Schwartz-Zippel alg:

For polynomial $P(x_1, ..., x_n)$ of degree $\leq Q$, choose $r_1, ..., r_n$ u.a.r. from $S$
then $\Pr(P(r_1, ..., r_n)=0 | P \neq 0) \leq \frac{Q}{|S|}$

Perfect matchings of bipartite graphs:

For $G=(L\cup R, E)$, let nxn matrix A be:

$$a_{ij} = \begin{cases} x_{ij} & \text{if } (i,j) \in E \\ 0 & \text{o/w} \end{cases}$$

where the $x_{ij}$'s are variables.

Then, $\det(A) \neq 0$ iff $G$ contains a perfect matching.
(can generalize to non-bipartite graphs, see homework)
Moreover, if $G$ contains a perfect matching then we can iteratively construct a perfect matching by checking if $\det(A_{ij}) \neq 0$ for an edge $(ij)$ where $A_{ij} = A$ with row $i$ & column $j$ removed, and recursing on $G \setminus \{i,j\}$ if it is $\neq 0$.

But can we find a perfect matching in parallel? Can all edges simultaneously check if $\det(A_{ij}) \neq 0$? If there is a unique perfect matching that all edges are checking against then we can do it in parallel.

**Isolation Lemma**: [Mulmuley, Vaziran, Vaziran, '87]

Let $S_1, \ldots, S_k$ be subsets of a set $S$ where $|S| = m$. For each $x \in S$, choose $w(x)$ u.a.r. from $\{1, \ldots, \ell\}$. Then, $\Pr(\exists$ unique subset $S_i$ of min weight $) \geq 1 - \frac{m}{\ell}$

where $w(S_i) = \sum_{x \in S_i} w(x)$.
Now let's apply the isolation lemma to the perfect matching problem. For each edge \((i,j)\), choose its weight \(w_{ij}\) u.a.r. from \(\{1, \ldots, 2m\}\). Hence, we know that with \(\text{prob.} \geq \frac{1}{2}\) there is a unique perfect matching of min weight. But this weights a matching \(M\) as \(w(M) = \sum_{e \in M} w(e)\).

And in the \(\det(A)\) the weight is \(\prod_{e \in M} x_e\).

Thus, in matrix \(A\), replace \(x_{ij}\) by \(2^{w_{ij}}\).

Denote this matrix as \(D\).

We know that if \(G\) does not contain a perfect matching then \(\det(D) = 0\). Since \(\det(A) = 0\).

What if \(G\) has a perfect matching? Lemma: If there is a unique perfect matching of min weight, then \(\det(D) \neq 0\) & the max power of 2 dividing \(\det(D)\) is \(W^* = \min \text{ weight of perfect matching}\)

\[= \min_{p \in \mathcal{M}} \sum_{e \in M} w(e)\]

where \(\mathcal{M}\) = set of all perfect matchings in \(G\).
Proof: For $\mathcal{P} =$ set of perfect matchings, we're assuming $\mathcal{P} \neq \emptyset$ & that there is a unique perfect matching of min weight, denote it as $M^*$ and $W^* = w(M^*)$.

\[
\det(D) = \sum_{M \in \mathcal{P}} (-1)^{\text{sgn}(M)} \prod_{i} 2 w_i m_i = \sum_{M \in \mathcal{P}} (-1)^{\text{sgn}(M)} \prod_{i} w_i m_i
\]

\[
= \sum_{M \in \mathcal{P}} \pm 2 w(M) = \pm 2 w^* + \sum_{j > w^*} k_j 2^j \quad \text{for some integers } k_j
\]

Since exactly one $M^* \in \mathcal{P}$ with $w(M^*) = W^*$, all $M \neq M^*$ have $w(M) > W^*$.

$\blacksquare$
We now have a parallel alg:

1. For each $(i,j) \in E$ pick $w_{ij}$ u.a.r. from $[1, \ldots, 2m]$.
2. Compute $\det(D)$ (this can be done in parallel).
3. If $\det(D) = 0$ then output NO perfect matching.
4. Let $W^*$ be the max $i$ where $2^i$ divides $\det(D)$.
5. For each $(i,j) \in E$:
   a. Evaluate $\det(D_{ij})$.
   b. If $\det(D_{ij}) = 0$ then stop considering $(i,j)$.
   c. Else, find max $j$ s.t. $2^j$ divides $\det(D_{ij})$
      and denote it as $W^*_{ij}$.
6. If $W^*_{ij} + w_{ij} = W^*$ then output edge $(i,j)$.
7. Finally, check that the outputted edges form a perfect matching.

Note, $\Pr(\text{alg. outputs a perfect matching}) \geq \frac{1}{2}$.
Alice has a \( n \)-bit number \( a = a_0 a_1 \ldots a_{n-1} \). Bob has a \( n \)-bit number \( b = b_0 b_1 \ldots b_{n-1} \). Where \( n \) is HUGE. Can they quickly check if \( a \equiv b \)?

**Fingerprinting:**

Alice picks a prime number \( p \) u.a.r. from \( \mathbb{F}_p \), where \( T \) will be specified later.

She computes \( F_p(a) = a \mod p \).

She sends \( p \) & \( F_p(a) \) to Bob.

Bob computes \( F_p(b) = b \mod p \)

& checks if \( F_p(a) = F_p(b) \)?

If \( a = b \) then we always have \( F_p(a) = F_p(b) \). But if \( a \neq b \) then we might still have \( F_p(a) \equiv F_p(b) \).
For an integer \(x > 0\), let \(\pi(x) = \# \text{ of primes } \leq x\).

**Prime number theorem:**  
\[
\lim_{{x \to \infty}} \frac{\pi(x)}{\frac{x}{\ln x}} = 1
\]

Moreover, for \(x \geq 17\),  
\[
\frac{x}{\ln x} \leq \pi(x) \leq 1.26 \frac{x}{\ln x}
\]

Back to the original problem,

If \(F_p(a) = F_p(b)\) then \(a \equiv b \mod p\),

so \(a - b \equiv 0 \mod p\)

which means \(p\) divides \(|a - b|\)

Note, \(|a - b|\) is \(\leq n\) bits,

and thus \(\leq \pi(n)\) primes divide \(|a - b|\)

(since each prime is \(\geq 2\))

actually \(\leq \pi(n)\) primes divide \(|a - b|\)

(that will save a log factor)
Therefore,
\[ \Pr(\text{FP}(a) = \text{FP}(b) | a+b) \leq \frac{\pi(n)}{\pi(T)} \leq 1.26 \frac{n \ln T}{\ln n T} \]

Let \( T = cn \),
\[ \leq \frac{1.26}{c} \frac{\ln(cn)}{\ln n} = 1.26 \frac{\ln(c)}{c} \left(1 + \frac{\ln c}{\ln n} \right) \]

This is small even for \( c=10 \),
and then can run \( t \) trials to reduce the error prob exponentially in \( t \).

Note, since \( T = O(n) \),
then \( P(\& \text{FP}(a)) \) are \( O(\log n) \) bits.
Applications of fingerprinting:

- Polynomial identity testing: intermediate computations when evaluating polynomials at \((r_1, \ldots, r_n)\) may be huge numbers. Can do modulo a small prime as in the fingerprinting scheme.

- Pattern matching:

  Binary text \(X = x_1, \ldots, x_n\) for large \(n\)
  & shorter text \(Y = y_1, \ldots, y_m\) for \(m \ll n\).

  Does \(Y\) occur as a contiguous substring of \(X\)?
  I.e., for \(j\) let \(X(j) = x_j x_{j+1} \ldots x_{j+m-1}\)
  Does there exist a \(j\) where \(X(j) = Y\)?

  Naive alg.: \(O(mn)\) time.

  Sophisticated algorithms in \(O(mn)\) time due to [Boyer Moore 77] & [Knuth Morris Pratt 77].

  Here's a simple \(O(m+n)\) time alg. due to [Knuth Rabin 81].
Pick random prime \( p \in [2, \ldots, T] \)
\[
\text{Compute } F_p(Y) = Y \mod p
\]
for \( j = 1 \rightarrow n - m + 1 \):
- Compute \( F_p(X(j)) \)
- If \( F_p(Y) = F_p(X(j)) \) then output match \( j \) & halt.

Output No match.
\[
Pr( F_p(Y) = F_p(X(j)) \mid Y \neq X(j)) \leq \frac{\pi(m)}{\pi(T)}
\]

Thus:
\[
Pr(\text{output match} \mid \text{No match}) \leq \frac{n \pi(m)}{\pi(T)}
\]

But, note that if \( p \) divides \( |Y - X(j)| \) for some \( j \),
then \( p \) divides \( \prod_j |Y - X(j)| \)
and this is \( \leq nm \) bits long.

Hence, we have a better bound: (replace \( n\pi(m) \) by \( \pi(nm) \))
\[
Pr(\text{error}) \leq \frac{\pi(nm)}{\pi(T)} \quad \text{which is small for } T = 10nm.
\]
Running time:

Note, \( p \) has \( O(\log(mn)) = O(\log n) \) bits, so let's assume arithmetic mod \( p \) in \( O(1) \) time.

Since \( y \) is \( m \) bits, then computing \( F_p(y) \) takes \( O(m) \) time.

Need to compute \( F_p(x(j)) \) for all \( j \):

Naive: \( O(m) \) time each & thus \( O(nm) \) total time.

But \( x(j) \) & \( x(j+1) \) are similar:

\[
x(j+1) = 2(x(j) - 2^{m-1} x_j) + x_{j+m}
\]

Thus,

\[
F_p(x(j+1)) = (2(F_p(x(j)) + 2^{m-1} x_j) + x_{j+m}) \mod p
\]

which takes \( O(1) \) time since this is either \( 0 \) or \( 2^{m-1} \) which can be precomputed.

\( \Rightarrow O(n+m) \) total time.