Классические и квантовые вычисления

А. Шень (вр.и.о. А. Китаева); написал М. Вялый

3 лекция (26/02/98)

Дополнение к предыдущей лекции

Приведём ещё один простой пример сведéния.

Задача ЦЛП (целочисленного линейного программирования). Дана система линейных неравенств с целыми коэффициентами. Есть ли у неё целочисленное решение? (Другими словами, *совместна* ли система?)

В этой задаче входом является матрица коэффициентов и вектор правых частей. То, что ЦЛП $\in \mathcal{NP}$, не совсем очевидно. В качестве подсказки Мерлин сообщает Артуру значения переменных, при которых выполнены все неравенства системы. Можно доказать, что длина этого сообщения не очень велика, другими словами, у совместной системы всегда есть целочисленные решения, размер записи которых ограничен полиномом от длины записи системы.

Сведём теперь 3-КНФ к ЦЛП. Построим по 3-КНФ систему линейных неравенств. Каждой логической переменной x_i из 3-КНФ сопоставим целочисленную переменную p_i. Система содержит для всех i неравенства $0 \le p_i \le 1$, кроме того, для каждой дизъюнкции $X_j \vee X_k \vee X_m$ в системе есть неравенство $P_j + P_k + P_m \ge 1$, где $P_j = p_j$, если $X_j = x_j$, и $P_j = 1 - p_j$, если $X_j = \neg x_i$. Очевидно, что выполнимость 3-КНФ будет эквивалентна совместности такой системы неравенств.

Заметим, что если не требовать целочисленности решений, то можно установить совместность системы линейных неравенств за полиномиальное время (для этого есть несколько алгоритмов: Хачияна, Кармаркара, Ренегара).

Обширный список $\mathcal{NP}$-полных задач содержится в книге Гэри и Джонсона «Вычислительные машины и труднорешаемые задачи». Как правило, они получаются сведéниями. Приведём несколько примеров.

3-раскраска. Дан граф. Спрашивается, можно ли раскрасить его вершины в три цвета так, что концы каждого ребра покрашены в разные цвета. (Аналогичная задача 2-раскраска решается за полиномиальное время; заметим также, что 2-КНФ (очевидный вариант рассмотренной выше задачи 3-КНФ) также решается за полиномиальное время.)

Клика. Даны граф и число k. Спрашивается, есть ли такое подмножество вершин графа, которое содержит k элементов и любые две вершины в нём связаны ребром.

1. Вероятностные алгоритмы и класс $\mathcal{BPP}$

В вероятностных МТ (вМТ), как и в недетерминированных, имеются состояния, из которых возможен переход в два разных состояния. Но отличие состоит в том, что состояние, куда вМТ делает переход, определяется результатом некоторого случайного процесса («подбрасывания монеты»). Нужно оговорить, какие монеты допускаются (если взять очень редкую монету, вероятность выпадения герба для которой равна невычислимому числу, то возможности такой вМТ будут значительно выше, чем хотелось бы; она сможет распознавать и некоторые неразрешимые предикаты).

Обычно считают, что вероятности одинаковы для обеих сторон монеты[1)] и равны 1/2.

В отличие от НМТ, вероятностные машины могут мыслиться как вполне реальные устройства. Хотя для вероятностной машины Тьюринга и нельзя сказать, какой ответ она выдаст, но можно определить вероятность того или иного ответа.

Определение 1. Предикат L принадлежит классу $\mathcal{BPP}$, если существуют вМТ M и полином $p(n)$ такие, что через время $\leq p(|x|)$ машина M остановится, причём
$L(x) = 1 \implies$ с вероятностью $> 2/3$ M выдаст ответ "да";
$L(x) = 0 \implies$ с вероятностью $> 2/3$ M выдаст ответ "нет".

Если в определении заменить число 2/3 на любое число, бóльшее 1/2, класс $\mathcal{BPP}$ не изменится. Есть простой способ добиться вероятности, сколь угодно близкой к 1. Возьмём несколько одинаковых машин, запустим их все, а окончательным результатом будем считать мнение большинства. Если вероятность правильного ответа для каждого экземпляра машины равна $c > 1/2$, то вероятность правильного ответа после голосования n машин не меньше $1 - \lambda^n$, где $\lambda < 2\sqrt{c(1-c)} < 1$.

Класс $\mathcal{BPP}$ можно определить с помощью предикатов от двух переменных, как это было сделано для класса $\mathcal{NP}$.

Определение 2. *$L \in \mathcal{BPP} \iff$ существуют многочлен $q(\cdot)$ и предикат, $R(\cdot,\cdot) \in \mathcal{P}$, такие, что*
$L(x) \implies$ *доля слов r длины $q(|x|)$, для которых выполнено $R(x,y)$, больше 2/3 ;*
$\neg L(x) \implies$ *доля слов r длины $q(|x|)$, для которых выполнено $R(x,y)$, меньше 1/3.*

Докажем, что эти два определения эквивалентны.

Опр. 1 $\implies$ опр. 2.
Полагаем $q(n) = p(n)$ (бросаний монеты не больше, чем общее число действий). Определим предикат $R(x,y)$ = «машина M на x даёт ответ "да" при указанной в y последовательности результатов бросаний монеты». Этот предикат по очевидным причинам удовлетворяет определению 2.

Опр. 2 $\implies$ опр. 1.
Случайно выбираем слово r длины $q(|x|)$ и подставляем в предикат, затем вычисляем значение предиката. Такая вМТ удовлетворяет определению 1.

Определение 2 допускает следующее наглядное толкование. Отметим в плоскости (x,y) точки, для которых выполнен некоторый предикат $R(x,y)$, т. е. множество $\{(x,y)|R(x,y)\&y = q(|x|)\}$ Для $x = x_0$ рассмотрим сечение этого множества $\{(x,y)|R(x,y)\&y = q(|x|)\&x = x_0\}$. Предикат $R(\cdot,\cdot)$, участвующий в определении 2, обладает таким странным свойством, что мера сечения при любом x_0 либо больше 2/3, либо меньше 1/3. Это разделяет значения x на две категории; одна соответствует предикату L, другая — его отрицанию.

[1)] Дальше будем считать, что стороны монеты помечены 0 и 1.

2. Проверка простоты числа — классический пример задачи из $\mathcal{BPP}$

Задача Проверка простоты: дано число n. Требуется определить, простое ли оно.

Для этой задачи существует вероятностный алгоритм, работающий полиномиальное время; он будет ниже описан.

Необходимые сведения из теории чисел

Теорема (малая теорема Ферма). *Если n — простое и $n \nmid a$, то $a^{n-1} \equiv 1 \pmod n$.*

Можно доказывать эквивалентное сравнение $a^n \equiv a \pmod n$, используя сравнение $(a+b)^n \equiv a^n + b^n \pmod n$. Последнее сравнение сразу вытекает из формулы бинома, если учесть, что почти все биномиальные коэффициенты $\binom{n}{k}$ делятся на (простое) n.

Для дальнейшего будет полезен другой способ доказательства. Пусть $a^s \equiv a^q \pmod n$. Тогда из взаимной простоты a и n следует, что $a^{s-q} \equiv 1 \pmod n$. Поэтому последовательность $1, a, a^2, \ldots, a^t, \ldots$ состоит из различных чисел вплоть до некоторой степени $s_0 - 1$, а $a^{s_0} \equiv 1 \pmod n$. Более того, аналогично доказывается, что в любой последовательности $\{ba^t\}_{t=0}^{\infty}$ ровно s_0 различных элементов.

Из малой теоремы Ферма следует, что неравенство $a^{n-1} \not\equiv 1 \pmod n$ позволяет утверждать, что n — составное (как говорят, a является *свидетелем* непростоты числа n). Это свидетельство косвенное — явного разложения n на множители мы не получаем. Это сильный способ проверки простоты: как правило, достаточно проверки при $a = 2$!

Однако, проверки малой теоремы Ферма даже при всех a может оказаться недостаточно. Алгоритм проверки будет использовать свидетелей ещё одного типа: есом $b^2 \equiv 1 \pmod n$, а $b \not\equiv \pm 1 \pmod n$, то n — составное; n и $b-1$ имеют общий делитель, бо́льший 1. Поэтому свидетели такого вида (вообще говоря, гораздо более редко появляющиеся) позволяют сразу же указать разложение составного n (о котором они свидетельствуют), на два множителя за полиномиальное время (наибольший общий делитель (x, y) двух чисел x и y можно найти за полиномиальное время, см. далее).

Теорема (китайская теорема об остатках). *Пусть $n = uv$, $(u, v) = 1$. Тогда $(\mathbb{Z}/n\mathbb{Z}) = (\mathbb{Z}/u\mathbb{Z}) \times (\mathbb{Z}/v\mathbb{Z})$.*

Другими словами, существует взаимно однозначное соответствие между остатками от деления на n и парами остатков от деления на u и на v.

Необходимые сведения из алгоритмической теории чисел

Арифметические операции над числами можно выполнять за полиномиальное время от длины их записи (число n записывается $l(n) \approx \log n$ знаками) самым обычным способом. Например, умножение чисел n, m столбиком требует времени $l(n)l(m)$, т. е. квадратично от длины входа (Хотя квадрат от длины входа уже достаточно мал, чтобы быть полиномиальным, отметим, что можно умножать числа и быстрее, за время $O(n \log n \log\log n)$.)

Остаток от возведения числа a в степень n можно найти за время $\log n$, если вычислить остатки от степеней a^{2^k} последовательным возведением в квадрат и затем перемножить те из полученных чисел, которые 1 в двоичной записи числа n.

Наибольший общий делитель двух чисел можно найти, пользуясь алгоритмом Евклида. Этот алгоритм использует рекурсивно равенство $(x, y) = (y, x \div y)$, где $x \div y$ обозначает остаток от деления x на y. За каждые два шага применения такого равенства длина записи наименьшего в рассматриваемой паре чисел уменьшается на константу.

И последнее. Во избежание теоретико-числовых нюансов алгоритм будет проверять, извлекается ли из числа нацело корень k-той степени. Это можно сделать, найдя достаточно точно приближённое значение корня и возведя ближайшее к нему целое число в k-тую степень. Детали полиномиального алгоритма для выполнения этих действий оставляются для самостоятельного обдумывания.

Основной алгоритм

Алгоритм. (Проверка простоты.)

Вход: число n.

Шаг 1. Проверяем, что n нечётное, а если n — чётное, то сравниваем его с 2. Если $n = 2$, то ответ "n — простое", если n — чётное и больше 2, то ответ "n — составное", в противном случае переходим к шагу 2.

Шаг 2. Проверяем, не есть ли n k-той степенью некоторого целого числа при $k = 2, \ldots, \log_2 n$. Если n — степень, то ответ "n — составное", иначе переходим к шагу 3.

Шаг 3. Записываем $n - 1$ в виде $2^k \cdot l$, $k > 0$, l — нечётное.

Шаг 4. Выбираем случайное a среди чисел от 1 до n.

Шаг 5. Вычисляем a^l, a^{2l}, $\ldots$, a^{n-1} по модулю n.

Проверка 1. Если $a^{n-1} \not\equiv 1 \pmod{n}$, то ответ "$n$ — составное".

Проверка 2. Если в процессе возведения в квадрат найдено x, такое что $x^2 \equiv 1 \pmod{n}$, $x \not\equiv \pmod{n}$, то ответ "n — составное".

В противном случае ответ "n — простое".

Теорема. *Если n — простое, то описанный выше алгоритм всегда (с вероятностью 1) выдаёт ответ "n — простое".*

Если n — составное, то ответ "n — составное" будет получен с вероятностью ≥ 2.

Замечание. Чтобы получить $\mathcal{BPP}$-алгоритм проверки простоты числа, нужно дважды применить приведенный алгоритм. Тогда вероятность ошибки станет меньше 1/4.

Доказательство теоремы

Из сказанного выше следует, что в доказательстве нуждается только второе утверждение.

Пусть $n = uv \equiv 1 \pmod{2}$, $(u, v) = 1$ (если такого представления нет, то на шаге 2 будет обнаружена непростота). Будем доказывать, что вероятность успешного тестирования не меньше 1/2 для любого a, взаимно простого с n (для остальных a вероятность успешного тестирования равна 1, так что полученная оценка только улучшится).

Обозначим множество $(\mathbb{Z}/u\mathbb{Z})^*$ остатков по модулю u, взаимно простых с u, через U; аналогичное множество остатков по модулю v через V. Из китайской теоремы следует, что $(\mathbb{Z}/n\mathbb{Z})^* = U \times V$.

Рассмотрим множества $U^k = \{x^k | x \in U\}$, $V^k = \{x^k | x \in V\}$. Заметим, что $|U|/|U^k|$ — целое (если $x^l \equiv y^l \pmod{u}$, то $(xy^{-1})^l \equiv 1 \pmod{u}$, рассуждение аналогичное приведенному выше доказательству малой теоремы Ферма показывает, что число прообразов для каждой k-той степени одинаково).

Очевидно, что $U^{2l} \subseteq U^l$ (степеней квадратов не больше, чем всех степеней). Поэтому получаем пару невозрастающих цепочек множеств

$$U^l \supseteq U^{2l} \supseteq \cdots \supseteq U^{n-1} \supseteq \{1\}, \qquad V^l \supseteq V^{2l} \supseteq \cdots \supseteq V^{n-1} \supseteq \{1\}.$$

1 случай. Пусть $U^{n-1} \neq \{1\}$ или $V^{n-1} \neq \{1\}$. Чтобы пройти проверку 1 в алгоритме, после возведения в $(n-1)$-ю степень должна получиться пара остатков $(1, 1)$. Поскольку прообразов каждого числа из $U^{n-1} \neq \{1\}$ одинаковое количество, вероятность получения пары $(1, 1)$ не выше $1/2$.

2 случай. Пусть $U^{n-1} = V^{n-1} = \{1\}$. Поскольку l — нечётное, $\{\pm 1\} \in U^l \cap V^l$, т. е. найдётся такое t, что $U^{2t} = V^{2t} = \{1\}$, а $U^t \neq \{1\}$ или $V^t \neq \{1\}$.

Будем доказывать, что с вероятностью не меньше $1/2$ в этом месте проверка 2 алгоритма обнаружит, что n — составное. Имеем $a^{2t} \equiv 1 \pmod n$, нужно понять, с какой вероятностью a^t не равно ± 1. Нужно рассмотреть два случая.

Случай 2а. Одно из множеств U^t, V^t не равно $\{1\}$. В этом случае можно утверждать, что $a^t \not\equiv -1 \pmod n$ (остатку -1 при делении на n соответствует пара остатков $(-1, -1)$ при делении на u, v.

Можно рассуждать так же, как и в случае 1. С вероятностью не меньше $1/2$ получим пару остатков $(1, \alpha)$, $\alpha \neq 1$.

Случай 2б. Оба множества U^t, V^t содержат по крайней мере два элемента. В этом случае может получиться как $a^t \equiv 1 \pmod n$, так и $a^t \equiv -1 \pmod n$, но вероятность этого две каких-то (по крайней мере, четвёртых), т. е. не больше $1/2$.

Доказательство закончено.

Замечание. Шаг 2 в алгоритме избыточен, дополнительными соображениями устанавливается, что и для чисел вида m^k описанные выше проверки дают правильный ответ со значительной вероятностью.

3. Ещё кое-что о $\mathcal{BPP}$

Теорема. $\mathcal{BPP} \subsetneq {}_{non-uniform}\mathcal{P}$.

Доказательство. Идея доказательства состоит в том, чтобы вначале усилить оценки вероятностей с $(1/3, 2/3)$ до $(\varepsilon, 1-\varepsilon)$, нужное значение ε будет указано ниже; а затем зафиксировать некоторые значения случайных бит и построить схему, как описано в предыдущей лекции.

Суть здесь в том, что повторение опытов за полиномиальное время экспоненциально уменьшает оценку вероятности ошибки ε, но не меняет размер входа $|x|$. Поэтому при $\varepsilon 2^{|x|} < 1$ найдётся такой набор случайных бит, при котором нет ошибок ни при каких x.

Действительно, вспомним интерпретацию $\mathcal{BPP}$, данную после второго определения. Если доля слов r, на которых совершается ошибка, для каждого x не превосходит ε, то доля тех слов r, на которых происходит ошибка при каком-нибудь x, не превосходит $\varepsilon 2^{|x|} < 1$. Значит, найдутся и такие r, на которых нет ошибок ни при каком x. ☐

Это типичное неконструктивное доказательство существования. Мы доказываем, что вероятность того, что объект с нужными свойствами не существует, меньше 1. Но это означает, что хотя бы один такой объект существует.

Теорема. $\mathcal{BPP} \subsetneq \Sigma^2$.

Оставим эту теорему пока без доказательства (и без объяснения формулировки).