Математическая Логика и Теория Алгоритмов

Конспект лекций по курсу

Руслан Кулагин

весна 2021

Дисклеймер

Данный конспект не претендует на полноту и корректность. Это всего лишь авторские заметки, которые он пожелал выразить в форме tex файла. Каждый решивший использовать данный конспект для подготовки или изучения делает это на свой страх и риск. Автор рекомендует читать книги и слушать лекции. В случае обнаружения опечаток, помарок или ошибкок, буду рад письму на rus.kulagin2001@yandex.ru или Pull Request на github.

Colophon

Документ был набран при помощи KOMA-Script и LaTeX при использовании класса kaobook.
Исходный код доступен по ссылке:
https://github.com/RKulagin/MLTA
(Вы можете внести свой вклад в развитие этого конспекта!)

Теория NP-полноты | 1

1.1 Два подхода к сложности задачи

Первый подход предлагает считать сложностью алгоритма число тактов работы машины, реализующей этот алгоритм (число тактов работы в Машине Тьюринга, число совершённых подстановок в НАМ, число команд выполненных РАМ).

Второй подход предлагает рассматривать в качестве алгоритма булеву функцию, реализующую этот алгоритм, и в качестве сложности рассматривать число функциональных элементов в схеме из функциональных элементов, реализующих эту функцию. [1]

В любом случае понятно, что для разных индивидуальных задач I сложность будет разной.

Возьмём все индивидуальные задачи I с входом длины $|I| \leq m$. Число таких задач обозначим Z_n. Очевидно, что для конечного алфавита число Z_n — конечно.

Definition 1.1.1 Сложность в худшем назовём $t_Z(n) = \max_{|I|\leq n} t(I)$.

Definition 1.1.2 Сложностью в среднем назовём

$$\frac{\sum\limits_{|I|\leq n} t(I)}{Z_n}.$$

Для каждой задачи может быть сколь угодно много различных алгоритмов $A_Z^1, A_Z^2, \ldots$.

Definition 1.1.3 Сложность задачи — наилучшая из сложностей алгоритмов, решающих эту задачу.

Definition 1.1.4 Будем называть задачу трудно разрешимой, если у всех алгоритмов, решающих её, сложность по меньшей мере экспоненциальная. [Aho]

Для сравнения масштабов, если 1 такт занимает 10^{-6} секунд, то для входа длины 60 алгоритм со сложностью n^6 будет работать 13 минут, а со сложностью 2^n — 366 веков.

При этом развитие техники на время работы экспоненциальных алгоритмов никак влиять не будет. Даже при увеличении мощностей в 1000 раз, экспоненциальный алгоритм будет работать за адекватное время на задаче с длиной входа на 10 символов больше, чем сейчас.[2]

1: Подробнее об этом подходе мы поговорим в соответствующей главе.

Максимальное время работы алгоритма на индивидуальных задачах I с длиной входа $|I| \leq n$.

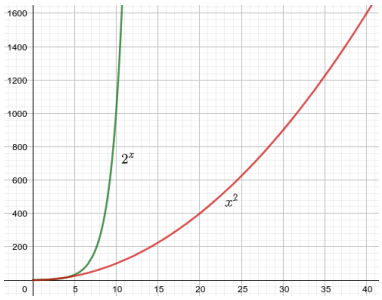

Рис. 1.1: График функций 2^x и x^2. Сравнение графиков функций x^2 и 2^x как пример, насколько сильна разница между экспоненциальной и полиномиальной сложностью. Сравните рост значений в окрестности 10, например.

2: Пусть сейчас наибольшая задача, которая решается за допустимое время имеет размер n, тогда при увеличении мощностей в 1000 раз мы имеем прирост в $\log_2 1000 \simeq 10$

1.2 Определение классов P, NP

Definition 1.2.1 Слово из входных символов допускается (воспринимается) тогда и только тогда, когда машина Тьюринга, начав работу в выделенном начальном состоянии, сделаем последовательность шагов, которые в конце концов приведут её в допускающее состояние.

Definition 1.2.2 Языком L(M), допускаемым машиной M, называют множество всех цепочек (слов) из входных символов, допускаемых M.

Definition 1.2.3 Класс P — это класс языков, допускаемых MT за полиномиальное время.

То есть MT допускает язык и её время работы ограничено полиномом.

Задачи (языки) из класса P называются полиномиально разрешимыми.

Definition 1.2.4 Класс NP — класс языков, допускаемых HMT за полиномиальное время.

Theorem 1.2.1 Если $Z \in NP$, то существует такой полином $p(n)$, что Z может быть решена на детерминированной MT за время $O(2^{p(n)})$.

Доказательство. Если Z решается на HMT за полином $q(n)$, то значит для каждой I найдётся отгадка U(I) после записи которой HMT будет работать как обычная MT и будет проверять отгадку. Время проверки отгадки, написанной угадывающей головкой — $O(q(n))$, а значит и длина отгадки — $O(q(n))$.

Для перебора всех отгадок понадобится в алфавите длины k потребуется $k^{q(n)}$, а значит в сочетании с проверкой всё займёт $O(k^{q(n)} * q(n)) = O(2^{p(n)})^*$. □

*: Просто возьмём достаточно большой полином

1.3 Полиномиальная сводимость

Definition 1.3.1 Язык $L_1 \subset A^*$ сводится к языку $L_2 \subset B^*$, если существует такое отображение $f : A^* \to B^*$, что выполняются два условия.

1. Функция f вычисляется за полиномиальное время. (Например, существует MT, которая за полиномиальное число тактов вычисляет эту функцию.)
2. Для любого входа $I \in A^*$ соотношение $I \in L_1$ выполняется тогда и только тогда, когда выполняется соотношение $f(I) \in L_2$.

Мы знаем, что произведение двух полиномов и полином от полинома также будет полиноном. Получается, если задача Z_1 имеет полиномиальную сложность $O(p(n))$, то, если Z_2 полиномиально сводится к Z_1 за $O(q(n))$, то сложность Z_2 — $O(p(q(n))$. Логично, что если задача может быть сведена к другой за время не меньшее, чем экспоненциальное время, то о принадлежности одним классам говорить нечего.

Полиномиальную сводимость будем обозначать $\leq_P$.

Lemma 1.3.1 Если $Z_1 \leq_P Z_2$ и $Z_2 \leq_P Z_3$, то $Z_1 \leq_P Z_3$

Доказательство. Пусть $Z_1 \leq_P Z_2$ путём отображения f, а $Z_2 \leq_P Z_3$ — g. Тогда рассмотрим суперпозицию gf, которое будет сводить Z_1

к Z_3. При этом, так как f и g полиномы, то и их композиция тоже будет полиномом. □

Lemma 1.3.2 Пусть $Z_2 \in P$, и $Z_1 \leq_P Z_2$. Тогда $Z_1 \in P$.

Доказательство. Пусть A и B — алфавиты языков L_1 и L_2, а функция $f : A^* \to B^*$ осуществляет полиномиальную сводимость L_1 к L_2.

$Z_2 \in P$, значит существует машина Тьюринга M_2, решающая Z_2 за $O(p(n))$ тактов. M_f — машина Тьюринга, реализующая функцию f, работающая за $O(q(n))$ тактов.

Тогда композиция Машин Тьюринга $M_1 = M_2 M_t$, которая сначала преобразует (сведёт) условие задачи Z_1 к условию Z_2, а потом решит её, будет работать за полином.

Если попросили решить задачу, которая может быть полиномиально сведена к уже известной задаче из P, то можно радоваться и писать решение за полином.

□

Theorem 1.3.3 Задача нахождения Гамильтонова Цикла[3] в графе сводится к задаче Комивояжёра.

Доказательство. На вход в задаче о Гамильтоновом цикле поступает граф G = (V, E). Требуется ответить на вопрос: существует ли в графе Гамильтонов цикл.

Входом в задаче Комивояжера является матрица расстояний

$$\left(a_{ij}\right)_{n\times n},$$

число B. Требуется ответить, существует ли обход[4] по всем вершинам суммарной длины $\leq B$.

3: Гамильтонов цикл — цикл в графе, который проходит по каждой вершине графа ровно один раз

$$n = |V|, B = n, a_{ij} = \begin{cases} 1, \{v_i, v_j\} \in G \\ 2, \{v_i, v_j\} \notin G \end{cases}.$$

Осталось доказать, что гамильтонов цикл в G существует тогда и только тогда, когда существует обход длины $\leq n$.

($\Longrightarrow$). Если гамильтонов цикл существует, то всем рёбрам этого цикла соответствует 1 в $\left(a_{ij}\right)$. А значит обход найдётся.

($\Longleftarrow$). Очевидно, что, если обход нашёлся, мы прошли по ребрам длины 1. Значит, в исходном графе мы прошли по существующим рёбрам. Так как обход подразумевает прохождение по всем вершинам, то мы нашли Гамильтонов цикл. □

Theorem 1.3.4 Задача КНФ-выполнимость полиномиально сводится к задаче о клике.

Доказательство. Прежде всего обозначим вход и вопрос для обеих задач.

КНФ-выполнимость на вход получает КНФ

$$K(x_1, \ldots, x_n) = \left(x_{i_1}^{\sigma_1} \cup x_{i_2}^{\sigma_2} \cup \ldots \cup x_{i_p}^{\sigma_p}\right) \& \ldots \left(x_{s_1}^{\kappa_1} \cup x_{j_2}^{\kappa_2} \cup \ldots \cup x_{j_s}^{\kappa_s}\right)$$

Количество скобок в К = m. Вопрос: существует ли такой набор

$$\begin{cases} x_1 = v_1 \\ x_2 = v_2 \\ \ldots \\ x_n = v_n \end{cases}$$

такой, что $K(v_1, v_2, \ldots, v_n) = 1$. Другими словами, ответ "да", если К не тождественный 0.

Задача о клике получает на вход граф $G = (V, E)$, а также число B. Вопрос: существует ли клика, размера $\geq B$ [5] .

4: посещение каждой вершины хотя бы по одному разу

5: Клика — полный подграф.

Пусть V — множество всех литералов[6] в КНФ. При этом даже если x_1 содержится в 1й скобке и во 2-й это будут два разных литерала. Примем B = m. $\{v_i, v_j\} \in G \Leftrightarrow v_i и v_j$ находятся в разных скобках и соответствуют либо разным переменным, либо одинаковым переменным в одинаковой степени.

Докажем, что КНФ выполнима тогда и только тогда, когда построенный нами граф имеет клику.

($\Longrightarrow$). Разложим КНФ на дизъюнкцию мономов. Если КНФ выполнима, то хотя бы один из мономов обращается в 1. Рассмотрим этот моном. Очевидно, что все его литералы были взяты из разных скобок и, так как он обращается в 1, там нет конъюнкции двух одинаковых переменных в разных степенях (иначе он был бы тождественным 0). Получается, что для всех вершин, соответствующих литералам в этом мономе, проведены рёбра между ними. Значит, этому моному соответствует клика. Размер монома — m, значит и размер клики тоже m.

($\Longleftarrow$). Пусть в сгенерированном графе нашлась клика размерности m. Значит, ей соответствуют m литералов из разных скобок, которые образуют моном. Так как произведений $x_i \& \overline{x_i}$ быть не может по условию построения графа, значит этот моном может быть обращён в единицу, а значит КНФ выполнима.

□

1.4 Класс NPC

Definition 1.4.1 Класс NPC (NP Complete) — класс таких задач, что для любой задачи $Z \in NPC$ выполнено условие, что $Z \in NP$ и $\forall Z' \in NP, Z' \leq_P Z$.

Lemma 1.4.1 Если $Z \in NPC$ и $Z \leq_P Z'$, то $Z' \in NPC$

Доказательство этого факта очевидно и прямо следует из определения.

Theorem 1.4.2 Задача о КНФ-выполнимости — NPC.

Доказательство. В данном доказательстве мы не можем пользоваться полиномиальной сводимостью к другим NPC задачам. Только определением класса NP. Задачу КНФ-выполнимость будем сокращать в этом доказательстве как КНФ-в.

По сути мы должны доказать, что $\forall Z' \in NP$, $Z' \leq_P$ КНФ-в.

Задачу Z' можно решить за полином на НМТ, для простоты рассуждений возьмём в качестве полинома — n^k, где n — длина входа $|I|$.

Понятно, что если задача решается за время, меньшее, чем n^k, то она может быть решена и за n^k.

Зададим часть ограничений на нашу НМТ. Пусть она останавливается только в одном конечном состоянии q_{final}

Рассмотрим ту часть работы НМТ, когда она работает как обычная МТ[7] , то есть отгадка уже написана.[8] .

Тогда мы за n^k шагов должны дойти до конечного состояния q_{filal}, проверив отгадку.

Представим работу МТ в виде таблицы, в i-й строке которой записано i-е слово конфигурации этой МТ.

Данная таблица конфигурации будет соответствовать МТ при выполнении 4-х условий одновременно.

1. В каждой ячейке содержится один и только один символ из набора $A \cup S \cup \{\lambda\}$
2. В 0-й строке находятся $-n^k - m$ символов λ после идёт m символов отгадки $u_m, \ldots, u_1$, начальное состояние q_0, l символов условия $v_1, \ldots, v_l$, а после $n^k - l$ символов λ
3. В n^k-й строке есть ячейка, содержащая q_{final}
4. k-я и $(k+1)$-я строки различаются не более чем окошком 2×3, имеющем вид одной из трёх таблиц, которые соответствуют переходам $qb \to q'cR$; $qb \to q'cSt$; $qb \to q'cL$:

№	$-n^k \ldots$	$-m$	$\ldots$	-1	0	1	$\ldots$	l	$\ldots n^k$
0	$\lambda \ldots \lambda$	u_m	$\ldots$	u_1	q_0	v_1	$\ldots$	v_l	$\lambda \ldots \lambda$
n^k	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	q_{final}	$\ldots$	$\ldots$

a	q	b
a	c	q'

a	q	b
a	q'	c

$a, b, c \in A; q, q' \in S$

Если мы сможем перевести эти четыре утверждения в булеву функцию, то значит мы свели NP-задачу к КНФ-в., потому что эти 4 условия выполняются тогда и только тогда, когда МТ успешно проверяет отгадку, а значит ответ на задачу I — да.

Пусть переменная $x_{i,j,\sigma} = 1$, если в ячейке (i, j) таблицы конфигураций содержится символ σ и $= 0$ во всех остальных случаях.

6: Будем разделять литералы и переменные в этом доказательстве. Литералы — конкретные переменные в конкретной степени в конкретной скобке КНФ.

7: Далее под работой МТ будет пониматься именно эта часть

8: Угадывающая головка НМТ работает также за $O(n^k)$

Первая формула запишется как

$$\phi_{line} = \bigcap_{-n^k \le i,j \le n^k} \left(\left(\bigcup_{\sigma \in A \cup S \cup \{\lambda\}} \left(x_{i,j,\sigma}\right) \right) \cap \left(\bigcap_{\substack{\sigma \in A \cup S \cup \{\lambda\}, \\ \xi \in A \cup S \cup \{\lambda\}, \\ \sigma \neq \xi}} \left(\overline{x_{i,j,\sigma}} \cdot \overline{x_{i,j,\xi}}\right) \right) \right).$$

Первая часть отвечает за то, чтобы каждая клетка содержала хотя бы один символ из допустимого множества, вторая часть, чтобы в каждой клетке находилось не более одного символа из множества допустимых значений.

Второе утверждение может быть записано формулой:

$$\phi_{start} = \left(\&_{\substack{-n^k \le j < -m \text{ или} \\ l < j \le n^k}} x_{0,j,\lambda} \right) \& \left(\&_{\substack{-m \le j < 0 \text{ или} \\ 0 < j \le l}} \left(\cup_{\sigma \in A} x_{0,j,\sigma} \right) \right) \& \left(x_{0,0,q_0} \right).$$

Форма записи через или является, возможно, не самое элегантной, зато экономит время.

Третье утверждение записывается формулой так:

$$\phi_{accept} = \cup_{-n^k \le j \le n^k} (x_{n^k,j,q_{final}}).$$

По сути эта формула аналогична фрагменту ϕ_1, мы проверяем, что нашёлся хотя бы один элемент q_{final} в последней строке. Тот факт, что q в каждой строке может быть только один, следует из утверждений 1 и 4.

Утверждение 4 обобщённо будет записано так:

$$\phi_{compute} = \&_{i \le i \le n^k} (\phi_{i,(i+1)}).$$

Продолжение следует... □

1.5 Методы доказательства NP полноты

Сужение задачи.

Метод заключается в установлении того, что задача Z включает в качестве частного случая задачу $Z' \in NP$. При этом даже не обязательно, чтобы Z' была полной копией частного случая, может возникнуть и очевидная биекция.

Примеры можно увидеть в [Geri] на страницах 85–88.

Локальная замена.

Выбирается некоторое характерное свойство известной NPC задачи, с помощью него образуется семейство основных модулей, а соответствующие индивидуальные задачи получаются путём единообразной замены каждого основного модуля некоторой другой структурой.

Например, КНФ-выполнимость может быть сведена к КНФ-3-выполнимости[9] .

a	q	b
q'	a	c

Theorem 1.5.1 Задача 3-выполнимость является NPC.

Доказательство. Доказательство можно посмотреть в [Geri] на страницах 67–69. □

Построение компоненты.

Основная идея: с помощью составных частей рассматриваемой задачи конструируем некоторые "компоненты", соединяя которые можно "реализовать" индивидуальные задачи известной NPC задачи.

Доказательства этим методом является самым сложным, поэтому приводить пример не буду (по крайней мере пока).

1.6 Практическое приложение теории P–NP

Пусть дана задача Z в форме распознавания, которую нужно решить.

- $Z \in P$. Значит для задачи существует полиномиальный алгоритм, который надо найти в источниках и реализовать.
- $Z \in NPC$. Полниомиального алгоритма за 200 лет не нашли, поэтому нужно скать какие-то частные решения или пробовать не точные алгоритмы (об этом в следующей части).
- Если неизвестно, что $Z \in P$ или $Z \in NPC$, то всё будет зависеть от конкретной задачи. В классе NPI (NP Intermediate) не так много задач, имеющих осмысленное приложение. Частные случаи NPC чаще всего будут иметь либо очевидное полиномиальное решение, либо не иметь его совсем.