Лекция 10

Вычислимость и сложность

10.1 Вычислимость

Определение 10.1. Обозначим через $M(x)$ результат работы машины M на входе x, если она останавливается; если же M не останавливается, будем писать $M(x) = \infty$.

В дальнейшем мы будем рассматривать языки и массовые задачи в некотором конечном алфавите (необязательно $\{0, 1\}$), но соответствующие обозначения будем опускать (будем просто писать «для любой строки x», подразумевая «для любой строки x в данном алфавите»).

Определение 10.2. Язык $L \subseteq \Sigma^*$ — *рекурсивно-перечислимый*, если существует РАМ M, такая, что $\forall x \in \Sigma^*$ $(x \in L \Leftrightarrow M(x) = 1)$.

Определение 10.3. Язык $L \subseteq \Sigma^*$ — *рекурсивный*, если существует РАМ M, такая, что $\forall x \in \Sigma^*$ $((x \in L \Leftrightarrow M(x) = 1) \wedge (x \notin L \Leftrightarrow M(x) = 0))$.

Заметим, что рекурсивные языки — в точности те, для которых проблема принадлежности разрешима.

Теорема 10.1. *Существует язык, являющийся рекурсивно-перечислимым, но не рекурсивным.*

Лемма 10.1. *Существует* универсальная РАМ U*: в начальный момент времени в первый регистр U подается описание машины T, во второй регистр — вход, а выдает она то, что выдала бы машина T на данном входе (в частности, зациклится, если T зацикливалась). Более того, время работы U полиномиально зависит от времени работы T.*

Упражнение 10.1. Доказать лемму 10.1.

Доказательство теоремы. Определим язык L так: $(M, x) \in L \Leftrightarrow M(x) = 1$. Он, очевидно, рекурсивно-перечислимый. Покажем, что он не рекурсивный.

Пусть он все же рекурсивный. Тогда существует машина A, такая, что
$A((M, x)) = 1 \Leftrightarrow M(x) = 1$,
$A((M, x)) = 0 \Leftrightarrow M(x) \neq 1$.

Построим еще одну машину, D, на вход которой подается описание машины R:
$D(R) = 0$, если $A((R,R)) = 1$;
$D(R) = 1$, если $A((R,R)) = 0$
(чтобы построить ее, воспользуемся леммой 10.1: считаем R и запишем (R,R) во второй регистр; запишем описание A в первый регистр; применим U к A с входом (R,R), предварительно поменяв в ее программе все операторы WRITE 0 на WRITE 1, и наоборот).

Чему равно $D(D)$? Если $D(D) = 1$, то $A((D,D)) = 0$, т.е. $D(D) \neq 1$ (противоречие). Если же $D(D) = 0$, то $A((D,D)) = 1$, т.е. $D(D) = 1$ (противоречие). По построению не может быть и $D(D) = +\infty$, т.е. машины D (а вместе с ней — и машины A) не существует. □

10.2 Элементы теории сложности

10.2.1 Классы P и NP

Пусть Σ — конечный алфавит. Напомним, что массовая задача M есть некоторое множество индивидуальных задач — пар (u,s) (где $u,s \in \Sigma^*$, u — условие, s — решение).

Определение 10.4. $M \in \widetilde{\mathbf{NP}}$, если

1. M — *полиномиально ограничена*, т.е. существует многочлен p, такой, что для любого условия u, если существует хотя бы одно такое s, что $(u,s) \in M$, то существует и s' длины не более $p(|u|)$, такое что $(u,s') \in M$.

2. M — *полиномиально проверяема*, т.е. существует многочлен p, существует алгоритм A, такие, что $\forall u,s \in \Sigma^*((u,s) \in M \Leftrightarrow A(u,s) = 1)$ и при этом A заканчивает свою работу за время, не превосходящее $p(|u| + |s|)$.

Пример 10.1. $\{(N,m) \mid N \vdots m, 1 < m < N\}$.

Пример 10.2 ($\widetilde{\mathtt{SAT}}$ (*задача о выполнимости формулы логики высказываний*)). Дана формула в конъюнктивной нормальной форме (конъюнкция конечного числа дизъюнкций, в каждую из дизъюнкций входят логические переменные либо их отрицания): например,

$$\{(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_4 \vee \neg x_2)\}.$$

Требуется найти значения переменных, такие, что значение всего выражения — истина (в приведенном примере: x_1 — истина, x_2 — ложь). Это задача из $\widetilde{\mathbf{NP}}$: решение — не длиннее условия, и подставить мы его также можем быстро. Формула, для которой такие значения существуют, называется *выполнимой*. □

Определение 10.5. Для каждой массовой задачи M определим язык

$$L(M) = \{u \mid \exists s \ (u,s) \in M\}.$$

Это множество всех условий, для которых существуют решения.

Определение 10.6. Язык L принадлежит классу $\mathbf{NP}$, если $\exists M \in \widetilde{\mathbf{NP}} : L = L(M)$.

Определение 10.7. $M \in \widetilde{\mathbf{P}}$, если существует многочлен p и существует алгоритм A (который может выдавать строку из Σ^* или останавливаться с результатом «решения нет»), такие, что A работает не дольше, чем p(размер входа), и решает задачу M, т.е.

- $A(u) = s \implies (u, s) \in M$;
- $\exists s\ (u, s) \in M \implies A(u) \neq$ «решения нет».

Определение 10.8. $L \in \mathbf{P}$, если существует многочлен p и существует алгоритм A (выдающий 0 или 1), такие, что A работает не дольше, чем p(размер входа), и $\forall u \in \Sigma^*(A(u) = 1 \Leftrightarrow u \in L)$.

Замечание 10.1. $\mathbf{P} \overset{?}{=} \mathbf{NP}$ — центральный (и нерешенный) вопрос теории сложности алгоритмов.

(Гипотеза: $\mathbf{P} \neq \mathbf{NP}$.)

Замечание 10.2. $\mathbf{P} = \mathbf{NP} \iff \widetilde{\mathbf{P}} = \widetilde{\mathbf{NP}}$ (хотя мы этого доказывать не будем).

При этом вполне может существовать массовая задача $T \in \widetilde{\mathbf{NP}}$, такая, что $T \notin \widetilde{\mathbf{P}}$, но $L(T) \in \mathbf{P}$. Возможный претендент — задача о нахождении нетривиального делителя (позже мы узнаем, что если она $\in \widetilde{\mathbf{P}}$, то криптосистему RSA можно взломать; однако, известно, что соответствующий ей язык составных чисел $\in \mathbf{P}$ — это недавний сложный результат, мы его доказывать не будем).

10.2.2 Сводимости и полнота

Заменим вопрос $\mathbf{P} \overset{?}{=} \mathbf{NP}$ на «более простой» (но эквивалентный исходному).

Определение 10.9. Язык L *полиномиально сводится* к языку L' (обозначим это $L \to L'$), если существует многочлен p и существует алгоритм A, работающий не дольше, чем p(длина входа), такие, что $\forall u \in \Sigma^*\ (A(u) \in L' \Leftrightarrow u \in L)$.

Определение 10.10. Язык называется $\mathbf{NP}$-*трудным*, если любой другой язык из $\mathbf{NP}$ к нему сводится. Язык называется $\mathbf{NP}$-полным, если он $\mathbf{NP}$-трудный и при этом сам принадлежит $\mathbf{NP}$.

Теорема 10.2. *Если L — $\mathbf{NP}$-полный и $L \in \mathbf{P}$, то $\mathbf{P} = \mathbf{NP}$.*

Доказательство. Очевидно. □

Теорема 10.3. `SAT` *(язык всех выполнимых формул логики высказываний в конъюнктивной нормальной форме) — $\mathbf{NP}$-полный.*

Доказательство. Напомним, что булева схема — это ...

$\boxed{\text{ПРОБЕЛ В КОНСПЕКТЕ.}}$

CircuitSAT — это ...

ПРОБЕЛ В КОНСПЕКТЕ.

Сначала мы докажем, что CircuitSAT — **NP**-полный, затем сведем `SAT` к CircuitSAT.

ПРОБЕЛ В КОНСПЕКТЕ.

$\square$

Следствие 10.1. *Если* `SAT` $\in P$*, то* $\mathbf{P} = \mathbf{NP}$.

Замечание 10.3. Задача о неэквивалентости булевых схем является **NP**-полной и легко формулируется в терминах `SAT`.

ПРОБЕЛ В КОНСПЕКТЕ.

10.2.3 Алгоритмы, использующие случайные числа

Определение 10.11. $M \in \widetilde{\mathbf{RP}}$, если

1. M — полиномиально ограничена.
2. M — полиномиально проверяема.
3. Каждое разрешимое условие M имеет не менее половины решений, т.е.

$$\forall u((\exists t(u,t) \in M) \Rightarrow$$
$$|\{s \mid (u,s) \in M,\ |s| \leqslant p(|u|)\}| \geqslant$$
$$\frac{1}{2} \cdot \text{кол-во всех строк длины не более } p(|u|))$$

(здесь $|\ldots|$ обозначает в одном случае — мощность множества, а в другом — длину строки; многочлен p — тот, что фигурирует в определении полиномиальной ограниченности).

Определение 10.12. $L \in \mathbf{RP} \Leftrightarrow \exists M \in \widetilde{\mathbf{RP}}\ L = L(M)$.

Очевидно, для задачи из $\widetilde{\mathbf{RP}}$ достаточно выбрать случайную строку длины $p(|u|)$, чтобы получить решение задачи u с вероятностью $\geqslant \frac{1}{2}$. Если повторить эту процедуру k раз, то вероятность успеха будет $1 - \frac{1}{2^k}$, чего для практических целей вполне достаточно.

Теорема 10.4. *Язык, состоящий из всех составных чисел, принадлежит* $\mathbf{RP}$.

Доказательство. В этом доказательстве все числа — неотрицательны. Для начала вспомним несколько определений.

Символ Лежандра:

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & , \text{ уравнение } x^2 \equiv a \pmod{p} \text{ имеет корни} \\ -1 & , \text{ в противном случае} \end{cases},$$

где p — простое, $a \neq 0$.

Символ Якоби:

$$\left(\frac{a}{N}\right) = \prod_i \left(\frac{a}{p_i}\right),$$

если $N = p_1 \cdots p_k$ — разложение N на простые множители (среди p_i могут быть одинаковые). Некоторые свойства:

$$\begin{aligned}
\left(\frac{a}{N}\right) &= (-1)^{\frac{N-1}{2}\frac{a-1}{2}}\left(\frac{N}{a}\right) \quad (\text{здесь } a, N \not\vdots 2,\ \text{НОД}(a,N)=1),\\
\left(\frac{a}{N}\right) &= \left(\frac{a'}{N}\right) \quad (\text{при } a \equiv a' (\text{mod } N)),\\
\left(\frac{1}{p}\right) &= 1,\\
\left(\frac{2}{p}\right) &= (-1)^{\frac{p^2-1}{8}}.
\end{aligned}$$

Упражнение 10.2. При помощи этих свойств вычислить эффективно символ Якоби.

Алгоритм 10.1.
Вход: число N.
Выход: «составное» или «простое».

Если $N \vdots 2$ или $N = 1$, выдать правильный ответ; (10.1)

$M \leftarrow random[2..N-1]$; (10.2)

if $(M, N) \neq 1$ then выдать ответ «составное» (10.3)

else if $\left(\frac{M}{N}\right) \not\equiv M^{\frac{N-1}{2}}$ (mod N) (10.4)

then выдать ответ «составное» (10.5)

else выдать ответ «простое» (*тут алгоритм может ошибиться*) (10.6)

□

Корректность шага (10.4) доказывает следующая лемма:

Лемма 10.2. *$N \in \mathbb{P}$, $N \neq 2 \implies M^{\frac{N-1}{2}} \equiv \left(\frac{M}{N}\right)$ (mod N).*

Доказательство.

ПРОБЕЛ В КОНСПЕКТЕ.

□

Лемма 10.3. *Пусть $N \not\vdots 2$, $N \neq 1$. Если для всех M, таких что $(M, N) = 1$, выполняется $\left(\frac{M}{N}\right) \equiv M^{\frac{N-1}{2}}$ (mod N), то N — простое.*

Доказательство. Будем доказывать от противного:

1. Пусть N не содержит квадратов: $N = p_1 \cdots p_k$, $p_i \neq p_j \in \mathbb{P}$. Фиксируем r такое, что $(\frac{r}{p_1}) = -1$ (такое есть: пересчитаем все квадраты $\bmod\ p_i \ldots$). По китайской теореме об остатках можно выбрать такое M, что

$$\begin{aligned} M &\equiv r \pmod{p_1}, \\ M &\equiv 1 \pmod{p_i} \quad \text{при } i \neq 1. \end{aligned}$$

С одной стороны,

$$\left(\frac{M}{N}\right) = \left(\frac{M}{p_1}\right) \cdot \prod_{i \neq 1} \left(\frac{M}{p_i}\right) = -1.$$

С другой стороны,

$$M^{\frac{N-1}{2}} \equiv 1 \pmod{p_2} \not\equiv -1 (\bmod\ N).$$

Противоречие.

2. Пусть N содержит квадраты: $N = p^2 n$, $p \in \mathbb{P}$. Пусть r — первообразный корень[1] по модулю p^2. По предположению,

$$r^{N-1} \equiv (r^{(N-1)/2})^2 \equiv \left(\frac{r}{N}\right)^2 \equiv 1 \pmod{N},$$

а значит, и $(\bmod\ p^2)$. Т.е., одновременно $N - 1 \vdots p(p-1)$ и $N \vdots p$, т.е., два последовательных числа делятся на p. Противоречие.

□

Лемма 10.4. *Если $N \notin \mathbb{P}$, то для более чем половины всех $M \in [2..N-1]$, взаимно простых с N, $\left(\frac{M}{N}\right) \not\equiv M^{\frac{N-1}{2}} \pmod{N}$.*

Доказательство. По лемме 10.3 существует такое число a, взаимно простое с N, что $\left(\frac{a}{N}\right) \not\equiv a^{\frac{N-1}{2}} \pmod{N}$. Пусть $b_1,\ b_2, \ldots,\ b_k$ — это все остатки, для которых выполнено сравнение $\left(\frac{M}{N}\right) \equiv M^{\frac{N-1}{2}} \pmod{N}$.

Рассмотрим $ab_1,\ ab_2, \ldots,\ ab_k \pmod{N}$. Они все различны, так как если $ab_i \equiv ab_j \pmod{N}$, то $b_i \equiv b_j \pmod{N}$ (ведь $(a, N) = 1$). Значит, их не менее k. При этом для них сравнение не выполняется:

$$\left(\frac{ab_i}{N}\right) = \left(\frac{a}{N}\right)\left(\frac{b_i}{N}\right) = \left(\frac{a}{N}\right) \cdot b_i^{\frac{N-1}{2}} \not\equiv (ab_i)^{\frac{N-1}{2}}.$$

□

Тем самым, вероятность ошибки нашего алгоритма не превосходит $1/2$.

□

[1] a называется первообразным корнем по модулю n, если $a^{\phi(n)} \equiv 1 \pmod{n}$ и $\forall k \in [1..\phi(n){-}1]$ $a^k \not\equiv 1 \pmod{n}$. Известно, что первообразные корни по модулю p^2 существуют.

10.3 Нижняя оценка на время работы алгоритмов для задачи о принадлежности языку

Определение 10.13. $L \in \mathbf{DTime}_R(f)$, если существует РАМ A, такая, что

- $\forall x\ A(x) = 1 \iff x \in L$,
- $\forall x\ A(x)$ работает время, не превосходящее $f(|x|)$.

ПРОБЕЛ В КОНСПЕКТЕ.

Следствие 10.2. $\mathbf{P} \neq \mathbf{EXP}$.

Определение 10.14. $L \in \mathbf{PSPACE}$, если существует РАМ A, такая, что

- $\forall x\ A(x) = 1 \iff x \in L$,
- $\forall x\ A(x)$ использует полиномиальное количество памяти (то есть на каждом шаге исполнения программы суммарная длина всех регистров с ненулевым значением, а также их номеров, не превосходит некоторого полинома от длины битового представления x).

Замечание 10.4. Несложно доказать[2], что

$$\mathbf{P} \subseteq \mathbf{RP} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP},$$

при этом $\mathbf{P} \neq \mathbf{EXP}$, но в каком именно из включений ($\subseteq$) из этой цепочки имеет место неравенство, мы не знаем!

[2]На лекции было пояснено; в конспекте — пробел.