CoCo: RANDOMIZED COMPUTATION

Most recently Boolean circuits
Polynomial size P/poly

Believe that NP cannot be decided by polynomial-size circuits

Circumstantial evidence: Prove that if NP ⊆ P/poly, then other unexpected things happen

KARP-LIPTON THEOREM
If NP ⊆ P/poly, then PH = Σ_2^P

"If NP has polynomial-size circuits, then the polynomial hierarchy collapses to the second level"

Regardless of what we believe about polynomial circuits for NP, as long as we believe P ≠ NP we should also believe that EXP cannot be decided by polynomial-size circuits

MEYER'S THEOREM
If EXP ⊆ P/poly, then EXP = Σ_2^P

(Some care is needed for the proof of this theorem, and the details in Arora-Barak don't seem quite right)

COROLLARY OF MEYER'S THEOREM
If $P = NP$, then $EXP \not\subseteq P/poly$

Proof If $P = NP$, then PH collapses
But then if $EXP \subseteq \Sigma_2^P$, this means that $EXP \subseteq P$. This we know is false by the time hierarchy theorem! □

So upper bounds can yield lower bounds.
Frequent theme in complexity theory.

Most functions have very high circuit complexity

THEOREM (SHANNON)
Almost all functions $f: \{0,1\}^n \to \{0,1\}$ require circuits of size $\Omega(2^n/n)$

Proof Count # functions
Count # circuits ▨

But very challenging to prove lower bounds for concrete functions

Now we move on to RANDOMIZED COMPUTATION
Turing machines that can flip coins

- RANDOMNESS is a computational resource, just like time & space
- Raises lots of deep and fascinating questions – read more in Chapter 8 in Arora – Barak
- We'll cut straight to the chase and define Turing machines that can flip fair random coins

DEF 1 PROBABILISTIC TURING MACHINE (PTM)

Standard Turing machine
Except 2 transition functions δ_0, δ_1
(just like NTM)
In each step, apply

δ_0 with probability $1/2$
δ_1 with probability $1/2$

Output $M(x)$ of M on x is now a RANDOM VARIABLE

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

What should it mean that a probabilistic Turing machine decides a language?

- Nondeterministic TM accepts if exists <u>one</u> accepting computational path (out of exponentially many)
- For probabilistic TM, look at <u>fraction</u> of accepting paths.

Helpful piece of notation:

For $L \subseteq \Sigma^*$ and $x \in \Sigma^*$, define

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

Our main way of modelling efficient randomized computation is bounded-error probabilistic polynomial time (BPP)

DEF 2 We say that a probabilistic Turing machine M decides a language L in time $T(n)$ if for all x M halts in $T(|x|)$ steps and

$$Pr[M(x) = L(x)] \geq 2/3$$

$BPTIME(T(n))$ = languages decided by probabilistic TMs in time $O(T(n))$

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

Some remarks

- Probability over random choices, not over inputs
- Constant 2/3 is arbitrary, essentially $1/2 + \varepsilon$ is fine (and yields same complexity class — we'll see this later)
- Don't need perfectly fair coins, but can use imperfect sources of randomness (we'll ignore this issue)

We could also think of probabilistic TM as having a special "random tape" filled with bits

PROP 3 $L \in BPP$ if exists polynomial-time (deterministic) TM $M(\cdot,\cdot)$ and polynomial p such that for every x

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

Notational aside To denote that r is sampled uniformly at random from $\{0,1\}^n$ can write

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

All these pieces of notation will mean the same for us

COR 4 $P \subseteq BPP \subseteq EXP$

Proof $P \subseteq BPP$ is clear — a deterministic TM is a probabilistic TM with $\delta_0 = \delta_1$.

For $BPP \subseteq EXP$, we can try all possible random strings in exponential time and compute success probability p_x for input x

Case analysis:

$p_x \geq 2/3 \Rightarrow x \in L$

$p_x \leq 1/3 \Rightarrow x \notin L$

$1/3 < p_x < 2/3 \Rightarrow$ error — TM is broken ▨

We cannot even prove $BPP \neq NEXP$

What about BPP vs. P?

Fairly strong reasons to believe $P = BPP$ ☺

[Discussed in Chapters 19-20 in Arora-Barak — we don't expect to have time to cover this]

But there are currently problems for which efficient algorithms known require randomness

POLYNOMIAL IDENTITY TESTING (PIT)

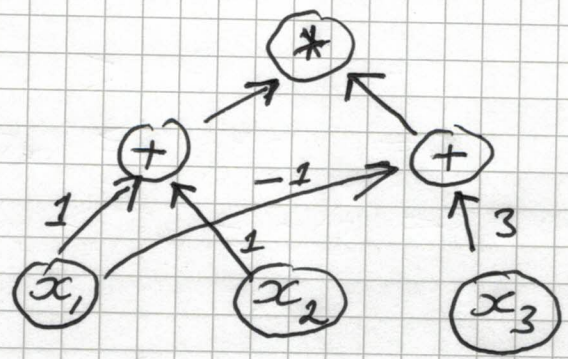
Input: Multivariate polynomial with integer coefficient

Output: Is the polynomial identically zero?

Polynomials represented as
ALGEBRAIC CIRCUITS

- Like Boolean circuits, but gates are + and *
- Can have integer multipliers on wires
- Inputs are variables $x_1, \ldots, x_n$ (or constants)
- Single output node (sink)

Ex



computes

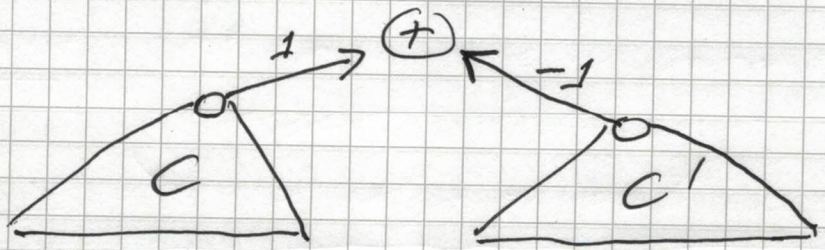
$$(x_1 + x_2)(3x_3 - x_1)$$

$$= -x_1^2 + 3x_1x_3 + 3x_2x_3 - x_1x_2$$

$$ZEROP = \{ \text{Well-formed algebraic circuits that expand to identically zero polynomial} \}$$

Why polynomial <u>identity</u> testing

If we want to know whether two polynomials are identical, then test whether their difference is 0



Note that algebraic circuits can yield very compact representation.

Polynomial $\prod_{i=1}^{n} (1 + x_i) =$

$$= \sum_{S \subseteq [n]} \prod_{i \in S} x_i$$

has

- 2^n terms
- algebraic circuit of size $O(n)$

Idea for randomized PIT algorithm

- Pick random input $(a_1, a_2, \ldots, a_n)$ $a_i \in \mathbb{Z}$
- Evaluate circuit on this input
- If result $\neq 0$, answer that definitely circuit $\notin$ ZEROP
- If result $= 0$, answer that probably circuit $\in$ ZEROP

Always correct if answer is "no"

But what about answers "yes"?

Phrased differently

If $C \in$ ZEROP, then guaranteed to get correct answer

If $C \notin$ ZEROP, then not so clear...

Useful fact Low-degree polynomials have few zeros

DEMILLO-LIPTON-SCHWARTZ-ZIPPEL LEMMA

Let $p(x_1, \ldots, x_n)$ be non-zero polynomial of total degree $\leq d$

Let S be finite set of integers

Then for $a_1, \ldots, a_n$ chosen uniformly and independently at random from S (with replacements), it holds that

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

Proof By induction over n.

Base case ($n=1$) A univariate polynomial of degree $\leq d$ has at most d distinct roots.

Induction step Write the polynomial p as

$$p(x_1, x_2, \ldots, x_{n-1}, x_n) = \sum_{i=0}^{d} x_n^i \cdot p_i(x_1, x_2, \ldots, x_{n-1})$$

p is non-zero $\Rightarrow$ some p_i is non-zero

Fix largest i^* such that $p_{i^*} \neq 0$

$\deg(p_{i^*}) \leq d - i^*$

By induction hypothesis

$$\Pr_{(a_1, \ldots, a_{n-1})}\left[p_{i^*}(a_1, \ldots, a_{n-1}) \neq 0\right] \geq 1 - \frac{d-i^*}{|S|}$$

When $p_{i^*}(a_1, \ldots, a_{n-1}) \neq 0$, we get that

$$p(a_1, \ldots, a_{n-1}, x_n) = \sum_{j=0}^{i^*} x_n^j \, p_j(a_1, \ldots, a_{n-1})$$

is a univariate polynomial of degree i^*

so $\Pr_{a_n}\left[\sum_{j=0}^{i^*} a_n \cdot p_j(a_1,\ldots,a_{n-1}) \neq 0\right] \geq 1 - \frac{i^*}{|S|}$

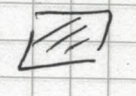
We have

$$\Pr[p \neq 0] = \Pr[p \neq 0 \mid p_{i^*} \neq 0] \cdot \Pr[p_{i^*} \neq 0] + \Pr[p \neq 0 \mid p_{i^*} = 0] \cdot \Pr[p_{i^*} = 0]$$

$$\geq \Pr[p \neq 0 \mid p_{i^*} \neq 0] \cdot \Pr[p_{i^*} \neq 0]$$

$$\geq \left(1 - \frac{i^*}{|S|}\right)\left(1 - \frac{d - i^*}{|S|}\right)$$

$$\geq 1 - \frac{d}{|S|}$$



Refined idea for randomized PIT algorithm

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

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

Let $S = \{1, 2, 3, \ldots, 10 \cdot 2^m\}$

Pick $a_i \in S$ randomly and evaluate circuit

By DLSZ lemma, if circuit encodes non-zero polynomial, then 90% chance of seeing non-zero output.

And if circuit encodes zero polynomial, then we will always see zero output

DONE ☺ Or are we?

Not quite... What is the problem?

Problem If degree $\approx 2^m$, then intermediate numbers can grow as large as $(10 \cdot 2^m)^{2^m} \Rightarrow$ exponentially many bits

Cannot compute with such numbers in polynomial time...

Solution "FINGERPRINTING"

Compute modulo some $k \in [2^{2m}]$

After each operation, divide by k and take remainder

Suppose polynomial computed by circuit $C(x_1, \ldots, x_n)$

Suppose $A = C(a_1, \ldots, a_n)$

If $A = 0$, then clearly $A \equiv 0 \pmod{k}$

CLAIM 5 If $A \neq 0$, then for a randomly chosen $k \in [2^{2m}]$ it holds that $k \nmid A$ with probability at least $\frac{1}{8m}$

Given this claim run test $K \cdot 8m$ times for suitably large constant K with different moduli k.

Suppose $A \neq 0$. Then test fails only if $k \mid A$

$$\Pr[k \mid A : A \neq 0] \leq 1 - \frac{1}{8m}$$

$$1 - x \leq e^{-x}$$

$$\Pr[\text{all } K \cdot 8m \text{ tests fail}] \leq$$

$$\left(1 - \frac{1}{8m}\right)^{K \cdot 8m} \leq$$

$$\left(e^{-\frac{1}{8m}}\right)^{K \cdot 8m} = e^{-K}$$

$$\Pr[\text{randomized PIT fails}] \leq$$

$$\Pr[A = 0] + \Pr[A \equiv 0 \bmod k_i \text{ for all } k_i]$$

$$\leq \frac{1}{10} + e^{-K} \leq \frac{1}{3}$$

for K chosen large enough. ∎

Proof of Claim 5

Assume $A \neq 0$ We know $A \leq (10 \cdot 2^m)^{2^m}$

Let B = prime factors of A

Sufficient to show that with probability $\geq \delta = \frac{1}{8m}$ k is a prime not in B

A has at most $\log A \leq 2^m \log(10 \cdot 2^m) \leq$

$\leq 5m \cdot 2^m$ prime factors

By Prime Number Theorem

$$\#\text{primes} \leq N \sim N / \ln N$$

Correct constant is actually 1, but see Thm A.23 in Arora-Barak for simpler version that is also sufficient

$$\#\text{primes} \le 2^{2m} \sim \frac{2^{2m}}{2m} > \frac{2^{2m}}{4m}$$

for large enough m

$$5m \cdot 2^m = o\left(\frac{2^{2m}}{2m}\right) < \frac{2^{2m}}{8m}$$

for large enough m.

So $\Pr[\text{k prime not in } B] \ge \dfrac{2^{2m}/8m}{2^{2m}}$

$$= \frac{1}{8m}$$ ▨

Many natural randomized algorithms have **ONE-SIDED ERROR**

Might make mistake when $x \in L$ but never when $x \notin L$, or other way round

(We just saw an algorithm that is always correct when $x \in L$, though not always when $x \notin L$)

DEF 6 RTIME($T(n)$) contains every language L for which exists probabilistic TM M running in time $O(T(n))$ such that

$$x \in L \Rightarrow \Pr[M(x)=1] \ge 2/3$$
$$x \notin L \Rightarrow \Pr[M(x)=0] = 1$$

$$RP = \bigcup_{c \in N^+} RTIME(n^c)$$

Never false positives Positive answers Right