines and computability

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

Recap

Turing machines and computability

1. Turing machines
   (i) Definition & variants
   (ii) Decidable and Turing recognizable languages
   (iii) 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 & variants
   (ii) Decidable and Turing recognizable languages
   (iii) 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 & variants
   (ii) Decidable and Turing recognizable languages
   (iii) 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.
   (i) Running time complexity
   (ii) Polynomial and exponential time complexity
   (iii) Nondeterministic polynomial time, and the P vs NP problem.

The P vs NP problem

- P: class of problems solvable in polynomial time
- NP: class of problems verifiable in polynomial time

The P vs NP problem

- P: class of problems solvable in polynomial time
- NP: class of problems verifiable in polynomial time
  $=$ class of problems solvable in polynomial time in a non-determistic TM.
- EXP: class of problems solvable in exponential time.
- $Co - \mathcal{C}$: class of problems whose complement is solvable in $\mathcal{C}$.

The class NEXP

Definition

NEXP is the class of languages that are decidable in exponential time on a non-deterministic single-tape Turing machine, i.e.,

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

The class NEXP

Definition

NEXP is the class of languages that are decidable in exponential time on a non-deterministic single-tape Turing machine, i.e.,

$$NP = \bigcup_{k} NTIME(2^{n^k})$$

Verifier

for language A is an algorithm V s.t. $w \in A$ iff V accepts $\langle w, c \rangle$ for some witness or proof string c.

The class NEXP

Definition

NEXP is the class of languages that are decidable in exponential time on a non-deterministic single-tape Turing machine, i.e.,

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

Verifier

for language A is an algorithm V s.t. $w \in A$ iff V accepts $\langle w, c \rangle$ for some witness or proof string c.

Exercises

- Define an EXP-time verifier.
- Prove or disprove: a language is in NEXP iff it has a exp-time verifier

NP completeness

The problem of satisfiability SAT

Given a Boolean formula, i.e., an expression with Boolean variables and operations, does it have a satisfying assignment?

NP completeness

The problem of satisfiability SAT

Given a Boolean formula, i.e., an expression with Boolean variables and operations, does it have a satisfying assignment?

$$SAT = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable Boolean formula.}\}$$

NP completeness

The problem of satisfiability SAT

Given a Boolean formula, i.e., an expression with Boolean variables and operations, does it have a satisfying assignment?

$$SAT = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable Boolean formula.}\}$$

Example: $\phi = (\overline{x} \wedge y \wedge z) \vee (x \wedge z)$

NP completeness

The problem of satisfiability SAT

Given a Boolean formula, i.e., an expression with Boolean variables and operations, does it have a satisfying assignment?

$$SAT = \{\langle\phi\rangle \mid \phi \text{ is a satisfiable Boolean formula.}\}$$

Example: $\phi = (\overline{x} \wedge y \wedge z) \vee (x \wedge z)$

Theorem

$SAT \in P$ iff $P = NP$

NP completeness

The problem of satisfiability SAT

Given a Boolean formula, i.e., an expression with Boolean variables and operations, does it have a satisfying assignment?

$$SAT = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable Boolean formula.}\}$$

Example: $\phi = (\overline{x} \wedge y \wedge z) \vee (x \wedge z)$

Theorem

$SAT \in P$ iff $P = NP$

$SAT = \{\langle \phi \rangle \mid \phi$ is a satisfiable 3-cnf-formula.$\}$

where 3-cnf-formula is a formula in special form:

- conjunction of "clauses"
- each clause has literals, i.e., variables or their negations, separated by disjunction
- each clause has 3 literals

Polynomial time reductions

Ptime computable functions

$f : \Sigma^* \to \Sigma^*$ is Ptime computable if there is a polytime TM M, which started on any input w halts with just $f(w)$ on its tape.

Polynomial time reductions

Ptime computable functions

$f : \Sigma^* \to \Sigma^*$ is Ptime computable if there is a polytime TM M, which started on any input w halts with just $f(w)$ on its tape.

Polynomial time reduction

Language A polynomtial time reducible to B, denoted $A \leq_P B$ if there is a Ptime computable function f s.t.

$$w \in A \Leftrightarrow f(w) \in B$$

Such a function is called a Ptime reduction of A to B.

Polynomial time reductions

Ptime computable functions

$f : \Sigma^* \to \Sigma^*$ is Ptime computable if there is a polytime TM M, which started on any input w halts with just $f(w)$ on its tape.

Polynomial time reduction

Language A polynomtial time reducible to B, denoted $A \leq_P B$ if there is a Ptime computable function f s.t.

$$w \in A \Leftrightarrow f(w) \in B$$

Such a function is called a Ptime reduction of A to B.

Theorem

If $A \leq_p B$ and $B \in P$, then $A \in P$

Polynomial time reductions

Ptime computable functions

$f : \Sigma^* \to \Sigma^*$ is Ptime computable if there is a polytime TM M, which started on any input w halts with just $f(w)$ on its tape.

Polynomial time reduction

Language A polynomtial time reducible to B, denoted $A \leq_P B$ if there is a Ptime computable function f s.t.

$$w \in A \Leftrightarrow f(w) \in B$$

Such a function is called a Ptime reduction of A to B.

Theorem

If $A \leq_p B$ and $B \in P$, then $A \in P$

(Note: if there is a "halting TM" reduction from A to B, then A undecidable implied B undecidable!)

Exercise (H.W)

- Show that $3SAT$ is polytime reducible to CLIQUE.
- Show that $3SAT$ is polytime reducible to SUBSETSUM.