COSE215: Theory of Computation

Lecture 21 — P, NP, and NP-Complete Problems

Hakjoo Oh
2017 Spring

Contents[1]

- The classes $\mathcal{P}$ and $\mathcal{NP}$
- Reductions
- NP-complete problems

[1] The slides are based on Siddhartha Sen's slides ("P, NP, and NP-Completeness")

Problems Solvable in Polynomial Time

- A Turing machine M is said to be of time complexity $T(n)$ if whenever M is given an input w of length n, M halts after making at most $T(n)$ moves, regardless of whether or not M accepts.
  - E.g., $T(n) = 5n^2$, $T(n) = 3^n + 5n^4$
- Polynomial time: $T(n) = a_0 n^k + a_1 n^{k-1} + \cdots + a_k n + a_{k+1}$
- We say a language L is in class $\mathcal{P}$ if there is some polynomial $T(n)$ such that $L = L(M)$ for some deterministic TM M of time complexity $T(n)$.
- Problems solvable in polynomial time are called *tractable*.
- Many familiar problems in a course on data structures and algorithms have efficient solutions and are generally in $\mathcal{P}$. E.g., finding a minimum-weight spanning tree (MWST)

Nondeterministic Polynomial Time

- We say a language L is in the class $\mathcal{NP}$ (nondeterministic polynomial) if there is a nondeterministic TM M and a polynomial time complexity $T(n)$ such that $L = L(M)$, and when M is given an input of length n, there are no sequences of more than $T(n)$ moves of M.
  - Example: TSP (Travelling Salesman Problem)
- $\mathcal{P} \subseteq \mathcal{NP}$ because every deterministic TM is a nondeterministic TM.
- However, it appears that $\mathcal{NP}$ contains many problems not in $\mathcal{P}$.
  - A NTM has the ability to guess an exponential number of possible solutions to a problem and check each one in polynomial time in parallel.
- It is one of the deepest open questions whether $\mathcal{P} = \mathcal{NP}$, i.e., whether in fact everything that can be done in polynomial time by NTM can in fact be done by a DTM in polynomial time, perhaps with a higher-order polynomial.

Implications of $\mathcal{P} = \mathcal{NP}$

If P=NP, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in creative leaps, no fundamental gap between solving a problem and recognizing the solution once its found. Everyone who could appreciate a symphony would be Mozart; everyone who could follow a step-by-step argument would be Gauss; everyone who could recognize a good investment strategy would be Warren Buffett.

— Scott Aaronson

NP-Complete Problems

- NP-complete problems are the hardest problems in the NP class.
- If any NP-complete problem can be solved in polynomial time, then all problems in NP are solvable in polynomial time.
- How to compare easiness/hardness of problems?

Problem Solving by Reduction

- L_1: the language (problem) to solve
- L_2: the problem for which we have an algorithm to solve
- Solve L_1 by reducing L_1 to L_2 ($L_1 \le L_2$) via function f:
  1. Convert input x of L_1 to instance $f(x)$ of L_2
     - ⋆ $x \in L_1 \iff f(x) \in L_2$
  2. Apply the algorithm for L_2 to $f(x)$
- Running time = time to compute f + time to apply algorithm for L_2
- We write $L_1 \le_P L_2$ if $f(x)$ is computable in polynomial time

Reductions show easiness/hardness

- To show L_1 is easy, reduce it to something we know is easy
  - $L_1 \leq easy$
  - Use algorithm for easy language to decide L_1
- To show L_1 is hard, reduce something we know is hard to it (e.g., NP-complete problem)
  - $hard \leq L_1$
  - If L_1 was easy, *hard* would be easy too

NP-Complete Problems

We say L is NP-complete if

1. L is in $\mathcal{NP}$
2. For every language L' in $\mathcal{NP}$, there is a polynomial time reduction of L' to L (i.e., $L' \leq_P L$)

The Boolean Satisfiability Problem

Boolean formulas:

$$\begin{array}{rcl} f & \rightarrow & T \mid F \\ & \mid & \neg f \\ & \mid & f \wedge f \\ & \mid & f \vee f \\ & \mid & f \implies f \end{array}$$

Example:

$$x \wedge \neg(y \vee z)$$

The satisfiability problem (SAT):

Given a boolean expression, is it satisfiable?

Theorem (Cook)

SAT is NP-complete.

Summary

Problems:

- Undecidable
- Decidable
  - Tractable ($\mathcal{P}$)
  - Intractable