Черемушкин Александр Васильевич

ЛЕКЦИИ ПО АРИФМЕТИЧЕСКИМ
АЛГОРИТМАМ В КРИПТОГРАФИИ

Научный редактор А. Б. Пичкур
Технический редактор В. Шувалов

Издательство Московского Центра
непрерывного математического образования

Лицензия ИД № 01335 от 24.03.2000 г.

Подписано в печать 11.11.2002 г. Формат 60×88 1/16. Печать офсетная.
Бумага офсетная № 1. Печ. л. 6,5. Тираж 2000 экз. Заказ №

МЦНМО
119002, Москва, Большой Власьевский пер., 11.

Отпечатано в ФГУП «Производственно-издательский комбинат ВИНИТИ».
140010, г. Люберцы Московской обл., Октябрьский пр-т, 403. Тел. 554-21-86.

А. В. Черемушкин

Лекции по арифметическим алгоритмам в криптографии

*Допущено Учебно-методическим объединением вузов
по образованию в области информационной безопасности
в качестве учебного пособия для студентов,
обучающихся по специальности «Компьютерная безопасность»*

МЦНМО
2002

ББК 32.81в6
Ч46

Рецензенты: Тювин Ю. Д., проректор Московского государственного института радиотехники, электроники и автоматики (технического университета), председатель учебно-меодической комиссии по вопросам компьютерной безопасности, кандидат техн. наук, профессор.

Зязин В. П., профессор Московского государственного института радиоэлектроники и автоматики (технического университета), кандидат физ.-мат. наук.

Лось А. Б., заведующий кафедрой информационной безопасности Московского государственного института электроники и математики (технического университета), кандидат техн. наук.

Черемушкин А. В.

Ч46 Лекции по арифметическим алгоритмам в криптографии. — М.: МЦНМО, 2002. — 104 с.

ISBN 5–94057–060–7

Пособие представляет собой краткое введение в область современной вычислительной теории чисел и ее приложений к криптографическим задачам.

Предназначено для студентов вузов, обучающихся по информационной безопасности, и всех желающих получить первоначальное представление о предмете.

ББК 32.81в6

ISBN 5–94057–060–7

Оглавление

Предисловие

В основу книги положены лекции, читавшиеся в течение 1994—2000 гг. в Институте криптографии связи и информатики на потоке «Компьютерная безопасность». Цель этой книги — привести с возможно более полными доказательствами начальные результаты в области современной вычислительной теории чисел и ее приложений к криптографическим задачам.

От вышедших на русском языке книг по теории чисел и ее приложениям к криптографии данный курс отличает компактность и простота изложения. При выборе материала автор стремился исходить из минимальных требований к начальной подготовке читателя, как правило, соответствующей двум курсам технического вуза. Поэтому в книгу не вошли методы, дающие наилучшие из существующих в настоящее время оценок сложности и требующие привлечения понятий современной алгебраической геометрии.

Для дальнейшего более детального ознакомления с предметом можно рекомендовать книги [9], [10], [13], [15].

Книга соответствует программе дисциплины «Теоретико-числовые методы в криптографии» государственного образовательного стандарта по специальности 075200 — «Компьютерная безопасность».

Автор благодарен Круглову И. А. и Семаеву И. А. за предоставленные материалы, а также Пичкуру А. Б. и Зязину В. П. за многочисленные замечания.

I. Оценка сложности арифметических операций

§ 1. Свойства функций оценки сложности

1.1. Понятие сложности алгоритма. Алгоритм решения вычислительной задачи представляет собой некую детерминированную процедуру, выполняемую над набором входных данных. В качестве оценки сложности алгоритма обычно выбирается время работы алгоритма. При разных наборах данных алгоритм может иметь различное время работы. Поэтому, под сложностью алгоритма понимается максимальное время работы для всех наборов данных из некоторого фиксированного множества. Всюду ниже в качестве множества исходных данных для алгоритма будет выступать множество чисел, которые в некоторой позиционной системе счисления (чаще всего двоичной) имеют длину записи не превосходящую некоторого числа n. Поэтому, сложность алгоритма оценивается некоторой функцией $f(n)$.

Под сложностью самой вычислительной задачи обычно понимается минимальная сложность алгоритма, решающего данную задачу. Поскольку практически невозможно описать множество всех алгоритмов, решающих конкретную задачу, то данный подход позволяет получать только верхние оценки сложности задачи. Функция сложности каждого алгоритма будет являться верхней оценкой сложности рассматриваемой задачи. Чем более эффективный алгоритм, тем получается более точная верхняя оценка сложности задачи.

Мы будем получать оценки сложности алгоритмов в виде $f(n) = O(g(n))$, т. е. определять функции с точностью до константного сомножителя, пренебрегая при этом малыми членами в выражении для функции $g(n)$. В связи с этим удобно ввести следующее обозначение.

Будем использовать запись $f(n) \prec g(n)$, если $f(n) = O(g(n))$, то есть найдется константа $c > 0$ такая, что начиная с некоторого n выполняется неравенство $f(n) \leqslant cg(n)$. Если одновременно выполняются неравенства $f(n) \prec g(n)$ и $g(n) \prec f(n)$, то будем использовать запись $f(n) \approx g(n)$.

1.2. Модель вычислений. Будем считать, что числа представлены в двоичной системе счисления, а под сложностью арифметической операции будем понимать число битовых операций, необходимых для ее реализации.

Данный подход удобен по следующим соображениям. Во-первых, в компьютерах данные представляются и обрабатываются в двоичном виде, а другие представления в основном используются только при вводе данных и выводе результатов. Во-вторых, перевод числа N от одного основания системы счисления к другому осуществляется быстро и требует выполнения $O(\log N)$ арифметических операций (деление с остатком, умножение или сложение чисел), где $\log N$ — длина записи числа N (основание логарифма не указываем, так как оно не влияет на вид оценки сложности). А так как переход к другому основанию является редкой операцией, то затратами на ее выполнение можно пренебречь. Наконец, битовая оценка достаточно хорошо отражает реальную сложность операций, поскольку, как правило, оценки сложности для других оснований систем счисления приводят лишь к отличиям в константном множителе функции оценки сложности.

1.3. Оценки функции сложности. При оценке сложности вычислительных задач обычно применяют методы редукции и сведения к другим задачам с известными оценками сложности. Эти методы основаны на построении некоторого алгоритма решения данной задачи, который заключается в разбиении задачи на подзадачи меньшей размерности, для решения которых может использоваться как данный, так и другие известные алгоритмы. При этом для функций сложности получатся неравенства вида

$$f(n) \leqslant cf\left(\frac{n}{a}\right) + bg(n) + dn,$$

где a, b, c и d — некоторые положительные константы.

Будем предполагать, что функции сложности $f(n)$ обладают двумя свойствами:

(a) $f(n) \geqslant n$;

(b) $kf(n) \leqslant f(kn)$, $k > 1$.

Для получения оценок функций сложности важны две леммы.

Лемма 1. *Если функции $f(n)$ и $g(n)$ удовлетворяют свойствам* (a) *и* (b) *и при некоторых константах $a > 1$, $b > 0$ и $d > 0$ выполняется неравенство*

$$f(n) \leqslant f\left(\frac{n}{a}\right) + b \cdot g(n) + d \cdot n,$$

то $f(n) = O(g(n))$.

Доказательство. Так как в силу свойства (b)

$$a \cdot f\left(\frac{n}{a}\right) \leqslant f\left(a \cdot \frac{n}{a}\right) = f(n),$$

то

$$f(n) \leqslant \frac{1}{a} f(n) + b \cdot g(n) + d \cdot n.$$

Отсюда получаем, что

$$f(n) \leqslant \frac{b+d}{1-1/a} \cdot g(n) = O(g(n)). \quad \square$$

Лемма 2. *Если при некоторых константах $a > 0$, $c > 0$ и $d > 0$ функция $f(n)$ удовлетворяет условию $f(1) = d$ и $f(n) = c \cdot f(\frac{n}{a}) + d \cdot n$ при $n > 1$, то она имеет вид*:

$$f(n) = \begin{cases} O(n), & a > c, \\ O(n \log n), & a = c, \\ O(n^{\log_a c}), & a < c. \end{cases}$$

Доказательство проведем только для случая $n = a^t$. Имеем

$$f(n) = d \cdot n \cdot \sum_{i=0}^{t-1} \left(\frac{c}{a}\right)^i + d \cdot c^t = d \cdot n \cdot \sum_{i=0}^{t} \left(\frac{c}{a}\right)^i,$$

так как $n = a^t$. Отсюда получаем:

$$f(n) \leqslant \frac{d \cdot n}{1 - c/a} = O(n), \qquad \text{если } a > c,$$

$$f(n) = d \cdot n \cdot t = d \cdot n \cdot \log n, \quad \text{если } a = c, \quad \text{и}$$

$$f(n) = d \cdot n \cdot \frac{(c/a)^{t+1} - 1}{c/a - 1} \leqslant \frac{c \cdot d \cdot n}{a} \cdot \frac{c^t}{a^t} \cdot \frac{1}{c/a - 1} = O(c^t) = O(n^{\log_a c}),$$

если $a < c$. $\square$

§ 2. Сложность арифметических операций с целыми числами

2.1. Сложение и вычитание. Прежде всего заметим, что стандартные школьные алгоритмы сложения и вычитания чисел столбиком, очевидно, имеют оценку сложности $O(\log N)$, где N — большее из двух чисел. Это минимальная возможная оценка сложности (напомним, что мы рассматриваем оценки сложности с точностью до константного множителя), и поэтому нет смысла заниматься оптимизацией выполнения этих операций.

2.2. Умножение и деление. Для умножения и деления двух чисел N и M алгоритмы умножения столбиком и деления углом дают оценку $O(\log N \log M)$. Поэтому для функций оценки сложности умножения и деления двух n-разрядных чисел выполняется неравенство

$$n \leqslant f(n) \leqslant O(n^2).$$

Сначала докажем, что сложности операций умножения, деления с остатком, возведения в квадрат и нахождения обратного к данному числу совпадают с точностью до константного множителя. Под обратным к данному числу N, записанному в виде последовательности из n двоичных знаков, здесь понимается правильная дробь, имеющая n двоичных знаков после запятой и полученная из числа $1/N$ отбрасыванием более младших разрядов.

Пусть $M(n)$ — сложность операции умножения двух n-разрядных чисел, $D(n)$ — сложность операции деления с остатком $2n$-разрядного числа на n-разрядное число, $S(n)$ — сложность операции возведения в квадрат n-разрядного числа и $R(n)$ — сложность операции обращения n-разрядного числа.

Теорема. *В предположении, что функции $M(n)$, $D(n)$, $S(n)$ и $R(n)$ удовлетворяют условиям* (a) *и* (b), *справедливо утверждение*:

$$M(n) \approx D(n) \approx S(n) \approx R(n).$$

Доказательство. В самом деле, $M(n) \prec S(n)$, так как в силу тождества

$$AB = \frac{1}{2}((A+B)^2 - A^2 - B^2)$$

справедлива оценка $M(n) \leqslant 3S(n) + 4n$. Делению на 2 соответствует операция сдвига.

Аналогично, $S(n) \prec R(n)$, так как в силу равенства

$$N^2 = \frac{1}{\dfrac{1}{N} - \dfrac{1}{N+1}} - N$$

получаем оценку $S(n) \leqslant 3R(n) + 2n$. Здесь следует сделать следующее важное замечание. Данное равенство выполняется для точных значений действительных чисел, а у нас числа вида $1/N$ являются их приближенными значениями. Поэтому для исправления возможных ошибок в младших разрядах в алгоритме необходимо предусмотреть специальные процедуры их уточнения. Включение таких процедур в целом не влияет на оценку сложности (подробнее см. в [2]).

Для доказательства оценки $R(n) \prec M(n)$ воспользуемся итерационным методом Ньютона для вычисления значения обратного. Он заключается в вычислении последовательности итераций $x(0), x(1), \ldots \ldots, x(i), \ldots$ по формуле

$$x(i+1) = 2x(i) - Nx(i)^2.$$

Данная последовательность действительно сходится к $1/N$, так как если $x(i) = \frac{1}{N}(1-\delta)$, то

$$x(i+1) = 2x(i) - Nx(i)^2 = \frac{2}{N}(1-\delta) - \frac{1}{N}(1-\delta)^2 = \frac{1}{N}(1-\delta^2).$$

Поэтому при выборе начального значения $x(0)$ так, что $\delta < \frac{1}{2}$ (а это легко сделать по двум старшим значащим разрядам числа N), мы получаем последовательность, в которой каждый раз число правильно вычисленных знаков после запятой удваивается. Благодаря этому мы можем просто заменить нулями те знаки, которым нельзя доверять при данной итерации, и, постоянно удваивая число правильно подсчитанных знаков, через $\log n$ шагов получим нужное количество знаков.

Отсюда вытекает оценка

$$R(n) \leqslant R\left(\frac{n}{2}\right) + 2M(n) + 2n.$$

В силу леммы 1, получаем $R(n) \prec M(n)$.

Из цепочки неравенств $M(n) \prec S(n) \prec R(n) \prec M(n)$ теперь вытекает эквивалентность этих трех функций.

Докажем, что $R(n) \prec D(n)$. Из равенства $\frac{A}{B} = A\frac{1}{B}$ следует, что $D(n) \prec M(n) + R(n) \prec R(n)$. Наконец, в силу очевидной оценки $R(n) \prec D(n)$, получаем требуемую эквивалентность. $\square$

Рассмотрим теперь вопрос об оценке сложности операции умножения.

Оценку, меньшую $O(n^2)$, наиболее просто можно получить, применяя рекурсивный алгоритм, основанный на разбиении чисел на два слагаемых. Очень удобная версия этого алгоритма была предложена в работах Карацубы и Офмана (1962). Его суть заключается в рекурсивном повторении следующего шага. Пусть A и B — два n-разрядных числа. Разобьем их на два слагаемых (для простоты считаем, что $n = 2k$)

$$A = 2^k A_1 + A_0, \quad B = 2^k B_1 + B_0.$$

Тогда

$$\begin{aligned} AB &= (2^k A_1 + A_0)(2^k B_1 + B_0) = 2^{2k} A_1 B_1 + 2^k (A_0 B_1 + A_1 B_0) + A_0 B_0 = \\ &= (2^{2k} + 2^k) A_1 B_1 + 2^k (A_0 B_1 + A_1 B_0 - A_0 B_0 - A_1 B_1) + (2^k + 1) A_0 B_0 = \\ &= (2^{2k} + 2^k) A_1 B_1 + 2^k (A_0 - A_1)(B_1 - B_0) + (2^k + 1) A_0 B_0. \end{aligned}$$

Таким образом, для вычисления произведения двух n-разрядных чисел нужно выполнить три умножения $n/2$-разрядных чисел и некоторое количество сложений, вычитаний и переносов. Поэтому для сложности данного алгоритма умножения справедлива оценка

$$f(n) = 3f\left(\frac{n}{2}\right) + cn, \quad c > 0,$$

откуда, по лемме 2, получаем $M(n) \prec f(n) = O(n^{\log_2 3})$, где $\log_2 3 = 1{,}585\ldots$

Заметим, что этот алгоритм иначе можно интерпретировать как способ вычисления в точке $x = 2^k$ значения многочлена, равного произведению двух многочленов $U(x) = xA_1 + A_0$ и $V(x) = xB_1 + B_0$. В более общем случае, подобный рекурсивный алгоритм, основанный на разбиении чисел на r слагаемых и сведении задачи умножения чисел к задачам умножения и вычисления значений многочленов, дает оценку $M(n) = O(n^{1+\log_{r+1} 2})$. Заметим, что здесь $\log_{r+1} 2 \to 0$ при $r \to \infty$.

Наиболее эффективным в настоящее время является алгоритм Шенхаге—Штрассена (1970), дающий оценку $M(n) = O(n \log n \times \log\log n)$ (подробнее см. [2]). Помимо разбиения чисел на слагаемые он использует технику быстрого преобразования Фурье и модульную арифметику, которые будут рассмотрены ниже.

2.3. Возведение в степень. В заключение приведем оценку сложности операции возведения в степень. Для вычисления степени N^a натуральное число a представляется в виде

$$a = \sum_{i=0}^{k-1} a_i 2^i = (\ldots(a_{k-1}2 + a_{k-2})2 + \ldots + a_1)2 + a_0.$$

Теперь возведение в данную степень можно свести к последовательному выполнению операций двух типов: умножению на N и возведению в квадрат. Отсюда получается следующая оценка сложности

$$O(M(\log N)\log a) = O(M(n)k).$$

§ 3. Сложность алгоритма Евклида

3.1. Обычный алгоритм Евклида. Будем обозначать наибольший общий делитель чисел A и B через (A, B). Напомним, что алгоритм Евклида заключается в последовательном выполнении операции деления с остатком до получения нулевого остатка. Пусть $A>B>0$. Обозначим $A=r_{-1}$, $B=r_0$ и $r_{i-2}=d_i r_{i-1}+r_i$ при $i=1,\ldots,k$ и $r_{k-1}=d_{k+1}r_k$. Тогда $(A,B)=r_k$ и до получения остатка $r_{k+1}=0$ необходимо выполнить $k+1$ деление.

Оценим число делений, выполняемое в алгоритме Евклида. Для этого рассмотрим последовательность чисел Фибоначчи $f_0, f_1, \ldots$ $\ldots, f_k, \ldots$, где $f_0=0$, $f_1=1$, $f_k=f_{k-1}+f_{k-2}$, $k\geqslant 2$.

Лемма. *При $k>1$ справедливо неравенство $f_k\geqslant R^{k-2}$, где $R=\frac{1+\sqrt{5}}{2}$.*

Доказательство. Применим индукцию по n. При $k=2$ утверждение очевидно.

Далее, используя предположение индукции, имеем

$$f_{k+1}=f_k+f_{k-1}\geqslant R^{k-2}+R^{k-3}=R^{k-3}(R+1)=R^{k-3}R^2=R^{k-1},$$

так как R является положительным корнем уравнения $x^2=x+1$. □

Теорема (Ламе, 1844). *Для любого натурального числа $N>0$ число делений в алгоритме Евклида для нахождения наибольшего общего делителя чисел A и B, $0<B<A\leqslant N$ не превосходит $1+\lfloor\log_R N\rfloor$.*

Доказательство. Докажем, что $f_i\leqslant r_{k+1-i}$ при $i=1,\ldots,k+2$. При $i=1$ верно. Для $i+1$, в силу предположения индукции, имеем

$$r_{k-i}=d_{k-i+2}r_{k-i+1}+r_{k-i+2}\geqslant r_{k-i+1}+r_{k-i+2}\geqslant f_i+f_{i-1}=f_{i+1}.$$

Поэтому, $A\geqslant r_{-1}\geqslant f_{k+2}\geqslant R^k$, откуда получается искомая оценка числа делений $k+1\leqslant 1+\lfloor\log_R N\rfloor$. □

Непосредственно из этой теоремы можно получить оценку сложности алгоритма Евклида для нахождения наибольшего общего делителя двух n-разрядных чисел

$$O(M(n)(k+1)) = O(M(n)\log n)\leqslant O(n^2\log n).$$

Вместе с тем, эту оценку можно уточнить, если заметить, что сложность операции деления углом числа A на число B, $0<B<A$, на самом деле имеет вид $O(\log A(\log A-\log B+1))$. Отсюда, обозначая через n_i длину записи остатка r_i, $i=-1,0,1,\ldots,k$, получаем более точную оценку сложности алгоритма Евклида

$$\begin{aligned}\sum_{i=0}^{k} O(n_i(n_{i-1}-n_i+1)) \leqslant \sum_{i=0}^{k} O(n_0(n_{i-1}-n_i+1)) &=\\ =O\left(n_0\sum_{i=0}^{k}(n_{i-1}-n_i+1)\right) &=\\ =O(n_0(n_{-1}-n_k+k+1)) &=\\ =O(n_0 n_{-1}) = O(\log A\cdot\log B) &= O(n^2).\end{aligned}$$

3.2. Другие алгоритмы. Оценка сложности алгоритма Евклида не является оптимальной для задачи нахождения наибольшего общего делителя.

Имеется множество различных модификаций алгоритма Евклида. Можно, например, уменьшить число шагов алгоритма, модифицируя алгоритм с целью уменьшения абсолютных значений остатков. А именно, будем заменять остаток r_i на r_i-r_{i-1}, если $r_i\geqslant\frac{r_{i-1}}{2}$. При таком исправлении получается последовательность чисел, удовлетворяющая условию $|r_i|<\frac{r_{i-1}}{2}$. То, что при этом некоторые из чисел будут отрицательными, не влияет на вид общих делителей. В результате получается оценка числа делений $k+1\leqslant 1+\lfloor\log_2 N\rfloor$, причем $R=1{,}618\ldots<2$. Однако, данный подход не улучшает общую оценку сложности модифицированного алгоритма Евклида $O(M(\log N)\log N)=O(M(n)n)$, где $n=\log N$ — длина записи чисел A и B.

Существуют алгоритмы, в которых вообще не выполняется операция деления. Например, в алгоритме LSGCD (left shift greatest common divisor) операция деления с остатком заменена на левый сдвиг делителя с последующим вычитанием из делимого, такой, что полученная последовательность на каждом шаге удовлетворяет условию из предыдущего алгоритма. В данном алгоритме выполняется $O(n)$ шагов, каждый из которых имеет сложность $O(n)$, поэтому общая оценка сложности имеет вид $O(n^2)$.

Заметим, наконец, что имеется алгоритм нахождения наибольшего общего делителя с оценкой сложности $O(M(\log N)\log\log N)=$ $=O(M(n)\log n)$, однако, его описание достаточно сложно и здесь приведено не будет (см. [2]).

3.3. Расширенный алгоритм Евклида. Рассмотрим теперь расширенный алгоритм Евклида, позволяющий наряду с наибольшим общим делителем чисел A и B находить натуральные числа x и y, удовлетворяющие равенству $Ax + By = (A, B)$. От обычного алгоритма Евклида он отличается тем, что наряду с последовательностью остатков r_i вычисляются еще две вспомогательные последовательности x_i и y_i:

```
r_{-1} = A, r_0 = B;
x_{-1} = 1, y_{-1} = 0, x_0 = 0, y_0 = 1;
for i = 0 until r_i > 0 do
    begin
        d_i = ⌊r_{i-2}/r_{i-1}⌋;
        r_i = r_{i-2} - d_i r_{i-1};
        x_i = x_{i-2} - d_i x_{i-1};
        y_i = y_{i-2} - d_i y_{i-1};
        i = i + 1;
    end
```

Значения x_k и y_k, при которых $r_k = (A, B)$, и будут искомыми в силу следующего утверждения:

Лемма. *При всех i, $-1 < i \leqslant k$, выполняется равенство*

$$x_i A + y_i B = r_i.$$

Доказательство. Применим индукцию по i. При $i = -1, 0$ равенство очевидно. Если равенства доказаны для всех значений индексов меньших i, то для i, пользуясь индуктивным предположением, получаем

$$x_i A + y_i B = (x_{i-2} - d_i y_{i-1})A + (y_{i-2} - d_i y_{i-1})B = \\ = (x_{i-2}A + y_{i-2}B) - d_i(x_{i-1}A + y_{i-1}B) = r_i. \quad \square$$

Легко видеть, что сложность данного алгоритма отличается от сложности обычного алгоритма Евклида не более, чем на константный сомножитель, и составляет $O(M(\log N)\log N) = O(M(n)n)$, где $n = \log N$ — длина записи чисел A и B.

§ 4. Сложность операций в кольце вычетов

Будем отождествлять элементы кольца вычетов $\mathbb{Z}_N$ с числами в интервале $0 \leqslant A < N$. Пусть $n = \lceil \log_2 N \rceil$.

4.1. Сложение и вычитание. При сложении чисел в интервале $0 \leqslant A, B < N$ сумма $A + B$ может выйти за границы интервала, поэтому может понадобиться еще одно вычитание $A + B - N$. Аналогично, при вычитании может потребоваться еще одно сложение $A - B + N$. Поэтому, сложность этих операций равна $O(n)$.

4.2. Умножение. Для нахождения вычета, соответствующего произведению двух вычетов, надо выполнить одно умножение n-разрядных чисел и одно деление $2n$-разрядного числа на n-разрядное. Поэтому, сложность данной операции равна $O(M(n))$.

Во многих случаях, особенно при аппаратной реализации алгоритмов, удобно отказаться от операций умножения и деления и заменить их операциями сложения. Один из таких алгоритмов, предложенный П. Л. Монтгомери в 1985 г., состоит в следующем. Пусть N — нечетное число, требуется умножить вычеты $A = \sum\limits_{i=0}^{n-1} 2^i a_i$ и B. Рассмотрим алгоритм

```
R = 0;
for i = 0 until i < n do
    begin
        if a_i = 1 then R = R + B;
        if R — нечетно then R = R + N;
        R = R/2;
    end
if R ⩾ N then R = R - N.
```

Суть данного алгоритма в том, что в силу равенства

$$A = \sum_{i=0}^{n-1} 2^i a_i = (\ldots(2a_{n-1} + a_{n-2})2 + \ldots a_1)2 + a_0$$

умножение числа B на число A сводится к вычислению выражения

$$AB = a_0 B + 2(a_1 B + \ldots 2(a_{n-2}B + 2a_{n-1}B)\ldots).$$

Оно выполняется за n шагов, на каждом из которых осуществляется прибавление к текущему значению R значения $a_i B$, $i = 0, \ldots, n-1$, с последующим делением на 2. Благодаря этому делению полученные значения всегда находятся в интервале $0 < R < N$. В результате работы данного алгоритма получается число $2^{-n}AB \bmod N$. Теперь для получения числа $AB \bmod N$ необходимо применить еще один раз данный алгоритм к числам $2^{2n} \bmod N$ и $2^{-n}AB \bmod N$. Поскольку число $2^{2n} \bmod N$ вычисляется с помощью сдвигов и вычитаний со сложностью $O(n^2)$ двоичных операций (его можно вычислить заранее и хранить полученное значение), а алгоритм также выполняется за $O(n^2)$ операций, то общая трудоемкость вычисления произведения оценивается величиной $O(n^2)$ двоичных операций.

4.3. Обращение. Для заданного числа A, $0 \leqslant A < N$, находим с помощью расширенного алгоритма Евклида числа x и y, удовлетворяющие равенству $xA + yN = (A, N)$. Если $(A, N) > 1$, то A не имеет обратного. Если же $(A, N) = 1$, то в качестве обратного можно взять $x \bmod N$. Таким образом, обращение в кольце вычетов можно выполнить за $O(M(n)n)$ битовых операций.

4.4. Деление. Так как $A/B = A\frac{1}{B}$, то деление в кольце вычетов выполняется также со сложностью $O(M(n)n)$.

§ 5. Использование модульной арифметики

5.1. При вычислениях с целыми числами часто применяется следующий прием. Если известно, что исходные числа и результаты вычислений ограничены некоторым числом M (при этом допускаются неравенства двух видов $0 \leqslant N < M$ или $-M/2 < N < M/2$), то вычисления можно производить в кольце вычетов $\mathbb{Z}_M$, отождествляя числа из указанных интервалов и соответствующие вычеты. Само число M можно выбирать различными способами, причем его выбор во многом определяет сложность вычислений.

Наиболее эффективным такой переход является в случае, когда число M представимо в виде произведения небольших взаимно простых чисел $M = m_1 m_2 \ldots m_k$, поскольку в этом случае можно воспользоваться изоморфизмом колец

$$\mathbb{Z}_M \cong \mathbb{Z}_{m_1} + \mathbb{Z}_{m_2} + \ldots + \mathbb{Z}_{m_k}.$$

При этом соответствии каждому числу u из интервала $0 \leqslant u < M$ соответствует набор $(u_1, u_2, \ldots, u_k)$, где $u_i = u \bmod m_i$, $i = 1, \ldots, k$. (Здесь и далее запись $u \bmod m$, в отличие от записи правой части сравнения $u \pmod m$, обозначает наименьший неотрицательный остаток от деления числа u на число m.) В данном случае вместо вычислений с исходными числами можно сначала перейти к их остаткам и производить все вычисления в кольцах $\mathbb{Z}_{m_i}$, $i = 1, \ldots, k$, а затем, получив результат, выполнить обратный переход и восстановить по остаткам искомое число.

5.2. Для вычисления набора $(u_1, u_2, \ldots, u_k)$ по числу u нужно выполнить k делений числа u на числа m_i, $i = 1, \ldots, k$. Если все числа m_i записываются с помощью b двоичных знаков, то число u записывается с помощью kb двоичных знаков. Сложность деления kb-битового числа на b-битовое равна $O(kM(b))$, поэтому сложность перехода оценивается величиной $O(k^2 M(b))$.

Эту оценку можно улучшить, если применить технику «разделяй и властвуй» и разбить процесс вычислений на два этапа. Пусть для простоты выполняется равенство $k = 2^t$. На первом этапе последовательно находятся произведения чисел

$$\begin{gathered}
m_1 m_2, \quad m_3 m_4, \ldots, \quad m_{k-1} m_k, \\
m_1 m_2 m_3 m_4, \quad \ldots, \quad m_{k-3} m_{k-2} m_{k-1} m_k, \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
m_1 m_2 \ldots m_{k/4}, \quad \ldots, \quad m_{3k/4+1} \ldots m_k, \\
m_1 m_2 \ldots m_{k/2}, \quad m_{k/2+1} \ldots m_k.
\end{gathered}$$

На втором этапе выполняются деления

$$\begin{gathered}
u_{11} = u \bmod (m_1 m_2 \ldots m_{k/2}), \quad u_{12} = u \bmod (m_{k/2+1} \ldots m_{k/2}), \\
u_{21} = u_{11} \bmod (m_1 m_2 \ldots m_{k/4}), \ldots, \quad u_{24} = u_{12} \bmod (m_{3k/4+1} \ldots m_{k/2}), \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
u_{t-1,1} = u_{t-2,1} \bmod (m_1 m_2), \quad u_{t-1,2} = u_{t-2,1} \bmod (m_3 m_4), \quad \ldots \\
\ldots, \quad u_{t-1,k/2} = u_{t-2,k/4} \bmod (m_{k-1} m_k), \\
u_1 = u_{t-1,1} \bmod m_1, \quad u_2 = u_{t-1,1} \bmod m_2, \quad \ldots, \quad u_k = u_{t-1,k/2} \bmod m_k.
\end{gathered}$$

Легко видеть, что для выполнения первого этапа требуется

$$2^{t-1} M(b) + 2^{t-2} M(2b) + \ldots + 2M(2^{t-1} b) = O(tM(kb))$$

двоичных операций. На втором этапе требуется выполнить

$$2M(2^{t-1} b) + 2^2 M(2^{t-2} b) + \ldots + 2^{t-1} M(b) = O(tM(kb))$$

двоичных операций.

Таким образом, общая оценка сложности имеет вид $O(M(kb) \log k)$ двоичных операций.

Для выполнения обратного перехода от набора чисел $(u_1, u_2, \ldots, u_k)$ к числу u применяется китайская теорема об остатках.

Теорема. *Пусть $M = m_1 m_2 \ldots m_k$, где числа m_i попарно взаимно просты, и*

$$c_i = m_1 \ldots m_{i-1} m_{i+1} \ldots m_k = M/m_i,$$
$$d_i = c_i^{-1} \bmod m_i,$$

$i = 1, \ldots, k$. Тогда решение системы сравнений

$$u \equiv u_i \pmod{m_i}, \quad i = 1, \ldots, k,$$

существует, однозначно по модулю M и находится по формуле

$$u = \sum_{i=1}^{k} c_i d_i u_i \bmod M.$$

Доказательство легко вытекает из следующих свойств выбранных чисел:

$$c_i d_i \equiv 0 \pmod{m_j} \quad \text{при } j \neq i,$$

и

$$c_i d_i \equiv 1 \pmod{m_i}, \quad i, j = 1, \ldots, k. \quad \square$$

Для вычисления значения u по данной формуле требуется

$$\sum_{i=1}^{k} (O((k-1)M(b)) + T_{XEA}(b)) + M(kb) =$$
$$= O((k^2 M(b) + kT_{XEA}(b) + M(kb))) =$$
$$= O(k^2 M(b) + kT_{XEA}(b))$$

двоичных операций, где через $T_{XEA}(b)$ обозначена сложность нахождения обратного элемента в кольце $\mathbb{Z}_{m_i}$, $\log m_i = b$, с помощью расширенного алгоритма Евклида. Трудоемкость можно уменьшить, если аналогично предыдущему случаю воспользоваться техникой «разделяй и властвуй», применяя каждый раз аналог данной формулы при $k = 2$. В этом случае получается оценка $O(M(kb)\log k + kT_{XEA}(b))$.

Существует и другая формула для вычисления числа u,

$$u = q_1 + q_2 m_1 + q_3 m_1 m_2 + \ldots + q_k m_1 \ldots m_{k-1},$$

где числа $q_1, q_2, \ldots, q_k$ вычисляются в процессе выполнения следующего алгоритма:

```
c = 1, u = u_1 mod m_1;
for i = 1 to k − 1 do
    begin
        c = c · m_i;
        d = c^{-1} mod m_{i+1};
        q = d(u_{i+1} − u) mod m_{i+1};
        u = u + qc;
    end
return(u).
```

Доказательство корректности данного алгоритма нахождения числа u проводится с помощью индукции и предлагается выполнить в качестве упражнения.

Хотя трудоемкость вычисления по этой формуле такая же, как и для исходной формулы, приведенной в теореме, и составляет $O(k^2 M(b) + kT_{XEA}(b))$ двоичных операций, данная формула во многих случаях оказывается более удобной. Это вызвано тем, что процесс восстановления решения системы

$$u \equiv u_i \pmod{m_i}, \quad i = 1, \ldots, k,$$

осуществляется в ней последовательно: сначала для первых двух сравнений, затем для первых трех, и так далее до получения общего решения. Если к системе добавить еще одно сравнение, то ход вычислений не изменится, и надо будет выполнить всего один дополнительный шаг. В то же время в случае исходной формулы добавление еще одного сравнения полностью меняет схему вычислений.

§ 6. Вычисления с многочленами

Между вычислениями в кольце многочленов над произвольным кольцом R и в кольце целых чисел, записанных в какой-либо системе счисления, много общего. Они выполняются по похожим формулам, а отличие заключается лишь в том, что для чисел необходимо учитывать знаки переноса от младших разрядов к старшим, в то время как в случае многочленов никаких переносов при операциях с коэффициентами многочленов не возникает — и исходные величины и значения лежат в кольце R. Поэтому вычисления с многочленами в чем-то даже проще, чем вычисления с целыми числами.

Сложность операций с многочленами обычно оценивают количеством арифметических операций кольца R, выполняемых над его коэффициентами. Если известна битовая сложность операций в поле, то можно также оценить результирующую битовую сложность операций с многочленами. Чтобы отличать арифметическую сложность от битовой в оценках мы будем использовать символы $O_A(\ )$ и $O_B(\ )$.

Вычисление значений многочленов. Пусть R — произвольное кольцо. Рассмотрим хорошо известный алгоритм Руффини—Горнера для вычисления значения многочлена

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$$

над кольцом R в точке $x = b$. Он основан на следующем представлении многочлена

$$f(x) = ((a_n x + a_{n-1})x + \ldots + a_1)x + a_0$$

и заключается в последовательном вычислении значений $p_0, p_1, \ldots, p_n$ по формулам

$$p_0 = a_n,$$
$$p_i = p_{i-1}b + a_{n-i},$$

$i = 1, \ldots, n$. Последнее число p_n и будет искомым значением многочлена. Арифметическая сложность алгоритма, очевидно, равна $O_A(n)$. Битовую сложность в случае, когда в качестве кольца R рассматривается кольцо целых чисел, можно оценить выражением $O_B(nml)$, где через m обозначен максимум из двух чисел: числа двоичных знаков в записи наибольшего коэффициента и числа b, а число $l = (n+2)m$ обозначает число двоичных знаков в записи наибольшего из чисел p_i, $i = 1, \ldots, k$. Таким образом, получается оценка $O_B(n^2m^2)$.

Алгоритм Руффини—Горнера позволяет получить не только значение $f(b) = p_n$. Как показывает следующая теорема, величины $p_0, p_1, \ldots, p_{n-1}$ являются в точности коэффициентами многочлена, являющегося остатком от деления многочлена $f(x)$ на $(x - b)$.

Теорема. *Справедливо равенство*

$$f(x) = (x-b)\left(\sum_{i=0}^{n-1} p_{n-i-1}x^i\right) + f(b).$$

Доказательство. Имеем

$$\begin{aligned}
(x-b)\left(\sum_{i=0}^{n-1} p_{n-i-1}x^i\right) + f(b) &= \\
&= \left(\sum_{i=0}^{n-1} p_{n-i-1}x^{i+1}\right) - \left(\sum_{i=0}^{n-1} p_{n-i-1}bx^i\right) + f(b) = \\
&= \left(\sum_{i=1}^{n} p_{n-i}x^i\right) - \left(\sum_{i=0}^{n-1} p_{n-i-1}bx^i\right) + f(b) = \\
&= \sum_{i=1}^{n} (p_{n-i} - p_{n-i-1}b)x^i - p_{n-1}b + f(b) = \\
&= \sum_{i=1}^{n} a_i x^i - p_{n-1}b + (p_{n-1}b + a_0) = \\
&= \sum_{i=0}^{n} a_i x^i = f(x).
\end{aligned}$$

Теорема доказана. □

§ 7. Дискретное преобразование Фурье

7.1. Пусть R — коммутативное кольцо с единицей. Тогда элемент ω кольца R называется *примитивным корнем степени n из единицы*, если выполняются свойства:

1. $\omega \neq 1$;
2. $\omega^n = 1$;
3. $\sum\limits_{j=0}^{n-1} \omega^{ij} = 0$, $1 \leqslant i < n$.

Если, кроме того, элемент n является обратимым элементом кольца R, то можно определить *дискретное преобразование Фурье* как отображение, которое каждому вектору $a = (a_0, a_1, \ldots, a_{n-1})$, $a_i \in R$, $0 \leqslant i \leqslant n-1$ ставит в соответствие вектор $F(a) = b = (b_0, b_1, \ldots, b_{n-1})$, где

$$b_i = \sum_{j=0}^{n-1} a_i \omega^{ij}, \quad 0 \leqslant i \leqslant n-1.$$

Обратное дискретное преобразование Фурье определяется как $F^{-1}(b)=c$, где координаты вектора c равны

$$c_i=\frac{1}{n}\sum_{j=0}^{n-1} b_i\omega^{-ij}, \quad 0\leqslant i\leqslant n-1.$$

То, что прямое и обратное дискретные преобразования Фурье взаимно обратны (т. е. $F^{-1}(F(a))=a$, $F(F^{-1}(b))=b$), легко следует из определения примитивного корня.

Заметим, что если вектору a поставить в соответствие многочлен $f(x)=\sum\limits_{i=0}^{n-1} a_i x^i$, то прямое дискретное преобразование Фурье соответствует вычислению значений многочлена в точках ω^i, $0\leqslant i\leqslant n-1$, а обратное — интерполяции многочлена по его значениям в этих точках. То, что данные точки образуют циклическую мультипликативную подгруппу в R, позволяет построить быстрый алгоритм вычисления значения как прямого, так и обратного дискретного преобразования Фурье.

7.2. Приведем алгоритм вычисления преобразования $F(a)$ (алгоритм для вычисления $F^{-1}(b)$ строится аналогично), называемый *алгоритмом быстрого преобразования Фурье.* Для простоты полагаем, что $n=2^k$. Напомним, что значение многочлена $f(x)$ в точке $x=\omega^i$ совпадает с остатком от деления многочлена $f(x)$ на многочлен $x-\omega^i$, $0\leqslant i\leqslant n-1$. При этом многочлены $x-\omega^i$, $0\leqslant i\leqslant n-1$ попарно взаимно просты и их произведение равно x^n-1. Поэтому можно применить подход, который рассматривался в п. 5.2 при нахождении значений остатков от деления элемента на взаимно простые модули.

Заметим, что $\omega^{n/2}=-1$, поэтому

$$x^n-1=\left(x^{n/2}-1\right)\left(x^{n/2}-\omega^{n/2}\right).$$

Далее

$$x^{n/2}-1=\left(x^{n/4}-1\right)\left(x^{n/4}-\omega^{n/2}\right),$$
$$x^{n/2}-\omega^{n/2}=\left(x^{n/4}-\omega^{n/4}\right)\left(x^{n/4}-\omega^{3n/4}\right).$$

Продолжаем этот процесс до получения линейных сомножителей $x-\omega^i$. Таким образом, нахождение остатков от деления многочлена $f(x)$ на $x-\omega^i$, $0\leqslant i\leqslant n-1$, можно провести с использованием техники «разделяй и властвуй» путем деления $f(x)$ сначала на многочлены $x^{n/2}-1$ и $x^{n/2}-\omega^{n/2}$, затем каждый из двух полученных остатков два раза делим на многочлены вида $x^{n/4}-\omega^{jn/4}$, и так далее.

Лемма. *Пусть*

$$f(x)=\sum_{i=0}^{n-1} a_i x^i$$

и $c\in R$. *Тогда остаток от деления* $f(x)$ *на* $(x^{n/2}-c)$ *равен*

$$r(x)=\sum_{i=0}^{\frac{n}{2}-1}(a_i+ca_{i+\frac{n}{2}})x^i.$$

Доказательство вытекает из равенства

$$f(x)=(x^{n/2}-c)\sum_{i=0}^{\frac{n}{2}-1} a_{i+\frac{n}{2}}x^i+r(x). \quad \square$$

Из этой леммы вытекает, что для вычисления вектора коэффициентов остатка надо разделить вектор коэффициентов многочлена $f(x)$ пополам, затем умножить на c коэффициенты второй половины и сложить их с коэффициентами первой половины вектора. Это требует выполнения не более n арифметических операций кольца R.

В результате получаем, что для выполнения всего алгоритма надо выполнить не более

$$n+2\,\frac{n}{2}+4\,\frac{n}{4}+\ldots+2^k\,\frac{n}{2^k}=kn$$

арифметических операций кольца R, т. е. сложность алгоритма быстрого преобразования Фурье равна $O(n\log n)$.

7.3. Благодаря алгоритму быстрого преобразования Фурье дискретное преобразование Фурье является очень удобным инструментом при проведении вычислений с многочленами. Так, например, с его помощью можно вычислять произведение многочленов со сложностью $O(n\log n)$ (выполнив сначала преобразование Фурье и получив значения многочленов в точках, затем перемножив полученные значения и, наконец, вернуться к коэффициентам многочлена с помощью обратного преобразования Фурье). Однако, оно обладает тем недостатком, что для существования преобразования Фурье над кольцом R требуется выполнение двух условий: существования примитивного корня степени n и обратимости числа n в кольце R, а эти условия выполняются далеко не всегда. Кроме того, как следует из алгоритма быстрого преобразования Фурье, желательно, чтобы число n было некоторой степенью числа 2.

Приведем один подход, позволяющий эффективно применять быстрое преобразование Фурье в кольце целых чисел. Если известно, что коэффициенты многочленов ограничены некоторым числом M, то вычисления можно производить в кольце вычетов $\mathbb{Z}_M$, отождествляя числа из указанных интервалов и соответствующие вычеты. Поскольку при этом число M можно выбирать различными способами, то мы будем искать такие числа M, для которых число $n=2^k$ и, кроме того, в качестве примитивного элемента ω можно выбрать также некоторую степень числа 2. Это позволяет очень эффективно реализовать быстрое преобразование Фурье на ЭВМ, где используется двоичное представление чисел.

Теорема. *Если $n=2^k$ и $\omega=2^q\neq 1$, то при $M=\omega^{n/2}+1$ элемент n является обратимым элементом кольца $\mathbb{Z}_M$, а элемент ω — примитивным корнем из единицы степени n.*

Нам потребуются две леммы.

Лемма 1. *В коммутативном кольце с единицей для любого элемента a и $n=2^k$ выполняется равенство*

$$\sum_{i=0}^{n-1} a^i = \prod_{j=0}^{k-1} (1+a^{2^j}).$$

Доказательство. Применим индукцию по k. При $k=1$ равенство очевидно. Если оно верно при $k-1$, то при k получаем

$$\sum_{i=0}^{n-1} a^i = (1+a) \sum_{i=0}^{\frac{n}{2}-1} a^{2i}.$$

По предположению индукции получаем

$$\sum_{i=0}^{\frac{n}{2}-1} a^{2i} = \prod_{j=0}^{k-2} (1+(a^2)^{2^j}) = \prod_{j=1}^{k-1} (1+a^{2^j}),$$

что и требовалось доказать.

Лемма 2. *При $M=\omega^{n/2}+1$, $0\neq\omega\in R$, для всех $1\leqslant i<n$ имеем*

$$\sum_{j=0}^{n-1} \omega^{ij} \equiv 0 \pmod{M}.$$

Доказательство. Согласно лемме 1 достаточно показать, что при всех $1\leqslant i<n$ найдется j такое, что

$$1+\omega^{i2^j} \equiv 0 \pmod{M}.$$

Если $i=2^s t$, где t нечетно, то полагаем $j=k-1-s$. Тогда

$$1+\omega^{i2^j} = 1+\omega^{2^{k-1}t} = 1+(\omega^{n/2})^t \equiv 1+(-1)^t \equiv 0 \pmod{M},$$

так как t нечетно.

Вернемся к доказательству теоремы. Элемент n обратим, так как числа $n=2^k$ и $M=\omega^{n/2}+1=2^{qn/2}+1$ взаимно просты. Элемент $\omega=2^q\neq 1$, причем $\omega^{n/2}=-1+M\equiv -1 \pmod{M}$. Отсюда $\omega^n\equiv 1 \pmod{M}$. Наконец, лемма 2 гарантирует выполнение третьего условия, необходимого для примитивности корня из единицы. Теорема доказана. □

II. Элементы теории чисел

§ 8. Непрерывные дроби и их свойства

8.1. Напомним, что алгоритм Евклида для нахождения наибольшего общего делителя чисел A и B, $A > B$, заключается в последовательном выполнении операции деления с остатком применительно к последовательности чисел $A = r_{-1}, B = r_0, r_1, r_2, \ldots, r_k$ до получения нулевого остатка $r_{k+1} = 0$, где $r_{i-2} = d_i r_{i-1} + r_i$ при $i = 1, \ldots, k$ и $r_{k-1} = d_{k+1} r_k$. Тогда $(A, B) = r_k$. Рассмотрим теперь получающуюся в данном алгоритме последовательность $d_1, d_2, \ldots, d_{k+1}$. Нетрудно видеть, что эти числа удовлетворяют равенству

$$\frac{A}{B} = d_1 + \cfrac{1}{d_2 + \cfrac{1}{d_3 + \cfrac{1}{\ldots + \cfrac{1}{d_{k+1}}}}}$$

Для краткости будем обозначать правую часть этого выражения через $[d_1, d_2, \ldots, d_{k+1}]$ и называть непрерывной (или цепной) дробью.

Непрерывные дроби появились в математике еще в XVI в. Широкое распространение они получили после работ Х. Гюйгенса (XVII в.), который применял их для подбора зубчатых колес, с заданным передаточным отношением. Теория непрерывных дробей была систематически разработана Л. Эйлером, а затем Ж. Лагранжем.

Отметим простейшие свойства таких дробей.

Для любого n и любых натуральных $d_1, d_2, \ldots, d_n$ справедливы равенства

$$[d_1, d_2, \ldots, d_n] = d_1 + \frac{1}{[d_2, \ldots, d_n]};$$

$$[d_1, d_2, \ldots, d_n] = \left[d_1, d_2, \ldots, d_{n-2}, d_{n-1} + \frac{1}{d_n}\right];$$

при $k = 1, \ldots, n-1$

$$[d_1, d_2, \ldots, d_n] = \left[d_1, \ldots, d_k + \frac{1}{[d_{k+1}, \ldots, d_n]}\right].$$

Теорема (о фундаментальном соответствии). *Для любой последовательности натуральных чисел $a_1, a_2, \ldots, a_n, \ldots$ равенства*

$$[a_1, a_2, \ldots, a_n] = \frac{P_n}{Q_n}, \quad n = 1, 2, \ldots$$

выполняются в том и только в том случае, когда выполняются матричные равенства

$$\begin{pmatrix} a_1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_2 & 1 \\ 1 & 0 \end{pmatrix} \ldots \begin{pmatrix} a_n & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} P_n & P_{n-1} \\ Q_n & Q_{n-1} \end{pmatrix}, \quad n = 2, 3, \ldots$$

Доказательство. Для удобства полагаем $P_0 = 1$, $Q_0 = 0$. Доказательство проводим индукцией по n. При $n = 1$ утверждение очевидно. Если для $n-1$ оно выполнено, то по предположению индукции для непрерывной дроби $[a_2, \ldots, a_n] = \frac{X_{n-1}}{Y_{n-1}}$ должно выполняться равенство

$$\begin{pmatrix} a_2 & 1 \\ 1 & 0 \end{pmatrix} \ldots \begin{pmatrix} a_n & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} X_{n-1} & X_{n-2} \\ Y_{n-1} & Y_{n-2} \end{pmatrix}.$$

С другой стороны, эти дроби связаны соотношением

$$\frac{P_n}{Q_n} = [a_1, a_2, \ldots, a_n] = a_1 + \frac{1}{[a_2, \ldots, a_n]} = a_1 + \frac{Y_{n-1}}{X_{n-1}} = \frac{a_1 X_{n-1} + Y_{n-1}}{X_{n-1}},$$

что полностью соответствует матричному равенству

$$\begin{pmatrix} P_n & P_{n-1} \\ Q_n & Q_{n-1} \end{pmatrix} = \begin{pmatrix} a_1 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} X_{n-1} & X_{n-2} \\ Y_{n-1} & Y_{n-2} \end{pmatrix}.$$

Теорема доказана. □

Дроби $[a_1, a_2, \ldots, a_n] = \frac{P_n}{Q_n}$, $n = 1, 2, \ldots$ называются *подходящими дробями.* В качестве следствия из этой теоремы мы получаем практически все основные свойства подходящих дробей.

1. $P_n Q_{n-1} - P_{n-1} Q_n = (-1)^n$, $n = 1, 2, \ldots$
2. *Числа P_n и Q_n взаимно просты при всех $n = 1, 2, \ldots$*
3. $\frac{P_n}{Q_n} - \frac{P_{n-1}}{Q_{n-1}} = \frac{(-1)^n}{Q_n Q_{n-1}}$, $n = 1, 2, \ldots$
4. $\frac{P_n}{Q_n} = a_1 + \sum\limits_{k=1}^{n} \frac{(-1)^k}{Q_k Q_{k-1}}$, $n = 1, 2, \ldots$
5. *Последовательности P_n и Q_n удовлетворяют рекуррентным соотношениям $P_n = a_n P_{n-1} + P_{n-2}$, $Q_n = a_n Q_{n-1} + Q_{n-2}$ при $n = 2, 3, \ldots$*
6. $Q_n = a_n Q_{n-1} + Q_{n-2} \geqslant Q_{n-1} + Q_{n-2} \geqslant 2Q_{n-2} \geqslant 2^{\frac{n-2}{2}}$, $n = 2, 3, \ldots$

8.2. Рассмотрим теперь вопрос о представлении действительных чисел непрерывными дробями. Пусть α — некоторое положительное действительное число. Выделим в нем целую и дробную части: $\alpha = \lfloor\alpha\rfloor + \{\alpha\}$. Полагаем $a_1 = \lfloor\alpha\rfloor$, и если $\{\alpha\} = 0$, то заканчиваем, а если $\{\alpha\} \neq 0$, то получаем равенство $\alpha = a_1 + \frac{1}{\alpha_1}$. Повторяя этот процесс для числа α_1, и т. д. получим в результате последовательность $a_1, a_2, \ldots, a_n, \ldots$, для которой при каждом n выполняется равенство $\alpha = [a_1, a_2, \ldots, a_n, \alpha_n]$, $n = 1, 2, \ldots$ Числа $a_n(\alpha_n)$ называются *неполными (полными) частными.*

Возможны два случая: либо процесс закончится на некотором шаге n и выполнено равенство $\alpha = [a_1, a_2, \ldots, a_n]$, либо при всех n выполняются неравенства $\alpha \neq [a_1, a_2, \ldots, a_n]$. В случае бесконечной последовательности $a_1, a_2, \ldots, a_n, \ldots$ определим значение бесконечной непрерывной дроби $[a_1, a_2, \ldots, a_n, \ldots]$ как предел

$$\lim_{n\to\infty} \frac{P_n}{Q_n} = \lim_{n\to\infty} \left(a_1 + \sum_{k=1}^{n} \frac{(-1)^k}{Q_k Q_{k-1}} \right),$$

который всегда существует в силу сходимости знакочередующегося ряда с бесконечно убывающими членами.

Рассмотрим как связаны между собой число α и подходящие дроби $[a_1, a_2, \ldots, a_n] = \frac{P_n}{Q_n}$, $n = 1, 2, \ldots$ для получившейся в результате применения данной процедуры (конечной, или бесконечной) последовательности $a_1, a_2, \ldots, a_n, \ldots$

Лемма. *Если на шаге $n+1$ число α не совпадает со значением подходящей дроби, то выполняется одно из двух неравенств:*

$$\frac{P_n}{Q_n} < \alpha < \frac{P_{n+1}}{Q_{n+1}} \quad \text{или} \quad \frac{P_{n+1}}{Q_{n+1}} < \alpha < \frac{P_n}{Q_n}.$$

Д о к а з а т е л ь с т в о. Рассмотрим действительную функцию $f(x) = [a_1, a_2, \ldots, a_{n-1}, x]$. Очевидно, что

$$f(a_n) = \frac{P_n}{Q_n},$$
$$f\left(a_n + \frac{1}{\alpha_n}\right) = \alpha,$$
$$f\left(a_n + \frac{1}{a_{n+1}}\right) = \frac{P_{n+1}}{Q_{n+1}}.$$

В силу приведенных выше свойств подходящих дробей для функции $f(x)$ должно выполняться равенство

$$f(x) = \frac{xP_{n-1} + P_{n-2}}{xQ_{n-1} + Q_{n-2}}.$$

Следовательно, $f(x)$ — гипербола, и поэтому является строго монотонной функцией. Учитывая неравенства $a_n < a_n + \frac{1}{\alpha_n} < a_n + \frac{1}{a_{n+1}}$, получаем, что в зависимости от того, монотонно возрастает или убывает функция $f(x)$, выполняется одно из требуемых неравенств. Лемма доказана. □

На самом деле нетрудно видеть, что должны выполняться неравенства

$$\frac{P_1}{Q_1} < \frac{P_3}{Q_3} < \ldots \leqslant \alpha \leqslant \ldots < \frac{P_4}{Q_4} < \frac{P_2}{Q_2}.$$

Значение конечной непрерывной дробей с целыми неполными частными, очевидно, является рациональным числом. Наоборот, справедлива

Теорема 1. *Рациональные числа однозначно представляются в виде конечных непрерывных дробей с целыми неполными частными.*

Д о к а з а т е л ь с т в о. Если α рациональное число, то согласно алгоритму Евклида указанный выше процесс построения непрерывной дроби закончится при нахождении наибольшего общего делителя. При этом значение непрерывной дроби будет совпадать с данным числом. Наоборот, любая конечная непрерывная дробь с целыми неполными частными является рациональным числом. Поэтому необходимо доказать только однозначность такого представления.

Предположим, что имеется два различных представления рационального числа $\alpha = [a_1, a_2, \ldots, a_n] = [b_1, b_2, \ldots, b_m]$. Пусть $a_k \neq b_k$ — первое отличие в данных последовательностях. Тогда при некоторых $0 \leqslant \varepsilon < 1$ и $0 \leqslant \delta < 1$ должно выполняться равенство

$$\alpha = \frac{(a_k + \varepsilon)P_{k-1} + P_{k-2}}{(a_k + \varepsilon)Q_{k-1} + Q_{k-2}} = \frac{(b_k + \delta)P_{k-1} + P_{k-2}}{(b_k + \delta)Q_{k-1} + Q_{k-2}}.$$

Отсюда, рассуждая аналогично лемме, в силу монотонности гиперболы получаем, что $a_k + \varepsilon = b_k + \delta$, противоречие. Теорема доказана. □

Теорема 2. *Иррациональные действительные числа однозначно представляются в виде бесконечных непрерывных дробей с целыми неполными частными. Наоборот, значением всякой бесконечной* *непрерывной дроби с целыми неполными частными является иррациональное число.*

Доказательство. Если α иррациональное число, то описанный выше процесс построения непрерывной дроби никогда не закончится, так как значение конечной непрерывной дроби является рациональным числом. При этом значение непрерывной дроби в силу леммы будет определяться пределом

$$\lim_{n\to\infty}\frac{P_n}{Q_n}=\lim_{n\to\infty}\left(a_1+\sum_{k=1}^{n}\frac{(-1)^k}{Q_kQ_{k-1}}\right),$$

который по лемме должен совпадать с числом α.

Однозначность представления произвольного действительного числа в виде непрерывной дроби доказывается аналогично предыдущей теореме.

Остается заметить, что значение любой бесконечной непрерывной дроби с целыми неполными частными является действительным числом и, в силу однозначности представления числа в виде непрерывной дроби, не может рациональным числом. Теорема доказана. □

8.3. В заключение отметим без доказательства еще два замечательных свойства непрерывных дробей. Во-первых, последовательности неполных частных, получаемые при разложении действительных чисел, могут рассматриваться как рациональные приближения этих чисел. При этом оказывается, что непрерывные дроби дают в определенном смысле наилучшие рациональные приближения. Во-вторых, бесконечные периодические непрерывные дроби и только они представляют множество чисел, являющихся квадратичными иррациональностями, т. е. действительными решениями квадратных уравнений с целыми коэффициентами (Лагранж).

Отметим интересную связь непрерывных дробей с календарными стилями. Как известно, продолжительность года составляет 365,24220… суток. Этому числу соответствует периодическая дробь $[365,4,7,1,3,\ldots]$, первые подходящие дроби которой равны соответственно

$$365,\quad 365\,\frac{1}{4},\quad 365\,\frac{7}{29},\quad 365\,\frac{8}{33}.$$

Приближение $365\frac{1}{4}$ — это так называемый Юлианский календарь, введенный в 47 г. до н. э. Юлием Цезарем, где каждый четвертый год — високосный. Новый Григорианский календарь дает приближение $365\frac{97}{400}$, что немного больше, чем $365\frac{8}{33}$. Этот стиль отличается тем, что каждый сотый год — не високосный, кроме тех, число сотен которых делится на 4 (т. е. 400 лет имеют 97, а не 100 лишних суток). Хотя отставание Юлианского календаря было замечено еще в XV в., реформа была проведена только в конце XIX в. Наиболее же точный календарь ввел в Персии в 1079 г. персидский астроном и математик Омар Альхайями. Он ввел цикл из 33 лет, в котором семь раз високосным считался четвертый, а восьмой раз — пятый. Это как раз приближение $365\frac{8}{33}$, так как имеется 8 лишних суток в 33 года.

§ 9. Квадратичные вычеты

9.1. Пусть p — нечетное простое число. Целое число a называется *квадратичным вычетом по модулю* p, если сравнение $x^2\equiv a \pmod p$ имеет решение.

Из определения следует, что свойство целого числа a быть или не быть квадратичным вычетом по модулю p определяется только свойствами остатка $a \bmod p$ от деления числа a на число p, и поэтому вместо чисел a можно рассматривать элементы кольца вычетов $\mathbb{Z}_p$.

Отметим простейшие свойства квадратичных вычетов.

1. *Среди чисел* $\{1,2,\ldots,p-1\}$ *ровно половина являются квадратичными вычетами, а оставшиеся числа — нет.*

Для доказательства рассмотрим отображение $\varphi\colon x\mapsto x^2$ мультипликативной группы кольца $\mathbb{Z}_p$ в себя. Очевидно, оно является гомоморфизмом с ядром $\operatorname{Ker}\varphi=\{-1,1\}$, причем образом $\varphi(\mathbb{Z}_p)$ будет множество ненулевых квадратичных вычетов.

2. *Если* $\mathbb{Z}_p^*=\{1,\theta,\theta^2,\ldots,\theta^{p-2}\}$, *где* θ — *примитивный элемент поля* $\mathbb{Z}_p$, *то элемент* $a=\theta^j$ *будет квадратичным вычетом в том и только в том случае, когда* j *четно.*

Доказательство очевидно, так как в сравнении $2y\equiv j \pmod{p-1}$ модуль $p-1$ является четным числом.

9.2. Символ Лежандра. При изучении свойств квадратичных вычетов обычно выделяют две проблемы. Во-первых, как можно узнать является ли данное число квадратичным вычетом (задача распознавания), и, во-вторых, как найти решение данного сравнения (задача поиска решений). Мы покажем, что ответ на первый вопрос дать несложно. Для этого надо просто вычислить соответствующий символ Лежандра, что является алгоритмически несложной задачей. В то же время, задача поиска решений является алгоритмически сложной задачей и для нахождения решений требуется построение специальных алгоритмов.

Символ Лежандра, введенный им в 1798 г., определяется следующим образом:

$$\left(\frac{a}{p}\right)=\begin{cases}0, & a\equiv 0 \pmod p,\\ 1, & \exists\, x,\ x^2\equiv a \pmod p,\ a \bmod p\neq 0,\\ -1, & \nexists\, x,\ x^2\equiv a \pmod p,\ a \bmod p\neq 0.\end{cases}$$

Приведем основные свойства символа Лежандра.

1. *Если* $a_1\equiv a \pmod p$, *то* $(\frac{a_1}{p})\equiv(\frac{a}{p})$.

2 (критерий Эйлера). $(\frac{a}{p})\equiv a^{\frac{p-1}{2}} \pmod p$.

Доказательство. При $a\equiv 0 \pmod p$ равенство очевидно. Если a имеет ненулевой вычет, то согласно малой теореме Ферма $a^{p-1}\equiv 1 \pmod p$, откуда $a^{\frac{p-1}{2}}\equiv\pm 1 \pmod p$. При этом, если $a=\theta^j$, где θ — примитивный элемент кольца $\mathbb{Z}_p$, то

$$a^{\frac{p-1}{2}}\equiv 1 \pmod p \Longleftrightarrow j\,\frac{p-1}{2}\equiv 0 \pmod{p-1}.$$

Последнее равносильно тому, что j четно и a является квадратичным вычетом, т.е. $(\frac{a}{p})=1$. Аналогично рассматривается случай $(\frac{a}{p})=-1$. □

3. $(\frac{ab}{p})=(\frac{a}{p})(\frac{b}{p})$.

Это и следующие два свойства являются очевидными следствиями свойства 2.

4. *Если* $(a,p)=1$, *то* $(\frac{a^2b}{p})=(\frac{b}{p})$.

5. $(\frac{1}{p})=1$, $(\frac{-1}{p})=(-1)^{\frac{p-1}{2}}$.

Для дальнейшего нам потребуется следующая лемма.

Лемма. *Для любых нечетных чисел s и t выполняются сравнения*:

(а) $\frac{s-1}{2}+\frac{t-1}{2}\equiv\frac{st-1}{2} \pmod 2$,

(б) $s^2\equiv 1 \pmod 8$,

(в) $\frac{s^2-1}{8}+\frac{t^2-1}{8}\equiv\frac{(st)^2-1}{8} \pmod 2$.

Доказательство первого сравнения вытекает из равенства

$$\frac{1}{2}\,(st-s-t+1)=\frac{1}{2}\,(s-1)(t-1),$$

где в правой части стоит четное число. Для доказательства второго сравнения надо заметить, что число $(s-1)(s+1)$ всегда делится на 8.

Наконец, третье сравнение вытекает из равенства

$$\frac{1}{8}(s^2t^2-s^2-t^2+1)=\frac{1}{8}(s^2-1)(t^2-1),$$

в правой части которого также стоит четное число. □

6. $(\frac{2}{p})=(-1)^{\frac{p^2-1}{8}}$.

Доказательство. Так как элементы $\{1,-1\}$ лежат в любом расширении поля $\mathbb{Z}_p$, то можно рассматривать данное равенство в поле $GF(p^2)$. В силу пункта (б) леммы справедливо сравнение $p^2\equiv 1 \pmod 8$, и поэтому в данном поле всегда содержится элемент ω порядка 8 (например, надо взять примитивный элемент, имеющий порядок (p^2-1), и возвести его в степень $\frac{(p^2-1)}{8}$). Рассмотрим функцию

$$f(x)=\begin{cases}(-1)^{\frac{x^2-1}{8}}, & x\equiv 1 \pmod 2,\\ 0, & x\equiv 0 \pmod 2.\end{cases}$$

Так как число 8 является периодом этой функции, то можно рассматривать ее как функцию, заданную на элементах кольца вычетов $\mathbb{Z}_8$. При этом в силу пункта (в) леммы для нечетных s и t выполняется равенство $f(st)=f(s)f(t)$.

Определим величину $G\in GF(p^2)$ равенством

$$G=\sum_{j=0}^{7} f(j)\omega^j.$$

Покажем, что $G\neq 0$. Имеем

$$G=\omega-\omega^3-\omega^5+\omega^7=2(\omega-\omega^3),$$

так как $\omega^4=-1$. Отсюда $G^2=4(\omega^2-2\omega^4+\omega^6)=8\neq 0$, а, следовательно, и $G\neq 0$ в поле $GF(p^2)$.

Подсчитаем теперь двумя способами величину G^p. С одной стороны

$$G^p=(G^{\frac{p-1}{2}})^2G=8^{\frac{p-1}{2}}G=\left(\frac{8}{p}\right)G=\left(\frac{2}{p}\right)G.$$

С другой стороны,

$$G^p=\left(\sum_{j=0}^{7} f(j)\omega^j\right)^p=\sum_{j=0}^{7} f(j)\omega^{pj},$$

так как p нечетно. Используя свойства функции $f(x)$ получаем

$$G^p = \sum_{j=0}^{7} f(p)f(pj)\omega^{pj} = f(p)\sum_{j=0}^{7} f(pj)\omega^{pj} = f(p)G.$$

Теперь, приравнивая оба выражения и сокращая на G, получаем требуемое равенство. Что и требовалось доказать. □

9.3. Квадратичный закон взаимности Гаусса. Докажем еще одно замечательное свойство символа Лежандра, называемое квадратичным законом взаимности. Эмпирически он был открыт в 1783 г. Л. Эйлером. Первое полное доказательство было дано в 1796 г. 19-летним К. Гауссом. Всего он предложил семь различных доказательств квадратичного закона взаимности.

Теорема. *Для любых простых нечетных чисел p и q справедливо равенство*

$$\left(\frac{p}{q}\right) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}\left(\frac{q}{p}\right).$$

Доказательство. Поступаем полностью аналогично доказательству последнего свойства. Перейдем к такому расширению $GF(p^m)$ поля $\mathbb{Z}_p$, в котором содержится элемент ω порядка q (например, можно положить $m = q-1$, так как выполняется сравнение $p^{q-1} \equiv 1 \pmod q$). Рассмотрим величину

$$G = \sum_{j=0}^{q-1}\left(\frac{j}{q}\right)\omega^j.$$

Сначала докажем, что $G \neq 0$. Имеем

$$G = \left(\sum_{j=0}^{q-1}\left(\frac{j}{q}\right)\omega^j\right)\left(\sum_{k=0}^{q-1}\left(\frac{-k}{q}\right)\omega^{-k}\right) = \left(\frac{-1}{q}\right)\sum_{j=0}^{q-1}\sum_{k=0}^{q-1}\left(\frac{j}{q}\right)\left(\frac{k}{q}\right)\omega^{j-k}.$$

Так как $\left(\frac{0}{q}\right) = 0$, то можно изменить нижний предел у обеих сумм. Одновременно заменяя во второй сумме индекс k на jk, получаем

$$G = \left(\frac{-1}{q}\right)\sum_{k=1}^{q-1}\sum_{j=1}^{q-1}\left(\frac{j}{q}\right)\left(\frac{jk}{q}\right)\omega^{j(1-k)} = \left(\frac{-1}{q}\right)\sum_{k=1}^{q-1}\left(\frac{k}{q}\right)\sum_{j=1}^{q-1}\omega^{j(1-k)}.$$

Теперь вернем второй сумме индекс $j=0$, добавив к первой сумме выражение

$$\sum_{k=1}^{q-1}\left(\frac{k}{q}\right) = (+1)\,\frac{q-1}{2} + (-1)\,\frac{q-1}{2} = 0.$$

Наконец, учитывая, что

$$\sum_{j=0}^{q-1}\omega^{ja} = \begin{cases} 0, & a \neq 0, \\ q, & a = 0, \end{cases}$$

получаем

$$G^2 = \left(\frac{-1}{q}\right)\sum_{k=1}^{q-1}\left(\frac{k}{q}\right)\sum_{j=0}^{q-1}\omega^{j(1-k)} = \left(\frac{-1}{q}\right)\left(\frac{1}{q}\right)q = (-1)^{\frac{q-1}{2}}q.$$

Отсюда $G^2 \neq 0$, а, следовательно, и $G \neq 0$ в поле $GF(p^m)$.

Подсчитаем теперь двумя способами величину G^p. С одной стороны

$$G^p = \left((-1)^{\frac{q-1}{2}}q\right)^{\frac{p-1}{2}}G = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}q^{\frac{p-1}{2}}G \equiv (-1)^{\frac{p-1}{2}\frac{q-1}{2}}\left(\frac{q}{p}\right)G \pmod p.$$

С другой стороны,

$$G^p = \left(\sum_{j=0}^{q-1}\left(\frac{j}{q}\right)\omega^j\right)^p = \sum_{j=0}^{q-1}\left(\frac{j}{q}\right)\omega^{pj},$$

так как p нечетно. Используя свойства символа Лежандра получаем

$$G^p = \sum_{j=0}^{q-1}\left(\frac{p}{q}\right)\left(\frac{pj}{q}\right)\omega^{pj} = \left(\frac{p}{q}\right)\sum_{j=0}^{q-1}\left(\frac{pj}{q}\right)\omega^{pj} = \left(\frac{p}{q}\right)G.$$

Теперь, приравнивая оба выражения и сокращая на G, получаем требуемое равенство. Теорема доказана.

9.4. Доказанные свойства символа Лежандра позволяют вычислять для любого простого нечетного p и любого целого a значение $\left(\frac{a}{p}\right)$ с помощью следующего алгоритма:

1) если число a отрицательно, то выделяем сомножитель $\left(\frac{-1}{p}\right)$;

2) заменяем число a на остаток $a \bmod p$ от деления числа a на p;

3) раскладываем число a в произведение простых сомножителей $a = p_1^{a_1}p_2^{a_2}\ldots p_k^{a_k}$ и переходим к разложению в произведение

4) $$\left(\frac{a}{p}\right) = \left(\frac{p_1}{p}\right)^{a_1}\left(\frac{p_2}{p}\right)^{a_2}\ldots\left(\frac{p_k}{p}\right)^{a_k};$$

5) если $p_1=2$ и a_1 нечетно, вычисляем $\left(\frac{2}{p}\right)$;

6) для каждого нечетного сомножителя p_i с нечетным значением степени a_i применяем квадратичный закон взаимности;

7) если необходимо, возвращаемся к п. 1.

Например,

$$\left(\frac{126}{53}\right)=\left(\frac{20}{53}\right)=\left(\frac{2}{53}\right)^2\left(\frac{5}{53}\right)=(-1)^{26\cdot 2}\left(\frac{53}{5}\right)=\left(\frac{-2}{5}\right)= \\ =(-1)^2(-1)^3=-1.$$

9.5. Символ Якоби. Изложенный выше метод вычисления символа Лежандра является неудобным в связи с тем, что при его выполнении приходится прибегать к сложной операции факторизации натуральных чисел. Чтобы избавиться от необходимости факторизовать числа рассмотрим обобщение символа Лежандра, называемое символом Якоби.

Пусть n нечетно и имеет следующее разложение на простые сомножители $n=p_1^{a_1}p_2^{a_2}\dots p_k^{a_k}$. Тогда для любого целого числа a *символ Якоби* определяется равенством

$$\left(\frac{a}{n}\right)=\left(\frac{a}{p_1}\right)^{a_1}\left(\frac{a}{p_3}\right)^{a_2}\dots\left(\frac{a}{p_k}\right)^{a_k}.$$

В данном случае равенство $\left(\frac{a}{n}\right)=1$ вовсе не обязательно означает, что число a является квадратичным вычетом по модулю n. Например,

$$\left(\frac{2}{15}\right)=\left(\frac{2}{3}\right)\left(\frac{2}{5}\right)=(-1)(-1)=1,$$

однако число 2, очевидно, не является квадратичным вычетом по модулю 15. Поэтому к символу Якоби надо относиться как к формальной функции от двух аргументов. Замечательное свойство этого символа заключается в том, что он удовлетворяет практически всем тем же свойствам, что и символ Лежандра.

1. *Если* $a_1\equiv a \pmod n$, *то* $\left(\frac{a_1}{n}\right)\equiv\left(\frac{a}{n}\right)$.

2. $\left(\frac{ab}{n}\right)=\left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$.

3. *Если* $(a,n)=1$, *то* $\left(\frac{a^2b}{n}\right)=\left(\frac{b}{n}\right)$.

4. $\left(\frac{1}{n}\right)=1,\quad \left(\frac{-1}{n}\right)=(-1)^{\frac{n-1}{2}}$.

5. $\left(\frac{2}{n}\right)=(-1)^{\frac{n^2-1}{8}}$.

Доказательство этих свойств проводится с помощью п. (а) и (в) леммы из п. 9.2. Их предлагается выполнить самостоятельно в качестве упражнения.

Докажем лишь обобщение квадратичного закона взаимности.

Теорема. *Для любых нечетных чисел* m *и* n *справедливо равенство*

$$\left(\frac{m}{n}\right)=(-1)^{\frac{m-1}{2}\frac{q-1}{2}}\left(\frac{n}{m}\right).$$

Доказательство. Достаточно считать, что $(m,n)=1$, в противном случае обе части равенства равны нулю. Предположим, что числа m и n имеют следующие разложения на простые сомножители

$$m=q_1^{b_1}q_2^{b_2}\dots q_s^{b_s},\quad n=p_1^{a_1}p_2^{a_2}\dots p_k^{a_k}.$$

Тогда по определению и свойствам символа Якоби получаем

$$\left(\frac{m}{n}\right)=\prod_{j=1}^{k}\prod_{i=1}^{s}\left(\frac{q_i}{p_j}\right)^{a_jb_i},\quad \left(\frac{n}{m}\right)=\prod_{i=1}^{s}\prod_{j=1}^{k}\left(\frac{p_j}{q_i}\right)^{a_jb_i}.$$

Согласно квадратичному закону взаимности при всех i и j имеем

$$\left(\frac{q_i}{p_j}\right)=(-1)^{\frac{p_j-1}{2}\frac{q_i-1}{2}}\left(\frac{p_j}{q_i}\right).$$

Остается заметить, что в силу п. (а) леммы справедливо равенство

$$\prod_{i=1}^{s}\prod_{j=1}^{k}(-1)^{a_j\frac{p_j-1}{2}b_i\frac{q_i-1}{2}}=(-1)^{\frac{n-1}{2}\frac{m-1}{2}}.$$

Теорема доказана. □

9.6. Доказанные свойства символа Якоби позволяют для любого простого нечетного p и любого целого a, не прибегая к факторизации числа, вычислять значение символа Лежандра $\left(\frac{a}{p}\right)$ с помощью следующего алгоритма.

Полагаем $n=p$.

1) если число a отрицательно, то выделяем сомножитель $\left(\frac{-1}{n}\right)$;

2) заменяем число a на $a \bmod n$ — остаток от деления числа a на n;

3) если a четно, представляем его в виде произведения $a=2^ta_1$, где $(a_1,2)=1$, и, если t нечетно, то вычисляем $\left(\frac{2}{n}\right)$;

4) применяем к $\left(\frac{a_1}{n}\right)$ квадратичный закон взаимности

$$\left(\frac{a_1}{n}\right) = (-1)^{\frac{n-1}{2}\frac{a_1-1}{2}}\left(\frac{n}{a_1}\right);$$

5) если необходимо, возвращаемся к п. 1.

Например,

$$\left(\frac{136}{53}\right) = \left(\frac{30}{53}\right) = \left(\frac{2}{53}\right)\left(\frac{15}{53}\right) = (-1)^{\frac{53^2-1}{8}}(-1)^{26\cdot 7}\left(\frac{53}{15}\right) =$$
$$= -\left(\frac{8}{15}\right) = -\left(\frac{2}{15}\right) = -(-1)^{28} = -1.$$

§ 10. Теорема Чебышева о распределении простых чисел

10.1. Вопрос о распределении простых чисел всегда вызывал большой интерес. В разной постановке он исследовался многими математиками.

Рассмотрим вопрос о поведении функции $\pi(x)$, равной числу простых чисел p в интервале $1 < p \leqslant x$. Еще четырнадцатилетний Гаусс в 1791 г., анализируя эту функцию при малых x, заметил, что лучше всего ее приближает функция $\frac{x}{\ln x}$. В 1808 г. Лежандр предложил формулу, которая с достаточно большой точностью дает число простых чисел — $\frac{x}{\ln x - 1{,}08366}$. Он проверил формулу для $10^4 < x < 10^6$. В 1850 г. П. Л. Чебышев показал, что формула Лежандра неверна, и с помощью достаточно сложной техники получил следующие оценки

$$0{,}921\,\frac{x}{\ln x} < \pi(x) < 1{,}106\,\frac{x}{\ln x}.$$

В дальнейшем им были получены значения констант, более близкие к 1. В 1896 г. Адамар и Валле-Пуссен независимо доказали асимптотический закон $\pi(x) \sim \frac{x}{\ln x}$ с помощью теории функций комплексного переменного. Точнее они показали, что функция

$$\int\limits_2^x \frac{dt}{\ln t} \sim \frac{x}{\ln x}$$

дает более точное приближение для $\pi(x)$, чем $\frac{x}{\ln x}$. Доказательство асимптотического закона элементарными методами было получено только в 1949 г. Селбергом.

10.2. Докажем следующую теорему, являющуюся упрощенным вариантом результата П. Л. Чебышева.

Теорема. *Существуют две постоянные c_1 и c_2, такие что $0 < c_1 < 1$, $c_2 > 1$ и для всех $x \geqslant 2$ выполняются неравенства*

$$c_1\,\frac{x}{\ln x} < \pi(x) < c_2\,\frac{x}{\ln x}.$$

Для доказательства данных неравенств удобнее оценивать функцию

$$\theta(x) = \sum_{p \leqslant x} \ln p,$$

называемую функцией Чебышева.

Лемма 1. *При всех $x \geqslant 2$ выполняется неравенство*

$$\theta(x) < (4\ln 2)x.$$

Доказательство. В силу очевидного неравенства

$$2^{2n} > \binom{2n}{n} > \prod_{n<p}^{p<2n} p$$

получаем

$$2n\ln 2 > \theta(2n) - \theta(n).$$

Отсюда

$$\theta(2^m) \leqslant 2\ln 2(1 + 2 + \ldots + 2^{m-1}) < (2\ln 2)2^m,$$

и для $x = 2^m$ лемма верна. При $2^{m-1} < x < 2^m$ имеем

$$\theta(x) \leqslant \theta(2^m) < (2\ln 2)2^m = (4\ln 2)2^{m-1} < (4\ln 2)x. \ \square$$

Лемма 2. *Существует значение $c > 0$ такое, что для всех $x \geqslant 2$ выполняется неравенство $\theta(x) > cx$.*

Доказательство. Введем обозначения

$$t_p(n) = \max\{k \geqslant 0\colon p^k \leqslant n\},$$
$$o_p(n) = \max\{k \geqslant 0\colon p^k \mid n\}.$$

Сначала заметим, что $t_p(n) = \lfloor \log_p n \rfloor$. Вторую величину подсчитаем по формуле

$$o_p(n!) = \sum_{j=1}^{t_p(n)} \left\lfloor \frac{n}{p^j} \right\rfloor.$$

Поэтому при всех $p<2n$ имеем

$$o_p\left(\binom{2n}{n}\right)=o_p\left(\frac{(2n)!}{(n!)^2}\right)=\sum_{j=1}^{t_p(2n)}\left\lfloor\frac{2n}{p^j}\right\rfloor-2\sum_{j=1}^{t_p(n)}\left\lfloor\frac{n}{p^j}\right\rfloor=$$
$$=\sum_{j=1}^{t_p(2n)}\left(\left\lfloor\frac{2n}{p^j}\right\rfloor-2\left\lfloor\frac{n}{p^j}\right\rfloor\right)\leqslant t_p(2n).$$

В последнем неравенстве использована очевидная оценка $0\leqslant\lfloor 2x\rfloor-2\lfloor x\rfloor\leqslant 1$. Отсюда

$$2^n<\frac{n+1}{1}\frac{n+2}{2}\ldots\frac{2n}{n}=\binom{2n}{n}=\prod_{p<2n}p^{o_p\left(\binom{2n}{n}\right)}\leqslant\prod_{p<2n}p^{t_p(2n)}.$$

Следовательно,

$$n\ln 2<\sum_{p<2n}t_p(2n)\ln p=\sum_{p<2n}\lfloor\log_p 2n\rfloor\ln p=\Sigma_1+\Sigma_2,$$
$$\Sigma_1\leqslant\sum_{p\leqslant\sqrt{2n}}\frac{\ln 2n}{\ln p}\ln p\leqslant\sqrt{2n}\ln 2n,$$
$$\Sigma_2=\sum_{p>\sqrt{2n}}^{p<2n}\left\lfloor\frac{\ln 2n}{\ln p}\right\rfloor\ln p\leqslant\sum_{p>\sqrt{2n}}^{p<2n}1\ln p\leqslant\theta(2n).$$

Отсюда вытекает неравенство

$$\theta(2n)>n\ln 2-\sqrt{2n}\ln 2n>cn,$$

где $c>0$ — некоторая константа. Наконец, при $2n<x\leqslant 2n+1$ получаем

$$\theta(x)\geqslant\theta(2n)>cn\geqslant c\,\frac{x-1}{2}>c_1x,\quad c_1>0.\quad\square$$

Доказательство теоремы. Верхняя оценка. Имеем

$$\theta(x)\geqslant\sum_{p\geqslant\sqrt{x}}^{p<x}\ln p>\ln\sqrt{x}(\pi(x)-\pi(\sqrt{x}))\geqslant\ln\sqrt{x}(\pi(x)-\sqrt{x}).$$

Следовательно, с учетом леммы 1 получаем

$$\pi(x)\leqslant\frac{2\theta(x)}{\ln x}+\sqrt{x}<8\ln 2\,\frac{x}{\ln x}+\sqrt{x}<c_2\,\frac{x}{\ln x}$$

при некоторой константе $c_2>1$.

Нижняя оценка. Из неравенства

$$\theta(x)=\sum_{p\leqslant x}\ln p\leqslant\pi(x)\ln x$$

с помощью леммы 2 получаем

$$\pi(x)\geqslant\frac{\theta(x)}{\ln x}>c\,\frac{x}{\ln x}.$$

Теорема доказана. $\square$

10.3. В качестве следствия из доказанной выше теоремы получаются оценки величины n-го простого числа.

Следствие 1. *Пусть p_n — n-е простое число. Тогда найдутся такие константы $0<c_3<c_4$, что при всех достаточно больших n выполняются неравенства*

$$c_3n\ln n<p_n<c_4n\ln n.$$

Доказательство. При $x=p_n$ формулировка теоремы приобретает вид

$$c_1\,\frac{p_n}{\ln p_n}<n<c_2\,\frac{p_n}{\ln p_n}.$$

Верхняя оценка. Логарифмируя левое неравенство получаем

$$\ln p_n+\ln c_1-\ln\ln p_n<\ln n,$$

и так как при достаточно больших n справедлива оценка

$$\frac{1}{2}\ln p_n<\ln p_n+\ln c_1-\ln\ln p_n,$$

то

$$p_n<\frac{1}{c_1}\,n\ln p_n<\frac{2}{c_1}\,n\ln n=c_4n\ln n.$$

Нижняя оценка. Так как $p_n>n$, имеем

$$p_n>\frac{1}{c_2}\,n\ln p_n>\frac{1}{c_2}\,n\ln n=c_3n\ln n.\quad\square$$

Следствие 2. *Пусть p_n — n-е простое число. Тогда при некоторых константах $0<c_5<c_6$ справедливы неравенства*

$$c_5\ln n<p_{n+1}-p_n<c_6\ln n.\quad\square$$

III. Арифметические алгоритмы

§ 11. Проверка простоты

11.1. Решето Эратосфена

Под этим названием понимают следующий метод построения всех простых чисел, не превосходящих некоторого заданного числа N. Берем число 2 и выбрасываем все числа кратные 2. Из оставшихся чисел оставляем наименьшее (в данном случае это число 3) и выбрасываем все числа кратные 3, и так далее. Если первое оставшееся число превышает $\lfloor\sqrt{N}\rfloor$, то работу прекращаем, поскольку все отобранные и оставшиеся числа являются простыми. При этом ни одно простое число в заданном интервале не будет упущено.

Данный метод позволяет строить множество простых чисел, но он неудобен для проверки простоты заданного числа. Тем не менее, идея решета и ее обобщения в настоящее время часто используются для «просеивания» множеств чисел, обладающих тем или иным условием. Более того, разрабатываются специальные микропроцессоры, на которых операции «просеивания» выполняются очень эффективно.

11.2. Критерий Вильсона

В 1770 г. Э. Варинг опубликовал следующую теорему, приписываемую Д. Вильсону.

Теорема. *Для любого n следующие условия эквивалентны*:

(а) n — *простое*;

(б) $(n-1)! \equiv -1 \pmod{n}$.

Доказательство. В случае $n=2$ утверждение очевидно. Если $n=p>2$ — простое, то каждый элемент a поля, отличный от 1 и -1 имеет обратный a^{-1}, причем $a \neq a^{-1}$. Поэтому

$$(n-1)! \equiv (-1) \prod_{a\neq 1,-1} aa^{-1} \equiv -1 \pmod{n}.$$

Если $n = ab$ — составное, $1 < a < n$, то $a \mid (n-1)!$ и, следовательно, $(n-1)!$ является необратимым элементом кольца $\mathbb{Z}_n$. Поэтому $(n-1)! \not\equiv -1 \pmod{n}$. Теорема доказана. □

Данный критерий иногда бывает удобен в доказательствах, но применять его для проверки простоты невозможно ввиду большой трудоемкости.

11.3. Тест на основе малой теоремы Ферма

Малая теорема Ферма утверждает, что если n простое, то выполняется условие:

при всех $a \in \{2, 3, \ldots, n-1\}$ имеет место сравнение

$$a^{n-1} \equiv 1 \pmod{n}. \tag{1}$$

Обратное утверждение неверно.

Из этой теоремы следует, что если сравнение (1) не выполнено хотя бы для одного числа a в интервале $\{1, 2, \ldots, n-1\}$, то n — составное. Поэтому можно предложить следующий *вероятностный тест простоты*:

1) выбираем случайное число из интервала $\{1, 2, \ldots, n-1\}$ и проверяем с помощью алгоритма Евклида условие $(a, n) = 1$;

2) если оно не выполняется, то ответ «n — составное»;

3) проверяем выполнимость сравнения (1);

4) если сравнение не выполнено, то ответ «n — составное»;

5) если сравнение выполнено, то ответ неизвестен, но можно повторить тест еще раз.

Если выполняется сравнение (1), то говорят, что число n является *псевдопростым по основанию a*. Заметим, что существует бесконечно много пар чисел (a, n), где n — составное и псевдопростое по основанию a. Например, при $(a, n) = (2, 341)$ получаем $2^{340} = (2^{10})^{34} \equiv 1 \pmod{n}$, хотя $431 = 11 \cdot 31$. Приведем наименьшие примеры минимальных псевдопростых чисел для оснований $a = 2, 3, 5, 7$.

a	n
2	$341 = 11 \cdot 31$
3	$91 = 7 \cdot 13$
5	$217 = 7 \cdot 31$
7	$25 = 5 \cdot 5$

Для любого $a > 1$ имеется бесконечно много псевдопростых чисел по основанию a. Например, *если пара $(2, n)$ удовлетворяет сравнению, то и пара $(2, 2^n-1)$ также ему удовлетворяет* (Р. Стейервальд, 1947).

Действительно, $2^n - 2 = 2(2^{n-1} - 1) = 2tn$ при некотором t. Поэтому

$$2^{2^n-2} - 1 = 2^{2tn} - 1 = (2^n - 1)(2^{(2t-1)n} + \ldots + 1) \equiv 0 \pmod{n}.$$

Но если число n составное, то и $2^n - 1$ также составное. Таким образом, имеется бесконечно много псевдопростых чисел по основанию 2. При $a > 2$ можно воспользоваться следующим утверждением, которое предлагается доказать в качестве упражнения, *для любого нечетного простого числа p и любого a такого, что $(a^2 - 1, p) = 1$, число $\frac{a^{2p}-1}{a^2-1}$ будет псевдопростым по основанию a.*

В то же время псевдопростых чисел относительно мало. Например, известно, что существует всего 21853 псевдопростых чисел по основанию 2 среди 1 091 987 405 простых чисел меньших, чем 25 000 000 000.

Приведем свойства псевдопростых чисел.

Утверждение 1. *Пусть n нечетное составное. Тогда*

(а) *n псевдопростое по основанию a в том и только в том случае, когда $(a, n) = 1$ и порядок элемента a в $\mathbb{Z}_n$ делит число $n - 1$;*

(б) *если n псевдопростое по основаниям a и $b \in \mathbb{Z}_n^*$, то n псевдопростое по основаниям ab и ab^{-1};*

(в) *множество $F_n = \{a \in \mathbb{Z}_n : a^{n-1} \equiv 1 \pmod{n}\}$ образует подгруппу мультипликативной группы $\mathbb{Z}_n^*$;*

(г) *если n не является псевдопростым по основанию a хотя бы одного числа a, то $|F_n| \leqslant \frac{1}{2}|\mathbb{Z}_n^*|$.*

Доказательство очевидно. □

Из пункта (г), в частности, следует, что если для числа n имеется хотя бы одно основание a, по которому оно не является псевдопростым, то имеется по крайней мере $(n-1)/2$ чисел, по которым данное число также не является псевдопростым.

Особый случай составляют составные числа, для которых условие (1) выполняется при всех основаниях. Они называются *псевдопростыми* числами, или *числами Кармайкла.*

Таким образом, при применении описанного выше теста может возникнуть три ситуации:

— число n простое и тест всегда говорит «не известно»;

— число n составное и не является числом Кармайкла; тогда с вероятностью успеха не меньше $1/2$ тест дает ответ «n — составное»;

— число n составное и является числом Кармайкла, тогда тест всегда дает ответ «не известно».

Наличие третьей ситуации является очень неудобным свойством данного теста. Для практики нужны такие тесты, в которых третья ситуация не возникает. Для построения таких тестов изучим сначала свойства чисел Кармайкла.

11.4. Свойства чисел Кармайкла

Нам потребуется следующая

Лемма (Гаусс, 1801). *Для любого нечетного простого p и любого $m>1$ мультипликативная группа кольца $\mathbb{Z}_{p^m}$ является циклической.*

Доказательство. Имеем

$$|\mathbb{Z}_{p^m}^*| = \varphi(\mathbb{Z}_{p^m}) = p^{m-1}(p-1).$$

Для любого элемента a определим его порядок по модулю p^k, как

$$\mathrm{ord}_k(a) = \min\{t \geqslant 1 : a^t \equiv 1 \pmod{p^k}\}.$$

Заметим, что функция $\mathrm{ord}_k(\ )$ имеет обычные свойства порядка. В частности, если порядки двух элементов взаимно просты, то порядок произведения равен произведению их порядков. Кроме того, при $1 \leqslant k < m$ выполняется равенство $\mathrm{ord}_k(a) \mid \mathrm{ord}_{k+1}(a)$, так как отображение $a \mapsto a \bmod p^k$ является гомоморфизмом кольца $\mathbb{Z}_{p^m}$ в кольцо $\mathbb{Z}_{p^k}$, а порядок образа делит порядок прообраза.

При $k = 1$ кольцо $\mathbb{Z}_p$ является полем, поэтому в $\mathbb{Z}_{p^m}$ найдется элемент g_0, имеющий по модулю p порядок $p - 1$.

Покажем, что один из элементов g_0 или $g = (p+1)g_0$ имеет порядок $p(p-1)$ по модулю p^2. Если элемент g_0 не такой, то он имеет порядок $p - 1$ по модулю p^2. Поскольку элемент g представим в виде произведения элемента $p + 1$, имеющего порядок p

$$(p+1)^p = \left(1 + pp + \binom{p}{2}p^2 + \ldots\right) \equiv 1 \pmod{p^2},$$

и элемента g_0 порядка $p - 1$ по модулю p^2, то его порядок должен равняться произведению $p(p-1)$.

Теперь индукцией по m докажем, что если у элемента g порядок $\mathrm{ord}_2(g) = p(p-1)$, то $\mathrm{ord}_m(g) = p^{m-1}(p-1)$ при любом $m \geqslant 2$. При $m = 2$ это верно. Предположим, что для $m - 1$ это верно, т. е. $\mathrm{ord}_{m-1}(g) = p^{m-2}(p-1)$. Тогда

$$g^{(p-1)p^{m-2}} \equiv 1 \pmod{p^{m-1}} \quad \text{и} \quad g^{(p-1)p^{m-3}} \not\equiv 1 \pmod{p^{m-1}}.$$

Для простоты записи положим $h=g^{(p-1)p^{m-3}}$. Тогда данные сравнения примут вид

$$h^p \equiv 1 \pmod{p^{m-1}} \quad \text{и} \quad h \not\equiv 1 \pmod{p^{m-1}}.$$

Следовательно,

$$h \equiv (1+kp^{m-2}) \pmod{p^{m-1}}, \quad (k,p)=1,$$

откуда

$$h^p \equiv (1+kp^{m-2})^p \equiv \left(1+pkp^{m-2}+\binom{p}{2}k^2p^{2(m-2)}+\ldots\right) \equiv$$
$$\equiv 1+kp^{m-1} \not\equiv 1 \pmod{p^m}.$$

С другой стороны, $h^{p^2}=(1+kp^{m-1})^p \equiv 1 \pmod{p^m}$. Таким образом,

$$p^{m-2}(p-1)=\mathrm{ord}_{m-1}(g) \neq \mathrm{ord}_m(g),$$

но $\mathrm{ord}_m(g) \mid p^{m-1}(p-1)$. Отсюда следует, что $\mathrm{ord}_m(g)=p^{m-1}(p-1)$. Лемма доказана. □

Докажите в качестве упражнения следующую теорему о существовании примитивного элемента в кольце вычетов.

Теорема (Гаусс, 1801). *Мультипликативная группа кольца $\mathbb{Z}_n$ является циклической в том и только в том случае, когда n — одно из чисел 2, 4, p^m, или $2p^m$, где $m \geqslant 1$, p — нечетное простое.*

Теорема (Кармайкл, 1912). *Пусть n нечетное составное. Тогда*

(а) *если $p^2 \mid n$, $p>1$, то n не является числом Кармайкла*;

(б) *если $n=p_1p_2\ldots p_k$, $p_i \neq p_j$, то n — число Кармайкла в том и только в том случае, когда при всех i выполнено условие $(p_i-1)\mid(n-1)$*;

(в) *если $n=p_1p_2\ldots p_k$, $p_i \neq p_j$ — число Кармайкла, то $k \geqslant 3$.*

Доказательство. (а) Пусть $n=p^tm$, $t\geqslant 2$, $(m,2)=1$. Пусть для элемента a выполнено условие $\mathrm{ord}_2(a)=p(p-1)$. Тогда число n не является псевдопростым по основанию a, так как если $a^{n-1}\equiv 1 \pmod{n}$, то и $a^{n-1}\equiv 1 \pmod{p^2}$. Отсюда должно быть $p(p-1)\mid(n-1)$, что противоречит условию $p\mid n$.

(б) Достаточность. Пусть $n=p_1p_2\ldots p_k$, $p_i\neq p_j$, и при всех i выполнено условие $(p_i-1)\mid(n-1)$, или $n-1=(p_i-1)m_i$. Тогда для любого основания a получаем, что при всех i выполнено сравнение $a^{n-1}\equiv (a^{p_i-1})^{m_i}\equiv 1 \pmod{p_i}$, и по китайской теореме об остатках $a^{n-1}\equiv 1 \pmod{n}$.

Необходимость. При каждом $i=1,\ldots,k$ найдется элемент a_i такой, что $\mathrm{ord}_1(a_i)=p_i-1$. Тогда из условия $a_i^{n-1}\equiv 1 \pmod{n}$ следует, что $(p_i-1)\mid(n-1)$.

(в) Если $k=2$, и $n=pq$, $p<q$, то $n-1=p(q-1+1)-1\equiv p-1 \pmod{q-1}$. Это противоречит условию $(q-1)\mid(n-1)$, так как $0<p-1<q-1$. □

Числа Кармайкла являются достаточно редкими. Так, имеется всего 2163 чисел Кармайкла не превосходящих 25 000 000 000. До 100 000 числами Кармайкла являются только следующие 16 чисел 561, 1105, 1729, 2465, 2821, 6601, 8911, 10585, 15841, 29341, 41041, 46657, 52633, 62745, 63973 и 75361. Проверка того, является ли заданное число числом Кармайкла, согласно доказанной теореме требует нахождения разложения числа на простые сомножители, т. е. факторизации числа. Поскольку задача факторизации чисел является более сложной, чем задача проверки простоты, то предварительная отбраковка чисел Кармайкла не представляется возможной. Поэтому в приведенном выше тесте простые числа и числа Кармайкла полностью неразличимы.

11.5. Тест Соловея—Штрассена

Теорема. *Для любого нечетного n следующие условия эквивалентны*:

(а) *n — простое*;

(б) *для любого $a \in \mathbb{Z}_n^*$ выполняется сравнение*

$$a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \pmod{n}. \tag{2}$$

Доказательство. Если n простое, то данное сравнение очевидно выполнено в силу свойств символа Лежандра. Пусть теперь выполнено условие (б), но n не простое. Тогда

$$a^{n-1}=\left(a^{\frac{n-1}{2}}\right)^2 \equiv \left(\frac{a}{n}\right)^2 = 1 \pmod{n}.$$

Поэтому n является числом Кармайкла, и по свойству (б) утверждения 2 оно должно иметь вид $n=p_1p_2\ldots p_k$, $p_i\neq p_j$. Выберем элемент b, не являющийся квадратичным вычетом по модулю p_1. По китайской теореме об остатках найдется элемент a, удовлетворяющий условиям

$$\begin{cases} a \equiv b \pmod{p_1}, \\ a \equiv 1 \pmod{p_2}, \\ \dots\dots\dots\dots\dots \\ a \equiv 1 \pmod{p_k}. \end{cases}$$

Для этого элемента должно выполняться равенство

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)\left(\frac{a}{p_2}\right)\dots\left(\frac{a}{p_k}\right) = \left(\frac{a}{p_1}\right) = \left(\frac{b}{p_1}\right) = -1.$$

По условию для данного элемента должно выполняться сравнение (2), откуда $a^{\frac{n-1}{2}} \equiv (\frac{a}{n}) \equiv -1 \pmod{p_2}$. Вместе с тем, согласно выбору элемента a должно быть $a \equiv 1 \pmod{p_2}$. Противоречие. □

Числа n, удовлетворяющие сравнению (2) при $a \in \mathbb{Z}_n^*$, называются *эйлеровыми псевдопростыми по основанию* a. В данном случае в силу доказанной выше теоремы получаем, что аналога чисел Кармайкла, которые были бы составными и эйлеровыми псевдопростыми для всех элементов a, здесь нет. Данный результат был независимо получен Д. Лемером в 1976 г. и Р. Соловеем и В. Штрассеном в 1977 г.

Р. Соловей и В. Штрассен предложили следующий вероятностный тест для проверки простоты чисел:

1) выбираем случайное число a из интервала $\{1, 2, \dots, n-1\}$ и проверяем с помощью алгоритма Евклида условие $(a, n) = 1$;

2) если оно не выполняется, то ответ «n — составное»;

3) проверяем выполнимость сравнения (2);

4) если сравнение не выполнено, то ответ «n — составное»;

5) если сравнение выполнено, то ответ неизвестен (и тест можно повторить еще раз).

Сложность данного теста, как и теста на основе малой теоремы Ферма, оценивается величиной $O(\log^3 n)$.

Данный тест полностью аналогичен тесту на основе малой теоремы Ферма, однако, он обладает решающим преимуществом — при его использовании возникает только две ситуации:

— число n простое и тест всегда говорит «не известно»;

— число n составное и тест с вероятностью успеха не меньше $1/2$ дает ответ «n составное».

После повторения теста k раз вероятность неотбраковки составного числа не превосходит $1/2^k$.

Доказательство оценки вероятности успеха вытекает из следующего очевидного утверждения, аналогичного утверждению 1.

Утверждение 2. *Пусть n нечетное составное. Тогда*

(а) *если n эйлерово псевдопростое по основанию $a \in \mathbb{Z}_n^*$, то оно псевдопростое по основанию a;*

(б) *если n эйлерово псевдопростое по основаниям $a, b \in \mathbb{Z}_n^*$, то n эйлерово псевдопростое по основаниям ab и ab^{-1};*

(в) *множество*

$$E_n = \left\{ a \in \mathbb{Z}_n^* \colon a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \pmod{n} \right\}$$

является подгруппой группы $F_n = \{a \in \mathbb{Z}_n \colon a^{n-1} \equiv 1 \pmod{n}\}$;

(г) *если n не является эйлерово псевдопростым по основанию a хотя бы одного числа a, то*

$$|E_n| \leqslant \frac{1}{2} |\mathbb{Z}_n^*|. \quad \square$$

Приведем без доказательства еще один результат.

Теорема. *Если верна обобщенная гипотеза Римана, то существует константа $C > 0$ такая, что если n является Эйлерово псевдопростым для всех оснований a из интервала $1 < a < C \log^2 n$, то n — простое.* □

11.6. Тест Рабина—Миллера

Пусть n — нечетное и $n-1 = 2^s t$, t — нечетное. Если число n является простым, то при всех $a \geqslant 2$ выполняется сравнение $a^{n-1} \equiv 1 \pmod{n}$. Поэтому, рассматривая элементы $a^t, a^{2t}, \dots, a^{2^{s-1}t}$ можно заметить, что либо среди них найдется равный $-1 \pmod{n}$, либо $a^t \equiv 1 \pmod{n}$.

На этом замечании основан следующий вероятностный тест простоты:

1) выбираем случайное число a из интервала $\{1, 2, \dots, n-1\}$ и проверяем с помощью алгоритма Евклида условие $(a, n) = 1$;

2) если оно не выполняется, то ответ «n — составное»;

3) вычисляем $a^t \pmod{n}$;

4) если $a^t \equiv \pm 1 \pmod{n}$, то переходим к п. 1;

5) вычисляем $(a^t)^2, (a^t)^4, \ldots (a^t)^{2^{s-1}} \pmod n$ до тех пор, пока не появится -1;

6) если ни одно из этих чисел не равно -1, то ответ «n — составное»;

7) если мы достигли -1, то ответ неизвестен (и тест можно повторить еще раз).

Арифметическая сложность данного теста, очевидно, составляет $O(sn)$.

Назовем число n *сильно псевдопростым по основанию* a, если выполняется условие:

$$a^t \equiv 1 \pmod n \quad \text{или} \quad \exists r,\ 0 \leqslant r < s,\ a^{2^r t} \equiv -1 \pmod n. \tag{3}$$

Покажем, например, почему число Кармайкла $561 = 3 \cdot 11 \cdot 17$ не является сильно псевдопростым по основанию 2. Имеем $561-1=16\cdot 35$. По китайской теореме об остатках число 2 представляется остатками $(2,2,2)$, поэтому его степени имеют следующий вид:

	mod 3	mod 11	mod 17	mod 561
2^{35}	-1	-1	8	263
$(2^{35})^2$	1	1	13	166
$(2^{35})^4$	1	1	-1	67
$(2^{35})^8$	1	1	1	1
$(2^{35})^{16}$	1	1	1	1

Так как число -1 представляется своими остатками в виде $(-1,-1,-1)$, то в правой колонке число -1 может появиться только в случае, когда имеется строка, состоящая из одних -1. Поскольку числа в колонках ведут себя независимо и случайно, то интуитивно понятно, почему вероятность появления этого события в случае составного n будет мала.

Дж. Миллер в 1975 г. доказал, что как и в методе Соловея—Штрассена сильно псевдопростых чисел по любому основанию, подобных числам Кармайкла, не существует, и предложил детерминированный вариант приведенного выше теста, когда проверяются все числа a не превосходящие $cn^{0.133}$. Он доказал, что данный тест делает $O(n^{1/7})$ шагов и правильно определяет, является ли число n составным или простым. В 1981 г. Л. Эдлеман и Ф. Лэйтон модифицировали алгоритм Миллера и улучшили оценку арифметической сложности до $O(n^{1/10.89})$.

М. Рабин в 1980 г. предложил данный вероятностный вариант теста и доказал, что доля чисел $a \in \mathbb{Z}_n^*$, для которых число n является псевдопростым по данному основанию, не меньше $3/4$. Поэтому после повторения данного теста k раз вероятность неотбраковки составного числа не превосходит $1/4^k$. Оценка $3/4$, по-видимому, неулучшаема, так как, например, для числа $652969351 = 271 \cdot 811 \cdot 2971$ доля чисел a, для которых данное число является сильно псевдопростым, составляет 0.7513, а для числа $2000436751 = 487 \cdot 1531 \cdot 2683$, соответственно, составляет 0.7507.

Наименьшими сильно псевдопростыми числами являются следующие:

a	n
2	$2047 = 23 \cdot 89$
3	$121 = 11 \cdot 11$
5	$781 = 11 \cdot 71$
7	$25 = 5 \cdot 5$

Хотя таких чисел мало, для каждого числа $a > 1$ существует бесконечно много сильно псевдопростых чисел по основанию a. Однако, объединяя тесты для нескольких значений баз, можно получить целый ряд интересных тестов, позволяющих проверять простоту маленьких простых чисел: если $n < 1\,373\,653$ сильно псевдопростое по основаниям 2 и 3, то n — простое; если $n < 25\,326\,001$ сильно псевдопростое по основаниям 2, 3 и 5, то n — простое, если $n < 25\,000\,000\,000$ сильно псевдопростое по основаниям 2, 3, 5 и 7, то либо $n = 3\,215\,031\,751$, либо n — простое; (Померанц, Селфридж, Вогстаф, 1980) если $n < 2\,152\,302\,898\,747$ сильно псевдопростое по основаниям 2, 3, 5, 7 и 11, то n — простое; если $n < 3\,474\,749\,660\,383$ сильно псевдопростое по основаниям 2, 3, 5, 7, 11 и 13, то n — простое; если $n < 341\,550\,071\,728\,321$ сильно псевдопростое по основаниям 2, 3, 5, 7, 11, 13 и 17, то n — простое (Джэшке, 1993); и т. п.

В заключение приведем без доказательства результат Миллера.

Теорема. *Если верна обобщенная гипотеза Римана и n является сильно псевдопростым по основанию a для всех чисел a из интервала $1 < a < 2\log^2 n$, то n — простое.* $\square$

11.7. Полиномиальный тест распознавания простоты

Приведем полиномиальный алгоритм распознавания простоты, появившийся в августе 2002 г. и изложенный в предварительном варианте статьи *Manindra Agrawal, Neeraj Kayal, Nitin Saxena.* PRIMES is in P (http://www.cse.iitk.ac.in/news/primality.pdf).

Алгоритм основан на следующем критерии простоты.

Теорема. *Пусть числа a и n взаимно просты. Тогда n — простое в том и только в том случае, когда выполнено сравнение*

$$(x-a)^n \equiv (x^n - a) \pmod{n}. \tag{1}$$

Доказательство. При $0<i<n$ коэффициент при x^i в выражении $((x-a)^n - (x^n-a))$ равен $(-1)^i\binom{n}{i}a^{n-i}$. Потому, если n — простое, то все коэффициенты сравнимы с нулем.

Пусть n составное и q — простой делитель числа n, такой, что $n = q^k t$, $(q,t)=1$. Тогда q^t не делит $\binom{n}{q}$, взаимно просто с a^{n-q} и, следовательно, коэффициент при x^q не сравним с нулем, что и доказывает теорему. □

При непосредственной проверке этого равенства требуется вычислить значения всех $n-2$ коэффициентов. Поэтому в приведенном ниже алгоритме вместо сравнения (1) рассматриваются сравнения (по двум модулям) вида

$$(x-a)^n \equiv (x^n - a) \pmod{x^r - 1,\ \bmod n}, \tag{2}$$

где значения a и r перебираются специальным образом: сначала ищется «подходящее» значение r, а затем для него проверяется сравнение (2) для всех «малых» значений a.

Приведем сам алгоритм.

```
Вход: целое n > 1.
 1. if (число n имеет вид a^b, b > 1) output «составное»;
 2. r = 2;
 3. while (r < n)
 4.     begin
 5.         if (НОД(n, r) ≠ 1) output «составное»;
 6.         if (r простое)
 7.             вычислить q — наибольший простой делитель r−1;
 8.             if (q > 4√r log n) и (n^((r−1)/q) ≢ 1 (mod r))
 9.                 break;
10.         r ← r + 1;
11.     end
12. if (r = n) output «простое»;
13. for a = 1 to 2√r log n
14.     if ((x − a)^n ≢ (x^n − a) (mod x^r − 1, mod n))
15.         output «составное»;
16. output «простое».
```


Замечание. Шаг 12 алгоритма внесен в связи с тем, что при малых значениях n первый цикл (while) может не найти искомого числа r. Действительно, из неравенств

$$\frac{r-1}{2} \geqslant q \geqslant 4\sqrt{r}\log n$$

получаем $r - 8\sqrt{r}\log n - 1 > 0$. Так как положительный корень уравнения $x^2 - (8\log n)x - 1 = 0$ имеет вид $x = 4\log n + \sqrt{16\log^2 n + 1} > 8\log n$, то $r > 64\log^2 n$. При этом цикл может заканчиваться значением $r = n$ только при простых n, поэтому последующие шаги оказываются ненужными.

Проанализируем алгоритм. Сначала покажем, что для завершения первого цикла (while) достаточно выполнить $O((\log n)^6)$ шагов. Отсюда будет следовать, что во втором цикле (for) надо выполнить $2\sqrt{r}\log n = O((\log n)^4)$ шагов, и поэтому алгоритм будет работать полиномиальное число шагов, каждый из которых имеет полиномиальную сложность.

Воспользуемся фундаментальным результатом Е. Фоуври из аналитической теории чисел (см. [24], [32]), который приведем без доказательства.

Лемма. *Пусть $P(n)$ — наибольший делитель числа n, $\pi_1(x)$ — число простых чисел p, $p \leqslant x$, удовлетворяющих условию $P(p-1) > x^{2/3}$. Тогда найдутся константа $c > 0$ и натуральное n_0 такие, что для всех $n > n_0$ справедлива нижняя оценка*

$$\pi_1(x) \geqslant c\,\frac{x}{\log x}.$$

Теорема 1. *Существуют положительные константы a_1 и a_2 и натуральное n_0 такие, что для всех $n > n_0$ в интервале $[a_1(\log n)^6, a_2(\log n)^6]$ найдется простое число r такое, что $r-1$ имеет простой делитель $q \geqslant 4\sqrt{r}\log n$ и $q \mid o_r(n)$.*

Доказательство. Оценим число N простых чисел в интервале $[a_1(\log n)^6, a_2(\log n)^6]$, удовлетворяющих условию

$$P(r-1) > (a_2(\log n)^6)^{2/3} > r^{2/3}$$

(будем называть такие числа специальными). Согласно теореме

Чебышева при некоторых константах $0 < c_1 < 1 < c_2$ выполняются неравенства

$$c_1 \frac{x}{\ln x} < \pi(x) < c_2 \frac{x}{\ln x}.$$

Поэтому с учетом леммы получаем, что при всех n начиная с некоторого n_0 выполняется цепочка неравенств

$$\begin{aligned} N &\geqslant \pi_1(a_2(\log n)^6) - \pi(a_1(\log n)^6) \geqslant \\ &\geqslant \frac{ca_2(\log n)^6}{\log(a_2(\log n)^6)} - \frac{c_2a_1(\log n)^6}{\log(a_1(\log n)^6)} \geqslant \\ &\geqslant \frac{ca_2(\log n)^6}{7\log\log n} - \frac{c_2a_1(\log n)^6}{6\log\log n} \geqslant \\ &\geqslant \frac{(\log n)^6}{\log\log n}\left(\frac{ca_2}{7} - \frac{c_2a_1}{6}\right) = c_3\frac{(\log n)^6}{\log\log n}, \end{aligned}$$

где константы a_1 и a_2 выбраны так, что $\log a_1 > 0$, $\log a_2 < \log\log n$ и $c_3 > 0$, что всегда можно сделать при достаточно больших n.

Полагаем $x = a_2(\log n)^6$. Тогда произведение

$$\Pi = (n-1)(n^2-1)\cdot\ldots\cdot(n^{x^{2/3}}-1)$$

имеет не более $x^{2/3}\log n$ простых делителей. С другой стороны,

$$x^{2/3}\log n < \frac{c_3(\log n)^6}{\log\log n} < N,$$

и поэтому должно существовать специальное простое число r, не являющееся делителем числа Π.

Это — искомое простое число, так как для него найдется простой делитель $q = P(r-1) > r^{2/3} > 4\sqrt{r}\log n$, удовлетворяющий условию $q \mid o_r(n)$. Действительно,

$$\frac{r-1}{q} \leqslant \frac{r-1}{r^{2/3}} < r^{1/3} < x^{1/3},$$

и по выбору числа Π должно быть

$$n^{\frac{r-1}{q}} \not\equiv 1 \pmod r.$$

С другой стороны, всегда $n^{r-1} \equiv 1 \pmod r$ и, значит, $o_r(n)$ не делит $\frac{r-1}{q}$ и $o_r(n) \mid (r-1)$. Теорема доказана. □

Теорема 2. *Алгоритм асимптотическую сложность $O((\log n)^{12} \times \times pol(\log\log n))$, где $pol(x)$ — некоторый многочлен.*

Доказательство. Будем для краткости использовать обозначение $O^{\sim}(f(n))$ для оценки $O(f(n)pol(\log\log n))$.

Полиномиальность первого шага алгоритма вытекает из того, что если число n имеет вид a^b, $b > 1$, то $b \leqslant \lfloor \log_2 n \rfloor$. Поэтому проверку можно выполнить путем последовательных попыток извлечения корня p-й степени для всех простых чисел $p \leqslant \lfloor \log_2 n \rfloor$. Для нахождения корня можно, например, применить алгоритм последовательного вычисления знаков в двоичной записи числа x, начиная со старшего. А именно, если x — решение уравнения $x^p = n$ и

$$x = \sum_{i=0}^{k} x_i 2^i,$$

то $x_k = 1$, если $(2^k)^p \leqslant n < (2^{k+1})^p$, $x_{k-1} = 1$, если $(2^k + 2^{k-1})^p \leqslant n$, и т. д. Сложность выполнения этого шага при использовании алгоритма умножения Шенхаге—Штрассена и последовательного возведения в квадрат можно оценить величиной $O((\log n)^4 \log\log n) = O^{\sim}((\log n)^4)$.

По теореме 1 в первом цикле (while) выполняется $O((\log n)^6)$ шагов. С учетом размера числа r для выполнения шагов 6 и 7 методом полного перебора всех делителей потребуется не более

$$O(\sqrt{r}\, pol(\log\log n)) = O^{\sim}((\log n)^3)$$

операций. Шаг 8 алгоритма имеет сложность $O(pol(\log\log n))$, поскольку все вычисления можно проводить по модулю r. Поэтому сложность выполнения первого цикла оценивается величиной $O^{\sim}((\log n)^9)$.

Во втором цикле (for) выполняется $2\sqrt{r}\log n = O((\log n)^4)$ шагов. Вычисление левой части в равенстве (2) можно проводить с применением метода повторного возведения в квадрат и алгоритма быстрого преобразования Фурье для умножения многочленов степени r. Поэтому сложность проверки равенства (2) составляет

$$O(r(\log n)^2 pol(\log\log n)) = O^{\sim}((\log n)^5),$$

а выполнения всего второго цикла — $O^{\sim}((\log n)^{12})$. Теорема доказана. □

Докажем теперь, что алгоритм выполняет свою задачу.

Теорема 3. *Алгоритм дает результат «n — простое» в том и только в том случае, когда n — простое.*

Доказательство. Если n простое, то в первом цикле (while) для всех r будет $(n,r)=1$, и результат «n — составное» невозможен. Во втором цикле (for) сравнение (2) также всегда выполнено, поэтому алгоритм может закончиться только с результатом «n — простое».

Пусть n составное. Предположим, что алгоритм дает результат «n — простое».

Если в первом цикле (while) не будет найдено искомое простое число r, то цикл закончится результатом $r=n$. В силу шага 5 алгоритма это может быть только при простых n.

Рассмотрим случай, когда в первом цикле будет найдено простое число r такое, что $r-1$ имеет простой делитель $q \geqslant 4\sqrt{r}\log n$ и $q \mid o_r(n)$. Пусть n имеет в качестве простых делителей числа p_i, $1 \leqslant i \leqslant k$. Поскольку в этом случае $o_r(n)$ должно делить НОК$(o_r(p_1), \ldots, o_r(p_k))$, то найдется простой делитель p числа n, для которого $q \mid o_r(p)$. Так как по предположению результатом работы алгоритма является «n — простое», то во втором цикле (for) для всех $1 \leqslant a \leqslant 2\sqrt{r}\log n$ выполнено сравнение (2), а следовательно, и сравнение

$$(x-a)^n \equiv (x^n - a) \pmod{x^r - 1,\ \bmod p}. \tag{3}$$

Это сравнение рассматривается уже над полем $F_p = GF(p)$. Пусть $h(x) \mid (x^r - 1)$ — неприводимый многочлен $h(x) \neq x-1$. Тогда сравнение (3) можно заменить сравнением

$$(x-a)^n \equiv (x^n - a) \pmod{h(x),\ \bmod p}, \tag{4}$$

которое на самом деле означает равенство элементов в поле $F_p(x)/(h(x))$. Тем самым задача свелась к изучению свойств элементов конечного поля.

Осталось показать, что сравнение (4) для всех $1 \leqslant a \leqslant 2\sqrt{r}\log n$ может выполняться только в случае, когда $n = p^k$ при некотором $k \geqslant 1$. Поскольку числа такого вида уже были отбракованы на первом шагу алгоритма, то полученное противоречие завершит доказательство теоремы.

Оставшуюся часть доказательства разобьем на несколько этапов.

1. Пусть $d = o_r(p)$, и k — степень многочлена $h(x)$. Покажем, что $k = d$.

Согласно выбору многочлена $h(x)$ все его корни должны иметь порядок r (так как их порядок делит r и r — простое). Если $\deg h(x) = k$, то в поле $F_p(x)/(h(x))$ все элементы являются корнями уравнения $X^{p^k} - X = 0$, и значит $r \mid (p^k - 1)$. Отсюда $p^k \equiv 1 \pmod r$ и $d \mid k$.

С другой стороны, так как $r \mid p^d - 1$, то

$$x^{p^d} - x \equiv 0 \pmod{h(x),\ \bmod p}$$

и для каждого элемента $g(x)$ поля $F_p(x)/(h(x))$ выполнено сравнение

$$g(x)^{p^d} \equiv g\big(x^{p^d}\big) \equiv g(x) \pmod{h(x),\ \bmod p}.$$

Значит все элементы поля являются решениями уравнения $X^{p^d} - X = 0$, и $k \leqslant d$.

2. Пусть $l = 2\sqrt{r}\log n$. Покажем, что элементы $(x-a)$, $1 \leqslant a \leqslant l$, порождают в поле $F_p(x)/(h(x)) = GF(p^d)$ циклическую подгруппу G порядка $|G| > n^{2\sqrt{r}}$.

Рассмотрим в группе G подмножество S, состоящее элементов, задаваемых многочленами вида

$$\prod_{1 \leqslant a \leqslant l} (x-a)^{u_a},$$

у которых

$$\sum_{1 \leqslant a \leqslant l} u_a \leqslant d - 1.$$

Покажем, что все такие многочлены задают различные элементы поля. Если бы существовали элементы $1 \leqslant a, a' \leqslant l$ такие, что $a \equiv a' \pmod p$, то было бы $p < l$. Поэтому в силу выбора числа r (как результата работы первого цикла алгоритма), имеем $r > q \geqslant 4\sqrt{r}\log n > l$, и значит $p < r$. Но это противоречит тому, что все делители числа n меньшие r были отсеяны на 5 шаге алгоритма.

Следовательно, все такие многочлены имеют разные корни в F_p, и их степень не превосходит $d-1$. Поэтому им соответствуют различные элементы поля $F_p(x)/(h(x))$.

Число элементов в S равно числу разбиений числа $d-1$ на $l+1$ слагаемых, откуда

$$|G| \geqslant |S| = \binom{l+d-1}{l} = \frac{(l+d-1)(l+d-2)\ldots d}{l!} > \left(\frac{d}{l}\right)^l > 2^l = n^{2\sqrt{r}},$$

поскольку $d = o_r(p) \geqslant q \geqslant 4\sqrt{r}\log n = 2l$.

3. Пусть $g(x)$ — образующий элемент группы G. Пусть

$$I_{g(x)} = \{m \colon g(x)^m \equiv g(x^m) \pmod{x^r - 1,\ \bmod p}\}.$$

Покажем, что множество $I_{g(x)}$ замкнуто относительно умножения.

Пусть $m_1, m_2 \in I_{g(x)}$. Имеем

$$g(x)^{m_1 m_2} \equiv (g(x)^{m_1})^{m_2} \equiv g(x^{m_1})^{m_2} \pmod{x^r - 1,\ \mathrm{mod}\ p}.$$

С другой стороны,

$$g(x^{m_1})^{m_2} \equiv g(x^{m_1 m_2}) \pmod{x^{m_1 r} - 1,\ \mathrm{mod}\ p},$$

откуда

$$g(x^{m_1})^{m_2} \equiv g(x^{m_1 m_2}) \pmod{x^r - 1,\ \mathrm{mod}\ p}.$$

4. Обозначим через o_g порядок элемента $g(x)$ в поле $F_p(x)/(h(x))$, $o_g = |G|$. Докажем, что если $m_1, m_2 \in I_{g(x)}$ и $m_1 \equiv m_2 \pmod r$, то $m_1 \equiv m_2 \pmod{o_g}$.

Пусть $m_2 = m_1 + kr$, $k \geqslant 0$. Тогда в поле $F_p(x)/(h(x))$ справедливы равенства

$$g(x)^{m_1} g(x)^{kr} = g(x)^{m_2} = g(x^{m_2}) = g(x^{m_1 + kr}) = g(x^{m_1}) = g(x)^{m_1}.$$

Так как $g(x) \neq 0$, то получаем $g(x)^{kr} = 1$, откуда $o_g \mid kr$ и $m_1 \equiv m_2 \pmod{o_g}$.

Завершим доказательство теоремы.

Пусть $E = \{n^i p^j : 0 \leqslant i, j \leqslant \lfloor\sqrt{r}\rfloor\}$. В силу п. 3 $E \subseteq I_{g(x)}$. Так как $|E| = (1 + \lfloor\sqrt{r}\rfloor)^2 > r$, то в E найдутся два элемента $n^{i_1} p^{j_1}$ и $n^{i_2} p^{j_2}$ такие, что $(i_1, j_1) \neq (i_2, j_2)$ и $n^{i_1} p^{j_1} \equiv n^{i_2} p^{j_2} \pmod r$. В силу предыдущего пункта получаем $n^{i_1} p^{j_1} \equiv n^{i_2} p^{j_2} \pmod{o_g}$. Отсюда

$$n^{i_1 - i_2} \equiv p^{j_2 - j_1} \pmod{o_g}.$$

Вместе с тем, $n^{|i_1 - i_2|}$, $p^{|j_2 - j_1|} < n^{\sqrt{r}}$, а $o_g > n^{2\sqrt{r}}$. Поэтому $n^{i_1 - i_2} = p^{j_2 - j_1}$, что возможно только в случае, когда $n = p^k$ при некотором $k \geqslant 1$.

Теорема доказана. □

Заметим, что хотя приведенный выше алгоритм и показывает полиномиальность задачи проверки простоты чисел, реальная сложность данного алгоритма настолько высока, что он представляет пока только теоретическое значение. Это связано, с одной стороны, с тем, что асимптотическая оценка леммы 1 начинает эффективно работать только для достаточно больших значений n. Поэтому первый цикл алгоритма найдет искомое число r также только для достаточно больших значений n, а для маленьких простых чисел даст ответ $r = n$, что фактически будет означать проверку простоты полным перебором всех делителей. С другой стороны, известный неполиномиальный детерминированный алгоритм, предложенный Адлеманом, Померанцем и Румли ([23]), и улучшенный Коэном и Ленстрой ([29]), имеет трудоемкость $O(\log n^{\log\log\log n})$, что при всех практически значимых значениях n дает приемлемую оценку, лучшую, чем у данного полиномиального алгоритма.

На практике во многих случаях удобнее использовать более эффективные рандомизированные алгоритмы, наподобие рассмотренных выше тестов Соловея—Штрассена и Рабина—Миллера, позволяющие доказывать простоту числа.

§ 12. Построение больших простых чисел

Рассмотрим теперь такие способы проверки чисел на простоту, в при применении которых можно утверждать, что проверяемые числа действительно являются простыми. В отличие от тестов предыдущего раздела, которые использовали необходимые условия простоты и давали ответы типа «n — не простое», либо «не знаю» или «вероятность того, что n — не простое, не выше заданного сколь угодно малого значения», данные тесты основаны на применении достаточных условий простоты. Поэтому они могут давать как ответы типа «n — не простое», «не знаю», так и «n — простое». Это свойство применяется для построения простых чисел. Общая схема в данном случае такова: выбирается некоторая последовательность чисел специального вида, среди которых требуется найти простое число, затем к числам из этой последовательности применяется тест до тех пор, пока он не даст утвердительный ответ. Если этот ответ «n — не простое», то выбирается следующее число. Если получен ответ «n — простое», то искомое простое число построено.

Рассмотрим достаточные условия простоты чисел, которые обычно применяют в алгоритмах построения доказуемо простых чисел.

12.1. Критерий Люка

Следующая хорошо известная теорема, впервые доказанная Люка в 1876 г., превращает малую теорему Ферма в критерий простоты числа n, достаточное условие которого может быть эффективно использовано для доказательства простоты этого числа.

Теорема (Люка). *Натуральное число n является простым в том и только в том случае, когда выполняется условие*

(1) $\exists a \in \mathbb{Z}_n^*,\ (a^{n-1} \equiv 1 \pmod n) \wedge \big(\forall q \,|\, (n-1),\ a^{\frac{n-1}{q}} \not\equiv 1 \pmod n\big)$.

Доказательство. Если n простое, то в поле $\mathbb{Z}_n$ есть примитивный элемент, который и будет искомым. Наоборот, пусть для элемента a выполняется условие (1). Если $\mathrm{ord}(a)=m$, то $m\,|\,(n-1)$, причем условие (1) гарантирует, что $m=n-1$. Следовательно, $\varphi(n)=n-1$ и n — простое. Теорема доказана. □

Замечание (Сэлфридж). *Условие* (1) *в данной теореме можно заменить на любое из следующих двух условий*:

(2) $\exists\, a\in\mathbb{Z}_n,\ \mathrm{ord}(a)=n-1$;

(3) $\forall\, q\,|\,(n-1),\ \exists\, a\in\mathbb{Z}_n^*,\ (a^{n-1}\equiv 1 \pmod n)\wedge\big(a^{\frac{n-1}{q}}\not\equiv 1 \pmod n\big)$.

Действительно, то, что (1) ⇔ (2) и (1) ⇒ (3), очевидно.

Докажем, что (3) ⇒ (2). Пусть $n-1=q_1^{k_1}\dots q_s^{k_s}$. По условию для каждого i найдется a_i такое, что $\mathrm{ord}(a_i)\,|\,(n-1)$, но $\mathrm{ord}(a_i)$ не делит число $\frac{n-1}{q_i}$. Следовательно, $q_i^{k_i}\,|\,\mathrm{ord}(a_i)$. Значит, найдутся элементы b_i такие, что $\mathrm{ord}(b_i)=q_i^{k_i}$. Теперь элемент $a=b_1\dots b_s$ будет искомым, так как порядки элементов b_i взаимно просты и

$$\mathrm{ord}(b_1\dots b_s)=q_1^{k_1}\dots q_s^{k_s}=n-1. \quad \square$$

Теорема Люка позволяет доказывать простоту числа n в случае, когда известно разложение на простые сомножители числа $n-1$. Для этого можно использовать детерминированный алгоритм, основанный на переборе всех возможных значений $a\in\mathbb{Z}_n^*$, либо воспользоваться следующим вероятностным методом:

1) выбираем случайно числа $a_1,\dots,a_s\in\mathbb{Z}_n^*$ и проверяем для них условие (1);

2) если условие (1) выполнено хотя бы для одного из этих чисел, то n — простое, если нет, то ответ неизвестен.

Аналогичный метод можно построить, используя условие (3).

Проиллюстрируем этот метод применительно к числам Ферма. *Числами Ферма* называются числа вида $F_k=2^{2^k}+1$, $k=1,2,\dots$ (Покажите, что число вида 2^m+1 может быть простым в том и только в том случае, когда $m=2^k$.)

Ферма высказывал предположение, что все числа такого вида — простые. При $n=0,1,2,3,4$ это действительно так. Но при $n=5$, как показал Эйлер в 1732 г., справедливо разложение

$$F_5=2^{2^5}+1=4294967297=641\cdot 6700417.$$

В 1878 г. Иван Михеевич Первушин показал, что числа F_{12} и F_{23} также не являются простыми. (Заметим, что число F_{23} имеет 2525223 цифры. При воспроизведении такого числа понадобилась бы строка длиной в 5 км или книга объемом в 1000 страниц.)

Теорема (Пепин, 1877). *Числа $F_k=2^{2^k}+1$ при $k\geqslant 1$ являются простыми в том и только в том случае, когда выполняется условие*

$$3^{(F_k-1)/2}\equiv -1 \pmod{F_k}.$$

Доказательство. Так как единственным простым делителем числа F_k-1 является 2, то достаточно проверить условие теоремы Люка при $q=2$. Покажем, что в качестве числа a можно взять число 3, т. е. достаточно проверить условие $3^{(F_k-1)/2}\not\equiv 1 \pmod{F_n}$. Используя формулу Эйлера для вычисления значений квадратичных вычетов и квадратичный закон взаимности Гаусса получаем, что при простом F_k должно быть

$$3^{(F_k-1)/2}\equiv\left(\frac{3}{F_k}\right)\equiv(-1)^{(F_k-1)/2}\left(\frac{F_k}{3}\right)\equiv\left(\frac{F_k}{3}\right) \pmod{F_n}.$$

Теперь заметим, что $F_k\not\equiv 1 \pmod 3$, и поэтому условие $F_k\not\equiv 0 \pmod 3$ равносильно равенству $F_k\equiv 2\equiv -1 \pmod 3$. Теорема доказана. □

Теорема Люка послужила отправной точкой для построения целой группы тестов, позволяющих проверять простоту чисел. Многие из них получили название $(n-1)$-методов, так как предполагают знание полной или частичной факторизации числа $n-1$ (см. ниже). Еще одно обобщение теоремы Люка основано на рассмотрении других групп вместо мультипликативной группы $\mathbb{Z}_n^*$. Фактически, доказательство простоты числа n в теореме Люка основано на изучении свойств группы $\mathbb{Z}_n^*$: если каким-либо образом удается установить, что ее порядок равен $n-1$, то число n — простое. Данная идея лежит в основе таких методов, как метод эллиптических кривых и метод числового поля.

12.2. Теорема Поклингтона

В 1914 г. Х. Поклингтоном была доказана следующая теорема.

Теорема (Поклингтон). *Пусть $n=q^kR+1>1$, где q — простое, не являющееся делителем R. Если существует целое a такое, что $a^{n-1}\equiv 1 \pmod n$ и $(a^{(n-1)/q}-1,n)=1$, то каждый простой делитель p числа n имеет вид $p=q^kr+1$ при некотором r.*

Доказательство. Пусть p — простой делитель числа n. Тогда из условия теоремы вытекает, что $a^{n-1}\equiv 1 \pmod p$ и $a^{(n-1)/q}\not\equiv 1 \pmod p$. Отсюда получаем, что порядок m элемента a по модулю p

удовлетворяет условиям: $m \mid (n-1)$ и m не делит $(n-1)/q$. Поэтому $q^k \mid m$. В силу малой теоремы Ферма $m \mid (p-1)$. Следовательно, $p-1 = q^k r$. Теорема доказана.

Применяя данную теорему для всех делителей q числа $n-1$, получаем следующий теорему, которая является обобщением теоремы Люка на случай $R > 1$.

Теорема. *Пусть $n = FR+1 > 1$, где $0 < R < F$. Если для любого простого делителя q числа F существует целое a такое, что $a^{n-1} \equiv 1 \pmod{n}$ и $(a^{(n-1)/q}-1, n) = 1$, то число n — простое.*

Доказательство. Пусть n — составное и p — нетривиальный простой делитель числа n. Заметим, что всегда можно выбрать делитель так, что $p \leqslant \sqrt{n}$. Тогда из условия теоремы вытекает, что для всех простых делителей q числа F существует целое a такое, что $a^{n-1} \equiv 1 \pmod{p}$ и $a^{(n-1)/q} \not\equiv 1 \pmod{p}$. Рассуждая аналогично замечанию к теореме Люка, получаем, что должен найтись элемент, имеющий порядок равный F по модулю p. В силу малой теоремы Ферма $F \leqslant p-1$. Следовательно, справедлива цепочка неравенств

$$p^2 \geqslant (F+1)^2 > R(F+1) \geqslant RF+1 \geqslant n.$$

Но $p \leqslant \sqrt{n}$, противоречие. □

Данная теорема показывает, что если удалось частично факторизовать число $n-1$, причем факторизованная часть удовлетворяет условию $F > \sqrt{n}$, то n — простое.

Прежде, чем переходить к дальнейшему, приведем два классических частных случая данной теоремы.

Теорема (Прот, 1878). *Пусть $n = 2^k R+1$, где $R < 2^k$. Если существует число a, для которого выполняется условие*

$$a^{(n-1)/2} \not\equiv 1 \pmod{n},$$

то n — простое. □

Теорема (Прот, 1878). *Пусть $n = 2^k R+1$, где $R < 2^k$, $3 < 2^k+1$ и 3 не делит R. Тогда n — простое в том и только в том случае, когда выполняется условие*

$$3^{(n-1)/2} \equiv -1 \pmod{n}.$$

Доказательство. В силу теоремы Поклингтона достаточно проверить условие $a^{(n-1)/q} \not\equiv 1 \pmod{n}$ при $a = 3$ и $q = 2$. Так как по условию $n = 2^k R + 1 \not\equiv 1 \pmod{3}$, то условие $3^{(n-1)/2} \neq 1 \pmod{n}$

равносильно выполнению равенства

$$3^{(n-1)/2} \equiv \left(\frac{3}{2^k R+1}\right) \equiv (-1)^{(n-1)/2} \left(\frac{2}{3}\right) \equiv -1 \pmod{F_n}. \quad \square$$

12.3. Теорема Диемитко

Заметим, что если в теореме Поклингтона заменить равенство $(a^{(n-1)/q}-1, n) = 1$ на более слабое условие $a^{(n-1)/q} \not\equiv 1 \pmod{n}$, то можно получить следующий результат.

Лемма. *Пусть $n = q^k R + 1 > 1$, где q — простое, не являющееся делителем R. Если существует целое a такое, что $a^{n-1} \equiv 1 \pmod{n}$ и $a^{(n-1)/q} \not\equiv 1 \pmod{n}$, то найдется простой делитель p числа n вида $p = q^k r + 1$ при некотором r.*

Доказательство. Пусть $n = p_1^{m_1} \ldots p_k^{m_k}$. Тогда из условия теоремы в силу китайской теоремы об остатках вытекает, что существует такое i, что $a^{n-1} \equiv 1 \pmod{p_i^{m_i}}$ и $a^{(n-1)/q} \not\equiv 1 \pmod{p_i^{m_i}}$. Отсюда получаем, что порядок t элемента a по модулю $p_i^{m_i}$ удовлетворяет условиям: $t \mid (n-1)$ и t не делит $(n-1)/q$. Поэтому $q^k \mid t$. В силу леммы Гаусса о цикличности мультипликативной группы кольца $\mathbb{Z}^*_{p_i^{m_i}}$ получаем $t \mid p_i^{m_i-1}(p_i-1)$. Заметим, что числа p_i и q взаимно простые как делители соседних чисел. Поэтому $q^k \mid (p_i - 1)$. Следовательно, $p_i - 1 = q^k r$. □

Хотя этот результат слабее, чем теорема Поклингтона, данный подход, как показал Н. Диемитко в 1988 г., также может быть эффективно использован для доказательства простоты чисел.

Теорема (Диемитко). *Пусть $n = qR+1 > 1$, где q — простое, R — четное и $R < 4(q+1)$. Если существует целое a такое, что $a^{n-1} \equiv 1 \pmod{n}$ и $a^{(n-1)/q} \neq 1 \pmod{n}$, то n — простое.*

Доказательство. Пусть n — не простое и $n = p_1^{m_1} \ldots p_k^{m_k}$. Тогда по лемме получаем, что существует такое i, что $q \mid (p_i - 1)$.

Обозначим $n = p_i Q$. Тогда $n \equiv p_i Q \pmod{q}$, где $n \equiv 1 \pmod{q}$ и $p_i \equiv 1 \pmod{q}$. Отсюда $Q \equiv 1 \pmod{q}$. Следовательно, $Q = qt + 1 \geqslant 2q+1$, где t не может быть равно 0, иначе n простое, или 1, так как Q нечетное. Аналогично, $p_i = qs + 1 \geqslant 2q+1$. Таким образом,

$$n = p_i Q \geqslant (1+2q)^2 = q \cdot 4(q+1) + 1 > qR + 1.$$

Противоречие. Теорема доказана. □

Заметим, что в условии теоремы числа n и R могут быть не взаимно просты.

Эта теорема лежит в основе алгоритма генерации простых чисел в отечественном стандарте на цифровую подпись ГОСТ Р 34.10-94. В нем в качестве a выбираются не очень высокие степени числа 2, а R находится перебором. (Заметим, что с 1 июля 2002 г. этот стандарт был заменен на новый ГОСТ Р 34.10-2001.)

12.4. Метод Маурера

В 1995 г. У. Маурер опубликовал быстрый алгоритм генерации доказуемо простых чисел, близких к случайным. В его основе лежит усиление теоремы Поклингтона на случай, когда факторизованная часть F числа $n-1$ удовлетворяет неравенству $F \geqslant \sqrt[3]{n}$. Кроме того, ему удалось оценить вероятность успеха при случайном поиске числа a в условии теоремы Поклингтона.

Следующая лемма является специальным частным случаем теоремы Поклингтона.

Лемма 1. *Пусть $n = 2FR + 1 > 1$. Если существует целое a такое, что для любого простого делителя q числа F выполнены условия $a^{n-1} \equiv 1 \pmod{n}$ и $(a^{(n-1)/q} - 1, n) = 1$, то каждый простой делитель p числа n имеет вид $p = mF + 1$ при некотором целом m. Если, кроме того, $F > \sqrt{n}$, или F — четное и $R < F$, то n — простое.*

Доказательство. Пусть n — составное и p — нетривиальный простой делитель числа n. Тогда из условия теоремы вытекает, что $a^{n-1} \equiv 1 \pmod{p}$ и $a^{(n-1)/q} \not\equiv 1 \pmod{p}$. Рассуждая аналогично замечанию к теореме Люка, получаем, что должен найтись элемент, имеющий порядок равный F по модулю p. В силу малой теоремы Ферма $F \mid (p-1)$.

Для доказательства второго утверждения, предположим, что $p<n$. Тогда $p \leqslant \sqrt{n}$. Если $F > \sqrt{n}$, то $p = mF+1 > \sqrt{n}$. Если F нечетно и $R<F$, то $p \geqslant 2F+1$ и

$$p^2 \geqslant (2F+1)^2 > (2R+1)(2F+1) > 2RF+1 = n.$$

Противоречие. □

Следующая лемма доказана К. Коувером и Дж. Куискуотером в 1992 г.

Лемма 2. *Пусть n, F, R и a удовлетворяют условию леммы 1. Определим числа $x \geqslant 0$ и $0 \leqslant y < F$ равенством $2R = xF + y$. Если $F \geqslant \sqrt[3]{n}$ и число $y^2 - 4x$ не равно нулю и не является полным квадратом, то n — простое.*

Доказательство. Согласно лемме 1 для каждого простого делителя p числа n выполняется неравенство $p \geqslant F + 1$. По условию $n \leqslant F^3$. Поэтому, если число n — составное, то оно не может иметь боле двух простых делителей. Пусть

$$n = 2RF + 1 = (m_1F+1)(m_2F+1) \quad \text{и} \quad m_1 \geqslant m_2.$$

Тогда

$$2R = m_1m_2F + m_1 + m_2.$$

Имеем $m_1m_2 < F$, иначе $n > F^3$.

Если $m_1 + m_2 \geqslant F$, то $F > m_1m_2 \geqslant m_1(F - m_2) \geqslant F - 1$. Отсюда $m_1 = F-1$, $m_2 = 1$, однако в данном случае $n = F^3 + 1$. Поэтому $m_1 + m_2 < F$.

Следовательно, $m_1m_2 = x$ и $m_1 + m_2 = y$. По теореме Виета m_1, m_2 являются корнями квадратного уравнения $m^2 - ym + x = 0$, которое имеет решения в целых числах в том и только в том случае, если $y^2 - 4x$ является полным квадратом или нулем. Лемма доказана. □

Заметим, что убедиться, что заданное число не является полным квадратом, можно вычислив символ Лежандра для нескольких маленьких простых модулей. Если при некотором модуле число не будет являться квадратичным вычетом, то оно не будет и полным квадратом.

Пусть $\varphi(x)$ — функция Эйлера.

Лемма 3. *Пусть p — простое и $d \mid (p-1)$. Обозначим через T число элементов $x \in \mathbb{Z}_p^*$, порядок которых делится на d. Тогда справедлива оценка*

$$T \geqslant \frac{\varphi(d)}{d}(p-1),$$

причем равенство выполняется в том и только в том случае, когда

$$(d, (p-1)/d) = 1.$$

Доказательство. Используя свойства функции Эйлера получаем

$$T = \sum_{d \mid d^* \mid (p-1)} \varphi(d^*) = \sum_{k \mid (p-1)/d} \varphi(kd) \geqslant$$

$$\geqslant \sum_{k \mid (p-1)/d} \varphi(k)\varphi(d) = \varphi(d) \sum_{k \mid (p-1)/d} \varphi(k) = \varphi(d)\,\frac{p-1}{d},$$

причем равенство выполнено в том и только в том случае, когда $(d, (p-1)/d) = 1$. □

Теорема. *Пусть* $n = 2FR + 1$ — *простое и* $F = q_1^{k_1} \dots q_s^{k_s}$, $F > R$ *и* $(2R, F) = 1$, *где* $q_1, \dots, q_s$ — *различные простые числа. Тогда вероятность того, что случайно выбранное основание* $a \in \mathbb{Z}_n^*$ *в лемме* 1 *будет доказывать простоту числа* n, *равна* $\varphi(F)/F$.

Доказательство. Так как n простое, то сравнение $a^{n-1} \equiv 1 \pmod{n}$ выполняется при всех $a \in \mathbb{Z}_n^*$. При $(2R, F) = 1$ условие $a^{(n-1)/q_j} \equiv 1 \pmod{n}$ при всех j равносильно тому, что порядок элемента a делится на F. Применяя лемму 3 при $d = F$ получаем нужное равенство. □

Следствие. *В условиях теоремы при достаточно больших* $q_1, \dots, q_s$ *в качестве нижней оценки вероятности успеха получаем величину*

$$1 - \sum_{j=1}^{r} \frac{1}{q_j}.$$

Так как $F = q_1^{k_1} \dots q_s^{k_s}$, то искомая оценка вытекает из неравенства

$$\frac{\varphi(F)}{F} = \sum_{j=1}^{s} \left(1 - \frac{1}{q_j}\right) \geqslant 1 - \sum_{j=1}^{s} \frac{1}{q_j},$$

где в случае $F = q_1$, $s = 1$ выполняется равенство. □

У. Маурер предложил следующий рекурсивный алгоритм построения больших простых чисел (впервые идея метода изложена им в 1991 г.). В нем на каждом шаге алгоритма берутся уже построенные на предыдущем этапе простые числа $q_1, \dots, q_s$ и для случайного набора показателей $k_1, \dots, k_s$ вычисляется число $F = q_1^{k_1} \dots q_s^{k_s}$. Затем случайно подбираются числа x, y, $R = R(x, y) < F$ так, чтобы $(2F, R) = 1$ и число $n = 2FR + 1$ при некотором a удовлетворяло условию лемм 1 или 2. По времени данный алгоритм оказывается очень эффективным, так как если n простое, то в силу доказанной теоремы и ее следствия число a находится очень быстро. Если же число n составное, то вероятность выполнения равенства $a^{n-1} \equiv 1 \pmod{n}$ для нескольких баз a мала, если только n не является числом Кармайкла, т. е. является числом очень специального вида. При этом, полученные в результате работы алгоритма простые числа будут иметь практически равномерное распределение. У. Маурер был фактически первым, кто предложил и обосновал использование случайного поиска вместе с достаточными условиями простоты, что позволило строить доказуемо простые числа.

12.5. Метод Михалеску

В 1994 г. П. Михалеску предложил еще более быстрый алгоритм генерации доказуемо простых чисел. По мнению Михалеску в методе Маурера потеря в эффективности происходит из-за усложненного механизма для построения простых чисел с «почти» равномерным распределением. Он предложил алгоритм поиска чисел в арифметической прогрессии, для всех чисел n из которой известна факторизация некоторой части числа $n - 1$. Это позволило ему использовать рекурсивную процедуру. Кроме того, он оптимизировал процедуру отбраковки составных чисел, используя решето Эратофена и тест простоты Рабина—Миллера.

Предложенный им алгоритм генерации доказуемо простых m-разрядных чисел состоит в следующем. Пусть $B > 0$ — целое число и $s, c > 0$ — действительные константы.

Шаг 1. Если $m < B$, то алгоритм возвращает случайное простое число с m двоичными разрядами (порожденное с помощью пробных делений).

Шаг 2. Строим с помощью рекурсии целое число F из интервала $2^{\varepsilon m} < F < 2^{c\varepsilon m}$, факторизация которого полностью известна, где ε можно положить 1/2 или 1/3 в зависимости от применяемого достаточного условия простоты, см. леммы 1 или 2 в методе Маурера.

Шаг 3. Выбираем случайное число $t \in (2^{m-2}/F, 2^{m-1}/F - sm)$.

Шаг 4. Ищем простое число в арифметической прогрессии

$$P = \{n = n_0 + ia \colon n_0 = ta + 1,\ a = 2F,\ 0 \leqslant i \leqslant s\}.$$

Тест простоты, применяемый на шаге 4, выполняется в три этапа.

Этап 1. Пробные деления на простые числа, не превосходящие A, где A — заданная верхняя граница.

Этап 2. Проверка простоты с помощью теста Рабина—Миллера.

Этап 3. Доказательство простоты с помощью лемм 1 или 2 (см. метод Маурера).

Выбирая подходящим образом параметры B, s и c можно оптимизировать данный алгоритм. При этом потери, связанные с усложнением процедуры доказательства простоты для случая, когда проверяемое число является простым (выполняется редко в алгоритме), во многом компенсируются ускоренной отбраковкой составных чисел (выполняется часто в алгоритме).

12.6. $(n+1)$-методы

Рассмотренные выше методы относятся к так называемым $(n-1)$-методам, которые применимы в случае, когда известна полная или частичная факторизация числа $(n-1)$. Еще один интересный класс методов составляют $(n+1)$-методы, в которых предполагается известной факторизация числа $(n+1)$.

Пусть p и q — такие целые числа, что p^2-4q не является квадратичным вычетом по модулю n. Тогда квадратное уравнение $x^2-px+q=0$ имеет различные корни, один из которых равен $r=(p+\sqrt{p^2-4q})/2$. Индукцией легко показать, что степени этого числа имеют специальный вид.

Лемма 1. *Степени числа r имеют вид $r^k=(V_k+U_k\sqrt{p^2-4q})/2$, где последовательности $\{U_k\}$, $\{V_k\}$ определяются рекуррентными соотношениями*

$$\begin{aligned} &U_0=0, &&U_1=1, &&U_{k+2}=pU_{k+1}-qU_k,\\ &V_0=2, &&V_1=p, &&V_{k+2}=pV_{k+1}-qV_k, \quad k\geqslant 0. \end{aligned}$$

Данные последовательности называются последовательностями Люка, ассоциированными с числами p и q, и имеют много свойств, позволяющих вычислять их аналогично методу повторного возведения в квадрат при возведении в степень, например,

$$U_{2k}=U_kV_k, \quad V_{2k}=V_k^2-2q^k, \quad k\geqslant 0.$$

(При $p=1$, $q=-1$ последовательность $\{U_k\}$ совпадает с последовательностью Фибоначчи.) Следующая лемма играет роль, аналогичную малой теореме Ферма.

Лемма 2. *Пусть числа p, q и r определены выше и $2r=a+b\sqrt{p^2-4q} \pmod n$ для некоторых чисел a и b одинаковой четности. Если n простое, то $2r^n=a-b\sqrt{p^2-4q} \pmod n$.* □

В терминах последовательности $\{U_k\}$ эта лемма имеет вид

Лемма 3. *Если n простое, то $U_{n+1}\equiv 0 \pmod n$.* □

Теорема. *Пусть $n>1$ — нечетное число. Если для любого простого делителя r числа $n+1$ существуют такие простые числа p и q, что p^2-4q не является квадратичным вычетом по модулю n, и такие, что*

$$U_{n+1}\equiv 0 \pmod n, \quad U_{(n+1)/r}\not\equiv 0 \pmod n,$$

то n — простое. □

Данная теорема играет роль, в точности аналогичную роли теоремы Люка $(n-1)$-методах. Аналогично можно использовать и вторую последовательность $\{V_k\}$. Также, аналогично $(n-1)$-методам, можно использовать частичное разложение $n+1$ на множители. Заметим, что имеются тесты, основанные на разложении чисел n^2+1, $n^2\pm n-1$.

12.7. Числа Мерсенна

Наиболее известным частным случаем этого подхода является следующий критерий простоты для чисел Мерсенна. Числами Мерсенна называются числа вида $M_n=2^n-1$, где n — простое. Легко видеть, что число M_n может быть простым только если n — простое.

Теорема (Люка—Лемер). *Пусть n нечетное число, и последовательность $\{L_m\}$ определена рекуррентным образом*

$$L_0=4, \quad L_{m+1}=L_m^2-2, \quad 0\leqslant m<n.$$

Число M_n простое тогда и только тогда, когда $L_{n-2}\equiv 0 \pmod n$. □

Критерий был первоначально открыт Люка в конце 1890-х гг., а данную краткую форму приобрел около 1930 г. в работах Лемера. Легко видеть, что частный случай $(n+1)$-метода при $q=2$, $k=2^m$, $L_m=V_{2k}/2^k$.

В таблице 1 приведен список всех известных в настоящее время простых чисел p, для которых число M_p является простым (в свою очередь число $P_p=M_p(M_p-1)$ является совершенным). Вопросительные знаки в таблице означают, что пока проверены не все числа, и поэтому неизвестно, являются ли они подряд идущими простыми числами Мерсенна.

Последние пять простых чисел были найдены в рамках программы широкомасштабного поиска, организованного с 1995 г. в сети Интернет — GIMPS (the Great Internet Mersenne Prime Search) — Г. Уолманом, написавшим быструю программу для персонального компьютера и разместившим ее на своем web-сервере. Он организовал также распределенную базу данных, в которой отражался ход поиска. В 1997 г. компание Entropia, Inc., основанной С. Куровски, была организована система поддержки распределенных вычислений PrimeNet, которая в настоящее время координирует работу нескольких сотен тысяч компьютеров. (В последнем столбце таблицы 1 в скобках указано число компьютеров, принимавших участие в поиске.)

§ 13. Алгоритмы факторизации целых чисел

Задача факторизации целого числа n заключается в нахождении разложения его в произведение простых сомножителей. Тривиальный алгоритм, часто называемый алгоритмом «*пробных делений*»,

Таблица 1. Известные числа Мерсенна

№	p	*цифр в* M_p	*цифр в* P_p	*год*	*автор*	*компьютер*
1	2	1	1	–	–	–
2	3	1	2	–	–	–
3	5	2	3	–	–	–
4	7	3	4	–	–	–
5	13	4	8	1456	неизв.	–
6	17	6	10	1588	Cataldi	–
7	19	6	12	1588	— » —	–
8	31	10	19	1772	Эйлер	–
9	61	19	37	1883	Первушин	–
10	89	27	54	1911	Powers	–
11	107	33	65	1914	— » —	–
12	127	39	77	1876	Lucas	–
13	521	157	314	1952	Robinson	SWAC
14	607	183	366	1952	Lehmer & Robinson	SWAC
15	1279	386	770	1952	— » —	SWAC
16	2203	664	1327	1952	— » —	SWAC
17	2281	687	1373	1952	— » —	SWAC
18	3217	969	1937	1957	Riesel	BESK
19	4253	1281	2561	1961	Hurwitz & Selfridge	IBM 7090
20	4423	1332	2663	1961	— » —	IBM 7090
21	9689	2917	5834	1963	Gillies	ILIAC 2
22	9941	2993	5985	1963	— » —	ILIAC 2
23	11213	3376	6751	1963	— » —	ILIAC 2
24	19937	6002	12003	1971	Tuckerman	IBM 360
25	21701	6533	13066	1978	Noll & Nickel	CDC 174
26	23209	6987	13973	1979	Noll	CDC 174
27	44497	13395	26790	1979	Nelson & Slowinski	CRAY-1
28	86243	25962	51924	1982	Slowinski	CRAY
29	110503	33265	66530	1988	Colquitt & Welsh	SGI
30	132049	39751	79502	1983	Slowinski	CRAY
31	216091	65050	130100	1985	— » —	CRAY
32	756839	227832	455663	1992	Slowinski & Gage	CRAY
33	859433	258716	517430	1994	— » —	CRAY
34	1257787	378632	757263	1996	— » —	CRAY-T94
35	1398269	420921	841842	1996	Armengaud, Woltman, et. al. (GIMPS)	Pentium 90
36	2976221	895932	1791864	1997	Spence, Woltman, et. al. (GIMPS)	Pentium 100 (2 000)
37	3021377	909526	1819050	1998	Clarkson, Woltman, Kurowski et. al. (GIMPS, PrimeNet)	Pentium 200 (4 000)
??	6972593	2098960	4197919	1999	Hairatwala, Woltman, Kurowski et. al. (GIMPS, PrimeNet)	Pentium II-350 (21 500)
??	13466917	4053946	8107892	2001	Cameron, Woltman, Kurowski et. al. (GIMPS, PrimeNet)	AMD 800 (205 000)

заключается в последовательном делении числа n на всевозможные простые числа, не превосходящие $\sqrt{n}$. При выполнении данного алгоритма возникает две трудности. С одной стороны, для больших чисел n число операций деления, которые необходимо выполнить, может оказаться настолько большим, что его просто невозможно выполнить. С другой стороны, необходимо заранее строить список всех простых чисел, не превосходящих $\sqrt{n}$, что уже само по себе является трудной задачей. Второй трудности можно легко избежать, если осуществлять деления числа n не на простые, а на все целые числа k, не превосходящие $\sqrt{n}$. Это избавит от необходимости строить исходный список простых чисел, причем сложность такого алгоритма увеличится не более, чем в $O(\log n)$ раз.

Сложность нахождения первого нетривиального делителя числа n для данного алгоритма оценивается величиной

$$O(\pi(\sqrt{n})\log^2 n) = O(\sqrt{n}\log n)$$

для деления на простые числа, и

$$O(\sqrt{n}\log^2 n)$$

для деления на подряд идущие целые числа.

Заметим, что при анализе алгоритмов факторизации часто ограничиваются рассмотрением только первого шага — нахождения первого нетривиального делителя. Легко видеть, что для нахождения полной факторизации числа n необходимо рекурсивно повторить выполнение данного шага не более $\log_2 n$ раз.

Хотя применение алгоритма «пробных делений» в полном объеме для больших n практически невозможно, он все равно находит широкое применение в своем «сокращенном» варианте, когда пробные деления осуществляются только на все простые числа, не превосходящие некоторой константы C.

Граница C	Число простых $\pi(C)$
256	54
1 000	168
10 000	1 229
100 000	9 592
1000 000	78 498
10 000 000	664 579
100 000 000	5 761 455

В таком виде алгоритм «пробных делений» входит составной частью практически во все современные методы факторизации.

Приступая к факторизации числа n, следует сначала убедиться, что оно не простое. Это можно сделать с помощью одного из описанных выше тестов простоты. Их трудоемкость, как правило, значительно меньше, чем у алгоритмов факторизации. Поэтому далее число n будет заведомо считаться составным.

13.1. Метод Полларда

Наиболее популярным вероятностным алгоритмом факторизации является так называемый «ρ-метод», предложенный Дж. Поллардом в 1975 г.

Алгоритм 1

Шаг 1. Выбираем многочлен $f(x) \in \mathbb{Z}[x]$ и число ω.

Шаг 2. Случайно выбираем $x_0 \in \mathbb{Z}_n$ и, вычисляя значения $x_i = f(x_{i-1}) \bmod n$, $i = 1, \ldots, m$, проверяем тест шага 3.

Шаг 3. Для $p_0 + 2k'rs$ полагаем $j = 2^h - 1$ и для каждого $2^h \leqslant k < 2^{h+1}$ вычисляем $d = (x_j - x_k, n)$. Если $1 < d < n$, то нетривиальный делитель числа n найден. Если $d = 1$ или $d = n$, то переходим к следующему значению h.

Чтобы оценить среднее время работы данного алгоритма, рассмотрим сначала более простой для анализа, но менее эффективный алгоритм, отличающийся от алгоритма 1 тем, что на шаге 3 вычисляется НОД$(x_j - x_k, n)$ для всех чисел пар чисел j, k, $0 \leqslant j < k \leqslant m$. Назовем его Алгоритм 2.

Воспользуемся известным «парадоксом дней рождения».

Теорема. *Пусть $\lambda > 0$. Для случайной выборки из n элементов объема $l + 1$, где $l = \sqrt{2\lambda n}$, вероятность того, что все элементы в выборке будут попарно различными, допускает следующую оценку сверху*:

$$p_{nl} < e^{-\lambda}.$$

Доказательство. Имеем

$$p_{nl} = \frac{n(n-1)\ldots(n-l)}{n^{l+1}} = \prod_{i=1}^{l} \left(1 - \frac{i}{n}\right).$$

Переходя к логарифмам и учитывая, что $\log(1-x) < -x$ при $0 < x < 1$,

получаем

$$\log p_{nl} < \sum_{i=1}^{l} \left(-\frac{i}{n}\right) = -\frac{l(l+1)}{2n} < -\frac{l^2}{2n} = -\lambda.$$

Теорема доказана. $\square$

В нашем случае будем рассматривать последовательность $x_0, x_1, \ldots, x_m$, $x_i = f(x_{i-1})$, $j = 1, \ldots, m$, как случайную выборку. Для случайного отображения $f\colon \mathbb{Z}_n \to \mathbb{Z}_n$ это оправдано. Хотя многочлен $f(x) \in \mathbb{Z}[x]$, конечно, нельзя рассматривать как случайное отображение, тем не менее, практические результаты говорят о хорошем соответствии приведенных ниже оценок трудоемкости с теми, которые получаются в действительности.

Пусть p — нетривиальный делитель числа n. Тогда условие $p \mid (x_j - x_k, n)$ равносильно условию $p \mid (x_j - x_k)$, или $x_j \equiv x_k \pmod p$. Поэтому, применяя доказанную выше теорему для выборки, образованной последовательностью $x'_0, \ldots, x'_l$, где $x'_i = x_i \bmod p$, получаем, что при $l = \sqrt{2\lambda n}$ вероятность существования пары i, k, $0 \leqslant i < k \leqslant l$ такой, что $p \mid (x_j - x_k, n)$, будет не меньше, чем $1 - e^{-\lambda}$.

Так как $p < \sqrt{n}$, то вероятность одновременного выполнения равенства $x_j \equiv x_k \pmod n$ будет мала, и можно считать, что $x_j \not\equiv x_k \pmod n$, и данная пара позволяет факторизовать число n.

Таким образом, для нахождения данной пары с надежностью $1 - e^{-\lambda}$ необходимо вычислить $l + 1$ знаков последовательности $\{x_i\}$, где $l = \sqrt{2\lambda p}$. Отсюда средняя сложность алгоритма 2 оценивается величиной $O(l^2 \log^3 n) = O(\lambda\sqrt{n} \log^3 n)$.

Перейдем теперь к рассмотрению алгоритма 1. Заметим, что поскольку $f(x)$ — многочлен над $\mathbb{Z}$, то для любого делителя p числа n и любой пары $0 \leqslant j < k$ с условием $x_j \equiv x_k \pmod p$ будут выполняться и сравнения $x_{j+1} \equiv x_{k+1} \pmod p$, $x_{j+2} \equiv x_{k+2} \pmod p$ и т. д.

Пусть алгоритму 2 для нахождения пары (j, k) с условием $(x_j - x_k, n) \neq 1$ необходима последовательность длины $l + 1$. Покажем, что в этом случае алгоритм 1 позволяет найти пару (j', k') с условием $(x_{j'} - x_{k'}, n) \neq 1$, где $j' = j + s$, $k' = k + s$ при некотором s, где (j, k) — пара, найденная алгоритмом 2, такая, что $k' < 4k$. Пусть число k имеет в своей двоичной записи $1 + h$ бит, т. е. $2^h \leqslant k < 2^{h+1}$. Тогда при $j' = 2^{h+1} - 1$ число $k' = k + (j' - j)$ будет находиться в интервале $2^{h+1} \leqslant k' < 2^{h+2}$, поскольку

$$k' = k + (j' - j) < 2^{h+1} + 2^{h+1} - 1 - j < 2^{h+2},$$

причем так как $2^h \leqslant k$, то $k' < 4k$.

Таким образом, если второму алгоритму потребуется последовательность длины $l+1$, то первому алгоритму необходима последовательность длины $4l+1$. При этом сложность первого алгоритма оценивается уже величиной $O(\sqrt{\lambda}\sqrt[4]{n}\log^3 n)$, что гораздо лучше, чем $O(\lambda\sqrt{n}\log^3 n)$. Тем самым доказана

Теорема. *Пусть n — составное число. Существует константа C такая, что для любого положительного числа λ вероятность события, состоящего в том, что ρ-метод Полларда не найдет нетривиального делителя n за время $C\sqrt{\lambda}\sqrt[4]{n}\log^3 n$, не превосходит величины $e^{-\lambda}$.*

13.2. Алгоритм Полларда—Штрассена

Алгоритм Полларда—Штрассена является детерминированным алгоритмом для нахождения минимального простого делителя и имеет оценку сложности $O(\sqrt[4]{n}\log^4 n)$. Он основан на следующей теореме.

Теорема. *Пусть $z\in\mathbb{N}$, $y=z^2$. Тогда для любого $t\in\mathbb{N}$ наименьший простой делитель числа $(t,y!)$ может быть найден за $O(z\log^2 z\log^2 t)$ двоичных операций.*

Доказательство. Сгруппируем сомножители в выражении

$$y! = (1\cdot 2\cdot\ldots\cdot z)[(z+1)\cdot\ldots\cdot(2z+1)]\ldots[((z-1)z+1)\cdot\ldots\cdot z^2] = \\ = \prod_{j=1}^{z}\frac{(jz)!}{((j-1)z)!} = f(1)f(2)\ldots f(z),$$

где $f(j)=((j-1)z+1)\ldots((j-1)z+z)$, $j=1,\ldots,z$. Поэтому для нахождения наименьшего простого делителя числа $(t,y!)$ можно воспользоваться следующим способом:

Шаг 1. Вычисляем значения $f(1),\ldots,f(z)$.

Шаг 2. Вычисляем $(t,f(j))$, $j=1,2,\ldots,z$ до получения первого нетривиального делителя.

Шаг 3. Последовательно осуществляем пробные деления $(t,f(j))$ на числа $(j-1)z+1,\ldots,(j-1)z+z$. Первое число из этого интервала, делящее нацело $(t,f(j))$, очевидно, и будет минимальным простым делителем числа $(t,y!)$.

Оценим теперь сложность выполнения каждого шага. Для выполнения первого шага найдем коэффициенты многочлена

$$f(x)=((x-1)z+1)\ldots((x-1)z+z).$$

Воспользуемся алгоритмом быстрого дискретного преобразования

Фурье и последовательно найдем $z/2$ произведений многочленов первой степени, затем $z/4$ произведений многочленов второй степени, и т.д. В итоге получается

$$\sum_{i=1}^{\log z}\frac{z}{2^i}\,2^i\log 2^i \leqslant z\log^2 z$$

арифметических операций. При его выполнении необходимо выполнять операции сложения и умножения целых чисел. Заметим, что нам достаточно знать значения $f(j) \bmod t$. В качестве операций здесь выступают умножение и деление с остатком, имеющие сложность $O(\log^2 t)$. Поэтому сложность нахождения коэффициентов многочлена равна $O(z\log^2 z\log^2 t)$. Вычисление значений теперь выполняется по той же схеме, только в качестве линейных сомножителей выступают многочлены $(x-1),\ldots,(x-z)$. Поэтому сложность первого этапа равна $O(z\log^2 z\log^2 t)$.

На втором шаге для каждого j один раз выполняется алгоритм нахождения НОД целых чисел. Двоичная сложность алгоритма нахождения НОД двух $\log t$-разрядных чисел равна $O(\log^2 t)$. Поэтому сложность второго этапа равна $O(z\log^2 t)$.

На третьем этапе выполняется не более z операций деления $\log t$-разрядного числа на $2\log z$-разрядное число. Поэтому сложность этого этапа равна $O(z\log t\log z)$. Теорема доказана. $\square$

Для факторизации числа n следует положить $z=\lfloor\sqrt[4]{n}\rfloor+1$, $y=z^2$, $t=n$, откуда вытекает, что сложность нахождения наименьшего простого делителя числа n данным алгоритмом равна $O(\sqrt[4]{n}\log^4 n)$ двоичных операций. Если положить $z<\lfloor\sqrt[4]{n}\rfloor+1$, то алгоритм будет искать только маленькие простые делители числа n.

Заметим, что современные методы факторизации на практике часто выполняются в 3 этапа.

Этап 1. Пробные деления на $1\div 2$ тысячи первых простых чисел.

Этап 2. Нахождение маленьких простых делителей методом Полларда или методом Полларда—Штрассена (в котором число z подбирается из соображений оптимизации общей трудоемкости алгоритма).

Этап 3. Для нахождения больших простых делителей применяется один из субэкспоненциальных алгоритмов.

13.3. Факторизация Ферма

Весьма плодотворной идеей при построении алгоритмов факторизации является поиск целых чисел x и y, для которых выполняется соотношение $x^2 \equiv y^2 \pmod n$. Если при этом $x \not\equiv \pm y \pmod n$, то числа $(x+y, n)$ и $(x-y, n)$ суть нетривиальные делители числа n.

По-видимому, первым в этом направлении является метод факторизации, примененный П. Ферма. Он основан на теореме Эйлера о представлении числа в виде разности двух квадратов.

Теорема. *Если $n>1$ — нечетно, то существует взаимно однозначное соответствие между разложениями на множители $n = a \cdot b$, $a \geqslant b > 0$ и представлениями в виде разности квадратов $n = x^2 - y^2$, $x > y > 0$. Это соответствие имеет вид*

$$x = \frac{a+b}{2}, \quad y = \frac{a-b}{2},$$
$$a = x+y, \quad b = x-y.$$

Доказательство очевидно. □

Метод Ферма заключается в том, что при малых значениях параметра y в представлении $n = x^2 - y^2$ можно найти пару (x, y), перебирая в качестве кандидатов на значение x числа $\lfloor\sqrt{n}\rfloor + 1, \lfloor\sqrt{n}\rfloor + 2, \ldots$ и проверяя для каждого из них равенства

$$(\lfloor\sqrt{n}\rfloor + i)^2 - n = y^2.$$

Для отбраковки ложных значений x можно воспользоваться тем, что если число не является квадратом, то оно с большой вероятностью не будет и квадратичным вычетом для одного из небольших простых чисел p. Последнее свойство легко проверяется путем вычисления соответствующего символа Лежандра.

Вход: n — нечетное число, $p_1, \ldots, p_k$ — небольшие простые числа.

Шаг 0. Проверить $p_i \mid n$, $i = 1, \ldots, k$. Если да, то делитель найден.

Шаг 1. Для каждого x от $\lfloor\sqrt{n}\rfloor + 1$ до $\lfloor\sqrt{n}\rfloor + n_0$ вычислить велич́ины

$$t = x^2 - n, \quad t_i = t \bmod p_i, \quad i = 1, \ldots, k.$$

Шаг 2. Если хотя бы для одного $i = 1, \ldots, k$ выполнено одно из условий:
— $t_i = 0$ и p_i^2 не делит t; или
— $t_i \neq 0$ и $\left(\frac{t_i}{p_i}\right) = -1$,
то перейти к следующему x на шаге 1. В противном случае перейти к шагу 3.

Шаг 3. Проверить, является ли $t = x^2 - n$ полным квадратом. Если $x^2 - n = y^2$, то выдать ответ: «n — составное, $n = a \cdot b$, $a = x+y$, $b = x-y$». Если $t = x^2 - n$ — не полный квадрат, то перейти к следующему x на шаге 1.

По сути, данный алгоритм, подобно методу «пробных делений», является разновидностью метода перебора всех возможных делителей. Сложность этого алгоритма оценивается величиной $O(n_0 \log^2 n)$. Параметр n_0 определяется практически из конкретных вычислительных возможностей. Чем больше значение n_0, тем большее число возможных делителей будет проверено. Однако, в отличие от метода «пробных делений», с помощью которого находится наименьший делитель, данный метод позволяет находить наибольший делитель числа n, не превосходящий $\sqrt{n}$.

Заметим, что вместо $\sqrt{n}$ можно брать $\sqrt{kn}$ при малых значениях k. Во многих случаях переход к уравнениям $(\lfloor\sqrt{kn}\rfloor + i)^2 - kn = x^2$, $k = 3, 5, \ldots$ позволяет найти искомое значение x путем более короткого перебора значений i, чем при $k = 1$. При этом, в случае успеха такое представление также позволяет факторизовать число n, так как числа

$$(\lfloor\sqrt{kn}\rfloor + i + x), \ (\lfloor\sqrt{kn}\rfloor + i - x)$$

лежат в интервале $(0, n)$ и удовлетворяют равенству

$$(\lfloor\sqrt{kn}\rfloor + i + x)(\lfloor\sqrt{kn}\rfloor + i - x) = kn.$$

13.4. Алгоритм Диксона

Во многих современных алгоритмах факторизации для нахождения делителей используется идея Лежандра (1798 г.), заключающаяся в поиске пары целых чисел x и y, удовлетворяющих условиям

$$x^2 \equiv y^2 \pmod n, \quad x \not\equiv \pm y \pmod n.$$

Этот подход является обобщением метода Ферма, в котором требуется выполнение строгого равенства.

Для поиска таких чисел воспользуемся понятием факторной базы.

Назовем факторной базой некоторое множество $B = \{p_1, p_2, \ldots, p_h\}$ небольших простых чисел. Обычно в качестве $p_1, p_2, \ldots, p_h$ берут простые числа, не превосходящие некоторой границы M, $h = \pi(M)$.

Будем говорить, что целое число $b \in \mathbb{N}$ является B-числом, если число $b^2 \bmod n$ разлагается в произведение простых чисел из факторной базы:

$$b^2 \bmod n = \prod_{p \in B} p^{\alpha_p(b)}.$$

Сопоставим каждому B-числу вектор показателей из этого разложения

$$\vec{\alpha}(b) = (\alpha_{p_1}(b), \ldots, \alpha_{p_h}(b)),$$

а также двоичный вектор, полученный из вектора $\vec{\alpha}(b)$ приведением всех его координат по модулю 2,

$$\vec{\varepsilon}(b) = (\alpha_{p_1}(b) \bmod 2, \ldots, \alpha_{p_h}(b) \bmod 2).$$

Если теперь каким-либо способом подобрать такое множество различных B-чисел $b_1, \ldots, b_m$, при котором выполняется линейное соотношение

$$\vec{\varepsilon}(b_1) \oplus \ldots \oplus \vec{\varepsilon}(b_m) = \vec{0},$$

то для произведения $x = b_1 \ldots b_m$ выполняется соотношение

$$x^2 \equiv y^2 \pmod n,$$

где число y определяется по векторам показателей равенством

$$y = \prod_{p \in B} p^{\frac{1}{2}(\vec{\alpha}_p(b_1) + \ldots + \vec{\alpha}_p(b_m))}.$$

Алгоритм Диксона заключается в следующем.

Шаг 1. Выбрать случайное b, $1 < b < n$, и вычислить $b^2 \bmod m$.

Шаг 2. Пробными делениями попытаться разложить $b^2 \bmod m$ на простые множители из факторной базы.

Шаг 3. Если b является B-числом, т. е.

$$b^2 \bmod n = \prod_{p \in B} p^{\alpha_p(b)},$$

то запомнить $\vec{\alpha}(b)$ и $\vec{\varepsilon}(b)$. Повторять процедуру генерации чисел b до тех пор, пока не будет найдено $m = h + 1$ B-чисел $b_1, \ldots, b_m$.

Шаг 4. Найти (например, решая с помощью алгоритма последовательного исключения неизвестных Гаусса однородную систему линейных уравнений $x_1\vec{\varepsilon}(b_1) \oplus \ldots \oplus x_m\vec{\varepsilon}(b_m) = \vec{0}$ относительно неизвестных $(x_1, \ldots, x_m)$) соотношение линейной зависимости

$$\vec{\varepsilon}(b_{i_1}) \oplus \ldots \oplus \vec{\varepsilon}(b_{i_t}) = \vec{0}, \quad 1 < t \leqslant m.$$

Положить

$$x = b_{i_1} \ldots b_{i_t}, \quad y = \prod_{p \in B} p^{\frac{1}{2}(\vec{\alpha}_p(b_{i_1}) + \ldots + \vec{\alpha}_p(b_{i_t}))}.$$

Шаг 5. Проверить $x \equiv \pm y \pmod n$. Если это так, то повторить процедуру генерации. Если нет, то найдено нетривиальное разложение

$$n = u \cdot v, \quad u = (x + y, n), \quad v = (x - y, n).$$

На трудоемкость алгоритма Диксона существенно влияет выбор факторной базы. Если число h выбрано так, что $M \approx \frac{\sqrt{n}}{2}$, то почти каждое число b будет B-числом, но получаемые при этом сравнения $x^2 \equiv y^2 \pmod n$ будут тривиальными. Кроме того, надо будет решать системы от очень большого числа неизвестных. Если h — мало, то B-числа будут появляться очень редко.

Заметим, что если число n — составное, то уравнение $x^2 \equiv a^2 \pmod n$ имеет по крайней мере 4 решения (докажите это в качестве упражнения). Поэтому вероятность появления пары (x, y) с $x \equiv \pm y \pmod n$ не превосходит 1/2. Следовательно, повторяя процедуру набора для получения нужной пары k раз мы получим, что надежность данного метода нахождения делителя не меньше, чем $1 - 2^{-k}$.

Обозначим через $\Psi(n, M)$ число целых чисел a в интервале $1 < a \leqslant n$, раскладывающихся в произведение простых чисел из множества $B = \{p \colon p \leqslant M\}$.

Из результатов Н. Де-Брейна (1966) и Е. Кэпфилда, Р. Эрдоша и К. Померанца (1983) вытекает следующая

Теорема (Де-Брейн, Кэпфилд, Эрдош, Померанц). *Для любых $\varepsilon > 0$, $x \geqslant 10$, $u \leqslant (\log x)^{1-\varepsilon}$ при $u \to \infty$ равномерно по x выполняется соотношение*

$$\Psi\big(x, x^{1/u}\big) = x \cdot u^{-u(1+o(1))}.$$

Отсюда при $u = \frac{\log x}{\log y}$ получаем

$$\Psi(x, y) = x \cdot u^{-u(1+o(1))}.$$

Поэтому в качестве приближенного значения вероятности получения числа, раскладывающегося в произведение простых чисел из множества $B = \{p \colon p \leqslant M\}$, при случайном выборе чисел в интервале $[2, n]$ можно использовать величину

$$\frac{\Psi(n, M)}{n} = u^{-u},$$

где $u = \frac{\ln n}{\ln M}$. Эту же величину можно принять в качестве приближенного значения вероятности появления B-числа на шаге 1 алгоритма.

Всего надо набрать $h+1$ различных B-чисел, причем при их проверке на шаге 2 надо выполнить h делений. Таким образом, среднюю временную арифметическую сложность алгоритма Диксона можно оценить выражением

$$T = O_A(u^u \cdot h^2 + h^3),$$

где $O_A(h^3)$ — сложность решения системы из $h+1$ линейных уравнений от h неизвестных. В нашем случае $h = \pi(M) = \frac{M}{\ln M}$.

Выберем в качестве M величину $M = L(n)^{1/2}$, где через $L(n)$ обозначено $\exp(\sqrt{\ln n \cdot \ln\ln n})$. Тогда, как нетрудно проверить,

$$u^u = L(n)^{1/2},$$
$$M = \pi(M) = L(n)^{1/2}.$$

Отсюда следует, что трудоемкость алгоритма Диксона оценивается следующим образом:

$$T = O_A(L(n)^2 + L(n)^{3/2}) = O(L(n)^2).$$

Алгоритм Диксона может быть усовершенствован по нескольким направлениям:

— можно заменить процедуру генерации B-чисел так, чтобы вероятность их порождения была большей;

— можно оптимизировать выбор факторной базы с тем, чтобы уменьшить число неизвестных в системе уравнений;

— можно усовершенствовать процедуру отсева «плохих» чисел b, не являющихся B-числами;

— можно использовать быстрый алгоритм решения системы линейных уравнений (например, алгоритм Видемана для разреженных матриц), и т. д.

Подобные улучшения не позволяют изменить общий вид оценки сложности $O(L(n)^c)$, но дают возможность уменьшить константу c, $c \geqslant 1$.

13.5. Алгоритм Брилхарта—Моррисона

Суть алгоритма Брилхарта—Моррисона, опубликованного ими в 1975 г., заключается в двух изменениях в алгоритме Диксона:

а) случайные числа b выбираются с помощью метода непрерывных дробей, предложенного Лагранжем, позволяющего генерировать

такие b, что число $b^2 \bmod n$ является малым, и поэтому вероятность его разложения более высока;

б) из факторной базы B исключаются те числа p, для которых

$$\left(\frac{n}{p}\right) = -1.$$

Рассмотрим его более подробно.

Теорема. *Пусть $x > 1$ — действительное число, и $\left\{\frac{P_k}{Q_k}\right\}$ — последовательность его подходящих дробей, $k = 1, 2, \ldots$ Тогда при всех $k \geqslant 1$ выполняется неравенство*

$$|P_k^2 - x^2 Q_k^2| < 2x.$$

Д о к а з а т е л ь с т в о. В силу свойств подходящих дробей имеем

$$|P_k^2 - x^2 Q_k^2| = Q_k^2 \cdot \left|x - \frac{P_k}{Q_k}\right| \cdot \left|x + \frac{P_k}{Q_k}\right| < Q_k^2 \cdot \frac{1}{Q_k Q_{k+1}} \cdot \left(2x + \frac{1}{Q_k Q_{k+1}}\right).$$

Отсюда

$$|P_k^2 - x^2 Q_k^2| - 2x < 2x \cdot \left(-1 + \frac{Q_k}{Q_{k+1}} + \frac{1}{2x Q_{k+1}^2}\right) <$$
$$< 2x \cdot \left(-1 + \frac{Q_k + 1}{Q_{k+1}}\right) < 2x \cdot \left(-1 + \frac{Q_{k+1}}{Q_{k+1}}\right) = 0.$$

Следствие. *Пусть n — не является квадратом и $\frac{P_k}{Q_k}$ — подходящая дробь для числа $\sqrt{n}$. Тогда минимальный по абсолютной величине вычет $P_k^2 \bmod n$ равен $P_k^2 - nQ_k^2$ и не превосходит $2\sqrt{n}$.*

Д о к а з а т е л ь с т в о. При $x = \sqrt{n}$

$$P_k^2 \equiv (P_k^2 - nQ_k^2) \pmod n,$$

причем в силу теоремы $|P_k^2 - nQ_k^2| < 2\sqrt{n}$. Поэтому минимальный по абсолютной величине вычет числа P_k^2 по модулю n совпадает с $P_k^2 - nQ_k^2$, что и требовалось доказать.

Данное следствие показывает, что если в качестве случайных чисел b на 1 этапе алгоритма Диксона брать числа P_k, то минимальный по абсолютной величине вычет числа $P_k^2 \bmod n$, равный $P_k^2 - nQ_k^2$, не будет превышать значения $2\sqrt{n}$. Поэтому вероятность его разложения в произведение чисел из базы B будет более высокой, чем при случайном выборе.

Покажем теперь, что такой выбор чисел b позволяет исключить из факторной базы все числа p, у которых $\left(\frac{n}{p}\right) = -1$. Покажем, что в разложении числа $P_k^2 - nQ_k^2$ на простые сомножители будут встречаться только те числа p, для которых n является квадратичным вычетом по модулю p.

Пусть $p \mid (P_k^2 - nQ_k^2)$. Так как $(P_k, Q_k) = 1$, то $(p, Q_k) = 1$. Отсюда вытекает, что элемент Q_k обратим по модулю p. Поэтому должно выполняться условие

$$\left(P_k Q_k^{-1}\right)^2 \equiv n \pmod{p},$$

но это и означает, что $\left(\frac{n}{p}\right) = 1$.

Такие улучшения позволяют добиться общей оценки сложности алгоритма $O(L(n)^c)$ при $c = \sqrt{2}$.

13.6. Метод квадратичного решета

В 1981 г. К. Померанц предложил для порождения B-чисел использовать подход, обобщающий метод Ферма.

Рассмотрим функцию

$$f(x) = (x + \lfloor\sqrt{n}\rfloor)^2 - n,$$

являющуюся квадратным трехчленом с целыми коэффициентами. При каждом целом $x \in \mathbb{Z}$ получается нетривиальное сравнение

$$(x + \lfloor\sqrt{n}\rfloor)^2 \equiv f(x) \pmod{n},$$

причем значение $f(x)$ при малых x невелико:

$$f(x) \leqslant x^2 + 2x\sqrt{n},$$

и поэтому $(x + \lfloor\sqrt{n}\rfloor)^2 \neq f(x)$. Поэтому, если при вместо случайных чисел b в алгоритме рассматривать значения $b = f(x)$ при случайных x, заключенных в некотором интервале $-M \leqslant x \leqslant M$, то вероятность порождения B-чисел возрастает.

При этом можно предварительно осуществить просеивание тех x из этого интервала, которые могут давать разложение $f(x)$ на множители из факторной базы B. Для этого для каждого $p \in B$, которое может выступать в качестве сомножителя в разложении

$$f(x) = (x + \lfloor\sqrt{n}\rfloor)^2 - n = \prod_{p \in B} p^{\alpha_p(x)},$$

решаем квадратное уравнение

$$(x + \lfloor\sqrt{n}\rfloor)^2 - n \equiv 0 \pmod{p}$$

в кольце $\mathbb{Z}_p$. Если $r_1^{(p)}, r_2^{(p)}$ — решения этого уравнения, то должно выполняться равенство $x = r_i^{(p)} + jp$, $i = 1, 2$, $j \in \mathbb{Z}$.

Проделав предварительно данную работу по нахождению корней $r_1^{(p)}, r_2^{(p)}$ для каждого $p \in B$, можно теперь оставить в интервале $[-M, M]$ только те числа x, для которых сравнение $x \equiv r_i^{(p)} \pmod{p}$ выполняется для достаточно большого количества чисел $p \in B$. Знание для каждого x списка тех $p \in B$, для которых выполняются эти сравнения, поможет также быстрее осуществить факторизацию числа $f(x)$, если она существует.

Дж. Дэвис и П. Монтгомери обобщили данный подход. Они предложили использовать многочлен

$$f_{ab}(x) = ax^2 + 2bx + c,$$

где коэффициенты a, b, c являются целыми числами и удовлетворяют условию

$$b^2 - ac = n, \quad 0 \leqslant b < a;$$

При таком выборе коэффициентов выполняется равенство

$$af_{ab}(x) = (ax + b)^2 - n,$$

откуда получаем сравнение

$$(ax + b)^2 \equiv af_{ab}(x) \pmod{n},$$

причем

$$(ax + b)^2 \neq af_{ab}(x).$$

Коэффициенты a, b, c выбираются, исходя из следующих соображений. Функция $f_{ab}(x)$ принимает максимальные значение в концах отрезка $[-M, M]$

$$f_{ab}(-M) \approx \frac{1}{a}(a^2M^2 - n).$$

Минимальное значение она принимает в точке $x = -b/a$, $-1 < x < 0$,

$$f_{ab}\left(-\frac{b}{a}\right) = -\frac{n}{a}.$$

Будем выбирать a, b, c так, чтобы максимальное и минимальное значения функции $f_{ab}(x)$ на отрезке $[-M, M]$ были равны по абсолютной величине и противоположны по знаку:

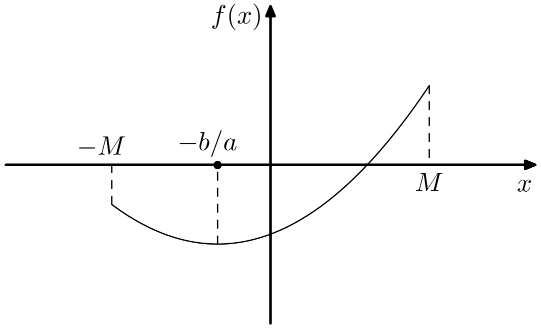

Рис. 1.

Поэтому полагаем:

$$a \approx \frac{\sqrt{2n}}{M}.$$

Число b находим как решение сравнения

$$b^2 \equiv n \pmod a, \quad 0 \leqslant b < a.$$

Наконец, число c выбираем из условия

$$b^2 - ac = n.$$

Заметим, что условие существования решения второго сравнения можно преобразовать следующим образом: для всех простых делителей $q \mid a$ выполняется $(\frac{n}{q}) = 1$. Значение факторизации числа a необходимо также потому, что оно участвует в качестве сомножителя в искомом сравнении. Поэтому, обычно, в качестве числа a выбирают либо полный квадрат, либо число, факторизуемое в произведение простых чисел из факторной базы.

При данном выборе параметров значения многочлена $f_{ab}(x)$ на интервале $[-M, M]$ удовлетворяют неравенству

$$|f_{ab}(x)| \leqslant \frac{1}{\sqrt{2}} M\sqrt{n},$$

что в $2\sqrt{2}$ раз лучше, чем у исходного алгоритма Померанца, в котором при $M \ll \sqrt{n}$

$$|f_{ab}(x)| \leqslant 2 + 2x\sqrt{n} \approx 2x\sqrt{n} \leqslant 2M\sqrt{n}.$$

Поэтому такие полиномы более предпочтительны для поиска B-чисел.

Метод Померанца называется методом квадратичного решета или qs-методом, а метод Дэвиса—Монтгомери — mpqs-методом (multiply polinomial variation of quadratic sieve algorithm).

13.7. $(p-1)$-метод факторизации Полларда

Предположим, что n — нечетное составное число, не имеющее небольших простых делителей. Обозначим через p наименьший простой делитель числа n. Наша задача заключается в его нахождении.

Предположим, что число $(p-1)$ разлагается в произведение небольших простых делителей. Выберем число k, которое является параметром метода. Для успешной работы алгоритма нужно, чтобы выполнялось условие

$$(p-1) \mid M(k),$$

где $M(k)$=НОК$(1, 2, \ldots, k)$ (вместо $M(k)$ можно использовать $k!$, либо произведение $p_1^{\alpha_1} \ldots p_2^{\alpha_2}$ первых k простых чисел в некоторых степенях $\alpha_1 \geqslant \ldots \geqslant \alpha_k$, которые выбираются из эвристических соображений). В силу малой теоремы Ферма выполняется сравнение

$$2^{M(k)} \equiv 1 \pmod p.$$

Если при этом

$$2^{M(k)} \not\equiv 1 \pmod n,$$

то

$$p \mid (2^{M(k)} - 1, n),$$

где $1 < p$, $(2^{M(k)} - 1, n) < n$. Таким образом, $d = (2^{M(k)} - 1, n)$ — является собственным делителем числа n, кратным p.

На этой идее основан следующий метод нахождения собственного делителя числа n. Так как число k неизвестно, то в нем использован последовательный перебор малых значений до некоторого фиксированного значения.

Пусть k — целое число, например $k < 10^6$, и c — небольшое целое с условием $(c, n) = 1$, например, $c = 2$.

Шаг 1. Для каждого i от 1 до k вычисляется $m_i = c^{M(i)} \mod n$ и проверяется тест шага 2.

Шаг 2. Вычислить $d = (m_i - 1, n)$. Если $1 < d < n$, то найден нетривиальный делитель числа n. В противном случае полагаем $i = i + 1$.

Можно модифицировать алгоритм, используя одновременно несколько различных оснований c.

Так как $(c,n)=1$, то $(c,p)=1$. Поэтому как только $(p-1)\,|\,M(i)$, то $m_i\equiv 1 \pmod p$, т.е. $p\,|\,(m_i-1)$, и на шаге 2 будет найден нетривиальный делитель n.

Параметр k должен выбираться не очень большим, иначе может быть выполнено условие $m_i\equiv 1 \pmod n$ и делитель не будет найден.

При выполнении алгоритма возведение в степень $c^{M(i)}$ надо осуществлять в кольце $\mathbb{Z}_n$ методом повторного возведения в квадрат. Так как $\log M(i)\leqslant i\log i$, то для вычисления $(m_i-1) \bmod n$ требуется $O(i\log i)$ арифметических операций кольца $\mathbb{Z}_n$. Поэтому общая сложность алгоритма может быть грубо оценена величиной $O(k^2\log k\log^3 n)$.

IV. Криптографическая система RSA

Криптографическая система RSA является классическим примером криптографической системы с открытыми ключами. Она была предложена в работе [50] и в настоящее время получила очень широкое распространение. Несмотря на многочисленные критические работы по изучению ее свойств, в которых указаны наборы параметров, приводящие к ослаблению схемы, при правильном использовании она считается безопасной.

Напомним общий алгоритм работы системы RSA. Каждый абонент вырабатывает свою пару открытых и секретных ключей. Для этого он генерирует два больших простых числа p и q и вычисляет произведение $n=pq$. Затем требуется взять случайное число e, взаимно простое с $\varphi(n)=(p-1)(q-1)$, и найти число d из условия $ed\equiv 1 \pmod{\varphi(n)}$. Пара (n,e) объявляется открытым ключом и помещается в открытый каталог. Остальные числа $(p,q,\varphi(n),d)$ образуют секретный ключ. Заметим, что числа $(p,q,\varphi(n))$ в дальнейшем не нужны и могут быть уничтожены. Для расшифрования достаточно знать пару (n,d).

Если абонент A хочет послать сообщение t абоненту B, то он выбирает из открытого каталога пару (n,e) абонента B и вычисляет шифрованное сообщение:

$$s\equiv t^e \pmod n.$$

Абонент B, получив данное сообщение, вычисляет:

$$t\equiv s^d \pmod n$$

§ 14. Выбор параметров системы RSA

При изучении криптографической системы RSA основным вопросом является выбор параметров p, q, e и d, при которых задача решения сравнения

$$t^e\equiv s \pmod n$$

относительно t, либо нахождения любого из параметров секретного ключа $(p,q,\varphi(n),d)$ является сложной.

14.1. Взаимосвязь между параметрами системы RSA

Покажем, что если имеется возможность найти любое из чисел $(p, q, \varphi(n), d)$, то можно найти сам секретный ключ и осуществить расшифрование. Действительно, если мы можем факторизовать число n и найти его делители p и q, то мы знаем $\varphi(n)=(p-1)(q-1)$, а параметр d легко находится из сравнения $ed \equiv 1 \pmod{\varphi(n)}$ с помощью расширенного алгоритма Евклида.

Если мы можем находить число $\varphi(n)$, то числа p и q легко находятся по формулам:

$$p+q=n-\varphi(n)+1,$$
$$p-q=\sqrt{(p+q)^2-4n}.$$

Наконец, если имеется алгоритм нахождения экспоненты d, то, как показывает следующая теорема, имеется эффективный вероятностный алгоритм факторизации.

Теорема. *Пусть $n=pq$, где p и q — простые. Если существует алгоритм нахождения числа d по открытому ключу (n, e), то существует вероятностный алгоритм факторизации числа n с вероятностью успеха больше $1/2$.*

Доказательство. Так как $ed \equiv 1 \pmod{\varphi(n)}$, то при некотором k должно выполняться равенство

$$ed-1=k\varphi(n),$$

левая часть которого известна. Пусть $ed-1=2^s r$, где r — нечетно, $s \geqslant 1$. По теореме Эйлера для каждого элемента $\mathbb{Z}_n^*$ должно выполняться сравнение:

$$a^{2^s r} \equiv 1 \pmod{n}.$$

Рассмотрим теперь вероятностный алгоритм, аналогичный тесту простоты Миллера—Рабина.

Шаг 1. Выбираем случайно k чисел $a_1, \ldots, a_k \in \mathbb{Z}_n^*$ и для каждого из них проверяем тест на шаге 2.

Шаг 2. Для данного a вычисляем a^r.
Если $a^r \equiv 1 \pmod{n}$, то тест прошел успешно.
Если $a^r \not\equiv 1 \pmod{n}$, то вычисляем последовательность

$$(a^r)^2, (a^r)^{2^2}, \ldots, (a^r)^{2^{s-1}}$$

и проверяем, встречается ли в ней $-1 \pmod{n}$. Если да, то тест прошел успешно. Если нет, то в результате будет найдено такое s', что

$$b=a^{2^{s'-1}r} \not\equiv \pm 1 \pmod{n}, \quad b^2=a^{2^{s'}r} \equiv 1 \pmod{n}.$$

Шаг 3. Для найденного на шаге 2 числа b вычисляем $(b-1, n)$ и $(b+1, n)$. Они и будут искомыми числами p и q.

Если для всех чисел $a_1, \ldots, a_k$ тест на шаге 2 прошел успешно, то разложение не найдено.

Покажем что данный алгоритм с вероятностью успеха не менее $1-2^{-k}$ позволяет найти факторизацию числа n.

Пусть $\mathbb{Z}_n^* = A_n \cup B_n$, где

$$A_n=\left\{a \in \mathbb{Z}_n^* \colon \exists s' \leqslant s, a^{2^{s'-1}r} \not\equiv \pm 1 \pmod{n}, a^{2^{s'}r} \equiv 1 \pmod{n}\right\},$$
$$B_n=\mathbb{Z}_n^* \backslash A_n.$$

Так как элементы $a \in A_n$ позволяют найти факторизацию n, то для доказательства теоремы надо доказать, что $|B_n| < \frac{|\varphi(n)|}{2}$.

Представим множество B_n в виде

$$B_n=B_n^0 \cup \bigcup_{s'=1}^{s} B_n^{s'},$$

где

$$B_n^0=\{a \in \mathbb{Z}_n \colon a^r \equiv 1 \pmod{n}\},$$
$$B_n^{s'}=\left\{a \in \mathbb{Z}_n \colon a^{2^{s'-1}r} \equiv -1 \pmod{n}, a^{2^{s'}r} \equiv 1 \pmod{n}\right\},$$

Пусть $p-1=2^i m$, $q-1=2^j l$, где m, l — нечетны и $i \leqslant j$. Сначала заметим, что множество $B_n^{s'}$ не пусто только при $s' \leqslant i$, Действительно,

$$\varphi(n)=2^i m 2^j l,$$

причем $2^i m 2^j l \,|\, (ed-1)=2^s r$. Значит, $ml \,|\, r$. Поэтому условие

$$a^{2^{s'-1}r} \equiv -1 \pmod{n},$$

а, следовательно, и

$$a^{2^{s'-1}r} \equiv -1 \pmod{p}$$

может выполняться только, если $s'-1<i$, т.е. $s' \leqslant i$.

Итак,

$$|B_n|=|B_n^0|+\sum_{s'=1}^{i}|B_n^{s'}|.$$

Учитывая изоморфизм $\mathbb{Z}_n \cong \mathbb{Z}_p + \mathbb{Z}_q$, получаем

$$|B_n^0| = |B_p^0| \cdot |B_q^0| = (r, p-1)(r, q-1) = ml,$$

так как порядок подгруппы

$$\{a \in \mathbb{Z}_p \colon a^r \equiv 1 \pmod{p}\}$$

группы $\mathbb{Z}_p$ равен $(r, p-1) = m$.

Рассуждая аналогично, получаем

$$|B_n^{s'}| = |B_p^{s'}||B_q^{s'}| = ((2^{s'}r, p-1) - (2^{s'-1}r, p-1))((2^{s'}r, q-1) - \\ - (2^{s'-1}r, q-1)) = (2^{s'}m - 2^{s'-1}m)(2^{s'}l - 2^{s'-1}l) = 2^{2(s'-1)}ml.$$

Отсюда

$$|B_n| = ml + \sum_{k=0}^{i-1} 2^{2k}ml = ml\left(1 + \frac{4^i - 1}{3}\right) = \\ = ml\,\frac{4^i + 2}{3} \leqslant ml\,\frac{2^{i+j} + 2}{3} < ml\,\frac{2^{i+j}}{2} = \frac{\varphi(n)}{2}.$$

Теорема доказана. □

В заключение заметим, что каждый пользователь должен независимо секретным образом генерировать свои собственные числа p, q. И хотя процесс генерации простых чисел p и q достаточно трудоемкий, нельзя делать число $n = pq$ общим хотя бы для двух разных пользователей системы RSA.

Теорема. *Если абонент A использует те же числа p и q, что и абонент B, то они оба могут без факторизации найти секретные экспоненты друг друга с помощью детерминированного алгоритма.*

Доказательство. Покажем, как пользователь B может найти секретную экспоненту d_A абонента A. По своим экспонентам e_B и d_B он может вычислить величину $e_B d_B - 1 = k\varphi(n)$, где k и $\varphi(n)$ — неизвестные ему значения.

Обозначим через t минимальный делитель числа $e_B d_B - 1$ такой, что число $(e_B d_B - 1)/t$ взаимно просто с e_A.

Покажем что существует детерминированный алгоритм его вычисления.

Шаг 1. Полагаем $q_0 = e_B d_B - 1$, $h_0 = (g^0, e_A)$, $t = h_0$.

Шаг 2. Для всех i от 1 до тех пор, пока $h_i > 1$, полагаем

$$g_i = g_{i-1} g_{i-1}/h_{i-1}, \quad h_i = (g_i, e_A), \quad t = th_i.$$

Полученное в результате выполнения шага 2 число t, очевидно, и будет искомым.

Легко видеть, что число $t = h_0 h_1 \ldots h_i$, будет удовлетворять условию

$$\left(\frac{e_B d_B - 1}{t}, e_A\right) = 1,$$

причем алгоритм выполнит не более $2\log n$ шагов, так как если $h_i \geqslant 2$, то

$$i \leqslant \log_2(e_B d_B - 1) \leqslant 2\log_2 n.$$

Воспользуемся теперь расширенным алгоритм Евклида для нахождения чисел a и b из условия

$$a\,\frac{e_B d_B - 1}{t} + b e_A = 1.$$

Далее, покажем, что $b \equiv d_A \pmod{n}$ — искомое число. Так как $t = h_0 h_1 \times \ldots \times h_i$, где $h_i \,|\, e_A$, а $(e_A, \varphi(n)) = 1$, то $(t, \varphi(n)) = 1$. Отсюда

$$\varphi(n) \,\Big|\, \frac{e_B d_B - 1}{t}.$$

Теорема доказана. □

В заключение сделаем еще одно замечание, относящееся к данному случаю. Предположим, что мы просмотрели список открытых ключей всех абонентов и нашли пару ключей с условием $(e_A, e_B) = 1$. Тогда для чисел x, y таких, что $xe_A + ye_B = 1$, будет верно простое соотношение, позволяющее осуществить бесключевое чтение сообщения t, зашифрованного на этих ключах. А именно, если $s_1 \equiv t^{e_A} \pmod{n}$, $s_2 \equiv t^{e_B} \pmod{n}$, то $s_1^x s_2^y \equiv t \pmod{n}$.

14.2. Условия на выбор чисел p и q

Поскольку успешное решение задачи факторизации числа n позволяет полностью дешифровать схему RSA и определить секретный ключ, то выбор простых чисел p и q во многом определяет стойкость шифрования.

Перечислим некоторые требования на выбор p и q, невыполнение которых может приводить к снижению стойкости шифрования.

Во-первых, числа не должны содержаться в списках известных больших простых чисел. Поэтому они не должны являться числами специального вида (например, числами Мерсенна или Ферма) и не должны иметь закономерностей.

Во-вторых, они не должны быть близкими, так как иначе можно воспользоваться методом факторизации Ферма и решить уравнение

$$\left(\frac{p+q}{2}\right)^2 - n = \left(\frac{p-q}{2}\right)^2.$$

В-третьих, огромную роль играют разложения на простые множители чисел $p-1$ и $q-1$. Заметим, что в силу изоморфизма $\mathbb{Z}_n \cong \mathbb{Z}_p + \mathbb{Z}_q$ для порядков элементов $a = a_1 + a_2$ этого кольца выполняется соотношение

$$\mathrm{ord}_n(a) = \text{НОК}(\mathrm{ord}_p(a_1), \mathrm{ord}_q(a_2)).$$

Поэтому для нахождения экспоненты d достаточно решить сравнение

$$ed \equiv 1 \pmod{(p-1, q-1)}$$

вместо сравнения

$$ed \equiv 1 \pmod{\varphi(n)}.$$

Так как числа $p-1$ и $q-1$ оба четны, то всегда

$$\text{НОК}\,(p-1, q-1) \,\Big|\, \frac{\varphi(n)}{2}.$$

Следовательно, в схеме RSA всегда есть эквивалентные по расшифрованию экспоненты, например, d и $d + \text{НОК}(p-1, q-1)$. При этом эквивалентных решений будет тем больше, чем больше $\text{НОД}(p-1, q-1)$.

В лучшем случае $(p-1, q-1) = 2$ и $p = 2t+1$, $q = 2s+1$, где s, t — нечетные числа с условием $(s, t) = 1$.

Чтобы исключить возможность применения $(p-1)$- и $(p+1)$-методов факторизации на числа p и q накладывает ограничение, состоящее в том, чтобы числа $p-1$, $p+1$, $q-1$, $q+1$ не разлагались в произведение маленьких простых сомножителей и содержали бы в качестве сомножителя хотя бы одно большое простое число. Наиболее сильное требование, сформулированное Ривестом в 1978 г., заключается в том, чтобы числа

$$p_1 = \frac{p-1}{2}, \quad p_2 = \frac{p+1}{2}, \quad q_1 = \frac{q-1}{2}, \quad q_2 = \frac{q+1}{2}$$

были простыми, причем числа $p_1 - 1$ и $q_1 - 1$ также не должны разлагаться в произведение маленьких простых чисел, а должны содержать в качестве одного из делителей большое простое число.

Определение. *Простое число p называется сильно простым, если выполняются условия*:

$$\begin{aligned} p &\equiv 1 \pmod r,\\ p &\equiv s-1 \pmod s,\\ r &\equiv 1 \pmod t, \end{aligned}$$

где p, r, s, t — большие простые числа.

Учитывая, что числа p, q, r и t должны быть нечетными, получаем при некоторых j, k, l выполнены равенства:

$$\begin{aligned} p &= 2jr+1,\\ p &= 2ks-1,\\ r &= 2lt+1. \end{aligned}$$

Дж. Гордон предложил в 1984 г. алгоритм генерации таких чисел.

Шаг 1. Строим простые случайные числа s и t. Для построения каждого из них используем следующий метод. Фиксируем случайное число x, имеющее нужное число знаков, и с помощью пробных делений оставляем в интервале $[x, x + \log x]$ только те числа, которые не имеют маленьких делителей. Проверяем оставшиеся числа на простоту одним из тестов простоты.

Шаг 2. Строим простое число r вида $2lt+1$ перебором чисел l в интервале $[1, \log t]$, используя аналогично шагу 1 просеивание с помощью пробных делений на маленькие простые числа и применяя затем к оставшимся числам тесты простоты.

Шаг 3. Вычисляем число $u(r,s) = (s^{r-1} - r^{s-1}) \mod rs$ и полагаем

$$p_0 = \begin{cases} u(r,s), & u(r,s) = 2g+1,\\ u(r,s) + rs, & u(r,s) = 2g, g \in \mathbb{Z}. \end{cases}$$

Перебором чисел k осуществляем проверку чисел вида $p_0 + 2krs$ до тех пор, пока не сработает один из тестов простоты.

В основе данного алгоритма лежит следующая

Теорема (Гордон). *Если r, s — нечетные простые числа, то число p удовлетворяет условиям*

$$\begin{aligned} p &\equiv 1 \pmod r,\\ p &\equiv s-1 \pmod s \end{aligned}$$

в том и только в том случае, когда оно имеет вид $p = p_0 + 2krs$, *где*

$$p_0 = \begin{cases} u(r,s), & u(r,s) = 2g+1, \\ u(r,s) + rs, & u(r,s) = 2g, g \in \mathbb{Z}, \end{cases}$$
$$u(r,s) = (s^{r-1} + r^{s-1}) \mod rs.$$

Доказательство. В силу малой теоремы Ферма имеем

$$s^{r-1} \equiv 1 \pmod r,$$
$$r^{s-1} \equiv 1 \pmod s.$$

Отсюда, учитывая, что $s^{r-1} \equiv 0 \pmod s$, $r^{s-1} \equiv 0 \pmod r$, получаем

$$\begin{cases} u(r,s) \equiv 1 \pmod r, \\ u(r,s) \equiv -1 \pmod s. \end{cases}$$

Поэтому числа вида $p = p_0 + 2krs$ являются искомыми.

Покажем, что других чисел, удовлетворяющих данному условию нет. Если p' — такое число, то $p - p' \equiv 0 \pmod r$ и $p - p' \equiv 0 \pmod s$. Отсюда $p \equiv p' \pmod{rs}$. Значит $p' \equiv u(r,s) \pmod{rs}$, и число p' имеет вид $p_0 + 2k'rs$. Теорема доказана. □

Еще одной возможной слабостью выбора чисел p и q в системе RSA может быть наличие большого числа элементов $\omega \in \mathbb{Z}_n^*$, представляющих открытые сообщения и имеющих малый порядок $\operatorname{ord}_n(\omega)$. В этом случае для нахождения неизвестного сообщения ω можно изменить метод итерации процесса зашифрования

$$s = \omega^e, (\omega^e)^e = \omega^{e^2}, (\omega^{e^2})^e = \omega^{e^3}, \ldots, \omega^{e^4}, \ldots$$

до тех пор, пока не выполнится условие

$$e^u \equiv 1 \pmod{\operatorname{ord}_n(\omega)},$$

и, следовательно,

$$\left(s^{e^u}\right)^{-1} = \omega^{e^u} = \omega \pmod n.$$

Минимальное число шагов до получения ω равно $\operatorname{ord}_{\operatorname{ord}_n(\omega)}(e)$. Значит для безопасности шифрования необходимо потребовать, чтобы для почти всех $x \in \mathbb{Z}_n^*$ и $e \in \mathbb{Z}_{\varphi(n)}^*$ величина $\operatorname{ord}_{\operatorname{ord}_n(\omega)}(e)$ была достаточно большой.

Приведем достаточное условие, доказанное в 1995 г. Мауэром, гарантирующее стойкость системы RSA относительно методов, использующих итерацию процесса зашифрования.

Лемма. *Пусть* $h = pq$ *и* $p-1 = 2R_pF_p$, $q-1 = 2R_qF_q$, *где каноническое разложение чисел* F_p *и* F_q *имеет вид*

$$F_p = \prod_{i=1}^{r} p_i'^{\alpha_i}, \quad F_q = \prod_{i=1}^{s} q_i'^{\beta_i}.$$

Тогда доля f *открытых сообщений* $x \in \mathbb{Z}_n^*$, *для которых* $\operatorname{ord}_n(x) \geqslant$ $\geqslant \text{НОК}(F_p, F_q)$, *удовлетворяет неравенству*

$$f \geqslant \frac{\varphi(F_p)}{F_p} \cdot \frac{\varphi(F_q)}{F_q} \geqslant 1 - \sum_{i=1}^{r} \frac{1}{p_i} - \sum_{i=1}^{s} \frac{1}{q_i}.$$

Доказательство. Воспользуемся оценкой, доказанной в лемме 3 из 12.4:

$$\left|\{x \in \mathbb{Z}_p^* \colon F_p \mid \operatorname{ord}_p(x)\}\right| \geqslant (p-1)\,\frac{\varphi(F_p)}{F_p},$$
$$\left|\{x \in \mathbb{Z}_q^* \colon F_q \mid \operatorname{ord}_q(x)\}\right| \geqslant (q-1)\,\frac{\varphi(F_q)}{F_q}.$$

Так как из условий $F_p \mid \operatorname{ord}_n(x)$, $F_q \mid \operatorname{ord}_n(x)$ непосредственно следует, что $\text{НОК}(F_p, F_q) \mid \operatorname{ord}_n(x)$, получаем

$$\left|\{x \in \mathbb{Z}_n^* \colon [F_p, F_q] \mid \operatorname{ord}_n(x)\}\right| \geqslant (p-1)(q-1)\,\frac{\varphi(F_p)}{F_p}\,\frac{\varphi(F_q)}{F_q},$$

что и доказывает лемму.

Теорема. *Пусть* $n = pq$ *из условия леммы. Пусть также*

$$p_i' - 1 = 2a_i'b_i', \quad i = 1, \ldots, r,$$
$$q_j' - 1 = 2a_j''b_j'', \quad j = 1, \ldots, s,$$

где каноническое разложение чисел b_i' *и* b_j'' *имеет вид*

$$b_i' = \prod_{j=1}^{2i} p_{ij}^{\alpha_{ij}}, \quad i = 1, \ldots, r, \qquad b_i'' = \prod_{j=1}^{2i} q_{ij}^{\beta_{ij}}, \quad j = 1, \ldots, s.$$

Тогда для любого t, взаимно простого с $(p-1)(q-1)$ и удовлетворяющего условиям:

$$t^{\frac{p'-1}{p_{ij}}} \not\equiv 1 \pmod{p_i'}, \quad i=1,\ldots,r, \quad j=1,\ldots,r_i,$$
$$t^{\frac{q'-1}{q_{ij}}} \not\equiv 1 \pmod{q_i'}, \quad i=1,\ldots,s, \quad j=1,\ldots,s_i,$$

доля сообщений $x \in \mathbb{Z}_n^*$, для которых $\mathrm{ord}_{\mathrm{ord}_n(x)}(t)$ не кратна числу НОД$(b_1',\ldots,b_r',b_1'',\ldots,b_s'')$, *не превышает* $\sum\limits_{i=1}^{r}\frac{1}{p_i'}+\sum\limits_{i=1}^{s}\frac{1}{q_i'}$.

Доказательство. Аналогично теореме Поклингтона можно показать, что $b_i' \,|\, \mathrm{ord}_{p_i'}(t)$, для $i=1,\ldots,r$. Следовательно $b_i' \,|\, \mathrm{ord}_{p_i'^{\alpha_i}}(t)$, а также $b_i' \,|\, \mathrm{ord}_{F_p}(t)$ при $i=1,\ldots,r$. Таким образом,

$$\text{НОК}(b_1',\ldots,b_r') \,|\, \mathrm{ord}_{F_p}(t),$$

Аналогично получаем

$$\text{НОК}(b_1'',\ldots,b_s'') \,|\, \mathrm{ord}_{F_q}(t),$$

Отсюда

$$\text{НОК}(b_1',\ldots,b_r',b_1'',\ldots,b_s'') \,|\, \mathrm{ord}_{\text{НОК}(F_q,F_p)}(t).$$

Согласно лемме, доля $x \in \mathbb{Z}_n^*$, для которых выполняется условие НОК$(F_p,F_q) \,|\, \mathrm{ord}_n(x)$, составляет не менее

$$1-\sum_{i=1}^{r}\frac{1}{p_i'}+\sum_{i=1}^{s}\frac{1}{q_i'}.$$

Поскольку для таких $x \in \mathbb{Z}_n^*$ должно быть

$$\mathrm{ord}_{\text{НОК}(F_p,F_q)}(t) \,|\, \mathrm{ord}_{\mathrm{ord}_n(x)}(t),$$

получаем требуемое. Теорема доказана. □

Данная теорема показывает, что условие Ривеста о том, чтобы для большого простого делителя p_1 (q_1) числа p (q) число p_1-1 (q_1-1) так же имело большой простой делитель не является необходимым для защиты от метода с использованием итерации процесса зашифрования.

Для генерации простых чисел p и q, дающих достаточную стойкость относительно этого метода, можно использовать метод Маурера построения простых чисел. Он носит рекурсивный характер и для генерации p и q используются сгенерированные на первом уровне рекурсии ранее числа p_i' и q_i', которые в свою очередь строятся по сгенерированным ранее на втором уровне рекурсии числам p_{ij}'' и q_{ij}''. Согласно теореме для обеспечения высокой стойкости относительно метода итерирования процесса зашифрования достаточно взять попарно взаимно простые числа p_{ij}'' и q_{ij}'', а в качестве экспоненты шифрования $e=t$ взять то число x, которое использовалось для доказательства простоты чисел с помощью теоремы Поклингтона p_{ij} и q_{ij} на втором уровне рекурсии.

Поэтому условия теоремы несложно обеспечиваются небольшой модификацией процедуры генерации простых чисел и не требуют больших затрат на гарантирование их выполнения.

14.3. Выбор параметров e и d

Рассмотрим теперь вопрос о выборе экспонент шифрования и расшифрования. Так как значения e и d определяют время зашифрования и расшифрования, то можно назвать ряд ситуаций, в которых желательно иметь малое значение e и d. Например, при использовании системы RSA при защите электронных платежей с применением кредитных карточек естественным является требование использования небольших значений экспоненты d у владельца карточки и большого значения экспоненты e у центрального компьютера.

Однако, выбор малых параметров e или d представляется небезопасным по ряду соображений. Если малым является секретный параметр d, то можно применить метод перебора малых значений до получения искомого числа d.

Если малым является параметр e, то достаточно большое число открытых сообщений, удовлетворяющих неравенству $t<\sqrt[e]{n}$ будут зашифровываться простым возведением в степень $t^e=s$ в кольце $\mathbb{Z}$, и поэтому их можно найти путем извлечения корня степени e.

Другая аналогичная ситуация может сложиться, когда у нескольких абонентов используется одинаковая экспонента e. Предположим, что k абонентов, использующих одинаковую экспоненту $e \leqslant k$, отправляют сообщения $s_1 \equiv t_1^e \pmod{n_1},\ldots,s_k \equiv t_k^e \pmod{n_k}$, можно считать, что практически всегда выполняется условие $(n_i,n_j)=1$ (в противном случае станет известной факторизация чисел n_i,n_j). По китайской теореме об остатках существуют числа t и s такие что $t \equiv t_i \pmod{n_i}$, $s \equiv s_i \pmod{n_i}$, $i = 1, \ldots, k$. Так как числа $s_1, \ldots, s_k$ известны, то число $0 \leqslant s < n_1 \cdot \ldots \cdot n_k$ можно вычислить. Поэтому в случае, когда набор $t_1, \ldots, t_k$ таков, и выполнено неравенство $t^e < n_1 \cdot \ldots \cdot n_k$, то число t можно получить путем извлечения корня степени e, а по нему восстановить исходные тексты $t_1, \ldots, t_k$. Данная ситуация наиболее очевидна в случае, когда всем абонентам отправлено одно и то же циркулярное сообщение t, $t < \sqrt[e]{n_1 \cdot \ldots \cdot n_k}$. В данном случае сразу получаем уравнение $t^e = s$ в кольце целых чисел, которое решается путем извлечения корня степени e.

Комментарии

При написании раздела 1 использовались книги [2], [9], [34] и [1]. Более подробную информацию по оценке сложности арифметических операций, алгоритмов Евклида, дискретному преобразованию Фурье и др. алгоритмам можно прочитать, например, в книгах [9] и [13], содержащих также большое число упражнений и задач, или в [14].

Раздел 2 написан на основе книг [10], [16]. Исторические сведения взяты из книг [3], [18].

Раздел 3 основан на книгах [10], [34]. Много дополнительных сведений можно получить в [9]. Свойства чисел Кармайкла опубликованы в [36]. Тест Соловея—Штрассена — в статье [53] с дополнением [54]. Тест Миллера—Рабина опубликован сначала в детерминированном варианте в [41], а затем в вероятностном в работе [49]. Тесты простоты для маленьких простых чисел приведены по курсу лекций К. Калдвела (C. K. Caldwell. Primality Proving. University of Tennessee, 1995), доступном в Интернет. В 1980 г. Л. Эдлеман предложил детерминированный метод проверки на простоту, основанный на теории целых алгебраических чисел. Его модификации описаны в работе Х. Ленстры [12]. Теорема Н. Диемитко [30] лежит в основе алгоритма построения ключей первого российского стандарта цифровой подписи. Методы построения простых чисел подробно описаны в [39] и [40]. Методы квадратичного решета предложен в [47] и модифицирован в [52]. Приведенная информация об истории открытия и поиска первых простых чисел Мерсенна, доступна в сети Интернет. За рамками курса лекций остался метод эллиптических кривых для факторизации чисел, подробно описанный в [10]. В отличие от методов Полларда и Полларда—Штрассена, которые ориентированы на поиск маленьких простых делителей чисел, метод эллиптических кривых позволяет находить достаточно большие делители. В настоящее время одними из наиболее действенных методов факторизации являются методы на основе решета числового поля (см. [38]). С историей развития теоретико-числовых алгоритмов можно ознакомиться, например, по обзору [37].

Раздел 4 опирается на книгу [17], а также статьи [33] и [39]. Дополнительную информацию по приложениям системы RSA можно получить, например, в книгах [55], [21] или [15].

Литература

[1] *Акритас А.* Основы компьютерной алгебры с приложениями. — М.: Мир, 1994.

[2] *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. — М.: Мир, 1979.

[3] *Бухштаб А. А.* Теория чисел. — 2-е изд. — М.: Просвещение, 1966.

[4] *Василенко О. Н.* Современные способы проверки простоты чисел // Киберн. сборник. — 1988. — Т. 25. — С. 162—188.

[5] *Виноградов И. М.* Основы теории чисел. — 9-е изд., перераб. — М.: Наука, 1981.

[6] *Галочкин А. И., Нестеренко Ю. В., Шидловский А. Б.* Введение в теорию чисел — М.: Изд-во МГУ, 1995.

[7] *Гашков С. Б., Чубариков В. Н.* Арифметика, алгоритмы, сложность вычислений. Учебное пособие. — М.: ВШ, 2000.

[8] *Дэвенпорт Дж., Сирэ И., Турнье Э.* Компьютерная алгебра. — М.: Мир, 1991.

[9] *Кнут Д.* Искусство программирования. Т. 2. Получисленные алгоритмы. — 3-е изд. — М.: Вильямс, 2000.

[10] *Коблиц Н.* Курс теории чисел и криптографии. — М.: Научное издательство ТВП, 2001.

[11] *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. — М.: МЦНМО, 1999.

[12] *Ленстра Х. У.* Алгоритмы проверки на простоту // Алгебра и теория чисел (с приложениями): Сб. статей. — М.: Мир, 1987. — Вып. 43. — С. 47—66.

[13] *Ноден П., Китте К.* Алгебраическая алгоритмика. — М.: Мир, 1999.

[14] *Ростовцев А.* Алгебраические основы криптографии. — СПб.: Мир и семья—Интерлайн, 2000.

[15] *Ростовцев А., Маховенко Е.* Введение в криптографию с открытым ключом. — СПб.: Мир и семья—Интерлайн, 2001.

[16] *Прахар К.* Распределение простых чисел. — М.: Мир, 1967.

[17] *Саломаа А.* Криптография с открытым ключом. — М.: Мир, 1996.

[18] *Сушкевич А. К.* Теория чисел. Элементарный курс. — Харьков: Изд. Харьковского ун-та, 1954.

[19] *Трост Э.* Простые числа. — М.: ГИФМЛ, 1959.

[20] *Хассе Г.* Лекции по теории чисел. — М.: ИЛ, 1953.

[21] Введение в криптографию / Под общ. ред. В. В. Ященко. — 3-е изд., доп. — М.: МЦНМО: «ЧеРо», 2000.

[22] *Adleman L. M.* Factoring numbers using singular integers // Proc. 23rd ACM Symp. on Theory of Comput., N.O., LA. — 1991. — P. 64—71.

[23] *Adleman L. M., Pomerance C., Rumely R. S.* On distinguishing prime numbers from composite numbers // Ann. Math. (2). — 1983. — V. 117, No. 1. — P. 173—206.

[24] *Baker R. C., Harman G.* The Brun—Titchmarsh Theorem on average // Proceedings of a conference in Honor of Heini Halberstam. — 1996. — Vol. 1. — P. 39—103.

[25] *Brent R. P.* Analysis of the binary Euclidean algorithm // Algorithms and Complexity: New Directions and Recent results / J. F. Traub Ed. — New York: Academic Press, 1976. — P. 321—355.

[26] *Brillhart J., Morrison M. A.* A method of factoring and the factorization of F // Math. Comp. — 1975. — V. 29. — P. 183—205.

[27] *Canfield E., Erdos P., Pomerance C.* On a problem of Oppenheim concerning Factorisatio Numerorum // Journal of Number Theory. — 1983. — V. 17. — P. 1—28.

[28] *Carmichael R. D.* On composite numbers which satisfy the Fermat congruence // American Mathematical Monthly. — 1912. — V. 19. — P. 22—27.

[29] *Cohen H., Lenstra H. W., Jr.* Primarility testing and Jacobi sums // Report 82–18 / Mathematical Institut of the University of Amsterdam. — Amsterdam, 1982.

[30] *Diemitko N.* Generating multiprecision integer with guaranted primality // Proc. of the SIFIP Int. Conf. on Comp. Sei., IFIP/Security 88, Amsterdam, 19—21 May, 1988. — P. 1—8.

[31] *Dixon J. D.* Asymptotically fast factorization of integers // Math. Comput. — 1981. — V. 36. — P. 255—260.

[32] *Fouvry E.* Theoreme de Brun—Titchmarsh; application au theoreme de Fermat. — Invent. Math. — 1985. — V. 79. — P. 383—407.

[33] *Gordon J.* Strong RSA keys // Electronics letters. — 1984. — V. 20. — P. 514—516.

[34] *Kranakis E.* Primality and Cryptography. — N.-Y., Toronto, 1985.

[35] *Lehmer D. H.* An extended theory of Lucas' functions // Ann. Math. — 1930. — Bd. 31. — S. 419—448. Reprinted in: Selected Papers. Vol. 1. / D. McCarthy, Ed. — Ch. Babbage Res. Center, St. Pierre, Manitoba Canada, 1981. — P.11—48.

[36] *Lehmer D. H.* Strong Carmichael numbers // J. Austral. Math. Soc. Ser. A. — 1976. — V. 21, №. 4, P. 508—510.

[37] *Lenstra A. K.*, *Lenstra H. W, Jr.* Algorithms in number theory // Handbook of theoretical computer science. Vol. A. Algorithms and complexity, Ch. 12. — Amsterdam: Elsevier, 1990. — P. 674—715.

[38] The development of the number field sieve / Lenstra A. K., Lenstra H. W, Jr., Eds. — Berlin: Springer-Verlag, 1993.

[39] *Maurer U. M.* Fast generation of prime Numbers and Secure Public-key Cryptographic parameters // J. Cryptology. — 1995. — V. 8. — P. 123—155.

[40] *Mihalescu P.* Fast generation of provable primes using search in arithmetic progressions // Advances in cryptology — CRYPTO'94 (LNCS'839). — 1994. — P. 282—293.

[41] *Miller G. L.* Riemann's hypothesis and tests for primality // J. Comput. System Sci. — 1976. — V. 13. — P. 300—317.

[42] *Montgomery P. L.* Modular Multiplication Without Trail Division // Math. of Comp. — 1985. — V. 44, No. 170. — P. 519—521.

[43] *Poclington H. C.* The detemination of the prime or composite nature of ladge numbers by Fermat's theorem // Proc. of the Cambridge Society. — 1914—1916. — V. 18. — P. 29—30.

[44] *Pollard J.* Theorems on factorization and primality testing // Proc. Cambridge Phil. Soc. — 1974. — V. 76. — P. 521—528.

[45] *Pollard J.* Monte-Carlo method for factorization // BIT. — 1974. — V. 15. — P. 331—334.

[46] *Pollard J.* Monte-Carlo method for index computation (mod p) // Mathematics of Computation. — 1978. — V. 32. — P. 918—924.

[47] *Pomerance C.* The quadratic sieve factoring algorithm // Advances in cryptology — EUROCRIPT'84 (LNCS'209). — 1985. — P. 169—183.

[48] *Pomerance C.* Fast and rigorous factorization and discrete logarithm algorithm // Discrete algorithms and complexity: Proc. of the Japan—US joint seminar. — London: Acad. Press, 1987. — P. 119—143.

[49] *Rabin M.* Probabilistic algorithms for testing primality // Journal of Number Theory. — 1980. — V. 12. — P. 128—138.

[50] *Rivest R. L.*, *Shamir A.*, *Adleman L.* A method for obtaining digital signatures and public-key cryptosystems // Commun. ACM. — 1978. — V. 21, No. 2. — P. 120—126.

[51] *Shallit J.*, *Sorenson J.* Analysis of a left-shift binary GCD algorithm // J. Symbolic Comput. — 1995. — P. 169—183.

[52] *Silverman R. D.* The multiple polinomial quadratic sieve // Math. Comput. — 1987. — V. 48, No. 177. — P. 329—339.

[53] *Solovay R.*, *Strassen V.* A Fast Monte-Carlo Test for Primality // SIAM J. Comput. — 1977. — V. 6 (1). — P. 84—85.

[54] *Solovay R.*, *Strassen V.* Erratum: A Fast Monte-Carlo Test for Primality // SIAM J. Comput. — 1978. V. 7 (1). — P. 118.

[55] *Stinson D. R.* Cryptography. Theory and practice — The CRC Press Series on Discrete Math. and Appl., 1995.

[56] *Western A.*, *Miller J.* Tables of indices and primitive roots // Royal Society Mathematical Tables. — 1968. — V. 9.

[57] *Williams H. C.* A modification of the RSA public-key cryptosystem // IEEE Trans. Inform. Theory. — 1980. — V. 26, No. 6. — P. 726—729.

[58] *Williams H. C.* A numerical investigation into the length of the period of the continued fraction expansion of $\sqrt{D}$) // Math. Comp. — 1981. — V. 36. — P. 593—601.

[59] *Wiedemann D.* Solving sparse linear equations over finite field // IEEE Trans. Inform. Theory. — 1986. — V. 32, No. 1. — P. 54—62.