Пространственная сложность. Классы $\mathcal{P}$SPACE и $\mathcal{NP}$SPACE. Вероятностные классы сложности: $\mathcal{BPP}, \mathcal{RP}, \mathcal{ZPP}$. Семинары 5-6. 7 и 14 марта 2019 г.

Подготовил: Горбунов Э.

Ключевые слова: классы $\mathcal{P}$SPACE и $\mathcal{NP}$SPACE, теорема Сэвича, классы $\mathcal{P}$SPACE-complete, $\mathcal{P}$SPACE-hard, $\mathcal{BPP}$, $\mathcal{ZPP}$, $\mathcal{RP}$, лемма Шварца-Зиппеля

Литература: [Кормен 1, §6], [Кормен 2, §5 и дополнение C], [Мусатов, Глава 5, §5.1 - 5.3]

Пространственная сложность

На ранних этапах развития компьютеров и вычислительной техники память была существенно ограничена и была чуть ли не самым дорогим элементом компьютера. Поэтому людям при написании программ приходилось существенно задумываться о том, сколько памяти их программа использует. Иными словами, возникла потребность в изучении алгоритмов не только с точки зрения времени их работы, но и с точки зрения расходуемой памяти. Чтобы изучать таким способам алгоритмы, нужна была математическая формализация, о которой мы немного поговорим в этом семинаре.

Во-первых, договоримся, что нас будет интересовать вопрос о памяти, дополнительной к той, что тратится на хранение входа.

Определение. Пусть $s(n)$ — неубывающая функция. Классом $\mathcal{D}$SPACE$(s(n))$ называется класс языков, которые можно распознать на детерминированной машине Тьюринга, которая на входе длины n использует $O(s(n))$ ячеек на рабочих лентах.

Определение. Пусть $s(n)$ — неубывающая функция. Классом $\mathcal{N}$SPACE$(s(n))$ называется класс языков, которые можно распознать на недетерминированной машине Тьюринга, которая на входе длины n использует $O(s(n))$ ячеек на рабочих лентах при любых исходах недетрминированного выбора.

Определение. $\mathcal{P}\text{SPACE} = \bigcup\limits_{k=0}^{\infty} \mathcal{D}\text{SPACE}(n^k)$, $\mathcal{NP}\text{SPACE} = \bigcup\limits_{k=0}^{\infty} \mathcal{N}\text{SPACE}(n^k)$.

Неформально говоря, $\mathcal{P}$SPACE — это класс языков, которые распознаются на полиномиальной памяти детерминированными машинами Тьюринга, а $\mathcal{NP}$SPACE — это класс языков, которые распознаются на полиномиальной памяти недетерминированными машинами Тьюринга. Если говорить ещё более неформально, то $\mathcal{P}$SPACE — это аналог класса $\mathcal{P}$, но относительно используемой памяти, а не времени, а $\mathcal{NP}$SPACE — это, соответственно, аналог класса $\mathcal{NP}$ в теории пространственной сложности.

Как введённые классы связаны с классами, введёнными ранее, и между собой? Во-первых, очевидно, что $\mathcal{P}\text{SPACE} \subseteq \mathcal{NP}\text{SPACE}$. Следующие 2 упражнения поясняют связь между $\mathcal{P}, \mathcal{NP}$ и $\mathcal{P}$SPACE.

Упражнение. Покажите, что $\mathcal{P} \subseteq \mathcal{P}$SPACE.

Решение. Пусть $L \in \mathcal{P}$. Это означает, что существует полиномиально вычислимая МТ M, что $x \in L \iff M(x) = 1$. Заметим, что если МТ работает на входе x время $T_M(x) \leq p(|x|)$, где p — некоторый полином, то за время $T_M(x)$ МТ M успеет побывать не больше, чем в $T_M(x)$ ячейках, а значит, не больше чем $p(|x|)$ ячеек будут задействованы. Следовательно, она использует полиномиальную память.

Упражнение. Покажите, что $\mathcal{NP} \subseteq \mathcal{P}$SPACE.

Решение. Вспомним определение $\mathcal{NP}$: $L \in \mathcal{NP}$ тогда и только тогда, когда существует полиномиально вычислимая МТ $R(x, y)$ и полином $q(x)$, что

$$x \in L \iff \exists y \in \Sigma^* : \ (|y| \leq q(|x|) \text{ и } R(x, y) = 1)\,.$$

Осталось понять, что полиномиальной памяти хватит, чтобы организовать процедуру полного перебора сертификатов. Пусть время работы $R(x, y)$ на входе (x, y) равно $T_R(x, y) \leq p(|x| + |y|)$. Рассмотрим машину

Тьюринга M, которая по входу x будет выделять у себя на рабочей ленте $q(|x|)$ ячеек под перебор сертификатов и $p(|x| + q(|x|))$ ячеек для работы (выделение этой памяти делается за полиномиальное время). В первых $q(|x|)$ ячейках M будет хранить текущий сертификат y (всего возможных сертификатов $1+|\Sigma|+|\Sigma|^2+\ldots+|\Sigma|^{q(|x|)} = \frac{|\Sigma|^{q(|x|)+1}-1}{|\Sigma|-1}$), а следующие $p(|x|+q(|x|))$ ячеек рабочей ленты будут использоваться для запуска машины Тьюринга $R(x, y)$, где y — текущее значение сертификата. Если на данном сертификате получаем ответ 1, то M останавливается и возвращает 1, в противном случае, она затирает всё, что напечатала в ячейках для имитации работы $R(x, y)$, и затем печатает в ячейках для перебора сертификата следующий сертификат. Таким образом, мы используем полиномиальную память и рано или поздно МТ выдаст ответ 1 (если хоть один нужный сертификат существует) либо переберёт все сертификаты и выдаст 0. Следовательно, мы можем на полиномиальной памяти определить, существует ли нужный сертификат, а значит, проверить, что слово принадлежит языку L. Отсюда следует, что $L \in \mathcal{P}$SPACE, а значит, в силу произвольности выбора L из $\mathcal{NP}$, получаем, что $\mathcal{NP} \subseteq \mathcal{P}$SPACE.

Упражнение. Покажите, что $\mathcal{P}$SPACE $\subseteq$ EXPTIME.

Решение. Чтобы это доказать, нужно заметить, что если машина Тьюринга на входе длины n использует не более $p(n)$ (дополнительной) памяти, то количество всевозможных конфигураций (напомню, что конфигурация — это символ, который читает головка на ленте в данный момент, и состояние головки) не превышает $a^{p(n)}p(n)nq$, где a — размер ленточного алфавита, q — мощность множества состояний машины Тьюринга: на рабочей ленте может быть написано $a^{p(n)}$ возможных слов, головка может быть в n позициях на входной ленте, в $p(n)$ позициях на выходной ленте и в q возможных состояниях. Если машина сделает $a^{p(n)}p(n)nq$ шагов и не остановится, то это означает, что в процессе работы некоторая конфигурация повториться, а значит, она будет повторяться бесконечное число раз, то есть МТ никогда не остановится. Следовательно, если МТ останавливается, то останавливается не более, чем за $a^{p(n)}p(n)nq$ шагов, а это как раз экспоненциальное время, так как $a^{p(n)}p(n)nq \leq 2^{p(n)} \cdot 2^{p(n)nq} = 2^{\overbrace{p(n)(1+nq)}^{\text{полином}}}$. Следовательно, если $L \in \mathcal{P}$SPACE, то $L \in$ EXPTIME.

Оказывается, если ещё немного видоизменить доказательство, то можно показать, что $\mathcal{NP}$SPACE $\subseteq$ EXPTIME (детали локазательства можно прочитать в [Мусатов, Глава 5, §5.1 - 5.3]), т.е. верна

Теорема. $P \subseteq \mathcal{NP} \subseteq \mathcal{P}$SPACE $\subseteq \mathcal{NP}$SPACE $\subseteq$ EXPTIME.

Оказывается, верно даже $\mathcal{P}$SPACE $= \mathcal{NP}$SPACE (*следствие теоремы Сэвича*), что нельзя так легко доказать для соответствующих временных классов $\mathcal{P}$ и $\mathcal{NP}$ (напомню вопрос о равенстве классов $\mathcal{P}$ и $\mathcal{NP}$ до сих пор не решён). Более того, ничего неизвестно про равенство $\mathcal{NP}$ и $\mathcal{P}$SPACE.

Теорема. (Сэвич). Если $s(n) \geq \log n$, то $\mathcal{N}$SPACE$(s(n)) \subseteq \mathcal{D}$SPACE$(s(n)^2)$.

На доказательстве теоремы мы останавливаться не будем, т.к. при детальном рассмотрении она требует отдельной лекции (доказательство можно прочитать, например, в [Мусатов, Глава 5, §5.1 - 5.3]). Отметим только, что, т.к. квадрат полинома — это тоже полином, из теоремы Сэвича вытекает

Следствие. $\mathcal{P}$SPACE $= \mathcal{NP}$SPACE.

Классы $\mathcal{P}$SPACE-hard и $\mathcal{P}$SPACE-complete. Примеры $\mathcal{P}$SPACE-полных языков.

Классы $\mathcal{P}$SPACE-hard и $\mathcal{P}$SPACE-complete определяются точно так же, как $\mathcal{NP}$-hard и $\mathcal{NP}$-complete.

Определение. $L \in \mathcal{P}$SPACE-hard $\iff \forall A \in \mathcal{P}$SPACE $\hookrightarrow A \leq_P L$, где $\leq_P$ — сводимость по Карпу, или просто полиномиальная сводимость.

Определение. $L \in \mathcal{P}$SPACE-complete $\iff \forall A \in \mathcal{P}$SPACE $\hookrightarrow A \leq_P L$ и $L \in \mathcal{P}$SPACE.

Рассмотрим несколько примером $\mathcal{P}$SPACE-полных языков.

Определение. Язык SPACETMSAT есть множество $\{(M, x, 1^s) \mid M(x) = 1$ и $M(x)$ занимает не больше s ячеек памяти$\}$.

Теорема. SPACETMSAT$\in \mathcal{P}$SPACE-complete.

Доказательство. Первый этап состоит в доказательстве SPACETMSAT$\in \mathcal{P}$SPACE: запустим $M(x)$, ограничив зону работы величиной s и контролируя, что машина не зациклилась. Если машина остановилась и выдала 1, то выдаём 1. Если МТ выдала 0, попыталась выйти за пределы зоны или зациклилась, то выдаём 0. Постро-

енный алгоритм использует $O(s)$ памяти, то есть полиномиален по памяти. Второй этап состоит в построении сведения произвольного языка $L \in \mathcal{P}$SPACE к SPACETMSAT. Если $L \in \mathcal{P}$SPACE, то существует МТ M, которая на входе x занимает память не больше $p(|x|)$. Рассмотрим сводящую функцию: $f(x) = (M, x, 1^{p(|x|)})$. По определению SPACETMSAT получаем, что $x \in L \iff f(x) \in$ SPACETMSAT, т.е. $L \leq_P$ SPACETMSAT, ибо сводящая функция полиномиально вычислима по времени.

Определение. Языком TQBF ("totally quantified boolean formulae", или булевы формулы с кванторами, БФК) называется множество булевых формул φ, таких что для некоторого $x_1 \in \{0,1\}$ найдётся $x_2 \in \{0,1\}$, такое что для некоторого $x_3 \in \{0,1\}$ найдётся ... (цепочка чередующихся кванторов по всем переменным) $\varphi(x_1, x_2, \ldots) = 1$.

Теорема. TQBF $\in \mathcal{P}$SPACE-complete.

Доказательство этой теоремы можно прочитать в [Мусатов, Глава 5, §5.1 - 5.3].

Вероятностные классы сложности

В этом разделе мы продолжим обсуждение вероятностных процедур, начатое ещё на второй неделе, и дадим их формальное описание. Из-за недостатка времени мы успеем только ознакомиться с определениями и решить несколько задач. Но хочется подчеркнуть, что вероятностный подход к алгоритмам сейчас является одним из основных во многих исследованиях, поэтому заметная часть студентов в том или ином виде будет встречаться с вероятностными алгоритмами в своей жизни. Всем заинтересованным лицам будет полезно продолжить изучение этого подхода самостоятельно.

Итак, с мотивацией разобрались, теперь к делу.

Определение. *Вероятностной машиной Тьюринга (ВМТ)* мы будем называть детерминированную машину Тьюринга M, которая принимает на вход 2 аргумента: x (вход, в классическом понимании) и r (случайные биты), где $|r|$ — некоторая функция от $|x|$. Результат работы $M(x, r)$ — вероятностное распределение, индуцированное конкретным выбором x и равномерным распределением на всех значениях r. Временем работы машины M на данном x мы будем считать максимальное время работы $M(x, r)$ для всех r указанной длины. Так же определяется и использованная память.

Формальное определение выше можно интерпретировать следующим образом. Пусть кто-то сгенерировал для машины Тьюринга M случайную последовательность битов (например, бросанием симметричной монетки) такой длины, чтобы нам хватило для наших вычислений. Далее МТ M работает на входе x, используя сгенерированные биты *детерминированно*, то есть вся случайность содержится в сгенерированной последовательности r. А можно считать, что в некоторых состояниях МТ подбрасывается симметричная монетка (возможно не один раз) для того, чтобы выбрать переход. Обе интерпретации эквивалентны друг другу и формальному определению.

Если сравнивать ВМТ с недетерминированной машиной Тьюринга (НМТ), то отличие состоит в том, что НМТ «перебирает параллельно» все возможные варианты, а ВМТ выбирает один из них случайно (можно говорить, что кто-то заранее для неё делает этот случайный выбор) и работает согласно этому выбору.

Рассмотрим некоторые классы языков, распознаваемых ВМТ. Отметим, что использование случайности в алгоритме может приводить к ошибкам. Ошибки распознавания языка L бывают двух родов: первого, когда $x \notin L$, но алгоритм выдал единицу, и второго, когда $x \in L$, но алгоритм выдал ноль. Классы, которые мы будем рассматривать, разделяют языки по вероятностям ошибок первого и второго рода.

Определение. Классом $\mathcal{BPP}$ ("bounded-error probabilistic polynomial") называется класс таких языков L, что существует полиномиальный в среднем вероятностный алгоритм (ВМТ) V (напомним, что это означает, что $\exists p(x)$ — полином, такой что $\mathbb{E}_r[T_V(x, r)] \leq p(|x|)$, где $T_V(x, r)$ — время работы V на входе x со случайными битами r), что вероятности ошибок первого и второго рода не больше $\frac{1}{3}$:

1) если $x \in L$, то $\mathbb{P}_r\{V(x, r) = 1\} \geqslant \frac{2}{3}$;
2) если $x \notin L$, то $\mathbb{P}_r\{V(x, r) = 1\} \leqslant \frac{1}{3}$.

Такого рода вероятностные алгоритмы (работают за полиномиальное в среднем время, но могут ошибаться) называют алгоритмами *стандарта Монте-Карло*. Но почему в определении $\mathcal{BPP}$ выбраны константы $\frac{2}{3}$ и $\frac{1}{3}$?

Упражнение. Для каждого $\varepsilon \in \left(0, \frac{1}{2}\right)$ рассмотрим класс $\mathcal{BPP}_\varepsilon$, который определяется точно так же как $\mathcal{BPP}$, если в определении в п. 1) заменить $\frac{2}{3}$ на $1-\varepsilon$, а в п. 2) заменить $\frac{1}{3}$ на ε. Получим класс языков, для которых существует полиномиальная в среднем ВМТ, которая имеет вероятность ошибок первого и второго рода не больше ε. Доказать, что $\mathcal{BPP}_\varepsilon = \mathcal{BPP}$. Доказать, что в определении класса $\mathcal{BPP}$ слова «полиномиальный в среднем вероятностный алгоритм» можно заменить на «полиномиальный в худшем случае вероятностный алгоритм».

Решение.

1. Чтобы доказать равенство $\mathcal{BPP}_\varepsilon = \mathcal{BPP}$ нужно показать два включения: $\mathcal{BPP}_\varepsilon \subseteq \mathcal{BPP}$ и $\mathcal{BPP}_\varepsilon \supseteq \mathcal{BPP}$. Не умаляя общности, будем считать, что $\varepsilon < \frac{1}{3}$ (как будет видно из доказательства, можно точно так же доказать $\mathcal{BPP}_{\varepsilon_1} = \mathcal{BPP}_{\varepsilon_2}$ для любых $\varepsilon_1, \varepsilon_2 \in \left(0, \frac{1}{2}\right)$).В таком случае включение $\mathcal{BPP}_\varepsilon \subseteq \mathcal{BPP}$ очевидно. Докажем $\mathcal{BPP}_\varepsilon \supseteq \mathcal{BPP}$. Для этого по вероятностному алгоритму V, который имеет ошибки первого и второго рода, ограниченные по вероятности константой $\frac{1}{3}$, построим вероятностный алгоритм V', у которого вероятности ошибок первого и второго рода не превосходят ε. Пусть k — это такое целое число, что $\left(\frac{1}{3}\right)^k \leq \varepsilon$. Приём, который мы сейчас опишем, называется *амплификацией*. Вероятностный алгоритм V' на входе x независимо сгенерирует k случайных последовательностей $r_1, \ldots, r_k$ и вызовет алгоритм V на входах $(x, r_1), \ldots, (x, r_k)$. Пусть $p = \mathbb{P}_r\{V(x,r) = 1\}$ для слова $x \in L$. По определению $\mathcal{BPP}$ вероятность $p \geqslant \frac{2}{3}$. Если хотя бы на $pk - \delta$ (нужно взять достаточно маленькое δ) входов V выдаст 1, то V' вернёт 1; в противном случае V' возвращает 0. Такое правило следует из Закона Больших Чисел и неравенства Хёффдинга. Первое означает, что если долго бросать монетку, то вероятность, что доля выпадения орла отклонится от $\frac{1}{2}$ на некоторое τ, стремится к нулю. Неравенство Хёффдинга (см. Следствие 2 из конспекта семинара по теории вероятностей) в нашем случае означает, что

$$\mathbb{P}_{r_1,\ldots,r_k}\left\{\left|\frac{\xi_1 + \ldots + \xi_k}{k} - p\right| \geqslant t\right\} \leqslant 2e^{-2kt^2},$$

где $\xi_i = V(x, r_i)$. Неформально говоря, среднее арифметическое с ростом числа слагаемых близко с математическому ожиданию с большой вероятностью и неравенство Хёффдинга оценивает эту вероятность. Более строго с этим куском теории вероятностей вы познакомитесь в следующем году, а пока просто примем на веру этот факт. Тогда, если правильно подобрать константы δ и t, то можно добиться, что

$$\mathbb{P}_{r_1,\ldots,r_k}\{V'(x, r_1, \ldots, r_k) = 1\} = \mathbb{P}_{r_1,\ldots,r_k}\left\{\frac{\xi_1 + \ldots + \xi_k}{k} \geqslant p - \frac{\delta}{k}\right\} \geqslant 1 - \varepsilon.$$

В домашнем задании вам предлагается аккуратно оценить эти константы. Аналогично показывается, что если $x \notin L$, то $\mathbb{P}_{r_1,\ldots,r_k}\{V'(x, r_1, \ldots, r_k) = 1\} \leqslant \varepsilon$ (нужно заметить, что в таком случае среднее арифметическое ξ_i уже будет близко к число $q \leq \frac{1}{3}$, поэтому при правильном выборе констант можно добиться, что среднее будет с большой вероятностью меньше чем $\frac{2}{3}$, а значит, и чем p).

2. Теперь докажем, что полиномиальная в среднем сходимость равносильно полиномиальной в худшем случае сходимости в классе $\mathcal{BPP}$. Во-первых, если алгоритм работает полиномиально в худшем случае, то он будет работать полиномиально и в среднем, поэтому такой заменой мы не сузим класс рассматриваемых языков. Осталось понять, что мы этот класс и не расширим. Пусть вероятностный алгоритм V на входе (x, r) работает время $T_V(x, r)$ причём $\mathbb{E}_r[T_V(x, r)] \leqslant p(|x|)$, где p — некоторый полином. Рассмотрим новый вероятностный алгоритм V', который на входе (x, r) будет делать не более $9p(|x|)$ шагов алгоритма V на входе (x, r). Если раньше, чем через $9p(|x|)$ шагов, будет получен результат, то выдаём его. В противном случае выдаём ответ 0. Заметим, что из неравенства Маркова (см. здесь и тут)

$$\mathbb{P}_r\{T_V(x, r) \geqslant 9p(|x|)\} \leqslant \frac{\mathbb{E}_r[T_V(x, r)]}{9p(|x|)} \leqslant \frac{1}{9}.$$

Это означает, что если $x \in L$, то

$$\mathbb{P}_r\{V'(x, r) = 1\} = \mathbb{P}_r\{V(x, r) = 1 \text{ и } T_V(x, r) < 9p(|x|)\} \geqslant \mathbb{P}_r\{V(x, r) = 1\} - \mathbb{P}_r\{T_V(x, r) \geqslant 9p(|x|)\}$$
$$\geqslant \frac{2}{3} - \frac{1}{9} = \frac{5}{9} > \frac{1}{2},$$

а если $x \notin L$, то

$$\mathbb{P}_r\{V'(x,r)=1\} = \mathbb{P}_r\{V(x,r)=1 \text{ или } T_V(x,r) \geqslant 9p(|x|)\} \leqslant \mathbb{P}_r\{V(x,r)=1\} + \mathbb{P}_r\{T_V(x,r) \geqslant 9p(|x|)\} \leqslant \frac{1}{3} + \frac{1}{9} = \frac{4}{9} < \frac{1}{2},$$

то есть $L \in \mathcal{BPP}_{\frac{4}{9}} = \mathcal{BPP}$.

Определение. Классом $\mathcal{RP}$ ("randomized polynomial") называется класс таких языков L, что существует полиномиальный в худшем случае вероятностный алгоритм (ВМТ) V (напомним, что это означает, что $\exists p(x)$ — полином, такой что для всех r выполнено $T_V(x,r) \leq p(|x|)$, где $T_V(x,r)$ — время работы V на входе x со случайными битами r), что вероятность ошибки первого рода равна 0, а вероятность ошибки второго рода не больше $\frac{1}{2}$:

1) если $x \in L$, то $\mathbb{P}_r\{V(x,r)=1\} \geqslant \frac{1}{2}$;
2) если $x \notin L$, то $\mathbb{P}_r\{V(x,r)=1\} = 0$.

Определение. Классом со-$\mathcal{RP}$ называется класс таких языков L, что существует полиномиальный в худшем случае вероятностный алгоритм (ВМТ) V, что вероятность ошибки первого рода не больше $\frac{1}{2}$, а вероятность ошибки второго рода равна 0:

1) если $x \in L$, то $\mathbb{P}_r\{V(x,r)=1\} = 1$;
2) если $x \notin L$, то $\mathbb{P}_r\{V(x,r)=1\} \leqslant \frac{1}{2}$.

Легко доказать, что если $L \in \mathcal{RP}$, то $\overline{L} \in$ со-$\mathcal{RP}$ и наоборот (достаточно взять тот же вероятностный алгоритм, но поменять результат на противоположный), что и оправдывает название со-$\mathcal{RP}$. Если мы хотим определить класс полиномиально разрешимых на ВМТ языков с нулевой вероятностью ошибки (*стандарт Лас-Вегас*), то существенно потребовать полиномиальность в среднем: если мы потребуем полиномиальность в худшем случае, то получим, что такой язык просто полиномиально разрешим: достаточно как-то задать вероятностные биты и запустить ВМТ.

Определение. Классом $\mathcal{ZPP}$ ("zero-error probabilistic polynomial") называется класс таких языков L, что существует полиномиальный в среднем вероятностный алгоритм (ВМТ) V, такой что $x \in L$ для всех r эквивалентно $V(x,r)=1$.

Иными словами, в $\mathcal{ZPP}$ лежат те языки, которые можно разрешить в среднем за полиномиальное время, но не обязательно вероятностный алгоритм будет разрешать язык за полиномиальное время для всех последовательностей случайных битов. Рассмотрим следующий пример.

Упражнение. Пусть имеется генератор случайных чисел, который выдаёт 1 или 0 с одинаковой вероятностью равной $\frac{1}{2}$. Придумайте алгоритм, который будет выдавать 0 с вероятностью $\frac{1}{3}$ и 1 с вероятностью $\frac{2}{3}$, используя только заданный генератор случайных чисел. Оцените время работы в среднем и худшем случае.

Решение. Хоть этот пример и не про разрешение языков, но он иллюстрирует тот факт, что в среднем алгоритм может работать за $O(1)$, но в худшем случае не останавливаться (это мы уже видели на втором семинаре, когда разбирали алгоритм 1). Рассмотрим следующий алгоритм:

1. Сгенерируем 2 бита, используя наш генератор.
2. Если получили 00, то выдаём ответ 0. Если получили 01 или 10, то выдаём ответ 1.
3. Если получили 11, то возвращаемся на шаг 1.

Тогда $\mathbb{P}\{\text{получен ответ 0}\} = \frac{1}{4} + \frac{1}{4^2} + \ldots = \frac{1}{4}\sum\limits_{k=0}^{\infty}\frac{1}{4^k} = \frac{1}{3}$, а $\mathbb{P}\{\text{получен ответ 1}\} = \frac{1}{2} + \frac{1}{2}\cdot\frac{1}{4} + \frac{1}{2}\cdot\frac{1}{4^2} + \ldots = \frac{1}{2}\sum\limits_{k=0}^{\infty}\frac{1}{4^k} = \frac{2}{3}$. Среднее время работы алгоритма равно $1\cdot\frac{3}{4} + 2\cdot\frac{3}{4}\cdot\frac{1}{4} + 3\cdot\frac{3}{4}\cdot\frac{1}{4^2} + \ldots = \frac{3}{4}\sum\limits_{k=0}^{\infty}(k+1)\frac{1}{4^k} = \frac{3}{4}\cdot\frac{1}{\left(1-\frac{1}{4}\right)^2} = \frac{4}{3} =$

$O(1)$. Время работы в худшем случае равно бесконечности, т.к. для любого k вероятность того, что алгоритм проработает хотя бы k итераций равна $\frac{1}{4^{k-1}} > 0$.

Немного обсудим, как введённые классы соотносятся друг с другом. Верна следующая цепочка из вложений:

$$\mathcal{P} \subseteq \mathcal{ZPP} \subseteq \mathcal{RP} \subseteq \mathcal{BPP}.$$

Первое вложение следует из того, что полиномиально разрешимый язык можно разрешить при помощи ВМТ, которая полностью игнорирует случайные биты и работает на входе x точно так же, как полиномиальная МТ, разрешающая данный язык. Второе вложение можно показать при помощи неравенства Маркова аналогично тому, как мы показывали, что для $\mathcal{BPP}$ можно полиномиальную разрешимость в среднем заменить на полиномиальную разрешимость в худшем случае. Третье вложение тоже очевидно следует из определений $\mathcal{RP}$, $\mathcal{BPP}$ и доказанного нами упражнения про эквивалентность определений $\mathcal{BPP}$.

Лемма Шварца-Зиппеля

Лемма Шварца-Зиппеля. Пусть $f(x_1, \ldots, x_n)$ — не равный тождественно нулю многочлен от n переменных степени не выше k. Пусть случайные величины $\xi_1, \ldots, \xi_n$ независимо и равномерно распределены на $\{0, 1, 2, \ldots, N-1\}$. Тогда $\mathbb{P}\{f(\xi_1, \ldots, \xi_n) = 0\} \leqslant \frac{kn}{N}$.

Доказательство. Докажем это утверждение индукцией по числу переменных n. Для $n = 1$ утверждение очевидно, поскольку многочлен степени k имеет не более k вещественных корней, а значит, среди чисел $\{0, 1, \ldots, N-1\}$ не более k таких, в которых f обращается в ноль, то есть их доля $\leqslant \frac{k}{N}$. При $n > 1$ разложим $f(x_1, \ldots, x_n)x_1$ по переменной x_1: $f(x_1, \ldots, x_n) = f_0(x_2, \ldots, x_n) + f_1(x_2, \ldots, x_n)x_1 + \ldots + f_t(x_2, \ldots, x_n)x_1^t$, причём $f_t(x_2, \ldots, x_n)x_1^t$ не равен тождественно нулю. Тогда по формуле полной вероятности

$$\begin{aligned}
\mathbb{P}\{f(\xi_1, \ldots, \xi_n) = 0\} &= \mathbb{P}\{f(\xi_1, \ldots, \xi_n) = 0 \mid f_t(\xi_2, \ldots, \xi_n) \neq 0\} \underbrace{\mathbb{P}\{f_t(\xi_2, \ldots, \xi_n) \neq 0\}}_{\leqslant 1} \\
&\quad + \underbrace{\mathbb{P}\{f(\xi_1, \ldots, \xi_n) = 0 \mid f_t(\xi_2, \ldots, \xi_n) = 0\}}_{\leqslant 1} \mathbb{P}\{f_t(\xi_2, \ldots, \xi_n) = 0\} \\
&\leqslant \mathbb{P}\{f(\xi_1, \ldots, \xi_n) = 0 \mid f_t(\xi_2, \ldots, \xi_n) \neq 0\} + \mathbb{P}\{f_t(\xi_2, \ldots, \xi_n) = 0\}.
\end{aligned}$$

Второе слагаемое в последней строчке предыдущего неравенства оцениваем по предположению индукции сверху через $\frac{k(n-1)}{N}$, а первое слагаемое не больше $\frac{t}{n}$, т.к. в каждом рассматриваемом элементарном исходе условного вероятностного пространства f будет нетривиальным многочленом от x_1, поэтому можно неравенство для случая $n = 1$.

Пример. При помощи доказанной леммы можно вероятностно проверять, равны ли 2 многочлена $f(x_1, \ldots, x_n)$ и $g(x_1, \ldots, x_n)$ как функции. Действительно, это эквивалентно тому, что многочлен $(x_1, \ldots, x_n) = f(x_1, \ldots, x_n) - g(x_1, \ldots, x_n)$ является тождественно нулевым. Рассмотрим следующую вероятностную процедуру решения этой задачи: сгенерировать случайный набор $x_1, \ldots, x_n \in \{0, 1, 2, \ldots, N\}$ для $N \geqslant 2kn$, где k — степень многочлена $p(x_1, \ldots, x_n)$, и подставить полученный набор в многочлен p. Если получим ответ 0, то говорим, что многочлен тождественный ноль (выдаём 1), иначе — выдаём 0. Заметим, что мы тем самым доказали, что язык многочленов, являющихся тождественными нулями, лежит в со-$\mathcal{RP}$: действительно, если p — тождественный ноль, то $\mathbb{P}\{\text{наш алгоритм выдаст } 1\} = 1$, а если p не является тождественным нулём, то вероятность ошибки $\mathbb{P}\{\text{наш алгоритм выдаст } 1\} \leqslant \frac{kn}{N} \leqslant \frac{1}{2}$.

В задании вас ждут ещё 2 интересных примера того, как использовать лемму Шварца-Зиппеля.