МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

РЕФЕРАТ ПО ИСТОРИИ НАУКИ

# Теория алгоритмов: истоки, открытия, приложения

Выполнил:
ПАВЛОВСКИЙ ЕВГЕНИЙ НИКОЛАЕВИЧ,
аспирант ММФ НГУ по специальности 01.01.06
«алгебра, математическая логика и теория чисел».

Научный руководитель:
ГОНЧАРОВ СЕРГЕЙ САВОСТЬЯНОВИЧ,
чл.-корр. РАН, д.ф.-м.н., профессор,
зав.отд. ИМ СО РАН.

Новосибирск
2006

## Содержание

## Введение

Понятие алгоритма уже давно вошло в математическую практику, более того, оно свободно используется и в других областях деятельности. Но знаем ли мы, когда оно вошло в столь широкое употребление? Откуда произошло само слово «алгоритм»?

Здесь делается попытка проследить образование современного понятия алгоритма. Рассматриваются древняя математика в виде предписаний для управляющих и писцов, исторические корни слова «алгоритм» и формирование математического понятия алгоритма.

Также рассматриваются формирование науки Теория Алгоритмов, её основы, приложения её результатов в математике, других науках и областях деятельности.

Интересно, что появившись в формальном виде, понятие алгоритма так фундаментально обогатило всю математику и другие науки. Удивительным оказывается и тот факт, что понятие алгоритма лежит в основе понятия доказательства.

## 1. Алгоритмы в математике древних цивилизаций

*а) Древний Египет*

История понятия «алгоритм» уходит корнями в древние цивилизации, как сама математика. Если нам и достоверно неизвестно о математике ещё более древних цивилизаций, чем древний Египет и древний Вавилон, то о последних всё же можно кое-что сказать.

О математике древних египтян известно из папируса Райнда, который был найден и расшифрован в XIX веке [Бобынин, 1882][1]. В папирусе перечислено множество задач на измерение площадей полей и подсчёт единиц. В нём также излагаются инструкции жрецов для писцов. Быть писцом в древнем Египте было почётно, т.к. писец заведовал административно-хозяйственной частью, подсчитывал и указывал, сколько куда человек нужно направить и на какие работы в каком количестве. От него зависело состояние дел.

В папирусе Райнда содержатся инструкции о том, как подсчитывать. И эти инструкции есть документальное подтверждение существования алгоритма ещё в древнем Египте. Однако такой алгоритм ещё не рассматривался самой математикой, к тому же сама математика ещё не была осознана как отдельная область деятельности. Основным объектом математики древнего Египта были числа и элементарные операции над ними. По папирусу Райнда видно, что основным алгоритмом при вычислении операций деления натуральных чисел и сложения дробей был перебор. На бумаге записывались последовательно числа, элементарные операции с этими числами (т.е. умножение на два, на пять и на десять и сложение), а затем перебором подбирались правильные ответы, которые сходились в проверке. Можно сказать, что это был алгоритм проверки.

Из тех материалов, что дошли до нас о древнем Египте нельзя заключить, что тогда существовало отдельное понятие алгоритма, хотя, по сути, в математику уже тогда закладывались основные алгоритмические истины: последовательное оперирование конечными объектами, с результатом вновь в виде конечного объекта. В книге [Юшкевич, 1970] так оценена математика древнего Египта: «математика представляла собой совокупность знаний, ещё не расчленившуюся на арифметику, алгебру, геометрию и выступающую, прежде всего как собрание правил для численного решения простейших арифметических, алгебраических и геометрических задач».

*б) Древний Вавилон*

О математике древнего Вавилона мы знаем из клинописных глиняных табличек. [Юшкевич, 1970]. Из них, в частности известно, что тогда решались математические задачи, в том числе не имеющие отношения к практической деятельности, хотя по большей части формулировки не были отвлечёнными: вместо привычных нам переменных тогда использовались слова «длина» ($x$), «ширина» ($y$), «глубина» ($z$), «площадь» ($xy$), «объём» ($xyz$).

Тогда же зародились начала алгебры, т.е. решения систем линейных и квадратичных уравнений. Причём сами задачи (из тех, что дошли до нашего времени в виде глиняных табличек) имели отвлечённый характер.

«Я вычел из площади сторону моего квадрата, это 14,30» - эту формулировку можно перевести на современный математический язык так: $x^2$—$x$=14,30. Далее следует вычисление: «Ты берёшь 1, коэффициент. Ты делишь пополам 1, это 0;30. Ты умножаешь 0;30 на 0;30, это 0;15. Ты складываешь [это] с 14,30 и это есть 14,30;15, что является квадратом для 29;30. Ты

[1] здесь и далее ссылка на литературный источник будет указываться в квадратных скобках: [фамилия(-и) автора(-ов), год публикации].

складываешь 0;30, которое ты умножал, с 29;30, получается 30, сторона квадрата», [Нейгебауэр].

Здесь, по сути, идёт описание действий вычислителя, т.е. описывается конкретная инструкция вычислителю.

*в) Древняя Греция. Пифагор. Евклид.*

В древней Греции математика приобретает самостоятельную ценность. Так возникает пифагорейская школа, где ученики Пифагора доказывают различные факты лишь на основании логических рассуждений. «Пифагор преобразовал эту науку [математику] в форму свободного образования. Он изучал эту науку, исходя из первых её оснований, и старался получать теоремы при помощи чисто логического мышления, вне конкретных представлений» [Прокл, V в.].

Появляется фундаментальный труд Евклида, в котором предложен знаменитый «алгоритм Евклида» нахождения наибольшего общего делителя. Конечно, словосочетание «алгоритм Евклида» появилось и установилось гораздо позже, до этого был «метод Евклида». Открытие этого алгоритма позволило доказать несколько фундаментальных теорем теории целых чисел (разложение любого числа на произведение простых делителей). Характерно для древней Греции, что алгоритм сформулирован для произвольных чисел, а не для конкретных, как в более древних цивилизациях (Вавилоне, Египте). Такая традиция общих формулировок была заложена именно в древней Греции. И именно из-за того, что алгоритм по своей сути представляет некоторый общий метод решения однотипных задач, он впоследствии приобретёт самостоятельную математическую ценность.

## 2. “Dixit Algorizmi”. Ал-Хорезми

Современная история математики большую роль формированию алгебры в том виде, который мы имеем сейчас, относит к трудам ал-Хорезми, арабского учёного, перенёсшего индийскую позиционную арифметику на запад [Депман, 1965, стр.77].

«Слово “алгорифм” в форме “алгоризм” часто (вслед за английским поэтом Чоусером с 1350 г.) употреблялось в качестве заглавия изложений индийского счисления в рукописях XII–XV вв. и в книгах XV–XVI вв.». [Депман, 1965, с.77-78]. Свои трактаты ал-Хорезми начинал со слов «dixit algorizmi», что означает «Говорит ал-Хорезми».

Алгоризм овладевал всё большими территориями. Арифметика, представленная ал-Хорезми была более практичной, т.к., в отличие от системы счёта на абаке[2], была основана на индийской позиционной системе счисления, позволявшей быстро и просто считать, не прибегая к громоздкому абаку.

Цифры проникли во все стороны деятельности, связанные со счётом. Новая позиционная нумерация постепенно вытеснила старую практически во всех странах.

Позднее особую устойчивость приобретает словоупотребление «алгоритм Евклида». И вплоть до XX века слово «алгоритм» употребляется только так.

[2] *абак* – инструмент счёта. Как правило абаки сначала изготавливались из плоского камня или дерева, на котором в разных помеченных областях (соответствующих разным разрядам – десяткам, сотням) раскладывали камешки

## 3. Основные результаты теории алгоритмов

Появлению формализованного понятия алгоритма в математике мы обязаны Тьюрингу, Посту, Чёрчу, Маркову и Колмогорову и др. Именно они явно поставили и решили задачу формального описания понятия «алгоритм». Само понятие алгоритма было осознано только к началу XX века. В работах Бореля (1912г.) и Вейля (1921 г.) встречаются едва ли не самые ранние примеры использования понятия эффективно вычислимой процедуры. «Я намеренно оставляю в стороне большую или меньшую практическую деятельность; суть здесь та, что каждая из этих операций осуществима в конечное время при помощи достоверного и недвусмысленного метода», — подчёркивает Борель, говоря о «вычислениях, которые могут быть реально осуществлены» [Борель, 1912, с.161-162].

С возникновением понятия алгоритма в математике начали появляться теоремы о несуществовании алгоритмов для решения некоторых задач. Эти теоремы впоследствии и заложили фундамент теории алгоритмов. [Гёдель 1931]

Теория алгоритмов очень быстро обнаружила ряд фундаментальных приложений в математике. Среди них: связь алгоритмов и исчислений, неразрешимость арифметики, алгоритмичность доказательств. О приложениях теории алгоритмов в математике подробно будет сказано ниже.

Также теория алгоритмов позволила определить некоторые подходы в теоретическом и практическом программировании.

В обзорной книге по теории алгоритмов [Успенский, 1987] перечислены основные открытия, связанные с общими понятиями алгоритма и исчисления:

1. Общее понятие алгоритма как самостоятельное (отдельное) понятие.
2. Представительные вычислительные модели.
3. Общее понятие исчисления как самостоятельное (отдельное) понятие.
4. Представительные порождающие модели.
5. Выяснение связей между алгоритмами и исчислениями.
6. Время и ёмкость как сложности вычисления и порождения.
7. Вычислимые функции и породимые множества.
8. Понятие частично-рекурсивной функции.
9. Возможность арифметического и даже диофантова представления любого перечислимого числового множества.
10. Построение неразрешимого породимого числового множества.
11. Проблема сводимости Поста.
12. Понятие относительного алгоритма, или алгоритма с оракулом.
13. Понятие вычислимой операции.
14. Понятие программы: программы как объекты вычисления и порождения.
15. Понятие нумерации и теория нумераций.
16. Начало создания инвариантной, или машинно-независимой, теории сложности вычислений.
17. Теория сложности энтропии конструктивных объектов.
18. Удобные и экономные вычислительные модели.

Таким образом, даже просто просматривая этот список, мы убеждаемся, что возникновение общего понятия алгоритма само по себе очень обогатило математику, фактически открыв новую её ветвь. Но мы будем ещё более удивлены, обнаружив, насколько глубоко проникла теория алгоритмов в другие области математики. Её приложения настолько фундаментальны, что заставили пересмотреть некоторые основы математики. Например, само понятие доказательства оказывается подпонятием алгоритма, в чём мы убедимся позже.

Далее мы поподробнее рассмотрим несколько наиболее интересных из перечисленных открытий.

*3.а) Общее понятие алгоритма*

«Самым главным открытием в науке об алгоритмах, безусловно, было открытие самого понятия алгоритма в качестве новой и отдельной сущности» [Успенский, 1987, с.29]. Сами формальные конструкции, уточняющие понятие алгоритма, были предложены несколько позднее, но до них уже существовало некоторое интуитивное понятие алгоритма (понятие вычислимой функции). Так, уже упомянутый Борель использует некоторое интуитивное представление о вычислимости функции, Чёрч в докладе, представленном в 1935 году пользуется термином «эффективно вычислимая<effectively calculable> функция», считая его общепонятным и предшествующим какой бы то ни было формализации. В ранних сочинениях по теории алгоритмов Тьюринг также пользуется интуитивным понятием «effectively calculable» [Тьюринг, 1936, 1937а].

«Понятие алгоритма, подобно понятиям множества и натурального числа принадлежит к числу понятий столь фундаментальных, что оно не может быть выражено через другие (в частности, теоретико-множественные), а должно рассматриваться как неопределяемое» [Успенский, 1987].

Марков предлагает следующее пояснение к понятию алгоритма: «Алгоритм — это точное предписание, которое задаёт вычислительной процесс, начинающийся с произвольного (но выбранного из фиксированной для данного алгоритма совокупности) исходного данного и направленный на получение полностью определяемого этим исходным данным результата» [Марков, 1957].

Более точная формулировка рассматривается Колмогоровым:

> «Мы отправляемся от следующих наглядных представлений об алгоритмах:
>
> 1) Алгоритм Г, применённый ко всякому «условию» («начальному состоянию») $A$ из некоторого множества $S$ («области применимости» алгоритма Г), даёт «решение» («заключительное состояние) $B$.
>
> 2) Алгоритмический процесс расчленяется на отдельные шаги заранее ограниченного сложности; каждый шаг состоит в «непосредственной переработке» возникшего к этому шагу состояния $S$ в состояние $S^*=\Omega_Г(S)$.
>
> 3) Процесс переработки <...> продолжается до тех пор, пока либо не произойдёт безрезультатная остановка <...>, либо не появится сигнал о получении «решения». При этом не исключается возможность неограниченного продолжения процесса <...>.
>
> 4) Непосредственная переработка $S$ в $S^*=\Omega_Г(S)$ производится лишь на основании информации о виде заранее ограниченной «активной части» состояния $S$ и затрагивает лишь эту активную часть» [Колмогоров, 1953].

Эта формулировка достаточно общая и включает в себя формальное понятие машины Тьюринга, Поста и многие другие известные формальные задания. Формулировка Колмогорова содержит две существенные идеи: итеративность алгоритмического процесса и локальность каждого отдельного шага. Для того же, чтобы задать вычислительную модель необходимо конкретизировать понятия «состояние», «непосредственная переработка», «активная часть», «сигнал о решении». В [Колмогоров, 1953] предложена общая схема такой конкретизации. Эта схема может рассматриваться как адекватная формализация точного понятия алгоритма. Здесь, при работе с моделями с нелокальным преобразованием информации нудно разбить нелокальные шаги на локальные, — и мы видим замечательную параллель одного известного метода, изложенного ещё Декартом: «Сложную задачу разбивай на простые». Только в случае алгоритма эти простые задачи должны решаться однотипно.

Кстати, много параллелей у современного понятия алгоритма прослеживается и с алгоритмом Евклида. Так, например, в алгоритме Евклида впервые были применены инструкции не к

конкретным числам, а вообще ко всем натуральным, т.е. указан класс объектов (в современной терминологии «конструктивных объектов»), к которому применяется алгоритм. Причём позже, уже в 18-19 века при изучении произвольных колец чисел с порождающими неразложимыми элементами было обнаружено, что для таких чисел алгоритм Евклида не даёт сходимости [Юшкевич, 1970], т.е. не приводит за конечное число шагов к конечному результату. Возможно, отсюда и появилась идея «неограниченного продолжения процесса» в случае непоявления сигнала о «решении».

Как правило, конкретный алгоритм задаётся в некоторой вычислительной модели. Среди первых вычислительных моделей были предложены: машина Тьюринга [Тьюринг, 1936], Поста [Пост, 1936], нормальные алгорифмы Маркова [Марков, 1951], машины Колмогорова [Колмогоров, 1953]. В формальное описание этих моделей мы углубляться не будем, если читателю интересно, он надёт их по указанным ссылкам.

Стоит отметить, что все вычислительные модели могут быть описаны с помощью конструктивных объектов и вновь выступать в качестве материала для переработки некоторым алгоритмом. Отсюда появляется понятие универсального алгоритма и возникает основание для построения общей теории алгоритмов. Можно подумать, что общая теория будет существенного зависеть от выбора вычислительной модели, однако такого не происходит, даже более того имеются все основания утверждать, что теория алгоритмов единственна. В основе этого убеждения лежит теорема Роджерса об эквивалентности гёделевских нумераций [Роджерс, 1958] и теорема Мучника об изоморфизме вычислительных структур [Мучник, 1958].

В связи с этим в теории алгоритмов приняты тезисы Чёрча и Тьюринга (которые по упомянутым выше теоремам эквивалентны друг другу по силе) о том, что интуитивно вычислимые функции вычислимы и при формальном задании.

### *3.б) Общее понятие исчисления*[3]

Успенский отмечает: «Общее понятие исчисления, или дедуктивной системы, столь же фундаментально, как и понятие алгоритма, и должно рассматриваться отдельно от каких бы то ни было формальных уточнений». Математические истоки понятия исчисления восходят ещё к древности [Яновская, 1962]. Так, например, игры: шахматы, домино, различные карточные игры, возможно и послужили прототипами исчислений. Дифференциальное и интегральное исчисления также могут служить примерами исчислений, если рассматривать их как процедуры, позволяющие порождать равенства вида:

$$dF(x)=f(x)dx \text{ и } \int f(x)dx=F(x),$$

отправляясь от исходных, или «табличных», равенств, применяя такие правила, дифференцирование сложной функции, интегрирование по частям. В формировании понятия исчисления сыграли большую роль исчисления математической логики (логистические системы), первые из которых разрабатывал Фреге [Фреге, 1879].

Подобно тому, как алгоритм задаёт *алгоритмический*, или *вычислительный*, *процесс*, каждое исчисление задаёт *исчислительный*, или *порождающий*, *процесс*, т.е. процесс работы исчисления. Принципиальная разница между этими процессами лежит в том, что в алгоритмическом процессе каждое возникающее состояние однозначно предопределено предшествующим ходом процесса, а в исчислительном процессе возникающее состояние является всего лишь одним из многих возможных допускаемых предшествующим ходом процесса. Можно сказать, что в алгоритмическом процессе время течёт линейно, а в исчислительном – разветвлённо.

[3] исчисление по-латински — calculus, а само слово calculus происходит от слова камешек (calculi). Такая этимология связана историей математических вычислений – в древние времена они производились перекладыванием камешков.

В настоящее время изучено уже много различных логистических исчислений, например: пропозициональное исчисление, исчисление предикатов 1-го порядка, пропозициональное исчисление темпоральной логики, исчисление темпоральной логики 1-го порядка, исчисление логики высказываний с ветвящимся временем. В большинстве из перечисленных истинны *теоремы полноты*, т.е. теоремы, которые утверждают, что все истинные формулы можно породить внутри этих исчислений (в данном случае породить означает – построить *доказательство*, т.е. последовательность формул, каждая из которых, либо аксиома исчисления, либо является формулой полученной из предыдущих по одному из правил исчисления). Так, например, известная Теорема Гёделя о полноте формулируется именно для логики предикатов 1-го порядка. «С точки зрения истории науки представляет интерес тот факт, что полное исчисление предикатов было угадано логиками за несколько десятилетий до теоремы Гёделя» [Успенский, 1987].

Другая знаменитая теорема Гёделя — *теорема о неполноте* — утверждает, что множество всех истинных формул арифметики (а значит, и множество всех общезначимых формул логики предикатов второго порядка) не может быть порождена никаким исчислением. Эта теорема имеет место благодаря тесной связи алгоритмов и исчислений, а именно: всякое исчисление может быть понято, как алгоритм и преобразовано в него, а всякий алгоритм задаёт некоторое исчисление. Теорема Гёделя о неполноте служит замечательной иллюстрацией соединения этих двух понятий в виде фундаментального математического, и даже философского результата.

## 4. Приложения теории алгоритмов в математике, других науках и областях деятельности.

Успенский отмечает: «огромное число теорем, предполагающих построение тех или иных алгоритмов и встречающихся в различных областях математики, не требует для своего понимания общего понятия алгоритма» [Успенский, 1987, с.144]. Появление общего понятия алгоритма позволило отвечать как раз на противоположный вопрос: несуществование алгоритма. «Теорема о несуществовании алгоритма обращается к идее всего класса алгоритмов в целом и, следовательно, является теоремой теории алгоритмов» [Успенский, 1987, там же.]

Следующий список приложений взят из [Успенский, 1987]:

1. Исследование массовых проблем.
2. Приложения к основаниям математики: конструктивная семантика.
3. Приложения к математической логике: анализ формализованных языков логики и арифметики.
4. Вычислимый анализ.
5. Нумерованные структуры.
6. Приложения к теории вероятностей: определения случайной последовательности.
7. Приложения к теории информации: алгоритмический подход к понятию количества информации.
8. Оценки сложностей решения отдельных задач.
9. Влияние теории алгоритмов на алгоритмическую практику.

Кроме приложений в математике теория алгоритмов имеет множество приложений в других науках: в биологии (описание рефлексов в терминах относительных алгоритмов, генетический код как программа), в психологии, в теории управления, в языковедении.

Мы же подробно рассмотрим лишь несколько, наиболее наглядных примеров приложений теории алгоритмов в математике:

*4. а) Исследование массовых проблем*

*Алгоритмическая проблема* — это проблема построения алгоритма, перечисляющего данное множество (например, алгоритма, перечисляющего данное множество, или алгоритма с данной оценкой сложности, решающего данную задачу). Частным случаем алгоритмических проблем являются *алгоритмические массовые проблемы*. «Массовые проблемы образуют основное поле приложения теории алгоритмов; более того, именно они и вызвали к жизни само понятие алгоритма» [Успенский, 1987, с.146].

*Единичная проблема* состоит в требовании предъявить объект, удовлетворяющий определённым условиям и называемый *решением* проблемы; *решить* проблему – значит указать такой объект; если решение существует, т.е. если проблему можно решить, проблема называется *решимой*.

*Массовая проблема* представляет собой серию (как правило, бесконечную) единичных проблем и состоит в требовании решить все эти проблемы; понимание того, что же именно считается решением массовой проблемы, нуждается в уточнении. Пример единичной проблемы: для уравнения $x^2$—$x$—1=0 найти рациональное приближение к отрицательному корню с точностью $10^{-6}$. Пример массовой проблемы: в той же ситуации для любого $n$ найти рациональное приближение с точностью $10^{-n}$.

Понятие массовой проблемы несколько расплывчато, и естественно попытаться найти его формальный эквивалент. Обычный способ нахождения такого эквивалента заключается во введении понятия *алгоритмической массовой проблемы*. Чтобы задать алгоритмическую массовую проблему, следует указать:

1) конструктивное пространство $X$ (множество вопросов, или единичных проблем).
2) конструктивное пространство $Y$ (множество ответов, или единичных решений).
3) подмножество $E \subseteq X$ (ограничение на вопросы).
4) подмножество $R \subseteq X \times Y$ (отношение «вопрос — ответ»), или вопросно-ответное отношение.

Тогда проблема состоит в требовании найти алгоритм из $X$ в $Y$, преобразующий каждый вопрос $a \in E$ в ответ $b \in Y$ со свойством $\langle a, b\rangle \in R$. В приведённом примере с квадратным уравнением

$X = \mathbb{N}$, $Y = \mathbb{Q}$, $E = X$, $\mathrm{R} = \{ \langle n, r\rangle \mid |r\text{-}x_0| < 10^{-n}$, где $x_0$ — требуемый корень$\}$.

Замена массовой проблемы на соответствующую алгоритмическую позволяет представить неясное понятие в виде точного определения. В частности, возникают точные понятия алгоритмических массовых проблем разрешимости и отделимости. Кроме того, указанная замена превращает массовую проблему в единичную, т.е. алгоритмическая массовая проблема заключается в требовании представить единичный объект, являющийся решением, а именно алгоритм.

Разумеется, наличие решения у каждой единичной проблемы из составляющих данную массовую проблему, т.е. наличие для каждого $a$ из $E$ такого $b$, что $\langle a, b\rangle \in \mathrm{R}$, ещё не означает наличия решения у соответствующей алгоритмической массовой проблемы. Вот два характерных примера из [Матиясевич, 1974].

Пример 1.Ещё в 1908 г. Туэ доказал, что для каждой неприводимой бинарной формы $F$ не менее чем третьей степени с целыми коэффициентами справедливо утверждение (все переменные целочисленные):

$$\forall a \exists b \forall x \forall y [\, F(x, y) \Rightarrow |x| + |y| < b]$$

Понадобилось, однако, 60 лет, чтобы доказать наличие алгоритма, дающего $b$ по $F$ и $a$.

Пример 2. Матиясевич построил полином $A$ с целочисленными коэффициентами, для которого

$$\forall a \exists b \forall y \forall z_1 \forall z_2 \ldots \forall z_n [\, A(a, z_1, \ldots, z_n) = y + 4^y \Rightarrow y + z_1 + \ldots + z_n \mathrel{\mathsf{m}} b]$$

(здесь $a$, $b$, $y$, $z_1$, ..., $z_n$ – натуральные числа), но невозможен алгоритм получения $b$ по $a$.

Построение алгоритма, служащего решением той или иной алгоритмической массовой проблемы, является скорее собственностью той части математики, к которой относится рассматриваемая проблема, чем приложением общей теории алгоритмов. С другой стороны, если такого алгоритма не существует, доказательство его несуществования принадлежит приложениям общей теории алгоритмов. Многие такие проблемы были доказаны для проблем разрешения, поставленных, как правило, для породимых множеств.

Первые доказательства нерешимости алгоритмических массовых проблем относятся к 1936 г.; они были опубликованы Чёрчем и Тьюрингом в [Чёрч, 1936] и [Тьюринг, 1936]. Эти проблемы являются проблемами разрешения и связаны с предложенными этими авторами представительными порождающими (в виде λ-исчисления Чёрча) и вычислительными моделями (в виде машин Тьюринга); однако уже Чёрч заметил, что тем самым возникает нерешимая массовая проблема внутри элементарной теории чисел.

Из числа наиболее замечательных нерешимых проблем Успенский выделяет семь «по принципу философской значимости» [Успенский, 1987, с.151]:

1. *Проблема распознавания истинности формул элементарной арифметики*. Эти формулы стоятся с помощью арифметических действий (сложения и умножения), логических операций (логических связок и кванторов) и знака равенства из константы 0 и натуральных переменных. Проблема состоит в требовании найти алгоритм, который по всякой такой формуле определял бы, истинна она на натуральном ряду или нет. Невозможность такого алгоритма немедленно следует из существования неразрешимого перечислимого множества натуральных чисел и того, что такое множество является арифметическим.

2. *Проблема решения для логики предикатов первого порядка*.

3. *Проблема сочетаемости Поста*.

4. *Проблема эквивалентности слов для ассоциативного исчисления*.

5. *Проблема представимости матриц*.

6. *Десятая проблема Гильберта*. Эта проблема формулируется так: «Задача о разрешимости диофантова уравнения с произвольными неизвестными и целыми рациональными числовыми коэффициентами. Указать способ, при помощи которого возможно после конечно числа операций установить, разрешимо ли это уравнение в целых рациональных числах» [Гильберт, 1935, с.310]. Несуществование алгоритма показал Матиясевич, доказав, что всякое перечислимое множество можно представить в виде решения некоторого диофантова уравнения. Учитывая, что существуют алгоритмически неразрешимые перечислимые множества, получаем неразрешимость десятой проблемы. Однако остаётся открытым вопрос о разрешимости диофантовых уравнений в рациональных числах.

7. *Проблема тождества элементарных функций вещественного переменного*. Определим класс термов $T$ индуктивно:

1° переменная $x$, число $\pi$ — термы;
2° если $u$, $v$ — термы, то $(u + v)$, $(u \cdot v)$, $(u : v)$, $\sin u$, $|u|$ — термы.

Проблема построения алгоритма, узнающего по двум термам из Т, задают ли они одну и ту же функцию одного вещественного переменного $x$, нерешима [Матиясевич, 1973]. Другим примером является проблема существования у функции, заданной термом из некоторого фиксированного класса, первообразной, задаваемой термом того же класса. Эта проблема нерешима для различных классов термов. Таким образом, уже обычное интегральное исчисление даёт возможность получения нерешимых алгоритмических проблем.

**Массовые проблемы в математике.** Массовые проблемы возникают во всех областях математики. Наиболее фундаментальные из них — это проблемы выяснения истинности

утверждений, формулируемых в некотором математическом языке. Реальные математические языки всегда допускают рассмотрение натуральных чисел с обычными операциями над ними, поэтому возникающие при этом множества истинных формул оказываются неразрешимыми.

Таким образом, вопрос о разрешимости множества истинных формул некоторой логической теории оказывается нетривиальным, только если выразительные средства и объекты рассмотрения этой теории весьма ограничены. В частности, отказ от автоматически приводящего к неразрешимости использования предикатных и функциональных переменных фактически приводит к тому, что рассматриваемые массовые проблемы касаются алгебраических объектов. Для этого есть и другая причина; дело в том, что во многих важных случаях алгебраические объекты допускают естественное конструктивное задание, поэтому вопросы о них легко формулируются в алгоритмической форме.

Итак, нетривиальные результаты о разрешимости или неразрешимости логических теорий относятся, в основном, к элементарным теориям алгебраических систем. Наиболее тонкие из них касаются алгебраических систем, расположенных вблизи границы между разрешимостью и неразрешимостью, важнейшие факты относятся к ходу этой границы среди полей: неразрешима элементарная теория поля рациональных чисел (см. [Робинсон, 1949]), разрешима элементарная теория поля действительных чисел (см. [Ершов Ю., 1980, гл. 5, § 5, предложение 1]) и элементарная теория поля $p$-адических чисел (при любом $p$) (см. [Ершов Ю., 1980, гл. 5, § 5, предложение 4]).

Для других алгебраических систем возникающие элементарные теории, как правило, неразрешимы (так обстоит дело, в частности, для групп, колец и многих классов этих алгебраических систем, см. [Тарский, Мостовский, Робинсон, 1953]); например, неразрешима элементарная теория класса всех простых конечных групп (см. [Ершов Ю., 1964а]). К немногим исключениям относятся абелевы группы и некоторые классы упорядоченных множеств (см. [Ершов Ю., 1964], [Рабин, 1969]), а также произвольные свободные алгебры (это следует из [Мальцев, 1962, теорема 5]). Подробные таблицы, суммирующие результаты о неразрешимости и разрешимости элементарных теорий, имеются в [Ершов Ю., Лавров, Тайманов, Тайцлин, 1965] и [Санкаппанавар, 1978] (в последней работе приводятся также сведения о некоторых неэлементарных теориях). О разрешимости логических теорий см. также [Семенов, 1984, 1986].

Есть еще одна область, в которой с математическими объектами изучения естественно сопоставляются их конструктивные описания. Это — комбинаторная топология, где с полиэдрами (в частности, триангулируемыми многообразиями) сопоставляются записи их триангуляции. Коль скоро такое сопоставление осуществлено, делается осмысленной проблема гомеоморфии, аналогичная проблеме распознавания равенства слов в группе или проблеме изоморфии групп. Согласно [Марков, 1958а], *общая проблема гомеоморфии* для полиэдров (соответственно для триангулируемых многообразий) есть проблема разыскания алгоритма, распознающего для любых двух полиэдров (соответственно многообразий), заданных своими триангуляциями, гомеоморфны ли они; *частные проблемы гомеоморфии* возникают, если наложить на рассматриваемую пару полиэдров какое-либо ограничение, например, указать их размерность или фиксировать первый член пары. Некоторые из этих проблем были давно решены, например проблема гомеоморфии двумерных многообразий. Как установил Марков, общая проблема гомеоморфии не имеет решения; более того, для всякого $n>3$ можно указать такое $n$-мерное многообразие, что проблема гомеоморфии многообразий этому многообразию нерешима (см. [Марков, 1958а], [Марков, 1958в]).

О нетривиальных алгоритмических проблемах в других областях математики мало что известно. «Иногда у специалистов имеется явное ощущение эффективности некоторого построения; в то же время попытка сформулировать соответствующую алгоритмическую проблему либо вообще ни к чему не приводит, либо же приводит к нерешимой проблеме и тем самым — к неадекватной формулировке» [Успенский, 1987]. Все же, касаясь других областей математики, выделим два примера из теории дифференциальных уравнений: нерешаемость

проблемы существования решения, определенного на отрезке [0, 1] у системы дифференциальных уравнений (см. [Адлер, 1969]), и формулировку алгоритмической проблемы об устойчивости (см. [Арнольд, 1976]).

*4. б) Вычислимый анализ*[4]

**Борель и Тьюринг.** Понятие *вычислимого действительного числа* и *вычислимой функции действительного переменного* восходят к статье Бореля [Борель, 1912]; в этой же статье намечены и некоторые основные факты вычислимого анализа. Раздел II указанной статьи называется «Вычислимые числа» (Nombres calculables) и начинается со следующего определения: «Мы скажем, что число *a* вычислимо, коль скоро для произвольного целого *n* можно получить рациональное число, которое отличается от a менее чем на 1/*n*». Сделанное в этой формулировке примечание о «достоверном и недвусмысленном методе» получения не оставляет сомнений, что Борель имел в виду самую общую концепцию *алгоритмической вычислимости.* Сейчас мы бы сказали: «Действительное число *a* вычислимо, коль скоро существует алгоритм, дающий по всякому целому положительному *n* рациональное приближение к *a* с точностью до 1/*n*».

Далее Борель указывает, что если два вычислимых числа не равны, то их неравенство может быть рано или поздно обнаружено путем подбора достаточно близких рациональных приближений (хотя точность, с которой необходимо брать такие приближения, и не может быть предвидена заранее); если же два вычислимых числа равны, то попытка обнаружить их равенство может натолкнуться на неразрешимые трудности <difficultes insolubles>. Современная формулировка: «Каково бы ни было конструктивное действительное число *y*, невозможен алгоритм, указывающий для любого конструктивного действительного числа *x* верный член дизъюнкции $(x=y)\vee(x\neq y)$» ([Кушнер, 1973, гл. 4, § 1, теорема 3]).

Раздел III статьи Бореля называется «Вычислимые функции и функции с асимптотическим определением» (Les fonctions calculables et les fonctions a definition asymptotique). Буквальная формулировка гласит: «Функция вычислима, коль скоро ее значение вычислимо для каждого значения аргумента». Однако в последующих комментариях Борель, по существу, требует наличия алгоритма, позволяющего по *a* и *n* вычислить $f(a)$ с точностью 1/*n*, поясняя при этом, что «задать вычислимое число *a* — это просто задать метод получения *a* с произвольной точностью».

Современное определение вычислимой функции вычислимого действительного переменного может поэтому трактоваться как уточнение определения Бореля. Борель формулирует и утверждение о непрерывности вычислимой функции (доказательство этого утверждения было найдено лишь в 1956 г. Цейтиным). Он пишет: «Функция не может быть вычислимой, если она не является непрерывной, по крайней мере, для вычислимых значений аргумента». «Более того,— указывает Борель, — нужно предполагать известной меру непрерывности функции, то есть инфинитезимальный порядок (в обобщенном смысле) изменения функции в сравнении с изменением аргумента». Если понимать эту «меру непрерывности» как вычислимый регулятор непрерывности (см. ниже), можно заключить, что Борель имел в виду не просто непрерывность, но вычислимую непрерывность.

Систематическое развитие вычислимого анализа на основе точных алгоритмических представлений началось со статьей Тьюринга ([Тьюринг, 1936], [Тьюринг, 1937]). История этого развития прослежена в [Кушнер, 1973, введение, п. 2].

**Конструктивный анализ.** Публикации в обсуждаемой области можно отнести к одному из двух направлений. За первым из них мы сохраняем название *«вычислимый анализ»,* второе обычно называется *«конструктивный анализ».* Объекты, изучаемые в первом направлении, носят названия «вычислимые числа», «вычислимые функции» и т. п.; объекты второго

[4] параграф целиком взят из [Успенский, 1987, с.173-176].

направления называются «конструктивные числа», «конструктивные функции» и т. п.; к сожалению, это разделение терминологии не всегда соблюдается.

Различие между направлениями состоит в следующем. Вычислимый анализ выделяет среди традиционных объектов анализа — чисел и функций — вычислимые, т. е. связанные определенным образом с алгоритмами. Конструктивный анализ рассматривает вычислимые числа и функции не как часть более обширной совокупности всех чисел и функций, а сами по себе. Более того, понятие программы числа или функции является для него исходным: конструктивным числом называется то, что в вычислимом анализе называется программой вычислимого числа, а конструктивной функцией — то, что в вычислимом анализе называется программой вычислимой функции; на конструктивных числах и конструктивных функциях затем вводится отношение равенства, означающее, разумеется, не совпадение соответствующих конструктивных объектов, а совпадение задаваемых ими чисел и функций.

Такой способ изложения позволяет говорить непосредственно об алгоритмах над конструктивными функциями. Ясно, что понятия и результаты вычислимого анализа и конструктивного анализа без труда переводятся друг в друга. Следует, однако, отметить, что в содержание понятия «конструктивный анализ», как правило, вкладывается ещё и требование использовать только конструктивную логику (см. [Кушнер, 1979а]).

*4. в) Нумерованные алгебраические структуры*

Нумерованная структура — это математическая структура, рассматриваемая вместе с нумерацией одного из составляющих множеств или с нумерациями нескольких таких множеств. Интерес к нумерованным структурам вызван желанием дать (конструктивные) имена рассматриваемым (не конструктивным) объектам. Среди примеров нумерованных структур можно дать: систему обозначений Клини конструктивных ординалов [Роджерс, 1967], стандартная нумерация элементов конечно-заданной алгебры посредством термов, нумерация всех частично-вычислимых функций.

Особенностью класса конструктивных ординалов как нумерованного множества является то, что исторически первое определение этого класса фактически использовало общее понятие нумерации (в то время, как алгебраические числа, например, появились безо всякой связи с нумерациями и вообще теории алгоритмов). Именно рассмотрение этого класса побудило Колмогорова сформулировать общие определения числовой нумерации и сводимости нумераций.

Вот пример алгебраической структуры, имеющей некоторую стандартную нумерацию. Это множество алгебраических вещественных чисел, занумерованное с помощью полиномов с целыми коэффициентами. Отношения равенства и порядка оказываются разрешимыми, а операции сложения и умножения — вычислимыми относительно этой нумерации. По существу, именно такая нумерация описывается в обычных доказательствах счётности множества алгебраических чисел.

Изложенный пример подводит нас к понятию нумерованной, и, далее, конструктивной алгебраической системы. Теория нумерованных и, прежде всего, конструктивных алгебраических систем была основана Мальцевым и развита Ю.Л. Ершовым (см. [Мальцев, 1961], [Ершов Ю., 1973, 1974, 1980], [Гончаров, 1979]). Сейчас эта область активно исследуется в частности Сибирской школой алгебры и логики.

*4. г) Приложения к теории вероятностей: определения случайной последовательности*

Отметим здесь лишь следующее. При рассмотрении бесконечных последовательностей, составленные из букв какого-либо конченого алфавита, наша интуиция каким-то образом выделяет случайные. «Традиционная теория вероятностей оказывается бессильной перед подобной задачей: она не в состоянии определить, что такое индивидуальная случайная последовательность», — пишет Успенский. Теория же алгоритмов как раз даёт возможность

определить индивидуальную случайную последовательность. Эти определения были предложены Чёрчем [Чёрч, 1940], Мизесом, Мартином-Лёфом, Колмогоровым (определение кратко приведены в [Успенский, 1987]. «Определение, которое, по-видимому, можно рассматривать как окончательное» — пишет Успенский об определении случайности по Колмогорову и равносильное ему определение случайности по Мартину-Лёфу, вышедшее в своей основе из теории алгоритмов.

*4. д) Приложения к теории информации*

Алгоритмическая теория информации была основана Колмогоровым (см. [Колмогоров, 1965]) с целью придать таким интуитивным понятиям, как «количество информации» и «энтропия», точный смысл в применении к индивидуальным объектам. В традиционной (основанной на вероятности) теории информации, основанной Шенноном, эти понятия, как известно, применяются к случайным объектам, т.е. говоря более строго, к случайным величинам.

Реальные достижения алгоритмической теории информации относятся к двум направлениям. Первое состоит в выяснении того, насколько формулы, полученные для случайных величин, оказываются справедливыми применительно к индивидуальным объектам. Второе заключается в установлении соотношений между колмогоровской и шенноновской энтропиями.

*4. е) Оценки сложности решения отдельных задач*[5]

В этой области теории алгоритмов естественно выделяются задачи получения верхних и задачи получения нижних оценок. Методы решения задач этих двух категорий совершенно различны.

**Верхние оценки.** Верхняя оценка строится следующим образом. Указывается неформальный алгоритм вычисления требуемой функции *f*. Затем этот алгоритм формализуется в виде алгоритма вычисления на подходящей модели и доказывается, что сложность (время или емкость) вычисления для этого алгоритма не превосходит значения подходящей функции $\varphi$ при всех значениях аргумента. Эта функция и объявляется верхней оценкой сложности вычисления функции *f*.

Естественно желать получать такие оценки сложности вычисления, которые соответствуют вычислительной практике.

С точки зрения времени вычисления самыми простыми являются функции, время (т. е. число шагов) вычисления которых совпадает с точностью до мультипликативной константы с размером аргумента и, более того, число шагов работы машины между двумя последовательными сдвигами входной головки ограничено некоторой константой (мы считаем, что вычисление происходит на многоленточных машинах Тьюринга с безвозвратной входной лентой (см. [Рабин, 1963], [Розенберг, 1967]). Такое вычисление называется *вычислением в реальное время*. Как выяснилось, среди задач, решаемых в реальное время, имеются многие интересные задачи, связанные с идентификацией слов (в частности, задача распознавания симметрии слова, см. [Слисенко, 1977]); правда, для наиболее важных из этих задач построенные алгоритмы не являются алгоритмами колмогоровского типа (см. [Слисенко, 1977а], [Слисенко, 1978]). Представляет интерес переход от таких алгоритмов к алгоритмам Колмогорова.

Следующий класс образуют полиномиальные верхние оценки времени вычисления, т. е. оценки, в которых время ограничено каким-либо полиномом от длины или нормы входа. Большинство получаемых для функций из класса $P$[6] оценок — это оценки «с точностью до», как правило, с точностью до мультипликативной константы. В этом отношении характерны названия публикаций, относящихся к упомянутым примерам: в этих названиях встречаются

[5] параграф целиком взят из [Успенский, 1987, с.220-224].
[6] класс P в данном упоминании определяется как класс всех функций, вычислимых за полиномиальное время.

выражения «алгоритм с линейным временем», «менее чем кубическое время», «полиномиальный алгоритм».

Оценки «с точностью до» иногда порождают парадоксальную ситуацию. Для какой-нибудь задачи дальнейший прогресс в направлении нахождения более «эффективного алгоритма» может состоять в переходе от алгоритма с верхней оценкой времени работы $c_1 n^a$ к алгоритму с верхней оценкой времени работы $c_2 n^b$, где $b<a$. Однако при этом $c_2$ может быть настолько больше $c_1$, что для всех практически мыслимых аргументов старый алгоритм эффективней нового. Поучительна в этом отношении ситуация с одной из важных верхних оценок — оценкой сложности умножения матриц. (Чтобы не загромождать изложение, мы будем говорить о числе арифметических операций, но картина для вычислений на представительной модели колмогоровского типа аналогична.) Классический алгоритм дает оценку $c_1 n^3$ для матриц $n$-го порядка. Алгоритм Штрассена — оценку $c_2 n^{2,81}$ (см. [Ахо, Хопкрофт, Ульман, 1974, п. 6.2]). Анализ работы этого алгоритма показывает, что при некоторой организации вычислений его использование дает выгоду по сравнению с классическим алгоритмом, начиная с матриц 14-го порядка (см. [Фишер, 1974]). В последние годы были предложены алгоритмы, дающие возможность понизить показатель от 2,81 до (приблизительно) 2,5, однако одновременно мультипликативная константа в полученных оценках растет таким образом, что «эффективный» метод оказывается лучше классического только для матриц астрономического порядка (заведомо большего $10^{10}$). Конечно, можно надеяться, что новые методы содержат продуктивные математические идеи, которые в дальнейшем смогут привести к получению действительно осмысленных с прикладной точки зрения алгоритмов.

Что касается емкости вычисления, то с прикладной точки зрения наибольший интерес представляют полиномиальные (от длины аргумента) оценки. Однако если такая оценка не сопровождается полиномиальной же оценкой времени вычисления, то с практической точки зрения и она является сомнительной, ведь полиномиальная оценка емкости автоматически дает лишь экспоненциальную оценку времени. Для многих важных функций их вычисление может быть осуществлено так, что емкость ограничена какой-либо степенью логарифма длины аргумента, а время (при том же вычислении) ограничено полиномом.

По-видимому, верхние оценки сложности, не мажорируемые полиномами (т. е. экспоненциальные и тому подобные оценки) следует рассматривать не как практически значимые, а как (вместе с соответствующими нижними оценками) позволяющие с чисто теоретической точки зрения расклассифицировать решимые алгоритмические проблемы по «степени сложности решения». О такой классификации для алгоритмических проблем теории групп см. [Каннонито, Гаттердам, 1973].

**Нижние оценки.** Как мы уже отмечали, ситуация с нижними оценками принципиально отличается от ситуации с верхними. Установить нижнюю оценку — это значит доказать, что никакой алгоритм вычисления на данной модели не имеет сложности вычисления меньше заданной функции $\varphi$. Основным методом получения нижних оценок является диагонализация (о других подходах см. [Слисенко, 1981, гл. 3, § 2]). Примерами использования метода диагонализации для построения множеств с заданными нижними оценками сложности являются теоремы об иерархии.

Диагональные конструкции были использованы также при получении нижних оценок для сложности разрешения логических теорий (см. [Ферранте, Раков, 1979]). В качестве возможных оценок рассматривались функции от длины формулы. Оказалось, что для многих (можно сказать, для большинства) логических теорий имеет место экспоненциальная нижняя оценка сложности (все равно какой, временной или емкостной) разрешения. А для такой теории, как слабая монадическая теория следования второго порядка, дело обстоит «еще хуже». Названная теория есть теория структуры с носителем $\mathbb{N}$ с единственной сигнатурной функцией — прибавлением единицы, но зато с допущением кванторов не только по натуральным числам, но и по конечным множествам натуральных чисел. Хотя для этой теории и существует

разрешающий (т. е; распознающий истинность формул) алгоритм, в практическом смысле она оказывается неразрешимой: действительно, как показано в [Мейер, 1975], для любого разрешающего алгоритма его (все равно какая) сложность не мажорируется никакой сверхэкспонентой $2^{\cdot^{\cdot^{2^n}}}$ с фиксированным числом этажей. Аналогичный результат имеет место для элементарной теории свободной группы; хотя вопрос о разрешимости этой теории остается открытым, она «практически неразрешима»: для нее не существует разрешающего алгоритма со сверхэкспоненциальной (с фиксированным числом этажей) верхней оценкой времени или емкости (см. [Семенов, 1980]). Популярный очерк решимых массовых проблем, не обладающих, однако, каким бы то ни было «практическим» решающим алгоритмом, см. в [Стокмейер, Чандра, 1979].

В теории сложности все же есть нижние оценки, полученные не диагональным методом. Это, например, квадратичные нижние оценки для задач типа идентификации слов при условии, что в качестве вычислительной модели берется однолентоточная машина Тьюринга. Для задачи распознавания симметрии слов такая оценка найдена в [Барздинь 1965]. Другой пример — это умножение чисел при условии, что очередной разряд результата должен выдаваться «достаточно рано» (см. [Патерсон, Фишер, Мейер, 1974]). Относительно емкости отметим нижнюю оценку для емкости распознавания множеств, не являющихся распознаваемыми с нулевой емкостью. Интересно было бы получить соответствующую оценку для вычисления функций (а не только для предикатов) и, с другой стороны, для емкости порождения (а не только для разрешения) множеств. При этом, конечно, надо соответствующим образом определить эту емкость порождения. С некоторой точки зрения все перечисленные «недиагональные» оценки имеют «негативный» смысл — они показывают, что рассматриваемая вычислительная модель недостаточно универсальна с точки зрения сложности вычисления.

Заметим, наконец, что соображения об оценках «с точностью до», высказанные в связи с верхними оценками, имеют не меньший смысл и в приложении к нижним оценкам.

*4. ж) Влияние теории алгоритмов на алгоритмическую практику*[7]

В настоящее время большинство явно сформулированных и используемых человеком в его деятельности алгоритмов — это программы для ЭВМ (см. [Кнут, 1974], [Кнут, 1974а]). О масштабах «алгоритмической» деятельности человечества важно судить по возникающим в ней организационным проблемам (см. [Брукс, 1975]). Таким образом, программирование — это алгоритмическая практика, теоретическое программирование — это (при широком понимании термина) вся теория алгоритмов (например, теорему Гёделя о неполноте можно рассматривать как теорему теоретического программирования, см. [Глушков,1979]).

Общепринято, однако, другое понимание термина «теоретическое программирование» — как области теории алгоритмов, концентрирующейся вокруг взаимоотношения программы как чисто синтаксического, неинтерпретированного объекта и содержания (смысла, значения) программы. Конечно, для теоретического программирования характерен интерес к порожденным практикой темам, которых не каялась классическая теория алгоритмов, таким как параллельное программирование (см. [Котов, 1974]) или структуры данных (см. [Скотт, 1970]). Тем не менее «общая часть» теоретического программирования и теории алгоритмов велика, и классическая теория алгоритмов оказала бесспорное влияние на программирование. Это влияние, однако, состояло не в использовании каких-либо теорем, оно носило скорее идейный характер. Попытаемся проследить, как оно происходило, перечислив соответствующие результаты и понятия общей теории алгоритмов.

**Общее понятие алгоритма и возможность его формализации.** В вычислительной практике важную роль сыграло осознание того, что любая вычислительная машина (если игнорировать

[7] параграф целиком взят из [Успенский, 1987, с.224-230].

физические ресурсы) может вычислять любую вычислимую функцию и никакая машина не может вычислять невычислимую. Также важную, хотя и не всегда однозначно полезную роль играло утверждение о том, что все задачи, решаемые человеком, могут быть решены подходящими алгоритмами, в частности на ЭВМ.

**Существование нерешимых алгоритмических проблем в математике и нерешаемость многих естественно возникших проблем.** О некоторых задачах стало заранее известно, что искать их полное и точное решение безнадежно и нужно вырабатывать реалистический подход, основанный на отказе от полноты, абсолютной достоверности или чего-то еще. Конечно, разделение задач на решимые и нерешимые имело и отрицательные последствия, возникло искушение рассматривать всякую решимую задачу как практически решаемую, почти решенную, если не сегодня, то при дальнейшем прогрессе вычислительной техники.

**Появление различных понятий сложности вычисления и порождения.** Возможность строгого абстрактного определения того, что такое сложность вычисления, стимулировало разработку эффективных алгоритмов и дало возможность их объективного сравнения (см. [Слисенко, 1981]). Большое практическое значение имело определение класса и доказательство полиномиальной эквивалентности «переборных» задач. Это, наряду с экспоненциальными нижними оценками, разрушило иллюзию, о которой говорилось в предыдущем абзаце; выяснилось, что одного только существования алгоритма, решающего ту или иную массовую проблему, далеко не достаточно для практики. Тем самым еще раз подтвердилась важность нестандартных (эвристических, приближенных и т. д.) подходов (см. [Рабин, 1974]). С другой стороны, теоретические алгоритмы для которых были доказаны «хорошие» полиномиальные верхние оценки сложности, нашли практические приложения.

**Неалгоритмическое описание вычислимых функций** (μ-рекурсивные функции, исчисление Эрбрана — Гёделя, исчисление λ-конверсии, неподвижные точки вычислимых операторов и т. д.; обзор и классификацию таких описаний см. в [Ершов А., 1982а]). Неалгоритмическое (непроцедурное) описание вычислимых функций оказалось важным средством программирования. Начиная с языка ЛИСП, соответствующие конструкции вошли во многие языки программирования, кроме того, неалгоритмическое описание является основой многих формальных определений семантики программ.

**Вычислительные и порождающие модели.** Основную роль в программировании играют не сами представительные модели, а их (уже непредставительные) модификации и ограничения. Типичные примеры таких ограничений — магазинные автоматы и контекстно-свободные грамматики. Контекстно-свободные грамматики широко использовались для задания синтаксиса языков программирования, начиная с алгола-60; при описании алгола-68 понадобился более общий вид исчислений — грамматики ван Вейнгаардена.

Именно непредставительность, ограниченность моделей и позволяет в какой-то степени использовать их при создании эффективных алгоритмов обработки программ, прежде всего алгоритмов трансляции. Разнообразие различных представительных вычислительных моделей, которое с точки зрения общей теории алгоритмов может показаться излишним, оказывается весьма осмысленным с практической точки зрения: ведь для практики вопрос о степени удобства, с которой тот или иной алгоритм записывается на языке программирования, оказывается жизненно важным. Рост числа языков программирования в конце 60-х — начале 70-х годов (речь идет о сотнях и даже тысячах языков, см. [Семенов, Семенова, 1974]) позволял говорить даже о «вавилонской башне» в программировании.

**Трактовка программ как объектов вычисления.** Фон Нейман был первым, кто ввел это фундаментальное укрытие теории алгоритмов в алгоритмическую практику (см. [Нейман, 1963]). Принцип хранимой и модифицируемой программы стал одной из основ системного программирования. Неотъемлемыми частями каждой ЭВМ являются компилятор и другие компоненты операционной системы, ориентированные на модификацию и выполнение программ потребителей. Машина с действующим интерпретатором есть в точности

универсальный алгоритм в смысле теории алгоритмов. Дальнейшим развитием этих идей явилась концепция смешанного вычисления (см. [Ершов А., 1982]), позволяющая с единой точки зрения взглянуть на многие кажущиеся различными способы обработки программ и данных.

**Рассмотрение программ как объектов порождения** (как и результаты о логических исчислениях) стимулировало развитие формальных систем, предназначенных для доказательства утверждений о программах (см. [Хоар, 1969], [Непомнящий, 1979]). Важнейший класс таких утверждений образуют утверждения о так называемой правильности программ, т. е. о том, что рассматриваемая программа «делает, что надо».

**Смешанные вычисления.** Поучительным примером взаимодействия теории алгоритмов и вычислительной практики является развитая А. П. Ершовым и его учениками теория смешанных вычислений [Ершов, 1982], [Ершов, 1984]. Мы коснемся самого начала этой теории. Начнем с двух замечаний.

1. Пусть аргументами алгоритма служат пары $\langle x, y \rangle$. Тогда фиксация $x$ дает алгоритм с аргументом $y$.
2. Программа алгоритма, получаемого фиксацией $x$, может быть эффективно построена по программе первоначального алгоритма, если используемый способ программирования — главный.

Эти два замечания содержательной теории алгоритмов составляют так называемую *s-m-n*-теорему. Отметим, что фиксация одного члена пары может рассматриваться как типичный пример сужения области исходных данных. Получаемый при этом алгоритм называется специализацией исходного.

*s-m-n*-теорема может рассматриваться как теоретическое подтверждение идеи автоматизации программирования. Действительно, как мы видим, имеется автоматический способ получать из общего алгоритма его специализации. Заметим, однако, что формулировка теоремы не дает конкретного рецепта специализации. Попытка же извлечь алгоритм специализации из обычного доказательства *s-m-n*-теоремы, конечно, не приводит к удовлетворительному результату. Это доказательство гласит: «Вот требуемая программа: образуй из $x$ и $y$ пару и примени к ней первоначальный алгоритм». В то же время содержательно понимаемая цель специализации состоит в получении алгоритма, более эффективного чем первоначальный, общий, от двух аргументов. Смысл ограничения области аргументов как раз и состоит в, получении более эффективного (т. е. более экономного по времени и емкости, с более короткой программой и т. д.) алгоритма.

Чтобы найти теоретические основания для специализации, приводящей к более эффективным алгоритмам, уже недостаточно ограничиваться качественной теорией алгоритмов. Нужно фиксировать какой-либо язык программирования. Возьмем, например, простой естественный язык, в котором программы строятся из элементарных команд и проверок с помощью основных конструкций структурного программирования: последовательного выполнения, разветвления и повторения. С текстом программы на таком языке (конструктивным объектом) можно выполнять различные операции. Можно, скажем, подставить в этот текст известное значение входной переменной, заметить, что в силу ложности какой-либо проверки некоторую команду не понадобится выполнять и выкинуть эту команду. Другой вариант — все аргументы функции оказались известны, можно вычислить значение функции и упростить выражение, ее содержащее. Операции над текстом программы, о которых мы сейчас говорили: устранение фиктивной ветви вычисления и подстановка значения функции, относятся к числу так называемых редукций. Как мы видим, при редукции текст программы в каком-то смысле, упрощается. С другой стороны, программа может содержать конструкцию повторения. В этом случае бывает полезно преобразовать текст команды повторения в более длинный — развернуть его, выделив первое повторение, за которым опять пойдет команда повторения. Усложнение текста программы при развертывании может компенсироваться последующей редукцией.

Процессы редукции и развертывания задают очень простое и естественное исчисление со входом, причем на вход подается первоначальная программа с заданными значениями некоторых переменных. Процесс специализации представляется как вывод в исчислении. Результатом работы исчисления со входом является специализация исходной программы. В частном случае специализация может состоять из одного объекта — результата работы программы.

Всякий процесс перехода от алгоритма к его специализации с использованием описанных операций редукции и развертывания называется *смешанным вычислением.* Единое рассмотрение процессов вычисления, компиляции (перевода на машинный язык) и выполнения программ явилось одним из мощных принципов современного программирования, в какой-то мере позволяющим устранить границу между человеческим — «предэкранным» миром и машинным — «заэкранным» [Кушниренко и др., 1985], [Бетелин, 1985].

**Методы программирования.** Здесь имеются в виду методы построения алгоритмов и доказательства их правильности, появившиеся внутри теории алгоритмов. Наиболее показателен в этом отношении пример структурированного программирования. Основные операторы образования структурированных программ (последовательное выполнение, разветвление, повторение) были введены в начале 50-х годов при описании нормальных алгорифмов Маркова (см. [Марков, 1954, гл. III], [Нагорный, 1977]). Одновременно были даны нетривиальные примеры индуктивного доказательства правильности программ, построенных с помощью этих операторов (в частности, программы универсального алгоритма, т. е. интерпретатора). В абстрактной, алгебраической форме операторы структурированного программирования были введены в системах алгоритмических алгебр Глушкова; одновременно были рассмотрены содержательные с точки зрения практического программирования примеры преобразования программ и доказательства их правильности (см. [Глушков, 1965]). В 70-е годы структурированное программирование стало одним из инструментов практического программиста. Показательные примеры взаимодействия алгоритмической теории и алгоритмической практики в области методов и языков программирования прослеживаются в [Ершов А., 1977], [Лавров С, 1984], [Тыугу, 1984].

**Программирование как вторая грамотность.** «...Мы сами живем в мире программ, подчас не сознавая того» [Ершов А., Звенигородский, 1979, с. 47]. Убыстряющееся развитие вычислительной техники и программирование будет включать в алгоритмическую практику (которая, в свою очередь, будет составлять все большую часть разумной деятельности человека) все более широкий круг идей теории алгоритмов (и требовать от теории алгоритмов новых открытий).

Алгоритмические концепции играют в процессе обучения и воспитания современного человека фундаментальную роль, сравнимую лишь с ролью письменности (отсюда предложенный А. П. Ершовым термин «вторая грамотность» см. [Ершов А., Звенигородский, 1979], [Ершов А., 1981а]). Возможность обучения учащихся начальной школы программированию на базе абстрактных вычислительных моделей (не только для подготовки их к будущей профессии, но и для развития способности к формальному мышлению) обсуждается в [Успенский, 1979, гл. 1, §5].

Согласно [Ершов А., Звенигородский, 1979, с. 48], «программирование... есть способность выразить любой «правильный» процесс средствами, доступными для передачи машине», т. е. прежде всего алгоритмическими средствами. Поэтому «мы естественно приходим к проблеме фундаментализации программирования, выделения в нем некоторых «натуральных» сущностей» [Ершов А., 1981а, с. 81]. Система «натуральных сущностей» программирования формируется под сильным влиянием системы основных понятий теории алгоритмов и исчислений (впрочем, и эта последняя система испытывает влияние первой). «Сумма знаний по этим вопросам должна подвергнуться тщательному концептуальному анализу и в объединении

с математическими и лингвистическими концепциями стать фундаментальной компонентой общего образования» [Ершов А., 1981а, с. 18].

**Математическая логика и вычислительная техника.** Один из самых выдающихся программистов нашего времени Э. Дейкстра в статье [Дейкстра, 1986] называет блестящим предвидением следующее высказывание другого известного специалиста, автора языка ЛИСП Дж. Маккарти, сделанное еще в 1967 г.: «Разумно ожидать, что связи между вычислительной техникой и математической логикой окажутся столь же плодотворными в следующем столетии, какими были связи между математическим анализом и физикой в столетии предыдущем». Еще одна попытка анализа этих связей предпринята в [Семенов, Успенский, 1986].

## Заключение

Как мы увидели, понятие алгоритма всегда неявно присутствовало в математике с самых её начал – арифметики. Однако сколько же лет прошло от изобретения первого алгоритма – нахождения НОД методом Евклида, до формального осмысления понятия «алгоритм»? Два тысячелетия! Такой срок удивляет. И, словно бы догоняя, теперь теория алгоритмов настолько быстро и интенсивно развивается, что успела проникнуть в самые основы математики. Также она стала родителем такой области деятельности ныне, как программирование, без которой нельзя даже помыслить дальнейшее развитие человечества.

## Основная литература


[Успенский, 1987] УСПЕНСКИЙ В.А., СЕМЁНОВ А.Л., Теория алгоритмов: основные открытия и приложения. М.: «Наука», 1987.

[Юшкевич, 1970] История математики с древнейших времён до начала XIX столетия. / Под ред. А.П. ЮШКЕВИЧА. Т.1, 2, М.: «Наука», 1970.


## Дополнительная литература[8]

**к 1-й части**


[Бобынин В.В.]
(1882) «Математика у древних египтян». (По папирусу Ринда). — М., 1882 г.

[Нейгебауэр О.]
(1968) Точные науки в древности. М., 1968.

[Прокл] PROCLUS DIADOCHUS. In primum Euclidis Elementorum librum commentarii. Ed. C.Friedlein. Leipzig, 1873.


**ко 2-й части**


[Депман И.Я,]
(1965) История Арифметики. М.: «Просвещение», 1965.


**к 3-й и 4-й частям**


[Адлер А.]
(1969) ADLER A. Some recursively unsolvable problems in analysis // Proceedings of AMS. — V. 22, 1969, № 2. — P.523-526.

[Арнольд В.И.]
(1976) ALNOLD V.I. Dynamic systems and differential equations (B), P.59 // Mathematical developments arising from the Hilbert problems Ed. F,E,Browder. — Providence: AMS, 1976 — 628 p.

[Ахо А., Хопкрофт Дж, Ульман Дж.]
(1974) Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. — 536с.

[Барздинь Я.М.]
(1965) Сложность распознавания симметрии на машинах Тьюринга. // ПК. — Вып. 15., 1965. — С.245–248.

[Бетелин В.Б.]
(1985) О проблеме автоматизации программирования. // ДАН. — Т.286, 1985, № 1. — с.66–69.

[Борель]
(1912) BOREL E. Le calcul des integrales definies // Journal de Mathematiques pures et appliquees. Ser. 6. — V.8, № 2. 1912. — P.159–210.

[Брукс Ф.П.]
(1975) Как проектируются и создаются программные комплексы. — М.: Наука, 1979. — 151с. (Оригинал: Brooks F.P., Jr. The mythical man-month. — AW, 1975)

[Гёдель К.]
(1931) GODEL K. On formally undecidable propositions of Principia Mathematica and related systems, p.596–616. // in book by van Heijenoort J. From Frege to Godel: A source book in mathematic logic, 1879—1931. — Cambridge, Mass.: Harvard University Press. — XII, 1967

[Гильберт Д.]
(1935) Математические проблемы // Проблемы Гильберта / Под. ред. Александрова П.С. — М.: Наука, 1969. — с.11–64.


[8] Далее приняты такие сокращения:
Перед названием статьи в скобках указан год первой публикации.
АиЛ —Алгебра и логика.
ДАН — Доклады Академии наук СССР.
МЭ — Математическая энциклопедия.
УМН — Успехи математических наук.
AMS — the American Mathematical Society
AW — Reading, Massachusetts, etc. : Addison — Wesley Publishing Company
NH — Amsterdam: North Holland Publishing Company

[Глушков В.М.]
(1965) Теория автоматов и формальные преобразования микропрограмм. // Кибернетика. — № 5. — с.1–9.
(1979) Теорема о неполноте формальных теорий с позиций программиста. // Кибернетика. — № 2. — с.1–5.
[Гончаров С.С.]
(1979) Конструктивных моделей теория. // МЭ. Т.2, 1979. — с.1058–1060.
[Дейкстра Э.В.]
(1986) DIJKSTRA E.W. On a cultural gap. // The mathematical intelligencer. — V.8, 1986, № 1. — p.48–52.
[Ершов А.П.]
(1977) Введение в теоретическое программирование: Беседы о методе. — М.: Наука. — 288 с.
(1981а) Программирование — вторая грамотность. — Препринт / ВЦ СО АН СССР. — Новосибирск, 1981. — № 293. — 18 с.
(1982) ERSHOV A.P. Mixed computation: potential applications and problems for study // Theoretical computer science. — V. 18, № 1 — p.41–67.
(1982а) Вычислимость в произвольных областях и базисах // Семиотика и информатика. — М.: ВИНИТИ. — Вып. 19. — с.3–58.
(1984) Смешанные вычисления. // В мире науки. — № 6 (июнь). — с.28–42.
[Ершов А.П., Звенигородский Г.Г]
(1979) Зачем нужно уметь программировать? // Квант. — 1979, № 9. — с.47–51.
[Ершов Ю.Л.]
(1964) Разрешимость элементарной теории дистрибутивных структур с относительными дополнениями и теории фильтров. // АиЛ. — Т.3, вып. 3. — с.17–38.
(1964а) Неразрешимость теорий симметрических и простых конечных групп // ДАН. Т. 158, № 4. — с.777–779.
(1973) Конструктивные модели // Избранные вопросы алгебры и логики. — Новосибирск: Наука. — с.111–130.
(1974) Теория нумераций. Ч. 3: Конструктивные модели. — Новосибирск. — 139 с. — (Библиотека кафедры алгебры и математической логики Новосибирского университета; Вып. 13.)
(1980) Проблемы разрешимости и конструктивные модели. — М.: Наука. — 415 с.
[Ершов Ю.Л, Лавров И.А., Тайманов А.Д., Тайцлин М.А.]
(1965) Элементарные теории // УМН. — Т.20, вып. 4 (124). — с.37–108.
[Каннонито Ф.Б., Гаттердам Р.В.]
(1973) CANNONITO F,B, GATTERDAM R.W. The computability of group constructions. I // Words problems: Decision problems and the Burnside problem in group theory. / Eds. W.W.Boone, F.B. Cannonito, R.C.Lyndon. — NH.— p.365–400.
[Кнут Д.Э.]
(1974) Информатика и её связь с математикой // Современные проблемы математики. — М.: Знание, 1977. — с.4–32. — (Новое в жизни, науке, технике. Серия «Математика, кибернетика», 1977, №12) (Оригинал опубликован в 1974 г.)
(1974а) KNUTH D.E. Computer programming as an art // Communications of ACM. — V.17, № 12. — p.667–673.
[Колмогоров А.Н.]
(1953) О понятии алгоритма // УМН. — Т.8, вып. 4 (56). — с.175–176.
(1965) Три подхода к определению поняти «количество информации» // Проблемы передачи информации. — Т. 1, вып. 1. — с.3–11.
[Котов В.Е.]
(1974) Теория параллельного программирования: прикладные аспекты // Кибернетика. — №1. — с.1–16; № 2. — с.1–18.
[Кушнер Б.А.]
(1973) Лекции по конструктивному математическому анализу. — М.: Наука. — 448 с.
(1979а) Конструктивный анализ // МЭ. — Т. 2. — с.1054–1057.
[Кушниренко А.Г. и др.]
(1985) Практическое программирование. Проектирование и разработка диалоговых систем. Нетрадиционный подход. / А.Г. Кушниренко. Д.В. Варсанофьев, А.Г. Дымченко, Г.В. Лебедев. — М.: Изд-во Моск. ун-та. — 89 с.
[Лавров С.С.]
(1984) Методология программирования. // Семиотика и информатика. — Вып. 23. — М.: ВИНИТИ. — с.5–26.
[Мальцев А.И.]
(1961) Конструктивные алгебры, I // УМН. — Т.16, вып. 3 (99). — с.3–60.
(1962) Аксиоматизируемые классы локально свободных алгебр некоторых типов // Сибирский математический журнал. — Т.3, № 5. — с. 729–743.

[Марков А.А.]
(1951) Теория алгорифмов. // Труды МИАН. — Т. 38. — с.176–189.
(1954) Теория алгорифмов. — М. — Л.: Изд-во АН СССР. — 37 с. — (Труды МИАН. Т. 42).
(1957) Математическая логика и вычислительная математика // Вестник Академии наук СССР. №8. — с.21–25.
(1958а) Неразрешимость проблемы гомеоморфии // ДАН. — Т. 121, №2. — с.218–220.
(1958в) Неразрешимость проблемы гомеоморфии // УМН. — Т. 13, вып.4 (82). — с.213–216.
[Матиясевич Ю.В.]
(1973) MATIJASEVIC YU.V. On recursive unsolvability of Hilbert's tenth problem // in Logic, Methodology and Philosophy of Science. IV / Eds. P. Suppes et al. — NH. — p.89–110.
(1974) Эффективные и неэффективные методы в теории чисел // ВКМЛ-3. — с.141–142.
[Мейер А.Р.]
(1975) MEYER A.R. Weak monadic second order theory of successor is not elementary-recursive // Logic colloquium, Boston, 1972 –1973 / Ed. R. Parikh. — Springer. (LN in mathematics. V.453). p. 132–153.
[Мучник А.А.]
(1958) Решение проблемы сводимости Поста и некоторых других проблем теории алгоритмов. I // Труды ММО. — М.: Физматгиз. — Т.7. — с.391—405.
[Нагорный Н.М.]
(1977) Алгоритмов сочетание // МЭ. Т.1. — с.225–226.
[Нейман Дж., фон]
(1963) Вычислительная машина и мозг. // Кибернетический сборник: Сб. переводов / Под. ред. А.А. Ляпунова, О.Б. Лупанова. — М.: ИЛ, 1960. — №1. — с.11–60.
[Непомнящий В.А.]
(1979) Практические методы проверки правильности программ. // Семиотика и информатика. — М.: ВИНИТИ. — Вып. 12. — с.86–87.
[Патерсон М.С., Фишер М.Дж., Мейер А.Р.]
(1974) Улучшенный метод частичного перекрытия для умножения, выполняемого в темпе поступления информации // КС, 1977. — Вып. 14. — с.77–94.
[Пост Э.Л.]
(1936) Финитные комбинаторные процессы, формулировка 1 // [Успенский, 1979] с.89–95.
[Рабин М.О.]
(1963) Вычисления в реальное время. с.156–167.// Проблемы математической логики: Сложность алгоритмов и классы вычислимых функций: Сб. переводов / Под.ред. В.А. Козмидиади, А.А.Мучника. — М.: Мир, 1970. —432 с. — (библиотека «Кибернетического сборника).
(1969) Разрешимость теорий второго порядка и автоматы над бесконечными деревьями // КС. — Вып. 8. — с.72–116.
(1974) Rabin M.O. Theoretical impediments to artificial intelligence // Information processing 74. Proceedings of IFIP congress 1974, Stockholm, August 3–10, 1974 / Ed. J.L.Rosenfeld. — NH. — p.615–619.
[Робинсон Дж.]
(1949) Robinson J. Definability and decision problems in arithmetic // JSL. — V.14, №2. — p.98–114.
[Роджерс Х., младший]
(1958) Rogers H., Jr. Godel numberings of partial recursive functions // JSL. — V. 23, №3. — p.331–341.
(1967) Теория рекурсивных функций и эффективная вычислимость. — М.: Мир., 1972. — 624 с.
[Розенберг А.Л.]
(1967) Языки, определимые в реальное время. с.168–193. // Проблемы математической логики: Сложность алгоритмов и классы вычислимых функций: Сб. переводов / Под.ред. В.А. Козмидиади, А.А.Мучника. — М.: Мир, 1970. —432 с. — (библиотека «Кибернетического сборника).
[Санкаппанавар Х.П.]
(1978) SANKAPPANAVAR H. P. Decision problems: History and methods // Mathematical logic. Proceedings of the first Brasilian conference / Eds. A. I. Arruda, C A. Newton da Costa, R. Chuaqui. — New York and Basel: Marcel Dekker.— (Lecture notes in pure and applied mathematics. V. 39).—P. 241—291.
[Семенов А.Л.]
(1980) Интерпретация свободных алгебр в свободных группах // ДАН.— Т. 252, № 6.— С. 1326—1332.
(1984) SEMENOV A.L. Decidability of monadic theories. // Mathematical foundations of computer sciences 1984: Proceedings, 11th Symposium (Praha, Czechoslovakia, September 3—7, 1984).— Springer.— (LN in computer science; V. 176).— P. 162—175.
(1986) Разрешающие алгоритмы для логических теорий. // Кибернетика и вычислительная техника.— Вып. 2.— М.: Наука.— С 134—146.
[Семенов А.Л., Семенова Е.Т.]
(1974) Программирование и математическое обеспечение // Радиоэлектроника в 1973 г.: Обзор по материалам иностранной печати.— Вып. 6; Вычислительная техника. Программирование.—

М.: Научно-исследовательский институт экономики и информации по радиоэлектронике.— с.76—91.

[Семенов А.Л., Успенский В.А.]

(1986) Математическая логика в вычислительных науках и вычислительной практике. // Вестник Академии наук СССР, № 7.— С.93—103.

[Скотт Д.]

(1970) Набросок математической теории вычислений // КС.— 1977.— Вып. 14.— С.105—121.

[Слисенко А.О.]

(1977) Упрощенное доказательство распознаваемости симметричности слов в реальное время на машинах Тьюринга // ТПММЛ. II.— Л.: Наука,— (Записки НС ЛОМИ. Т. 68.).— С.123—139.

(1977а) Распознавание предиката вхождения в реальное время.— Препринт / Ленинградское отделение МИАН.— Л., 1977.— № Р7— 77.— 24 с.

(1978) S1ISENKO A. O. String-matching in real time: some properties of the data structure *Ц* Mathematical foundations of computer science 1978/Ed. J. Winkowski.—Springer,—(LN in computer science. V. 64).— P.493—496.

(1981) Сложностные задачи теории вычислений // УМН.— Т.36, вып.6.—С.21—103.

[Стокмейер Л. Дж., Чандра А. К.]

(1979) STOCKMEYER L. J., CHANDRA A. K. Intrinsically difficult problems. // Scientific American.— V. 240, №5.—P.124—133.

[Тарский А., Мостовский А., Робинсон Р.М.]

(1953) TARSKI A., MOSTOWSKI A., ROBINSON R. M. Undecidable theories.— NH.— XI + 98 p.

[Тьюринг А.М.]

(1936) TURING A.M. On computable numbers, with an application to the Entscheidungsproblem // Proceedings of LMS. Ser. 2. — V.42, №3, 4. — P.230—265.

(1937) On computable numbers, with an application to the Entscheidungsproblem. A correction // Proceedings of LMS. Ser. 2.— V. 43, $s 7.— P. 544—546.

(1937а) Computability and λ-definability // JSL— V. 2, № 4.— P. 153—163.

[Тыугу Э.Х.]

(1984) Концептуальное программирование.— М.5 Наука.— 255 с.

[Успенский В.А.]

(1979) Машина Поста.— М.: Наука.— 95 с.

[Ферранте Дж., Раков Ч.]

(1979) FERRANTE J., RACKOFF C. W. The computational complexity of logical theories.— Springer.— 243 p. (LN in mathematics. V.718).

[Фишер П.]

(1974) FISCHER P. C. Further schemes for combining matrix algorithms // Automata, languages and programming: 2nd colloquium (Saarbrucken, July 29 —August 2, 1974).—Springer.—P.428—436.— (LN in computer science. V.14.)

[Фреге Г.]

(1879) FREGE G. Begriffsschrift, a formula language, modeled upon that of arithmetic, for pure thought// in book by van Heijenoort J. From Frege to Godel: A source book in mathematic logic, 1879—1931. — Cambridge, Mass.: Harvard University Press. — XII, 1967.— P.1–82.

[Хоор К.А.Р.]

(1969) HOARE C.A.R. An axiomatic basis for computer programming // Communications of ACM.—V. 12, № 10.—P.576–580, 583.

[Чёрч А.]

(1936) CHURCH A. An unsolvable problem of elementary number theory / American journal of mathematics.— V. 58, № 2.— P.345–363.

(1940) On the concept of a random sequence // Bulletin of AMS.— V. 46, № 2.— P.130–135.

[Яновская С.А.]

(1962) Исчисление // Философская энциклопедия. Т. 2.— С. 387—390.

## Интернет-источники

1. Типологические особенности арифметики Древнего Египта и Месопотамии (http://www.egypt-info.ru/about/sience/ancient_sience/arithmetics.html)
2. Достижения Древних Египтян в Математике (http://www.egypt-info.ru/about/sience/ancient_sience/mathematics.html)
3. Гриценко Дарья, Математика в древнем Египте. (http://konkurs.dtn.ru/cgi-bin/end_text.pl?/referats/vostok/1/14_2002.html)