Лекция 6

Вероятностные алгоритмы. Булевы схемы

(Конспект: С. Федин)

6.1 Булевы схемы

Определение 6.1. Булевой схемой C называется ориентированный граф без (ориентированных) циклов, вершины которого помечены входными битами $x_1, x_2, \ldots, x_m$ и булевыми операциями $\vee$, $\wedge$, $\neg$; одна выделенная вершина является выходом. Вершины этого графа называются гейтами, во входные гейты не ведет ни одно ребро, из выходного гейта не исходит ни одного ребра. Входная степень гейта должна совпадать с количеством аргументов той операции, которой она помечена (так, например, в гейт помеченный $\neg$ должно входить ровно одно ребро, а в гейт, помеченный $\vee$ или $\wedge$, должно входить ровно два ребра).

Вычисления в булевой схеме происходят следующим образом: на входные гейты подается набор $X \in \{0,1\}^m$ значений переменных x_i $(i \in 1..m)$; значение в каждом следующем гейте определяется через значения в гейтах, из которых выходят ребра в данный гейт: например, если в гейт, помеченный $\vee$, входят ребра из гейтов, значения в которых равны y_1 и y_2, то значение в этом гейте равно $y_1 \vee y_2$. Результатом вычисления схемы является значение, вычисленное в выходном гейте.

Определим теперь класс **P/poly**, состоящий из булевых схем полиномиального размера.

Определение 6.2. $L \in \mathbf{P/poly}$ если существует семейство схем $\{C_i\}_{i\in\mathbb{N}}$ и полином p, такие что $\forall i\ |C_i| \le p(i)$ и $x \in L \Leftrightarrow C_{|x|}(x) = 1$.

Альтернативное определение:

Определение 6.3. Класс **P/poly** состоит из тех языков L, принимамых недетерминированной машиной Тьюринга, подсказка которой не зависит от входа (но может зависеть от его длины), т.е. имеется полиномиально вычислимое отношение R, для которого

$$\exists\{y_i\}_{i\in\mathbb{N}}\ \forall x\ (x \in L \Leftrightarrow R(x, y_{|x|}) = 1).$$

Нетрудно видеть, что эти определения эквивалентны.

Хотелось бы понять взаимосвязь классов **P** и **NP** с новым классом **P/poly**. Очевидно, что $\mathbf{P} \subseteq \mathbf{P/poly}$. Легко увидеть, что $\mathbf{P} \neq \mathbf{P/poly}$ (ведь **P/poly** содержит даже некоторые языки, проблема принадлежности которым алгоритмически неразрешима). Вопрос $\mathbf{NP} \overset{?}{\subseteq} \mathbf{P/poly}$ открыт. Однако он не является совершенно независимым вопросом.

Теорема 6.1. $\mathbf{NP} \subseteq \mathbf{P/poly} \Rightarrow \mathbf{PH} = \Sigma^2\mathbf{P}$.

Доказательство[1]. Рассмотрим $\Sigma^3\mathbf{P}$-полный язык

$$L = \{F \text{ — формула в КНФ} \mid \exists x \forall y \exists z F(x, y, z)\}. \tag{6.1}$$

Достаточно показать, что он лежит в $\Sigma^2\mathbf{P}$.

Действительно, пусть $\mathbf{NP} \subseteq \mathbf{P/poly}$. Тогда для `SAT` имеются схемы полиномиального размера. При помощи их мы и справимся с формулой F. Правда, они нам не даны; поэтому мы изобретем их при помощи квантора $\exists$ и проверим их корректность при помощи квантора $\forall$. Корректность семейства схем для `SAT` можно проверить (у нас есть квантор всеобщности!), зная, что для любой формулы G и переменной k выполняется $G = G[k := 0] \vee G[k := 1]$ — две последние формулы — меньше, и их можно проверить схемой меньшего размера (и т д. до тривиальных). Более формально, семейство схем $\{C_i\}_{i=1}^n$ корректно для формулы w длины n, если для каждой переменной k выполняется $C_{|w|}(w) = C_{|w[k:=0]|}(w[k := 0]) \vee C_{|w[k:=1]|}(w[k := 1])$, а если w вовсе не содержит переменных, то $C_{|w|}(w)$ выдает соответствующую константу (`true` или `false`).

Итак, следующее выражение доказывает, что $L \in \Sigma^2\mathbf{P}$:

$\exists$ схемы $C_1, \ldots, C_n$ для входов размера $1, \ldots, n$ соответственно;
$\exists x$ (это строка как в (6.1));

такие, что

$\forall y$ (это строка как в (6.1));

$\forall w$ — булевой формулы длины $\leq n$;

$$\text{семейство } \{C_i\} \text{ корректно для выражения } w \text{ и}$$
$$C_{|F|}(F(x, y, z)) = 1.$$

(В последнем выражении свободные переменные — только z!)

□

Эту теорему можно переформулировать следующим образом:

Теорема 6.2. $\mathbf{NP} \subseteq \mathbf{P/poly} \Leftrightarrow \exists$ *редкое* **NP**-*трудное множество в смысле сводимости по Куку.*

Доказательство. Достаточно доказать, что SAT $\in$ **P/poly** $\Leftrightarrow$ SAT сводится по Куку к редкому множеству.

Необходимость. Пусть SAT $\in$ **P/poly**. Зафиксируем семейство схем $\{C_i\}_{i\in\mathbb{N}}$ полиномиального размера (можно считать, что размер схемы C_i — *ровно* $p(i) = dc^i$) для языка SAT. Построим теперь требуемое редкое множество L. Закодируем все схемы, и будем составлять множество L из кодов всех схем и всех их префиксов т.е. для схемы C_n с кодом $a_1a_2\ldots a_{p(n)}$ (где $a_i \in \{0,1\}$) в множество L попадут: $a_1a_2\ldots a_{p(n)}$, $a_1a_2\ldots a_{p(n)-1}$#, $a_1a_2\ldots a_{p(n)-2}$##, ..., $a_1\#^{p(n)-1}$. Понятно, что построенное множество редко (для данной длины строки $p(n)$ в множество L попадает одна схема и $p(n)-1$ ее префиксов). Теперь, используя L как оракул, можно найти схему нужного размера: спросим его $0\#^{p(|x|)-1} \stackrel{?}{\in} L$, если он ответит «да», то спросим $00\#^{p(|x|)-2}$; если «нет», то $10\#^{p(|x|)-1}$ и т. д. Итак, задав $p(|x|)$ запросов оракулу, мы построим схему $C_{|x|}$. Остается промоделировать ее вычисления на входе x на детерминированной машине Тьюринга.

Достаточность. Пусть у нас имеется **NP**-трудное редкое множество L, к которому сводится SAT. Докажем, что SAT $\in$ **P/poly**, пользуясь определением 6.3. В качестве подсказки y_i возьмем все строчки из L размера i (объединенные в одну: $l_1\#l_2\#l_3\ldots$. Поскольку L — редкое множество, суммарная длина такой подсказки y_i ограничена некоторым полиномом. □

6.2 Вероятностные алгоритмы

Определение 6.4. Язык L принадлежит классу **RP**, если существует вероятностный полиномиальный по времени алгоритм, который для $x \in L$ принимает с вероятностью $\geq \frac{3}{4}$, а для $x \notin L$ отвергает с вероятностью 1.

Замечание 6.1. Класс **RP** не изменится, если в определении заменить константу $\frac{3}{4}$ на любую другую, не равную 0 и 1.

Замечание 6.2. Можно сделать, чтобы такой алгоритм «врал» со сколь угодно маленькой вероятностью. Действительно, повторив вычисления k раз и ответив «да» если хотя бы один раз вычисление было принимающим, мы получим вероятность ошибки не более $\frac{1}{4^k}$.

Определение 6.5. Язык L принадлежит классу **BPP**, если существует вероятностный алгоритм, который для $x \in L$ принимает с вероятностью $\geq \frac{3}{4}$, а для $x \notin L$ отвергает с вероятностью $\geq \frac{3}{4}$. Другими словами, существует алгоритм, выдающий правильный ответ с вероятностью $\geq \frac{3}{4}$.

Замечание 6.3. Класс **BPP** не изменится, если в определении заменить константу $\frac{3}{4}$ на любую другую *константу*, строго большую $\frac{1}{2}$.

Замечание 6.4. Снова можно сделать так, чтобы вероятность ошибки была экспоненциально мала. Повторим вычисления k раз и окончательный результат будем давать мнением большинства. Для доказательства того, что в итоге вероятность ошибки мала, воспользуемся неравенством Чернова.

Факт 6.1.

$$P\{X > (1+\varepsilon)pk\} < \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{1+\varepsilon}}\right)^{pk} \leq e^{-\frac{pk\varepsilon^2}{4}},$$

где $X = \sum_{i=1}^{k} x_i$, *а* x_i *— независимые случайные величины, которые равны 1 и 0 с вероятностями* p *и* $(1-p)$ *соответственно.*

Правая часть этого неравенства — это вероятность того, что «многие» $x_i = 1$. В нашем случае $x_k = 1$ соответствует неправильному ответу в k-ом вычислении; вероятность ошибки $p = \frac{1}{4}$. Возьмем $\epsilon = 1$. Тогда, неравенство имеет вид: $P\{X > \frac{1}{2}k\} \leq e^{-\frac{k}{16}} = 2^{-\Omega(k)}$. Итак, для того, чтобы получить вероятность ошибки $\frac{1}{2^m}$, достаточно повторить вычисления $k = O(m)$ раз.

Определение 6.6. Класс задач, для которых есть алгоритмы с нулевой вероятностью ошибки: $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{co\text{-}RP}$.

Упражнение 6.1. Для определения $x \in L$ для $L \in$ **ZPP** будем запускать соответствующие алгоритмы из **RP** и **co-RP** параллельно, пока хотя бы один из них не даст тот ответ, в котором он заведомо не «врет». Докажите, что математическое ожидание времени, которое пройдет до этого момента, полиномиально. □

Классы, определенные выше, можно определить по-другому, с помощью отношений:

Определение 6.7. $L \in \mathbf{RP}$, если имеется полиномиально проверяемое отношение R, ограниченное полиномом p, такое, что если $x \in L$, то для по крайней мере $\frac{3}{4}$ всех возможных подсказок y длины $p(|x|)$ выполнено $R(x, y) = 1$; если $x \notin L$, то для любой подсказки — $R(x, y) = 0$.

Определение 6.8. $L \in \mathbf{BPP}$, если имеется полиномиально проверяемое отношение R, ограниченное полиномом p, такое, что если $x \in L$, то для по крайней мере $\frac{3}{4}$ всех возможных подсказок y длины $p(|x|)$ выполнено $R(x, y) = 1$; если $x \notin L$, то для по крайней мере $\frac{3}{4}$ всех возможных подсказок y длины $p(|x|)$ выполнено $R(x, y) = 0$.

Замечание 6.5. Сравним классы **RP** и **BPP** с **NP**. Для **NP** надо было, чтобы хотя бы в одном листе дерева вычислений машины Тьюринга была 1 для слова из языка, и все 0 — для слова не из языка. Для **RP** надо, чтобы доля единиц была хотя бы $\frac{3}{4}$ для слова из языка, и все нули — для слова не из языка. Для **BPP** — доля $\frac{3}{4}$ единиц и доля $\frac{3}{4}$ нулей для слова из языка и не из языка соответственно.

Определение 6.9. Замечание 6.5 дает возможность определить **RP** и **BPP** при помощи недетерминированных машин Тьюринга (полиномиальных по времени).

Определим еще один класс, **BPP/poly**: класс схем, которым разрешено пользоваться случайными числами.

Определение 6.10. $L \in \mathbf{BPP/poly}$, если существует семейство схем $\{C_i\}_{i \in N}$ и полином p, такой, что для $\forall i\ |C_i| \le p(i)$ и $x \in L$ тогда и только тогда, когда для хотя бы $\frac{3}{4}$ случайных строчек y выполнено $C_{|x|}(x, y) = 1$; $x \notin L$ тогда и только тогда, когда для хотя бы $\frac{3}{4}$ случайных строчек y выполнено $C_{|x|}(x, y) = 0$.

Замечание 6.6. Повторяя вычисления на схеме «много» раз, можно добиться, чтобы вероятность ошибки была экспоненциально мала.

Верно следующее утверждение:

Теорема 6.3. $\mathbf{BPP} \subseteq \mathbf{BPP/poly} = \mathbf{P/poly}$.

Доказательство. Достаточно доказать $\mathbf{BPP/poly} \subseteq \mathbf{P/poly}$. Пусть нам дано семейство схем из **BPP/poly**; потребуем, чтобы вероятность ошибки была не более $1 - \frac{1}{2^{2n}}$. Заметим, что для конкретного входа x длины n

«хороших» случайных строчек y (т.е. таких строчек, что если они попались на вход, то схема выдаст правильный ответ) имеется не менее $1-\frac{1}{2^{2n}}$ от всех возможных y (длины, используемой схемой для входа x длины n). Покажем, что существует строчка y, для которой схема дает правильный ответ для всех входов x данной длины (она и будет подсказкой для этой длины).

Рассмотрим некую случайную строчку. Заметим, что для конкретного x длины n эта строчка «плоха» с вероятностью $\frac{1}{2^{2n}}$. Значит, вероятность, что она будет «плоха» хотя бы для одного входа длины n, составляет не более $2^n\frac{1}{2^{2n}} = \frac{1}{2^n} < 1$. Таким образом, вероятность того, что эта строчка «хороша» для всех входов данной длины — строго положительна. Следовательно, такая строчка существует! $\square$

Теорема 6.4. $\mathbf{BPP} \subseteq \Sigma^2\mathbf{P}$.

Доказательство. Пусть $L \in \mathbf{BPP}$. Будем считать, что соответствующий алгоритм $A \in \mathbf{BPP}$ имеет вероятность ошибки $\frac{1}{2^n}$. Обозначим через A_x множество «хороших» случайных строчек для входа $x \in L$ (т.е. тех, что приводят к принятию x алгоритмом A), а через U_n — множество всех случайных строчек для входов длины n. Покажем, что полиномиальным количеством k копий A_x мы можем покрыть U_n, а именно, докажем, что что

$$\exists\{t_i\}_{i=1}^{k}\ \forall r \in U_n\ \bigvee_{i=1}^{k}(r \in A_x \oplus t_i). \tag{6.2}$$

(Заметим, что для $x \notin L$ так покрыть U_n нельзя из мощностных соображений.) Этого достаточно, так как мы доказывали, что именно в такой форме представляется задача из $\Sigma^2\mathbf{P}$; чтобы убедиться, что внутреннюю дизъюнкцию можно проверить за полиномиальное время, заметим, что в ней лишь полиномиальное количество формул, а проверка $r \in A_x \oplus t_i$ может быть произведена за полиномиальное время (она эквивалентна $r \oplus t_i \in A_x$ — просто дадим нашему алгоритму из $\mathbf{BPP}$ строчку $r \oplus t_i$ случайных чисел). Если высказывание 6.2 неверно, очевидно, что $x \notin L$.

Ясно, что для доказательства того, что $x \in L \Rightarrow$ (6.2) достаточно показать, что для k строчек t_i, взятых из U_n случайно и независимо друг от друга, справедливо

$$P\{\neg(\forall r \in U\ \bigvee_{i=1}^{k}(r \in A_x \oplus t_i))\} < 1.$$

Докажем это:

$$P\{\neg(\forall r \in U \ \bigvee_{i=1}^{k}(r \in A_x \oplus t_i))\} = P\{\exists r \in U \ \bigwedge_{i=1}^{k}(r \notin A_x \oplus t_i)\} \leq$$

$$\leq \sum_{r \in U} P\{\bigwedge_{i=1}^{k}(r \notin A_x \oplus t_i)\} = \sum_{r \in U} \prod_{i=1}^{k} P\{r \notin A_x \oplus t_i\} \leq \frac{1}{2^{nk}} 2^{p(n)}.$$

Последнее неравенство верно, так как $|U| = 2^{p(n)}$, где $p(n) = \mathrm{poly}(n)$ — длина случайной строки, которой пользуется наш алгоритм из **BPP**; а предпоследнее равенство верно, так как выбор t_i независим. Взяв k, равное $p(n)$, получим требуемое. □

Замечание 6.7. Понятно, что $\mathbf{BPP} \subseteq \Sigma^2\mathbf{P} \cap \Pi^2\mathbf{P}$, так как $\mathbf{BPP} = \mathbf{co\text{-}BPP}$.

Замечание 6.8. На самом деле, можно доказать, что $\mathbf{BPP} \subseteq \mathbf{ZPP}^{\mathbf{NP}}$.