Randomization and Algebra

[These are the rough notes for the lectures on these topics covered in E0234 at IISc. Please do not trust everything that is written. If you notice a glaring error, please let us know.]

1 Communication Complexity

Consider the following communication complexity problem between two parties: $\mathcal{A}$ and $\mathcal{B}$.

1.1 Equality

Input: $\mathcal{A}$ has a string $x \in \{0,1\}^n$, and $\mathcal{B}$ has a string $y \in \{0,1\}^n$.
Goal: Determine with the fewest bits of communication whether x and y are equal.
One can show tyhat at least n bits are needed in any deterministic protocol. However, we will see that with randomization we can do much better.

1.1.1 With shared randomness:

Assume: $\mathcal{A}$ and $\mathcal{B}$ have access to a *shared* random string z.
Protocol: Regard z as a random string in $\{0,1\}^n$. $\mathcal{A}$ sends *a bit* $b = x \cdot z \mod 2$. $\mathcal{B}$ checks if $b = y \cdot z \mod 2$. Thus, if $x = y$, then $\Pr[\text{Error}] = 0$; and if $x \neq y$, then $\Pr[\text{Error}] \leq 1/2$.

With k bits of such communication (using k such random strings), we have the following guarantees:

$$\text{if } x = y, \text{ then } \Pr[\text{Error}] = 0; \qquad \text{if } x \neq y, \text{ then } \Pr[\text{Error}] \leq 2^{-k}.$$

1.1.2 Without shared randomness:

There is a corresponding solution with only $O(\log n)$ bits of communication.
Idea: we show that that there exists a small set of z such that for all x and y, picking a random z from this set works equally well.

Assume there is a linear error correcting code with encoding function $\texttt{Enc} : \{0,1\}^n \to \{0,1\}^{n^2}$, with the property that if $x \neq y$, then $\texttt{Enc}(x)$ and $\texttt{Enc}(y)$ differ in at least $n^2/10$ of the coordinates. Note that $\texttt{Enc}(x)$ is obtained by multiplying row x by the generator matrix of the code. Then, the protocol is as follows, $\mathcal{A}$ picks a random index i and sends $b = \texttt{Enc}(x)[i]$ and the index i to $\mathcal{B}$. $\mathcal{B}$ checks if $b = \texttt{Enc}(y)[i]$. This requires $O(\log n)$ bits of communication and yields the following guarantees:

$$\text{if } x = y, \text{ then } \Pr[\text{Error}] = 0; \qquad \text{if } x \neq y, \text{ then } \Pr[\text{Error}] \leq 9/10$$

(the latter can be reduced by repetition).

Indeed, a slight modification of this idea yields a protocol with $O(\log n)$ communication and $\Pr[\text{Error}] < 1/\text{poly}(n)$. Instead of a binary code, we work with a code over the field $\mathbb{F}_p$ (integers

mod p), for some $p = \Theta(n^2)$. Let the columns of the generator matrix be the Vandermonde matrix $\mathbb{V}$. [1] That is, we pick scalars $\alpha_1, \ldots, \alpha_n$ from the field, and let the i-th column of the matrix be $[1, \alpha_i, \alpha_i^2, \ldots, \alpha_i^{n-1}]^T$. The protocol is the similar; this time, we let $\texttt{Enc}(x) = x \cdot \mathbb{V}$. Then, using the fact that a non-zero polynomial of degree $(n-1)$ has at most $(n-1)$ roots, we conclude that

$$\text{if } x = y, \text{ then } \Pr[\text{Error}] = 0; \quad \text{if } x \neq y, \text{ then } \Pr[\text{Error}] \leq (n-1)/n^2 \leq 1/n.$$

Furthermore, the communication is at most $(2 \log n)$ bits.

Building on this idea, there is a general theorem due to Ilan Newman (1991).

Theorem 1.1. *For any function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, if there is a communication protocol for f with shared randomness with error ϵ, and c bits of communication, then there is a protocol with private randomness and $c + O(\log n)$ bits of communication and error at most $\epsilon + 1/n$.*

1.2 Freivalds's Algorithm for verifying matrix multiplication

Input: $n \times n$ integer matrices A, B, and C.
Goal: Determine if $AB = C$.
Protocol: Pick a random $z \in \{0,1\}^n$ and check if $A(Bz) = C$.
Guarantee:

$$\text{if } AB = C, \text{ then } \Pr[\text{Error}] = 0; \quad \text{if } AB \neq C, \text{ then } \Pr[\text{Error}] \leq 1/2.$$

More generally, if z is picked uniformly from S^n, for a set S of integers, then the error probability is at most $1/|S|$. Similar conclusions hold for verifying matrix multiplications over other fields.

1.3 Communication Protocol for the Greater Than function

Input: Two parties, $\mathcal{A}$ and $\mathcal{B}$. $\mathcal{A}$ has $x \in \{0,1\}^n$ and $\mathcal{B}$ has $y \in \{0,1\}^n$.
Task: Determine if $y \geq x$ (where x and y are treated as n-bit integers in $\{0, 1, \ldots, 2^n - 1\}$).

A natural randomized divide and conquer protocol would determine the most significant bit where x and y differ with $(\log n) \cdot (\log \log n)$ bits of communication, where we use $k = O(\log \log n)$ in the Equality protocol to keep the error down to $\ll 1/\log n$.

An interesting protocol of Nisan (1993)/ Feige-Peleg-Raghavan-Upfal (1990) allows us to complete the task with $O(\log \log n)$ bits of communication.

Idea: $\mathcal{A}$ and $\mathcal{B}$ build identical binary trees on the indices $\{1, 2, \ldots, n\}$. The indices are listed in increasing order from left to right. For a node v of the tree, let $\text{left}_x(v)$ denote the subsequence of bits under the left-child of v. (For a leaf v, the left-child and right-child are v itself). Similarly, let $\text{right}_x(v)$ denote the subsequence of bits under v's right-child. Let $\text{prefix}_x(v)$ denote the substring to the left of $\text{left}_x(v)$.

Protocol: $\mathcal{A}$ and $\mathcal{B}$ start at the root. In general, when they are at a node v, the compare $\text{prefix}_x(v)$, $\text{left}_x(v)$, and $\text{right}_x(v)$ with $\text{prefix}_y(v)$, $\text{left}_y(v)$, and $\text{right}_y(v)$ and move to either v.parent, v.left-child or v.right-child. The comparison is done using the Equality protocol, with $k = 10$ bits (say), so the error is $\ll 1/10$. Note that at all times, if $\mathcal{A}$ and $\mathcal{B}$ are at a node v, the left-most position where x and y differ is either under v or in the prefix of v. $\mathcal{A}$ and $\mathcal{B}$ perform this for $20 \log n$ rounds, and in the end if they are at leaf, declare the corresponding index i to be the left-most bit where x and y differ and declare iff $x_i \leq y_i$.

[1] Vandermonde matrices are generators for Reed-Solomon code.

Analysis: Let ℓ be the leaf corresponding to the left-most position where x and y differ. We track $d_t = dist(\ell, v_t)$ in the tree, where v_t is the vertex currently being considered by $\mathcal{A}$ and $\mathcal{B}$ at time t.

Initially, $d_0 = \log n$. At each round, d_t either increases by 1 or decreases by 1 (unless v is the desired leaf). Thus, the protocol can be thought of as a random walk on the integer line, with an absorbing state at 0, and probability of moving to the left at least 9/10 for all points to the right of 0. To bound the probability of error, it suffices to show that the probability of a net displacement to the left being less than $\log n$ (the initial distance to ℓ) is small. After $20 \log n$ steps, we expected the number of left moves to be at least $18 \log n$ steps. The probability that fewer than $12 \log n$ left moves are made can be upper bounded by Chernoff bound as follows. [Note that that the moves are not necessarily independent We an only claim that the at each step with probability at least 9/10, we move to the left. One can show that in this situation, we can still invoke the Chernoff bound, e.g., by coupling our walk with a walk with independent moves (Thanks Daanish Mahajan for pointing this out)]

$$\Pr[X \leq (1-\varepsilon)\mu] \leq \exp(-\mu\varepsilon^2/2).$$

We have $\mu = 18 \log n, \varepsilon = 1/3$. So,

$$\Pr[\text{Error}] \leq \exp(-18 \log n (1/3)^2/2) < 1/n.$$

We have so far mainly considered linear functions of the form $w \cdot z$ with z random: in the application with the Vandermonde application we considered univariate polynomials of degree $n-1$. The principle that polynomials of low degree are rarely zero is applicable to multivariate polynomials as well.

2 Polynomial Identity Lemma

Lemma 2.1. *Suppose $P(X_1, X_2, \ldots, X_n)$ is a non-zero polynomial of degree at most d over a field $\mathbb{F}$. Suppose S is a subset of $\mathbb{F}$. Then,*

$$\Pr_{z \in S^n}[P(z) = 0] \leq 1/|S|.$$

Proof. We proceed by induction. Write

$$P(X) = X_1^k P_k(X_2, \ldots, X_n) + X_1^{k-1} P_{k-1}(X_2, \ldots, X_n) + P_0(X_2, \ldots, X_n)$$

where k is the highest degree with which X_1 appears in any monomial. Then, "$P(z) = 0$" is a subset of the event "$P_k(z_2, \ldots, z_N) = 0$ and $P(z) = 0$". Thus,

$$\Pr[P(z) = 0] \leq \Pr[P_k(z_2, \ldots, z_N) = 0] + \Pr[P(Z) = 0 \mid P_k(z_2, \ldots, z_N) \neq 0] \cdot \Pr[P_k(z_2, \ldots, z_N) \neq 0]$$

We may drop $\Pr[P_k(z_2, \ldots, z_N) \neq 0]$ in the second term on the right, and use induction. Note $\deg(P_k) \leq d - k$, and that for each choice of $z_2, \ldots, z_N$ where $P_k(z_2, \ldots, z_N) \neq 0$, polynomial $P(Z)$ reduces to a univariate polynomial of degree k. We conclude that

$$\Pr[P(z) = 0] \leq (d-k)/|S| + k/|S| = d/|S|. \qquad \square$$

We will present two applications of this lemma.

2.1 Perfect matchings in bipartite graphs

Consider a bipartite graph $G = (V \cup W, E)$, where both V and W have n vertices. Construct the $n \times n$ adjacency matrix, and replace the entry a_{ij} by $a_{ij} \cdot X_{ij}$. Let the resulting matrix of variables be M.

Note: The $\det(M)$ is a multivariate polynomial of degree at most n, and $\det(M)$ is a non-zero polynomial iff G has a perfect matching. This gives us a natural algorithm:

Pick $z \in 1, \ldots, n^{2n^2}$ and determine if $\det(M)(z) = 0$. The computation can be performed modulo a prime $p > n^2$. Then, if G does not have a matching, then Pr[Error] $= 0$. if G has a matching, then Pr[Error] $\leq n/n^2 = 1/n$.

This is an important algorithm, which can be parallelized to run in time $O((\log n)^2)$ with polynomially many processors.

2.2 Rooted tree isomorphism

Given T_1, T_2, two rooted unordered trees on n vertices, determine if T_1 and T_2 are isomorphic. (It is possible to put the trees in a canonical form, and then check that the two canonical forms are identical). The following randomized algorithm uses commutativity of multiplication to avoid putting the trees in a canonical form.

To each leaf, assign a variable X_i, where i is the distance of the leaf from the root. Then, for an internal node v at level ℓ, inductively construct the polynomial

$$P_v(X) = (X_\ell - P_{v_1}(X))(X_\ell - P_{v_2}(X)) \ldots (X_\ell - P_{v_r}(X))$$

where $v_1, \ldots v_r$ are children of the vertex v. Note that the order of the children does not matter. For a tree T, let P_T be the polynomial associated with its root.

Claim 2.2. *$P_{T_1} = P_{T_2}$ iff T_1 and T_2 are isomorphic.*

Proof Idea. Use induction on the depth of the trees. Note that commutativity automatically takes care of the reordering of the children of the nodes. □

Claim 2.3. *If T has L leaves and depth D, then T is a polynomial of degree at most L in $D + 1$ variables.*

Algorithm: Pick z from $\{1, \ldots, n^2\}^D$ at random and check if $P_{T_1}(z) = P_{T_2}(z)$.
Guarantee: if T_1 is isomorphic to T_2, then Pr[Error] $= 0$ if T_1 is not isomorphic to T_2, then Pr[Error] $\leq 1/n$.

Note that the scalar $P_T(z)$ can be computed inductively, starting at the leaves and moving up the tree. As before, to keep the computation efficient, we can calculate modulo a prime $\gg L$.

3 Primality Testing

Input: A number N (n bits long).
Output: 1 if N is prime, 0 otherwise.
Let $a \in \{1, 2, \ldots, N-1\}$. Consider the following tests:

1. The gcd test: If $\texttt{gcd}(a, N)$ is not 1, then output 0, STOP.
2. The non-trivial-square-root test:[2] If $a^{(N-1)/2} \mod N$ not in $\{+1, -1\}$, then output 0, STOP.

[2]Note that Fermat's little theorem states that if N is a prime then $a^{(N-1)} = 1 \mod N$, for all $a \in [N-1]$.

We will assume that N is not 1, is not even, and is not of the form a^b (for some $a, b \geq 2$).

1. Set Flag = 0.
2. Perform step (2a) to step (2d) five times.
   (a) Pick $a \in \{1, 2, \ldots, N-1\}$ uniformly at random.
   (b) Perform the gcd test.
   (c) Perform the non-trivial-square-root test.
   (d) If $a^{(N-1)/2} = -1$, set Flag = 1.
3. If Flag = 0: output 0, STOP. else: output 1, STOP.

(The above algorithm is a slightly weaker version of the Rabin-Miller algorithm; this algorithm appears in Victor Shoup's book available here.)

Claim 3.1. *If N is prime, then* $\Pr[\textit{Error}] \leq 1/2^5$.

Proof. Clearly, N cannot fail Step (b), and (c). Now, we consider failure probability due to Step (d). If N is prime, then for exactly half the $a \in \{1, 2, \ldots, N-1\}$ we have $a^{(N-1)/2} = -1$. Thus, the probably that Flag = 0, even when 5 attempts were made to set it to 1, is $(1/2)^5$. □

Claim 3.2. *If N is not prime, then* $\Pr[\textit{Error}] \leq (1/2)^5$.

Proof. If there is no a such that $a^{(N-1)/2} = -1$, then Pr[Error] = 0. So, we assume that there is an a such that $a^{(N-1)/2} = -1$. Fix such an a, call it c. Now, suppose $N = N_1 \cdot N_2$, where $\texttt{gcd}(N_1, N_2) = 1$ (We assumed above that N is not of the form $a^b (a, b \geq 2)$).

Then, consider the isomorphism $\mathbb{Z}_N \to \mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}$, where $b \in \mathbb{Z}_N$ maps to ($b \mod N_1, b \mod N_2$). Under this map, suppose c maps to (c_1, c_2).

Now, if a picked in Step 2(a) has a common factor with N, then we output 0 in Step 2(b). So, we condition on $\texttt{gcd}(a, N) = 1$ and show that for at least half of all such a's the square-root test in Step (d) fails. Indeed consider the following four set (all computations are in $\mathbb{Z}_N$ or equivalently in $\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}$).

$$
\begin{aligned}
S(+1,+1) &: \{a \in \mathbb{Z}_N^* : \texttt{gcd}(a, N) = 1, \text{ and } a^{(N-1)/2} \text{ maps to } (+1,+1)\} \\
S(-1,-1) &: \{a \in \mathbb{Z}_N^* : \texttt{gcd}(a, N) = 1, \text{ and } a^{(N-1)/2} \text{ maps to } (-1,-1)\} \\
S(-1,+1) &: \{a \in \mathbb{Z}_N^* : \texttt{gcd}(a, N) = 1, \text{ and } a^{(N-1)/2} \text{ maps to } (-1,+1)\} \\
S(+1,-1) &: \{a \in \mathbb{Z}_N^* : \texttt{gcd}(a, N) = 1, \text{ and } a^{(N-1)/2} \text{ maps to } (+1,-1)\}
\end{aligned}
$$

We claim that all these sets have the same size. It will follow, that the probability that N passes the square-root-of-unity test is at most (1/2) in any iteration. Then it follows that for N not prime, Pr[Error] $\leq (1/2)^5$.

We first observe that all sets are non-empty. Clearly, 1 maps to $(+1,+1)$, -1 maps to $(-1,-1)$. As $c^{(N-1)/2} = -1 \mod N$, we have $c_1^{(N-1)/2} = -1 \mod N_1$ and $c^{(N-1)/2} = -1 \mod N_2$. Thus, $(c_1^{(N-1)/2}, 1) = (-1,+1)$ and $(1, c_2^{(N-1)/2}) = (+1,-1)$. Hence, $c_1 \in S(-1,+1)$ and $c_2 \in S(+1,-1)$.

Now, note that $S(+1,+1)$ is a subgroup of $\mathbb{Z}_N^*$, and the other three sets are its cosets. Thus, all sets have the same size. □

Running time: We perform computations of the form $a^b \mod N$ using repeated squaring. Thus, all steps can be executed in polynomial time.

4 Roots of quadratic equations, mod p

Input: A polynomial of the form $A(X) = X^2 + bX + c \mod p$ an odd prime.
Output: A factorization of $A(X) = (X - \alpha)(X - \beta)$, or "$A(X)$ is irreducible".

We can check if $A(X)$ has the form $(X - \alpha)^2$, by checking if $b/2$ is a root of $A(X)$. So in the following we only distinguish between polynomials with distinct roots, and irreducible polynomials.

1. Pick numbers $r \in \{1, \ldots, p-1\}$ and $s \in \{0, \ldots, p-1\}$, uniformly at random.
2. Consider the quadratic polynomial $B(X) = A(rX + s)$.
3. Compute $X^{(p-1)/2} - 1 \mod B(X)$. The result is a polynomial of degree at most 1. If it is of the form $kX + \ell$; check if $\alpha = -\ell/k$ and $\beta = -b - \ell/k$ are roots of $A(X)$. Otherwise, output "$A(X)$ is irreducible".

Suppose $A(X)$ is of the form $(X-\alpha)(X-\beta)$, for distinct $\alpha, \beta \in \mathbb{F}_p$. Note that the roots of B are $(\alpha-s)/r$ and $(\beta-s)/r$. Now, $c = 1/r$ is uniformly distributed in $\{1, 2, \ldots, p-1\}$, and for each choice of r, when s is chosen at random, $d = s/r$ is uniformly distributed over $\{0, 1, \ldots, p-1\}$. Thus, the two roots of B are: $\gamma = c\alpha + d$ and $\delta = c\beta + d$, where (c, d) ranges over $(1, \ldots, p-1) \times (0, \ldots, p-1)$; then, (γ, δ) ranges over all $p(p-1)$ pairs of distinct elements in $\mathbb{F}_p$. In particular, the probability that only one of them is the root of $X^{(p-1)/2} - 1$ is precisely:

$$= [2 \cdot \frac{(p-1)}{2} \cdot \frac{(p+1)}{2}]/[p(p-1)] = (p+1)/(2p) > 1/2.$$

Whenever, this happens, we recover γ or δ by solving the linear equation.

The main computation $X^{(p-1)/2} \mod B(X)$ can be performed by repeated squaring in polynomial time in n (the bit-length of p).

5 The Goldreich-Levin list-decoding algorithm

Recall the problem from HW 3: We are given a parity function on n boolean variables: $\chi_S : \{0,1\}^n \to \{0,1\}$, where $S \subseteq [n]$, i.e., $\chi_S = \sum_{i \in S} x_i \mod 2$. We have access to a function $f : \{0,1\}^n \to \{0,1\}$ that computes χ_S for at least a $1/2 + \varepsilon$ fraction of the inputs, i.e., $\Pr[f(x) = \chi_S(x)] \geq \frac{1}{2} + \varepsilon$, for some constant $\varepsilon > 0$. Goal was to determine S.

Idea 1: $i \in S$ iff $\chi_S(e_i) = 1$, where $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ has a 1 only in its i-th coordinate.

Note that $\chi_S(x) = f(x + e_i) + f(x)$ with probability at least 2ε, where x is a uniformly chosen string in $\{0,1\}^n$. So, if $\varepsilon \geq 1/4 + \delta$, then we have a randomized estimator for $\chi_S(x)$ that is correct with probability at least $1/2 + 2\delta$. To compute $\chi_S(e_i)$, we use this estimator t times independently and take the majority vote. Then, the probability that $\chi_i(S)$ is decoded incorrectly is at most $\exp(-2(2\delta)^2 t)$.

We take, $t = (1/\delta^2) \log n$ and ensure that $\Pr[\text{Error}] < 1/n^2$. Thus, we recover S with probability $1 - 1/n$. This was the homework problem.

Unfortunately, we do not have such a large ε. In fact, for $\varepsilon < 1/4$, more than one χ_S may be compatible with f. So it is unreasonable to expect to recover S exactly. It turns out, however, that

there are at most $O(1/\varepsilon^2)$ such S that agree with a fixed f for a fraction $1/2+\varepsilon$ of the inputs. Can we efficiently produce the list of all such functions given oracle access to f? This is the list-decoding problem.

Idea 2: The difficulty we have now arises from the union bound to take care of errors from two probes. If somehow we could guess the values of one of the probes in a systematic way, then we need to account for the error from only one probe, not both.

1. Fix an S that agrees with f on at least $1/2+\varepsilon$ fraction of the inputs $x \in \{0,1\}^n$.
2. Pick vectors $v_1, v_2, \ldots, v_t$ at random from $\{0,1\}^n$. Guess the value of χ_S on $v_1, v_2, \ldots, v_t$, say $(a_1, a_2, \ldots, a_t)$, each in $\{0,1\}$. Extend the guess to every vector in the subspace spanned by $v_1, \ldots, v_t$. Let the vectors in this subspace be v_T for $T \subset [t]$ and let the correspond value be a_T, that is, $a_T = \sum_{i \in T} a_i$.
3. Now, to estimate $\chi_S(e_i)$, we take the majority of the values $\{a_T + f(v_T + e_i) : T \text{ a non-empty subset of } [t]\}$; call this majority value $G(i)$. By computing $G(1), \ldots, G(n)$ using this strategy, for each choice of $a = (a_1, a_2, \ldots, a_t)$, we output a set S_a. Let $P = S_a : a \in \{0,1\}^t$.

Claim 5.1. $\Pr[S_a \in P] \geq 1 - n/(\varepsilon^2(2^t - 1))$.

We will choose t about $\log(n/\varepsilon^4)$ and claim that particular, the probability that some set S compatible with f fails to appear in P is at most $1/10$.

Proof. Note that the vectors in $\{v_T : T \subset [t], T \text{ not empty}\}$ are pairwise independent, and each is uniformly distributed in $\{0,1\}^n$. Thus, the values $b_T = \chi_S(v_T) + f(e_i + v_T)$ are pairwise independent and $\Pr[b_T = \chi_S(e_i)] \geq 1/2 + \varepsilon$. We we have that the number of agreements among the $N = 2^t - 1$ samples has expectation $\geq Np = N/2 + \varepsilon N$ and variance $= Npq \leq N$. By Chebyshev's inequality,

$$\Pr[\text{error in decoding } \chi_S(e_i)] \leq N/(\varepsilon N)^2 \leq 1/(\varepsilon^2 N).$$

The probability that $\chi_S(e_i)$ is decoded incorrectly for some i is at most $n/(\varepsilon^2 N)$.

Now, among our guesses for $a = (a_1, a_2, \ldots, a_t)$ is the choice $a = (\chi_S(v_1), \chi_S(v_2), \ldots, \chi_S(v_t))$. Thus, Pr[S is not in P] $\leq n/(\varepsilon^2 N)$.

We set $t = \log(10n/\varepsilon^4 + 1)$ (so that $N = 10n/\varepsilon^4$), and conclude that the probability that P misses an S for which χ_S is compatible with f is at most $(1/\varepsilon^2)(n/\varepsilon^2)(1/N) = (1/\varepsilon^2)(n/\varepsilon^2)\varepsilon^4/(10n) \leq 1/10$. □

Now, the set P might contain some spurious sets. For each candidate solution, we verify that it agrees with f by picking $10 \log n$ vectors v and checking that the candidate and f agree on at least a $1/2 + \varepsilon$ fraction of them.

Thus, overall, we compute f on $2^t - 1 + 10 \log n$ vectors. The running time of the algorithm is $\text{poly}(n, 1/\varepsilon)$.

Algorithm 1: Algorithm summary (following a suggestion of Dhruv in class)

Input: Function $f : \{0,1\}^n \to \{0,1\}$; we have oracle access to f.

Output: Short list of sets P, that with high probability contains every S for which $\Pr[f(x) = \chi_S(x)] \geq 1/2 + \varepsilon$.

```
P ← ∅.
Pick v_1, ..., v_t at random from {0,1}^n.
Construct (v_T : T ⊂ [t], T ≠ ∅) where v_T = Σ_{i∈T} v_i. There are 2^t such vectors in the
  sequence (we don't assume they are distinct).
for each choice of a = (a_1, a_2, ..., a_t) do
    Construct (a_T = Σ_{i∈T} a_i : T ⊂ [t], T not empty).
                                        /* a_T are our guesses for χ_S(v_T).  */
    Let S ← ∅.
    for i = 1, 2, ..., n do
        Compute c_i = Majority(b_T = a_T + f(v_T + e_i) : T ⊂ [t], T ≠ ∅).
        if c_i = 1 then
            Add i to S.
    Add S to P (note that for each choice of a, we add one set S to P.
return P
```