Лекция 1

Задачи поиска; классы $\widetilde{\mathrm{P}}$ и $\widetilde{\mathrm{NP}}$; свед́ения; $\widetilde{\mathrm{NP}}$-полные задачи

(Конспект: В. Моргенштерн)

1.1 Задачи поиска.

Базовым понятием теории сложности является понятие вычислительной задачи, которую мы решаем с помощью той или иной вычислительной модели. Мы будем говорить о сложности решения массовых задач, то есть множества (однотипных) индивидуальных задач.

“Главный” тип вычислительной задачи — это задача поиска. Такая (индивидуальная) задача состоит из условия и множества решений (из которых требуется найти любое); более формально, это множество пар вида (условие, решение). Массовой задаче соответствует предикат, определяющий по условию и решению, что решение удовлетворяет условию. Условие подается на вход вычислительному устройству, и мы ожидаем, что это устройство выдаст решение, удовлетворяющее этому условию (или сообщит, что решения не существует).

Итак, массовая задача — это бинарное отношение на строках. Для отношения R мы будем также обозначать R его характеристическую функцию и писать не только $(x, y) \in R$, но и просто $R(x, y)$.

Разумно предполагать, что мы умеем быстро проверять, что найденное решение – правильное, т.е. упомянутый предикат можно быстро вычислить. В первую очередь мы займемся именно такими вычислительными задачами.

Пример 1.1. **FACTOR** $= \{\ (n, d) \mid d, n \in \mathbb{N},\ n \vdots d,\ 1 < d < n\ \}$ — задача о нахождении нетривиального делителя.

1.2 Классы $\widetilde{\mathbf{P}}$ и $\widetilde{\mathbf{NP}}$.

Рассмотрим конечный алфавит Σ. В дальнейшем Σ^* обозначает множество всех конечных строк в алфавите Σ, $|x|$ обозначает длину строки x.

Определение 1.1. Бинарное отношение $R \subseteq \Sigma^* \times \Sigma^*$ называется полиномиально ограниченным, если существует полином p, такой, что $\forall (x, y) \in R\ (|y| \le p(|x|))$.

Определение 1.2. Бинарное отношение $R \subseteq \Sigma^* \times \Sigma^*$ называется полиномиально проверяемым, если существует полином q, такой, что для любой пары $(x, y) \in \Sigma^* \times \Sigma^*$ можно проверить за время $q(|(x, y)|)$, принадлежит ли (x, y) отношению R или нет.

Важно, что в обоих определениях полиномы не зависят от конкретной пары (x, y).

Определение 1.3. $\widetilde{\mathbf{NP}}$ — класс задач поиска, задаваемых полиномиально ограниченными полиномиально проверяемыми бинарными отношениями.

Определение 1.4. $\widetilde{\mathbf{P}}$ — класс задач поиска из $\widetilde{\mathbf{NP}}$, разрешимых за полиномиальное время, т.е. задаваемых отношениями R, такими, что $\forall x \in \Sigma^*$ за полиномиальное время можно найти $y \in \Sigma^* : (x, y) \in R$.

Замечание 1.1. Заметим, что задача поиска, разрешимая за полиномиальное время, может не принадлежать $\widetilde{\mathbf{P}}$: возьмем просто вычислимую функцию и добавим к ней в качестве "побочных" решений произвольные.

Ключевой вопрос теории сложности: $\widetilde{\mathbf{P}} \stackrel{?}{=} \widetilde{\mathbf{NP}}$.

1.3 Сведе́ния.

Одним из базовых понятий теории сложности является понятие сведе́ния. Если мы умеем сводить задачу D_1 к задаче D_2, значит, из любого эффективного алгоритма для задачи D_2 можно будет сделать эффективный алгоритм для задачи D_1. Понятие сведе́ния (соответственно, "эффективности") здесь может быть разным.

Для задач поиска два основных све́дения таковы.

Определение 1.5. Пусть есть два бинарных отношения $R_1 \subseteq \Sigma^* \times \Sigma^*$ и $R_2 \subseteq \Delta^* \times \Delta^*$. Сведéние R_1 к R_2 *по Левину* состоит из трех функций f, g и h, вычислимых за полиномиальное время[1] и удовлетворяющих следующим условиям:

- $R_1(x_1, y_1) \Leftrightarrow R_2(f(x_1), g(x_1, y_1))$,
- $R_1(x_1, h(f(x_1), y_2)) \Leftrightarrow R_2(f(x_1), y_2)$.

Первое условие говорит о том, что если R_1-задача x_1 имеет решение, то и R_2-задача $f(x_1)$ имеет решение. Второе условие позволяет превратить решение этой R_2-задачи в решение исходной R_1-задачи.

Определение 1.6. Сведéние R_1 к R_2 *по Куку* (оно же по Тьюрингу) — это полиномиальный по времени алгоритм, решающий задачу R_1 при условии, что функция, находящая решение задачи R_2, ему дана "как оракул", т.е. обращение к ней занимает всего один шаг.

Замечание 1.2. Всякое сведéние по Левину является сведéнием по Куку.

1.4 $\widetilde{\mathbf{NP}}$-трудные и $\widetilde{\mathbf{NP}}$-полные задачи.

Мы сейчас определим класс самых трудных задач в $\widetilde{\mathbf{NP}}$. Решив за полиномиальное время хотя бы одну задачу из этого класса мы автоматически получим решение и для всех остальных задач.

Определение 1.7. Задача поиска C называется $\widetilde{\mathbf{NP}}$-трудной, если любая другая задача $R \in \widetilde{\mathbf{NP}}$ сводится по Левину к C.

Замечание 1.3. Естественно, понятие трудности для какого-либо класса зависит от сведéний, которые мы используем. Следует явно упоминать выбранное понятие сведéния, когда это важно. Пока это не для нас не слишком важно; утверждение об $\widetilde{\mathbf{NP}}$-трудности некоторой задачи по Левину является более сильным, и его-то мы в дальнейшем и докажем.

Определение 1.8. Задача C называется $\widetilde{\mathbf{NP}}$-полной, если она $\widetilde{\mathbf{NP}}$-трудная и принадлежит $\widetilde{\mathbf{NP}}$.

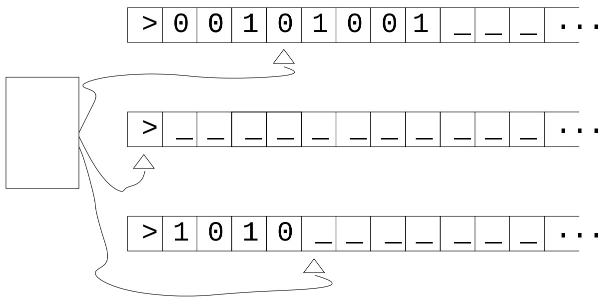
[1] На самом деле достаточно предполагать, что f и h вычислимы за полиномиальное время, а g просто существует. Однако, в исходной статье Левина формулировка была именно такая — более сильная (и определения, и соответствующих утверждений).

Замечание 1.4. Из определения ясно, что все $\widetilde{\mathbf{NP}}$-полные задачи сводятся друг к другу.

Замечание 1.5. Ясно, так же, что $\widetilde{\mathbf{P}} = \widetilde{\mathbf{NP}}$ тогда и только тогда, когда в $\widetilde{\mathbf{P}}$ имеется $\widetilde{\mathbf{NP}}$-полная задача.

Определение 1.9 (отступление — (детерминированная) машина Тьюринга). Детерминированная машина Тьюринга (ДМТ) отвечает интуитивному понятию вычислимости на машине с бесконечной памятью, доступ к которой осуществляется шаг за шагом (бесконечно длинная полка с книгами). Она имеет

- несколько **лент**, т.е. массивов, бесконечных в одну сторону;
- конечный **алфавит**, т.е. множество символов, которые могут быть записаны в клетках лент (в том числе специальные символы "**начало ленты**", записанные в начале каждой из лент, и "**пробел**", которыми заполнены все неиспользованные клетки лент);
- читающие/пишущие **головки**, по одной для каждой ленты, каждая из которых умеет последовательно двигаться по своей ленте, читая или записывая символы на ленту (в один момент времени головка находится у одной позиции ленты — с символом именно в этой клетке она и может работать);
- конечное множество **состояний**, в котором выделены **начальное**, **принимающее** и **отвергающее** состояния;
- и самое главное — **управляющее устройство** (**программу**), содержащее инструкции, *однозначно* определяющие, как по состоянию и символам, обозреваемым головками, решить, в какое состояние перейти, какие символы записать, куда сдвинуть головки (на одну позицию влево, вправо или никуда).



Вычисление на **детерминированной машине Тьюринга** похоже на вычисление на любом реальном вычислительном устройстве:

- в начале работы все ленты пусты, за исключением одной, на которой написано задание (**входное слово**), машина находится в начальном состоянии, все головки находятся в крайней левой позиции;
- шаг за шагом выполняются инструкции программы;
- если машина попадает в конечное (принимающее либо отвергающее) состояние, она заканчивает свою работу.

ДМТ **принимает** входное слово, если она заканчивает свою работу в принимающем состоянии; она **отвергает** его, если заканчивает работу в отвергающем состоянии. Если для всех входных слов ДМТ заканчивает работу, то множество принимаемых ей слов называется **языком, принимаемым** этой машиной. Иначе говоря, машина **решает задачу** принадлежности данному языку.

ДМТ может также **вычислять какую-нибудь функцию**. Значением этой функции на данном входном слове будем считать содержимое первой ленты после достижения конечного состояния.

В некотором роде самой естественной $\widetilde{\mathbf{NP}}$-полной задачей является задача об ограниченной остановке. Прежде чем сформулировать эту задачу, заметим, что любую машину Тьюринга M (т.е. ее функцию перехода) можно закодировать строкой в некотором алфавите. Теперь мы может сформулировать задачу.

Задача 1.1 (об ограниченной остановке). Бинарное отношение R этой задачи вводится следующим образом: $(\langle M, x_1, 1^t\rangle, x_2) \in R$ тогда и только тогда, когда M на входе $\langle x_1, x_2\rangle$ останавливается не более чем через t шагов в принимающем состоянии.

Теорема 1.1. *Задача об ограниченной остановке — $\widetilde{\mathbf{NP}}$-полна.*

Доказательство. **$\widetilde{\mathbf{NP}}$-трудность.** Сведем произвольную задачу $R \in \widetilde{\mathbf{NP}}$ к задаче об ограниченной остановке. Условием будет $\langle M, x, 1^t\rangle$, где M — проверяющий алгоритм, x — условие задачи R, t — время, за которое M должен успеть проверить (самое короткое) решение задачи R (очевидно, оно ограничено полиномом от длины x). Ясное дело, решением будет как раз решение задачи R.

Принадлежность $\widetilde{\mathbf{NP}}$. Можно доказать, что существует так называемый универсальный алгоритм, который получает на вход пару $\langle M, x\rangle$ и, если M останавливается на x, тоже останавливается и выдает то же самое значение, что и M на x, причем время его работы полиномиально зависит от времени работы M на x. Моделируя таким образом машину M из определения задачи об ограниченной остановке, мы сможем за полиномиальное время проверить решение. □

Теорема 1.2 (Кук-Левин). $\widetilde{\mathtt{SAT}}$ *(задача нахождения выполняющего набора для булевой формулы в КНФ) — $\widetilde{\mathbf{NP}}$-полна.*

Доказательство. Подойдет любое известное доказательство, см., например, книгу Гэри и Джонсона. □

1.5 Задачи распознавания.

Большинство задач классической теории сложности формулируются как задачи распознавания, т.е. принадлежности языку. В частности, классы языков $\mathbf{P}$ и $\mathbf{NP}$ определяются следующим образом.

Определение 1.10. $L \in \mathbf{NP}$, если существует $R \in \widetilde{\mathbf{NP}}$, такое, что $x \in L \Leftrightarrow \exists y R(x, y)$.

Определение 1.11. $L \in \mathbf{P}$, если для любой строки x вопрос о принадлежности ее языку L может быть решен за полиномиальное время от ее длины.

На первый взгляд, $\mathbf{NP}$-задачи проще $\widetilde{\mathbf{NP}}$-задач. Тем не менее, верна следующая теорема.

Теорема 1.3. *Пусть $R \in \widetilde{\mathbf{NP}}$, а L — соответствующий ей язык из $\mathbf{NP}$. Если L — $\mathbf{NP}$-труден*[2]*, то R сводится к L по Куку.*

Доказательство. Сведем R к $\widetilde{\mathtt{SAT}}$ (по теореме 1.2).

Ее, в свою очередь, к $\mathtt{SAT}$:

$$F[x_1 \leftarrow a_1, \vec{y} \leftarrow \vec{b}] \in \mathtt{SAT} \Leftrightarrow F[x_1 \leftarrow a_1] \in \mathtt{SAT}$$

(здесь $\vec{y}$ — вектор $(x_2, \ldots, x_n)$ всех переменных, кроме x_1), так что проверим[3] $F[x_1 \leftarrow 0] \in \mathtt{SAT}$ и $F[x_1 \leftarrow 1] \in \mathtt{SAT}$ и присвоим значение переменной x_1 соответственно. Узнав значение переменной x_1, будем узнавать остальные, применив ту же процедуру к переменной x_1 в формуле $F[x_1 \leftarrow a_1]$, и т. д.

$\mathtt{SAT}$ же сводится в L по условию теоремы. □

Наряду со сведением по Куку, для языков можно использовать его более простой частный случай — сведение по Карпу.

---

[2]Полные и трудные языки определяются аналогично полным и трудным задачам поиска.

[3]$F[x_1 \leftarrow a_1]$ является формулой в КНФ: все константы 0 и 1, появившиеся после подстановки, могут быть устранены очевидным образом.

Определение 1.12. Пусть есть два языка L_1 и L_2. Сведéние L_1 к L_2 *по Карпу* (many-one reduction) — это функция f, вычислимая за полиномиальное время, такая, что $\forall x \ (x \in L_1 \Leftrightarrow f(x) \in L_2)$.

Замечание 1.6. Всякое сведéние по Карпу является сведéнием по Куку. Всякое сведéние по Левину является сведéнием языков по Карпу и задач поиска — по Куку.