DD2445 COMPLEXITY THEORY: LECTURE 10

Last time: Polynomial-size Boolean circuits
P/poly

Believe that NP doesn't have poly-size circuits

Karp-Lipton theorem If $NP \subseteq P/poly$, then $PH = \Sigma_2^p$

P/poly also seems unlikely to contain EXP (regardless of what we believe about Karp-Lipton

Meyer's theorem If $EXP \subseteq P/poly$, then $EXP = \Sigma_2^p$

Some care needed for the proof...
Arora-Barak doesn't seem quite right.
Hopefully the notes for lecture 9 are better.

Corollary (of Meyer's theorem) If $P = NP$, then $EXP \not\subseteq P/poly$

Otherwise get contradiction to time hierarchy theorem

Upper bounds can yield lower bounds

Most functions are really hard

Theorem (Shannon)
A majority of functions $f:\{0,1\}^n \to \{0,1\}$ require circuits of size $\geq 2^n/(10n)$

Proof technique
- Probabilistic method
- Union bound $Pr[\bigcup_i A_i] \leq \sum_i Pr[A_i]$

<u>TODAY</u>

- Some subclasses of P/poly (that we will talk more about later in the course)

- Randomized computation (Turing machines that can flip coins)

Massively parallel computing

(idealized model)

- Lots of processors (say, n)
- Fast communication network (communication between any pair in $O(\log n)$ steps)
- Synchronized computation (global clock)
- Small amount of communication between each "clock tick" (operation on $O(\log n)$ bits, say)

Say that problem has <u>efficient parallel algorithm</u> if instances of size n can be solved on parallel computer with $n^{O(1)}$ processors in time $\log^{O(1)} n$

Recall: DEPTH of circuit = length of longest (directed) path from any input to output

0 is also OK

<u>DEF</u> For $d \in \mathbb{N}^+$, language L is in NC^d if decided by circuits $\{C_n\}_{n \in \mathbb{N}^+}$ of poly(n) size and depth $O(\log^d n)$

$$NC = \bigcup_{d \in \mathbb{N}^+} NC^d$$

For the next definition, relax requirements on AND- and OR-gates so that they can have arbitrary fan-in

DEF For $d \in \mathbb{N}^+$ (0 also OK), Language L is in AC^d if decided by circuits $\{C_n\}_{n \in \mathbb{N}^+}$ with unbounded fan-in AND-/OR-gates (and unary NOT-gates) of poly(n) size and depth $O(\log^d n)$.

NC^0 not so interesting

Constant depth – dependence on constant # bits

Already AC^0 interesting

Fan-in for AC^d at most poly(n) (why?)

So unbounded fan-in can be simulated in log depth

$$AC^0 \subseteq NC^1 \subseteq AC^1 \subseteq NC^2 \subseteq \ldots$$

This containment is known to be strict! (Yay!)
Will prove it later in the course

NC/AC III

THEOREM
Language L has efficient parallel algorithm iff $L \in NC$

Note Algorithm is uniform if circuit is uniform
Non-uniform circuit $\Rightarrow$ Algorithm with advice

Proof sketch:

($\Rightarrow$) Suppose N processors, time D

Build D layers of N subcircuits each
Circuit i in layer d does computation of processor i at time step d.
Communication network — circuit wires between subcircuits

($\Leftarrow$) Suppose $L \in NC$ decided by $\{C_n\}_{n \in \mathbb{N}^+}$. Let parallell computer read description of C_n.
Now let every processor take responsibility for simulating a gate
Send output to processors simulating gates that use this value

Is it possible for every problem in P to find an efficient parallel implementation?

In many cases: yes! (addition, multiplication, division, matrix determinant, matrix rank, matrix inverse, etc)

But always? Probably no.

What are the hardest problems in P?

<u>DEF</u> Language L is P-complete if

a) $L \in P$

b) $\forall L' \in P$ it holds that L' is logspace-reducible to L

If L P-complete and $L \in NC$, then $P = NC$

$$\text{CIRCUIT EVAL} = \left\{ \langle C, x \rangle \;\middle|\; \begin{array}{l} x \in \{0,1\}^n \\ C \text{ } n\text{-input circuit} \\ C(x) = 1 \end{array} \right\}$$

<u>THEOREM</u> CIRCUIT EVAL is P-complete.

RANDOMIZED COMPUTATION

- Randomness as a computational resource
- Lots of deep & fascinating questions here — see Ch 8 in Arora-Barak
- We'll get straight to the point: study Turing machines that can flip fair random coins

DEF 1 PROBABILISTIC TURING MACHINE (PTM)

Turing machine with two transition functions δ_0, δ_1

In each step, apply

δ_0 with probability $1/2$

δ_1 with probability $1/2$

Output of M on x $M(x)$ now random variable

PTM runs in time $T(n)$ if $\forall x$ halts in $\leq T(|x|)$ steps regardless of random choices

What should it mean that such a machine decides a language?

Compare to nondeterminism

- NDTM accepts if $\exists$ one (out of exponentially many) accepting branch
- PTM: look at fraction of accepting branches

For language $\mathcal{L} \subseteq \{0,1\}^*$ and $x \in \{0,1\}^*$, define

$$\mathcal{L}(x) = \begin{cases} 1 & \text{if } x \in \mathcal{L} \\ 0 & \text{if } x \notin \mathcal{L} \end{cases}$$

Our (main) model for efficient probabilistic/randomized computation:

$\boxed{\text{BPP}}$ bounded-error probabilistic polynomial time

<u>DEF 2</u> A PTM M decides $\mathcal{L}$ in time $T(n)$ if $\forall x$ M halts in $T(|x|)$ steps and $\Pr[M(x) = \mathcal{L}(x)] \geq 2/3$.

$\text{BPTIME}(T(n))$ = languages decided by PTMs in $O(T(n))$ time

$\text{BPP} = \bigcup_{c \in \mathbb{N}^+} \text{BPTIME}(n^c)$

Probability over <u>random choices</u>, not over input.

Constant 2/3 arbitrary (will see later)

Don't need perfectly fair coins (but we'll ignore this)

<u>PROP 3</u> $\mathcal{L} \in \text{BPP}$ if exist poly-time (deterministic) TM M and polynomial p s.t. for every x

$$\Pr_{r \in_R \{0,1\}^{p(|x|)}} \left[M(x,r) = \mathcal{L}(x) \right] \geq 2/3$$

<u>Notational aside</u>

Uniform sampling from $\{0,1\}^n$: $x \in_R \{0,1\}^n$

$x \sim \{0,1\}^n$

$x \sim \mathcal{U}_n$

<u>COR 4</u> $P \subseteq \text{BPP} \subseteq \text{EXP}$

<u>Proof</u> Can try all possible random strings in exponential time and compute success probability

Can't prove even $BPP \neq NEXP$

What about BPP vs P?

Fairly strong reasons to believe P = BPP!

[Discussed in Chs 19-20 in Arora-Barak — we probably won't have time to cover this.]

Example of the power of randomization

POLYNOMIAL IDENTITY TESTING

Given: polynomial (multivariate) with integer coeff. In implicit form

Decide: Is the polynomial identically zero?

Representation algebraic circuit

Like Boolean circuits, but gates are $+, -, \times$

Can also have constants $0, 1, \ldots$ if we wish

Inputs $x_1, \ldots, x_n$

Single output node (sink)

Not hard to see: computes some polynomial

ZEROP = { algebraic circuits corresponding to polynomials that are identically zero }

Why identity testing?

Given C, C', construct $D = C - C'$ and check if $D \in$ ZEROP.

Note compact representation

$\prod_{i=1}^{n} (1 + x_i)$ has 2^n terms

circuit of size $2n$

SCHWARTZ-ZIPPEL LEMMA

Let $p(x_1, x_2, \ldots, x_m)$ be non-zero poly of total degree $\leq d$. Let S finite set of integers. Then for $a_1, \ldots, a_m$ chosen from S uniformly randomly with replacements

$$\Pr\left[\, p(a_1, \ldots, a_m) \neq 0 \,\right] \geq 1 - \frac{d}{|S|}$$

<u>Proof</u> Induction over ~~m~~.

<u>Base case $m=1$</u>: Univariate polynomial

Degree $\leq d$ $\Rightarrow$ at most d roots

So p can evaluate to zero on at most d out of $|S|$ integers.

<u>Inductive step</u> See Arora-Barak App A.6

TESTING IDEA

Circuit of size m $\Rightarrow$ $\leq m$ multiplications

$\Rightarrow$ degree $\leq 2^m$

So pick $a_1, \ldots, a_m \in [1, 10 \cdot 2^m]$, evaluate circuit, and apply Schwartz-Zippel

If circuit C encodes zero poly $\Rightarrow$ result always 0

if non-zero poly $\Rightarrow$ 90% of non-zero output

<u>Problem</u> If degree $\approx 2^m$, then numbers grow as large as $(10 \cdot 2^m)^{2^m}$ $\Rightarrow$ exponentially many bits.

Hard to do in poly time...

<u>Solution</u> "fingerprinting" compute modulo $k \in [2^{2m}]$