Derandomization

The Facts about Randomness:

1) Randomness can speed-up algorithms (e.g., sub-linear time algorithms)

2) Under reasonable assumptions, we know that the speed-up can't be too large:

A randomized algo that on input of size $n$ outputs in time $t$ the correct answer with prob $\geq 2/3$ can be converted to a deterministic algo that runs in time $\text{poly}(n, t)$, and always outputs the correct answer.

(Proof in 6.841)

3) Many times randomness doesn't buy any speed-up. More than that - the randomized algorithm, which is based on unique ideas and techniques, can lead to a deterministic algo.
Today Two basic derandomization techniques:

1. "The method of conditional expectations"

A randomized algorithm

```
  head  tail
     /   \
head  tail
     /   \head  tail
     /     /     /     /
```

possible outputs; most of them correct.

The method is to find a path down the tree that leads to a correct output, by choosing at each point head/tail according to the fraction of correct outputs at each.
(2) Pairwise independence

General theme: randomness is in the eyes of the beholder.

Suppose the algorithm can't distinguish between a truly random string and a string that is drawn from a short list of strings.

Just run the algo on the N strings and take the output that repeats the largest number of times.
We'll demonstrate the two techniques using the following example:

**Big-Cut**

Given an undirected graph $G = (V, E)$, find a cut $(S, V - S)$, $S \neq \emptyset, V$, that contains at least half of the edges, i.e.,

$$|\{(u, v) \in E | u \in S, v \in V - S\}| \geq |E|/2.$$

**Randomized Algorithm**

1) $S \leftarrow \emptyset$
2) For every $v \in V$, with prob $\frac{1}{2}$, $S \leftarrow S \cup \{v\}$.
3) Return $(S, V - S)$.

Time $\Theta(|V|)$. 

Correctness of randomized algo:

For every edge $(u,v) \in E$,

$$P\left( \frac{u \in S \text{ or } v \in S}{u \notin S \text{ and } v \notin S} \right) = \frac{1}{2}$$

\[ \text{⇒ The expected number of edges in the cut is } \frac{1}{2}|E|. \]

(This is not quite enough to produce a cut containing half of the edges with good prob, but using Chernoff, with extremely high prob, the size of the cut ≥ 0.999999|E|. )
Big Cut From Conditional Expectations

Suppose we already decided whether \( v_k \in V \) will be in \( S \) or in \( V - S \).

\[ E_1 \text{: the edges } (v_i, v_j) \quad 1 \leq i, j \leq k \]
\[ E_2 \text{: the edges } (v_k, v_{k+1}) \quad 1 \leq i \leq k \]
\[ E_3 \text{: the edges } (v_i, v_j) \quad i, j > k + 1 \]

Moreover, suppose at least half of \( E_1 \) is in the cut so far.

We can compute how many \( E_2 \) edges are in the cut if \( v_{k+1} \in S \), and how many if \( v_{k+1} \in V - S \). Suppose, without loss of generality, that more \( E_2 \) edges are in the cut if \( v_{k+1} \in S \).

\[ \Rightarrow \text{At least } |E_2|/2 \text{ } E_2 \text{ edges are in the cut then.} \]

If for \( i > k + 1 \) whether \( v_i \in S \) is decided independently at random, then the expected \# edges in the cut?

\[ |E_1|/2 + |E_2|/2 + |E_3|/2 = |E_2|/2 \Rightarrow \text{Continue until all decisions are made.} \]
Big Cut From Pairwise Independence

The randomized algorithm only needs:

\[ \forall u \neq v \quad P\left( \frac{u \in S \land v \in S}{u \in \overline{S} \land v \in \overline{S}} \right) \leq \frac{1}{2} \]

Suppose \( \mathcal{H} = \{ h : V \rightarrow \{0, 1\} \} \) is a universal hash family. Then,

\[ \forall u \neq v \quad \mathbb{P}_{h \in \mathcal{H}}\left( h(u) = h(v) \right) \leq \frac{1}{2} \]

Thus, if we pick at random \( h \leftarrow \mathcal{H} \), and take \( S \) if \( h(v) = 1 \) and \( \overline{S} \) if \( h(v) = 0 \), then the expected number of edges in the cut is \( \geq \frac{1}{2} |E| \).

\[ \Rightarrow \exists H \in \mathcal{H} \text{ for which the # of edges in the cut is } \geq \frac{1}{2} |E|. H \text{ is much smaller than the number of all cuts. Eq. can take } \square \]