Что значит, что задача проста для типичного входа?

Пусть фиксирована некоторая задача разрешения, то есть, некоторая функция A из множества всех двоичных слов Ξ со значениями 0,1.

Игра между Загадчиком (Challenger) и Отгадчиком (Solver) состоит из двух ходов. Первым ходит З, выбирая некоторый вопрос $x \in \Xi$, затем ходит О, указывая некоторое $y = 0, 1$. О выиграл, если $y = A(x)$.

В терминах этой игры можно дать определение классов P, BPP, а также интересующего нас класса простых в типичном случае задач.

По условию О имеет выигрышную стратегию в этой игре, однако эта стратегия не обязательно полиномиально вычислима (в обычном смысле, как функция $x \mapsto y$). Если О имеет полиномиально вычислимую стратегию, то говорят, что задача разрешима за полиномиальное время.

Пусть p положительное число от 0 до 1. Мы говорим, что задача разрешима с вероятностью успеха p, если О имеет полиномиально вычислимую вероятностную стратегию, гарантирующую выигрыш с вероятностью не менее p. Другими словами, если существует вероятностный алгоритм R, который на вход получает x и останавливается за полиномиальное от длины x время (при любых исходах бросания монеток), при этом для всех x вероятность события $R(x) = A(x)$ не меньше p.

Любая задача всегда разрешима с вероятностью успеха $1/2$ (стратегия О выдает случайный бит). Если же $p > 1/2$, то это уже не так, и мы получаем класс BPP (один и тот же, независимо от p).

Основным в теории сложности для типичного случая является ограничение вычислительных возможностей Загадчика. Может оказаться, например, что у Отгадчика имеется полиномиальный по времени алгоритм, который правильно решает задачу за исключением очень редкого множества входов, причем входы из этого множества очень трудно найти (то есть требуется много времени). Тогда такая полиномиальная стратегия О будет обыгрывать любого ограниченного во времени Загадчика. Сказанное требует уточнения, поскольку следует определить, чем именно ограничивается время Загадчика.

Сложность для типичного входа

Будем считать, что оба игрока получают на вход натуральный параметр n, называемый параметром безопасности. Будем рассматривать вероятностные стратегии Загадчика, вычислимые за полиномиальное от n время: стратегия получает на вход число n в унарной записи и выдает на выход некоторое $x \in \Xi$. (Мы считаем, что зафиксирована некоторая массовая проблема, не обязательно являющаяся проблемой разрешения.)

Полиномиальные по времени вероятностные стратегии для Загадчика будем называть генераторами примеров. То есть генератор примеров — это полиномиальный по времени алгоритм, который на входе n (в унарной записи) выдает некоторый $x \in \Xi$. Для каждого n рассмотрим распределение вероятностей μ_n, получаемое на выходе этого алгоритма: $\mu_n(x)$ есть вероятность того, что алгоритм на входе n выдаст x. Семейство μ_n распределений вероятностей, для которых существует генератор примеров, будем называть "генерируемыми семействами распределений". (В литературе очень похожее понятие называется "samplable distributions". Распределение μ (а не семейство распределений) называется samplable, если существует вероятностный алгоритм без входа, который останавливается за время ограниченное полиномом от длины своего результата и распределение на выходах которого совпадает с μ.)

Полиномиальные по времени вероятностные стратегии Отгадчика называются вероятностными полиномиальными алгоритмами. Такой алгоритм получает на вход число n в унарной записи и вопрос x и выдает за полиномиальное (от n и длины x) время ответ y.

Понятие простой для типичного входа задачи можно уточнять двумя способами. Зафиксирум некоторую функцию $t(n)$ со значениями от 0 до 1 (желаемая вероятность правильного ответа).

Первый способ. Задача называется простой для типичного входа с уровнем доверия $t(n)$, если для любого генератора существует полиномиальная вероятностная стратегия З, выигрывающая в игре против этого генератора с вероятностью не менее $t(n)$ для всех n. Иными словами, для любого генерируемого семейства распределений μ_n существует полиномиальный вероятностный алгоритм R такой, что для всех n вероятность по распределению μ_n и по случайным бросаниям алгоритма события $R(1^n, x) = A(x)$ не меньше $t(n)$.

В этом определении слова "для всех n" можно заменить на слова "для почти всех n".

Второй способ получается из первого перестановкой кванторов. Задача называется *очень* простой для типичного входа с уровнем доверия $t(n)$, если существует полиномиальный вероятностный алгоритм R такой, что для любого генерируемого семейства распределений μ_n для *почти всех* n вероятность по распределению μ_n и по случайным бросаниям алгоритма события $R(1^n, x) = A(x)$ не меньше $t(n)$.

В этом определении слова "для почти всех n" уже нельзя заменить на слова "для всех n". Действительно, после такой замены утверждение означало бы, что для всех n, x вероятность правильного ответа $R(n, x)$ не меньше $t(n)$. (Фиксируем n и x. Сосредоточим всю вероятность всех распределений в семействе μ_k на этом конкретном x. Такое распределение μ_k генерируется за полиномиальное время, более того, за константное время, равное длине x.) Тогда если $t(n) > 1/2$ хотя бы для одного n, то задача принадлежала бы BPP: вероятностный алгоритм $R(n, \cdot)$ решал бы ее правильно с вероятностью $t(n)$. А если $t(n) \leq 1/2$ для всех n, утверждение было бы тривиально истинным (алгоритм R выдает случайный бит).

Для обоих определений получаемый класс простых задач может зависеть от функции $t(n)$, даже если $t(n)$ равно константе.

В этой статье мы примем первый способ определения простоты задачи для типичного случая. Но все равно остается большая свобода для выбора уровня доверия $t(n)$. Наиболее естественным мне представляется вариант определения, когда уровень доверия не фиксирован, а дается алгоритму решения в качестве дополнительного входа.

А именно, будем называть задачу простой для типичного входа, если для любого генерируемого семейства распределений μ_n существует вероятностный алгоритм R, который на вход получает два числа — n, k и слово x и за полиномиальное от $n+k+|x|$ время выдает 0 или 1, причем для всех n вероятность по распределению μ_n и по случайным бросаниям алгоритма R события $R(1^n, 1^k, x) = A(x)$ не меньше $1 - 1/k$.

Менее естественный вариант: для каждой постоянной функции $t(n) \equiv \varepsilon > 0$ рассмотрим класс простых задач с уровнем доверия $t(n)$ и пересечем все такие классы (по всем ε). То есть, будем называть задачу простой для типичного входа, если для любого положительного ε и для любого генерируемого семейства распределений μ_n существует полиномиальный вероятностный алгоритм R такой, что для всех n вероятность по распределению μ_n и по случайным бросаниям алгоритма R события $R(1^n, x) = A(x)$ не меньше $1 - \varepsilon$.

Еще менее естественный вариант: для каждой функции вида $t(n) = 1 - 1/\mathrm{poly}(n)$ рассмотрим класс простых задач с уровнем доверия $t(n)$ и пересечем все такие классы (по всем полиномам). То есть, будем называть задачу простой для типичного входа, если для любого полинома $q(n)$ и для любого генерируемого семейства распределений μ_n существует полиномиальный вероятностный алгоритм R такой, что для всех n вероятность по распределению μ_n и по случайным бросаниям алгоритма R события $R(1^n, x) = A(x)$ не меньше $1 - 1/q(n)$.

Сводимости, которые рассматриваются в этой работе, сохраняют простоту в первом и третьем смыслах, но к сожалению не сохраняют простоту во втором смысле. Поскольку нас будут интересовать не простые, а трудные проблемы, точнее NP-полные проблемы относительно рассматриваемых сводимостей, не так важно, какое из двух определений простоты выбрать.

Для любого из трех определений простоты, класс простых задач включает в себя BPP. Если задача проста в первом смысле, то она проста и во втором и в третьем смысле. Соотношение второго и третьего определений неясно.

Для определенности остановимся на первом определении и будем называть задачу простой для типичного входа, если для любого генерируемого семейства распределений μ_n существует вероятностный полиномиальный алгоритм R такой, что для всех n вероятность по распределению μ_n и по случайным бросаниям алгоритма R события $R(1^n, 1^k, x) = A(x)$ не меньше $1 - 1/k$.

Рандомизованные задачи

Пары (A, μ_n), состоящие из задачи A и некоторого генерируемого распределения μ_n на ее входах, называется рандомизированной задачей. Простота рандомизированной задачи определяется точно так же, как выше (только квантора существования по распределению вероятностей нет). То есть, задача является простой для типичного входа (в одном из трех смыслов) тогда и только тогда, когда все ее рандомизации просты (в том же смысле).

Пример: пусть задача состоит в определении, содержит ли данный граф гамильтонов цикл, а распределение μ_n равномерно на графах с n вершинами (каждое из возможных $n(n-1)/2$ ребер присутствует с вероятностью $1/2$ и все ребра независимы). Эта задача является простой для типичного входа (я не знаю, как это доказать). Более того, она является простой *в среднем*.

Рандомизованная задача (A, μ_n) называется простой в среднем, если существует (возможно неполиномиальный) вероятностный алгоритм R, для которого выполнено следующее. (1) Алгоритм $R(n, x)$ для любых

n, x и для любых результатов бросаний правильно решает задачу A на входе x (но возможно за большое время). (2) Среднее (по распределению μ_n) время работы $R(n, x)$ ограничено полиномом от n.

К сожалению, это определение машинно зависимо. Дествительно, при переходе к другой модели вычислений время может возводиться в квадрат. А возведение в квадрат может сильно увеличить среднее значение. Левин добивается машинной независимости, ослабляя определение следующим образом. В пункте (2) потребуем, чтобы для некоторого положительного δ среднее значение величины (время работы $R(n, x))^\delta$ было ограничено полиномом от n (переход к другой модели вычислений может потребовать уменьшения δ).

Для любой вычислительной модели, простые в среднем (для этой модели) функции являются простыми в среднем для машинно независимого определения (полагаем $\delta = 1$). Обратное может быть неверным.

Простые в среднем (для машинно независимого определения) рандомизованные задачи просты и для типичного входа. Действительно, пусть алгоритм $R(n, x)$ удовлетворяет пунктам (1) и (2) (для машинно независимого определения) и пусть полином $p(n)$ ограничивается среднее время работы R (в степени δ). Рассмотрим следующий алгоритм: запускаем $R(n, x)$ и даем ему проработать не более, чем $(kp(n))^{1/\delta}$ шагов. Если ответ за это количество шагов не получен, то выдаем произвольный ответ, а иначе выдаем ответ, полученный от R. Вероятность первого события не больше $1/k$, поскольку иначе, среднее время работы было бы больше $1/k$.

Оступление: простые в среднем рандомизованные задачи по Левину и другие варианты определений

Кажется, в литературе встречалось определение, приведенное выше с одним чисто техническим (и неудобным) ограничением: носитель μ_n должен состоять только из слов длины n.

В исходной постановке Левина рандомизованной задачей называется пара (A, μ), где μ — это это одно распределение на всех возможных входах, а не семейство распределений. Рандомизованная (в смысле Левина) задача (A, μ) называется простой в среднем, если существует (возможно неполиномиальный) вероятностный алгоритм R с одним входом для которого выполнено следующее. Алгоритм R для любого исходного данного и для любых результатов бросаний правильно решает задачу A

(но возможно за большое время). Для некоторого положительного ε его время работы $t_R(x)$ на входе x удовлетворяет следующему требованию: среднее (по распределению μ) значение величины $(t_R(x))^\varepsilon/|x|$ конечно. Класс простых в среднем рандомизованных задач (A, μ) обозначается через AveP. Это определение также машинно независимо.

Соотнесем определение Левина с нашим машинно независимым определением простоты в среднем. Предположим, что для всех n носитель μ содержит хотя бы одно слово длины n и более того, суммарная вероятность всех слов длины n не меньше $1/\mathrm{poly}(n)$. Задаче (A, μ) естественным образом можно сопоставить рандомизованную задачу в нашем смысле. А именно, определим последовательность распределений μ_n, где $\mu_n(x)$ есть условная вероятность x при условии, что длина x равна n.

Пусть задача (A, μ) принадлежит классу AveP. Тогда так определенная задача (A, μ_n) будет простой в среднем (в нашем смысле). Действительно, семейство распределений μ_n мажорируется семейством μ (одно распределение для всех n). Поэтому среднее значение $(t_R(x))^\delta/|x|$ по распределению μ_n ограничено полиномом от n.

Трудные для типичного входа задачи

Наибольшее генерируемое распределение

Оказывается для каждой задачи A существует самая трудная ее рандомизация (A, μ_n): если эта рандомизация проста, то и все остальные рандомизации просты. Более того, распределение μ_n можно взять одним для всех задач A. Чтобы доказать это, введем следующее понятие.

Скажем, что распределение μ_n мажорирует распределение ν_n, если для некоторых полиномов $p(n), q(n)$ и для всех n, x выполнено $\nu_n(x) \leq p(n)\mu_{q(n)}(x)$. Если распределение μ_n мажорирует распределение ν_n и рандомизованная задача (A, μ_n) проста, то и рандомизованная задача (A, ν_n) тоже проста. (Довольное очевидное доказательство опускаем.) Иными словами, чем больше распределение, тем сложнее становится задача.

(Заметим в скобках, что любое распределение ν_n мажорируется некоторым монотонным распределением μ_n. Мы называем распределение монотонным, если для некоторого полинома $p(n)$ и для всех $i < n$ и x выполнено $\mu_i(x) \leq p(n)\mu_n(x)$. В качестве μ_n можно взять среднее арифметическое $\nu_0, \dots, \nu_n$. При этом, если ν_n генерируемо, то и μ_n таково. Поэтому без ограничения общности в определении простоты задачи можно ограничиться монотонными распределениями. Это соотвествует нашей интуиции — с ростом параметра безопасности распределение должно усложняться.)

Существует генерируемое распределение μ_n, которое мажорирует все другие генерируемые распределения. Неформально, это взвешенная сумма всех полиномиально генерируемых распределений.

Точнее, чтобы сгенерировать случайную строку по этому распределению μ_n мы выбираем случайным образом генератор с длиной программы не более n и выбираем случайным образом $k < n$. Запускаем этот генератор на входе k, даем ему проработать n шагов, и если он закончит работу, выдаем его результат на выход.

Мы не уточнили, какое распределение вероятностей на генераторах мы используем. Нам достаточно, чтобы это распределение было одинаковым для всех n, больших длины программы генератора, присваивало бы положительную вероятность каждому генератору и чтобы описанный выше алгоритм можно было выполнить за полиномиальное от n время. (Нетрудно понять, что такое распределение существует.)

Генерируемое распределение, мажорирущее все другие генерируемые распределения, будем называть наибольшим.

Сводимость рандомизированных задач

Мы хотим определить, что значит, что рандомизованная задача (A, μ_n) сводится к рандомизованной задаче (B, ν_n). Хотелось бы, чтобы отношение сводимости уважало простоту (если задача сводится к простой задаче B, то она тоже проста) и было рефлексивным и транзитивным.

Годится ли обычная полиномиальная сводимость (сводимость Карпа): A сводится к задаче B, если для некоторой полиномиально вычислимой функции f для всех x выполнено $A(x) = B(f(x))$?

Если не накладывать дополнительных ограничений, то это определение не годится. Причина в том, что f может отображать входы с большой вероятностью (относительно μ) во входы с малой вероятностью (относительно ν). Может оказаться, что вход x, имеющий вероятность, близкую к 1, отображается во вход $f(x)$, имеющий вероятность близкую к нулю. Вероятностный алгоритм для решения задачи B может выдавать на входе $f(x)$ случайный ответ (0 или 1 с равными вероятностями). Ошибка с вероятностью $1/2$ на этом входе мало повлияет на общую вероятность его успеха, поскольку $f(x)$ имеет очень малую вероятность. А нам ответ этого алгоритма не дает никакой информации (мы и сами могли подбросить монетку).

Поэтому надо запретить функции f отображать входы с большой вероятностью во входы с малой вероятностью. А именно, потребуем, чтобы распределение ν мажорировало распределение, которое генерируется функцией f при распределении ν на ее входах. Это распределение обозначается $f(\mu_n)$ и определяется как

$$f(\mu_n)(y) = \mu_n\{x \mid f(x) = y\}.$$

Разрешим функции f также зависеть от n. Итак, определение аналога сводимости Карпа таково. Рандомизованная задача (A, μ_n) сводится к рандомизованной задаче (B, ν_n), если существует алгоритм T, который по данным $1^n, x$ за полиномиальное время выдает некоторое y, такое, что, во-первых, $A(x) = B(T(1^n, x))$ и, во-вторых, распределение $T(1^n, \mu_n)$ мажорируется распределением ν_n. Нетрудно убедиться, что это понятие сводимости удовлетворяет нужным свойствам. (Доказательство опускается.)

Эту сводимость можно обобщить, разрешив алгоритму R быть вероятностным и разрешив равенству $A(x) = B(T(1^n, x))$ не выполняться с небольшой вероятностью. Но проще сразу рассмотреть значительно более слабую сводимость — аналог полиномиальной сводимости Тьюринга (сводимости Кука).

Сводимость Кука естественным образом определяется на задачах более общего типа. А именно, пусть задано бинарное отношение E на множестве двоичных слов, такое, что для каждого x существует y, для которого $E(x, y)$. Задача состоит в том, что для данного x надо найти такое y.

Будем говорить, что рандомизованная задача (A, μ_n) сводится к рандомизованной задаче (B, ν_n), если существует вероятностная полиномиальная машина Тьюринга M с оракулом обладающая следующим свойством. Назовем вероятностную процедуру S (точнее семейство независимых в совокупности случайных величин $S(1^n, 1^k, x)$) решателем задачи (B, ν_n), если для любых n, k вероятность по мере ν_n того, что $S(1^n, 1^k, x)$ правильно решает задачу B не меньше $1 - 1/k$. Требуется, чтобы для любого решателя S задачи (B, ν_n), вероятность по мере μ_n того, что $M^S(1^n, 1^k, x)$ правильно решает задачу A не меньше $1 - 1/k$.

Очевидно, что эта сводимость сохраняет простые для типичного входа задачи (если задача (B, ν_n) проста для типичного входа, то в качестве

S можно взять полиномиальный ее решатель, после чего M^S превратится в полиномиальный решатель задачи (A, μ_n)).

Эта сводимость сохраняет простоту в третьем смысле (для каждого уровня доверия вида $1-1/\mathrm{poly}(n)$ существует полиномиальный алгоритм решения) при следующем дополнительном условии: сводящей машине запрещено делать вызовы $S(1^m, 1^l, y)$ для которых $l \gg m$. Точнее, нужно потребовать, чтобы для всех вызовов было выполнено $l \leq q(m)$ где $q(m)$ — некоторый полином. Если это условие выполнено, то для решения задачи A с вероятностью успеха $1 - 1/p(n)$, надо запустить машину $M^S(1^n, 1^{p(n)}, x)$ и снабдив ее полиномиальным решателем S с уровнем доверия $1 - 1/q(m)$ (по условию такой существует). Мы получим полиномиальный решатель задачи (A, μ_n).

Простота во втором смысле (для каждого уровня доверия вида $1 - \varepsilon$ существует полиномиальный алгоритм решения) сохраняется при следующем условии: при работе $M^S(1^n, 1^k, x)$ для всех сделанных вызовов $S(1^m, 1^l, y)$ число l ограничено некоторой функцией $f(k)$ от k. Если это условие выполнено, то для решения задачи A с вероятностью успеха $1 - 1/k$, надо запустить машину $M^S(1^n, 1^k, x)$ и снабдив ее полиномиальным решателем S с уровнем доверия $1 - 1/f(k)$ (по условию такой существует). Мы получим полиномиальный решатель задачи (A, μ_n).

DistNP-полные задачи

Мы говорим, что рандомизованная задача (A, μ_n) принадлежит классу DistNP, если задача A принадлежит классу NP. Этот класс содержит полные задачи.

Теорема. Существует задача в классе DistNP, к которой сводится любая другая задача из этого класса.

Доказательство. Если спарить любую NP-полную задачу A с наибольшим генерируемым распределением μ_n, то получится рандомизованная задача, которая полна. Действительно, пусть дана произвольная задача (B, ν_n) из DistNP. Пусть f — функция, сводящая B к A. Рассмотрим распределение вероятностей $\rho_n = f(\nu_n)$. Это распределение генерируемо, а значит мажорируется распределением μ_n. (Конец доказательства.)

DistNP-полные задачи с вычислительно простым распределением на входах

Недостаток доказательства этой теоремы состоит в том, что наибольшее генерируемое распределение на входах сложно устроено с вычислительной точки зрения. Например, мы не можем по n и x даже приблизительно вычислить $\mu_n(x)$.

Оказывается, этот недостаток можно устранить, и построить полные задачи с очень просто устроенным распределением (почти равномерным). Главное содержание теории Левина в том, как следует определять вычислительную простоту распределений и как доказывать полноту для вычислительно простых распределений.

Назовем распределение μ_n вычислительно простым, если существуют полиномиально вычислимые функции $(n, x) \mapsto \mathit{code}(n, x)$ и $(n, y) \mapsto \mathit{decode}(n, y)$ такие, что $\mathit{decode}(n, \mathit{code}(n, x)) = x$ и

$$|\mathit{code}(n, x)| \leq -\log \mu_n(x) + O(\log n) \leq \mathrm{poly}(n)$$

для всех x из носителя μ_n. Если x не принадлежит носителю μ_n, то алгоритм вычисления функции $\mathit{code}(n, x)$ должен сообщать об этом. Кроме того, если y не принадлежит образу носителя μ_n относительно функции $\mathit{code}(n, x)$, то алгоритм вычисления функции $\mathit{decode}(n, x)$ должен сообщать об этом.

Определение по существу означает, что существует код Шеннона — Фано для распределения μ_n с полиномиально вычислимыми кодирующей и декодируещей функциями. (Кодом Шеннона — Фано называется кодирование с длиной кода $-\log \mu_n(x) + O(1)$. Такой код существует для любого распределения вероятностей. Мы требуем существование эффективного кода Шеннона — Фано с немного ослабленными требованиями на длину кода.)

Каждое вычислительно простое распределение μ_n генерируемо. Точнее, существует генерируемое распределение ν_n, мажорирующее μ_n. В самом деле, зафиксируем полином $p(n)$, ограничивающий длину слова $\mathit{code}(n, x)$. Алгоритм генерации строки с распределением ν_n выбирает случайное слово z длины $p(n)$, а затем выбирает случайно его префикс y. Если y не является кодом слова из носителя μ_n, то алгоритм выдает что угодно. А иначе алгоритм выдает слово с этим кодом. Если $\mu_n(x) > 0$, то вероятность того, что y окажется равным $\mathit{code}(n, x)$ (а значит алгоритм выдаст x) равна $2^{-|\mathit{code}(n,x)|}/p(n) \geq \mu_n(x)/\mathrm{poly}(n)$.

Обратное скорей всего не так: распределения, порождаемые генераторами псевдослучайных чисел, не мажорируются вычислительно простыми распределениями.

Отметим, что данное нами определение вычислительно простых распределений отличается от определения Левина. Во-первых, как уже сказано, по Левину распределение в рандомизованной задаче одно. Но главное отличие в том, что по Левину требуется, чтобы функция $y \mapsto \sum_{x<y} \mu(x)$ была вычислима за полиномиальное время (слова одной длины сравниваются лексикографически, а слова разных длин сравниваются по длине). Для простоты мы считаем, что $\mu(x)$ принимает только двоично-рациональные значения и алгоритм должен вычислить указанную функцию точно. Требование Левина обеспечивает возможность построения кода Шеннона — Фано с полиномиально вычислимыми кодирующей и декодирующей функциями. Поэтому, для вычислительно простых по Левину распределений, семейство условных распределение $\mu_n(x)$ (вероятность x при условии, что длина x равна n) является вычислительно простым в нашем смысле (если дополнительно известно, что суммарная вероятность всех слов длины n не меньше $1/poly(n)$).

Определение Левина обладает следующим недостатком: пусть $f(x)$ вычислимая и обратимая за полиномиальное время перестановка множества всех слов. Может оказаться, что распределение μ вычислительно просто по Левину, а распределение $\mu(f(x))$ — нет. Вспомним, что в исходными данными для задач обычно являются двоичные коды конечных объектов, отличных от слов, например, пропозициональных формул. Перекодирование из одной системы в другую выполняется вычислимыми и обратимыми за полиномиальное время перестановками. Представляется неправильным, что в одной кодировке распределение на формулах может быть вычислительно простым, а в другой кодировке — нет. Заметим, что наше определение лишено этого недостатка.

Будем обозначать через Dist'NP класс рандомизованных задач (A, ν_n), у которых A принадлежит NP, а семейство ν_n вычислительно просто.

Теорема. Существуют Dist'NP-полные задачи.

Доказательство. Задача U состоит в том, что по тройке $(1^n, M, x)$ надо узнать, останавливается ли программа M недетерминированной машины Тьюринга на входе x за n шагов. При разумном выборе способа записи программ это множество принадлежит классу NP.

Теперь определим распределение вероятностей μ_n на тройках. Первая компонента равна n, длины второй и третьей компонент выбираются независимо среди слов длины менее n, а сами они выбираются равномерно среди слов выбранной длины. То есть вероятность тройки $(1^n, M, x)$ равна $n^{-2}2^{-|M|-|x|}$ при условии, что длины M и x меньше n, и нулю иначе. (Отметим, что это распределение монотонно.)

Назовем квазиравномерным распределением λ_n следующее распределение на двоичных словах длины n: выбираем случайно натуральное число в интервале $0, 1, \ldots, n$, затем выбираем случайно слово x длины $n-k-1$ и выдаем слово $0^k 1x$.

Сначала докажем, что к определенной выше рандомизованной задаче сводится любая задача с квазиравномерным распределением на входах. Пусть (A, λ_n) рандомизованная задача с квазиравномерным распределением на входах. Зафиксируем недетерминированную машину Тьюринга M, допускающую A и обозначим через $p(n)$ полином, ограничивающий время ее работы на входах длины n. Без ограничения общности мы можем считать, что $p(n)$ больше $|M|$ и n для всех n. Сводящая функция отображает вход $0^i 1x$ в тройку $(1^{p(n)}, M, x)$. По определению вероятность этой тройки относительно $\mu_{p(n)}$ равна $p(n)^{-2}2^{-|M|-|x|}$. Эта вероятность отличается от вероятности входа $0^i 1x$ по распределению λ_n полиномиально ограниченным множителем.

Теперь докажем, что любая задача (B, ν_n) у которой распределение на входах простое, сводится к некоторой задаче (A, λ_n) с квазиравномерным распределением на входах. Пусть *code*, *decode* полиномиально вычислимые кодирующая и декодирующая функции из определения простоты распределения ν_n. Положим $A(y) = B(x)$, если $y = 0^i 1 code(n, x)$ и длина y равна полиному $p(n)$ ограничивающему длину $code(n, x)$. Если полином $p(n)$ возрастает, то такие n и x единственны, а значит определение корректно. (Если y не имеет такого вида, то положим $A(y) = 0$.)

Отображение $(n, x) \mapsto 0^i 1 code(n, x)$ (где и выбирается так, чтобы длина образа была равна $p(n)$) сводит B к A. По условию образ меры ν_n относительно этого отображения с точностью до полиномиального множителя совпадает с квазиравномерной мерой на словах длины $p(n)$. Осталось доказать, что множество A принадлежит NP. Это следует из того класс NP замкнут вниз относительно сводимости Кука и что A сводится к B с помощью полиномиально вычислимого отображения $0^i 1z \mapsto decode(n, z)$, где $n = |0^i 1z|$. (Конец доказательства.)

Более того, для большинства NP-полных задач B существует вычислительно простое распределение ν_n, для которого рандомизованная задача (B, ν_n) Dist'NP-полна.

Действительно, как правило, доказательство NP-полноты данной задачи B устроено так. Для любой NP-задачи A мы строим сводящую A к B *инъективную* функцию f, которая не только вычислима, но и *обратима* за полиномиальное время. Возьмем в качестве A любую задачу, имеющую вычислительно простую DistNP-полную рандомизацию (A, μ_n). Построим сводящую A к B функцию с указанными свойствами. Поскольку f сводит (A, μ_n) к $(B, f(\mu_n))$, последняя задача тоже DistNP-полна. При этом распределение $f(\mu_n)$ вычислительно просто: кодирующая и декодирующая функции для $f(\mu_n)$ получаются из кодирующей и декодирующей функции для μ_n применением f или ее обратной.

Тем не менее представляют интерес поиск естественных распределений на входах NP-полных задач, для которых рандомизованная задача DistNP-полна. Такое распределение существует для задачи о замощении.

Dist'NP-полнота задачи о замощении.

Пусть зафиксировано некоторое конечное множество цветов $\{0, 1, 2, \ldots, c-1\}$. Исходным данным в задаче о замощении является набор квадратиков с раскрашенными в эти цвета ребрами и последовательность цветов z. Надо выяснить, можно ли замостить квадрат размера $|z| \times |z|$ маленькими квадратиками так, чтобы цвета на смежных ребрах квадратиков совпадали и чтобы последовательность цветов на нижнем крае большого квадрата была равна z. Любой из исходных квадратиков можно дублировать в любом количестве, но поворачивать и переворачивать квадратики нельзя.

Распределение ρ_n на входах следующее: мы случайно выбираем набор квадратиков (все 2^{c^4} наборов считаются равновероятными). Затем мы выбираем пару натуральных чисел a, b равномерно среди всех пар таких, что $a + b < n$. Затем выбираем раскраску нижнего края по следующему правилу: первый цвет в z равен 2, следующие a цветов равны 1, следующие b цветов z выбираются случайно и независимо среди цветов 0,1, а остальные цвета равны 0.

Нетрудно проверить, что указанное распределение вычислительно просто и что задача о замощении принадлежит NP.

Теорема. При достаточно большом c задача о замощении Dist'NP-полна.

Доказательство. Мы доказали, что существует DistNP-полная задача с простым распределением на входах и что любая NP-задача с простым распределением сводится к некоторой NP-задаче с квазиравномерным распределением. Поэтому достаточно свести к задаче о замощении любую задачу (A, λ_n) из DistNP с квазиравномерным распределением на входах. Зафиксируем такую задачу.

Закодируем каждое исходное данное $0^i 1x$ длины n задачи A словом

$$u = u(i, x) = 1^n 0ix,$$

где i записывается $\log n$ битами.

Нам известно, что множество слов $u(i, x)$, для которых $A(0^i 1x) = 1$ принадлежит NP. Значит для некоторого полинома $p(n)$ некоторая недетерминированная машина Тьюринга V допускает слово u за $p(n)$ шагов тогда и только тогда, когда $A(0^i 1x) = 1$. Мы можем считать, что допускание определяется так: V допускает слово u за $p(n)$ шагов, если существует вычисление V на входе u, длящееся по крайней мере $p(n)$ шагов.

Ниже мы докажем, что для некоторого фиксированного набора цветов существует набор квадратиков α, со следующим свойством. Если $z = 2u$, то квадрат размера $|z| \times |z|$ можно замостить так, чтобы последовательность цветов на нижнем краю была равна z, тогда и только тогда, когда машина V может проработать $|z|$ шагов при начальном содержимом $0u$ входной ленты.

Сведение задачи (A, λ_n) к задаче о замощении устроено так. Слово $0^i 1x$ отображается в тройку $(\alpha, 2u00\ldots0)$, где слово $u = u(i, x)$ определено выше. Напомним, что вероятность исходного слова относительно квазиравномерного распределения λ_n равна $n^{-1}2^{-|x|}$. Сравним эту величину с вероятностью тройки-образа (относительно распределения на входах задачи о замощении). Последняя с точностью до ограниченного множителя равна

$$2^{-c^4} 2^{-\log n - |x|}/(p(n))^2 = 2^{-c^4} 2^{-|x|}/n(p(n))^2.$$

Видно, что они отличаются полиномиальным множителем.

Осталось построить набор квадратиков α с нужным свойством. Возьмем множество цветов равным алфавиту Γ конфигураций машины V (ее внутренний алфавит, объединенный с множеством пар (буква, состояние машины)) плюс множество пар букв из Γ. Квадратики на нижней и верхней сторонах будут раскрашены в цвета из Γ, а на боковых гранях — в цвета из $\Gamma \times \Gamma$. При этом верхний цвет нижнего ребра всегда будет второй компонентой цвета левого вертикального ребра и первой компонентой цвета правого вертикального ребра. Цвет верхнего ребра будет равен значению функции перехода машины на тройке, состоящей из первой компоненты цвета левого ребра, цвета нижнего ребра и второй компоненты цвета правого ребра. Пара (0, начальное состояние) отождествляется с цветом 2. Любое замощение квадрата $|z| \times |z|$, в котором цвета нижней стороны равны $z = 2u$ моделирует $|z|$ шагов работы V на входе $0u$, и обратно. (Конец доказательства.)

Замечание. Можно разными естественными способами определять, распределение вероятностей для задачи о замощении так, чтобы полученная задача стала полной. Но, к сожалению, не удается доказать полноту для равномерного распределения на последовательностях цветов нижнего края.

Сведение произвольной задачи из DistNP к некоторой задаче из Dist'NP

В этом разделе мы докажем следующее утверждение.

Теорема. Каждая задача (A, μ_n) из DistNP сводится к некоторой задаче (B, ν_n) из Dist'NP.

Из этой теоремы следует, что задача о замощении DistNP-полна.

В доказательство этой теоремы нам понадобятся так называемые задачи поиска. Задача поиска задается полиномиально разрешимым бинарным отношением $R(x, y)$ на множетстве двоичных слов таким, что для некоторого полинома $p(n)$ для всех x выполнено $R(x, y) \Rightarrow |y| = p(|x|)$. Алгоритм решает эту задачу, если для каждого x он находит такое y, для которого $R(x, y)$ (если он есть), а если такого y нет, то алгоритм должен сообщить об этом.

Сначала мы сведем произвольную задачу поиска с вычислительно простым распределением на входах к некоторой задаче из DistNP, у которой распределение на входах также вычислительно простое.

Теорема. Каждая рандомизованная задача поиска с вычислительно простым распределением на входах сводится (по Куку) к некоторой задаче из Dist'NP.

Доказательство. В доказательстве мы используем универсальное семейство хэш-функций. Что это такое? Семейство функций H типа $\{0,1\}^l \to \{0,1\}^k$ называется универсальным семейством хэш-функций, если для любого $y \in \{0,1\}^l$ случайная величина $h(y)$ (где h выбирается случайно из H) равномерно распределена в $\{0,1\}^k$ и для любых различных y_1, y_2 случайные величины $h(y)_1, h(y_2)$ независимы (но тройки могут быть уже зависимы).

Приведем пример такого семейства. Рассмотрим булеву матрица A размера $k \times l$ и булев вектор u длины k. Пара A, u задает линейное отображение $h(y) = Ay + u$ (все операции выполняются в поле вычетов по модулю 2). При любых k, l мы получаем полиномиально вычислимое универсальное семейство хэш-фукнций, которое мы будем обозначать через H_{lk}.

Лемма. Пусть $y_1 \neq y_2$. Тогда случайные величины $Ay_1 + u$ и $Ay_2 + u$ независимы и равномерно распределены среди строк длины k (если A, u выбираются случайно по равномерному распределению).

Доказательство. Равномерная распределенность $h(y)$ очевидна. Докажем независимость. Отображение $(A, u) \mapsto (Ay_1 + u, Ay_2 + u)$ является линейным отображением из пространства матриц размера $k \times (l+1)$ в пространств матриц размера $k \times 2$. Поэтому все элементы в образе этого отображения имеют одинаковую кратность (равную мощности его ядра). Значит, нам достаточно доказать его сюръективность. Нетрудно видеть, что это отображение состоит в умножении справа на некоторую матрицу M. Нам нужно доказать, что ранг этой матрицы равен 2. Это следует из того, что столбцы этой матрицы различны и имеют 1 в качестве последней, $l+1$-ой, координаты. Лемма доказана.

Мы будем использовать еще и то, что функции из определенного нами семейства полиномиально вычислимы равномерно по k, l. Это означает, что по описанию функции h (матрице и вектору) и ее аргументу y мы можем за полиномиальное время найти $h(y)$.

Пусть данная нам задача поиска задается полиномиально разрешимым отношением $R(x, y)$ таким, что $R(x, y) \Rightarrow |y| = p(|x|)$ (где $p(n)$ — некоторый полином). Пусть также дано вычислительно простое семейство распределений вероятностей μ_m. Сначала предположим, что носитель распределения μ_m содержит слова некоторой одной полиномиальной длины $n = q(m)$. (Избавиться от этого предположения будет совсем нетрудно.)

Рассмотрим такую задачу разрешения: даны строка x (длины n), число $i \leq p(n)$, функция $h : \{0,1\}^{p(n)} \to \{0,1\}^k$ из универсального семейства хэш-функций $H_{p(n)k}$. Надо выяснить, найдется ли такое y длины $p(n)$, для которого верно $R(x, y)$, у которого i-ый бит равен 1 и при этом $h(y) = \mathbf{0}$. Ясно, что эта задача принадлежит NP. Распределение ν_m

на входах такое: x выбирается случайно по распределению μ_m, данному нам. Затем мы выбираем случайно число $k \le p(n)$, выбираем случайно хэш-функцию h и вектор $\mathbf{0}$. Число $i \le p(n)$ также выбирается случайно.

Очевидно, что описанное распределение вычислительно просто. Установим, что исходная задача поиска сводится к построенной задаче разрешения. Пусть нам дана процедура S решения задачи разрешения. Мы построим полиномиальную процедуру решения задачи поиска, вероятность ошибки которой (по построенному распределению) в полином от n раз больше, чем вероятность ошибки процедуры S (по распределению μ_n).

Пусть для данного x длины n надо найти y для которого $R(x, y)$. Мы выбираем случайное $k \le p(n)$ и хэш-функцию h из семейства $H_{p(n)k}$. Затем для каждого $i \le p(n)$ применяем процедуру S к тройке (x, i, h). Пусть $b_i = 0, 1$ — ответ, выданный S. Формируем вектор $y = b_1 \dots b_{p(|n|)}$ и проверяем, верно ли $R(x, y)$. Если верно, то останавливаемся. Иначе повторяем то же самое с новым h. Если после $mp(n)$ повторений нужное y не найдено, то говорим, что его нет.

Оценим вероятность ошибки этого алгоритма. Ошибка для данного x может быть только в том случае, если существует такое y, для которого $R(x, y)$, но мы его не нашли. Пусть дано любое такое x. Возьмем число k, для которого количество y, для которых истинно $R(x, y)$, находится в пределах от 2^{k-1} до 2^k. Докажем, что с вероятностью более $3/8$ для случайно выбранной хэш-функции $h \in H_{p(n)k}$ существует ровно одно y, для которого $R(x, y)$ и $h(y) = \mathbf{0}$ (в этом случае вектор $b_1 \dots b_{p(|n|)}$ совпадает с этим единственным y, если процедура S не сделала ни одной ошибки).

Лемма. С вероятностью более $3/8$ существует ровно одно y, для которого $R(x, y)$ и $h(y) = \mathbf{0}$.

Доказательство. Рассмотрим множество Y, состоящее из тех y, для которых $R(x, y)$. По формуле включений-исключений вероятность рассматриваемого события не меньше разности

$$\sum_{y} P[h(y) = \mathbf{0}] - \sum_{y_1 \neq y_2} P[h(y_1) = h(y_2) = \mathbf{0}].$$

Здесь y, y_1, y_2 пробегают множество Y, а вероятность берется по случайному выбору $h \in H_{p(n)k}$. Поскольку $P[h(u) = \mathbf{0}] = 2^{-k}$ для любого y, первая сумма не меньше $|Y|2^{-k}$. Поскольку $h(y_1), h(y_2)$ независимы, вто-

рая сумма равна
$$\binom{|Y|}{2} \cdot 2^{-2k} < \frac{|Y|^2}{2 \cdot 2^{2k}}.$$
Поэтому оцениваемая вероятность больше
$$\frac{|Y|}{2^k} - \frac{1}{2}\left(\frac{|Y|}{2^k}\right)^2.$$

По выбору k число $|Y|2^{-k}$ находится в пределах от $1/2$ до 1. Минимальное значение многочлена $t - t^2/2$ на этом интервале равно $3/8$. Поэтому рассматриваемая вероятность больше $3/8$. Лемма доказана.

Назовем h удачной для данного x, если k (размерность области значений h) определено, как выше, и существует ровно одно y для которого $h(y) = \mathbf{0}$ и $R(x, y)$. По лемме для каждого x с вероятностью не менее $3/8p(n)$ случайно выбранная функция h удачна.

Назовем x удобным, если вероятность того, что $S(m, x, i, h)$ дает неправильный ответ не больше $1/4p(n)^2$. Вероятность берется по выше опреденному распределению на парах (i, h).

Вероятность ошибки можно оценить сверху суммой двух величин: общая μ_m-вероятность всех неудобных слов и вероятность того, что вход x является удобным, но все $mp(n)$ выбранных четверок оказались неудачными.

Пусть δ обозначает ν_m-вероятность ошибки процедуры S. Тогда первое слагаемое (общая μ_n-вероятность всех неудобных слов) не превосходит $\delta \cdot 4p(n)^2$. Действительно, каждое неудобное слово вносит вклад $1/4p(n)^2$ в общую вероятноть ошибки, и суммарная вероятность всех неудобных слов не может превышать общую вероятность ошибки, деленную на этот минимальный вклад.

Для любого удобного слова x вероятность того, что хотя бы для одного i процедура $S(n, x, i, h, v)$ ошибается не больше $p(n)/4p(n)^2 = 1/4p(n)$. Поэтому для любого удобного слова вероятность найти y в каждой из $mp(n)$ попыток не меньше $3/8p(n) - 1/4p(n) = 1/8p(n)$. Значит второе слагаемое не больше $(1 - 1/8p(n))^{mp(n)} \approx e^{-m/8}$.

Для завершения доказательства теоремы осталось только понять, почему мы можем предпологать, что распределение μ_m сосредоточено на словах одной длины. Мы знаем, что любое слово из носителя μ_m ограничено по длине некоторым полиномом $q(m)$. Закодируем каждое x из носителя μ_m словом длины 0^j1x длины $q(m)+1$ и рассмотрим новую NP-задачу: по слову 0^j1x требуется найти такое y, для которого $R(x,y)$, и распределение $\mu'_m(0^j1x) = \mu_m(x)$ на ее входах. Исходная задача сводится к новой, причем в носителе нового распределения μ'_m все слова имеют одну длину $q(m) + 1$. Теорема доказана.

Нам осталось доказать, что каждая задача из DistNP сводится к некоторой рандомизованной задаче поиска, распределение на входах которой вычислительно просто. На самом деле мы докажем даже больше — каждая рандомизованная задача поиска, распределение на входах которой генерируемо, сводится к некоторой рандомизованной задаче поиска, распределение на входах которой вычислительно просто.

Теорема. Каждую рандомизованную задачу поиска можно свести к другой рандомизованной задаче поиска, у которой распределение на входах вычислительно просто.

Доказательство. Пусть данная нам задача состоит в поиске по данному x (длины n) такого y, для которого выполнено $R(x,y)$. И пусть распределение на входах μ_m генерируется полиномиальным алгоритмом K. Этот алгоритм получает на вход строку из m единиц и за полиномиальное от m время выдает на выход строку, имеющую распределение μ_m. Через $K_m(r)$ мы будем обозначать выход K при условии, что r есть случайная строка полиномиальной длины $l = q(m)$, использованная алгоритмом K. То есть, $\mu_i(x)$ равно доле тех r, для которых $K_m(r) = x$.

Без ограничения общности мы предположим, что носитель распределения μ_m содержит слова некоторой одной полиномиальной длины $n = p(m)$.

Сначала изложим идею доказательства. Допустим исходная задача трудна. Рассмотрим два “крайних” случая.

Случай 1: задача “по случайному взятому $x \in \{0,1\}^n$ найти такое r, для которого $K_m(r) = x$” трудна. В этом случае мы имеем трудную зазачу с равномерным распределения на входах.

Случай 2: функция $r \mapsto K_m(r)$ однозначна и легко обратима. Тогда задача “по r найти такое y для которого выполнено $R(K_m(r), y)$” трудна для равномерного распределения на входах. Действительно, если бы была возможность по r находить такое y, то мы бы могли и по x находить y, обращая функцию $r \mapsto K_m(r)$. Более того, однозначность отображения $r \mapsto K_m(r)$ нам на самом деле не нужна. В общем случае, достаточно по x уметь выбирать равномерно r из множества тех r, для которых $K_m(r) = x$. В этом случае мы также имеем трудную задачу поиска с равномерным распределением на входах.

Нам надо решить следующие две проблемы. Первая состоит в том, что отображение $r \mapsto K_m(r)$ может не подпадать ни под первый, ни под второй случай. Вторая сложность в том, что нам нужно не просто доказать существование трудной задачи поиска с простым распределением, исходя из предположения, что данная нам задача трудна. Нам нужно указать конкретную задачу, к которой данная задача сводится.

Первая трудность преодолевается следующим образом. Если отображение $r \mapsto K_m(r)$ не подпадает под второй случай, то оказывается, что функция $(s, g, h) \mapsto (g(K_m(h(s)), g, h)$ подпадает под первый случай. Здесь g, h случайно выбираемые функции из подходящего универсального семейства хэш-функций. То есть задача "по x выбрать равномерно элемент из множества тех r, для которых $K_m(r) = x$" сводится к задаче "по случайно выбранным v, g, h найти s для которого $g(K_m(h(s)) = v$". (Сводить первую задачк ко второй нужно так: для данного x случайно выбираем g, h, и получив решение s второй задачи для исходной тройки $g(x), g, h$, выдаем $h(s)$".)

Вторая трудность обходится с помощью соединения задач обращения функции $(s, g, h) \mapsto (g(K_m(h(s)), g, h)$ и задачи "по r найти такое y для которого выполнено $R(K_m(r), y)$" в одну задачу.

Итак, перейдем к формальному доказательству. Рассмотрим следующую задачу поиска. Даны натуральные числа k, m, хэш-функции h, g из универсальных семейств $H_{k-1,q(m)}, H_{p(m),k}$, соответственно, и бинарное слово v длины k. Надо найти такие слова s, x длин k, $n = p(m)$ и слово y, для которых $K_m(h(s)) = x$, $g(x) = v$ и $R(x, y)$. (Слово x можно и не искать, поскольку его можно вычислить по m, h, s.) Напомним, что $q(m)$ обозначает полином, ограничивающий количество случайных битов алгоритма K_m, а $p(m)$ обозначает длину выхода алгоритма K_m.

Теперь зададим распределение вероятностей ν_m на входах этой задачи. Число m уже задано, число k выбирается равномерно среди всех чисел от 1 до $l = q(m)$. Функции g, h и слово v выбираются независимо и равномерно в $H_{k-1,q(m)}, H_{p(m),k}$, и $\{0, 1\}^k$, соответственно (после того, как уже выбрано k). Таким образом g, h, v независимы при любом фиксированном k (но зависимы, если k не фиксировано).

Очевидно, что построенное семейство случайных величин вычислительно просто. Докажем, что исходная задача сводится к построенной. Пусть нам дано исходное слово x, выбранное случайно по распределению μ_m (мы знаем m). Длина x равна $n = p(m)$. Выбираем случайным образом k от 1 до $l = q(m)$ и случайным образом h, g из семейств $H_{k-1,l}, H_{n,k}$. Запускаем решатель второй задачи на пятерке $m, k, h, g, g(x)$. Решатель нам возвратит некоторое y. Если $R(x, y)$ истинно, то выдаем y и останавливаемся, а иначе повторяем то же самое с новыми k, h, g. Повторяем эту процедуру ml раз, и если нам ни разу не повезет, то говорим, что такое y отсутствует.

Пусть вероятность того, что решатель новой задачи ошибается на входе, взятом случайно по распределению ν_m, равна ε. Докажем, что вероятность того, что для случайного по распределению μ_m входа x мы неправильно решим задачу поиска (то есть, не найдем существующее y, для которого $R(x, y)$), ограничена сверху величиной $16l\varepsilon$ плюс экспоненциально малая (от m) величина.

Назовем четверку k, h, g, v удачной, если в образе отображения $K_m(h(s))$ существует и притом единственное x, для которого $g(x) = v$ и кроме того, $2^{-k} \leq \mu_m(x) \leq 2^{-k+1}$ (то есть, количество прообразов x относительно K_m находится в пределах от 2^{l-k} до 2^{l-k+1}).

Лемма 1. Для любого x из носителя μ_m вероятность того, что четверка $k, h, g, g(x)$ удачна, больше $3/16l$.

Доказательство. С вероятностью не менее $1/l$ окажется, что выбранное k удовлетворяет неравенству $2^{-k} \leq \mu_m(x) \leq 2^{-k+1}$. Оценим теперь вероятность того, что при так выбранном k для случайного h слово x содержится в образе отображения $K_m(h(s))$. Обозначим через R полный прообраз x относительно отображения $r \mapsto K_m(r)$. По формуле включений исключений эта вероятность не меньше

$$\sum_s P[h(s) \in R] - \sum_{s_1 \neq s_2} P[h(s_1) \in R,\ h(s_2) \in R].$$

В силу универсальности семейства эта разность равна

$$\sum_s \mu_m(x) - \sum_{s_1 \neq s_2} \mu_m(x)^2 > 2^{k-1}\mu_m(x) - (2^{k-1}\mu_m(x))^2/2.$$

В силу выбора k, это не меньше $3/8$.

Теперь для фиксированных k, h оценим вероятность того, что при случайном выборе g в образе функции $K_m(h(s))$ имеется $x' \neq x$, для которого $g(x') = g(x)$. Мощность этого образа ограничена 2^{k-1}, а множество значений g имеет 2^k элементов. При фиксированном $x' \neq x$ вероятность события $g(x') = g(x)$ равна 2^{-k}. Умножая ее на количество различных $x' \neq x$, получаем $1/2$.

Итак, вероятность события указанного в условии леммы больше $(1/l)(3/8)(1/2) = 3/16l$. Лемма доказана.

Назовем слово x удобным (для решения исходной задачи поиска), если для случайных k, g, h вероятность того, что четверка $(k, g, h, g(x))$ удачна, но решатель второй задачи ошибается на этой четверке, не больше $1/8l$. По Лемме, для любого удобного слова x вероятность того, что наш алгоритм с одной попытки решит исходную задачу поиска на слове x не меньше $3/16l - 1/8l = 1/16l$. Поэтому вероятность того, что алгоритм не решит ее ни в одной из ml попыток не больше $(1 - 1/16l)^{ml} \approx e^{-m/16}$.

Для неудобных слов мы ничего не знаем про вероятность правильного решения, поэтому мы просто оценим общую μ_m-вероятность всех неудобных слов. Для этого рассмотрим следующую вспомогательную меру на четверках (k, g, h, v). Первые три компонента выбираются случайно, как описывалось выше, а v выбирается следующим образом. Мы случайно выбираем x по распределению μ_m и выдаем четверку $(k, g, h, g(x))$, если она окажется удачной. Иначе мы не выдаем ничего, таким образом, определенная мера не является вероятностной — сумма мер всех четверок может быть меньше 1. Мы докажем что определенная таким образом мера ρ_m мажорируется мерой ν_m. Более точно, выполнено такое неравенство, которое доказано в следующей лемме: $\rho_m(k, g, h, v) \leq 2\nu_m(k, g, h, v)$.

Отсюда следует желаемая оценка. Действительно, оценим двумя способами вероятность события "четверка $(k, g, h, g(x))$ удачна, но решатель второй задачи ошибается на ней" (слово x выбирается по распределению μ_m, а первые три компоненты выбираются, как описано выше; кроме того это распределение "умножается" на равномерное распределение на бросаниях решателя). Каждое неудобное слово дает вклад не менее $1/8l$ в эту вероятность. Поэтому она не меньше, чем $p/8l$, где p — общая вероятность всех неудобных слов. С другой стороны, вероятность этого события есть в точности вероятность события "решатель ошибается на четверке (k, g, h, v)" по мере ρ_m, умноженной на равномерную меру на случайных бросаниях решателя. Последняя вероятность по Лемме 2 не более чем вдвое превосходит вероятность ошибки решателя по мере ν_m (умноженной на равномерную меру на случайных бросаниях решателя). Поэтому мы имеем неравенство $p/8l \leq 2\varepsilon$, то есть $p \leq 16l\varepsilon$. Нам осталось доказать лемму.

Лемма 2. Для всех четверок выполено $\rho_m(k, g, h, v) \leq 2\nu_m(k, g, h, v)$.

Доказательство. Поскольку первые три компоненты для обоих мер выбираются идентичным образом, нам достаточно доказать следующее утверждение. Для любой удачной четверки k, g, h, v вероятность события $g(x) = v$ (при случайном выборе x по мере μ_m) не больше 2^{-k+1}. По определению удачности такое x ровно одно, и его вероятность не больше 2^{-k+1}, так что это очевидно. Лемма доказана.

Теорема доказана.