DD2445 COMPLEXITY THEORY: LECTURE 16 | I

Last two lectures

Interactive proofs

Adding interactivity + randomization to polynomial time $\Longrightarrow$ PSPACE

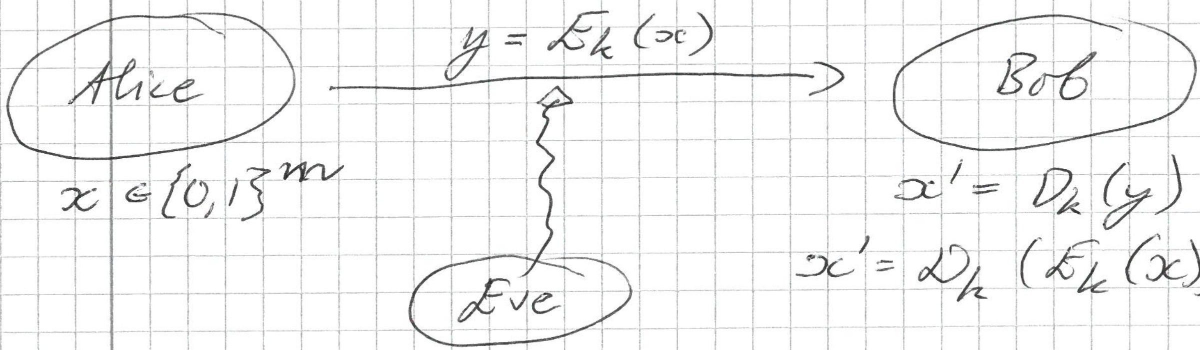
Probabilistically checkable proofs (PCPs)

$NP = PCP[O(\log n), O(1)]$

NP has polynomial size proofs that can be checked reading constant # bits (and with small risk of error)

Cryptography

key $k \in_R \{0,1\}^n$



$x \in \{0,1\}^m$

$x' = D_k(y)$

$x' = D_k(E_k(x)) = x$

Notion of secrecy: For two messages $x_1 \neq x_2$ Distributions $E_{U_n}(x_1)$ and $E_{U_n}(x_2)$ should "look the same" to Eve

For $n = m$: ONE-TIME PAD $y = x \oplus k$

Perfect secrecy

For $n < m$ perfect, information theoretic secrecy is impossible

Solution: Aim for secrecy against computationally bounded adversaries

DEF 17 $\varepsilon : \mathbb{N} \to [0,1]$ is NEGLIGIBLE if $\varepsilon(n) = n^{-\omega(1)}$ [decays faster than $1/p(n)$ for any polynomial $p(n)$]

DEF 18 $f : \{0,1\}^* \to \{0,1\}^*$ is a ONE-WAY FUNCTION if
a) f poly-time computable
b) f is hard to invert in the sense that $\forall$ probabilistic poly-time A $\exists$ negligible ε s.t.

$$\Pr_{\substack{x \in_R \{0,1\}^n \\ y = f(x)}} \left[A(1^n, y) \cancel{A(y)} = x' \text{ s.t. } f(x') = y \right] < \varepsilon(n)$$

CONJECTURE / ASSUMPTION 19 One-way functions exist.

CLAIM 20 If one-way functions exist, then $P \neq NP$.

CANDIDATE 21 Take $x \in \{0,1\}^n$ for $n = 2n'$
View x as concatenation of two n'-bit integers A and B
Let $f(x) = A \cdot B$ = n-bit number

Factoring number $N = A \cdot B$ can be done in time $\sqrt{N}$!
But this is exponential in size of input ($\log N$ bits)

Can set up this more carefully requiring $A = P$ and $B = Q$ to be prime numbers

Another candidate is the function used in the RSA crypto system

INFORMAL THEOREM 22
If one-way functions exist, then computationally secure encryption schemes exist

Proof idea Given truly random string $k \in \{0,1\}^n$, "stretch it out" to pseudorandom string k' in $\{0,1\}^m$

Then do one-time pad with this string k'

What does "pseudorandom" mean?

(1) KOLMOGOROV COMPLEXITY
"A string of length n is pseudo random if no TM whose description has size $< 0.99n$ outputs this string when started on blank tape"

"Right" definition, but (almost) totally useless

(2) STATISTICS
A string is pseudo random if it has expected # substrings of different types (0, 1, 00, 01, 10, 11, 000, etc) for truly random string. TOO WEAK definition!

(3) CRYPTOGRAPHY

Focus on distributions over strings

A distribution is pseudorandom if it is indistinguishable from truly uniformly random distribution for every probabilistic polynomial-time algorithm

DEF 23 A poly-time computable function $G: \{0,1\}^* \to \{0,1\}^*$ mapping n-bit strings to $\ell(n)$-bit strings is a $\boxed{\text{SECURE PSEUDORANDOM GENERATOR}}$ OF STRETCH $\ell(n)$ if $\forall$ probabilistic poly-time A $\exists$ negligible $\varepsilon(n)$ s.t. $\forall (n) \in \mathbb{N}^+$

$$\Big| \Pr\big[A(G(U_n)) = 1 \big] - \Pr\big[A(U_{\ell(n)}) = 1 \big] \Big| < \varepsilon(n)$$

A gets pseudorandom bits   A gets truly random bits

can't tell difference

THM 24 [Håstad, Impagliazzo, Luby, Levin '99]
If one-way functions exist, then secure pseudorandom generators exist for any polynomial stretch

Lots of nice math in here...

Read Secs 9.2 – 9.3 in Arora-Barak if you are interested (and then take our foundations of crypto course!)

We won't talk any more about this in this course (which is a computational complexity course after all...)

For the rest of this lecture, let's switch focus back to interactive proofs (but with a crypto perspective)

ZERO-KNOWLEDGE PROOFS

In interactive proofs so far "we" verifier want to benefit from, but not be fooled by, powerful but dangerous prover

Now let's switch the tables
Suppose "we" are the prover
Want to convince verifier we can prove $x \in L$
(e.g., F is satisfiable CNF)

But without leaking any more information
(don't reveal anything about satisfying assignment to F)

Concrete example

We want to authenticate by proving to verifier that we know secret password
But don't want to reveal anything about that password

Sounds a bit like science fiction...
But, amazingly, this can be done

DEF 25 ZERO-KNOWLEDGE PROOF [Arora-Barak style] VI

Let $L \in NP$; M verifier of (poly-size) certificates for $x \in L$

A pair (P, V) of interactive, probabilistic, poly-time algorithms form a ZERO-KNOWLEDGE PROOF for L if following 3 conditions hold:

(1) COMPLETENESS $\forall x \in L$ and u s.t. $M(x,u) = 1$

$$\Pr\left[out_V \langle P(x,u), V(x) \rangle = 1 \right] \geq 2/3$$

(2) SOUNDNESS $\forall x \notin L \quad \forall u \quad \forall P^*$ (all-powerful)

$$\Pr\left[out_V \langle P^*(x,u), V(x) \rangle = 1 \right] \leq 1/3$$

(3) (PERFECT) ZERO KNOWLEDGE $\forall$ probabilistic, interactive poly-time V^* $\exists$ algorithm S^* (probabilistic) such that

a) S^* runs in expected polynomial time

b) $\forall x \in L \quad \forall u$ s.t. $M(x,u) = 1$

$$out_{V^*} \langle P(x,u), V^*(x) \rangle \equiv S^*(x) \quad (\dagger)$$

That is, random variables in $(\dagger)$ identically distributed although S^* doesn't get access to u and doesn't even interact with P

Formalizes meaning of "learning nothing"
Since transcript can be reproduced without even talking to prover, conversation can't be leaking info

This should hold even for cheating verifier V^* who doesn't follow protocol [Verifier who follows protocol is "honest but curious"]

Flavours of zero knowledge:

1. PERFECT Simulator S^* gets transcript distribution exactly right

2. STATISTICAL Transcript distributions (extremely) close in statistical distance

3. COMPUTATIONAL Transcript distributions are computationally indistinguishable

Guarantees stronger in 1 than in 2 than in 3

Potentially more languages have zero-knowledge (ZK) protocols as in 3 than in 2 than in 1

THM 26 [Goldreich, Micali, Wigderson '86] If one-way functions exist, then every $L \in NP$ has a computational ZK proof.

Simulation Central concept in cryptography
Important in security definitions
Attacker can't learn or do anything that isn't also possible in "sandbox" (which is obviously secure)

EXAMPLE 27 Just to see a ZK protocol, let's look at

GRAPH ISOMORPHISM (GI) =

$= \{ \langle G_0, G_1 \rangle \mid G_0 \cong G_1 \}$

Public input G_0, G_1 graphs on n vertices

Prover's private input Permutation $\pi : [n] \to [n]$ s.t. $G_1 = \pi(G_0)$

Protocol

1. Prover: Choose random permutation $\pi_1 : [n] \to [n]$
   Send $H = \pi_1(G_1)$ to verifier
2. Verifier: Choose $b \in_R \{0,1\}$; send to prover
3. Prover: If $b = 1$, send π_1
   If $b = 0$, send $\pi_1 \circ \pi$
   Let sent permutation be π_H
4. Verifier: Accept if $H = \pi_H(G_b)$;
   reject otherwise

Analysis

Completeness If $G_0 \cong G_1$ as witnessed by π, and if P and V follow protocol, then V accepts with probability 1.

Soundness If $G_0 \not\cong G_1$, then $\exists b \in \{0,1\}$ s.t. $H \not\cong G_b$. Verifier chooses this b with probability $1/2$, in which case prover fails the test in step 4 and verifier rejects

(Repeat protocol twice to get $1/4 < 1/3$)

<u>Zero knowledge</u> Consider a fixed (misbehaving) verifier V^* and construct simulator S^* as follows:

(i) Choose $b' \in_R \{0,1\}$ and random $\pi : [n] \to [n]$
Compute $H = \pi(G_{b'})$ and feed into V^*

<u>Important</u> Note that we can run V^* as a subroutine. It is just a piece of code (and does not have a state or memory or anything)

(ii) Let $b \in \{0,1\}$ be message from V^* [and terminate protocol if V^* doesn't send a bit]

(iii) If $b \neq b'$ then <u>abort attempt</u> at generating transcript and <u>restart</u> from (i)

<u>Important</u> Note that this reboots V^* – since V^* is just a probabilistic TM it will happily run again without "knowing" that it has been restarted

else $b = b'$
Feed π into V^*
Let final transcript be
"H, b, π, V^*'s final output"

Need to argue

(a) Distribution of generated transcript is correct

(b) S^* runs in expected polynomial time

(a) Suppose S^* terminates. Then

- H is random permutation of G_1
- b is response from V^* generated based on message with correct distribution
- π is correct (as determined by H and b)

Since distributions of all messages so far is the same as in real conversation between V^* and P, the final message of V^* will also have the correct distribution

So the transcripts are identically distributed

(b) V^* runs in poly time, so every attempt takes poly time for S^*

Expected # attempts before S^* succeeds

$$\sum_{i=1}^{\infty} i \cdot Pr[\, i \text{ attempts needed} \,] =$$

$$\sum_{i=1}^{\infty} i/2^i = O(1)$$

So S^* runs in expected polynomial time.

This concludes the analysis.

With this we wrap up the "compulsory" (roughly) first half of the course

We have now covered most of what you "should know" about computational complexity theory after an introductory course

This should hopefully mean that:

- We can credibly claim to have given you a reasonably well-rounded (computational complexity) education
- You have a fighting chance of being able to read and understand papers that appear in, say, the Computational Complexity Conference (CCC) [which is useful if you want grade A]

But of course we haven't covered everything. Perhaps the <u>one major omission</u> is

<u>QUANTUM COMPUTATION</u>

2nd, "advanced part" of course

Selection (by necessity biased) of some more specialized topics
[but not necessarily technically harder]

Details decided as we go along

Probably:

- Communication complexity [already happened]
- Circuit complexity [lots of news in last few years, although we will look at classic results]
- Proof complexity [My area + nice connections to both circuit complexity and communication complexity]
- And/or maybe some other stuff...