Lec 1. Computational Complexity Crash Course

#project-hardness

Changki Yun (TAMREF)
tamref.yun@snu.ac.kr

Seoul National University

September 20, 2022

Course info

- Course page: `https://github.com/koosaga/project-hardness`
- Hosts: Aeren, ainta, Karuna, koosaga, leejseo, TAMREF
- Book: `https://hardness.mit.edu/`
- Time: Tue 22:00-24:00+ KST

Roadmap of the course

I’ve managed to guess some main questions of this course.

- How we deduce **NP-hardness** of a problem?
- How hard is **SAT**? what does it imply?
- Among all the NP-hard problems, which are **tractible** with some parameters fixed?
- Which problems are even hard to **approximate**, finding efficient approximation is **NP-hard**?
- Which problems are strongly believed to have **polynomial lower bound**?

Roadmap of the course

There are some **optional** topics in this course.

- **Unique Game Conjecture**: I didn't make enough research for this topic to discuss importance.
- How can we classify the complexities of **Counting Problems**?
- We have too little memory to store all the input, how can we solve the problem by **online** or **streaming** setup?

Roadmap of the course

There are some questions **out of** this course.

- Does a problem have a **polynomial memory** solution?
- Can we adopt a **random algorithm** to find a solution with certain probability?

Content

1. Basic Complexities (P, NP and NP-hardness)
2. Randomized Complexities
3. Higher Complexities

Today is the “Crash course” for complexity theory, to settle basic notions and concepts.
I tried to briefly introduce a variety of topics.

Definition of $\mathbb{P}$

A *decision problem* is a set membership problem. It can be formulated into "Given a string x, Does it belong to a set A?"

$\mathbb{P}$ and $\mathbb{FP}$

- $\mathbb{P}$ is the class of decision problems that can be solved in time bounded by a polynomial p of input size $|x|$.
  - in short, in "polynomial time".
- $\mathbb{FP}$ is the class of functions that can be computed in polynomial time.

In this sense, we just identify a problem to the underlying set.

Polynomial time, by which machine?

- Standard model of computation theory is the **Turing Machine**, using tape/head/state/action table.
  - We won't define it precisely due to its complexity.
- **Word-RAM** model is somewhat more familiar, consists of a sequence of fixed-size bit/arithmetic operations and random accesses, assumed to run in constant time.
- Or, we can just simply assume any fine Programming Laguages as C++, or Python.

Existence of polynomial time algorithm for all models above are in equivalency.

Turing Machine as the standard

- **Church-Turing Thesis** insists that *anything that can be computed at all* is computable in TMs.
- **Extended Church-Turing Thesis** claims that *anything that can be computed in polynomial time* is in $\mathbb{P}$.
  - Some recent results, such as *Shor's algorithm* are currently inconsistent of the thesis.

Reductions

Reduction is the simplest way to compare the "hardness" of problems.

Reductions

For a decision problem A, B, we say $A \leq_p B$ if there is a function $f \in \mathbb{FP}$ such that $x \in A \iff f(x) \in B$.

- Aliases: "A reduces to B". or "There's a reduction from A to B".

Indeed, $\leq_p$ is transitive. i.e. $A \leq_p B, B \leq_p C$ leads to $A \leq_p C$.

Types of reduction

- (Polynomial-time) **Karp reduction** from A to B is a type of reduction that we learned above, requiring $f \in \mathbb{FP}$ such that $x \in A \iff f(x) \in B$. Note that $f(x)$ must be in B.
- (Polynomial-time) **Cook reduction** from A to B is a reduction, being a polynomial-time algorithm requiring polynomial B-oracles. It generalizes Karp reduction.

Mostly we deal with only Karp reductions.

$\mathbb{NP}$

To prove the "hardness" of a problem B, we may take an "easier" problem $A \notin \mathbb{P}$ and prove that $A \leq_p B$. And we all know that having a good problem out of $\mathbb{P}$ is hard.

$\mathbb{NP}$

A decision problem A is in $\mathbb{NP}$, if there is a problem $B \in \mathbb{P}$ such that

$$A = \{x \mid \exists^p y \text{ s.t. } (x, y) \in B\}$$

Here $\exists^p y$ denotes that "there is a polynomial-sized y", and $\forall^p y$ is analogously defined.
y is called **witness**. It's hard to find unless $A \in \mathbb{P}$.

List of $\mathbb{NP}$ problems

- Indeed $\mathbb{P} \subseteq \mathbb{NP}$, taking $B = A \times \{0\}$.
- 4-colorability of graph is an $\mathbb{NP}$-problem. Given a graph G with n vertices, a 4-coloring of G can be its witness.
  - The coloring requires at most $2n$ bits.
  - Validating a given 4-coloring requires $O(n^2)$ time.
- **SAT** is in $\mathbb{NP}$, indeed.
- **HAM CYCLE** (Hamiltonian Cycle) problem is in $\mathbb{NP}$.
- **FACTORING** is in $\mathbb{NP}$.
  - Given a positive integer N, is there any integer $1 < a < N$ such that N/a is also an integer?

$\mathrm{co}-\mathbb{NP}$

Quickly introducing its dual to prevent misconception.

$\mathrm{co}-\mathbb{NP}$

A problem A is in $\mathrm{co}-\mathbb{NP}$ if there is a problem $B \in \mathbb{P}$ such that

$$A = \{x \mid \forall^p y \text{ s.t. } (x, y) \in B\}$$

It seems tricky, but it states the existence of (anti-)witness y, that falsifies $x \in A$.

$\mathrm{co}-\mathbb{NP}$ problems

- Indeed, $\mathbb{P} \subseteq \mathrm{co}-\mathbb{NP}$.
- It is believed that $\mathbb{NP} \neq \mathrm{co}-\mathbb{NP}$. In other words, no $\mathbb{NP}-\mathrm{complete}$ problems are found to be in $\mathrm{co}-\mathbb{NP}$. Also, no $\mathrm{co}-\mathbb{NP}-\mathrm{complete}$ problems are verified to be in $\mathbb{NP}$.
- Surprisingly, **FACTORING** is in $\mathrm{co}-\mathbb{NP}$, as **PRIMES** is in $\mathbb{P}$. (AKS primality test)
    - Hence **FACTORING** $\in \mathbb{NP} \cap \mathrm{co}-\mathbb{NP}$, implying that we don't know if **FACTORING** is $\mathbb{NP}-\mathrm{complete}$.

$\mathbb{NP}$−hard **and** $\mathbb{NP}$−complete

We omitted the definition of $\mathbb{NP}$−complete.

$\mathbb{NP}$−hard

A problem A is $\mathbb{NP}$−hard if $B \leq_p A$ holds for all $B \in \mathbb{NP}$.
Also, A is $\mathbb{NP}$−complete if $A \in \mathbb{NP} \cap \mathbb{NP}$−hard.

SAT is $\mathbb{NP}$−complete, thus **SAT** $\in \mathbb{P}$ implies that $\mathbb{P} = \mathbb{NP}$. We will make a further research on **SAT** in the following lecture.

Strongly $\mathbb{NP}$–complete problems

The **SUBSET SUM** problem is $\mathbb{NP}$-complete.

SUBSET SUM

- Input: $a_1, \cdots, a_n, B$
  - Input size: $\lg a_1 + \cdots + \lg a_n + \lg B$
- Is there any subset $S \subseteq \{1, \cdots, n\}$ with size $\leq k$, giving $\sum_{i \in S} a_i = B$?

However, if inputs are given **unary** instead of binary, (i.e. input size is $a_1 + \cdots + a_n + B$) **SUBSET SUM** is in $\mathbb{P}$ by dynamic programming.

Strongly $\mathbb{NP}$–complete problems

Strongly $\mathbb{NP}$–complete problems

Given a problem A,

- A is **(weakly)** $\mathbb{NP}$–complete if it's NP-complete with binary input.
- A is **strongly** $\mathbb{NP}$–complete if it's NP-complete with even unary input.

Indeed, same definition goes for $\mathbb{NP}$–hard problems. The problem **TSP** is strongly $\mathbb{NP}$–complete.

Strong $\mathbb{NP}$–complete problems are studied in Ch. 6 of the book, which is out of our scope.

Intermediate problems

Most common problems are in dichotomy: $\mathbb{P}$ or $\mathbb{NP}-\text{complete}$. Are there problems in $\mathbb{NP}$ but neither in $\mathbb{P}$ nor in $\mathbb{NP}-\text{complete}$?

$\mathbb{NP}-\text{intermediate}$

A is $\mathbb{NP}-\text{intermediate}$ if $A \in \mathbb{NP} \backslash (\mathbb{P} \cup \mathbb{NP}-\text{complete})$.

And they are very likely to exist.

Theorem. (Ladner 1975)

If $\mathbb{P} \neq \mathbb{NP}$, there exists an $\mathbb{NP}-\text{intermediate}$ problem.

Candidates of the intermediate problems

Most of them are currently working as "one-way functions" in cryptography.

- **FACTORING**: no polynomial algorithm for factoring. The best is $2^{\widetilde{O}(n^{1/3})}$ for n-digit numbers.
- **DLP**: Given a prime p and generator g and $a \in \mathbb{Z}_p$ and upper bound U, is there an integer $0 \le x \le U$ such that $g^x = a$?
    - Both **FACTORING** and **DLP** are in $\mathbb{NP} \cap \mathrm{co}{-}\mathbb{NP}$, thus they are believed to be intermediates.

Candidates of the intermediate problems

Other intriguing examples follow.

- **GRAPH ISOMORPHISM**: Given a pair G, H of graphs, is there an isomorphism $\phi : V(G) \to V(H)$ such that $uv \in E(G) \iff \phi(u)\phi(v) \in E(G)$?
  - It's solvable in $2^{(\lg n)^{O(1)}}$ time, implying its $\mathbb{NP}\text{−complete}$ness gives **quasi-polynomial** solution for all $\mathbb{NP}\text{−complete}$ problems, violating some kinds of **ETH**.
  - If **GRAPH ISOMORPHISM** is NP-complete, $\Sigma_2 = \Pi_2$ follows. We'll define the class below and discuss its aftermath.
- **MINIMUM CIRCUIT PROBLEM**: Given a truth table of boolean function f, is there a circuit with $\leq N$ logic gates?
  - OK...

$\mathbb{RP}$

Class of problems coping with **Randomized Algorithm** is nothing to do with *hardness*, as they don't have complete problems.

Thus we simply introduce them, and estimate their positions in Complexity Hierarchy.

$\mathbb{RP}$

A problem A is in $\mathbb{RP}$ if there is a polynomial-time algorithm M mapping the input x into $\{\text{yes}, \text{no}\}$, such that

- $\Pr[M(x) = \text{yes} \mid x \in A] \geq \frac{1}{2}$.
- $\Pr[M(x) = \text{no} \mid x \notin A] = 1$.

Thus, yes of M is absolutely true.

$\mathrm{co}{-}\mathbb{RP}$ is analogously defined.

$\mathbb{RP}$-problems

- **PRIMES**(until 2002) were known to be in $\mathbb{RP}$ by Miller-Rabin algorithm (1980).
- **POLYNOMIAL ZERO TESTING**: Given $f \in \mathbb{Z}[x_1, \cdots, x_n]$ and a prime p, is f identically zero in $\mathbb{Z}_p$?

Note that the constant $1/2$ can be replaced to any $0 < \alpha < 1$, giving a few independent runs of M.

$\mathbb{ZPP}$

$\mathbb{ZPP}$ are the class adopting a polynomial-time Las-Vegas algorithm. Equivalently,

$\mathbb{ZPP}$

A problem A is in $\mathbb{ZPP}$ if there is a polynomial-time algorithm M mapping the input x into $\{\text{yes}, \text{no}, \text{idk}\}$, such that

- $\Pr[M(x) = \text{idk}] \leq \frac{1}{2}$.
- M is always correct when it outputs yes or no.

It is known that $\mathbb{ZPP} = \mathbb{RP} \cap \text{co}-\mathbb{RP}$, being a rare example of precise agreement of some complexity classes.

It is believed that $\mathbb{P} = \mathbb{ZPP} = \mathbb{RP}$.

$\mathbb{BPP}$

$\mathbb{BPP}$ relaxes both $\mathbb{RP}$ and $\mathrm{co}-\mathbb{RP}$ by allowing two-sided error.

$\mathbb{BPP}$

A problem A is in $\mathbb{BPP}$ if there is a polynomial-time algorithm M mapping the input x into $\{\text{yes}, \text{no}\}$, such that

- $\Pr[M(x) = \text{yes} \mid x \in A] \geq \frac{3}{4}$.
- $\Pr[M(x) = \text{no} \mid x \notin A] \geq \frac{3}{4}$.

We can reach the answer by a majority vote after running M several times.

$\mathbb{BPP}$

$\mathbb{BPP}$ relaxes both $\mathbb{RP}$ and $\mathrm{co}\!-\!\mathbb{RP}$ by allowing two-sided error.

$\mathbb{BPP}$

A problem A is in $\mathbb{BPP}$ if there is a polynomial-time algorithm M mapping the input x into $\{\text{yes}, \text{no}\}$, such that

- $\Pr[M(x) = \text{yes} \mid x \in A] \geq \frac{3}{4}$.
- $\Pr[M(x) = \text{no} \mid x \notin A] \geq \frac{3}{4}$.

There are no known problems in $\mathbb{BPP}$ beyond $\mathbb{RP}$ (or $\mathrm{co}\!-\!\mathbb{RP}$), but at least some facts are known.

- $\mathbb{BPP} \subseteq \Sigma_2 \cap \Pi_2 \subseteq \mathbb{EXPTIME}$. (Sipser-Lautemann theorem)
- $\mathbb{BPP} \subseteq \mathbb{PSPACE} \subseteq \mathbb{EXPTIME}$.

Higher complexities

Higher Complexities

A problem A is in:

- $\mathbb{EXPTIME}$ if it can be solved in $2^{n^{O(1)}}$ time.
- $\mathbb{PSPACE}$ if it can be solved using $n^{O(1)}$ memory.
- $\mathbb{NPSPACE}$ if there is a $B \in \mathbb{PSPACE}$ such that

$$A = \{x \mid \exists^p y \text{ s.t. } (x, y) \in B\}$$

- $\mathbb{EXPSPACE}$ if it can be solved using $2^{n^{O(1)}}$ memory.

Complexity Separation

Proper inclusions

- $\mathbb{P} \neq \mathbb{EXPTIME}$. Thus, if A is $\mathbb{EXPTIME}$–complete, $A \notin \mathbb{P}$.
- $\mathbb{PSPACE} \neq \mathbb{EXPSPACE}$.

Note that, $\mathbb{EXPTIME}$-hardness is defined under relation $\leq_p$. We won't employ a notation like $\leq_e$ or sth.

Other inclusions

- $\mathbb{NP} \subseteq \mathbb{PSPACE} \subseteq \mathbb{EXPTIME}$.

Oracle complexity

Although it's the last time to treat higher complexities, I decided to introduce the concept of **Oracle machine**.

Oracle machine

Given a complexity class $\mathcal{C}, \mathcal{D}$ and a problem B, the class $\mathcal{C}^B$ is defined to be the problems being in $\mathcal{C}$, assuming the "oracle" $x \in B$ as a constant-time operation.
Also, $\mathcal{C}^{\mathcal{D}} := \bigcup_{B \in \mathcal{D}} \mathcal{C}^B$.

If $\mathcal{C}$ is large enough to conduct a reduction to a $\mathcal{D}-\text{complete}$ problem X, then $\mathcal{C}^{\mathcal{D}} = \mathcal{C}^X$.
e.g. $\mathbb{P}^{\mathbb{NP}} = \mathbb{P}^{\mathbf{SAT}}$.

Polynomial Hierarchy

Polynomial Hierarchy is an infinite sequence of complexity classes.

Polynomial Hierarchy

- $\Sigma_0, \Pi_0, \Delta_0 := \mathbb{P}$.
- $\Sigma_{i+1} := \mathbb{NP}^{\Sigma_i}$.
- $\Pi_{i+1} := \text{co}-\mathbb{NP}^{\Pi_i}$.
- $\Delta_{i+1} := \mathbb{P}^{\Sigma_i}$.

This is neat, but hard to break down.

Polynomial Hierarchy

In fact, the alternative definition of PH can be established.

Alternative definition

- $\Sigma_0, \Pi_0 := \mathbb{P}$.
- $A \in \Sigma_{i+1} \iff \exists B \in \Pi_i$ s.t. $A = \{x \mid \exists^p y \text{ s.t. } (x, y) \in B\}$.
- $B \in \Pi_{i+1} \iff \exists A \in \Sigma_i$ s.t. $B = \{x \mid \forall^p y \text{ s.t. } (x, y) \in A\}$.

Inclusions between Polynomial Hierarchies

There is a set of interwoven inclusion relations between hierarchies.

$$
\begin{tikzcd}
\Sigma_3^{\mathrm{P}} & \vdots & \Pi_3^{\mathrm{P}} \\
 & \Delta_3^{\mathrm{P}} \arrow[ul] \arrow[ur] & \\
\Sigma_2^{\mathrm{P}} \arrow[uu] \arrow[ur] & & \Pi_2^{\mathrm{P}} \arrow[uu] \arrow[ul] \\
 & \mathrm{P}^{\mathrm{NP}} = \Delta_2^{\mathrm{P}} \arrow[ul] \arrow[ur] & \\
\mathrm{NP} = \Sigma_1^{\mathrm{P}} \arrow[uu] \arrow[ur] & & \Pi_1^{\mathrm{P}} = \mathrm{coNP} \arrow[uu] \arrow[ul] \\
 & \Delta_0^{\mathrm{P}} = \Sigma_0^{\mathrm{P}} = \mathrm{P} = \Pi_0^{\mathrm{P}} = \Delta_1^{\mathrm{P}} \arrow[ul] \arrow[ur] &
\end{tikzcd}
$$

Polynomial Hierarchy Collapse

Theorem. (PHC)

If $\Sigma_i = \Pi_i$, $\Sigma_i = \Sigma_j = \Pi_j$ for all $j \geq i$.

- Thus, $\mathbb{NP}-\text{completeness}$ of **GRAPH ISOMORPHISM** removes all complexity classes beyond Σ_2.

Higher order SAT

Σ_2 **SAT** is defined as below.

Σ_2-SAT

Given a boolean formula (CNF) $\phi(\vec{x}, \vec{y})$, is there an assignment $\vec{x^0} = (x_1, \cdots, x_n)$ such that $\phi(\vec{x^0}, \vec{y})$ is true regardless of $\vec{y}$?

Known that Σ_2-**SAT** is $\Sigma_2-\text{complete}$. Σ_i-**SAT** can be defined equivalently.

QBF

QBF is the most comprehensive form of Σ_k **SAT**.

QBF

Given a boolean formula $\phi(\vec{x}_1, \cdots, \vec{x}_k)$, answer the question:

$$\exists \vec{x}_1 \forall \vec{x}_2 \exists \cdots Q_k \vec{x}_k \text{ s.t. } \phi(x_1, \cdots, x_k) = \text{true?}$$

while Q_j is $\exists$ for j odd, otherwise $\forall$.

Hardness of QBF

QBF is $\mathbb{PSPACE}$-complete.

Beyond the $\mathbb{PSPACE}$?

Mostly, the notion $\mathbb{NPSPACE}$ is tedious.

Theorem. (Savitch)

For $f(n) \geq n$, $\mathbb{NPSPACE}(f(n)) \subseteq \mathbb{PSPACE}(f(n)^2)$. Thus, $\mathbb{NPSPACE} = \mathbb{PSPACE}$.
Here $\mathbb{NPSPACE}(f(n))$ denotes the $\mathbb{NPSPACE}$ problem allowing $f(n)$ extra space.

Sublinear $\mathbb{NPSPACE}$ problems

If $f(n) = \mathcal{O}(\log n)$, we define $\mathbb{NL} := \mathbb{NPSPACE}(f(n))$ as the problem could be solved in logarithmic extra r/w memory other than input, and the input is read-only.

Reachibility problem

Given a directed graph G and $s, t \in V(G)$, is there a path from s to t?

The problem above is $\mathbb{NL}-\text{complete}$. It is weakly believed $\mathbb{L} \neq \mathbb{NL}$.

Thank You for Your Attention!