## 1. Коррекция гомоморфизма

Пусть $f\colon U \to V$ — гомоморфизм абелевых групп, и в таблице (или схеме, его вычисляющей), испорчено $\varepsilon$-доля значений, причём $\varepsilon < 1/4$. Тогда с вероятностью $1-2\varepsilon$ можно за два запроса к таблице восстановить значение в любой точке.

В самом деле, $f(u) = f(r+u) - f(r)$, и для случайного $r$ с вероятностью как минимум $1-2\varepsilon$ оба значения справа правильны (и $r$, $x+r$ равномерно распределены). Повторяя это и беря большинство, можно сделать вероятность ошибки экспоненциально малой (по числу повторений).

В частности, в коде Адамара можно корректировать $\varepsilon < 1/4$ ошибок таким образом (что как раз соответствует кодовому расстоянию $1/2$).

## 2. Тест на гомоморфизм

Пусть $f\colon U \to V$ — отображение абелевых групп, и $f(x+y) = f(x) + f(y)$ для большинства пар $(x, y)$ (кроме доли $\varepsilon$). Тогда $f$ является $O(\varepsilon)$-близким к некоторому (точному) гомоморфизму.

Будем восстанавливать как раньше: для данного $u$ брать случайное $r$ и вычислять $f(r+u) - f(r)$. Лемма: по крайней мере $(1-2\varepsilon)$-доля значений одинаковы. В самом деле, для случайных независимых $r$ и $s$ равенство

$$f(r+u) - f(r) = f(s+u) - f(s) \qquad (*)$$

имеет место с вероятностью как минимум $1-2\varepsilon$, так как его можно переписать в виде

$$f(r+u) + f(s) = f(s+u) + f(r),$$

а обе части с вероятностью $1-\varepsilon$ равны $f(r+u+s)$, так как обе пары $(r+u, s)$ и $(s+u, r)$ равномерно распределены в $U^2$. Теперь, раз $(*)$ выполнено с вероятностью $1-2\varepsilon$, то при каком-то $r$ оно выполнено с той же вероятностью, и потому $(1-2\varepsilon)$-доля значений одинаковы. Считая $\varepsilon < 1/2$ (это законно, так как у нас $O(\varepsilon)$), обозначим через $F(u)$ это наиболее частое значение. Осталось проверить две вещи:

(1) $F$ — точный гомоморфизм, то есть $F(u+v) = f(u) + f(v)$ для всех $u$ и $v$. Для этого возьмём треугольник, иллюстрирующий сложение векторов $u$ и $v$ и сдвинем его на $r$:

$$f(r+u+v) - f(r) =$$
$$= [f((r+u)+v) - f(r+u))] + [f(r+u) - f(r)]$$

Эти три выражения равны $F(u+v)$, $F(v)$ и $F(u)$ соответственно с вероятностью (по $r$) не менее $(1-2\varepsilon)$ каждая, поэтому при $6\varepsilon < 1$ равенство $F(u+v) = F(u) + F(v)$ выполнено с положительной вероятностью по $r$, но оно от $r$ не зависит, значит, выполнено всегда.

(2) $F$ близко к $f$. В самом деле, для случайной пары $(u, r)$ равенство $F(u) = f(r+u) - f(r)$ верно с вероятностью ошибки $2\varepsilon$, а $f(u) = f(r+u) - f(r)$ с вероятностью $\varepsilon$, поэтому $f(u) = F(u)$ с вероятностью как минимум $1-3\varepsilon$ (и можно считать только по $u$, так как это событие от $r$ не зависит).

(Другое рассуждение: для тех $u$, для которых $f(u) \neq F(u)$, по крайней мере для половины всех $r$ нарушается равенство $f(u) = f(u+r) - f(r)$, поэтому вероятность нарушения этого равенства при случайных $u$ и $r$, которая не больше $\varepsilon$, не меньше половины доли тех $u$, при которых $f(u) \neq F(u)$. Получаем чуть лучшую оценку $1-2\varepsilon$.)

## 3. Self-testing/correcting

Это можно интепретировать так: если нам дают микросхему, якобы вычисляющую некоторый гомоморфизм, то можно сначала её протестировать и убедиться, что она близка к гомоморфизму в смысле теста предыдущего раздела, а затем ответить на некоторое число вопросов типа «найти значение гомоморфизма в такой-то точке» с помощью корректирующего алгоритма из первого раздела.

При этом выполнены два свойства:

(1) Если микросхема без ошибок (точный гомоморфизм), то она наверняка пройдёт тест, и выданные на втором этапе значения будут соответствовать этому гомоморфизму.

(2) Какова бы ни была микросхема, с большой вероятностью произойдёт одно из двух: либо она будет отвергнута, либо выданные на втором этапе значения все соответствуют ближайшему к ней точному гомоморфизму.

(Число проб на шаге тестирования полиномиально зависит от числа вопросов на втором шаге, потому что при увеличении числа вопросов надо соответственно уменьшать вероятность ошибки.)

## 4. Многочлены малой степени

То же самое можно сделать с многочленами малой степени (меньше некоторого $d$) над конечным полем вычетов $\mathbb{F}_p$; при этом мы считаем $p \gg d$ (чтобы многочлены отличались от просто функций; конкретно нам будет достаточно $p > 2d$). При этом будем рассматривать многочлены нескольких переменных (в терминах теории кодирования мы переходим от кода Адамара к кодам Рида – Маллера), но сначала напомним свойства многочленов с одной переменной.

Многочлен степени меньше $d$ от одной переменной может принимать любые значения в $d$ точках, но в $(d+1)$-точке уже появляется соотношение. Если $d+1$ точек $x_0, \ldots, x_d$ образуют арифметическую прогрессию, то это соотношение можно выписать явно:

$$P(x_0) - C_d^1 P(x_1) + C_d^2 P(x_2) - \ldots = 0$$

(последний член имеет коэффициент $\pm 1$ в зависимости от чётности $d$). Это соответствует тому, что $d+1$-ые разности равны нулю.

Для многочленов от $n$ переменных мы рассматриваем суммарную степень по всем переменным, поэтому сужение на любую прямую в $\mathbb{F}^n$ будет многочленом не большей степени (от одной переменной), и можно применять то же соотношение вдоль любой арифметической прогрессии в $\mathbb{F}^n$.

Это можно делать и для коррекции (восстанавливаем значение в точке, выпуская из этой точки прогрессию со случайной разностью), и для тестирования (выпуская из случайной точки случайную прогрессию и проверяя соотношение). Точнее:

(1) если функция $\varepsilon$-близка к многочлену степени меньше $d$, и $d\varepsilon < 1/2$, то восстановление в любой точке даёт этот многочлен;

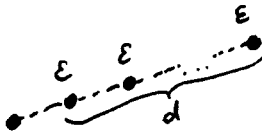
(2) если для какой-то функции тест удаётся с вероятностью $1-\varepsilon$, при этом $2d\varepsilon < 1/2$, то для любой начальной точки большинство восстановлений (по разным прогрессиям) даст один и тот же результат;

(3) если $2d(d+1)\varepsilon + d\varepsilon < 1$, то восстановленная таким образом (по большинству) функция проходит тест всегда;

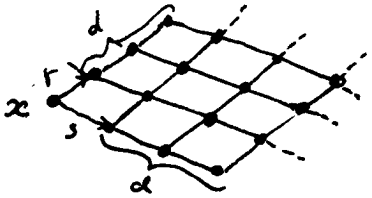
наконец (это уже не связано с вероятностью)

(4) если какая-то функция проходит тест вдоль любой арифметической прогрессии длины $d+1$, то она является многочленом степени меньше $d$ по совокупности переменных.

Доказательство. (1) Если $p > d$, то числа $1, \ldots, d$ взаимно просты с $p$, и потому при случайном шаге прогрессии и фиксированном начале любой член прогрессии равномерно распределён по всему пространству, поэтому отличается от «истинного» с вероятностью не более $\varepsilon$. Поэтому все члены «истинны» с вероятностью не менее $1-d\varepsilon > 1/2$, так что восстановленное значение истинно с вероятностью больше половины.

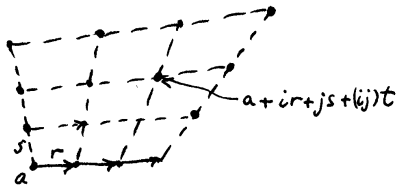



(2) Пусть мы восстанавливаем значение в точке $x$ с помощью двух арифметических прогрессий со случайными шагами $r$ и $s$. С какой вероятностью два восстановленных значения будут согласованы?




Это заведомо так, если тест проходит на каждой из $2d$ пунктирных прямых (линейная комбинация значений, проходящих тест, также проходит тест). Каждая из этих прямых имеет равномерно распределённое начало (см. выше) и независимый от него случайный шаг, так что вероятность не пройти тест не более $\varepsilon$. Если $1 - 2d\varepsilon > 1/2$, то в большинстве случаев все тесты проходят и потому $r$- и $s$-восстановления согласованы; далее рассуждаем как раньше.

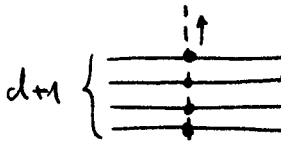
(3) Попытаемся провести тест для восстановленных значений в точках арифметической прогрессии с началом $a$ и шагом $r$ (нижняя прямая на рисунке) и будем использовать шаги, сами образующие прогрессию (шаги $s$, $s+t$ и так далее: узел номер $(i,j)$ равен $a + ir + js + (ij)t$).




При этом $a$ и $r$ фиксированы и от нас не зависят, а $s$ и $t$ мы выбираем случайно и независимо. Каждая из восстановительных прогрессий (с шагами $s, s+t, \ldots$) даёт восстановленное по большинству значение с вероятностью ошибки $2d\varepsilon$. Значит, все они дают это значение с вероятностью ошибки $2d(d+1)\varepsilon$. Допуская ещё $d\varepsilon$ ошибки, мы можем предполагать, что по каждой из $d$ «горизонталей» (прогрессии с началом $a+js$ и шагом $r+jt$) тест проходит. (Здесь важно, что эта прогрессия получается случайной с независимым началом и шагом: именно для этого нам нужно $t$.) А если это так, то проходит и тест по нижней прямой (см. выше).

(4) Для многочленов от одной переменной: проведём интерполяционный многочлен через точки $0, 1, 2, \ldots, d-1$. Далее заметим, что раз тест выполнен в точках $0, 1, 2, \ldots, d$, то этот многочлен совпадает с нашим в $d$; так как тест выполнен в точках $1, 2, \ldots, d+1$, то совпадает и в $d+1$ и так далее.

Для двух переменных: на каждой горизонтали и вертикали (и вообще вдоль любой прямой) у нас многочлен степени меньше $d$. Взяв $d$ горизонталей (параллельных оси $x$), получим $d$ многочленов от $x$. Значения этих многочленов можно использовать для интерполяции вдоль вертикалей, и мы получаем, что функция есть многочлен степени меньше $d$ *по каждой переменной.*



Осталось проверить, что *суммарная* степень меньше $d$. Разложим многочлен на однородные, включив в $p_i$ члены суммарной степени $i$:

$$p(x,y) = \sum_{i=0}^{2d-2} p_i(x,y);$$

Тогда

$$p(tx,ty) = \sum_{i=0}^{2d-2} p_i(x,y)\,t^i;$$

при любых $x, y$ как многочлен от $t$ имеет степень меньше $d$; предполагая, что $p > 2d$, мы можем от равенства многочленов перейти к равенству коэффициентов, поэтому $p_i(x,y)$ при $i \geqslant d$ равно 0 и эти члены суммы можно выбросить, получив многочлен суммарной степени меньше $d$.

Переход от двух переменных к трём (и далее) происходит аналогично: сначала мы получаем многочлен, у которого суммарная степень по $x, y$, а также степень по $z$ меньше $d$, а потом доказываем, что и суммарная степень по трём переменным меньше $d$.

## 5. Мультилинейные многочлены

Многочлен от $n$ переменных называют *мультилинейным*, если степень по каждой переменной не выше 1. Такой многочлен задаётся $2^n$ своими коэффициентами. Использование мультилинейных многочленов для кодирования основано на таком факте (в котором мы считаем $\mathbb{B}$ частью поля $\mathbb{F}$): *любая функция* $\mathbb{B}^n \to \mathbb{F}$ *однозначно продолжается до мультилинейного многочлена* $\mathbb{F}^n \to \mathbb{F}$.

(Индукция: разобьём куб на две параллельные грани, на каждой грани такое продолжение однозначно по предположению, а в перпендикулярном направлении мы должны восстановить линейную функцию по значениям в двух соседних точках.)

Мультилинейный многочлен можно рассматривать как код его ограничения на $\mathbb{B}^n$. При этом кодовое расстояние не меньше $1 - (n/|F|)$ от длины кода: два мультилинейных многочлена совпадают не более чем в $n/|F|$ доле точек.

(Индукция по $n$: многочлен

$$p_0(x_2, \ldots, x_n) + x_1 p_1(x_2, \ldots, x_n)$$

при каждых $x_2, \ldots, x_n$ есть линейная функция от $x_1$; эта линейная функция нулевая не более чем в доле $(n-1)/|F|$ случаев по предположению индукции, а в остальных случаях у неё не более одного корня, всего $1/|F|$.)

Для мультилинейных многочленов также существуют способы коррекции и тестирования, но нам достаточно тестирования суммарной степени (которое более наглядно).

## 6. MIP и PCP(poly, poly)

**MIP**: интерактивные доказательства, где полиномиальный вероятностный проверяющий (verifier) общается с двумя доказывающими, разделёнными друг от друга. Говорят, что язык $L$ принадлежит классу MIP, если существует такой алгоритм проверки, что

- При $x \in L$ существует пара стратегий для доказывающих, которая гарантирует ответ «да»;
- При $x \notin L$ никакая пара стратегий для доказывающих не позволяет получить ответ «да» с вероятностью больше $1/2$.

Как обычно, доказывающих можно считать детерминированными (но уже не обязательно в PSPACE), и последовательное повторение $k$ раз превращает вероятность ошибки в $1/2^k$ (тут важно, что последовательное — с параллельным дело сложнее).

**PCP**(poly, poly): в качестве доказательства предъявляется таблица булевой функции от полиномиального числа переменных (микросхема с полиномиальным числом входов); проверяющий вероятностный и полиномиальный и имеет доступ к этой функции как к оракулу.

- При $x \in L$ существует функция («доказательство»), которая гарантирует ответ «да»;
- При $x \notin L$ никакая функция не позволяет получить ответ «да» с вероятностью больше $1/2$.

(Здесь $1/2$ можно уменьшить до $1/2^k$ при $k$ повторениях теста на одном и том же доказательстве.)

Эти два определения задают один и тот же класс языков.

В самом деле, стратегию доказывающих в MIP-определении можно представить в виде функции (точнее, набора функций: первый ответ как функция

первого вопроса, второй ответ как функция первых двух вопросов и так далее; набор функций очевидно соединяется в функцию). Отсутствие взаимодействия между проверяющими гарантируется автоматически (функции фиксированы).

В другую сторону: если бы мы спрашивали доказывающего только раз, то его стратегия в точности была бы функцией, но функцию мы можем запрашивать несколько раз, и если эти вопросы задаются доказывающему, он может жульничать и, скажем, выбирать второй ответ в зависимости от первого вопроса. Поэтому мы используем второго доказывающего для проверки случайно выбранного ответа первого. Второму задаётся единственный вопрос и потому его поведение есть функция; если первый отклоняется от этой функции, то будет разоблачён с вероятностью не меньше $1/k$, где $k$ — число вопросов. Повторяя это $m$ раз, можно сделать $(1-1/k)^m < 1/2$, и диалог остаётся полиномиальным.

## 7. MIP $\subset$ NEXPTIME

В самом деле, доказательство (функция) для входов размера $n$ имеет размер $2^{\mathrm{poly}(n)}$. Когда оно предъявлено, за время $2^{\mathrm{poly}(n)}$ можно промоделировать все варианты случайных битов и вычислить вероятность ответа «да» при таком доказательстве. (Это рассуждение сохраняет силу, и если не требовать разрыва между $1/2$ и $1$ в вероятностях, так что на самом деле класс останется тем же и без такого разрыва, как следует из следующего результата.)

## 8. NEXPTIME $\subset$ MIP

Согласно предыдущему разделу, мы можем строить алгоритм проверки оракула-доказательства.

Это делается в два шага. Сначала мы сводим вопрос о попадании в NEXPTIME-язык к вопросу о том, обращается ли некоторый многочлен небольшой степени на $\mathbb{F}^n$ в тождественный нуль на $\mathbb{B}^n$ (этот раздел). Затем (в следующем разделе) мы строим протокол доказательства факта обращения в нуль для такого многочлена.

Пусть $L$ — язык из NEXPTIME. Тогда доказательство принадлежности некоторого слова $x$ к $L$ можно записать в виде протокола допускающей работы недетерминированной машины Тьюринга. Такой протокол представляет собой квадратную таблицу размера $2^{\mathrm{poly}(n)} \times 2^{\mathrm{poly}(n)}$ (где $n$ — длина входа $x$) с некоторыми свойствами.

А именно, эта таблица должна иметь правильное начало и конец, а также быть локально правильной. Представляя таблицу как булеву функцию от полиномиального числа переменных, видим, что правильность начала и конца легко проверить, а вот локальная правильность требует экспоненциального числа проверок, и в этом проблема.

Несколько замечаний:

(1) мы можем считать, что эта таблица (соответствующая ей булева функция от $m = \mathrm{poly}(n)$ аргументов) дана нам не только сама по себе, но и с полиномиальным продолжением на $\mathbb{F}^m$ (с $\mathbb{B}^m$), где $m$ и размер поля полиномиальны от $n$. Заметим, что продолжение не сильно увеличивает размер таблицы: от $2^m$ мы переходим к $|\mathbb{F}|^m$, что соответствует увеличению показателя степени в $\log_2 |\mathbb{F}|$ раз.

(2) техника тестирования и коррекции позволяет предполагать, что чудесным образом гарантирована полиномиальность этого продолжения (что оно является полиномом малой суммарной степени): мы инкапсулируем реальную таблицу в механизм предварительного тестирования с последующей коррекцией при запросах.

Теперь надо разобраться с условием локальной корректности этой таблицы. Удобно считать, что в каждой клетке таблицы стоит символ конечного алфавита (а не бит), и что условие локальной корректности связывает соседние символы по горизонтали или вертикали. (При обычном представлении протокола работы машины условие относится к большей группе клеток,

но можно закодировать содержимое нескольких соседних клеток в одной.)

Символы конечного алфавита можно заменить наборами битов, так что таблица в целом будет представлена несколькими булевыми функциями $S^1, \ldots, S^c$. Значения $S^i(x_1 \ldots x_k, y_1 \ldots y_k)$ (при $i = 1, \ldots, c$) представляют собой содержимое клетки таблицы, имеющей координаты $x_1 \ldots x_k$ по горизонтали и $y_1 \ldots y_k$ по вертикали (в двоичной записи). Значение $c$ зависит от алфавита и числа состояний машины Тьюринга, но не от входа, так что $c$ мы считаем константой.

Условия согласованности: если

$$(x_1 \ldots x_k, y_1 \ldots y_k) \text{ и } (x'_1 \ldots x'_k, y'_1 \ldots y'_k)$$

являются координатами соседних клеток, то биты

$$S^1(x_1 \ldots x_k, y_1 \ldots y_k), \ldots, S^c(x_1 \ldots x_k, y_1 \ldots y_k),$$
$$S^1(x'_1 \ldots x'_k, y'_1 \ldots y'_k), \ldots, S^c(x'_1 \ldots x'_k, y'_1 \ldots y'_k)$$

определённым образом согласованы. И условия отличия координат на единицу, и условия согласованности в протоколе можно записать формулой полиномиального размера, и интерпретируя логические

операции в этой формуле как арифметические, мы записываем условие согласованности так: *некоторый многочлен небольшой степени, значение которого в любой точке $\mathbb{F}^m$ мы можем вычислить, обращается в нуль на булевом кубе $\mathbb{B}^m$.*

## 9. Доказательство обращения в нуль

Итак, осталось разобрать такую задачу. Есть многочлен $P(x_1, \ldots, x_m)$ небольшой степени от $m$ переменных в достаточно большом поле $\mathbb{F}_p$, представленный в виде оракула. Доказывающий (он имеет неограниченные вычислительные ресурсы) знает, что этот многочлен обращается в нуль на булевом кубе $\mathbb{B}^m \subset \mathbb{F}^m$ и хочет убедить в этом проверяющего (полиномиальный вероятностный алгоритм, допускается небольшая вероятность ошибки). При этом проверяющий заранее знает, что это действительно многочлен небольшой степени.

Как же доказывающему убедить проверяющего, что $2^m$ элементов поля $P(b_1, \ldots, b_m)$ для любых $b_1, \ldots, b_m \in \mathbb{B}$ все равны нулю? Если бы эти значения были коэффициентами мультилинейного многочлена, то достаточно было бы вычислить значение этого многочлена в одной случайной точке: если он на самом деле не равен тождественно нулю, то с большой вероятностью это обнаружится при такой проверке. Рассмотреть такой многочлен, конечно, нам никто не запрещает, но надо уметь вычислять его значения (в произвольной точке), а мы умеем вычислять только коэффициенты. Оказывается, что при помощи дополнительной информации от доказывающего можно перейти от одного к другому.

Для начала запишем многочлен, о котором идёт речь (назовём его переменные $v_1, \ldots, v_n$):

$$\sum_{b_1,\ldots,b_n \in \mathbb{B}} P(b_1, \ldots, b_n) v_1^{b_1} \ldots v_n^{b_n}.$$

Возведение в степень здесь встречается лишь с показателем 0 и 1, поэтому можно считать это обозначение сокращением для многочлена:

$$v^b := (1 - b) + bv$$

Тем самым это выражение под знаком суммы становится многочленом от $b_i$ и $v_i$, и проверяющий (с помощью доказывающего, но без доверия к нему) должен вычислить значение суммы в случайной точке $v_1, \ldots, v_n$.

Выделим суммирование по $b_1$ в отдельный (внешний) шаг и скажем, что интересующее нас выражение равно $Q(0) + Q(1)$, где

$$Q(b_1) = \sum_{b_2,\ldots,b_m \in \mathbb{B}} P(b_1, \ldots, b_n) v_1^{b_1} \ldots v_n^{b_n}.$$

Наше соглашение о смысле обозначения $v^b$ позволяет утверждать, что $Q(b)$ — многочлен небольшой степени от одной переменной $b$ (которая теперь уже принимает не только булевы значения).

Доказывающий может предъявить многочлен $Q$ (он от одной переменной и небольшой степени, так что коэффициентов немного), и тогда проверяющий сам может вычислить $Q(0) + Q(1)$. Вопрос только в том, как доказать, что многочлен $Q$ указан правильно. Поскольку он небольшой степени, достаточно проверить совпадение в одной случайной точке (элементе поля) $r_1$.

Таким образом, мы свели исходную задачу к аналогичной задаче с меньшей суммой: проверяющий с помощью доказывающего должен найти значение

$$Q(r_1) = \sum_{b_2,\ldots,b_m \in \mathbb{B}} P(r_1, b_2, \ldots, b_n) v_1^{r_1} v_2^{b_2} \ldots v_n^{b_n}.$$

Повторим уже использованный приём: заметим, что $Q(r_1)$ равно сумме двух значений $Q_{r_1}(0) + Q_{r_1}(1)$, где

$$Q_{r_1}(b_2) = \sum_{b_3,\ldots,b_m \in \mathbb{B}} P(r_1, b_2, \ldots, b_n) v_1^{r_1} v_2^{b_2} \ldots v_n^{b_n}.$$

Если доказывающий сообщит коэффициенты этого многочлена, то проверяющий сам сможет вычислить его значение в нуле и единице, так что останется убедиться в правильности коэффициентов, для чего сверить значения в случайной точке $r_2$.

И так далее. В итоге все суммы раскроются, и останется убедиться в правильности значения

$$P(r_1, \ldots, r_n) v_1^{r_1} v_2^{r_2} \ldots v_m^{r_m}$$

для случайных элементов $r_1, \ldots, r_m, v_1, \ldots, v_m$ поля $\mathbb{F}$. (Напомним, что возведение в степень понимается в описанном выше смысле, так что под знаком суммы стоит многочлен небольшой степени. Собственно, поскольку $r_i$ теперь не нули с единицами, а произвольные элементы поля, никакого иного смысла и нет.)

Ещё можно прикинуть, какие тут получаются степени многочленов и какого размера надо брать поле, чтобы суммарная вероятность ошибки была мала. Суммирование (и это очень важное место!) не увеличивает степень, степень $P$ была ограничена полиномом от $m$ (который сам есть полином от $n$), поскольку степень полинома, соответствующего формуле, не превосходит её длины. Число шагов, где мы можем ненароком попасть в неудачную точку, тоже полиномиально, так что достаточно сделать вероятность ошибки в каждом случае полиномиально малой — для чего надо взять поле полиномиального размера.