LAST TIME
- Wrapped up circuit complexity (for now)
- Started talking about RANDOMIZED COMPUTATION

Probabilistic Turing machine (PTM)
- Two transition functions in every state
- Flips coin which one to take
- Output $M(x)$ random variable

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

$\mathbb{BPP}$ Class of languages for which $\exists$ PTM $M$ that always runs in polynomial time (regardless of coinflips) and $\forall x$

$$\Pr\left[M(x) = L(x)\right] \geq \frac{2}{3}$$

Probability not over input $x$

only over internal randomness of algorithm

$P \leq \mathbb{BPP} \leq \mathbb{EXP}$

Believe $P = \mathbb{BPP}$

But there are things we currently don't know how to do efficiently without randomness
POLYNOMIAL IDENTITY TESTING

Input: Algebraic circuit representing multivariate polynomial with integer coefficients

Task: Decide whether polynomial identically zero

\[(x_1 + x_2)(x_1 - 3x_3)\]
\[= x_1^2 - 3x_1x_3 + x_1x_2 - 3x_2x_3\]

\(\text{ZERO}_P = \{ \text{algebraic circuits computing 0} \} \)
\[\text{identically zero polynomials}\]

Idea for randomized algorithm

- Pick random input \((x_1, x_2, \ldots, x_n)\)
- Set \(x_i = x_j\) for \(i = 1, \ldots, n\) and evaluate circuit
- If result \(\neq 0\), declare circuit \(\notin \text{ZERO}_P\)
- If result = 0, say probably \(\in \text{ZERO}_P\)

Always correct on no-instances

But what about yes-instances?
SCHWARTZ-ZIPFER LEMMA

Let \( p(x_1, \ldots, x_m) \) non-zero polynomial of
total degree \( \leq d \), let \( S \) finite set of
integers. Then for \( a_2, \ldots, a_m \) chosen
from \( S \) uniformly randomly with replacements
it holds that

\[
P[r \mid p(a_1, \ldots, a_m) \neq 0] \geq 1 - \frac{d}{|S|}
\]

Proof

By induction over \( m \)

Base case \( m = 1 \): Univariate polynomial of
degree \( \leq d \Rightarrow \) at most \( d \) distinct roots.

Induction step: Write

\[
p(x_1, x_2, \ldots, x_m) = \sum_{i=0}^{d} x_1^i p_i(x_2, \ldots, x_m)
\]

\( p \) is non-zero \( \Rightarrow \) some \( p_i \) non-zero.
Pick largest \( i \) s. t. \( p_i \) non-zero.
By induction hypothesis

\[
P[r \mid p_i(a_2, \ldots, a_m) = 0] \geq 1 - \frac{d-i}{|S|}
\]

When \( p_i(a_2, \ldots, a_m) \neq 0 \) we get that

\[
p(x_1, a_2, \ldots, a_m) = \sum_{i=0}^{d} x_1^i p_i(a_2, \ldots, a_m)
\]

is a univariate polynomial of degree \( \leq i \),
and so = 0 for at most \( i \) values, so

\[
P[r \mid p(a_1, a_2, \ldots, a_m) \neq 0] \geq (1 - \frac{i}{|S|}) \left(1 - \frac{d-i}{|S|}\right) \geq 1 - \frac{d}{|S|}
\]
Refined testing idea

Circuit of size \( m \Rightarrow \leq m \) multiplications
\[ \Rightarrow \text{degree } \leq 2^m \]

Let \( S = \{ 1, 2, \ldots, 10 \cdot 2^m \} \)

Pick \( a \in S \) randomly & evaluate circuit

By Schwartz-Zippel, if circuit encodes
non-zero polynomial, then 90% chance of seeing non-zero output

If circuit encodes zero polynomial, then output always zero

Done? Not quite...

Problem: If degree \( \geq 2^m \), then
numbers grow as large as
\( (10 \cdot 2^m)^{2^m} \Rightarrow \) exponentially many CRTs

Hard to achieve in polynomial time...

Solution: "FINGERPRINTING"

Compute modulo some \( k \in \mathbb{Z}[2^m] \)
Computing modulo $k$
After each operation, divide by $k$
and take remainder

Suppose $y = C(a_1, \ldots, a_m)$

If $y = 0$, then $y = 0 \pmod{k}$

If $y \neq 0$, then randomly chosen $k \in [2^m]$ will not divide $y$ with prob $\geq \delta = \frac{1}{4m}$

Given this claim, run test $O(m)$ times
and accept only if always get 0 output

Proof of Claim 5
Assume $y \neq 0$. $y \leq (10 \cdot 2^m) \cdot 2^m$

Let $B$ = prime factors of $y$.
Sufficient to show that with prob $\geq \delta$ $k$ is a prime not in $B$

If $y$ has at most $\log y \leq 5m \cdot 2^m$ prime factors

By Prime Number Theorem, constant is actually $1$ (*)

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

$\# \text{ primes} \leq 2^{2m} \sim \frac{2^{2m}}{2m} > \frac{2^{2m}}{4m}$ for large enough $m$

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

$P_s \left[ k \ \text{prime not in } B \right] = \frac{2^{2m}/8m}{2^{2m}} = \frac{1}{8m}$

*) See Thm A.2.3 in front-back for sufficient, simplified version
Many natural randomized algorithms have one-sided error.

Might make mistake when $x \in L$ but never when $x \notin L$ or the other way round (we just saw one such example).

**DEF** \( \text{RTIME}(T(n)) \) contains every language \( L \) for which \( \exists \text{PTM} M \) running in time \( O(T(n)) \) such that:

\[
\begin{align*}
    x \in L & \implies \Pr[M(x) = 1] \geq \frac{2}{3} \\
    x \notin L & \implies \Pr[M(x) = 0] = 1
\end{align*}
\]

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

**OBS** \( \text{RP} \subseteq \text{NP} \)

*Every accepting branch is a certificate.*

Don't know if \( \text{BPP} \subseteq \text{NP} \). (positive answer)

\( \text{RP} : \) "Never false positives" (always right)

\( \text{coRP} = \{ L \mid L \text{ is in RP} \} \) "Never false negatives"

Given general PTM \( M \), can define random variable:

\[ T_{M,x} = \text{running time of } M \text{ on } x. \]

Take expectation of this random variable:

\[
E[T_{M,x}] = \sum_{t=1}^{\infty} t \cdot \Pr[T_{M,x} = t]
\]

Say \( M \) has expected running time \( T(n) \) if:

\( \forall x \in \{0,1 \}^* \ E[T_{M,x}] \leq T(|x|) \)
**DEF 8** \( \text{ZTIME}(T(n)) \) contains all languages \( L \) for which \( \exists \text{PTM } M \) that runs in expected time \( O(T(n)) \) such that
\[
\Pr \left[ M(x) = 1(x) \mid M \text{ halts} \right] = 1
\]
\( \text{ZPP} = \bigcup_{c \in \mathbb{N}^+} \text{ZTIME}(n^c) \)

"Zero-sided error"

\( \text{ZPP} \) zero-error probabilistic polynomial time

**THM 9** \( \text{ZPP} = \text{RP} \cap \text{coRP} \)

**Proof** Exercise.

Also immediately clear from def
\( \text{RP} \subseteq \text{BPP} \)
\( \text{coRP} \subseteq \text{BPP} \)

**ROBUSTNESS OF DEFINITIONS**

(a) Error probability: constant \( 2/3 \) arbitrary
(b) Can use expected running time instead of worst case
(c) Can use biased coins
(d) Can even use imperfect random sources ("weak random sources")

Will show (a) — see Sec 7.4 for the rest
LEMMA 10. For \( c > 0 \) constant, let 
\[ \text{BPP}_{1/2 + u - c} \] 
denote class of languages \( L \) for which \( \exists \) poly-time PTM \( M \) s.t. \( \forall x \in \{0,1\}^* \) 
\[ \Pr \left[ M(x) = L(x) \right] \geq \frac{1}{2} + |x|^{-c} \]
Then \( \text{BPP}_{1/2 + u - c} = \text{BPP} \).

Need to show: can go from success prob \( \frac{1}{2} + |x|^{-c} \) to \( \frac{2}{3} \).
Show sel. stronger: can go to \( 1 - 2^{-\text{nd}} \) exponentially small failure prob \( b \).

THEOREM 11 (ERROR REDUCTION FOR BPP)
Suppose \( \exists \) poly-time PTM \( M \) for \( L \) s.t. 
\[ \forall x \quad \Pr \left[ M(x) = L(x) \right] \geq \frac{1}{2} + |x|^{-c} \]
Then \( \forall d > 0 \) \( \exists \) poly-time PTM \( M' \) s.t. 
\[ \forall x \quad \Pr \left[ M'(x) = L(x) \right] \geq 1 - 2^{-d |x|^d} \]

Proof:
\( M' \) runs \( M \) for \( k = 8/|x| 2c + d \) times, collects answers, and takes majority vote.

How confident can we be that this is correct?
Use material from App A.2.1 and A.2.4

Let \( X_i = \begin{cases} 1 & \text{if } i^{\text{th}} \text{ run of } M \text{ gets } x \text{ right} \\ 0 & \text{otherwise} \end{cases} \)
\[ \Pr \left[ X_i = 1 \right] = p \quad \text{for } p \geq \frac{1}{2} + |x|^{-c} \]
[Suppose \( p = \frac{1}{2} + |x|^{-c} \) for simplicity]

\[ \mathbb{E} \left[ \sum_{i=1}^{k} X_i \right] = k p = \frac{8/|x| 2c + d}{2} + \frac{8/|x| c + d}{\text{margin}} \]
If you repeat independent trials sufficiently many times, you will get very close to expected value with very high probability.

**Lemma 12 (Chernoff Bound)**

\[
Pr \left[ \left| \sum_{i=1}^{k} X_i - \rho k \right| > \delta \rho k \right] < \exp \left( -\frac{\delta^2}{4} \rho k \right)
\]

Plug in \( \rho = \frac{1}{2} + 1 \times 1^{-c} \)

\[
\delta = \frac{1 \times 1^{-c}}{2}
\]

We will be correct unless \( \sum_{i=1}^{k} X_i < \rho k - \delta \rho k \).

That probability is bounded by

\[
\exp \left( -\frac{1}{4 \times 1^{-c}} \cdot \frac{81 \times 1^{2c+1d}}{2} \right) = \exp \left( -\frac{131^{d}}{2} \right) < 2^{-131^{d} \frac{13}{2}}
\]

Relationship between BPP and other classes:

**Theorem 12**

\[ \text{BPP} \subseteq \text{P/poly} \]

**Theorem 13**

\[ \text{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p \ (\subseteq \text{P/NE}) \]

Both proofs use error reduction in Theorem 12 plus some other ideas.

Proof of Theorem 13 is extremely neat...
But will have to skip it due to time constraints.

Try to sketch proof of Theorem 12...
If \( L \subseteq \text{BPP} \), then by Thm 11 (and Prop 3) there exists an EPTM \( M \) that on input size \( n \):
- uses \( n \) random bits
- gets answer right except with prob \( 2^{-n} \)

Let \( r \) be the random bits.

Say \( r \) bad for \( x \) if \( M(x, r) \neq 2^n(x) \)

For every \( x \), \( M \) succeeds with prob \( 1 - 2^{-n+1} \).
- out of \( 2^n \) random strings, \( \leq 2^n / 2^{n+1} \) bad for \( x \).

\[ \left| \{ r \mid r \text{ bad for some } x \} \right| \leq \sum_{x \in \{0,1\}^n} \left| \{ r \mid r \text{ bad for } x \in \{0,1\}^n \} \right| \]
\[ \leq \frac{1}{2} 2^n \cdot \frac{2^m}{2^{n+1}} = 2^m / 2 \]

But this means that there is at least one random string \( r^* \in \{0,1\}^m \) (in fact, at least half)
that are good for all \( x \in \{0,1\}^n \).

Run \( M \) with this \( r^* \) as advice!

Checking Thm 11 again, \( r^* \) will have poly size.
What about complete problems for BPP?

Typical complete problems
- TM of correct type running with resource bound such-and-such

"Correct type": DTM - easy to check, NDTM - easy to check
- Synthetic

BPP-style: accept \( x \) with prob \( \geq 2/3 \)
or prob \( \leq 1/3 \)
but not in between

Undecidable to check

Hierarchy theorems?

Fail for similar reasons.

A final useful notion: Randomized reductions

**DEF 14** Language \( B \) reduces to language \( C \) under randomized reductions, denoted \( B \leq_r C \),
if \( \exists \) PTM \( M \) s.t.
\[
\forall x \in \{0,1\}^* \quad P(\left\lceil C(M(x)) = B(x) \right\rceil) \geq 2/3
\]

Not transitive
But if \( C \in \text{BPP} \) and \( B \leq_r C \) then \( B \in \text{BPP} \)

Could have defined \( \text{NP} \) in terms of randomized reductions instead (if \( \text{BPP} \) better formalization of "efficient computation")
\[ NP = \{ L \mid L \leq_p 3\text{-SAT} \} \]

**DEF 15**

\[ BP \cdot NP = \{ L \mid L \leq_r 3\text{-SAT} \} \]