1 Основные понятия сложности в среднем

Распределением мы называем функцию $R : \{0,1\}^* \to [0,1]$, такую что $\sum_x R(x) = 1$. Носителем распределения называется множество $x \in \{0,1\}^*$ для которых $R(x) > 0$. Мы будем обозначать носитель R так: $\operatorname{supp} R$.

Определение 1.1. Ансамблем распределений $\{D_n\}_{n=1}^{\infty}$ называется последовательность распределний $D_i : \{0,1\}^* \to [0,1]$ с конечным носителем. Тут n — это параметр задачи, в криптографии его называют параметром надежности.

Равномерным ансамблем мы будем называть ансамбль U_n для которого $U_n(x) = 2^{-n}$, если $|x| = n$.

Распределенной задачей (распознавания) мы будем называть пару (L, D), где L — это язык, а D — это ансамбль распределений.

Что значит, что распределенная задача распознавания (L, D) решается за полиномиальное в среднем время? Наивное определение такое: если существует такой алгоритм A, что он распознает язык L и математическое ожидание времени его работы на входах, задаваемых ансамблем распределением D_n, ограничено некоторым полиномом от n.

Почему данное выше определение нельзя считать хорошим? Дело в том, что класс определенных выше задач не замкнут относительно такой простой операции как возведение в квадрат (напомним, что класс P как раз тем и хорош, что замкнут относительно сложения и умножения). Действительно, пусть D_n — это равномерное распределение на входах длины n, A — алгоритм, распознающий язык L, а $t_A(x)$ — время работы алгоритма A на входе x. Пусть $t_A(x) = 2^n, x = 0^n$ и $t_A(x) = n, x \neq 0^n$. В таком случае $E[t_A(x)] \leq 2n$ ограничено полиномом, а $E[t_A^2(x)] \geq 2^n$ уже экспоненциально.

Рассмотрим следующее определение понятия

Определение 1.2 (Левин)**.** Алгоритм $A(x, 1^n)$ решает распределенную задачу распознавания (L, D) за полиномиальное в среднем время, если $\exists \epsilon > 0 : \mathop{\mathrm{E}}_{x \leftarrow D_n}[t_A^{\epsilon}(x, 1^n)] = O(n)$, где $t_A(x, 1^n)$ — это время работы алгоритма A на входе $(x, 1^n)$.

Это определение решает проблему с возведением времени работы в квадрат.

Есть и другое эквивалентное, хоть и не очень похожее на первое определение.

Определение 1.3 (Импальяццо)**.** Распределенная задача (L, D) решается за полиномиальное в среднем время, если существует алгоритм $A(x, \delta, 1^n)$ (здесь δ это рациональное число из интервала $(0, 1)$) и полином p такие, что:

1. Для всех $x \in \operatorname{supp} D_n$ время работы $A(x, \delta, 1^n)$ ограничено некоторым $p(\frac{|x|}{\delta})$;
2. $x \in \operatorname{supp} D_n$; $A(x, \delta, 1^n) \in \{L(x), \perp\}$;
3. $\mathop{\Pr}_{x \leftarrow D_n}[A(x, \delta, 1^n) = \perp] < \delta$

Теорема 1.1. Определения Левина и Импальяццо эквивалентны.

Доказательство. Сначала проверим, что из определения Левина следует определение Импальяццо.

Пусть задача распознавания (L, D) решается за полиномиальное в среднем время в определении Левина. Тогда существуют такой алгоритм A и константы ϵ, c, что $\mathrm{E}_{x \leftarrow D_n}[t_A(x, 1^n)^\epsilon] \leq cn$. В таком случае мы можем построить алгоритм $B(x, \delta, 1^n)$, который удовлетворяет определению Импальяццо. Алгоритм B запускает $A(x, 1^n)$ на $(\frac{c|x|}{\delta})^{\frac{1}{\epsilon}}$ шагов. Если A успевает остановиться, то B выдает его ответ, иначе выдает $\perp$.

Свойства 1 и 2 из определения Импальяццо выполнены, осталось проверить выполнение свойства 3:

$$\Pr_{x \leftarrow D_n}[B(x, \delta, 1^n) = \perp] = \Pr_{x \leftarrow D_n}[t_A(x, 1^n) > (\frac{cn}{\delta})^{\frac{1}{\epsilon}}] = \Pr_{x \leftarrow D_n}[t_A^\epsilon(x, 1^n) > \frac{cn}{\delta}] < \delta$$

Последнее неравенство следует из неравенства Маркова.

Теперь докажем, что из определения Импальяццо следует определение Левина.

Пусть алгоритм $A(x, \delta, 1^n)$ решает распределенную задачу распознавания за полиномиальное в среднем время в определении Импальяццо. Построим алгоритм $B(x, 1^n)$ такой, что он решает эту же задачу за полиномиальное в среднем время в определении Левина. Алгоритм B запускает алгоритм A на следующих входах: $A(x, \frac{1}{2}) \dots A(x, \frac{1}{2^k})$ пока не встретится первый ответ, отличный от $\perp$. Этот ответ B и выдает в качестве результата своей работы.

Докажем, что построенный алгоритм удовлетворяет определению Левина. Пусть время работы $A(x, \delta, 1^n)$ ограничено $(\frac{n}{\delta})^c$ и пусть $\epsilon = \frac{1}{2c}$. Тогда:

$$\mathrm{E}_{x \leftarrow D_n}[t_B^\epsilon(x, 1^n)] \leq \sum_{k=1}^{\infty} (n2^k)^{\frac{1}{2}} \frac{1}{2^k} = \sum_{k=1}^{\infty} \frac{n^{\frac{1}{2}}}{2^{\frac{k}{2}}} = O(n^{\frac{1}{2}}) = O(n)$$

□

Определим нескольких сложностных классов, связанных со сложностью в среднем.

Определение 1.4. AvgP — это класс распределенных задач распознавания, которые решаются за полиномиальное в среднем время.

Определение 1.5. HeurP — это класс распределенных задач распознавания (L, D), для которых существует алгоритм $A(x, \delta, 1^n)$ и полином P такие, что:

1. Для всех $x \in \operatorname{supp} D_n$ время работы $A(x, d, 1^n)$ ограничено $p(\frac{|x|}{\delta})$;
2. $\Pr_{x \leftarrow D_n}[A(x, \delta, 1^n) \neq L(x)] < \delta$.

Легко видеть, что AvgP $\subseteq$ HeurP. Действительно, для этого достаточно ответ $\perp$ в алгоритме из определения Импальяццо заменить на 0.

В определениях классов AvgP и HeurP вероятность ответа $\perp$ и ошибки можно понижать, увеличивая время работы. Можно определить классы с непонижаемой ошибкой.

Определение 1.6. $\mathrm{Avg}_{\delta(n)}$P — это класс распределенных задач распознавания (L, D), для которых существует алгоритм $A(x, 1^n)$ такой, что:

1. $A(x, 1^n)$ работает полиномиальное время.
2. $A(x, 1^n) \in \{L(x), \perp\}$

3. $\Pr_{x \leftarrow D_n}[A(x, 1^n) = \perp] < \delta(n)$

Аналогично определяется класс $\mathrm{Heur}_{\delta(n)}\mathrm{P}$

Определение 1.7. $\mathrm{Heur}_{\delta(n)}\mathrm{P}$ — это класс распределенных задач распознавания (L, D), для которых существует алгоритм $A(x, 1^n)$ такой, что:

1. $A(x, 1^n)$ работает полиномиальное время.
2. $\Pr_{x \leftarrow D_n}[A(x, 1^n) \neq L(x)] < \delta(n)$

Легко видеть, что $\mathrm{AvgP} \subsetneq \mathrm{Avg}_{\frac{1}{\mathrm{n^c}}}\mathrm{P} \subseteq \mathrm{Heur}_{\frac{1}{\mathrm{n^c}}}\mathrm{P}$ и $\mathrm{Heur_P} \subsetneq \mathrm{Heur}_{\frac{1}{\mathrm{n^c}}}\mathrm{P}$. Строгость включений следует из того, что в HeurP и AvgP лежат только разрешимые языки (с равномерным распределением), а в $\mathrm{Avg}_{\frac{1}{\mathrm{n^c}}}\mathrm{P}$ и $\mathrm{Heur_P} \subsetneq \mathrm{Heur}_{\frac{1}{\mathrm{n^c}}}\mathrm{P}$ содержатся (унарные) неразрешимые языки.

Можно показать, что $\mathrm{AvgP} \subsetneq \mathrm{HeurP}$. Это следует из следующей теоремы.

Теорема 1.2. Существует такой язык L, что $(L, U) \in \mathrm{HeurP}$, но $(L, U) \notin Avg_{\frac{1}{8}}P$.

Доказательство. Пусть M_i — это перечисление всех машин Тьюринга с будильником 2^n, ответ которых интерпретируется как элемент множества $\{0, 1, \perp\}$. Будем считать, что каждая машина встречается в этом перечислении бесконечное число раз.

Определим язык L. Язык L будет содержать не более, чем одну строчку каждой длины. Пусть x — это лексикографически первая строка длины n, на которой M_n не выдает $\perp$. Тогда включим x в L тогда и только тогда, когда M_n отвергает x. Если такой строки x нет, то язык L не будет иметь строк длины n совсем.

Легко понять, что $(L, U) \notin \mathrm{Avg}_{\frac{1}{8}}\mathrm{P}$ (каждая машина дает неверный ответ для бесконечного числа длин входов). Осталось показать, что $(L, U) \in \mathrm{HeurP}$. Действительно, алгоритм $A(x, \delta)$ выдает 0, если $\delta > 2^{-n}$ (мы обозначили $n = |x|$) и если $\delta \leq 2^{-n}$, то проверяет, не является ли x той строчкой, которая входит в L. На эту проверку потребуется $O(2^{3n})$ шагов, следовательно сложность алгоритма A ограничена $\left(\frac{|x|}{\delta}\right)^3$. □

1.1 Пример распределения, для которого сложность в худшем случае и в среднем совпадают

В этом разделе мы покажем, что существует ансамбль распределений, по отношению к которому понятия сложности в среднем и в худшем случае совпадают. По этой причине при изучении сложности в среднем обычно не рассматривают все возможные распределения, а рассматривают только некоторые их классы.

Временно будем кодировать пару строк (a, b) следующим образом: сначала запишем префиксный код $|a|$, затем префиксный код $|b|$, затем ab. Где префиксный код строки $s_1 s_2 \dots s_k$ — это строка $s_1 s_1 s_2 s_2 \dots s_k s_k 01$. Длина такой записи $|a| + |b| + 2\log a + 2\log b + O(1)$. Заметим, что при таком кодировании код никакой одной пары строк не является началом кода другой пары строк.

Для двоичной строки x обозначим через $K(x)$ длину самой короткой пары (M, w) такой, что машина Тьюринга M на входе w выдает x. Величина $K(x)$ называется префиксной колмогоровской сложностью строки x.

Лемма 1.1. $\sum_{x \in \{0,1\}^n} 2^{-K(x)} \leq 1$.

Доказательство. Для каждой строки x длины n выпишем код пары (M, w) на котором достигается значение $K(x)$. Множество этих строк будет префиксным, т.е. никакая строка не будет началом другой. Утверждение леммы следует из такого утверждения: для каждого префиксного множества битовых строк $\{w_1, w_2, \ldots, w_\ell\}$ выполняется неравенство $\sum_{i=1}^{\ell} 2^{-|w_i|} \leq 1$. Последнее утверждение легко доказывается по индукцией по суммарной длине строк. □

Теорема 1.3. Существует такой ансамбль распределений D, что для любого разрешимого языка L, если распределенная задача распознавания (L, D) лежит в $\mathrm{Heur}_{\frac{1}{n^3}}\mathrm{P}$, то $L \in \mathrm{P}$.

Доказательство. Определим

$$D_n(x) = \frac{2^{-K(x)}}{\sum_{y\in\{0,1\}^n} 2^{-K(y)}}$$

Пусть A это полниомиальный эвристический алгоритм, который подтверждает тот факт, что $(L, D) \in \mathrm{Heur}_{\frac{1}{n^3}}\mathrm{P}$. Покажем, что лишь для колечного числа строк x может выполняться $A(x) \neq L(x)$, из чего следует, что $L \in \mathrm{P}$.

Пусть x — это строка длины n такая, что $A(x) \neq L(x)$. Поскольку полная вероятность всех таких строк не более, чем $\frac{1}{n^3}$, то в частности мы имеем $D_n(x) \leq \frac{1}{n^3}$. Из леммы 1.1 следует, что $2^{-K(x)} \leq \frac{1}{n^3}$ иначе $K(x) \geq 3\log n$.

Рассмотрим лексикографически первую такую строчку $x \in \{0,1\}^n$ (если она есть), что $A(x) \neq L(x)$. Такая строка может быть вычислена алгоритмом, который получает n и вычисляет $A(x)$ и $L(x)$ для всех строк $x \in \{0,1\}^n$ и выводит лексикографически первый x такой, что $A(x) \neq L(x)$ (здесь мы используем предположение о разрешимости языка L). Существование такого алгоритма доказывает, что $K(x) \leq \log n + 2\log\log n + O(1)$, то есть для достаточно больших n это противоречит приведенному выше неравенству для $K(x)$.

Из этого мы делаем вывод о том, что существует только конечное число длин входов, на которых A и L отличаются, а значит и конечное число самих таких входов. □

2 Сведения и полная задача в $(\mathrm{NP}, \mathrm{PSamp})$

Определение 2.1. Ансамбль распределений D будем называть моделируемым за полиномиальное время, если существует такой полиномиальный вероятностный алгоритм S, распределение выхов которого на входе 1^n совпадает с D_n. Множество всех полиномиально моделируемых ансамблей будем обозначать PSamp.

Очевидно, что все строчки в носителе полиномиально моделируемого ансамбля D_n имеет длину $poly(n)$.

Будем обозначать $(\mathrm{NP}, \mathrm{PSamp})$ множество распределенных задач (L, D), в которых $L \in \mathrm{NP}$, $D \in \mathrm{PSamp}$.

Определение 2.2. Будем говорить, что ансамбль D_n доминирует ансамбль D'_n, если существует полиномы p, q, что для всех достаточно больших n для всех x выполняется неравенство $D_{p(n)}(x)q(n) \geq D'_n(x)$. Обозначение: $D \geq D'$.

Нетрудно проверить, что отношение доминирования распределений является транзитивным.

Лемма 2.1. Любой ансамбль из PSamp доминируется ансамблем, который генерируется за время n^2.

Доказательство. Пусть D — ансамбль, который задается генератором S, время работы которого ограничено n^c. Определим генератор S', который на входе 1^{n^c} запускает $S(1^n)$. □

Теорема 2.1. Существует ансамбль распределений $W \in$ PSamp, который доминирует любой ансамбль из PSamp.

Доказательство. Пусть S_i — это перечисление всех генераторов с будильником n^2. Опишем генератор S, который определяет ансамбль W. На входе 1^n генератор S с вероятностью $\frac{1}{2}$ запускает $S_1(1^n)$, с вероятностью $\frac{1}{4}$ запускает $S_2(1^n)$, ... с вероятностью $\frac{1}{2^{n-1}}$ запускает $S_{n-1}(1^n)$ и с вероятностью $\frac{1}{2^{n-1}}$ запускает $S_1(1^n)$. По предыдущей лемме любой полиномиально моделируемый ансамбль доминируется каким-то ансамблем с генератором S_i, а последний доминируется ансамблем W, для полиномов $p(n) = n$ и $q(n) = \frac{1}{2^i}$. □

Лемма 2.2. Пусть ансамбль D доминирует ансамбль D' и $(L, D) \in$ AvgP. Тогда $(L, D') \in$ AvgP. (Аналогичное утверждение верно и для HeurP.)

Доказательство. Пусть $A(x, \delta, 1^n)$ — это алгоритм, который решает задачу (L, D) согласно определению Импальяццо. Пусть для всех достаточно больших n выполняется $D_n(x)q(n) \geq D'_{p(n)}(x)$. Определим алгоритм $B(x, \delta, 1^n) = A(x, \delta/q(n), 1^{p(n)})$. Нужно проверить, что выполняется только последнее свойство: $\Pr_{x \leftarrow D'_n}[B(x, \delta, 1^n) = \perp] = \Pr_{x \leftarrow D'_n}[A(x, \delta/q(n), 1^{p(n)}) = \perp] = \sum_{x: A(x,\delta/q(n),1^{p(n)})=\perp} D'_n(x) \leq q(n) \sum_{x: A(x,\delta/q(n),1^{p(n)})=\perp} D_n(x) = q(n) \Pr_{x \leftarrow D_n}[A(x, \delta/q(n), 1^{p(n)}) = \perp] < \delta$. □

Определение 2.3. Будем говорить, что распределенная задача (L', D') сводится к задаче (L, D), если существует такая полиномиально вычислимая функция $f(x, 1^n)$, что для всех $x \in \operatorname{supp} D'_n$ выполняется $x \in L' \iff f(x, 1^n) \in L$ и ансамбль $f(D')$ доминируется ансамблем D, где $f(D')_n(y) = \sum_{x: f(x,1^n)=y} D'_n(x)$. Обозначение: $(L', D') \leq (L, D)$.

Предложение 2.1. Если (L', D') сводится к (L, D) и $(L, D) \in$ AvgP, то и $(L', D') \in$ AvgP. Аналогичное верно и для класса HeurP.

Доказательство. Следует из определения 2.3 и леммы 2.2. □

Теорема 2.2. Распределенная задача (SAT, W) является полной в классе (NP, PSamp) относительно сведений.

Доказательство. Для любого языка $L \in$ NP существует такая полиномиально вычислимая функция f, что для всех x выполняется $x \in L$ тогда и только тогда, когда $f(x) \in$ SAT. Распределение W доминирует любое распределение. □

Наша ближайшая цель построить полную задачу с простым распределением.

3 Полная задача с простым распределением

Определение 3.1. Ансамбль распределений D_n называется вычислимым за полиномиальное время, если вероятность любой строчки при распределении D_n является двоично рациональным числом и функция распределения ($F_{D_n}(x) = \sum_{y<x} D_n(y)$ где $<$ — это лексикографический порядок на строках) вычислима за полиномиальное время. Множество полиномиально вычислимых ансамблей будем обозначать PComp.

Пусть S — это генератор для некоторого распределения. Если S, используя случайную строку r, генерирует строку x, то будем считать, что генератор отображает отрезок $[0.r0, 0r1)$ в строку x.

Определение 3.2. Полиномиально моделируемый ансамбль D_n называется обратимым, если генератор S для этого ансамбля обладает такими свойством: прообраз каждой строки x — это один отрезок, концы которого вычисляются за полиномиальное от n время. Множество обратимых полиномиально моделируемых ансамблей будем обозначать PISamp.

Нетрудно доказать следующее утверждение:

Предложение 3.2. PComp $\subseteq$ PISamp $\subseteq$ PSamp.

Мы построим полную распределенную задачу в классе (NP, PISamp). Эта же задача будет полной и в классе (NP, PComp).

Будем называть отрезок стандартным, если он имеет вид $[0.r0, 0.r1)$, где r — это строка из 0 и 1, которую мы будем называть кодом стандартного отрезка.

Лемма 3.1. Пусть I — это отрезок в $[0, 1]$ с двоично рациональными концами. Тогда существует стандартный отрезок, лежащий в I, длина которого не меньше, чем $\frac{1}{4}|I|$ и найти этот стандартный отрезок можно за полиномиальное от длины записи I время.

Доказательство. Стандартных интервалов длины 2^{-k} ровно 2^k штук и они плотно прилегают друг к другу. Если максимальный стандартный интервал меньше четверти интервала I, то в I влезает как минимум три подряд идущих стандартных интервала. А из трех подряд идущих стандартных интервалов два из них образуют больший стандартный интервал. Таким образом, возможных значений для длины стандартных интервалов остается немного и найти такой интервал становится просто. □

Рассмотрим задачу об ограниченной остановке BH $= \{(M, x, 1^n) \mid$ недетерминированная машина M принимает x за не более, чем n шагов$\}$. Определим ансамбль распределений $\hat{U}$. $\hat{U}_n$ будет иметь ненулевые значения только на входах вида $(M, x, 1^n)$, где $|M| \leq n$ и $|x| \leq n$. При выполнении этих условий $\hat{U}_n(M, x, 1^n) = 2^{-|M|-|x|-2\log n}$.

Легко показать, что $\hat{U} \in$ PISamp. Действительно, генератор для распределения $\hat{U}$ устроен так: на входе 1^n генератор генериует два случайных числа i, j из множества $\{1, \ldots, n\}$. Затем генерирует случайную строку M из $\{0, 1\}^i$ и случайную строку x из $\{0, 1\}^j$, на выход выдается $(M, x, 1^n)$. Обратимость этого распределения очевидно, так как по входу $(M, x, 1^n)$ очевидно, из какой случайной строки он получен. Нетрудно также понять, что $\hat{U} \in$ PComp.

Теорема 3.1. Задача $(\text{BH}, \hat{U})$ полна в классе (NP, PISamp).

Доказательство. Пусть $(L, D) \in (\mathrm{NP}, \mathrm{PISamp})$. Пусть язык L решается недетерминированной машиной тьюринга M за время $p(n)$. Пусть S — это генератор распределения D, время его работы на входе 1^n ограничено полиномом $s(n)$. Для $x \in \operatorname{supp} D_n$ обозначим через r_x код наибольшего стандартного интервала, который содержится в интервале, который $S(1^n)$ отображает на x. Из леммы 3.1 следует, что $2^{-|r_x|} \leq D_n(x) \leq 2^{-|r_x|+2}$

Будем обозначать через $\hat{n}$ префиксный строки n (каждый бит удваиваем и приписываем 01). Сведение f определяется так: $f(x, 1^n) = (N, \hat{n}r_x, 1^{p(n)+s(n)})$, где N — это такая недетерминированная машина, которая сначала запускает $S(1^n)$ со случайными битами r_x и получает x, после этого запускает M на входе x. Нетрудно понять, что если $x \in L$, то $f(x, 1^n) \in \mathrm{BH}$, осталось проверить доминирование распределений. Функция f инъективна, поэтому условие доминирования следует из того, что $\hat{U}_{p(n)+s(n)}(N, \hat{n}r_x, 1^{p(n)+s(n)}) = 2^{-|N|-2\log n-2-|r_x|-2\log(p(n)+s(n))} = \frac{2^{-|r_x|+2}}{2^{|N|}n^2} \geq \frac{D_n(x)}{2^{|N|}n^2(p(n)+s(n))^2}$. Осталось заметить, что $|N|$ — это константа.

□

4 Вероятностные классы задач распознавания и поиска

Дадим определения классов распределенных задач распознавания, решающихся *вероятностными* алгоритмами за полиномиальное в среднем время.

Определение 4.1. Распределенная задача (L, D) содержится в классе AvgBPP, если существует вероятностный алгоритм $A(x, \delta, 1^n)$:

1. Время работы $A(x, \delta, 1^n)$ ограниченно $poly(n, 1/\delta)$ при $x \in \operatorname{supp} D_n$;
2. $\forall x \in \operatorname{supp} D_n$ верно, что $\Pr_r[A(x, \delta, 1^n) \notin \{L(x), \perp\}] < \frac{1}{8}$;
3. $\Pr_{x \leftarrow D_n}[\Pr_r[A(x, \delta, 1^n) = \perp] \geq \frac{1}{8}] < \delta$, тут внутренняя вероятность берется по случайным битам алгоритма A.

Это определение дано в стиле Импальяццо. Эквивалентное определение в стиле Левина выглядело бы так: пусть $t_A(x)$ — это случайная величина, которая обозначает время работы алгоритма A на входе x. Пусть $\tau_A(x)$ — это медиана случайной величины $t_A(x)$. По Левину распределенная задача (L, D) принадлежит классу AvgBPP, если существует алгоритм A, что на каждом входе $x \in \operatorname{supp} D$ выполняется $\Pr[A(x) = L(x)] \geq \frac{3}{4}$ и функция $\tau_A(x)$ полиномиальна в среднем при распределении D, т.е. существует $\epsilon > 0$, что $\mathrm{E}_{x \leftarrow D_n}[\tau_A(x)^\epsilon] = O(n)$.

Определение 4.2. Распределенная задача (L, D) содержится в классе HeurBPP, если существует вероятностный алгоритм $A(x, \delta, 1^n)$:

1. Время работы $A(x, \delta, 1^n)$ ограниченно $poly(n, 1/\delta)$ при $x \in \operatorname{supp} D_n$;
2. $\Pr_{x \leftarrow D_n}[\Pr_r[A(x, \delta, 1^n) \neq L(x)] > \frac{1}{4}] < \delta$, тут внутренняя вероятность берется по случайным битам алгоритма A.

Легко понять, что AvgBPP $\subseteq$ HeurBPP.

С помощью повторений алгоритмов можно легко убедиться, что эквивалентно эти два класса можно было бы определить так:

Определение 4.3. Распределенная задача (L, D) содержится в классе AvgBPP, если существует вероятностный алгоритм $A(x, \delta, 1^n)$:

1. Время работы $A(x, \delta, 1^n)$ ограниченно $poly(n, 1/\delta)$ при $x \in \operatorname{supp} D_n$;
2. $\forall x \in \operatorname{supp} D_n$ верно, что $\Pr_r[A(x, \delta, 1^n) \notin \{L(x), \perp\}] < 2^{-n}$;
3. $\Pr_{x \leftarrow D_n, r}[A(x, \delta, 1^n) = \perp] < \delta$.

Определение 4.4. Распределенная задача (L, D) содержится в классе HeurBPP, если существует вероятностный алгоритм $A(x, \delta, 1^n)$:

1. Время работы $A(x, \delta, 1^n)$ ограниченно $poly(n, 1/\delta)$ при $x \in \operatorname{supp} D_n$;
2. $\Pr_{x \leftarrow D_n, r}[A(x, \delta, 1^n) \neq L(x)] < \delta$.

До сих пор мы занимались распределенными задачами распознавания. Определим теперь понятие распределенной задачи поиска. Мы будем говорить только об NP-задачах поиска. Пусть Q_L полиномиально проверяемый и полиномиально ограниченный предикат, ему соответствует язык $L = \{x \mid \exists y : Q_L(x, y) = 1\}$. Будем обозначать $\widetilde{(L, D)}$ распределенную задачу поиска подсказки. Если $x \in L$, то решением задачи поиска является любая строка y, что выполняется $Q_L(x, y) = 1$. Если $x \notin L$, то решением является любая строка.

Определение 4.5. Распределенная задача поиска $\widetilde{(L, D)}$ содержится в классе $\widetilde{\text{AvgP}}$, если существует алгоритм $A(x, \delta, 1^n)$:

1. Время работы $A(x, \delta, 1^n)$ ограниченно $poly(n, 1/\delta)$ при $x \in \operatorname{supp} D_n$;
2. $\forall x \in \operatorname{supp} D_n \cap L$ верно, что или $A(x, \delta, 1^n) = \perp$ или $Q_L(x, A(x, \delta, 1^n)) = 1$
3. $\Pr_{x \leftarrow D_n}[A(x, n, \delta) = \perp] < \delta$

Определение 4.6. Распределенная задача поиска $\widetilde{(L, D)}$ содержится в классе $\widetilde{\text{HeurP}}$, если существует алгоритм $A(x, \delta, 1^n)$:

1. Время работы $A(x, \delta, 1^n)$ ограниченно $poly(n, 1/\delta)$ при $x \in \operatorname{supp} D_n$;
2. $\Pr_{x \leftarrow D_n}[x \in L \wedge Q_L(x, A(x, \delta, 1^n)) = 0] < \delta$

Главным отличием эвристических и полиномиальных в среднем алгоритмов для задач поиска являются поведение на отрицательных входах. В определении эвристических алгоритмов нет никаких условий на результат, если $x \notin L$. В определении полиномиальных в среднем алгоритме различаются ответы $\perp$ и другие ответы, причем слишком часто отвечать $\perp$ нельзя.

Стоит отметить, что если носитель распределения содержится в L, тогда принадлежность задачи поиска классу $\widetilde{\text{HeurP}}$ и $\widetilde{\text{AvgP}}$ эквивалентно, поскольку правильность подсказки можно проверить.

Осталось определить вероятностные классы для задач поиска.

Определение 4.7. Распределенная задача поиска $\widetilde{(L, D)}$ содержится в классе $\widetilde{AvgBPP}$, если существует вероятностный алгоритм $A(x, \delta, 1^n)$:

1. Время работы $A(x, \delta, 1^n)$ ограниченно $poly(n, 1/\delta)$ при $x \in \operatorname{supp} D_n$;
2. $\forall x \in \operatorname{supp} D_n \cap L$ верно, что $\Pr_r[A(x, \delta, 1^n) \neq \perp \wedge Q_L(x, A(x, \delta, 1^n) = 0)] \leq \frac{1}{4}$;
3. $\Pr_{x \leftarrow D_n}[\Pr[A(x, \delta, 1^n) = \perp] \geq \frac{1}{4}] < \delta$.

Определение 4.8. Распределенная задача поиска $\widetilde{(L, D)}$ содержится в классе $\widetilde{\text{HeurBPP}}$, если существует вероятностный алгоритм $A(x, n, \delta)$:

1. Время работы $A(x, \delta, 1^n)$ ограниченно $poly(n, 1/\delta)$ при $x \in \operatorname{supp} D_n$;
2. $\Pr_{x \leftarrow D_n}[x \in L \wedge \Pr[Q_L(x, A(x, \delta, 1^n)) = 0] \geq \frac{1}{4}] < \delta$.

Аналогично тому, как это делается для задач распознавания можно показать, что второе условие в определении класса $\widetilde{\text{HeurBPP}}$ можно эквивалентным образом заменить на такое: $\Pr_{x \leftarrow D_n}[x \in L \wedge Q_L(x, A(x, n, \delta)) = 0] < \delta$. Мы будем использовать каждый раз то определение, которое удобно в конкретном случае.

В дальнейшем мы будем говорить, что распределенная задача распознавания (или поиска) простая в среднем, если она лежит в классе HeurBPP (или $\widetilde{\text{HeurBPP}}$).

5 Сведение задач поиска к задачам распознавания

Теорема 5.1. Если $(\text{NP}, \text{U}) \subseteq \text{HeurBPP}$, то $\widetilde{(\text{NP}, \text{U})} \subseteq \widetilde{\text{HeurBPP}}$. Аналогичное верно и для AvgBPP.

В принципе мы могли бы не ограничиваться равномерными распределениями, но позже мы покажем, что если $\widetilde{(\text{NP}, \text{U})} \subseteq \widetilde{\text{HeurBPP}}$, то и $\widetilde{(\text{NP}, \text{PSamp})} \subseteq \widetilde{\text{HeurBPP}}$.

Доказательство. Напомним, что стандартное сведение задачи поиска к задаче распознавания состоит к поиску подсказки бинарным поиском. А именно для языка L создается язык L', который состоит из пар (x, a), что для x есть L-подсказка, которая лексикографически предшествует a. На языке сложности в среднем это доказательство изложить не удается, поскольку непонятно как определить распределение, чтобы вероятность строки (x, a) не была бы экспоненциально меньше, чем вероятность строки x. Напомним, что нельзя сводить входу к входу с экспоненциально меньшей вероятностью.

Предположим, что подсказка для каждого x ровно одна. В таком случае можно узнать подсказку, делая запросы к принадлежности к языку, состоящему из пар (x, i), что у x есть L-подсказка i-й бит которой равен единице. Это уже значительно лучше, поскольку вероятность строки (x, i) можно сделать лишь в полином меньше, чем вероятность строки x. Чтобы добиться того, чтобы подсказка стала одна мы применем хеш-функции (лемму Вэлианта-Вазирани).

Определение 5.1. Семейством попарно независимых хеш-функций $H_{n,k}$ называется множество функций $\{0,1\}^n \to \{0,1\}^k$, $k \leq n$ такое, что:

1. $\forall x \in \{0,1\}^n$, $\forall a \in \{0,1\}^k$ верно, что $Pr_{h \leftarrow H_{n,k}}[h(x) = a] = 2^{-k}$
2. $\forall x' \neq x'' \in \{0,1\}^n$, $\forall a', a'' \in \{0,1\}^k$ верно, что $Pr_{h \leftarrow H_{n,k}}[h(x') = a' \wedge h(x'') = a''] = 2^{-2k}$

Первое из этих свойств следует из второго. Важным требованием к семейству является его небольшой размер. В следующем примере каждая хеш-функция задается строкой длины $2n$.

Пример 5.1. Множество линейных функций $h_{a,b} = (ax + b)_{\leq k}$ (операции в поле F_{2^n}) с усеченными образами представляет собой семейство попарно независимых хэш функций.

Лемма 5.1 (Вэлианта-Вазирани)**.** Пусть $S \subset \{0,1\}^n$, $2^{k-2} \leq |S| \leq 2^{k-1}$ и $H_{n,k}$ - семейство попарно независимых хеш-функций. Тогда $\Pr_{h \leftarrow H_{n,k}}[\exists! x \in S : h(x) = 0^k] \geq \frac{1}{8}$.

Пусть $L \in$ NP, пусть длина подсказки к слову $x \in L$ ограничена полиномом $p(|x|)$. Определим язык $L' = \{(x,h,i,k) \mid x \in L, h \in \{0,1\}^{2n}, 1 \leq i \leq p(n), 2 \leq k \leq p(n)+1, \text{существует } w - L\text{-подсказка для x}, h|_k(w) = 0^k, w_i = 1\}$. Здесь $h|_k$ кодирует хеш-функцию из примера 5.1. Запятые можно не кодировать, так как части однозначно следуют из длины строки.

Очевидно, что $L' \in$ NP. Пусть алгоритм $B(x,\delta)$ решает (L',U) в HeurBPP. Опишем алгоритм $A(x,\delta)$.

1. Повторить N раз:
2. Для всех $k \in \{2, p(n)+1\}$
   - Выберем $h \leftarrow U_n$ и r — случайная строка, которая будет использоваться для запусков алгоритма B.
   - Для всех $i \in \{1, p(n)\}$
     - $u_i = B_r((x,h,i,k),\delta')$
   - Если $u_1 u_2 \dots u_{p(n)}$ — подсказка для x, выдать ее.

Значения параметров N и δ' мы выберем позже.

Пусть S_x — это множество подсказок для x. Пусть k такое число, что выполняется неравенство $2^{k-2} \leq |S| \leq 2^{k-1}$ и $H_{n,k}$.

Временно положим $N = 1$. В этом случае вероятность (по x и случайным битам алгоритма A) того, что $x \in L$ и алгоритм A не выдаст подсказку не превосходит вероятности того, что алгоритм не выдаст подсказку на итерации с правильным k. При правильном k событие неудачи содержится в объединении следующих событий: хеш-функция h неудачная (вероятность этого события не больше $\frac{7}{8}$) и для некоторого i $B((x,h,i,k),\delta')$ выдает неправильный ответ. Вторую вероятность можно оценить как $p^2(n) \Pr_{x,h,i,k,r}[B_r((x,h,i,k),\delta')$ выдает неправильный ответ$]$, что не превосходит $p^2(n)\delta'$. Выберем $\delta' = \frac{1}{16p^2(n)}$. Получили, что при $N = 1$ вероятность того, что алгоритм не найдет подсказку не больше $\frac{15}{16}$. Выбрав $N = O(\log \frac{1}{\delta})$ добьемся того, чтобы вероятность ошибки стала меньше, чем δ. □

6 Сведение к равномерному распределению

Для задач поиска мы определим вероятностные сведения. Они будут замкнуты относительно класса $\widetilde{\text{HeurBPP}}$.

6.1 Вероятностное сведение

Определение 6.1. Задача поиска $\widetilde{(L, D)}$ вероятностно сводится к задаче поиска $\widetilde{(L', D')}$, если существует пара алгоритмов $f(x, n)$ — вероятностный алгоритм, который преобразует входы первой задачи во входы второй задачи, g — дететерменированный алгоритм, который преобразует подсказки для второй задачи в подсказки для первой задачи. Кроме того для каждого $x \in L$ определено множество $V_x \subseteq \{0,1\}^*$, которое соответсвует удачным «сведениям». Выполняются следующие свойства:

1. (Инъективность) $\forall x, x' \in L, x \neq x' : V_x \cap V_{x'} = \emptyset$
2. (Объемность) найдется полином q_1, что $\forall x \in \operatorname{supp} D_n$ выполняется $\Pr[f(x,n) \in V_x] \geq 1/q_1(n)$
3. (Доминирование) найдется пара полиномов $p(n)$ и $q_2(n)$, что выполняется неравенство $D_n(x) \leq q_2(n) D'_{p(n)}(V_x)$
4. (Равномерность) для $y \in V_x$ выполняется $\Pr[f(x,n) = y \mid f(x,n) \in V_x] = \frac{D'_{p(n)}(y)}{D'_{p(n)}(V_x)}$
5. (Полезность) для всех $x \in L$ для всех $y \in V_x$ выполняется $y \in L'$ и для любой L'-подсказки w для строки y строка $g(w, y, x)$ является подсказкой для x.

Вероятностное сведение действует следующим образом. Функция f переводит слова из языка L в слова из языка L'. При этом f может ошибаться, т.е. переводить слова из L в $\bar{L}'$, но делает это не слишком часто. Фунция g переводит подсказки для языка L' в подсказки для языка L.

Лемма 6.1. Пусть $\widetilde{(L, D)}$ сводится к $\widetilde{(L', D')}$ и $\widetilde{(L', D')} \in \widetilde{\text{HeurBPP}}$, тогда $\widetilde{(L, D)} \in \widetilde{\text{HeurBPP}}$.

Доказательство. Пусть $A'(x, \delta, 1^n)$ — алгоритм для $\widetilde{(L', D')}$. Построим алгоритм $A(x, \delta, 1^n)$ для $\widetilde{(L, D)}$. Независимо $N = 16q_1(n)$ раз выберем $y_1 = f(x,n), y_2 = f(x,n), \ldots, y_N = f(x,n)$. Для каждого y_i запустим $A'(y_i, \frac{\delta}{2q_2(n)}, 1^{p(n)}) = w_i$. Если найдется такое i, что $g(w_i, y_i, x)$ — подсказка для x, то возвращаем $g(w_i, y_i, x)$, иначе 0.

Оценим вероятность того, что $x \in L$ и алгоритм $A(x, \delta, 1^n)$ не выдает подсказку для x. Эта вероятность не превосходит вероятности того, что ни один из y_i не попадет в V_x и вероятности дать неверный ответ при условии, что при некотором i выполняется $y_i \in V_x$.

Пусть $x \in L$, обозначим через F множество строк на которых $A'(y, \frac{\delta}{2q_2(n)}, 1^{p(n)})$ ведет себя некорректно. Формально $F = \{y \in L' \mid \Pr[A'(y, p(n), \frac{\delta}{2q_2(n)}$не выдает подсказку для $y] \geq 1/4\}$. Сразу можно заключить, что $D'_{p(n)}(F) < \frac{\delta}{2q_2(n)}$

Определим множество B — множество плохих входов: $B = \{x \in L \mid D'_{p(n)}(V_x \cap F) \geq D'_{p(n)}(V_x)/2\}$.

$$D_n(B) = \sum_{x \in B} D_n(x) \overset{\text{доминирование}}{\leq} q_2(n) \sum_{x \in B} D'_{p(n)}(V_x) \leq$$

$$2q_2(n) \sum_{x \in B} D'_{p(n)}(V_x \cap F) \overset{\text{инъективность}}{\leq} 2q_2(n) D'_{p(n)}(F) < \delta.$$

Покажем, что при $x \notin B$ подсказка будет найдена с вероятностью не меньше $\frac{3}{4}$. Для каждого $i \in \{1, 2, \ldots, N\}$ оценим вероятность того, что $g(w_i, y_i, x)$ — это подсказка для x.

$$\Pr[g(w_i, y_i, x) \text{ — это подсказка для } x] \geq \Pr[y_i \in V_x \wedge w_i \text{ — это подсказка для } y_i] \geq$$

$$\Pr[y_i \in V_x \setminus F \wedge w_i \text{ — это подсказка для } y_i] =$$

$$\Pr[y_i \in V_x] \Pr[y_i \in V_x \setminus F \mid y_i \in V_x] \Pr[w_i \text{ — это подсказка для } y_i \mid y_i \in V_x \setminus F] \overset{\text{объемность}}{\geq}$$

$$\frac{1}{q_1(n)} \Pr[y_i \in V_x \setminus F \mid y_i \in V_x] \Pr[w_i \text{ — это подсказка для } y_i \mid y_i \in V_x \setminus F] \overset{\text{равномерность}}{\geq}$$

$$\frac{1}{q_1(n)} \frac{1}{2} \Pr[w_i \text{ — это подсказка для } y_i \mid y_i \in V_x \setminus F] \overset{\text{определение } F}{\geq} \frac{1}{q_1(n)} \frac{1}{2} \frac{3}{4} = \frac{3}{8q_1(n)}$$

Поскольку мы повторяем все N раз, то вероятность того, что подсказка будет найдена хотя бы для одного i не меньше, чем $\frac{3}{4}$. □

Будем обозначать $S_n = \{0,1\}^n$ и оттождествлять S_n с полем из 2^n элементов, в котором сложение покоординатное. Если $k \leq n$, то будем обозначать $S_k = \{0,1\}^k$, мы считаем, что есть стандартное вложение S_k в S_n — первые k битов прежние, оставшиеся биты — нули.

Лемма 6.2. Пусть $X \subseteq \{0,1\}^n, |X| \geq 2^{n+1-k}$. Тогда размер множества пар $\{(a,b) \in \{0,1\}^n \mid \exists r \in \{0,1\}^{k-1} : ar + b \in X\}$ больше либо равно $2^{2n}/2$.

Доказательство. Рассмотрим множество пар $\{(r,p) \mid r \in S_{k-1}, p \in X\}$, всего этих пар 2^n штук. Для каждой пары (r,p) существует 2^n пар (a,b), удовлетворяющих условию $ar + b = p$. Действительно, для каждого $a \in \{0,1\}^n$ однозначно определяется $b = p - ar$. При этом для двух различных пар $(r_1, p_1), (r_2, p_2)$ может существовать не более одной подходящей пары (a,b), поскольку, если

$$\begin{cases} ar_1 + b = p_1 \\ ar_2 + b = p_2, \end{cases}$$

то $a(r_1 - r_2) = p_1 - p_2$. Если $r_1 = r_2$, то $p_1 = p_2$, в противном случае однозначно находятся a и b.

По формуле включений-исключений интересуеще множество пар можно оценить снизу как число пар, соответсвующее каждой паре (r,p) минус число пар во всех попарных пересечениях. Таким образом число пар не меньше, чем существует не менее $2^n \times 2^n - 2^n \times (2^n - 1)/2 > 2^{2n}/2$. □

Теорема 6.1. Для любой задачи $\widetilde{(L, D)} \in (\mathrm{NP}, \widetilde{\mathrm{PSamp}})$ найдется такой язык $L' \in \mathrm{NP}$, что задача $\widetilde{(L, D)}$ сводится к $\widetilde{(L', U)}$

Доказательство. Пусть распределение D_n генерируется самплером R, пусть многочлен $m(n)$ с запасом больше, чем время работы самплера R, когда он генерирует строки распределения D_n. Далее мы будем считать, что R принимает на вход строку $\{0,1\}^{m(n)}$ и использует ее как случайные биты. Для $x \in \operatorname{supp} D_n$ обозначим через $pad_n(x)$ строку $x01^{m(n)-|x|-2}$.

Определим язык L' следующим образом:

$L' = \{(k, \rho, h, h_k(pad_n(x)), a, b) \mid 1 \leq k \leq m(n), \rho \in \{0,1\}^{m-k}, x \in L, h \in \{0,1\}^{2m}, a, b \in S_m, \exists r \in S_{k-1} : x = R(ar + b)\}$. Это определение требует пояснений:

1. k — это число от 1 до $m(n)$, которое представлено битовой строкой длины $\log m$;
2. h — это элемент семейства попарно независимых хеш-функций $H_{m,m}$, мы считаем, что каждый элемент этого множества записывается битовой строкой длины $2m(n)$.
3. h_k — это элемент семейства попарно независимых хеш-функций $H_{m,k}$, эта функция получается из h отбрасыванием последних $m - k$ битов.
4. ρ — это «мусор», цель которого выровнять длину строки до $5m + \log m$;
5. Все элементы картежа $(k, \rho, h, h_k(pad_n(x)), a, b)$ имеют фиксированный размер, поэтому необязательно каким-то особенным образом кодировать картеж;
6. a, b — произвольные два элемента S_m.

$L' \in \mathrm{NP}$, т.к. сертификатом для $(k, \rho, h, h_k(pad_n(x)), a, b)$ является x, L-подсказка w и r такое, что $x = R(ar + b)$.

Теперь опишем вероятностное сведение:

$f(x, n) = (k, \rho, h, h_k(pad_n(x)), a, b)$, где сначала строка k выбирается случайно из строк длины $\log m$, затем $\rho \leftarrow U_{m-k}$, $h \leftarrow U_{2m}$, $a, b \leftarrow U_m$, $k \leftarrow U_{\log n}$.

Функция g определяется тривально, поскольку подсказка для $f(x, n)$ содержит подсказку для x.

Опишем множество V_x с помощью условий, которым должна удовлетворять строка $f(x)$:

1. $2^{m+2-k} \geq |R^{-1}(x)| \geq 2^{m+1-k}$;
2. Существует $r \in S_{k-1}$, что $R(ar + b) = x$;
3. Если $R(ar' + b) \neq x$ для некоторого $r' \in S_{k-1}$, то $h_k(R(ar' + b) \neq h_k(pad_n(x))$.

Проверим, что выполняются все условия определения 6.1.

(Объемность) Вероятность того, что k удовлетворяет второму условию не меньше, чем $\frac{1}{m}$. При условии того, что выполняется первое условие, вероятность второго как минимум $\frac{1}{2}$ по лемме 6.2. При условии того, что выполняется первые два условия, вероятность того, что выполняется третье условие как минимум $1 - \frac{1}{2^k}$, это следует из того, что h_k — это элемент попарно независимого семейства хеш-функций. Отсюда следует, что $\Pr[f(x, n) \in V_x] \geq \frac{1}{4m(n)}$.

(Инъектовность) Следует из третьего условия.

(Полезность) Очевидно.

(Равномерность) Следует из того, что все значения из V_x принимаются с одинаковыми вероятностями.

(Доминирование) Пусть k_0 такое число, что выполняется неравенство $2^{m+2-k_0} \geq |R^{-1}(x)| \geq 2^{m+1-k_0}$, тогда $D_n(x) \leq 2^{-k_0+1}$. Пусть $p(n) = 5m(n) + \log m(n)$. Обозначим через F_x множество всех значений, которое может принимать $f(x,n)$, все они принимаются с равными вероятностями. Пусть $F_{x,0}$ — это подмножество F_x в котором $k = k_0$. Элементы $F_{x,0}$ устроены следующим образом: в них фиксирована компонента $k = k_0$ и компонента $h_k(pad_n(x))$ зависит от h, остальные компоненты могут быть любыми, отсюда $U_{p(n)}(F_{x,0}) \geq 2^{-q}\frac{1}{m(n)}$, из объемности следует, что $U_{p(n)}(V_x) \geq 2^{-q}\frac{1}{4m(n)}$. Итого, получаем, что $D_n(x) \leq U_{p(n)}(V_x)8m(n)$. □

Таким образом, любая задача поиска $\widetilde{(L,D)}$ сводится к задаче поиска $\widetilde{(L',U)}$, которая в свою очередь сводится к задаче распознавания (L'',U), которая сводится к задаче $(BH,\hat{U})$. Все эти сведения замкнуты относительно класса HeurBPP ($\widetilde{\text{HeurBPP}}$).

Не смотря на то, что вероятностные сведения определяются для задач поиска, а не для задач распознавания, для задач распознавания сведение тоже есть, так как если для некоторой задачи $(L,D) \in (\text{NP}, \text{PSamp})$ выполняется, что задача поиска $\widetilde{(L,D)} \in (\widetilde{\text{HeurBPP}})$, то $(L,D) \in$ HeurBPP.

Естественный вопрос, обязательно ли нужны именно вероятностные сведения для того, чтобы была полная задача с равномерным распределением. Оказывается, что ответ на этот вопрос положительный, если EXP $\neq$ NEXP.

Будем называть ансамбль распределений плоским, если существует такая константа c, что для любой строки x выполняется $D_n(x) \leq 2^{-|x|^c}$.

Теорема 6.2. Если существует полная задача в $(\text{NP}, \text{PSamp})$ относительно детерминированных сведений (согласно определению 2.3) с плоским распределением, то NEXP = EXP.

Доказательство. Рассмотрим язык $L \in$ NEXP, пусть этот язык решается за время 2^{n^c} на недерминированной машине Тьюринга. Рассмотрим язык $L' = \{x01^{2^{|x|^c}} \mid x \in L\}$. Ясно, что $L' \in$ NP. Рассмотрим ансамбль распределений D_n, который определяетс так: $D_n(x01^{2^{|x|^c}}) = 2^{-|x|}$, где $n = |x|+1+2^{|x|^c}$. Очевидно, что $(L',D) \in (\text{NP}, \text{PSamp})$. Пусть (A,F) — полная задача в $(\text{NP}, \text{PSamp})$, ансамбль F плоский. Существует такая константа d, что $F_n(y) \leq 2^{-|y|^d}$. Пусть f — это сведение задачи (L',D) к задаче (A,F), пусть $f(x01^{2^{|x|^c}}) = y$. Из доминирования распределений следует, что существует такой полином p и q, что $F_{q(n)}(y)p(1+|x|+2^{|x|^c}) \geq 2^{-|x|}$, поскольку ансамбль F плоский, имеем $2^{-|y|^d} \geq F_{q(n)}(y)$. Отсюда следует, что для некоторой константы a выполняется неравенство $2^{|y|^d} \leq 2^{|x|^a}$, следовательно $|y| \leq poly(|x|)$. Таким образом мы имеем, что $x \in L$ тогда и только тогда, когда $g(x) = f(x01^{2^{|x|^c}}) \in A$. Про функцию g известно, что g вычислима за экспоненциальное время и $|g(x)| \leq poly(|x|)$. Поскольку $A \in \text{NP} \subseteq \text{EXP}$, то $L \in$ EXP. □

7 Односторонние функции и сложность в среднем

В этом разделе мы рассматриваем односторонние функции против равномерного противника.

Теорема 7.1. Если существуют односторонние функции, то $(\text{NP}, \text{U}) \notin$ HeurBPP.

Доказательство. Пусть $f_n : \{0,1\}^n \to \{0,1\}^n$ сильно односторонняя функция. Обозначим через IF язык образов функции f, т.е. $IF = f(\{0,1\}^*)$. Из определения сильной односторонней функции следует, что задача поиска $\widetilde{(IF, f(U))} \notin \widetilde{\text{HeurBPP}}$. Задача поиска $\widetilde{(IF, f(U))}$ вероятностно сводится к задаче поиска $\widetilde{(L,U)}$ для некоторого языка $L \in \text{NP}$, которая в свою очередь сводится к задаче распознавания (L', U) для языка $L' \in \text{NP}$. Из того, что $\widetilde{(IF, f(U))} \notin \widetilde{\text{HeurBPP}}$ следует, что $\widetilde{(L,U)} \notin \widetilde{\text{HeurBPP}}$, а из этого следует, что $(L', U) \notin \text{HeurBPP}$. □

Интереснее, конечно, иметь обратное следствие, а именно построить одностороннюю функцию, основываясь на трудной в среднем задаче. К примеру построить функцию, которая будет односторонней, если $(\text{NP}, \text{PSamp}) \notin \text{HeurBPP}$. Можно было бы надеятся построить такую функцию на основе одной из полных задач, к примеру на задаче BH. На данный момент сделать это не удается, основные проблемы следующие:

1. В криптографии и в теории сложности разные понятия о том, что значит решить задачу. В теории сложности алгоритм решает задачу, если он решает ее для всех длин входов. А в криптографии противник считается успешным, если он взламывет функцию на бесконечном числе входов.

2. Другая проблема состоит в том, что в трудной в среднем задаче распределение задано на экземплярах задачи, а в задаче, соотвтествующей задаче обращения односторонней функции, распределения берется по входам функции, т.е. по подсказкам.

На сегодняшний день неизвестно, как бороться с этими двумя проблемами. Мы определим ослабленное понятие односторонней функции — бесконечно часто односторонние функции.

Определение 7.1. Функция $f_n : \{0,1\}^n \to \{0,1\}^n$ называется бесконечно часто слабо односторонней, если она вычислима за полиномиальное время и существует полином $p(n)$, что для любого вероятностного полиномиального алгоритма B для бесконечного числа n выполняется $\Pr_{x \leftarrow U_n}[B(f(x)) \notin f^{-1}(f(x))] \geq \frac{1}{p(n)}$.

Аналогично классическому случаю можно определить понятие бесконечно часто сильно односторонней функции и доказать эквивалентность их существования.

Пусть f — полиномиально вычислимая функция, $f_n : \{0,1\}^n \to \{0,1\}^n$. Пусть $IF = f(\{0,1\}^*)$. Допусим известно, что $\widetilde{(IF, f(U))} \notin \widetilde{\text{HeurBPP}}$, означает ли это, что функция f является слабой бесконечно часто односторонней? Заметим, что распределение сосредоточено только на положительных экземплярах задачи (когда прообраз есть), для таких задач принадлежности классам $\widetilde{\text{HeurBPP}}$ и $\widetilde{\text{AvgBPP}}$ эквивалентны. Представим себе, что функция f легко (за время $O(n)$) обращается на доле входов $1 - \frac{1}{2^{\sqrt{n}}}$ и трудно обращается (за время $\Omega(2^n)$) на доле входов $\frac{1}{2^{\sqrt{n}}}$. Так определенная функция f не будет слабой односторонней, поскольку трудных входов слишком мало. Обратить функцию f за полиномиальное в среднем время тоже не удается, поскольку для каждого положительного ϵ математическое ожидание времени работы в степени ϵ будет как минимум $2^{\epsilon n - \sqrt{n}}$.

Однако можно показать, что функцию f можно легко модифицировать так, чтобы она стала бесконечно часто слабой односторонней. Определим функцию g следующим образом $g(x,r) = f(x, 1^{|r|})$. Если вход y функции g не является кодом пары, то $g(y) = y$. Пары мы будем кодировать следующим образом: (a,b) кодируется строчкой $\ell\widehat{(a)}ab$, где $\ell\widehat{(a)}$ — это префиксный код длины строки a. Длина кода пары равняется $|a| + |b| + 2\log|a| + 1$.

Теорема 7.2. Пусть f — полиномиально вычислимая функция, $f_n : \{0,1\}^n \to \{0,1\}^n$. Пусть $IF = f(\{0,1\}^*)$. Пусть $(\widetilde{IF, f(U)}) \notin \widetilde{\text{HeurBPP}}$. Рассмотрим функцию g, которая определяется так: $g(x,r) = f(x, 1^{|r|})$, а если вход y функции g не является кодом пары, то $g(y) = y$. Тогда функция g является бесконечно часто слабо односторонней.

Доказательство. Пусть g не является слабо односторонней функцией, в частности g не является слабо односторонней функцией для полинома n. Т.е. найдется полиномиальный по времени алгоритм B, что для всех достаточно больших n выполняется $\Pr_{y \leftarrow U_n}[B(g(y)) \notin g^{-1}(g(y))] < \frac{1}{n}$.

Опишем алгоритм $A(x, \delta)$, который решает задачу $(\widetilde{IF, f(U)})$ в $\widetilde{\text{HeurBPP}}$. Определим $z = (x, 1^{\Delta})$, где $\Delta = 2\lceil \frac{1}{\delta} \rceil n^2$. Пусть $y = B(x)$, если $y = (a, r)$, то выдать a.

$$\Pr_{x \leftarrow U_n}[A(f(x), \delta) \notin f^{-1}(f(x))] = \Pr_{x \leftarrow U_n}[B(f(x), 1^{\Delta}) \notin g^{-1}(f(x), 1^{\Delta})] \leq$$

$$2n^2 \Pr_{y \leftarrow U_{n+\Delta+2\log n+1}}[B(g(y)) \notin g^{-1}(g(y))] < 2n^2 \frac{1}{n + \Delta + 2\log n + 1} \leq \delta.$$

□