1 Today’s topics

- Private coins $\equiv$ public coins (that is, $\text{IP}[k] \equiv \text{AM}$)
  - Goldwasser-Sipser approximate counting protocol
- Towards protocols for the Permanent, in the goal of showing that $\#P \subseteq \text{IP}$.

2 Review of last lecture

2.1 Graph Non-Isomorphism

The graph non-isomorphism problem is to decide the language $\text{GNI} = \{(G_0, G_1) \mid G_0 \not\sim G_1\}$. Last lecture we saw a private-coin 2-round protocol for GNI:

- The Verifier chooses a permutation $\pi \in R \mathcal{S}_n$, where $n$ is the number of nodes in the graph, and a bit $b \in R \{0, 1\}$. It sends $H = \pi(G_b)$ to the Prover.
- The Prover returns with a bit $b'$.
- The Verifier accepts if $b = b'$.

If $G_0 \sim G_1$, then the bit $b$ is independent of $H$, which means that the Prover is essentially guessing $b'$ and has a probability of $1/2$ of being correct. If $G_0 \not\sim G_1$, then $H$ identifies $b$ uniquely and the Prover will always be correct.

2.2 Kilian’s protocol: $\text{IP}[k] \subseteq \text{AM[\text{poly}]}$

Last time we saw a public-coin protocol through which the Prover could convince the Verifier that there are “many” coin tosses that would have made the Verifier accept in the original private-coin protocol. Kilian’s protocol is public coin and has completeness of 1, but the number of rounds in the protocol depends on the number of random coins used in the original private-coin protocol. For example, the public-coin protocol we would get for GNI would have $O(n \log n)$ rounds instead of 2.
3 The Goldwasser-Sipser Protocol

Let $S \subseteq \{0,1\}^n$ be a set, such that membership in $S$ is verifiable in AM. We are interested in solving the promise problem given by

$$
\Pi_{\text{YES}} = \{S : |S| \geq f(n)\}
$$

$$
\Pi_{\text{NO}} = \{S : |S| < \frac{f(n)}{10n^2}\}
$$

3.1 Initial attempt

Suppose that $f(n)$ is “very large”: $f(n) \approx 2^n$ (actually a little less than that). In this case, for YES instances we have $\frac{|S|}{|\{0,1\}^n|} \approx 1$, and for NO instances we have $\frac{|S|}{|\{0,1\}^n|} \lesssim \frac{1}{n^2}$. In other words, when we choose a random string, for YES instances our chance of hitting a member of $S$ is very high, and for NO instances it is about 1 in $n^2$. We can re-use ideas from the proof that $\text{BPP} \subseteq \Sigma_2^P$ to convince the Verifier that $|S| \geq f(n)$. The protocol would be as follows.

```
Verifier

$\leftarrow x_1, \ldots, x_k \in \{0,1\}^n$

$\leftarrow x \in_R \{0,1\}^n$

$i : x_i \oplus x \in S$

```

Verifi that

```
$\leftarrow \{ x_i \oplus x \in S \}$

```

```
Query

Response

Verifier accepts iff it is “convinced” that $x_i \oplus x \in S$.
```

Unfortunately, this only works when $f(n) \approx \frac{2^n}{\text{poly}}$.

3.2 Using hasing to overcome small $f(n)$’s

We can use pairwise-independent hashing to map $S$ to a smaller space $\{0,1\}^m$, so that if $S$ is large, then $h(S)$ will be a very large fraction of $\{0,1\}^m$. If $h$ is a hash function that has no collisions in $S$ then $|h(S)| = |S|$; therefore, if we choose $m \approx \log f(n)$ and $S$ is a YES instance ($|S| \geq f(n)$), then in the hash-space we will have $|h(S)| = |S| \approx 2^m$. Then we can execute the previous protocol in the hash-space:
where $\mathcal{H}$ is a pairwise-independent family of hash functions.

The problem with this is that $h(S)$ is only guaranteed to be large if $h$ has no (or few) collisions in $S$; if $h(S)$ is too small we have the same problem we had when $f(n)$ was small — the prover may not have a good response $x_1, \ldots, x_k$ in the second round.

To decrease the chances of this bad event we will use more than one hash function, so that the probability that one of the hash functions is good will be very high. We need to make sure that if $S$ is small (a NO instance), our multiple hash functions will not give the Prover too much freedom by mapping $S$ to a large fraction of $\{0,1\}^m$.

### 3.3 Final protocol

Let $m = \log f(n) + 2$. The protocol is as follows.
3.3.1 Analysis

First suppose that $S$ is a NO instance, that is, $|S| < \frac{f(n)}{2m}$. Then

$$\frac{|\bigcup_j h_j(S)|}{|\{0,1\}^m|} \leq \frac{m \cdot |S|}{2^m} < \frac{(\log f(n) + 2) \cdot f(n)}{10n^2 \cdot 4f(n)} \leq \frac{1}{20n}$$

(In the last step we used the fact that $2 \leq f(n) \leq 2^n$, otherwise the problem definition does not make sense). It follows that for NO instances, for every choice of $x_1, \ldots, x_n$ that the Prover may send, the Verifier has a high probability of selecting a string $x$ such that for all $i$, $x_i \oplus x \not\in \bigcup_j h_j(S)$. The Prover will not have a response $i, j, y$ that has a high probability of making the Verifier accept.

Now suppose that $S$ is a YES instance, that is, $|S| \geq f(n)$. We are interested in the probability of choosing $h_1, \ldots, h_m$ such that for some $j \in \{1, \ldots, m\}$ we have $|h_j(S)| \geq f(n)$. This ensures that the Prover has a response in the second round.

Since each $\mathcal{H}$ is pairwise-independent, for any fixed $x \neq y \in \{0,1\}^n$ and for every $a, b \in \{0,1\}^m$ we have

$$\Pr_{h \in \mathcal{H}}[h(x) = a \land h(y) = b] = \frac{1}{4^m}$$

and hence,

$$\Pr_{h \in \mathcal{H}}[h(x) = h(y)] = \sum_{a \in \{0,1\}^m} \Pr_{h \in \mathcal{H}}[h(x) = a \land h(y) = a] \leq 2^m \cdot \frac{1}{4^m} = \frac{1}{2^m}$$

Let $S' \subseteq S$ be some subset of size $f(n)$ of $S$ (recall that $|S| \geq f(n)$). By a union bound over
elements of $S'$ ("unfixing" $y$) we obtain
\[
\Pr_{h \in H} [\exists y \in S' : x \neq y \land h(x) = h(y)] \leq \frac{f(n)}{2^m} = \frac{f(n)}{4f(n)} = \frac{1}{4}
\]

Therefore,
\[
\Pr_{h_1, \ldots, h_m \in H} [\forall j \exists y \in S' : x \neq y \land h_j(x) = h_j(y)] \leq \frac{1}{4^m}
\]

Finally, by another union bound over the elements of $S'$ ("unfixing" $x$ this time),
\[
\Pr_{h_1, \ldots, h_m \in H} [\exists x \in S' \forall j \exists y \in S' : x \neq y \land h_j(x) = h_j(y)] \leq \frac{f(n)}{4^m} = \frac{f(n)}{8f(n)} < \frac{1}{f(n)}
\]

It follows that with probability all but $1/f(n)$ there will be at least one $h_j$ that maps $f(n)$ elements of $S$ to distinct elements of $\{0, 1\}^m$, which implies $|\bigcup_j h_j(S)| \geq f(n)$. This is sufficient to use the BPP $\subseteq \Sigma^P_2$-style protocol above.

4. $\text{IP}[2] \subseteq \text{AM}$

We will use the Goldwasser-Sipser approximate counting protocol to convert a 2-round private coin protocol into a constant-round public-coin protocol. Since $\text{AM}[k] = \text{AM}$ for any constant $k$ (see problem set #3), this will show that $\text{IP}[2] \subseteq \text{AM}$.

A 2-round private-coin protocol looks like this:

<table>
<thead>
<tr>
<th>Verifier</th>
<th>Prover</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chooses</td>
<td>$R \in_R {0, 1}^{r(n)}$</td>
</tr>
<tr>
<td>$q = q(R)$</td>
<td>$a$</td>
</tr>
<tr>
<td>Verifier checks that $(R, q, a)$ is acceptable</td>
<td></td>
</tr>
</tbody>
</table>

Define $S_{q,a} = \{R \mid (R, q, a) \text{ is acceptable}\}$. If we make the simplifying assumption that there is some number $N$ such that

1. In YES instances, $\forall q \exists a : |S_{q,a}| \geq N$, and
2. In NO instances, $\forall q \forall a : |S_{q,a}| < \frac{N}{f(n)}$, and
3. \( N \) is known to the Verifier, then the Prover could convince the Verifier that it should accept by sending a pair \((q, a)\) that it claims has \(|S_{q,a}| \geq N\), and then the Verifier and the Prover would run the GS protocol to verify that \(|S_{q,a}| \geq N\). However, in reality, \( N \) is not known to the verifier, and furthermore \( N \) can depend on \( q \) (that is, \( N = N(q) \)).

The solution is to have the Prover provide a number \( N \) in the first round that it claims satisfies \(|Q_N| \geq 2^{3/2} \cdot \epsilon \cdot 2^n \cdot N\), where \( \epsilon \) is some constant and where \( Q_N = \{ q \mid \exists a : |S_{q,a}| > N \} \). That is, the prover provides a number \( N \) such that “many” queries \( q \) have an answer \( a \) with \(|S_{q,a}| > N\). Then the Verifier and Prover execute the GS protocol to verify the size of \( Q_N \). After this, \( N \) is used in the protocol from before.

5. **Properties of the Permanent**

Recall that \( \text{perm}(A) = \sum_{\sigma} \prod_{i=1}^{n} A_{i,\sigma(i)} \), where \( A \in \mathbb{Z}_p^{n \times n} \).

- **Random self-reducibility**: for fixed \( A, R \in \mathbb{Z}_p^{n \times n} \), let \( p(i) = \text{perm}(A + i \cdot R) \). This is a degree-\( n \) polynomial in \( i \). For \( i = 0 \) we have \( p(i) = \text{perm}(A) \).

Suppose that we have an algorithm \( M \) that computes \( \text{perm}(R) \) with probability \( 1 - \epsilon \) for random matrices \( R \). Since \( \mathbb{Z}_p \) is a field, for \( i \neq 0 \) and randomly chosen \( R \), \( A + i \cdot R \) is also distributed uniformly. Therefore,

\[
\Pr[M(A + i \cdot R) = \text{perm}(A + i \cdot R)] \geq 1 - \epsilon.
\]

It follows that

\[
\Pr[\forall i \in \{1, \ldots, n\} : M(A + i \cdot R) = \text{perm}(A + i \cdot R)] \geq 1 - n \cdot \epsilon,
\]

that is, we can compute w.h.p. the values \( p(1), \ldots, p(n) \). These values can be used to interpolate \( p \) (which is of degree \( n \)) and compute \( p(0) = \text{perm}(A) \).

- **Downward self-reducibility**: suppose we have an algorithm \( M \) that computes \( \text{perm}(B) \) for \( B \in \mathbb{Z}_p^{n^{-1} \times n^{-1}} \) (a smaller matrix). For an \( n \)-by-\( n \) matrix \( A \), we can write \( \text{perm}(A) = \sum_{i=1}^{n} (a_{1,i} \cdot \text{perm}(A \setminus i)) \), where \( A \setminus i \) is the matrix obtained from \( A \) by removing the first row and the \( i \)-th column. Thus, we can compute \( \text{perm}(A) \) using \( n \) calls to \( M(B) \), and if \( M \) terminates in polynomial time then so will the computation of \( \text{perm}(A) \).

These two properties are used in an alternating manner in an interactive proof for the Permanent problem.