CS 221 Computational Complexity, Lecture 1 Jan 22, 2018

Lecture 1

Instructor: Madhu Sudan *Scribe: Yuna Joung*

1 Welcome to CS221

Contacts
Lecturer: Madhu Sudan (madhu@cs.harvard.edu)
TF: Preetum Nakkiran (preetum@cs.harvard.edu)

Todo

1. Sign up on Piazza
2. Enroll in course
3. Sign up for scribing
4. Look at PS1 (due Friday)

The course website can be found at http://madhu.seas.harvard.edu/courses/Spring2018
We will use the Arora and Barak textbook as a reference in the course.
Grading

1. 3-5 Problem sets [40%]
2. Scribe work [30%]
3. 1 Project with Partner [15%]
4. Participation [15%]

Goals of Computational Complexity

1. Identify important problems and related phenomena
2. Analyze resources needed to compare tradeoffs (i.e. space complexity and time complexity)

   The resources we will work with are nontraditional resources like nondeterminism, randomness, interaction, and quantum)
3. Compare with other problems

In this course, we will deal with solving interesting problems. What makes a problem interesting? Let's look at some classic problems in the field.

2 Interesting Problems

2.1 Problem 0: 3SAT

Input: 3 CNF Formula

$$\Phi = (\overline{X} = (X_1 ... X_n); \overline{C} = (C_1 ... C_m))$$

$$C_j = X_{i1} \vee \neg X_{i2} \vee X_{i3}$$

Output: Decide if there exists a satisfying assignment such that Φ is satisfied by assignment if every clause in Φ is satisfiable. Output YES if Φ is satisfiable and NO if Φ is not satisfiable.

2.2 Problem 1: SAT

Input: 3CNF formula Φ
Output: Output the number of satisfying assignments to Φ.

2.3 Problem 1.5: Approximate SAT

Input: Φ
Output: N such that N is less than or equal to the number of satisfiable assignments and Φ has less than or equal to 2N constraints.

2.4 Problem 2: minCNF

Given: 3CNF formula Φ
Output: 3CNF formula Ψ with the minimum number of clauses where $\Phi(x) = \Psi(x) \forall x$

This problem appears to be more difficult than problem 0 because solving problem 0 is pre-requisite to solving problem 2. In this course we will think about these kinds of problems.

2.5 Problem 3: Permanent

Input: Given an n x n matrix $A = [a_0]$
Output:

$$perm(A) \triangleq \sum_{\pi=[n]\to[n]} \prod_{i=1}^{n} a_{i\pi(i)}$$

where π is a one to one function (permutation).

Does this equation look familiar? This problem was created by removing the sign term from the formula for the determinant. Recall that the formula for calculating the determinant is as follows.

$$det(A) = \sum_{\pi=[n]\to[n]} (-1)^{sign(\pi)} \prod_{i=1}^{n} a_{i\pi(i)}$$

By dropping the sign term from the determinant formula, we get a significantly more difficult problem.

3 What makes a problem interesting?

1. Natural ("subjective")
2. Captures other interesting problems ("objective")

The goal of the field is to understand the exact amount of resources needed for a problem. There is a lot of work to be done here!

Consider the following questions:

1. Can SAT be solved in time $O(nlog^{10}n)$?

2. Can SAT be solved in space $O(log1.5n)$?

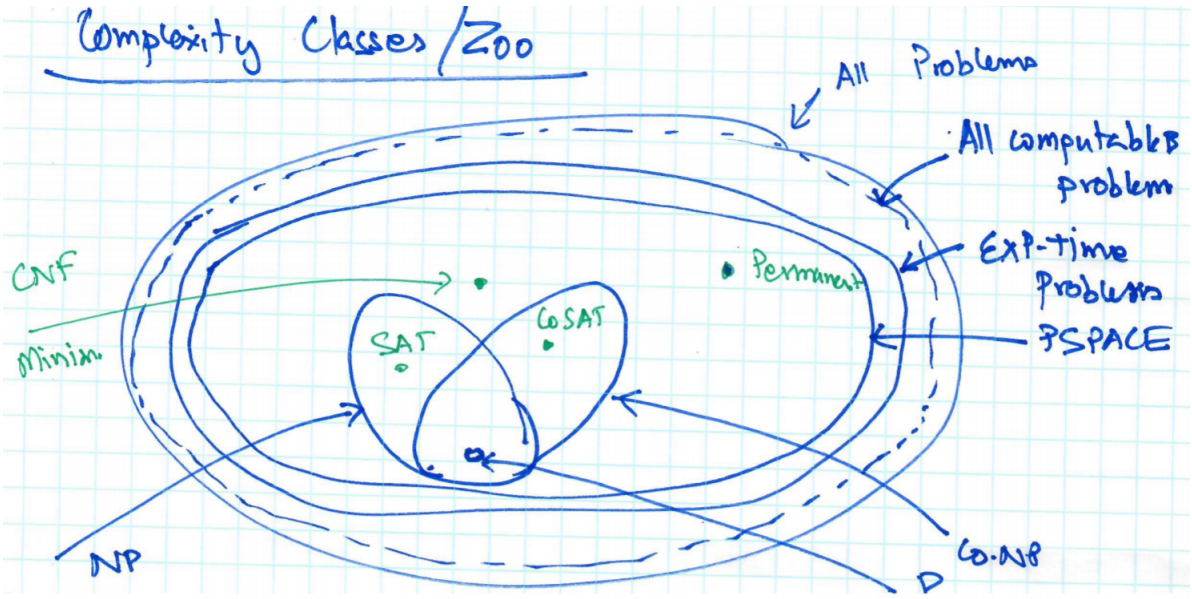
3. Can we multiply 2 n-bit numbers in O(n) time?

We don't know the answers to any of these questions. Even though we cannot answer these questions, we are able to reason about them in a comparative sense.

3.1 Karp

A seminal paper in this field was "Reducibility among combinatorial problems" by Karp '72. In this paper, Karp described 14 interesting problems along with 3SAT and used transitivity to show equivalence among these problems.

3.2 Complexity Zoo



The goal of this field is to separate problems based on resources. The state of the field is that we don't know anything! There are many conjectures and it is hard to prove lower bounds. However, we are able to compare problems to other problems quite well.

We may not know much about problem A and problem B inherently. However, we can still show that A is "no harder than" B. The phrase "no harder than" is colloquially replaced with the phrase "easier than". When we use our comparison tools to find many equivalent problems, it is usually time to define a new complexity class.

3.3 Basic Tools for Comparison

1. Languages ≡ Decision Problems ≡ Boolean functions

   Convert all problems into Decision Problems (Yes or No problems)

2. Reduce Problems to one another

   Many-many reductions / Turing reduction

   Many-one reduction / Karp reduction

3.3.1 Decision Problems

SAT = already defined = Φ such that there is a SAT assignment to Φ
coSAT = Φ such that No SAT assignment exists for Φ
Is SAT ≡ coSAT?

We can construct a decision problem that will help us answer this question as follows.
Answer 1: YES. Polynomial time algorithm for SAT can be complemented to get an algorithem for coSAT.
Answer 2: NO. Easy to "prove" Φ is SAT. However we do not yet have the tools to "prove" that Φ is in coSAT.