2. Polynomial identity testing

- Given: polynomial $p(x_1, x_2, \ldots, x_n)$ as arithmetic formula (fan-out 1):
  - multiplication (fan-in 2)
  - addition (fan-in 2)
  - negation (fan-in 1)

Polynomial identity testing

- Question: Is $p$ identically zero?
  - i.e., is $p(x) = 0$ for all $x \in \mathbb{F}^n$
  - (assume $|\mathbb{F}|$ larger than degree…)

"polynomial identity testing" because given two polynomials $p, q$, we can check the identity $p \equiv q$ by checking if $(p - q) \equiv 0$

**Lemma** (Schwartz-Zippel): Let $p(x_1, x_2, \ldots, x_n)$ be a total degree $d$ polynomial over a field $\mathbb{F}$ and let $S$ be any subset of $\mathbb{F}$. Then if $p$ is not identically 0,

$$\Pr_{r_1, r_2, \ldots, r_n \in S} [p(r_1, r_2, \ldots, r_n) = 0] \leq d/|S|.$$
Polynomial identity testing

- write $p(x_1, x_2, \ldots, x_n)$ as
  $p(x_1, x_2, \ldots, x_n) = \sum_i p_i(x_2, \ldots, x_n)$
- $k = \text{max. } i \text{ for which } p_i(x_2, \ldots, x_n) \text{ not id. zero}$
- by induction hypothesis:
  $\Pr[p_k(r_2, \ldots, r_n) = 0] \leq \frac{d-k}{|S|}$
  whenever $p_k(r_2, \ldots, r_n) \neq 0$, $p(x_1, r_2, \ldots, r_n)$ is a univariate polynomial of degree $k$

$\Pr[p(r_1, r_2, \ldots, r_n) = 0 | p_k(r_2, \ldots, r_n) \neq 0] \leq \frac{k}{|S|}$

- conclude:
  $\Pr[p(r_1, r_2, \ldots, r_n) = 0] \leq \frac{d-k}{|S|} + \frac{k}{|S|} = \frac{d}{|S|}$

Note: can add these probabilities because
$\Pr[E_1] = \Pr[E_1 | E_2] \Pr[E_2] + \Pr[E_1 | \neg E_2] \Pr[\neg E_2]$
Unique solutions

Question: given polynomial-time algorithm that works on SAT instances with at most 1 satisfying assignment, can we solve general SAT instances efficiently?

• Answer: yes
  – but (currently) only if “efficiently” allows randomness

Theorem (Valiant-Vazirani): there is a randomized poly-time procedure that given a 3-CNF formula

\( \varphi(x_1, x_2, \ldots, x_n) \)

outputs a 3-CNF formula \( \varphi' \) such that

– if \( \varphi \) is not satisfiable then \( \varphi' \) is not satisfiable
– if \( \varphi \) is satisfiable then with probability at least \( 1/(8n) \) \( \varphi' \) has exactly one satisfying assignment

Proof:

– given subset \( S \subset \{1, 2, \ldots, n\} \)
  • pick random subset \( S_i \)
  • set \( \varphi_i = \varphi_{i-1} \land \theta_S \)
  – output random one of the \( \varphi_i \)

Claim: if \( |T| > 0 \), then

\[ \Pr[T \text{ agrees with } t \text{ on } S_1, S_2, \ldots, S_{k+2}] < \frac{1}{2} \]

– sum over at least \( 2^k \) different \( t \in T \) (disjoint events); claim follows.

Claim: if \( 2^k \leq |T| \leq 2^{k+1} \), then the probability \( \varphi_{k+2} \) has exactly one satisfying assignment is \( \geq 1/8 \)

– fix \( t, t' \in T \)

\[
\begin{align*}
S_i & \text{ contains even } \# \text{ of positions } i \text{ where } t \neq t' \\
\Pr[t \text{ agrees with } t' \text{ on } S] & = \frac{1}{2} \\
\Pr[t \text{ agrees with } t' \text{ on } S_1, S_2, \ldots, S_{k+2}] & = \left(\frac{1}{2}\right)^{k+2}
\end{align*}
\]
Randomized complexity classes

- model: probabilistic Turing Machine
  - deterministic TM with additional read-only tape containing "coin flips"

- BPP (Bounded-error Probabilistic Poly-time)
  - $L \in \text{BPP}$ if there is a p.p.t. TM $M$:
    - $x \in L \Rightarrow \Pr_y[M(x,y) \text{ accepts}] \geq 2/3$
    - $x \notin L \Rightarrow \Pr_y[M(x,y) \text{ rejects}] \geq 2/3$
  - "p.p.t" = probabilistic polynomial time

Randomized complexity classes

- RP (Random Polynomial-time)
  - $L \in \text{RP}$ if there is a p.p.t. TM $M$:
    - $x \in L \Rightarrow \Pr_y[M(x,y) \text{ accepts}] \geq \epsilon$
    - $x \notin L \Rightarrow \Pr_y[M(x,y) \text{ rejects}] = 1$

- coRP (complement of Random Polynomial-time)
  - $L \in \text{coRP}$ if there is a p.p.t. TM $M$:
    - $x \in L \Rightarrow \Pr_y[M(x,y) \text{ accepts}] = 1$
    - $x \notin L \Rightarrow \Pr_y[M(x,y) \text{ rejects}] \geq \frac{1}{2}$

One more important class:

- ZPP (Zero-error Probabilistic Poly-time)
  - $\text{ZPP} = \text{RP} \cap \text{coRP}$
  - $\Pr_y[M(x,y) \text{ outputs "fail"}] \leq \frac{1}{2}$
  - otherwise outputs correct answer

These classes may capture "efficiently computable" better than $P$.

- "1/2" in ZPP, RP, coRP definition unimportant
  - can replace by $1/\text{poly}(n)$

- "2/3" in BPP definition unimportant
  - can replace by $\frac{1}{2} + 1/\text{poly}(n)$

- Why? error reduction
  - we will see simple error reduction by repetition
  - more sophisticated error reduction later

Randomized complexity classes

Randomized complexity classes

Randomized complexity classes

Randomized complexity classes

Error reduction for RP

- given $L$ and p.p.t TM $M$:
  - $x \in L \Rightarrow \Pr_y[M(x,y) \text{ accepts}] \geq \epsilon$
  - $x \notin L \Rightarrow \Pr_y[M(x,y) \text{ rejects}] = 1$

- new p.p.t TM $M'$:
  - simulate $M \times \epsilon$ times, each time with independent coin flips
  - accept if any simulation accepts
  - otherwise reject

Error reduction

- $x \in L \Rightarrow \Pr_y[M(x,y) \text{ accepts}] \geq \epsilon$
- $x \notin L \Rightarrow \Pr_y[M(x,y) \text{ rejects}] = 1$

- if $x \in L$:
  - probability a given simulation "bad" $\leq (1 - \epsilon)$
  - probability all simulations "bad" $\leq (1 - \epsilon)^{\epsilon} \leq e^{1 - \epsilon}

- if $x \notin L$:
  - $\Pr_y[M'(x,y) \text{ accepts}] \geq 1 - e^{1 - \epsilon}$
  - $\Pr_y[M'(x,y) \text{ rejects}] = 1$
Error reduction for BPP

• given L, and p.p.t. TM M:
  \[ x \in L \Rightarrow \Pr_y[M(x,y) \text{ accepts}] \geq \frac{1}{2} + \varepsilon \]
  \[ x \notin L \Rightarrow \Pr_y[M(x,y) \text{ rejects}] \geq \frac{1}{2} + \varepsilon \]

• new p.p.t. TM M':
  – simulate M \((k/\varepsilon)^2\) times, each time with independent coin flips
  – accept if majority of simulations accept
  – otherwise reject

Randomized complexity classes

• We have shown:
  – polynomial identity testing is in coRP
  – a poly-time algorithm for detecting unique solutions to SAT implies \( \text{NP} = \text{RP} \)

Relationship to other classes

• ZPP, RP, coRP, BPP, contain P
  – they can simply ignore the tape with coin flips
• all are in PSPACE
  – can exhaustively try all strings \( y \)
  – count accepts/rejects; compute probability
• RP ⊆ NP (and coRP ⊆ coNP)
  – multitude of accepting computations
  – NP requires only one

Relationship to other classes
BPP

• How powerful is BPP?
• We have seen an example of a problem in BPP that we only know how to solve in EXP.

Is randomness a panacea for intractability?

BPP

• It is not known if BPP = EXP (or even NEXP!)
  – but there are strong hints that it does not
• Is there a deterministic simulation of BPP that does better than brute-force search?
  – yes, if allow non-uniformity

Theorem (Adleman): BPP ⊆ P/poly

BPP and Boolean circuits

• Proof:
  – language L ∈ BPP
  – error reduction gives TM M such that
    • if x ∈ L of length n
      \[ Pr[y \text{ is bad for } x] \leq \left(\frac{1}{2}\right)^n \]
    • if x ∉ L of length n
      \[ Pr[y \text{ is bad for some } x] \leq 2 \left(\frac{1}{2}\right)^n < 1 \]
  – Conclude: there exists some y on which M(x, y) is always correct
  – build circuit for M, hardwire this y

BPP

• Does BPP = EXP?
• Adleman’s Theorem shows:
  BPP = EXP implies EXP ⊆ P/poly

If you believe that randomness is all-powerful, you must also believe that non-uniformity gives an exponential advantage.

BPP

• Next:
  further explore the relationship between randomness and nonuniformity
• Main tool: pseudo-random generators
Derandomization
• Goal: try to simulate BPP in subexponential time (or better)
• use Pseudo-Random Generator (PRG):
  seed \rightarrow G \rightarrow output string
• often: PRG “good” if it passes (ad-hoc) statistical tests

Simulating BPP using PRGs
• Use a PRG G with
  - output length m
  - seed length t = m
  - error \epsilon < 1/6
  - fooling size s = m
• Compute \Pr_z [C_x (G(z)) = 1] exactly
  - evaluate C_x (G(z)) on every seed z \in \{0, 1\}^t
• running time (O(m) + (time for G))2^t

Blum-Micali-Yao PRG
• Initial goal: for all 1 > \delta > 0, we will build a family of PRGs \{G_m\} with:
  - output length m
  - fooling size s = m
  - seed length t = m^\delta
  - running time m^{2c}
  - error \epsilon < 1/6
• implies: BPP \subseteq \bigcap_{\delta > 0} \text{TIME}(2^{n^\delta}) \nsubseteq \text{EXP}
• Why? simulation runs in time
  O(m + m^\delta)(2^{m^\delta}) = O(2^{m^{2\delta}}) = O(2^{n^\delta})