Derandomization, part 2

Jonathan Kelner
3/31/2011

Derandomizing BPP

- Today, will start building up tools we need to derandomize BPP under some complexity assumptions
- Will aim to show BPP=P assuming certain things are hard on average
- Later, we’ll talk about how to get average-case hardness from worst-case hardness in the instances we care about
- Very high-level sketch:
  - Pseudorandom generator is a thing that takes a few random bits and makes a larger number of bits that look random to a poly sized circuit
  - Can use pseudorandom generator to derandomize BPP
  - Can use function that is hard on average to make a pseudorandom generator
- Work will be in the last step, which will have 2 stages:
  - Show how to get 1 extra bit
    - Idea: Use hard on average function f on random bits. If this breaks, should be able to get some nontrivial info about f, contradicting average case hardness
    - Show how to get a lot of extra bits
      - Easy to get a few—just use above step independently on different parts of the random string—but can’t get enough this way
      - To get a lot more, apply to slightly overlapping parts of the random string

Deterministic Error Reduction

- Will show here that we can get error prob 1/t with t repetitions and no additional randomness (!)
  - For simplicity, will do RP, not BPP
- Will use the following (true) assertion without proof
- **Claim:** There is a constant N such that, for any n≥N and d≥32, there exists a graph G with the following properties:
  - G is bipartite with n vertices on each side (call L and R)
  - Each vertex in L has exactly d neighbors in R
  - If T(S) is set of neighbors of verts in S:
    - |T(S)| ≥ (5d/8)|S| for any S ⊆ L with |S| ≤ n/(10d)

Further, we can make a graph like this of exponential size s.t. we can get a list of a vertex’s neighbors in poly time
- See board for what this means and how to use it.

Pseudorandom Generators

- Notational warning—Goldreich and Arora-Barak use opposite version of what ℓ means
  - We’ll now (roughly) follow A-B & sources, so flip from last lecture
- **Definition:** G=ℓ[G : {0,1}n → {0,1}ℓ] is an ℓ-pseudorandom generator (PRG) for circuits of size r if for all circuits C on r inputs of size ≤ r,
  \[
  \Pr_{\sigma \in \{0,1\}^{4d/3}} [C(G_{r}(\sigma)) = 1] - \Pr_{\rho \in \{0,1\}^{4d/3}} [C(\rho) = 1] < \epsilon
  \]
  - Will say quick if runs in time 2O(ℓ/ε)
- **Notes:**
  - Size=#inputs is for convenience, to keep # of parameters down
  - Important that we fool circuits, not just TMs
  - Note only need exp on running time. Why?
- **Claim:** If exists a quick ε-PSG with d(r)=O(log r) and constant ε>0, then BPP=P
- Why?
  - Note that need to stretch random bits exponentially
Indistinguishability vs. Predictability

- Pseudorandomness means can’t use small circuits to distinguish pseudorandom bits from uniform ones.
- Could ask a weaker question: can you predict the $i$th bit from the first $i$-1?
  - Will be easier to focus on constructing unpredictable bits.
- Can show these are the same using a “hybrid argument”.

**Theorem:** If there’s a circuit $C$ of size $\leq r$ s.t.
\[
\Pr_{\sigma \in \{0,1\}^{\lfloor r \rfloor}}[C(G_r(\sigma)) = 1] - \Pr_{\rho \in \{0,1\}^r}[C(\rho) = 1] \geq \epsilon
\]
then, for some $i \in \{1, \ldots, r\}$, exists a circuit $P$ of size $\leq r$ s.t.
\[
\Pr_{\sigma \in \{0,1\}^{\lfloor r \rfloor}}[P(G_r(\sigma)_1, \ldots G_r(\sigma)_{i-1}) = G_r(\sigma)_i] \geq \frac{1}{2} + \frac{\epsilon}{r}
\]
- See blackboard

Average-Case Hardness

- **Definition:** Average-case hardness $H_L(m)$ of lang $L$ on inputs of length $m$ is largest $s$ s.t. no circuit of size $\leq s$ can match $L$ on at least $1/2 + 1/r$ fraction of inputs of length $m$.
- Can we use average-case hard function to one extra random bit?
  - See blackboard
- How about more?
  - See blackboard