1982
Обзор вычислительной сложности

Стивен А. Кук

Университет Торонто

Премия Тьюринга 1982 г. была вручена Стивену Артуру Куку, профессору информатики университета Торонто, на ежегодной конференции ACM в Далласе 25 октября 1982 г. Эта награда — высшее признание Ассоциацией технических достижений в информатике.

Достижения Кука охарактеризованы следующим образом: «Д-р Кук существенно и глубоко продвинул наше понимание сложности вычислений. Его плодотворная работа о сложности процедуры доказательства теорем, представленная на симпозиуме ACM SIGACT по теории вычислений 1971., заложила основу теории *NP*-полноты. Последовавшие за этим исследование границ и природы *NP*-полных классов задач было одним из наиболее активных и важных направлений в информатике за последнее десятилетие.

Кук хорошо известен благодаря своим результатам, оказавшим большое влияние на фундаментальные области информатики. Он внес существенный вклад в теорию сложности, соотношения затрат времени и объема памяти при вычислениях и в логику языков программирования. Его работы характеризуются элегантностью и интуицией и высвечивают истинную природу вычислений».

В течение 1970—1979 гг. Кук интенсивно работал при поддержке Национального исследовательского совета. В 1977—1979 гг. он получал стипендию имени Э. У. Р. Стеси. Автор многочисленных основополагающих статей, в настоящее время он занят доказательством того, что не существует «хорошего» алгоритма для *NP*-полных задач.

Тьюринговская премия ACM учреждена в память А. М. Тьюринга, английского математика, сделавшего крупный вклад в информатику.

Представлен исторический обзор сложности вычислений. Акцент сделан на два фундаментальных вопроса — определение внутренней вычислительной сложности задачи и доказательство верхних и нижних оценок сложности задач. Обсуждаются вероятностные и параллельные вычисления.

Это вторая лекция лауреата премии Тьюринга по вычислительной сложности. Первая была прочитана Микаэлем Раби-

ном в 1976 году[1]. Сейчас при чтении отличной статьи Рабина [62] мое внимание привлекает то обстоятельство, насколько возросла с тех пор активность в данной области. В этом кратком обзоре я хочу упомянуть наиболее важные и интересные на мой взгляд результаты по сложности вычислений с начала этих исследований примерно в 1960 г. В столь обширной области выбор тем неизбежно индивидуален, однако я надеюсь, что включенные в обзор статьи по любым стандартам являются фундаментальными.

1. РАННИЕ РАБОТЫ

Предыстория вопроса восходит к Алану Тьюрингу. В своей работе 1937 г. «О вычислимых числах с приложением к проблеме разрешения» [85] Тьюринг ввел свою знаменитую машину, которая обеспечила наиболее убедительную (по тому времени) формализацию понятия эффективно (или алгоритмически) вычислимой функции. Как только это понятие было точно сформулировано, стали возможными доказательства невыполнимости вычислений для компьютеров. В той же статье Тьюринг доказал, что никакой алгоритм (т. е. машина Тьюринга) не может по произвольной формуле исчисления предикатов решить за конечное число шагов, выполнима ли эта формула.

После того как теория, объяснявшая, какие задачи могут и какие не могут быть решены компьютером, стала хорошо разработанной, было естественно задаться вопросом о сравнительной вычислительной сложности вычислимых функций. Этим и занимается теория сложности вычислений. Рабин [59, 60] был одним из первых (1960), кто в явном виде сформулировал следующий вопрос: что означает, что f труднее вычислить, чем g? Рабин предложил аксиоматический подход, заложивший основы абстрактной теории сложности, развитой Блюмом [6] и другими.

Вторая важная ранняя (1965) работа — статья Хартманиса и Стирнса «О вычислительной сложности алгоритмов» [37][2]. Эта статья стала широко известна и дала название описываемой области исследований. Было введено важное понятие меры сложности как времени вычислений на многоленточных машинах Тьюринга, и были доказаны теоремы об иерархии. Эта статья также поставила один интригующий вопрос, все еще остающийся открытым. А именно, вычислимо ли за реальное время любое алгебраическое иррациональное число (вроде $\sqrt{2}$), т. е. сущест-

[1] Микаэль Рабин и Дана Скотт удостоены Тьюринговской премии 1976 г.
[2] Интересные воспоминания см. у Хартманиса [36].

вует ли машина Тьюринга, печатающая десятичное разложение этого числа со скоростью один разряд за 100 шагов?

Третья основополагающая статья (1965) — «Внутренняя вычислительная трудность функций» Алана Кобэма [15]. Кобэм подчеркивает слово «внутренняя» (intrinsic), т. е. его интересует машинно-независимая теория. Он спрашивает, сложнее ли умножение, чем сложение, и считает, что на этот вопрос невозможно ответить, покуда соответствующая теория не будет полностью развита. Кобэм также определил и охарактеризовал важный класс функций, который он обозначил $\mathscr{L}$: это те функции натуральных чисел, которые вычислимы за время, ограниченное полиномом от десятичной длины входа.

Три другие статьи, повлиявшие на упомянутых авторов, так же как и на других исследователей теории сложности (включая меня), — статьи Ямады [91], Беннета [4] и Ритчи [66]. Интересно отметить, что Рабин, Стирнс, Беннет и Ритчи — все были студентами в Принстоне примерно в одно и то же время.

2. РАННИЕ РЕЗУЛЬТАТЫ И ПОНЯТИЯ

Некоторые из авторов ранних работ задавались вопросом: что же на самом деле следует считать мерой сложности? Большинство выбирало время вычисления или объем памяти как наиболее очевидные характеристики сложности, но не были убеждены, что эти характеристики единственные или правильные. Например, Кобэм [15] отмечал: «...некоторая мера, связанная с физическим понятием работы, возможно, приведет к наиболее удовлетворительному анализу». Рабин [60] ввел аксиомы, которым должна удовлетворять мера сложности. Опираясь на двадцатилетний опыт, я теперь считаю очевидным, что время и объем памяти — особенно время — находятся в ряду наиболее существенных мер сложности. Представляется, что при оценке эффективности алгоритма в первую очередь принимается в расчет время его выполнения. Однако в последнее время становится ясным, что параллельное время и размеры оборудования — также важные меры сложности (см. разд. 6).

Другая важная мера сложности, восходящая в каком-то смысле еще к Шеннону [74] (1949), — сложность булевой схемы (или комбинационная). Здесь удобно предположить, что рассматриваемая функция f преобразует конечные цепочки бит в конечные цепочки бит, и сложность $C(n)$ функции f — размер наименьшей булевой схемы, вычисляющей f для всех входных наборов длины n. Эта очень естественная мера тесно связана с временем вычисления (см. [57 a], [57 b], [68 b]) и имеет свою хорошо развитую теорию (см. Савидж [68 a]).

Другой вопрос, поставленный Кобэмом [15], состоит в том,

что же представляет собой «шаг» в вычислении. Это равносильно вопросу, что именно является правильной моделью компьютера для измерения времени работы алгоритма. Обычно в литературе используют многоленточные машины Тьюринга, но им присущи искусственные ограничения с точки зрения эффективной реализации алгоритмов. Например, не существует разумного объяснения, почему память должна быть организована линейно. Почему бы не плоские массивы деревьев? Почему бы не допустить память с произвольным доступом?

Фактически с 1960 г. было предложено довольно много моделей компьютеров. Поскольку в реальных компьютерах память с произвольным доступом (RAM) имеется, представляется естественным допустить ее и в модели. Но как именно сделать это — вопрос не очевидный. Если машина может запомнить целые числа за один шаг, должны быть введены ограничения на их величину. (Если число 2 возвести 100 раз в квадрат, то результат имеет 2^{100} бит, которые не уместятся во все существующие средства памяти.) В [19] я предложил платную RAM, в которой плата (число шагов) размером примерно $\log|x|$ взимается всякий раз, когда x запоминается или извлекается из памяти. Этот подход работает, но не является вполне убедительным. Более популярная модель произвольного доступа та, что использовалась Ахо, Хопкрофтом и Ульманом [3], в которой каждая операция над целым числом имеет единичную стоимость, но не допускается, чтобы целые числа становились неразумно большими (например, их величина может быть ограничена некоторым фиксированным полиномом, зависящим от размера входа). Вероятно, с математической точки зрения наиболее удовлетворительной является машина с модификацией памяти Шёнхаге [69], которую можно рассматривать либо как машину Тьюринга, строящую свою собственную структуру памяти, либо RAM с единичной стоимостью, которая за один шаг может только копировать, прибавлять (или вычитать) единицу или запоминать (или извлекать из памяти). Машина Шёнхаге — это слегка обобщенная машина Колмогорова — Успенского [46], предложенная гораздо раньше (1958), и мне кажется, что это наиболее общая машина, которая за один шаг выполняет ограниченную работу. Беда в том, что она, вероятно, слишком мощная. (См. разд. 3 «Умножение больших чисел».)

Возвращаясь к вопросу Кобэма «что есть шаг», я думаю, что за последние 20 лет стало ясным, что единого четкого ответа на этот вопрос нет. К счастью, конкурирующие компьютерные модели не слишком сильно отличаются временем вычислений. Вообще говоря, каждая из них может моделировать любую другую самое большее с квадратичным увеличением объема вычислений (некоторые из первых аргументов в пользу этого

приведены в [37]). Что касается широко распространенных (основных) моделей произвольного доступа, то они отличаются только множителем, логарифмически зависящим от времени вычислений.

Это приводит к последнему важному понятию, выработанному к 1965 г., — идентификации класса задач, разрешимых за время, ограниченное полиномом от длины входа. Различие между полиномиальными и экспоненциальными алгоритмами было проведено еще в 1953 г. фон Нейманом [90]. Однако этот класс не был определен формально и не изучался, покуда Кобэм [15] не ввел в 1964 г. класс $\mathscr{L}$ функций. Кобэм указал, что этот класс определен корректно независимо от того, какая компьютерная модель выбрана, и охарактеризовал его в духе теории рекурсивных функций. Мысль о том, что вычислимость за полиномиальное время, грубо говоря, соответствует практической выполнимости (tractability), была впервые высказана в печати Эдмондсом [27], который назвал полиномиальные алгоритмы «хорошими алгоритмами». Принятое теперь стандартное обозначение P для класса множеств цепочек, распознаваемых за полиномиальное время, было позднее введено Карпом [42].

Отождествление P с практически выполнимыми (или осуществимыми) задачами стало общепринятым с начала 1970-х. Не является непосредственно очевидным, почему это должно быть истинным, так как алгоритм, время выполнения которого есть полином n^{1000}, конечно, неосуществим, и наоборот, алгоритм, время выполнения которого экспоненциально зависит от n по формуле $2^{0{,}00001n}$, практически осуществим. Представляется *эмпирическим* фактом, однако, что для естественно возникающих задач нет оптимальных алгоритмов с такими временами работы[1]. Самый известный практический алгоритм с экспоненциальным временем работы в худшем случае — симплексный алгоритм линейного программирования. Смэйл [75, 76] попытался объяснить это, показав, что в некотором смысле среднее время работы невелико, но также важно отметить, что Хачиян [43] показал принадлежность линейного программирования классу P, воспользовавшись другим алгоритмом. Таким образом, наш общий тезис, что класс P эквивалентен классу практически осуществимых задач, не нарушен.

3. ВЕРХНИЕ ОЦЕНКИ ВРЕМЕНИ

Значительная доля исследований в информатике состоит из построения и анализа огромного числа эффективных алгорит-

[1] См. [31], с. 6—9, где это обсуждается.

мов. Наиболее важные алгоритмы (с точки зрения вычислительной сложности) должны быть в некотором смысле особенными; они в основном дают удивительно быстрый способ решения какой-нибудь простой или важной задачи. Ниже я перечисляю некоторые из наиболее интересных алгоритмов, изобретенных начиная с 1960 г. (Кстати, интересно порассуждать о том, какие именно алгоритмы признаны за это время наиболее важными. Бесспорно, арифметические операции $+$, $-$, $*$, $\div$ над десятичными числами являются основными. После этого, я думаю, следующими кандидатами являются быстрая сортировка и поиск, исключение по Гауссу, алгоритм Эвклида и симплексный алгоритм.)

Параметр n относится к размеру входа, а временные оценки суть оценки времени в худшем случае и относятся к многоленточной машине Тьюринга (или любой разумной машине с произвольным доступом к памяти), за исключением случаев, где это будет специально оговариваться.

(1) **Быстрое преобразование Фурье** [23], требующее $O(n \log n)$ арифметических операций, — один из наиболее часто используемых алгоритмов в научных вычислениях.

(2) **Умножение больших чисел.** Школьный метод требует $O(n^2)$ битовых операций для умножения двух n-разрядных чисел. В 1962 г. Карацуба и Офман [41] опубликовали метод, требующий всего $O(n^{1,59})$ шагов. Вскоре после этого Тоом [84] показал, как построить булеву схему сложности $O(n^{1+\varepsilon})$ для произвольно малого $\varepsilon > 0$, выполняющую умножение. Я был в это время студентом старшего курса в Гарварде, и, вдохновленный вопросом Кобэма: «Сложнее ли умножение, чем сложение?», наивно пытался доказать, что умножение требует $\Omega(n^2)$ шагов на многоленточной машине Тьюринга. Статья Тоома сильно удивила меня. С помощью Стола Аандераа [22] я показал, что умножение требует $\Omega(n \log n/(\log \log n)^2)$ шагов машины Тьюринга[1]) при вычислении «on line». Я отметил в своей диссертации, что метод Тоома может быть приспособлен к многоленточным машинам Тьюринга, с тем чтобы выполнить умножение за $O(n^{1+\varepsilon})$ шагов, — то, что, я уверен. Тоома не удивило бы.

В настоящий момент асимптотически наилучшее время выполнения умножения чисел на многоленточной машине Тьюринга есть $O(n \log n \log \log n)$ — результат, полученный Шёнхаге и Штрассеном [70] (1971) с использованием быстрого преобразования Фурье. Однако Шёнхаге [69] недавно показал посредством сложного рассуждения, что его машины с модификацией памяти (см. разд. 2) могут умножать за $O(n)$ (линейное

[1]) Эта нижняя оценка была слегка улучшена. См. [56] и [64].

время!). Мы вынуждены заключить, что либо умножение проще, чем мы думали, либо машины Шёнхаге «жульничают».

(3) **Умножение матриц.** Очевидный метод требует $n^2(2n-1)$ арифметических операций для умножения двух матриц размера $n \times n$, и в 1950—60-е годы пытались доказать, что этот метод оптимальный. Неожиданно Штрассен [81] (1969) опубликовал свой метод, требующий только $4{,}7\,n^{2{,}81}$ операций. Серьезные усилия были приложены к уменьшению показателя 2,81, и сейчас наилучшее известное время $O(n^{2{,}496})$ операций — результат, принадлежащий Коперсмиту и Винограду [24]. Здесь еще большой простор для улучшения оценки, потому что наилучшая известная нижняя оценка $2n^2-1$ (см. [13]).

(4) **Максимальные паросочетания в общих неориентированных графах.** Возможно, это была первая среди принадлежащих P задач, для которой было явно показано, что алгоритм, принадлежащий классу P, трудный. В оказавшей заметное влияние статье Эдмондса [27] излагался результат и обсуждалось понятие алгоритма, выполнимого за полиномиальное время (см. разд. 2). Он также отметил, что простое понятие расширяющегося пути, которого достаточно для двудольного случая, не годится для неориентированных графов общего вида.

(5) **Распознавание простых чисел.** Основной вопрос здесь: принадлежит ли эта задача классу P? Другими словами, существует ли алгоритм, который всегда сообщает нам, является ли произвольное n-разрядное число на входе простым, и останавливается после числа шагов, ограниченных фиксированным полиномом от n? Гэри Миллер [53] (1976) показал, что такой алгоритм существует, но его доказательство опирается на расширенную гипотезу Римана. Соловей и Штрассен [77] построили быстрый алгоритм Монте-Карло (см. разд. 5) для распознавания простого числа, но если входное число составное, то с малой вероятностью алгоритм может ошибочно распознать его как простое. Наилучший из доказуемых детерминированных алгоритмов принадлежит Адлеману, Померанцу и Рамли [2], он требует времени $n^{O(\log\log n)}$, что чуть хуже полиномиального. Его вариацией является алгоритм Коэна и Ленстры мл. [17], который может, как правило, обрабатывать числа до 100 десятичных разрядов примерно за 45 с.

Недавно показана принадлежность классу P трех важных задач. Первая — линейное программирование, для которого это было доказано Хачияном [43] в 1979 г. (Изложение можно найти, например, в [55].) Для второй — распознавания изоморфизма двух графов степени не более d — вопрос решен Лаксом [50] в 1980 г. (Алгоритм полиномиален относительно числа вершин при фиксированном d, но экспоненциален относительно d.) Третья — разложение на множители полиномов

с рациональными коэффициентами. Это было показано для полиномов от одной переменной Ленстрой, Ленстрой и Ловасом [48] в 1982 г. Как показывает результат Калтофена [39, 40], это можно обобщить на полином от любого фиксированного числа переменных.

4. НИЖНИЕ ОЦЕНКИ

Самая серьезная проблема в теории сложности, отделяющая эту теорию от анализа алгоритмов, — доказательство нижних оценок сложности отдельных задач. Есть нечто вызывающее большое удовлетворение, когда удается доказать, что задача, требующая ответа да—нет, не может быть решена за n или n^2 или 2^n шагов независимо от того, какой алгоритм применяется для ее решения. В доказательстве нижних оценок удалось многого достичь, но остались открытыми еще более важные и обескураживающие вопросы.

Все важные нижние оценки времени вычислений и объема памяти основаны на «диагональных рассуждениях». Диагональные рассуждения использовались Тьюрингом и его современниками для доказательства того, что некоторые задачи алгоритмически неразрешимы. Они были также использованы еще до 1960-х годов для определения иерархии вычислимых 0—1 функций[1]. В 1960 г. Рабин [60] доказал, что для всякой разумной меры сложности, такой, как время вычисления или объем памяти, достаточное увеличение допустимого времени или объема памяти всегда позволяет вычислять большее число 0—1 функций. Примерно в то же время Ритчи в своей диссертации [65] определил специальную иерархию функций (которая, как он показал, нетривиальна для 0—1 функций) в терминах объема доступной памяти. Несколько позднее результат Рабина был отчасти улучшен для требуемого времени на многоленточных машинах Тьюринга Хартманисом и Стирнсом [37] и для объема памяти — Стирнсом, Хартманисом и Льюисом [78].

4.1. ДОКАЗАТЕЛЬСТВО ПРАКТИЧЕСКОЙ НЕОСУЩЕСТВИМОСТИ ЕСТЕСТВЕННЫХ РАЗРЕШИМЫХ ЗАДАЧ

Упомянутые выше результаты по иерархиям дают нижние оценки для времени и памяти, необходимых для вычисления конкретных функций, но все такие функции кажутся «надуманными». Например, легко видеть, что функция $f(x, y)$, которая дает первый разряд на выходе машины x при входе y после

[1] См., например, Гжегорчик [35].

$(|x|+|y|)^2$ шагов, не может быть вычислена за время $(|x|+|y|)^2$. Только в 1972 г., когда Альберт Мейер и Ларри Стокмейер [52] доказали, что проблема эквивалентности для регулярных выражений с возведением в квадрат требует экспоненциальной памяти и, следовательно, экспоненциального времени, была найдена нетривиальная нижняя оценка для общих моделей вычислений для «естественной» задачи (естественной в смысле интересной и имеющей смысл независимо от вычислительных машин). Вскоре после этого Мейер [51] нашел очень сильную нижнюю оценку времени, требуемого для определения истинности формул в некоторой формально разрешимой теории, именуемой WSIS (слабая монадическая теория следования второго порядка). Он доказал, что никакой компьютер, время работы которого ограничено фиксированным числом экспонент (2^n, 2^{2^n}, $2^{2^{2^n}}$ и т. д.), не может правильно разрешить WSIS. Аспирант Мейера Стокмейер вычислил [79], что всякая булева схема (считай, компьютер), которая корректно решает задачу об истинности произвольной WSIS формулы длины 616 символов, должна иметь более чем 10^{123} элементов. Число 10^{123} было выбрано как число протонов, которые могут поместиться в известной части Вселенной. Это очень убедительное доказательство практической неосуществимости!

После Мейера и Стокмейера было получено большое число нижних оценок сложности разрешимых формальных теорий (обзоры см. [29, 80]). Одной из наиболее интересных является нижняя оценка в форме двойной экспоненты для времени, требуемого для разрешения пресбургеровской арифметики (теория натуральных чисел со сложением), полученная в работе Фишера и Рабина [30]. Это недалеко от наилучшей известной верхней оценки времени для этой теории, которая представляет собой тройную экспоненту [54]. Наилучшая верхняя оценка для памяти — двойная экспонента [29].

Несмотря на перечисленные успехи, отметим, что список доказанных нижних оценок в задачах меньшей сложности шокирует. Фактически неизвестны нелинейные нижние оценки времени для общего вида вычислительных моделей ни для какой естественной задачи класса NP (см. разд. 4.4), в частности, ни для одной из 300 проблем, перечисленных в [31]. Конечно, можно доказать из диагональных соображений существование задач класса NP, требующих времени n^k для любого фиксированного k. Однако для нижних оценок памяти мы даже не знаем, как доказывать существование NP-задач, не решаемых с памятью $O(\log n)$ для off-line машин Тьюринга (см. разд. 4.3). И это несмотря на то, что наиболее известные верхние оценки объема памяти для многих естественных случаев по существу линейны относительно n.

4.2. СТРУКТУРИРОВАННЫЕ НИЖНИЕ ОЦЕНКИ

Несмотря на то что успехи в доказательстве интересных нижних оценок для конкретных задач на общих вычислительных моделях невелики, для «структурированных» моделей такие результаты есть. Термин «структурированный» был введен Бородиным [9] для компьютеров с ограниченным набором операций, подходящим для рассматриваемой задачи. Простой пример этого — задача сортировки n чисел. Можно доказать (см. [44]) без особых затруднений, что это требует по крайней мере $n \log n$ сравнений при условии, что единственная допустимая операция компьютера над выходами — попарное сравнение входов. Эта нижняя оценка ничего не говорит о машинах Тьюринга или булевых схемах, но она была распространена на машины с единичной стоимостью обращения к памяти с произвольным доступом в предположении, что деление запрещено.

Вторая и очень элегантная структурированная нижняя оценка, полученная Штрассеном [82] (1973), устанавливает, что полиномиальная интерполяция, т. е. нахождение коэффициентов полинома степени $n-1$, проходящего через n заданных точек, требует в предположении, что допустимы только арифметические операции $\Omega(n \log n)$ умножений. Интересно это отчасти потому, что первоначальное доказательство Штрассена основано на теореме Безу, глубоком результате алгебраической геометрии. Совсем недавно Баур и Штрассен [83] усилили нижнюю оценку, показав, что даже средний коэффициент интерполяционного полинома по n точкам требует для своего вычисления $\Omega(n \log n)$ умножений.

Привлекательность всех этих структурированных результатов состоит в том, что нижние оценки близки к наилучшим известным верхним оценкам[1)] и наилучшие известные алгоритмы могут быть реализованы на структурированных моделях, к которым относятся нижние оценки. (Отметим, что корневая сортировка, про которую иногда говорят, что она требует линейного времени, на самом деле требует по крайней мере $n \log n$ шагов, если предположить, что числа на входах имеют достаточное число разрядов, чтобы все они могли быть различными.)

4.3. НИЖНИЕ ОЦЕНКИ ПРОИЗВЕДЕНИЯ ВРЕМЕНИ И ПАМЯТИ

Другой путь обойти трудности доказательства «абсолютных» нижних оценок времени и объема памяти состоит в доказательстве нижних оценок времени в предположении малой памяти. Кобэм [16] доказал первый такой результат

[1)] Верхние оценки для интерполяции см. у Бородина и Манро [12].

в 1966 г., когда он показал, что произведение времени и памяти для распознавания n-разрядного совершенного квадрата на off-line машине Тьюринга должно быть $\Omega(n^2)$. (То же самое верно для n-символьных палиндромов.) Здесь входное слово записано на двухдорожечной входной ленте, с которой возможно только чтение, и используемая память по определению — число квадратов, просмотренных рабочими лентами, имеющимися в данной машине Тьюринга. Так, если, например, память ограничена $O(\log^3 n)$ (чего более чем достаточно), то время должно быть $\Omega(n^2/\log^3 n)$ шагов.

Слабость результата Кобэма состоит в том, что, хотя off-line машина Тьюринга разумна для измерения времени вычислений и памяти в отдельности, эта модель слишком ограничительна, когда память и время рассматриваются вместе. Например, ясно, что палиндромы могут быть распознаны за $2n$ шагов с постоянной зоной, если две головки осуществляют совместный просмотр входной ленты. Бородин и я [10] частично исправили эту слабость, когда мы доказали, что сортировка n целых чисел по возрастанию в диапазоне от 1 до n^2 требует произведения времени и памяти $\Omega(n^2/\log n)$. Доказательство применимо к любой «последовательной машине общего вида», что включает off-line машину Тьюринга со многими входными головками или даже с произвольным доступом к входной ленте. К сожалению, в нашем доказательстве очень важный момент состоит в том, что сортировка требует много выходных битов, и остается открытым вопрос: можно ли подобную нижнюю оценку получить для какой-нибудь задачи распознавания множества, такой, как распознавание того, все ли n входных чисел различны? (Наша нижняя оценка для сортировки недавно слегка улучшена в [64].)

4.4. NP-ПОЛНОТА

Теория NP-полноты, конечно, самое значительное достижение в теории сложности вычислений. Я не буду здесь на ней останавливаться, потому что это теперь хорошо известно и описано в учебниках. В частности, книга Гэри и Джонсона [31] — прекрасный источник сведений по этому вопросу.

Класс NP состоит из всех множеств, распознаваемых за полиномиальное время на недетерминированной машине Тьюринга. Насколько мне известно, впервые математически эквивалентный класс был определен Джеймсом Беннетом в его докторской диссертации [4] в 1962 г. Беннет использовал для этого класса название «расширенные положительные рудиментарные отношения», и в его определении использованы логические кванторы вместо вычислительных машин. Я прочитал эту часть его дис-

сертации и осознал, что его класс может быть охарактеризован как то, что сейчас называют *NP*. Я использовал обозначение $\mathscr{L}^+$ (вслед за классом $\mathscr{L}$ Кобэма) в моей статье [18] в 1971 г., а Карп дал ныне принятое название *NP* для этого класса в своей статье [42] в 1972 г. Тем временем совершенно независимо Эдмондс еще в 1965 г. [28] рассуждал неформально о задачах с «хорошими характеристиками» — понятии, по существу эквивалентном *NP*.

В 1971 г. [18] я ввел понятие *NP*-полноты и доказал, что 3-выполнимость и задача о подграфе *NP*-полны. Год спустя Карп [42] доказал *NP*-полноту 21 задачи, продемонстрировав тем самым важность этого предмета. Независимо от этого и чуть-чуть позднее Леонид Левин [49] в Советском Союзе (теперь в Бостонском университете) определил подобное (и более сильное) понятие и доказал про шесть задач, что они полны в этом смысле. Неформальное понятие «переборной задачи» было стандартным в советской литературе, и Левин назвал свои задачи «универсальными переборными задачами».

Класс *NP* включает огромное количество практических задач, появляющихся в бизнесе и промышленности (см. [31]). Доказательство того, что *NP*-задача *NP*-полна, есть доказательство того, что задача не принадлежит классу *P* (не имеет детерминированного алгоритма с полиномиальным временем), разве что всякая *NP*-задача принадлежит классу *P*. Поскольку последнее условие революционизировало бы информатику практический результат *NP*-полноты — это нижняя оценка. Это и явилось причиной того, что я включил этот предмет в раздел о нижних оценках.

4.5. #P-ПОЛНОТА

Понятие *NP*-полноты применимо к множествам, и доказательство того, что множество *NP*-полно, обычно интерпретируется как доказательство его необозримости. Есть, однако, большое количество очевидно невычислимых функций, для которых, по-видимому, нельзя найти никакого доказательства *NP*-полноты, к ним применимого. Лесли Валиант [86, 87], чтобы выйти из положения, ввел понятие $\#P$-полноты. Доказательство того, что функция $\#P$-полна, показывает, что она практически не поддается вычислению, тем же самым способом, что и доказательство *NP*-полноты множества показывает, что оно практически необозримо для распознавания; а именно, если $\#P$-полная функция вычислима за полиномиальное время, то $P=NP$.

Валиант дал много примеров $\#P$-полноты функций, но, возможно, наиболее интересный из них — перманент целочисленной матрицы. Перманент определяется формально очень похоже на

детерминант, но в то время как детерминант легко вычисляется посредством гауссова метода исключения, все многочисленные попытки за последние сто с лишним лет найти подходящий метод вычисления перманента были безуспешными. Валиант дал первое убедительное объяснение этой безуспешности, когда доказал, что перманент $\#P$-полон.

5. ВЕРОЯТНОСТНЫЕ АЛГОРИТМЫ

Использование случайных чисел для моделирования или аппроксимации случайных процессов вполне естественно и стало общепринятым в вычислительной практике. Однако идея о том, что случайные входы могут быть полезны в решении детерминированных комбинаторных задач, гораздо медленнее проникала в сообщество специалистов по информатике. Здесь я ограничусь вероятностными (бросание монеты) алгоритмами с полиномиальным временем, которые «решают» (в некотором разумном смысле) задачу, для которой детерминированные алгоритмы с полиномиальным временем неизвестны.

Первым таким алгоритмом, кажется, был алгоритм Берлекэмпа [5], 1970 г., для разложения на множители полинома f над полем $GF(p)$ с p элементами. Алгоритм Берлекэмпа работает полиномиальное время от степени f и $\log p$ и с вероятностью по крайней мере 1/2 находит неприводимые разложения на сомножители f; в противном случае его работа заканчивается неудачей. Так как алгоритм можно повторять любое число раз и неудачные исходы все независимы, на практике алгоритм всегда разлагает полином за приемлемое (разумное) время.

Более сильный пример — алгоритм распознавания простых чисел Соловея и Штрассена [77], предложенный в 1974 г. Этот алгоритм работает полиномиальное время от длины входа m и выдает результат «простое» или «составное». Если m и в самом деле простое, то на выходе определенно «простое», но если m составное, то с вероятностью самое большее половина ответ может также оказаться «простое». Алгоритм с m на входе можно повторять любое число раз, и все результаты будут независимы от предшествующих попыток. Значит, если ответ всегда «составное», то пользователь знает, что m составное: если результат «простое» неизменно выдается после, скажем, 100 испытаний, то пользователь имеет веские основания считать, что m простое, так как всякое фиксированное составное m давало бы такой результат с очень малой вероятностью (менее чем 2^{-100}).

Рабин [61] разработал еще один вероятностный алгоритм со свойствами, подобными вышеупомянутым, и нашел его очень быстрым при пробах на компьютере. Число $2^{400}-593$ было рас-

познано как простое (с высокой вероятностью) за несколько минут.

Одно из интересных приложений вероятностных проверок чисел на простоту было предложено Райвестом, Шамиром и Адлеманом [67а] в их основополагающей статье по криптосистемам общего доступа в 1978 г. Их система требует порождения больших (100-значных) случайных простых чисел. Они предложили проверять случайные 100-значные числа, используя метод Соловея—Штрассена, покуда этот метод не обнаружит «вероятностно» простое число в описанном выше смысле. На самом деле, с помощью нового мощного детерминированного метода проверки чисел на простоту Коэна и Ленстры [17], упомянутого в разд. 3, после обнаружения случайного 100-значного «вероятностного простого числа» его можно проверить примерно за 45 с., если важно знать наверняка.

Класс множеств, распознаваемых вероятностными алгоритмами с полиномиальным временем работы в смысле Соловея и Штрассена, известен в литературе как R (или иногда как RP). Таким образом, множество принадлежит классу R тогда и только тогда, когда для него существует вероятностный алгоритм распознавания, срабатывающий всегда за полиномиальное время и никогда не ошибающийся, если вход не принадлежит R, и для каждого входа из R на его выходе правильный ответ для каждого прогона алгоритма появляется с вероятностью по крайней мере 1/2. Следовательно, множество составных чисел принадлежит R, и вообще $P \subseteq R \subseteq NP$. Существуют другие интересные примеры множеств из R, про которые неизвестно, принадлежат ли они классу P. Например, Шварц [71] показал, что множество невырожденных матриц, элементы которых — полиномы от многих переменных, принадлежит R. Алгоритм вычисляет полиномы для случайных небольших целочисленных значений переменных и вычисляет детерминант результата. (Детерминант, по-видимому, не может быть обозримым образом вычислен непосредственно, потому что вычисляемые полиномы имели бы в общем случае экспоненциально много членов.)

Любопытен открытый пока вопрос: верно ли, что $R=P$? Заманчиво ответить на него «да», исходя из философских соображений, что случайное бросание монеты не может принести много пользы, когда ответ должен быть определенным — да или нет. С ним связан другой вопрос: является ли вероятностный алгоритм (показывающий принадлежность задачи классу R) для всех практических целей столь же хорошим, что и детерминированный алгоритм? В конце концов, вероятностные алгоритмы можно выполнять, используя генераторы псевдослучайных чисел, имеющиеся для большинства компьютеров, и вероятность ошибки 2^{-100} пренебрежимо мала. Загвоздка в том, что

генераторы псевдослучайных чисел не генерируют на самом деле случайные числа, и непонятно, насколько хорошо они будут работать для данного вероятностного алгоритма. Практический опыт показывает, что они работают, видимо, хорошо. Но если они *всегда* работают хорошо, то, следовательно, $R=P$, потому что псевдослучайные числа порождаются детерминированно, так что подлинная случайность в конце концов не нужна. Другая возможность состоит в использовании физического процесса, такого, как тепловой шум, для порождения случайных чисел. Но насколько реальной может быть случайность в природе — это открытый вопрос философии науки.

Позвольте мне завершить этот раздел упоминанием интересной теоремы Адлемана [1] о классе R. Легко видеть [57b], что если множество принадлежит классу P, то для каждого n существует булева схема размера, ограниченного полиномом от n, которая определяет, всегда ли произвольная последовательность символов длины n принадлежит этому множеству. Адлеман доказал, что то же самое истинно для класса R. Так, например, для каждого n существует малая «схема вычисления» (компьютерная схема), которая корректно и быстро распознает простоту n-разрядных чисел. Загвоздка в том, что схемы не регулярны по n, и, возможно, построение схемы для 100-разрядных чисел может оказаться неразрешимой задачей[1]).

6. СИНХРОННЫЕ ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ

С появлением VLSI (СБИС) технологии, в которой один или более процессоров могут быть помещены на четвертьдюймовый чип, естественно представить себе в будущем устройство, составленное из многих тысяч таких процессоров, совместно работающих параллельно для решения одной задачи. Хотя никакой очень большой машины общего назначения (универсальной) такого рода еще не построено, такие замыслы существуют (см. Шварц [72]). Это мотивирует развитие в последнее время очень привлекательной ветви вычислительной сложности: теории крупномасштабных параллельных синхронных вычислений, в которых число процессоров есть ресурс, ограниченный параметром $H(n)$ (H — от hardware) таким же образом, что в последовательной теории сложности ограничение памяти параметром $S(n)$. Обычно $H(n)$ — фиксированный полином от n.

Было предложено немало моделей параллельных вычислений (обзор см. [21]), так же как существует и много конкурирующих последовательных моделей (см. разд. 2). Однако среди них есть две наиболее предпочтительные. Первая — класс моделей

[1]) Подробнее теорию вероятностных вычислений см. Гилл [32].

с общей памятью, в которой большое число процессоров связано посредством памяти с произвольным доступом, которая объединяет их. Для таких моделей опубликовано много параллельных алгоритмов, так как реальные параллельные машины, когда они будут построены, могут быть очень похожими на них. Однако для математической теории эти модели не вполне удовлетворительны, потому что слишком многие подробности их архитектуры произвольны. Как разрешить конфликты в общей памяти при чтении и записи? Какие базисные операции допустимы для каждого процессора? Следует ли одно обращение к памяти оценивать в $\log H(n)$ единиц времени?

Поэтому я предпочитаю более чистую модель, обсуждаемую Бородиным [8] (1977), в которой параллельный компьютер есть однородное семейство $\langle B_n \rangle$ ациклических булевых схем, такое, что B_n имеет n входов (и, следовательно, столько же входных наборов длины n). Тогда $H(n)$ (аппаратная сложность) есть просто число элементов в B_n, а $T(n)$, время параллельного вычисления, есть глубина схемы B_n (т. е. длина длиннейшего пути от входа к выходу). Практическое обоснование этой модели состоит в том, что, вероятно, все реальные машины (включая машины с общей памятью) построены из булевых схем. Кроме того, определение минимальной булевой сложности и глубины, необходимых для вычисления функции, есть естественная математическая задача, и она рассматривалась задолго до возникновения теории параллельных вычислений.

К счастью для этой теории, минимальные значения аппаратной сложности $H(n)$ и параллельного времени $T(n)$ не слишком сильно различаются для различных конкурирующих моделей параллельных компьютеров. В частности, имеет место интересный общий факт, верный для всех моделей, доказанный впервые для одной конкретной модели Праттом и Стокмейером [58] (1974) и названный в [33] «тезисом о параллельных вычислениях», а именно, задача может быть решена за полиномиальное время относительно $T(n)$ параллельной машиной (с неограниченной аппаратной сложностью) тогда и только тогда, когда она может быть решена с полиномиальной памятью относительно $T(n)$ последовательной машиной (с неограниченным временем).

Один из основных вопросов параллельных вычислений состоит в следующем: какие задачи могут быть решены с использованием многих процессоров существенно быстрее, чем с одним процессором? Николас Пиппенджер [57а] формализовал этот вопрос, определив класс (ныне называемый NC, «Nick's class») задач, решаемых сверхбыстро (время $T(n) = (\log n)^{O(1)}$) параллельным компьютером с практически прием-

лемой ($H(n)=n^{O(1)}$) аппаратной сложностью. К счастью, класс NC остается одним и тем же независимо от конкретной выбранной модели параллельного компьютера, и легко видеть, что NC — подкласс класса FP функций, вычислимых последовательно за полиномиальное время. Наш неформальный вопрос можно теперь сформулировать так: какие задачи из FP принадлежат также NC?

Мыслимо (хотя и маловероятно), что $NC=FP$, так как для доказательства $NC\neq FP$ потребовалось бы совершить прорыв в теории сложности (см. конец разд. 4.1). Поскольку мы не знаем, как доказать, что функция f, принадлежащая FP, не принадлежит NC, лучше всего попытаться доказать, что f логарифмически (по памяти) полна для FP. Это аналог доказательства NP-полноты задач, и практически это делает почти безнадежными попытки найти сверхбыстрые параллельные алгоритмы для f. Причина в том, что если f логарифмически (по памяти) полна для FP и f принадлежит NC, то $FP=NC$, что было бы большим сюрпризом.

Немалый прогресс достигнут в классификации задач из FP по принадлежности их NC или логарифмической (по памяти) полноте в FP (конечно, они могут быть не принадлежащими ни одному из этих классов). Первый пример задачи, полной в P, был предложен в 1973 г. мною в [20], хотя я не формулировал этот результат в терминах полноты. Вскоре после этого Джонс и Лаазер [38] определили это понятие полноты и привели около 5 примеров, включая проблему пустоты для контекстно-свободных грамматик. Возможно, простейшая задача, для которой доказана полнота в FP, есть так называемая проблема значения схемы [47]: для заданной булевой схемы и значений на ее входах найти значение на выходе. Наиболее интересным для меня является пример, принадлежащий Гольдшлягеру, Шоу и Стейплу [34], — нахождение (четности) максимального потока через данную сеть с (большими) положительными целочисленными пропускными способностями ее ребер. Он интересен тонкостью доказательства полноты. Наконец, я упомянул бы, что линейное программирование полно в FP. В этом случае трудная часть состоит в доказательстве того, что задача принадлежит P (см. [43]), после чего полнота устанавливается непосредственно [26].

Среди задач, принадлежность которых классу NC известна, — четыре арифметические операции ($+$, $-$, $*$, $\div$) над двоичными числами, сортировка, связность графа, матричные операции (умножение, обращение, детерминант, ранг), наибольший общий делитель полиномов, контекстно-свободные языки и нахождение в графе минимального остова (см. [11], [21], [63], [67]). Про размер максимального паросочетания для заданно-

го графа известно, что он принадлежит «случайному» *NC* (*NC*, в котором допускается бросание монеты), хотя вопрос о том, принадлежит ли хотя бы одному «случайному» *NC* задача нахождения фактического максимального паросочетания, пока не решен. Результаты [89] и [76b] предлагают общие методы доказательства принадлежности задач классу *NC*.

Наиболее интересной задачей в *FP*, про которую неизвестна ни полнота в *FP*, ни принадлежность случайному *NC*, является нахождение наибольшего общего делителя двух целых чисел. Есть много других интересных задач, которые еще должны быть классифицированы, включая нахождение максимального паросочетания или максимальной клики в графе (см. [88]).

7. БУДУЩЕЕ

Позвольте мне снова сказать, что область вычислительной сложности велика, а этот обзор краток. Существуют обширные разделы этой науки, которые я опустил вовсе или едва их коснулся. Приношу свои извинения исследователям в этих областях.

Один относительно новый и волнующий раздел, названный Яо [92] «вычислительной теорией информации», продолжает классическую шенновскую теорию информации, введя в рассмотрение информацию, доступную с помощью практически осуществимого вычисления. Эта тематика стала широко обсуждаться в основном после работ Диффи и Хелмана [25] и Райвеста, Шамира и Адлемана [67a] об общедоступных криптосистемах, хотя ее корни в теории вычислений восходят к Колмогорову [45] и Чайтину [14a], [14b], которые первыми придали смысл понятию «случайности» одной конечной последовательности, используя теорию вычислений. Интересная идея в этой теории, рассмотренная Шамиром [73] и Блюмом и Микали [7], касается порождения псевдослучайных последовательностей, в которых будущие разряды доказуемо трудно предсказать в терминах предшествующих. Яо [92] доказывает, что существование таких последовательностей имело бы положительные последствия в вопросе о детерминированной сложности вероятностного класса R (см. разд. 5). Фактически вычислительная теория информации обещает пролить свет на роль случайности в вычислении.

В добавление к вычислительной теории информации мы можем ожидать интересные результаты для вероятностных алгоритмов, параллельных вычислений и (при везении) нижних оценок. Что касается нижних оценок, единственный прорыв, на который, по-моему, можно надеяться в недалеком будущем, — это доказательство того, что не всякая задача из P решается со

сложностью (по памяти) $O(\log n)$, и, возможно, также того, что $P \neq NC$. Во всяком случае, активность в области вычислительной сложности остается очень высокой, и я с нетерпением жду, что же принесет будущее.

БЛАГОДАРНОСТИ

Я благодарен своим коллегам по теории сложности в Торонто за многие полезные замечания и предложения, особенно Аллану Вородину, Иоахиму фон цур Гатену, Сильвио Микали и Чарльзу Ракову.

ЛИТЕРАТУРА

1. Adleman L. Two theorems on random polynomial time. Proc. 19th Symp. on Foundations of Computer Science. IEEE Computer Society, Los Angeles (1978), 75—83.
2. Adleman L., Pomerance C., and Rumley R. S. On distinguishing prime numbers from composite numbers. Annals of Math. 117 (January 1983), 173—206.
3. Aho A. V., Hopcroft, J. E., and Ullman, J. D. The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, Mass., 1974. [Имеется перевод: Ахо А., Хопкрофт Д., Ульман Д. Построение и анализ вычислительных алгоритмов. — М.: Мир, 1979.]
4. Bennett J. H. On Spectra. Doctoral dissertation, Department of Mathematics, Princeton University, 1962.
5. Berlecamp E. R. Factoring polynomials over large finite fields. Math. Comp. 24 (1970), 713—735.
6. Blum M. A machine independent theory of the complexity of recursive functions. JACM 14, 2 (April 1967), 322—336. [Имеется перевод: Блюм М. Машинно-независимая теория сложности рекурсивных функций. В сб.: Проблемы математической логики. — М.: Мир, 1970, с. 401—422.]
7. Blum M., and Micali S. How to generate cryptographically strong sequences of pseudo random bits. Proc. 23rd IEEE Symp on Foundations of Computer Science. IEE Computer Society, Los Angeles (1982), 112—117.
8. Borodin A. On relating time and space to size and depth. SIAM J. Comp. 6 (1977), 733—744.
9. Borodin A. Structured vs. general models in computational complexity. In: Logic and Algorithmic. Monograpie no. 30 de L'Enseignement Mathematique Universite de Geneve, 1982.
10. Borodin A., and Cook S. A time-space tradeoff for sorting on a general sequential model of computation. SIAM J. Comput. 11 (1982), 287—297.
11. Borodin A., von zur Gathen, J., and Hopcroft, J. Fast parallel matrix and GCD computations. 23rd IEEE Symp. on Foundations of Computer Science. IEEE computer society, Los Angeles (1982), 65—71.
12. Borodin A., and Munro, I. The Computational Complexity of Algebraic and Numeric Problems. Elsevier, New York, 1975.
13. Brockett R. W., and Dobkin, D. On the optimal evaluation of a set of bilinear forms. Linear Algebra and its Applications 19 (1978), 207—235.
14a. Chaitin G. J. On the length of programs for computing finite binary sequences. JACM 13, 4 (October 1966), 547—569; JACM 16, 1 (October 1969), 145—159.
14b. Chaitin G. J. A theory of program size formally identical to informational theory. JACM 22, 3 (July 1975), 329—340.

15. Cobham A. The intrinsic computational difficulty of functions. Proc. 1964 International Congress for Logic, Methodology, and Philosophy of Sciences. Y. Bar-Hellel, Ed., North Holland, Amsterdam, 1965, 24—30.
16. Cobham A. The recognition problem for the set of perfect squares. IEEE Conference Record Seventh SWAT (1966), 78—87.
17. Cohen H., and Lenstra, H. W., Jr. Primarily testing and Jacobi sums. Report 82—18, University of Amsterdam, Dept. of Math., 1982.
18. Cook S. A. The complexity of Theorem proving procedures. Proc. 3rd ACM Symp. on Theory of Computing. Shaker Heights, Ohio (May 3—5, 1971), 151—158. [Имеется перевод: Кук С. А. Сложность процедур вывода теорем. Кибер. сб., н. с., вып. 12. — М.: Мир, 1975, с. 5—16.]
19. Cook S. A. Linear time similation of deterministic two-way pushdown automata. Proc. IFIP Congress 71 (Theoretical Foundations). North Holland, Amsterdam, 1972, 75—80.
20. Cook S. A. An observation on time-storage tradeoff. JCSS 9 (1974), 308—316. Первоначально опубликовано в Proc. 5th ACM Symp. on Theory of Computing. Austin, TX (April 30 — May 2, 1973), 29—33.
21. Cook S. A. Toward a complexity theory of synchronous parallel computation. L'Enseignement Mathematique XXVII (1981), 99—124.
22. Cook S. A., and Aanderaa, S. O. On the minimum computation time of functions. Trans. AMS 142 (1969), 291—314. [Имеется перевод: Кук С. А., Аандреаа С. О. О минимальном времени вычисления функций. Кибер. сб., н. с., вып. 8. — М.: Мир, 1971, с. 168—200.]
23. Cooley J. M., and Tukey, J. W. An algorithm for the machine calculation of complex Fourier series. Math. Comput. 19 (1965), 297—301.
24. Coppersmith D., and Winograd, S. On the asymptotic complexity of matrix multiplication. SIAM J. Comp. 11 (1982), 472—492.
25. Diffie W., and Hellman, M. E. New direction in cryptography. IEEE Trans. on Inform. Theory IT-22, 6 (1976), 644—654.
26. Dobkin D., Lipton, R. J., and Reiss, S. Linear programming is log-space hard for P. Inf. Processing Letters 8 (1979), 96—97.
27. Edmonds J. Paths, trees, flowers. Canad. J. Math. 17 (1965), 49—67.
28. Edmonds J. Minimum partition of a matroid into independent subsets. J. Res. Nat. Bur. Standards Sect. B, 69 (1965), 67—72.
29. Ferrante J., and Rackoff, C. W. The Computational Complexity of Logical Theories. Lecture Notes in Mathematics. # 718, Springer Verlag, New York, 1979.
30. Fisher M. J., and Rabin, V. O. Super-exponential complexity of Pressburger arithmetic. In Complexity of Computation. SIAM-AMS Proc. 7, R. Karp, Ed., 1974, 27—42.
31. Garey M. R., and Jonson, D. S. Computers and Intractability: A Guide to the Theory of *NP*-completness. W. H. Freeman, San Francisco, 1979. [Имеется перевод: Гэри Майкл Р., Джонсон Дэвид С. Вычислительные машины и труднорешаемые задачи. — М.: Мир, 1982, 416 с.]
32. Gill J. Computational complexity of probabilistic Turing machines. SIAM J. Comput. 6 (1977), 675—695.
33. Goldschlager L. M. Synchronous Parallel Computation. Doctoral dissertation, Dept. of Computer Science, Univ. of Toronto, 1977. См. также JACM, 2., 4 (October 1982), 1073—1086.
34. Goldschlager L. M., Shaw, R. A., and Staples, J. The maximum flow problem is log space complete for P. Theoretical Computer Science 21 (1982), 105—111.
35. Grzegorczyk A. Some classes of recursive functions. Rozprawy Matematyczne, 1963. [Имеется перевод: Гжегорчик А. Некоторые классы рекурсивных функций. В сб.: Проблемы математической логики. — М.: Мир, 1970, с. 9—49.]
36. Hartmanis J. Observations about the development of theoretical computer science. Annals Hist. Comput. 3, 1 (Jan. 1981), 42—51.

37. Hartmanis J., and Stearns, R. E. On the computational complexity of algorithms. Trans. AMS 117 (1965), 285—306. [Имеется перевод: Хартманис Дж., Стирнс Р. О вычислительной сложности алгоритмов. Кибер. сб., н. с., вып. 4. — М.: Мир, 1967, с. 57—85.]
38. Jones N. D., and Laazer, W. T. Complete problems for deterministic polynomial time. Theoretical Computer Science 3 (1977), 105—427.
39. Kaltofen E. A polynomial reduction from multivaruate to bivariate integer polynomial factorization. Proc. 14th ACM Symp. in Theory Comp., San Francisco, CA (May 5—7, 1982), 261—266.
40. Kaltofen E. A polynomial-time reduction from bivariate to univariate integral polynomial factorization. Proc. 23rd IEEE Symp. on Foundations of Computer Science. IEEE Computer Society, Los Angeles (1982), 57—64.
41. Карацуба А., Офман Ю. Умножение многозначных чисел на автоматах. — ДАН СССР 145, 2 (1962), 293—294.
42. Karp R. M. Reducibility among combinatorial problems. In: Complexity of Computer computations. R. E. Miller and J. W. Thatcher, Eds., New York, 1972, 85—104. [Имеется перевод: Карп Р. М. Взаимная сводимость комбинаторных проблем. Кибер. сб., н. с., вып. 12. — М.: Мир, 1975, с. 16—38.]
43. Хачиян Л. Г. Полиномиальный алгоритм для линейного программирования. ДАН СССР 224, 5 (1979), с. 1093—1096.
44. Knuth D. E. The Art of Computer Programming, Vol. 3 Sorting and Searching. Addison-Wesley, Reading, MA, 1973. [Имеется перевод: Кнут Д. Е. Искусство программирования для ЭВМ, т. 3. Сортировка и поиск. — М.: Мир, 1978.]
45. Колмогоров А. Н. Три подхода к определению понятия «количество информации». Проблемы передачи информации 1, 1 (1965), с. 3—11.
46. Колмогоров А. Н., Успенский В. А. К определению понятия алгорифма. УМН 13 (1958), с. 3—28.
47. Ladner R. E. The circuit value problem is log space complete for P. SIGAST News 7, 1 (1975), 18—20.
48. Lenstra A. K., Lenstra, H. W., and Lovasz, L. Factoring polynomials with rational coefficients. Report 82—05, University of Amsterdam, Dept. of Math., 1982.
49. Левин Л. А. Универсальные задачи перебора. Проблемы передачи информации 9 (1973), с. 115—116.
50. Luks E. M. Isomorphism of graphs of bounded valence can be tested in polynomial time. Proc. 21st Symp. on Foundations of Computer Science. IEEE Computer Society, Los Angeles (1980), 42—49.
51. Meyer A. R. Weak monadic socond-order theory of successor is not elementary-recursive. Lecture Notes in Mathematics 453. Springer Verlag, New York, 1975. 132—154. [Имеется перевод: Мейер А. Р. Слабая сингулярная теория второго порядка функционирования не элементарно разрешима. Кибер. сб., н. с., вып. 12. — М.: Мир, 1975, с. 62—77.]
52. Meyer A. R., and Stockmeyer L. J. The equivalence problem for regular expressions with squaring requires exponential space. Proc. 13th IEEE Symp. on Switching and Automata Theory (1972), 125—129.
53. Miller G. L. Riemann's hypothesis and tests for primanty. J. Comp. System Sci. 13(1976), 300—317.
54. Oppen D. C. A. $2^{2^{2pn}}$ upper bound on the complexity of Presburger arithmetic. J. Comp. Syst. Sci. 16 (1978), 323—332.
55. Papadimitriou C. H., and Steiglitz, K. Combinatorial Optimization Algorithms and Complexity. Prentice-Hall, Englewood Cliffs, NJ, 1982. [Имеется перевод: Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность. — М.: Мир, 1985.]
56. Paterson M. S., Fischer M. J., and Meyer A. R. An improved overlap argument for on-line multiplication. SIAM—AMS Proc. 7, Amer. Math. Soc., Providence, 1974, 97—111.

57a. Pippenger N. On simultaneous resource bounds (preliminary version). Proc. 20th IEEE Symp. on Foundations of Computer Science. IEEE Computer Society, Los Angeles (1979), 307—311.
57b. Pippenger N. J., and Fischer, M. J. Relations among complexity measures. JACM 26, 2 (April 1979), 361—381.
58. Pratt V. A., and Stockmeyer L. J. A characterization of the power of vector machines. J. Comput. System Sci. 12 (1976), 198—221. Originally in Proc. 6th ACM Symp. on Theory of Computing, Seattle, WA (April 30—May 2, 1974), 122—134.
59. Rabin M. O. Speed of computation and classification of recursive sets. Third Convention Sci. Soc., Israel, 1959, 1—2.
60. Rabin M. O. Degree of difficulty of computing a function and a partial ordering of recursive sets. Tech. Rep. No. 1, O. N. R, Jerusalem, 1960.
61. Rabin M. O. Probabilistic algorithms. In Algorithms and Complexity, New Directions and Recent Trends, J. F. Traub, Ed. Academic Press, New York, 1976, 29—39.
62. Rabin M. O. Complexity of computations. Comm. ACM 20, 9 (September 1977), 625—633.
63. Reif J. H. Symmetric complementation. Proc. 14th ACM Symp. on Theory of Computing, San Francisco, CA (May 5—7, 1982), 201—214.
64. Reisch S., and Schnitger, G. Three applications of Kolmogorov complexity. Proc. 23rd IEEE Symp. on Foundations of Computer Science. IEEE Computer Society, Los Angeles (1982), 45—52.
65. Ritchie R. W. Classes of Recursive Functions of Predictable Complexity. Doctoral Dissertation, Princeton University, 1960.
66. Ritchie R. W. Classes of predictably computable functions. Trans AMS 106 (1963), 139—173. [Имеется перевод: Ричи Р. В. Классы предсказуемо вычислимых функций. — Проблемы математической логики. — М.: Мир, 1970, с. 50—93.]
67a. Rivest R. L., Shamir A., and Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Comm. ACM 21, 2 (February 1978), 120—126.
67b. Ruzzo W. L. On uniform circuit complexity. J. Comput. System Sci. 22 (1981), 365—383.
68a. Savage J. E. The Complexity of Computing. Wiley, New York, 1976.
68b. Schnorr C. P. The network complexity and the Turing machine complexity of finite functions. Acta Informatica 7 (1976), 95—107.
69. Schönhage A. Storage modification machines. SIAM J. Comp. 9 (1980), 490—508.
70. Schönhage A., and Strassen V. Schnelle Multiplication grosser Zahlen. Computing 7 (1971), 281—292. [Имеется перевод: Шёнхаге А., Штрассен В. Быстрое умножение больших чисел. Кибер. сб., н. с., вып. 10. — М.: Мир, 1973, с. 87—98.]
71. Schwartz J. T. Probabilistic algorithms for verification of polynomial identities. JACM 27, 4 (October 1980), 701—717.
72. Schwartz J. T. Ultracomputers. ACM Trans. on Prog. Languages and Systems 2, 4 (October 1980), 484—521.
73. Shamir A. On the generation of cryptographically strong pseudo random sequences. 8th Int. Colloquium on Automata, Languages, and Programming (July 1981). Lecture Notes in Computer Science 115. Springer Verlag, New York, 544—550.
74. Shannon C. E. The synthesis of two terminal switching circuits. BSTJ 28 (1949), 59—98. [Имеется перевод: Шеннон К. Синтез двухполюсных переключательных схем. В кн.: Шеннон К. Работы по теории информации и кибернетике. — М.: ИЛ, 1962, с. 59—105.]
75. Smale S. On the average speed of the simplex method of linear programming. Preprint, 1982.

76. Smale S. The problem of the average speed of the simplex method. Preprint, 1982.
77. Solovay R., and Strassen V. A fast monte-carlo test for primality. SIAM J. Comput. 6 (1977), 84—85.
78. Stearns R. E., Hartmanis J., and Lewis, P. M. II Hierarhies of memory limited computations. 6th IEEE Symp. on Switching Circuit Theory and Logical Design (1965), 179—190. [Имеется перевод: Стирнз Р. Е., Хартманис Дж., Льюис П. М. II. Иерархии вычислений с ограниченной памятью. В сб.: Проблемы математической логики. — М.: Мир, 1970, с. 301—319.]
79. Stockmeyer L. J. The complexity of decision problems in automata theory and logic. Doctoral Thesis, Dept. of Electrical Eng., MIT, Cambridge, MA., 1974; Report TR-133, MIT, Laboratory for Computing Science.
80. Stockmeyer L. J. Classifying the computational complexity of problems. Research Report RC 7606 (1979), Math Sciences Dept., IBM T. J. Watson Research Center, Yorktown Heights, N. Y.
81. Strassen V. Gaussian elimination is not optimal. Num. Math. 13 (1969), 354—356. [Имеется перевод: Штрассен В. Алгоритм Гаусса не оптимален. Кибер. сб., н. с., вып. 7. — М.: Мир, 1970, с. 67—70.]
82. Strassen V. Die Berechnungskomplexitat von elementarsymmetrischen Funktionen und Interpolationskoeffizienten. Numer. Math. 20 (1973), 238—251. [Имеется перевод: Штрассен Ф. Сложность вычисления элементарных симметрических функций и коэффициентов интерполяционного полинома. Кибер. сб., н. с., вып. 15. — М.: Мир, 1978, с. 5—21.]
83. Baur W., and Strassen V. The complexity of partial derivatives. Preprint, 1982.
84. Тоом А. Л. О сложности схемы из функциональных элементов, реализующей умножение целых чисел. ДАН СССР 150, 3 (1963), с. 496—498.
85. Turing A. M. On computable numbers with an application to the Entscheidnunsproblem. Proc. London Math. Soc. ser. 2, 42 (1937), 230—265. A correction, ibid. 43 (1937), 544—546.
86. Valiant L. G. The complexity of enumeration and reliability problems. SIAM J. Comput. 8 (1979), 189—202.
87. Valiant L. G. The complexity of computing the permanent. Theoretical Computer Science 8 (1979), 189—202. [Имеется перевод: Вэлиант Дж. Сложность вычисления перманента. Кибер. сб., н. с., вып. 18. — М.: Мир, 1981, с. 125—140.]
88. Valiant L. G. Parallel computation. Proc. 7th IBM Japan Symp. Academic 6 Scientific Programs, IBM Japan, Tokyo (1982).
89. Valiant L. G., Skyum S., Berkowitz S., and Rackoff, C. Fast parallel computation on polynomials using few processors. Preprint. (Preliminary version in Springer Lecture Notes in Computer Science 118 (1981), 132—139).
90. von Neumann J. A certain zero-sum two-person game equivalent to the optimal assignment problem. Contributions to the Theory of Games II. H. W. Kahn and A. W. Tucker, Eds. Princeton Univ. Press, Princeton, NJ, 1953.
91. Yamada H. Real time computation and recursive functions not real-time computable. IRE Transactions on Electronic Computers EC-11 (1962), 753—760. [Имеется перевод: Ямада Х. Вычисления в реальное время и рекурсивные функции, не вычислимые в реальное время. В кн.: Проблемы математической логики. — М.: Мир, 1970, с. 139—155.]
92. Yao, A. C. Theory and applications of trapdoor functions (extended abstract). Proc. 23rd IEEE Symp. on Foundations of Computer Science. IEEE Computer Society, Los Angeles (1982), 80—91.