DD2445 COMPLEXITY THEORY

L3 I

LECTURE 3

Last time:

Polynomial-time computation $\boxed{P}$

Efficient computation and Church-Turing thesis

Reductions
- solve problems
- relate hardness of problems

Polynomial-time verifiability $\boxed{NP}$

Nondeterministic Turing machines

Gave examples of problems in NP
Mentioned that some of them are "the hardest":

NP-complete

Let us define what that means

<u>DEF 6</u> $\mathcal{L} \subseteq \{0,1\}^*$ is <u>polynomial-time Karp reducible</u> to $\mathcal{L}' \subseteq \{0,1\}^*$, denoted $\mathcal{L} \leq_p \mathcal{L}'$, if $\exists$ poly-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ s.t. $\forall x$ $\quad x \in \mathcal{L}$ iff $f(x) \in \mathcal{L}'$

<u>DEF 7</u> $\mathcal{L}'$ is <u>NP-hard</u> if $\mathcal{L} \leq_p \mathcal{L}' \; \forall \mathcal{L} \in NP$

$\mathcal{L}'$ is <u>NP-complete</u> if in addition $\mathcal{L}' \in NP$

$\boxed{\text{$\mathcal{L}'$ is as hard as any problem in NP, since any algorithm efficient for $\mathcal{L}'$ can be used to decide any language in NP efficiently.}}$

<u>LEMMA 8</u>

1. If $\mathcal{L} \leq_p \mathcal{L}'$ and $\mathcal{L}' \leq_p \mathcal{L}''$ then $\mathcal{L} \leq_p \mathcal{L}''$ (TRANSITIVITY)
2. If $\mathcal{L}$ is NP-hard and $\mathcal{L} \in P$, then $P = NP$.
3. If $\mathcal{L}$ is NP-complete, then $\mathcal{L} \in P$ iff $P = NP$

<u>Proof</u> See textbook.

Sooo... Are there NP-complete problems? Yes, tons of them. In fact, all over the place in math, CS, physics, chemistry, biology, economics, industry... [But this course focuses on <u>theory</u> not applications.]

The most important (?) ones: variants of SAT

CNF formula (conjunctive normal form)

Variables x, y, z set to true $= 1$ or false $= 0$

Logical connectives AND $\wedge$, OR $\vee$, NOT $\neg$

Except for now negate only variables, written $\overline{x}$

Literal x or $\overline{x}$ True if $x=1$ or $x=0$

(Disjunctive) clause $C = x \vee \overline{y} \vee z$ denoted F or φ, usually

Satisfied ~~True~~ if one literal true

CNF formula conjunction of clauses

$$C_1 \wedge C_2 \wedge \cdots \wedge C_m = \bigwedge_{i=1}^{m} C_i$$

Satisfied if all clauses satisfied.

CNFSAT (just SAT in Arora-Barak)

Given CNF formula F, does it have a satisfying assignment?

Two variants

3-SAT: Each clause has size at most 3

E3-SAT: Each clause has size exactly 3

[Notation varies]

COOK-LEVIN THEOREM ('71 and '73, resp)

1. CNFSAT is NP-complete.
2. 3-SAT is NP-complete.

Satisfying assignment clearly easy to verify. Need to show every other $L \in NP$ reduces to CNFSAT.

What can we do?

- Only thing we know is that $\exists$ TM that verifies witnesses to claim $x \in L$ and accepts if correct.
- Write computation of such TM on x as ~~Bo~~ CNF formula
- Show that can plug in witness so that TM computation is satisfying.

PROP 9 Any Boolean function $f : \{0,1\}^\ell \to \{0,1\}$ can be expressed as a CNF of size $\ell \cdot 2^\ell$

(size = # connectives $\wedge/\vee$ or total # literals)

Look at all assignments α s.t. $f(\alpha) = 0$

Suppose $\alpha = \langle 1, 1, 0, 1 \rangle$

Write down clause ruling out this assignment $C_\alpha = \overline{x}_1 \vee \overline{x}_2 \vee x_3 \vee \overline{x}_4$

Then $F = \bigwedge_{\alpha \in f^{-1}(0)} C_\alpha$ represents f

(But is of exp size)

Want to construct reduction $x \to F_x$ s.t.

$$F_x \in SAT \iff \exists u \in \{0,1\}^{p(|x|)} \text{ s.t. } M(x,u) = 1$$

Apply Prop 9 ⇒ gives formula of size

$$p(|x|) \cdot 2^{p(|x|)}$$

Use that TM does <u>local</u> computation

Two simplifying (but justifiable) assumptions:

1. M has two tapes, input and work/output
2. M is <u>oblivious</u>: head movements don't depend on tape contents, only on input length $|x|$

At most quadratic loss in running time — see AB Ch 1

Can run M on x and $0^{p(|x|)}$ to determine head positions at every time step.

Q : set of states of TM (= lines in program)
Γ : alphabet (including 0, 1, ⊔ (blank) etc)

y = x and u concatenated

SNAPSHOT $z = \langle a, b, q \rangle \in \Gamma \times \Gamma \times Q$

a, b symbols read from tapes
q current state

z can be encoded as binary string of fixed length say c bits

Snapshot at time i depends on
(a) state at time $i-1$
(b) contents of current locations of tapes.

Suppose we're given sequence of snapshots

$z_1, z_2, z_3, \ldots, z_t$ claimed to be correct computation. How to verify?

To check z_i, only need to look at

1. z_{i-1} — did we jump to current state?
2. $y_{inputpos(i)}$ — The input tape head is at position inputpos(i), which we have computed — is the symbol in the snapshot consistent with this
3. $z_{prev(i)}$ — prev(i) was the last time current pos of work tape was visited. Looking at $z_{prev(i)}$ we can see what symbol was written to work tape then. Is it consistent).

1 - 3 uniquely determine z_i

So $\exists$ function $f(z_i, z_{i-1}, y_{inputpos(i)}, z_{prev(i)})$ that evaluates to true when transition correct

function of $c + c + 1 + c$ bits $=$ constant

Apply Prop 9 $\Rightarrow$ constant-size formula

L3 VII

Write down CNF formula F_x saying

1. First $|x|$ bits on input tape are x
2. z_1 encodes starting position of TM
3. For every $i > 1$, z_i is correct transition given $z_{i-1}, y_{inputpos(i)}, z_{prev(i)}$.
4. The last snapshot $z_{T(n)}$ is that of an accepting computation (1 is on the worktape).

Size of formula : $T(n) \cdot$ (exponential in c)

Can be computed in poly time
- First simulate $M(x, 0^{p(|x|)})$ to compute positions.
- Then output CNF encoding transitions for $T(n)$ steps and correct conditions at start & stop.

Reducing from CNFSAT to 3-SAT

Straightforward exercise — see textbook.

Two observations

1. Formula F_x (and time to compute it) can be made very small — $O(T \log T)$
2. From satisfying assignment to F_x, can read off witness u for x

Levin reduction

Now to prove NP-completeness of L, sufficient to reduce from CNFSAT / 3-SAT.

And then to prove L' NP-complete, reduce from CNFSAT, 3-SAT, or L. Etcetera...

(And, in fact, on the positive side, many practical problems can be solved by reducing to CNFSAT and then solved if you have a good SAT solver — more about that later).

Reductions are all over complexity theory and have proven to be an extremely useful tool (and in TCS in general).

THM 10 ~~LINEAR PROGRAM~~ 0/1-INTEGER PROGRAMMING is NP-complete.

Proof Easy to verify suggested solution.

Reduction: Translate clauses to lin ineq

$$x \rightarrow x$$

$$\overline{x} \rightarrow 1-x$$

$$x \vee \overline{y} \vee z \longrightarrow x + (1-y) + z \geq 1$$

This lin ineq satisfied by 0/1-assignment exactly when clause satisfied

23 IX

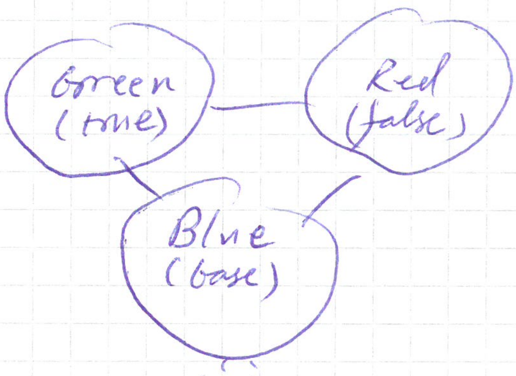
3- COLOURING

Graph $G=(V,E)$. Colour vertices red/blue/green so that if $(u,v)\in E$ then u and v have different colours.

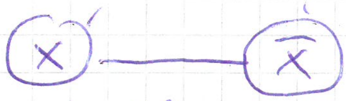
THEOREM 11
3-COLOURING is NP-complete

Proof Easy to verify a colouring.
Reduce from 3-SAT

Triangle

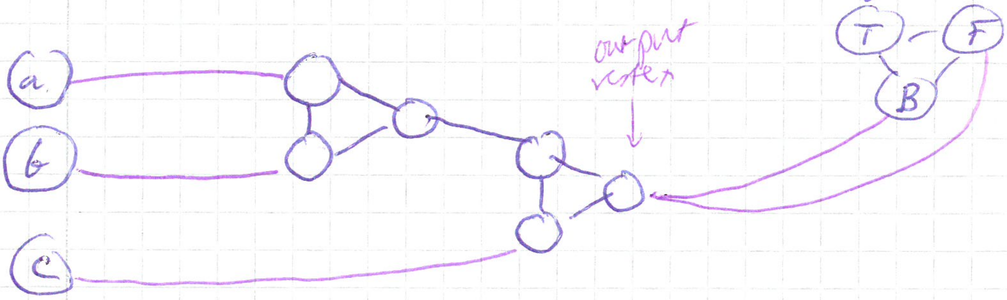


For each variable, x



connect both x and $\bar{x}$ to Blue (base)

For clause $a \vee b \vee c$ vertices a, b, c already exist.



Turns F into graph G_F L3 X

Need to prove

F SAT $\Longrightarrow$ G 3-colourable

G 3-colourable $\Rightarrow$ F SAT

($\Rightarrow$) Colour true literals green
Colour false literals red
Then output vertex of clause gadget can always be
coloured Green/True if
Clause contains true literal

($\Leftarrow$) Given a colouring, identify
T-colour with true
F- colour with false

We get same truth value assignment
since x and $\bar{x}$ have opposite values.

Every clause gadget has output vertex
coloured true

Only possible if some literal in clause true.

Decision vs search

THEOREM

Suppose P=NP. Then for every NP-language L with verifier M can find poly-time TM B that finds certificate for x (i.e., solves x).

Proof sketch:

Start with CNFSAT

Given $F = F(x_1, \ldots, x_n)$

Set $x=1$, simplify, check if $F\restriction_{x=1}$ satisfiable

If not check if $F\restriction_{x=0}$ satisfiable

Fix $x=b$ so that $F\restriction_{x=b}$ sat, and now move on to x_2

Will yield satisfying assignment

For general L, reduce $x \to F_x$, find sat assignment for F_x, read off a.

CNFSAT is <u>downward self-reducible</u>

Given an efficient algorithm that solves CNFSAT on input size $< n$, can solve CNFSAT instances of size n.

All NP-complete problems have similar property (follows from Cook–Levin)

For $L \subseteq \{0,1\}^*$ language, the <u>complement</u> of L is $\overline{L} = \{0,1\}^* \setminus L$

<u>DEF</u> $\quad coNP = \{ L \mid \overline{L} \in NP \}$

Aside: If strings in L encode objects such as e.g. graphs, then we usually think of $\overline{L}$ as only containing <u>correctly encoded</u> instances. I.e. L and $\overline{L}$ will both be sets of graphs satisfying and not satisfying same property respectively, while "garbage strings" are not contained in either L or $\overline{L}$.
Not an issue, really

coNP <u>not</u> the complement of NP

Intersection non-empty
E.g. $P \subseteq NP \cap coNP$ (why?)

<u>Example</u> CNFUNSAT $\in$ coNP

In fact, CNFUNSAT is coNP-complete — any other coNP-language is reducible to it. Given $L \in coNP$, run Cook-Levin on $\overline{L}$. $x \in L \Leftrightarrow x \notin \overline{L} \Leftrightarrow F_x \notin CNFSAT \Leftrightarrow F_x \in CNFUNSAT$

How is coNP related to NP?
Could there be short certificates that none of exponentially many assignments satisfy a formula?

Most researchers believe $NP \neq coNP$
We don't know, though.

Nondeterministic exponential time

<u>DEF</u> $NEXP = \bigcup_{c \in \mathbb{N}} NTIME(2^{n^c})$

clearly $P \subseteq NP \subseteq EXP \subseteq NEXP$

Why care about such classes?

THEOREM If $EXP \neq NEXP$, then $P \neq NP$

<u>Proof</u> Contrapositive: suppose $P = NP$, prove $EXP = NEXP$

Suppose $L \in NTIME(2^{n^c})$; NDTM M decides L.

Consider $L_{pad} = \{ \langle x, 1^{2^{|x|^c}} \rangle : x \in L \}$

claim
($L_{pad} \in NP$. First check that input is on correct form then run M. Since input exp large, this is poly time. By assumption $L_{pad} \in P$.
But then $L \in EXP$. Namely given x, pad it with ones and then check whether new string is in L_{pad}.

PADDING Useful technique to "scale up" or "scale down" results between weaker and stronger complexity classes

---

What does all of this mean?

Sec 2.7 in Arora-Barak highly recommended reading

Don't have time to discuss — our time is up

<u>TODAY</u> Focus on NP and NP-completeness

Should have seen most of this before

Though probably not presented in this way

<u>FUTURE LECTURES</u>

- New stuff (probably)
- Can we separate cplx classes ~~att~~ all? Is P distinct from EXP?
- Can we say anything about landscape between P and NP?