

Глава 1

Класс NP .

1.1 Определение класса NP .

Определение 1.1 Язык $L \subset \Sigma^*$ принадлежит классу NP , если существует предикат $R(x, y) \in P$ и полином $q(n) = n^k$ такие, что для всех $x \in \Sigma^*$

$$x \in L \Leftrightarrow \exists y(|y| < q(|x|) \wedge R(x, y));$$

соответствующий y называется свидетелем принадлежности $x \in L$.

Замечание. Класс не изменится, если в определении ограничиться предикатами $R(x, y) \in P$ с дополнительным условием $R(x, y) \Rightarrow |y| < q(|x|)$, т.е. функцию отбраковки "длинных" свидетелей передать предикату R тоже. Тогда распознавание принадлежности языку L можно представить как интерактивный процесс, в котором человек пытается "убедить" машину (распознающую предикат R машину Тьюринга) в том, что " $x \in L$ ", а машина верифицирует его доводы. На первой входной ленте написано x . Человек пишет на вторую входную ленту потенциального свидетеля y , а машина вычисляет значение $R(x, y)$ и "соглашается", если результат вычислений есть 1. Слово x принадлежит языку L , если у человека есть принципиальная возможность (но не обязательно эффективный метод) убедить в этом машину.

Замечание. В качестве алфавита для свидетелей достаточно использовать $\{0, 1\}$ (он моделирует остальные). Можно также заменить условие " $|y| < q(|x|)$ " на " $|y| = q(|x|)$ ", т.к. $\varphi(v10^k) = v$ есть вычисляемая за полиномиальное время биекция множества $\{y \mid |y| = m\}$ на $\{y \mid |y| < m\}$ и вместо предиката $R(x, y)$ можно взять $R(x, \varphi(y))$.

Замечание. Каждый язык $L \in NP$ разрешим за время $2^{poly(n)}$ с помощью перебора всех возможных свидетелей.

Другое эквивалентное определение класса NP использует интерактивную модель вычислений – *недетерминированные машины Тьюринга*. Отсюда происходит название класса NP – nondeterministic polynomial. Недетерминированная машина отличается от детерминированной тем, что в программе разрешается использовать несколько команд с одинаковой правой частью “ $q\bar{a} \rightarrow \dots$ ”. Вычисление происходит интерактивно: человек выбирает одну из возможных в данный момент команд, а машина проверяет допустимость выбора и производит действия. Таким образом, для данного входа может быть не одно, а несколько вычислений. *Недетерминированная машина распознает (допускает) язык L за время $q(n)$, если*

$$\begin{aligned} x \in L &\Rightarrow \text{на входе } x \text{ существует вычисление} \\ &\quad \text{длины } \leq q(|x|) \text{ с результатом } 1; \\ x \notin L &\Rightarrow \text{все вычисления на входе } x \text{ не дают} \\ &\quad \text{результата } 1. \end{aligned}$$

И в том и в другом случаях допускается наличие незаканчивающихся вычислений.

Определение 1.2 Язык L принадлежит классу NP , если существует полином q и недетерминированная машина Тьюринга, которая распознает L за время $q(n)$.

Лемма 1.3 Определения 1.1 и 1.2 эквивалентны.

Доказательство. (\Rightarrow). Считаем, что неравенство в определении 1.1 заменено на равенство. Процесс угадывания свидетеля y длины $q(|x|)$ моделируется недетерминированным заполнением ленты последовательностью из $q(|x|)$ нулей и единиц. Соответствующие команды с одинаковой левой частью:

$$\begin{aligned} q\bar{a} &\rightarrow \dots 0 \dots \\ q\bar{a} &\rightarrow \dots 1 \dots \end{aligned}$$

На отдельной рабочей ленте организован счетчик числа повторений, обеспечивающий передачу управления дальше после $q(|x|)$ итераций. После них недетерминированная машина работает детерминированно и вычисляет значение $R(x, y)$.

(\Leftarrow). Легко видеть, что неоднозначность в выборе команд недетерминированной машины можно свести к случаю двух команд с данной

левой частью. Тогда каждое ее вычисление длины $\leq q(|x|)$ однозначно определяется 0,1-последовательностью выборов y той же длины (0 означает, что выбран первый вариант очередной команды, 1 – второй). Рассмотрим детерминированную машину M , которая получает y в качестве дополнительного входа и моделирует интерактивную работу исходной недетерминированной машины, заменяя человеческое управление чтением очередного бита y . Позаботимся также, чтобы эта машина всегда останавливалась за $\leq q'(|x|)$ шагов (для некоторого полинома q') и возвращала 0 всякий раз, когда исходная не печатала 1 за $q(|x|)$ шагов. Такая машина M задает требуемый в определении 1.1 предикат $R \in P$; в качестве оценки на длину свидетелей подходит полином q . ■

1.2 О проблеме $P \neq NP$.

Имеет место включение $P \subset NP$. Для доказательства принадлежности языка $L \in P$ классу NP достаточно взять

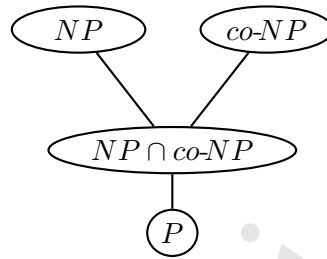
$$R(x, y) = 1 \Leftrightarrow (x \in L), \quad q(n) \equiv 1.$$

Вопрос о строгости этого включения, т.е.

$$P \neq NP?$$

остаётся центральной нерешённой проблемой в теории сложности вычислений уже более 30 лет. Хотя общественное мнение давно склонилось в пользу положительного ответа (т.е. неравно) и полезность многих алгоритмов основывается на этом (предполагаемом) ответе, строгость включения до сих пор не доказана.

Имеется некоторая (частичная) аналогия между качественной теорией алгоритмов и количественной, т.е. теорией сложности. Классы P и NP соответствуют классам разрешимых и перечислимых множеств. Так, гипотеза о несовпадении классов P и NP есть аналог известной теоремы о существовании перечислимого неразрешимого множества. Другой известный результат качественной теории – критерий разрешимости Поста (множество A разрешимо т. и т.т., когда оно и его дополнение A^c перечислимы). Его аналогом является недоказанное и не опровергнутое (но сомнительное) равенство $P = NP \cap co-NP$, где $co-NP$ состоит из всех языков L , чье дополнение L^c принадлежит NP . В настоящее время известны лишь вытекающие непосредственно из определений (нестрогие) включения: $P \subset NP \cap co-NP \subset NP$.

Рис. 1.1: Иерархия классов P , NP , $co-NP$.

1.3 Примеры задач класса NP .

Дальнейшие примеры задач из класса NP требуют кодирования рассматриваемых конечных объектов словами подходящего алфавита. Мы предполагаем, что подобное кодирование уже выбрано некоторым разумным (обычно, очевидным) образом.

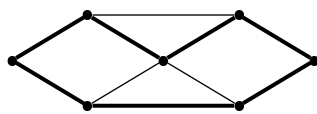
Проблема выполнимости булевых формул (SAT). Булевы формулы – это правильные выражения, построенные из булевых переменных p_1, p_2, \dots с помощью булевых связок \wedge (конъюнкция), \vee (дизъюнкция), \neg (отрицание) и скобок. Например,

$$(\neg(p_1 \wedge p_2) \vee (\neg p_1 \wedge p_3)) \wedge p_2.$$

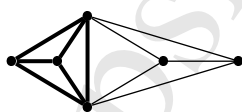
По булевой формуле $\varphi(p_1, \dots, p_n)$ выяснить, существует ли выполняющий ее набор – набор истинностных (0 или 1) значений b_1, \dots, b_n , для которого $\varphi(b_1, \dots, b_n) = 1$.

Свидетелем правильности ответа “да, выполнима” здесь служит любой выполняющий набор битов (b_1, \dots, b_n) . Его размер n не превосходит длины формулы, т.е. ограничен полиномом первой степени от ее длины. Свидетели имеются у всех выполнимых формул и только у них. Проверка того, что некоторый набор из n битов в самом деле свидетельствует о выполнимости формулы, сводится к вычислению ее истинностного значения при заданных значениях переменных, что проверяется за полиномиальное время. Отсюда, $SAT \in NP$.

Проблема существования гамильтонова цикла. По графу выяснить, существует ли в нем гамильтонов цикл (замкнутый путь по ребрам графа, проходящий через каждую вершину ровно 1 раз). Свидетель здесь – гамильтонов цикл.



Задача о клике. Подграф данного графа называется кликой, если он полный, т.е. каждая пара его вершин соединена ребром. Размер клики – это число ее вершин. По графу выяснить, существует ли в нем клика данного размера d . Свидетель здесь – клика.



Полимино, краевая задача. Полимино – это двухмерное домино. Костяшки – квадратные, на всех четырех сторонах – пометки (буквы фиксированного алфавита). Для каждого варианта игры фиксирован свой набор типов квадратиков. Краевая задача: на сторонах клетчатого прямоугольника $m \times n$ пометки заданы. По краевой задаче и варианту игры определить, можно ли замостить весь прямоугольник по правилам домино. Для определенности считаем, что крутить костяшки нельзя (фиксирован верх). Свидетель – замощение.

Существование целочисленного решения системы линейных неравенств. Дана система линейных неравенств с целыми коэффициентами. Выяснить, имеет ли она целочисленное решение. Свидетель – решение. (Для доказательства принадлежности классу NP нужна полиномиальная оценка на длину свидетеля. Можно показать, что если такая система имеет целочисленные решения, то длина некоторого из них оценивается сверху полиномом от длины системы.)

Другие примеры. Следующие задачи лежат в классе NP , но представляются более простыми, чем приведенные выше:

- Проверить, что данное натуральное число – составное.¹ Свидетель здесь – разложение натурального числа в произведение двух натуральных сомножителей, каждый из которых больше 1.

¹В августе 2002 года индийские математики М. Agrawal, N. Kayal и N. Saxena объявили результат, утверждающий принадлежность этой задачи классу P .

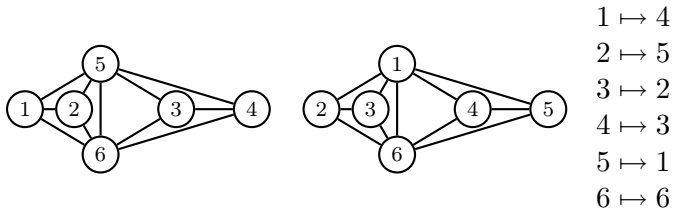


Рис. 1.2: Изоморфизм двух графов.

- Два графа называются изоморфными, если между их вершинами можно установить взаимно-однозначное соответствие, которое сохраняет смежность: если две вершины одного графа соединены ребром, то их образы в другом графе – тоже. Само соответствие называется изоморфизмом. Исходными данными задачи является пара графов. Проверить, что данные графы изоморфны. Свидетель здесь – изоморфизм.

Глава 2

Примеры NP -полных задач.

2.1 Сводимость \leq_m^p (Карп), NP -полнота.

Общий подход к сравнению задач по трудности основан на понятии сводимости задач. Он позволяет сравнивать задачи, имеющие форму массовых проблем (под массовой проблемой понимается семейство однотипных индивидуальных задач, различающихся лишь значениями исходных данных). Неформально, задача \mathcal{A} сводится к задаче \mathcal{B} (т.е. не труднее, " $\mathcal{A} \leq \mathcal{B}$ "), если существует метод (протокол), позволяющий решать каждую индивидуальную задачу из \mathcal{A} , пользуясь решениями индивидуальных задач из \mathcal{B} , поставляемыми "оракулом" извне (по требованию). Естественно ожидать рефлексивность и транзитивность отношения сводимости \leq . Известно много различных по силе формализаций этого понятия. При изучении сложных классов в первую очередь используют "много-однозначную сводимость за полиномиальное время" \leq_m^p (сводимость Карпа).

Класс задач ограничивается массовыми проблемами распознавания языков. Для языка $L \subset \Sigma^*$ задача распознавания такова:

Вход:	Вопрос:
слово $x \in \Sigma^*$	$x \in L$?

Определение 2.1 *Говорят, что язык $L_1 \subset \Sigma^*$ сводится по Карпу к языку L_2 , т.е. $L_1 \leq_m^p L_2$, если существует функция $f \in P$ такая, что для всех слов $x \in \Sigma^*$ выполнено: $x \in L_1 \leftrightarrow f(x) \in L_2$.*

$$L_1 \equiv_m^p L_2 \Leftrightarrow (L_1 \leq_m^p L_2) \wedge (L_2 \leq_m^p L_1).$$

Лемма 2.2 1. Отношение \leq_m^p рефлексивно и транзитивно.

2. $L_1 \leq_m^p L_2, L_2 \in P \Rightarrow L_1 \in P$

3. $L_1 \leq_m^p L_2, L_2 \in NP \Rightarrow L_1 \in NP$

Доказательство. Простое упражнение. ■

Определение 2.3 Язык L называется NP -трудным, если все языки из NP к нему \leq_m^p -сводятся. NP -трудный язык называется NP -полным, если он сам также принадлежит классу NP .

Замечание. Понятие NP -полноты первоначально возникло в связи с попытками доказать гипотезу $P \neq NP$: если это так, то NP -полные языки должны заведомо лежать в разности $NP \setminus P$. В настоящее время известно много примеров NP -полных задач, но ни про одну из них не удалось доказать (или опровергнуть), что она не лежит в классе P . Сейчас характеристика задачи как NP -полной (и NP -трудной тоже) воспринимается как довод в пользу невозможности ее практического решения программными средствами в столь общей постановке: написать программу вполне реально, но обеспечить эффективность ее работы на всех исходных данных не удастся. Естественный выход в этой ситуации – сужение задачи, переход к частным случаям, как раз к тем, в которых программа работает приемлемым образом. (Разработчикам программных продуктов автор настоятельно рекомендует делать это самостоятельно и заблаговременно, не дожидаясь уличения во лжи со стороны пользователей.)

2.2 NP -полнота проблемы выполнимости булевых формул.

Теорема 2.4 Проблема выполнимости булевых формул NP -полна.

Доказательство. Мы видели, что язык SAT , состоящий из всех записей (связки \neg, \wedge, \vee , скобки, переменная p_5 записывается $p101$) выполнимых булевых формул принадлежит классу NP . Достаточно доказать, что произвольный язык $L \in NP$ сводится к SAT . Общий случай сводится к следующему: язык $L \subset \{0, 1\}^*$ задается полиномом p и предикатом $R \in P$ так

$$x \in L \Leftrightarrow \exists y \in \{0, 1\}^* (|y| = p(|x|) \wedge R(x, y))$$

и

$$|y| \neq p(|x|) \Rightarrow \neg R(x, y).$$

В доказательстве теоремы ?? о включении $P \subset P/Poly$ мы предложили метод, который для произвольного фиксированного предиката из класса P по длине входа n строит булеву схему полиномиального размера, вычисляющую этот предикат на словах длины n . Этот метод оказывается эффективным в следующем смысле: само отображение

$$n \mapsto \text{булева схема } S_n$$

оказывается вычислимым за полиномиальное время (см. его описание). Применим эту конструкцию к предикату R . Тогда схема $S_{n+p(n)}$ вычисляет значение предиката $R(x, y)$ по набору битов его аргументов $x_1, \dots, x_n, y_1, \dots, y_{p(n)}$.

Переименовывая рабочие переменные схемы добьемся того, чтобы переменные в левых частях операторов присваивания не повторялись. После этого сопоставим схеме булеву формулу $\varphi(x_1, \dots, x_n, y_1, \dots, y_{p(n)}, z_1, \dots, z_k)$ так: каждому оператору присваивания $z \leftarrow t$ схемы сопоставим булеву формулу $(z \leftrightarrow t)$, возьмем конъюнкцию их всех и еще переменной, возвращающей значение всей схемы. Переменными формулы φ являются все переменные схемы, а длина φ также ограничена сверху полиномом от n .

Обозначим через $b(v)$, $v \in \{0, 1\}^n$, набор формул той же длины n с компонентами

$$b_i = \begin{cases} w \wedge \neg w & , v_i = 0, \\ w \vee \neg w & , v_i = 1, \end{cases}$$

где w – не встречавшаяся ранее булева переменная. Рассмотрим следующее отображение f :

$$v \in \{0, 1\}^* \mapsto \left\{ \begin{array}{l} n := |v| \\ b := b(v) \\ S_{n+p(n)} \end{array} \right\} \mapsto \psi(w, y, z) := \varphi(b, y, z).$$

Оно вычислимо за полиномиальное время, причем

$$v \in L \Leftrightarrow \text{формула } \psi \text{ выполнима} \Leftrightarrow f(v) \in SAT.$$

■

Замечание. Если в качестве встроенных функций языка булевых схем использовать только одноместные и двухместные операции (например, \neg, \wedge, \vee), то в каждом операторе присваивания будет не более трех

переменных. Тогда конъюнктивные члены формулы $\psi = \bigwedge \psi_j$ также будут содержать не более трех переменных каждый. Приведем каждый из них к КНФ. Это за полиномиальное время даст приведение формулы ψ к КНФ специального вида, в котором каждый конъюнктивный член содержит не более 3 переменных (так называемые 3-КНФ). Так доказывается

Следствие 2.5 Проблема выполнимости 3-КНФ (обозначение: 3SAT) NP -полна.

Замечание. Заметим, что проблемы выполнимости 2-КНФ и ДНФ принадлежат классу P . Как следствие этого получим, что если $P \neq NP$, то задача приведения 3-КНФ к виду ДНФ за полиномиальное время не решается.

2.3 NP -полнота задачи о клике.

Теорема 2.6 Задача о клике NP -полна.

Доказательство. Язык $CLIQUE \in NP$ состоит из слов, кодирующих пары (G, d) , где G - граф, d натуральное число и в графе G есть полный подграф из d вершин (клика). Достаточно доказать, что $3SAT \leq_m^p CLIQUE$.

Напомним, что литералом называется булева переменная или ее отрицание. 3-ДНФ φ есть конъюнкция скобок, каждая из которых есть дизъюнкция не более трех литералов:

$$\varphi = \dots \wedge (l_{i,1} \vee l_{i,2} \vee l_{i,3}) \wedge \dots \wedge (l_{j,1} \vee l_{j,2} \vee l_{j,3}) \wedge \dots$$

Сопоставим ей граф G , вершинами которого будут все вхождения всех литералов в формулу φ . Два вхождения соединим ребром, если они находятся в разных скобках и не являются "противоположными", т.е. не есть пара x и $\neg x$. Граф G будет выглядеть примерно так:

$$\dots x \dots \wedge (\neg x \vee y \vee \neg z) \wedge \dots \neg x \dots$$

Для каждой клики в графе нетрудно подобрать истинностные значения переменных, обращающие все литералы клики в 1. Если при этом размер клики d равен количеству конъюнктивных членов в формуле, то

вся формула обратится в 1. Верно и обратное: если для некоторой истинностной оценки переменных $\varphi = 1$, то в каждом конъюнктивном члене можно выбрать по литералу так, чтобы получилась клика (надо выбрать $l_{i,k} = 1$ по одному из скобки). Т.е. для функции $f(\varphi) := \text{код}(G, d)$, где d количество дизъюнктивных членов φ , выполняется

$$\varphi \in 3SAT \Leftrightarrow f(\varphi) \in CLIQUE.$$

Сама f вычислима за полиномиальное время, если все кодирования выбраны достаточно естественно (например, когда граф кодируется матрицей смежностей, числа – двоичным представлением, формулы – своей записью). Заметим, что константа 3 в этом доказательстве не использовалась. ■

2.4 NP-трудность задачи целочисленного линейного программирования.

Собственно говоря, задача линейного программирования – это задача поиска экстремума линейного функционала при наличии ограничений, выраженных системой линейных неравенств. Она не является задачей распознавания языков, поэтому формально нельзя говорить о ее NP-трудности. Но любые методы ее решения также вынуждены распознавать совместность ограничений, поэтому задача линейного программирования не менее трудна, чем задача распознавания совместности систем линейных неравенств. В целочисленном случае последняя задача уже оказывается (после надлежащей кодировки) задачей распознавания языка, причем NP-полной задачей. В этом смысле мы утверждаем NP-трудность задачи целочисленного линейного программирования.

Теорема 2.7 *Задача распознавания разрешимости в целых числах системы линейных неравенств с целыми коэффициентами NP-полна.*

Доказательство. Условие выполнимости КНФ можно эквивалентно переписать в виде условия разрешимости в целых числах следующей системы неравенств. Запишем условие булевозначности для каждой из переменных:

$$0 \leq x_j \leq 1,$$

а каждому конъюнктивному члену $(l_1 \vee \dots \vee l_k)$ сопоставим неравенство

$$p_1 + \dots + p_k > 0, \quad \text{где } p_i = \begin{cases} x_j, & \text{если } l_i \text{ есть } x_j, \\ 1 - x_j, & \text{если } l_i \text{ есть } \neg x_j. \end{cases}$$

■

V. Krupski
COMPLEXTU
Lecture Notes, draft