CMSC 858M: Algorithmic Lower Bounds: Fun with Hardness Proofs
Fall 2020

Instructor: Mohammad T. Hajiaghayi
Scribe: Jacob Prinz

December 11, 2021

1 Computational Complexity Crash Course

2 Overview

A language A is a set of strings. The goal is to write a program, or algorithm, which given a string x, can determine if $\mathsf{x} \in \mathsf{A}$. And we want the algorithm to be fast.

3 Polynomial Time (P)

Definition 1 *A* problem *is a set of strings, and the goal of the problem is to find an algorithm which determines membership to this set.*

Definition 2 *P is the set of problems which can be solved with an algorithm whose runtime is bounded by a polynomial of the lenth of the input string, for some specific polynomial.*

In order to show that a problem is in P, all that one has to do is find a polynomial-time algorithm which solves the problem. But how can one show that an algorithm is not in P? (Well, nobody knows, but presumably) One would need a formal model of computation. Traditionally, the Turing Machine is used. However, four our purposes:

- Turing machines run for a specific number of steps. This is the time.
- Turing machines are general computers: any algorithm can be computed on them
- There is a polynomial time Java program for a given problem if and only if there is a polynomial time Turing Machine for a given program.

Scribe: Jacob Prinz
Lecture 1b **Date:** 09/01/2010

4 Reductions

A reduction is the idea that given two problems A and B if B can be solved quickly then A can be solved quickly.

Definition 3 $A \leq_p B$ *if and only if there is a polynomial time* f *so that* $x \in A \iff f(x) \in B$

We say "A reduces to B". Note that $\leq_p$ is an equivalence relation.
Also note that if $A \notin P$, and $A \leq_p B$, then $B \notin P$.

5 Nondeterministic Polynomial Time

Definition 4 (SAT) *Given a boolean formula, is there a way to set the variables so that it is true?*

Clearly there is a 2^n time algorithm, which is just to try every possibility. BUT! Note that if you already knew an assignment of the variables which worked, it would be very fast to verify it.

Definition 5 $A \in NP$ *if and only if there is a set* $B \in P$ *so that*

$$A = \{x : \exists y \text{ of length polynomial in } x \text{ so that } (x, y) \in B\}$$

The idea is that y is a proof of x.
Here are some examples of problems in NP:

- Given a graph G, and a number k, is there a set of k nodes which are all connected?
- Given a graph, is there a cycle which visits each edge exactly once?
- Given two numbers, do they share a factor?
- Given two graphs, are they isomorphic?
- SAT
- Given some subsets of $\{1, ..., n\}$, are there k of them which cover $\{1, ..., n\}$?
- Tetris: Given a tetris board, is there a sequence of moves possible to reach a given score?

Note that any problem in P is also in NP. There are some problems which computer scientists know are in NP and think arent in P, but have no idea how to prove it. The following formalizes this idea:

Definition 6 B *is NP-hard if for all* $A \in NP$, $A \leq_p B$.
B *is NP-complete if* $B \in NP$ *and* B *is NP-hard.*

It seems like it would be very difficult to show that a problem is NP-hard, because we don't know what all of the NP -problems are. However, we have found thousands of NP-hard problems. See Cook [**?**], Levin [**?**]. For a translation of Levin, see [**?**].

Theorem 1 *SAT is NP-complete*

Once we have that theorem, we dont need to think about turing machines. We an simply reduce SAT to another problem to show that it is NP hard. For example, Karp [**?**] showed that 21 natural problems are NP-complete.

5.1 P vs NP

It is unknown whether $\mathsf{P} = \mathsf{NP}$. However, Gasarch had polled theorists on their opinion. 88% believe that $\mathsf{P} \neq \mathsf{NP}$. Here are reasons why:

- Thousands of problems are known to be NP-complete. Nobody has found a polynomial solution for any of them.
- It feels like it would be true
- If $\mathsf{P} = \mathsf{NP}$, then why can't we find any of those polynomial time algorithms?

6 Fixed Parameter Tractability

Recall the clique problem CLIQ and the vertex cover problem VC from earlier. Both of these problems are NP-complete, and no polynomial time solution is known.

But, consider the following variations:

$$\mathrm{CLIQ}_k = \text{Does G have a clique of size k}$$

$$\mathrm{VC}_k = \text{DOeos G have a vertex cover of size k}$$

Both can be solved in time $O(n^k)$. For a fixed k, therefore, they are in P. This is called Fixed Parameter Tractable.

7 Exponential Time Hypothesis

It is strongly believed that NP-complete problems not only are not in P, but that they take exponential time. This is called the Exponential Time Hypothesis.

8 Strong NP-Complete

There is a question of whether number inputs to problems should be given in binary or unary. These can give different hardness results.

- A problem is weakly NP-hard if it is NP-hard when inputs are binary
- A problem strongly NP-hard if it is NP-hard when the inputs are unary.

9 Complexity of Functions

Normally when talking about complexity, one is referring to a set membership problem. However, sometime one talks about a function.

Definition 7 (FP) *FP is the set of functions which can be run by an algorithm in polynomial time.*

- LCLIQ is the problem of finding the largest clique in a graph. $\mathsf{LCLIQ} \in \mathsf{FP} \iff \mathsf{CLIQ} \in \mathsf{P}$.
- FINDCLIQ is the problem of finding a clique of size LCLIQ(G). $\mathsf{CLIQ} \in \mathsf{P} \iff \mathsf{FINDCLIQ} \in \mathsf{FP}$

So in general, thinking about P is as good as thinking about FP, although perhaps not always.

10 Classes Above NP : The polynomial heirarchy

Definition 8 (Minimum Formula (MINFML)) *Given a boolean formula, find the MINIMUM satisfying assignment in the sense of number of variables set to true.*

It turns out that MINFML is not in NP.

Definition 9 (Polynomial Hierarchy) $A \in \Sigma_1$ *if there is* $B \in \mathsf{P}$ *so that*

$$A = \{x : \exists y \text{ of length polynomial in } x \text{ so that } (x, y) \in B\}$$

$A \in \Pi_1$ *if there exists* $B \in \mathsf{P}$ *so that*

$$A = \{x : \forall y \text{ of length polynomial in } x \text{ so that } (x, y) \in B\}$$

$A \in \Sigma_2$ *if there is* $B \in \mathsf{P}$ *so that*

$$A = \{x : \exists y_1 . \forall y_2 \text{ of length polynomial in } x \text{ so that } (x, y_1, y_2) \in B\}$$

$A \in \Pi$ *if there is* $B \in \mathsf{P}$ *so that*

$$A = \{x : \forall y_1 . \exists y_2 \text{ of length polynomial in } x \text{ so that } (x, y_1, y_2) \in B\}$$

Here are some facts:

- $\Sigma_1 \subseteq \Sigma_2 \subseteq \Sigma_3 ...$
- $\Sigma_i \subseteq \Pi i + 1$
- $\Pi \subseteq \Pi \subseteq \Pi ...$
- $\Pi \subseteq \Pi i + 1$

It is believed that all of these are not equal.
Given any NP problem, one can make a Σ_2 problem. For example:

Definition 10 ($\Sigma_2 - SAT$) *Given a boolean formula with variables* $X \cup Y$*, does there exists an assignment of* X *such that for all assignments of* Y *the formula holds?*

$\Sigma_2 - SAT$ is Σ_2-complete.

11 Intermediary Problems

Most problems which are known to be in NP but not P are also known to be $NP - complete$. However, there are some problems which are not.

A famous example is the problem of factoring numbers.

Definition 11 (FACT) *Given* n *and* a*, is there a factor of* n *less than* a*?*

This problem is in NP. However, the problem is not known to be in P. Also, it is now known to be NP—complete. There are various algorithm for factoring which are somewhat faster than the naive algorithm, but none below exponential. For more about factoring, see Wagstaff's book [**?**].

There is no known polynomial time algorithm for graph isomorphism. It is also not known to be NP-complete. The fastest known algorithm, due to Babai [**?**], runs in $O(2^{(\log n)^c})$. It is believed that this is about the best that one can do. Boppana [**?**] proved that if Graph Isomorphism is NP-complete, then $\Sigma_2 = \Pi_2$. So we believe that it is not the case.

Ladner [**?**] showed the following theorem:

Theorem 2 *If* $P \neq NP$*, then there is some* $A \in NP$ *where* $A \notin P$*, and* A *is NOT NP-complete.*

12 Counting Problems

Definition 12 (#3SAT) *Given a 3CNF formula, how many satisfying assignments does it have?*

- Valiant [**?**] showed that #3SAT is in FP if computing the permanant of a matrix is in FP.
- Toda [**?**] showed that if A is in the polynomail hierarchy, then A can be reduced to #3SAT. So, #3SAT can solve everything in the entire polynomial hierarchy in polynomial time.

13 SPACE

Definition 13 (EXP, PSPACE, EXPSPACE)

- *EXP is the set of problems that can be solved in exponential time*
- *PSPACE is the set of problems solvable with polynomial memory*
- *EXPSPACE is the set of problems which can be solved with exponential space*

Theorem 3

- $\mathsf{NP} \subseteq \mathsf{EXP}$
- $\mathsf{NP} \subseteq \mathsf{PSPACE}$

Note that there is a diagonalization argument which can show that a program can be solved in a particular time class, and not lower ones. Therefore, $\mathsf{P} \neq \mathsf{EXP}$.

14 Decidability

R is the set of problems which can be solved by a program at all. See [**?**] on why R used to stand for recursive.

- Most problems encountered in discrete mathematics are decidable.
- The halting problem (Does this program halt?) is undecidable.
- Given a polynomial $p \in \mathbb{Z}[x_1, ..., x_n]$, is there a root? This is undecidable.

15 Extra Related Problems

Various puzzles have been shown to be NP-complete. The following are from a survey by Kendall, Parkes, and Spoerer [**?**], and are all NP-complete:

- (Yato and Seta, 2003): Given a Soduku instance, does it have any solutions?

- (Helmert, 2003) Solitaire. The exact decision problem is complicated.
- (Friedman, 2002) Corral puzzle is a puzzle game in a grid. Some squares have a number, and some are empty. The goal is to find a closed loop around all of the numbers, with the following property: for each number in the loop, the number of squares either horizontally or vertically next to it until the loop must be that number. The decision problem is if a given puzzle is solvable.
- (Robertson and Munro, 1978): Given $\mathfrak{n}$ cubes where each face has a color chosen from a set of $\mathfrak{n}$ colors, can the cubes be stacked so that each color appears exactly one time on each of the four sized of the tower?
- (Stuckman and Zhang) Mastermind is a classic board game. Given a set of guesses and the answers that the mastermind gave, is there a solution?
- (Kaye, 2000): Minesweeper is another classic game. Here is decision problem is not playing the game, but related to it. Given a board and an assignment of numbers to each square, can the mines be placed so that all of the numbers make sense?

References

[1] Graham Kendall, Andrew Parkes, Kristian Spoerer, *A Survey of NP-Complete Puzzles*. University of Nottingham.

[2] Erik Demaine, William Gasarch, Mohammad Hajiaghayi *Fun with Hardness: Algorithmmic Lower Bounds*. Not yet Published.

[3] Robert Soare. Why Turing's thesis is not a thesis. InTuring centenaryvolume. Cambridge University Press, 2013

[4] Stephen A. Cook. The complexity of theorem-proving procedures. InProceedings of the Third IEEE Symposium on the Foundations of Computer-Science, pages 151–158, 1971.

[5] Leonid Levin. Universal search problems.Problems of Information Transmission, 9:115–116, 1973.

[6] Boris Trakhenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms.Annals of the History of Computing, 6:384–400, 1984.

[7] Richard Karp. Reducibilities among combinatorial problems. In Ray-mond E. Miller and James W. Thatcher, editors,Complexity of computercomputations, pages 85–103. Plenum Press, 1972.

[8] William Gasarch. Complexity Theory Column 100: The P=NP poll.SIGACT News, 50(1):28–34, 2019.https://www.cs.umd.edu/users/gasarch/papers/poll3.pdf.

[9] Vaclav Chvatal. A greedy heuristic for the set-covering problem.Mathematics of Operations Research, 4:233–235, 1979.

[10] Irit Dinur and David Steurer. Analytical approach to parallel repetition.InProceedings of the 46th annual ACM Sympsosium on Theory of Com-puting, pages 624–633, 2013.https://dl.acm.org/doi/pdf/10.1145/2591796.2591884.

[11] L. J. Stockmeyer. The polynomial-time hierarchy.Theoretical Computer-Science, 3(1):1–22, October 1976.

[12] Samuel Wagstaff.The joy of factoring. AMS, Providence, 2013.

[13] Laszlo Babai. Graph isomorphism is in quasipolynomial time, 2016.https://arxiv.org/abs/1512.03547.

[14] Ravi Boppana, Johan Hastad, and Stathis Zachos. Does co-NP haveshort interactive proofs?Information Processing Letters, 25, 1987.

[15] Richard Ladner. On the structure of polynomial time reducibilities.Jour-nal of the Association of Computing Machinery, 22:155–171, 1975.

[16] Leslie G. Valiant. The complexity of computing the permanent.Theoretical Computer Science, 8:189–201, 1979.http://dx.doi.org/10.1016/0304-3975(79)90044-6.

[17] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy.SIAMJournal of Computing, 20(5):865–877, 1991.https://doi.org/10.1137/0220053.

[18] Robert Soare. Computability and recursion.Bulletin of Symbolic Logic,2(4), 1996.

[19] William Gasarch. Hilbert's 10th problem for fixeddandn, 2021.https://www.cs.umd.edu/users/gasarch/BLOGPAPERS/h10_final.pdf.