КРИПТОГРАФИЧЕСКИЕ КОНСТРУКЦИИ

Ю.В. Нестеренко

конспект лекций годового спецкурса для школьников СУНЦ МГУ
2018-2019 учебный год

Введение

Специальный курс **Криптографические конструкции** подготовлен для инженерного класса СУНЦ МГУ, но лекции могут посещать все, кому интересна не только теоретическая математика, но и её далеко не всегда очевидные приложения. Он будет читаться в течение учебного года, один раз в неделю. Основное содержание курса составляют так называемые информационные технологии, возникшие сравнительно недавно, в связи с образовавшимися потребностями, и переживающие сейчас период становления, они широко обсуждаются, некоторые уже используются, а другие в ближайшее время получат широкое распространение. Инженерная деятельность в этой области направлена на создание информации и механизмов её обработки, безопасного хранения и передачи, а в роли инструментов выступают математические алгоритмы, основанные на методах теории чисел, алгебры, дискретной математики. Так что сначала мы освоимся с такими понятиями, как вычислительная сложность алгоритма, познакомимся с конкретными алгебраическими и теоретико-числовыми алгоритмами, имеющими различную сложность, узнаем, что такое хэш-функция и эллиптическая кривая, состоящая из конечного числа точек, научимся строить очень большие простые числа, вычислять дискретные логарифмы и делать многое другое. Всё это составляет основу криптографической науки - важной части информационных технологий. Опишем программу этого лекционного курса.

Часть I. Алгоритмы и сложность.

Алгоритм Евклида, теорема Ламе о сложности этого алгоритма. Быстрое умножение целых чисел. Быстрый алгоритм возведения в степень. Вероятностные алгоритмы. Быстрый вероятностный алгоритм вычисления квадратичного невычета. Псевдослучайные последовательности чисел. Быстрый вероятностный алгоритм решения полиномиальных сравнений по большому простому модулю. Быстрый алгоритм построения первообразных корней по простому модулю. Алгоритмы отсеивания составных чисел. Построение больших простых чисел. Задача факторизации целых чисел. Задача дискретного логарифмирования.

Часть II. Информационные технологии.

Шифрование и расшифрование (немного истории). Ключевой обмен. Алго-

ритм RSA шифрования и расшифрования информации. Электронная цифровая подпись. Эллиптические кривые над полем вычетов по простому модулю p. Цифровая подпись с помощью эллиптических кривых. Простейшие булевы функции. Хэш-функции. Схемы обязательств. Разделение секрета. Электронное голосование. Технологии блокчейн (blockchain). Криптовалюты.

Творческая работа

Слушателям спецкурса будут предложены две самостоятельные творческие работы, каждая для небольшой группы школьников.

1. *Построение больших простых чисел.* Нужно будет построить алгоритм и соответствующую компьютерную программу, производящие простые числа на заданном интервале. Например, от 3^{500} до 4^{500} или от 3^{5000} до 4^{5000}. Для вычислений планируется использовать суперкомпьютер НИВЦ МГУ.

2. *Электронное голосование.* Построить систему удалённого электронного голосования для определения рейтинга преподавателей СУНЦ МГУ. Вопрос в том, как организовать систему голосования, для того, чтобы все голоса были переданы по обычным каналам сети Интернет и зафиксированы в центральном компьютере с соблюдением указанных в задании требований. Важную роль в этом процессе играют средства криптографии.

Кроме того, будет предложен ряд индивидуальных проектов. Например,

3. *Решение показательных сравнений.* Разработать быстрый алгоритм для решения сравнений вида $b^x \equiv a \pmod{10^n}$, где a, b натуральные числа, взаимно простые с 10, а n - достаточно большое натуральное число. Подготовить соответствующую компьютерную программу, сосчитать несколько примеров, скажем при $n > 1000$.

Криптография, как область научной деятельности, стала широко известной примерно 40 лет назад, см. [4]. Это было вызвано необходимостью обмена большими массивами конфиденциальной информации (не только государственной и военной, но и банковской, экономической, медицинской, юридической и т.п.), а также возможностью такого обмена, в связи с появлением доступных и эффективных компьютерных средств обработки этой информации. Любая информация может быть закодирована последовательностью чисел. Например, букве "а"можно сопоставить число 1, букве "б" — число 2 и так далее, букве "я"— число 32. Можно сопоставить числа пробелам, точке, другим знакам препинания. После этого процессы зашифрования и расшифрования информации представятся как некоторые алгоритмы, перерабатывающие одни массивы целых чисел в другие.

Стойкость криптографических алгоритмов напрямую зависит от невозможности найти быстрые алгоритмы для решения некоторых задач, другими словами, от того, что некоторые математические задачи сложны в вычислительном отношении.

Сложность алгоритмов теории чисел обычно принято измерять количеством арифметических операций (сложений, вычитаний, умножений и делений с остатком), необходимых для выполнения всех действий, предписанных алгоритмом. Впрочем, это определение не учитывает величины чисел, участвующих в вычислениях. Ясно, что перемножить два стозначных числа значительно сложнее, чем два однозначных, хотя и в том, и в другом случае выполняется лишь одна арифметическая операция. Поэтому иногда учитывают еще и величину чисел, сводя дело к так называемым битовым операциям, т.е. оценивая количество необходимых операций с цифрами 0 и 1 в двоичной записи чисел. Это зависит от рассматриваемой задачи, от целей автора и т.д.

На первый взгляд кажется странным, что операции умножения и деления приравниваются по сложности к операциям сложения и вычитания. Житейский опыт подсказывает, что умножать числа значительно сложнее, чем складывать их. В действительности же, вычисления можно организовать так, что на умножение или деление больших чисел понадобится не намного больше битовых операций, чем на сложение. Примерно таким же количеством битовых операций можно обойтись при выполнении деления с остатком двух двоичных чисел. Говоря далее о сложности алгоритмов, мы будем иметь в виду количество арифметических операций, необходимых для их выполнения.

Сложной, например, является следующая фундаментальная задача.

Пример. *Дано простое число p. Для заданных чисел $a, b \in \mathbb{Z}$ требуется решить сравнение*

$$a^x \equiv b \pmod{p}. \tag{1}$$

Следующая задача также важна в криптографических приложениях и является сложной.

Пример. *Дано составное натуральное число N. Требуется разложить его на нетривиальные множители.*

Еще Ферма предложил алгоритм разложения чисел на множители. Различные видоизменения его были предложены Эйлером, Гауссом, Лежандром и другими классиками теории чисел. Современные алгоритмы используют вычисления в полях алгебраических чисел, эллиптические кривые, разнообразные математические конструкции.

Наконец, рассмотрим одну "житейскую" историю. Её герои - Алиса и Боб. Живут они в разных городах и достаточно далеко друг от друга. Алиса хотела бы отправить письмо Бобу, но так, чтобы никто не смог узнать его содержимого. Для этого она берёт крепкую коробку, кладёт в неё письмо, запирает коробку на замок и отправляет её по почте Бобу. Получив запертую коробку, Боб некоторое время раздумывает, ни один из его ключей не подходит к замку Алисы. Затем запирает

коробку своими замком и ключом и отправляет теперь уже дважды запертую разными ключами коробку назад Алисе. Получив коробку, Алиса снимает свой замок, и отправляет её, всё ещё запертую, назад Бобу. Получив коробку, Боб отпирает свой замок своим ключом и получает возможность прочитать секретное письмо.

Попробуем переписать эту историю без коробок, замков и почтовой службы, но с Интернетом и целыми числами. Допустим, что, закодированное каким-либо общеизвестным способом письмо Алисы, превращается в число 111. Алиса запирает его на "замок", вычисляя $111^7 \equiv 871 \pmod{1000}$, и отправляет его Бобу. В свою очередь Боб запирает полученное сообщение, вычисляя $871^3 \equiv 311 \pmod{1000}$, и отправляет его Алисе. Алиса снимает свой замок, сосчитав $311^{43} \equiv 631 \pmod{1000}$, а Боб, получив сообщение 631, отпирает свой замок $631^{67} \equiv 111 \pmod{1000}$. Боб получил незашифрованное письмо Алисы.

Отметим, что каждый из наших героев имеет два ключа. Алиса запирает свой замок возводя сообщение в степень 7, а отпирает, возводя в степень 43. Точно так же Боб запирает замок, возводя в степень 3 и отпирает, возводя в степень 67. Конечно, 111 не единственное число, с которым действует указанная процедура. Можно вместо 111 взять 739, 827 и многие другие числа, удовлетворяющие некоторому условию. Безопасность системы основана на том, что сравнения $x^3 \equiv 311 \pmod{1000}, x^{43} \equiv 631 \pmod{1000}, x^7 \equiv 871 \pmod{1000}$ трудно разрешимы.

Глава 1

Алгоритмы и сложность

Лекция 1.

Числа

$$1, 2, 3, \ldots, 100, 101, \ldots$$

называются натуральными. Для обозначения множества натуральных чисел используется символ $\mathbb{N}$. Если к ним добавить отрицательные числа и ноль, получится множество целых чисел. Оно обозначается символом $\mathbb{Z}$. Рассматривая отношения целых чисел с ненулевыми знаменателями, можно определить множество рациональных чисел $\mathbb{Q}$. В свою очередь рациональные числа составляют подмножество совокупности действительных чисел $\mathbb{R}$.

1.1 Делимость целых чисел. Алгоритм Евклида.

Для любых целого числа a и натурального b существуют единственным образом определенные целые числа q, r, удовлетворяющие условиям

$$a = bq + r, \qquad 0 \leqslant r < b.$$

Числа r и q называются, соответственно, *остатком от деления* числа a на b и *неполным частным* при делении a на b. В случае $r = 0$ слово "неполное" в названии q опускают и говорят, что a делится на b.

Множество всех делителей целого отличного от нуля числа a конечно.

Общим делителем целых чисел $a_1, a_2, \ldots, a_n$ называется любое целое d с условием $d|a_1, d|a_2, \ldots, d|a_n$. Если среди чисел $a_1, a_2, \ldots, a_n$ есть не равное нулю, то множество общих делителей этих чисел конечно. Оно всегда содержит ± 1 и потому не пусто. Говоря в дальнейшем об общих делителях мы, даже если это не

оговаривается особо, всегда будем подразумевать, что в соответствующем наборе $a_1, a_2, \ldots, a_n$ содержится хотя бы одно не равное нулю число.

Определение 1.1. *Наибольшим общим делителем совокупности целых чисел называется самый большой из общих делителей этих чисел.*

Целые числа называются взаимно простыми, если их наибольший общий делитель равен 1.

Наибольший общий делитель чисел $a_1, a_2, \ldots, a_n$ обозначается $(a_1, \ldots, a_n)$.

Рассмотрим следующую задачу. Пусть a, b — натуральные числа. Требуется найти (a, b) — наибольший общий делитель a, b. Задача эта решается достаточно быстро с помощью хорошо известного алгоритма Евклида без разложения чисел на множители. В основе его лежит равенство $(a, b) = (b, r)$, где r — остаток от деления числа a на b.

Алгоритм 1.1. Даны: *Натуральные числа* a *и* b; $b < a$.
Найти: *Наибольший общий делитель* (a, b).

1. Вычислить r *— остаток от деления* a *на* b, *т.е. найти целое* r, *удовлетворяющее условиям* $a = bq + r,\ 0 \leqslant r < b$.
2. Если $r = 0$, *то* $(a, b) = b$, *СТОП.*
3. Если $r \neq 0$, *то заменить пару* $\{a, b\}$ *парой* $\{b, r\}$ *и перейти в пункт 1 алгоритма.*

Пусть r — остаток от деления числа a на b, т.е. $a = bq + r$, $0 \leqslant r < b$. По свойствам делимости каждый общий делитель чисел b и r делит число $bq + r = a$ и, значит, принадлежит множеству общих делителей чисел b и a. Точно так же, каждый общий делитель чисел a и b делит число $a - bq = r$, так что принадлежит множеству общих делителей чисел b и r. Отсюда следует совпадение наибольших общих делителей пар чисел a, b и b, r, т.е. равенство

$$(a, b) = (b, r). \tag{1.1}$$

Это равенство позволяет при нахождении наибольшего общего делителя заменять пару чисел a, b другой парой b, r. Заметим, что $r < b$, т.е. одно из двух чисел, участвующих в алгоритме уменьшилось. Повторяя несколько раз деление с остатком и заменяя каждый раз пару целых чисел новой мы будем каждый раз уменьшать одно из двух чисел (большее), участвующих в работе алгоритма. Ясно, что в какой-то момент одно из чисел станет равным 0 и наибольший общий делитель будет равен второму из чисел.

Рассмотрим алгоритм немного подробнее. Положим $r_0 = a$, $r_1 = b$ и обозначим через $r_2, \ldots, r_n$ — последующие делители в алгоритме Евклида. Тогда получаются

следующие равенства

$$\begin{aligned}
a =& r_0 = bq_1 + r_2, && 0 \leqslant r_2 < b,\\
b =& r_1 = r_2q_2 + r_3, && 0 \leqslant r_3 < r_2,\\
& r_2 = r_3q_3 + r_4, && 0 \leqslant r_4 < r_3,\\
& \ldots\ldots\ldots\ldots && \ldots\ldots\ldots\ldots\\
& r_{n-2} = r_{n-1}q_{n-1} + r_n, && 0 \leqslant r_n < r_{n-1},\\
& r_{n-1} = r_nq_n.
\end{aligned} \tag{1.2}$$

Алгоритм останавливается, когда деление произойдет без остатка. В приведенном выше тексте последний остаток $r_{n+1} = 0$. В соответствии с равенством (1.1) находим

$$(a, b) = (b, r_2) = (r_2, r_3) = (r_3, r_4) = \cdots = (r_{n-1}, r_n) = (r_n, 0) = r_n.$$

Таким образом, наибольший общий делитель равен последнему делителю (он же последний ненулевой остаток) в алгоритме Евклида.

Пример. *Найти наибольший общий делитель чисел* 3009 *и* 894.

Пользуясь алгоритмом Евклида, находим

$$\begin{aligned}
3009 &= 894 \cdot 3 + 327, & 894 &= 327 \cdot 3 + 240,\\
327 &= 240 \cdot 1 + 87, & 240 &= 87 \cdot 2 + 66,\\
87 &= 66 \cdot 1 + 21, & 66 &= 21 \cdot 21 + 3,\\
21 &= 3 \cdot 7.
\end{aligned}$$

Последний ненулевой остаток равен 3, поэтому $(3009, 894) = 3$.

На следующей лекции мы обсудим оценку сложности алгоритма Евклида.

Конец первой лекции.

Лекция 2.

На первой лекции рассматривался вопрос о том, как Алиса может переслать секретное сообщение Бобу. При этом Алиса использовала ключи 7 и 43, а у Боба были свои ключи 3 и 67. Напомним, что Алиса пересылала число 111. Сначала она зашифровала его ключом 7, вычисляя $111^7 \equiv 871 \pmod{1000}$, и отправила результат Бобу. В свою очередь Боб зашифровал полученное сообщение своим ключом 3, вычисляя $871^3 \equiv 311 \pmod{1000}$, и отправил результат назад к Алисе. Алиса снимает свой шифр, сосчитав $311^{43} \equiv 631 \pmod{1000}$, а Боб, получив сообщение 631, расшифровывает его с помощью своего второго ключа $631^{67} \equiv 111 \pmod{1000}$. В результате Боб получил незашифрованное письмо Алисы. Заметим также, что всё время сообщение Алисы гуляло по просторам Интернета в зашифрованном виде, ключи Алисы были известны только Алисе, а ключи Боба только ему.

Задача. Определите все числа a из промежутка $2 \leqslant a < 1000$, которые Алиса может секретно переслать Бобу при использовании тех же четырёх ключей и той же процедуры шифрования и расшифрования.

Опишем коротко функцию Эйлера $\varphi(n)$ и теорему Эйлера, необходимые для решения этой задачи.

Для каждого целого числа $n \geqslant 1$ обозначим символом $\varphi(n)$ количество натуральных чисел, не превосходящих n и взаимно простых с n. Другими словами $\varphi(n)$ есть количество целых чисел k, удовлетворяющих условиям

$$1 \leqslant k \leqslant n, \qquad (k, n) = 1.$$

Так определенную функцию натурального аргумента n называют функцией Эйлера.

В частности, имеем $\varphi(1) = 1$. Для каждого простого числа p имеем $\varphi(p) = p - 1$, а также

$$\varphi(p^r) = p^r - p^{r-1} = p^r \cdot (1 - p^{-1})$$

при любом натуральном r. Свойства функции Эйлера описываются следующей теоремой.

Теорема 1. *1. Для любого натурального $n \geqslant 2$ выполняется равенство*

$$\varphi(n) = n \cdot \prod_{p|n} \left(1 - \frac{1}{p}\right). \tag{1.3}$$

2. Функция Эйлера мультипликативна, т.е. для любых двух натуральных взаимно простых чисел a и b имеем

$$\varphi(a \cdot b) = \varphi(a) \cdot \varphi(b).$$

Например, среди целых чисел от 1 до 12 взаимно простыми с 12 будут лишь 4 числа $1, 5, 7, 11$. Их количество может быть также сосчитано по формулам

$$\varphi(12) = 12 \cdot \prod_{p|12} (1 - p^{-1}) = 12 \cdot (1 - 2^{-1}) \cdot (1 - 3^{-1}) = 4 \quad \text{и} \quad \varphi(12) = \varphi(3) \cdot \varphi(4) = 2 \cdot 2 = 4.$$

Следующее утверждение было доказано в 1760г. Л. Эйлером и носит его имя.

Теорема 1.1 (Теорема Эйлера)**.** *Для каждого целого числа a, взаимно простого с модулем m, выполняется сравнение*

$$a^{\varphi(m)} \equiv 1 \pmod{m}.$$

Рассмотрим частный случай теоремы Эйлера, в котором $m = p$ есть простое число. Тогда $\varphi(p) = p - 1$ и получается утверждение, называемое малой теоремой Ферма.

Теорема 1.2 (Малая теорема Ферма)**.** *Если целое число a не делится на простое число p, то*

$$a^{p-1} \equiv 1 \pmod{p}.$$

Из этой теоремы следует, что при любом целом a число $a^p - a = a(a^{p-1} - 1)$ делится на p.

Перейдём теперь к решению поставленной выше задачи.

Решение. Справедливо разложение на множители $1000 = 8 \cdot 125$. С помощью равенства (1.3) находим $\varphi(125) = 125 \cdot (1 - 5^{-1}) = 100$. Поэтому согласно малой теореме Ферма для каждого a, не делящегося на 5 выполняется сравнение $a^{100} \equiv 1 \pmod{125}$. Легко видеть, что для каждого нечётного числа a справедливо сравнение $a^2 \equiv 1 \pmod{8}$. Действительно, имеем $a^2 - 1 = (a-1)(a+1)$. Но из двух последовательных чётных чисел $a-1$ и $a+1$ одно обязательно делится на 4, а другое на 2. Поэтому $(a^2 - 1)$ делится на 8. Возводя обе части сравнения $a^2 \equiv 1 \pmod{8}$ в степень 50, находим $a^{100} \equiv 1 \pmod{8}$. Теперь можно утверждать, что для каждого числа a, взаимно простого с 10, разность $a^{100} - 1$ делится на 125 и на 8, а потому делится на произведение этих чисел, равное 1000. Итак, для каждого целого числа a, взаимно простого с 10, а значит и с 1000, выполняется сравнение $a^{100} \equiv 1 \pmod{1000}$. Учитывая, что $7 \cdot 43 = 301$ и $3 \cdot 67 = 201$, заключаем, что для указанных чисел a выполняются сравнения

$$\left(a^7\right)^{43} = a^{301} = a \cdot \left(a^{100}\right)^3 \equiv a \pmod{1000},$$
$$\left(a^3\right)^{67} = a^{201} = a \cdot \left(a^{100}\right)^2 \equiv a \pmod{1000},$$

и

$$\left(\left(\left(a^7\right)^3\right)^{43}\right)^{67} \equiv a^{7 \cdot 3 \cdot 43 \cdot 67} \equiv a \pmod{1000}.$$

□

1.2 Теорема Ламе

Рассмотрим сначала один пример вычисления наибольшего общего делителя двух чисел, а именно, вычислим $(89, 55)$. Пользуясь алгоритмом Евклида, находим

$$\begin{aligned} 89 &= 55 \cdot 1 + 34, & 55 &= 34 \cdot 1 + 21, & 34 &= 21 \cdot 1 + 13, \\ 21 &= 13 \cdot 1 + 8, & 13 &= 8 \cdot 1 + 5, & 8 &= 5 \cdot 1 + 3, \\ 5 &= 3 \cdot 1 + 2, & 3 &= 2 \cdot 1 + 1, & 2 &= 1 \cdot 2 + 0. \end{aligned}$$

Итак, $(89, 55) = 1$, и для вычисления наибольшего общего делителя понадобилось 9 делений с остатком. Числа 55 и 89 есть числа Фибоначчи с номерами 10 и 11. Напомним, что последовательность чисел Фибоначчи определяется рекуррентным уравнением $F_{i+1} = F_i + F_{i-1}$ и начальными данными $F_0 = 0$, $F_1 = 1$. Приведённое выше вычисление показывает, и это легко доказать по индукции, что для вычисления наибольшего общего делителя чисел (F_n, F_{n+1}) нужно не менее $n - 1$ делений с остатком.

Теорема Ламе, доказываемая в этом параграфе, даёт верхнюю оценку для количества делений с остатком в алгоритме Евклида вычисления наибольшего общего делителя двух чисел, т.е. даёт верхнюю оценку сложности алгоритма. Заметим, что эта оценка может подходить очень близко к возможной нижней оценке, доставляемой примером с числами Фибоначчи. Таким образом оценка теоремы Ламе очень точна.

Теорема 1.3 (Ламе, 1845)**.** *Пусть $a > b$ — натуральные числа. При вычислении (a, b) с помощью алгоритма Евклида будет выполнено не более $5m$ операций деления с остатком, где m есть количество цифр в десятичной записи числа b.*

Доказательство. Положим $r_0 = a$ и обозначим $r_1, r_2, \ldots, r_n$ — последовательность делителей в алгоритме Евклида. Тогда $r_1 = b$,

$$r_{i-1} = q_i r_i + r_{i+1}, \quad 0 \leqslant r_{i+1} < r_i,\ a_i \in \mathbb{N}, \quad i = 1, 2, \ldots, n-1,$$

$r_n = (a, b)$.

Докажем справедливость следующих неравенств

$$r_i \geqslant \lambda^{n-i}, \qquad i = 1, 2, \ldots, n, \tag{1.4}$$

где $\lambda = \frac{1+\sqrt{5}}{2} = 1,61...$ есть корень квадратного уравнения $\lambda^2 - \lambda - 1 = 0$. Для этого воспользуемся индукцией по индексу i, двигаясь в обратном направлении от n к 1. При $i = n$ имеем $r_n \geqslant 1 = \lambda^0$. При $i = n - 1$ находим $r_{n-1} \geqslant r_n + 1 \geqslant 2 > \lambda$, так что неравенство (1.4) опять справедливо. Предположим теперь, что $k < n$

и неравенство (1.4) выполняется при всех $i \geqslant k$. Имеем следующую цепочку равенств и неравенств

$$r_{k-1} = q_k r_k + r_{k+1} \geqslant r_k + r_{k+1} \geqslant \lambda^{n-k} + \lambda^{n-k-1} = \lambda^{n-k-1}(\lambda + 1) = \lambda^{n-k+1}.$$

Таким образом, неравенство (1.4) выполняется и при $i = k - 1$. Это доказывает справедливость (1.4) при всех $i = 1, 2, \ldots, n$.

Поскольку десятичная запись b содержит m знаков, то $10^m > b$, и

$$10^m > b = r_1 \geqslant \lambda^{n-1}.$$

Из этих неравенств следует, что

$$m > (n-1)\log_{10}\lambda > (n-1)/5.$$

Последнее неравенство выполняется, поскольку $\lambda > 10^{1/5} = 1,58\ldots$. Таким образом, $n < 5m + 1$, что завершает доказательство теоремы. □

Из теоремы 1.3 следует, что алгоритм Евклида работает достаточно быстро. Количество арифметических операций в нём оценивается многочленом от длины записи его входных данных. Такие алгоритмы называются *полиномиальными*. **Конец второй лекции.**

Лекция 3.

1.3 Решение уравнений и сравнений.

В этом параграфе мы будем рассматривать простейшие уравнения и сравнения. Главная цель здесь не описание теории, этому будут посвящены разделы базового курса алгебры. Мне же хотелось объяснить здесь удобные методы решения.

Рассмотрим уравнение

$$ax + by = c, \tag{1.5}$$

где a, b, c - целые числа, а неизвестные x, y также предполагаются целыми. Основной инструмент здесь - алгоритм Евклида. Метод решения таких уравнений опишем на примере уравнения $15x + 19y = 2$.

Пользуясь алгоритмом Евклида, находим наибольший общий делитель чисел 19 и 15

$$19 = 15 \cdot 1 + 4, \qquad 15 = 4 \cdot 3 + 3, \qquad 4 = 3 \cdot 1 + 1, \qquad 3 = 1 \cdot 3.$$

Последний ненулевой остаток равен 1, поэтому $(19, 15) = 1$. Составим теперь следующую таблицу.

частные	-	1	3	1	3	-
остатки	19	15	4	3	1	0
x	0	1	-1	4	-5	19
y	1	0	1	-3	4	-15

В верхней её части, состоящей из одной строки, стоят частные при делении в алгоритме Евклида. В первой строке нижней части стоят все остатки при делении в алгоритме Евклида, включая и коэффициенты уравнения. Каждый столбец нижней части таблицы, начиная с четвёртого, получается вычитанием двух предыдущих. Из столбца с меньшим номером вычитается следующий столбец, умноженный на то же частное алгоритма Евклида, что и его первый элемент. Все элементы вычитаемого столбца умножаются на один и тот же множитель. Так имеем

$$\begin{pmatrix} 4 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 19 \\ 0 \\ 1 \end{pmatrix} - \begin{pmatrix} 15 \\ 1 \\ 0 \end{pmatrix}, \qquad \begin{pmatrix} 3 \\ 4 \\ -3 \end{pmatrix} = \begin{pmatrix} 15 \\ 1 \\ 0 \end{pmatrix} - 3 \cdot \begin{pmatrix} 4 \\ -1 \\ 1 \end{pmatrix},$$

$$\begin{pmatrix} 1 \\ -5 \\ 4 \end{pmatrix} = \begin{pmatrix} 4 \\ -1 \\ 1 \end{pmatrix} - \begin{pmatrix} 3 \\ 4 \\ -3 \end{pmatrix}, \qquad \begin{pmatrix} 0 \\ 19 \\ -15 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ -3 \end{pmatrix} - 3 \cdot \begin{pmatrix} 1 \\ -5 \\ 4 \end{pmatrix}.$$

Заполнение таблицы лучше производить по столбцам, двигаясь слева направо. В строке x единица ставится под коэффициентом уравнения при переменной x. Точно так же в строке y единица ставится под коэффициентом уравнения при переменной y. Остальные элементы второго и третьего столбцов нижней части таблицы заполняются нулями.

Завершив создание таблицы, мы тут же можем выписать все решения уравнения

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{2}{1}\begin{pmatrix} -5 \\ 4 \end{pmatrix} + t \cdot \begin{pmatrix} 19 \\ -15 \end{pmatrix} \quad \text{или по другому} \quad x = -10 + 19t, \quad y = 8 - 15t,$$

где t - произвольное целое число. В этих формулах присутствуют два последних столбика строк x, y сосчитанной таблицы. Коэффициент при первом из них равен дроби $\frac{c}{d}$, где c - правая часть уравнения и d - наибольший общий делитель коэффициентов уравнения a, b. Если отношение $\frac{c}{d}$ не целое, то уравнение не имеет целых решений. Каждое решение получается по этим формулам.

Уравнения с иным выбором знаков коэффициентов могут быть сведены к рассмотренному.

Так уравнение $15x - 19y = 2$ может быть переписано в виде $15x + 19(-y) = 2$. Множество его решений имеет вид $x = -10 + 19t, \quad y = -8 + 15t$.

Сравнение вида

$$ax \equiv c \pmod{b} \tag{1.6}$$

означает, что разность $ax - c$ делится на b и потому с некоторым целым числом y должно выполняться равенство (1.8). Решив это уравнение и отбросив за ненадобностью найденные значения y, получим решение сравнения (1.6) в виде $x = x_0 + bt, t \in \mathbb{Z}$. Эту формулу можно переписать в виде $x \equiv x_0 \pmod{b}$. Например, **решим сравнение**

$$41x \equiv 8 \pmod{125}. \tag{1.7}$$

Вместо этого сравнения можно записать уравнение в целых числах

$$41x + 125y = 8.$$

Составляем, как это указывалось выше, таблицу

частные	-	3	20	2	-
остатки	125	41	2	1	0
x	0	1	-3	61	-125
y	1	0	1	-20	41

Последний ненулевой остаток в строке остатков равен 1, значит, коэффициенты уравнения взаимно просты. Все его решения имеют вид

$$\begin{pmatrix} x \\ y \end{pmatrix} = 8 \begin{pmatrix} 61 \\ -20 \end{pmatrix} + t \cdot \begin{pmatrix} -125 \\ 41 \end{pmatrix}.$$

Отбрасывая значение y, получаем решение сравнения $x = 488 - 125t$ или $x \equiv 488 \pmod{125} \equiv 113 \pmod{125}$.

Ответ: $x \equiv 113 \pmod{125}$.

Так как при решении сравнения (1.7) нас не интересуют значения переменной y, то все связанные с ней вычисления, можно опустить, и, в частности, опустить строку y в таблице. В результате таблица примет следующий вид

частные	-	3	20	2	-
остатки	125	41	2	1	0
x	0	1	-3	61	-125

. Ответ, конечно, будет тем же.

1.4 Возведение в степень.

Пусть A — некоторое множество, замкнутое относительно умножения. Это может быть множество целых чисел или многочленов, классов вычетов по какому-нибудь модулю и другие множества. Пусть также d — натуральное число. В этом параграфе мы обсудим быстрый алгоритм вычисления степени a^d. Тривиальный алгоритм, состоящий в последовательном умножении на a результата предшествующего вычисления, требует $d-1$ умножений. Для некоторых чисел d вычисления могут быть проделаны намного быстрее. Так, например, для $d = 32$ можно обойтись всего лишь 5 умножениями

$$a^2, a^4 = (a^2)^2, a^8 = (a^4)^2, a^{16} = (a^8)^2, a^{32} = (a^{16})^2.$$

А для $d = 23$ достаточно 6 умножений

$$a^2, a^3 = a^2 \cdot a, a^5 = a^3 \cdot a^2, a^{10} = (a^5)^2, a^{13} = a^{10} \cdot a^3, a^{23} = a^{13} \cdot a^{10}.$$

Следующий общий алгоритм вычисляет степень существенно быстрее тривиального.

Алгоритм 1.2. Данные: *Элемент $a \in A$ и натуральное число d.*
Найти: *Элемент a^d.*

1. Представить d в двоичной системе счисления, т.е. найти такие числа $d_j \in \{0,\ 1\}$, что $d = d_0 2^r + \cdots + d_{r-1}2 + d_r,\ d_0 = 1$.
2. Положить $a_0 = a$ и затем для $i = 1, \ldots, r$ вычислить

$$a_i = a_{i-1}^2 \cdot a^{d_i}. \tag{1.8}$$

3. Положить $a^d = a_r$.

Теорема 1.4. *Алгоритм действительно вычисляет степень a^d. Он использует для этого не более $2[\log_2 d]$ умножений в кольце A.*

Доказательство. При всех $i = 0, 1, \ldots, r$ справедливы равенства

$$a_i = a^{d_0 2^i + \cdots + d_i}. \tag{1.9}$$

Докажем это индукцией по i. При $i = 0$ утверждение выполняется, т.к. $d_0 = 1$. Подставляя (1.9) в равенство $a_{i+1} = a_i^2 \cdot a^{d_{i+1}}$, находим равенство (1.9) для a_{i+1}. Это доказывает (1.9) для всех i. При $i = r$ получаем $a_r = a^d$, что доказывает корректность алгоритма.

Для оценки сложности обозначим символом $c(d)$ количество умножений, необходимых алгоритму для вычисления a^d. Докажем индукцией по d неравенство

$$c(d) \leqslant 2[\log_2 d]. \tag{1.10}$$

Обозначим для этого $m = d_0 2^{r-1} + \cdots + d_{r-1}$. Тогда $d = 2m + d_r \geqslant 2m$.

При $d = 1,\ 2$ неравенство (1.6), очевидно, выполняется. Пусть теперь $d \geqslant 3$. В силу алгоритма справедливы соотношения

$$c(d) = c(m) + 1 + d_r \leqslant c(m) + 2 \leqslant 2[\log_2 m] + 2 = 2[\log_2 2m] \leqslant 2[\log_2 d],$$

доказывающие нужное неравенство. □

Конец третьей лекции.

Лекция 4.

1.5 Быстрое умножение целых чисел.

Обычный метод умножения n - значных чисел "столбиком" требует n^2 перемножений цифр. В настоящее время имеются более эффективные методы, использующие существенно меньше операций умножения. Первый такой метод был предложен в 1960г. А.А. Карацубой.

Пусть a и b — произвольные натуральные числа. Они имеют, соответственно, по $[\log_2 a]+1$ и $[\log_2 b]+1$ цифр в двоичной записи. Пусть n — минимальное целое число, такое что

$$\max\Big([\log_2 a]+1,\ [\log_2 b]+1\Big) \leqslant 2^n.$$

Тогда a, b не более чем 2^n - разрядные целые числа.

Описываемый ниже алгоритм сводит задачу умножения чисел a и b к нескольким умножениям чисел, длина которых в двоичной записи на превосходит $k = 2^{n-1}$, и рекурсивно применяется к этим новым числам. Вычислительный алгоритм называется рекурсивным, если в процессе работы он обращается к самому себе, но с меньшими параметрами. Возникающее при этом множество обращений в конечном итоге сводит задачу к некоторым начальным значениям параметров, при которых выполнение алгоритма становится тривиальным.

Представим числа a и b в виде

$$a = a_1 + 2^k a_2, \qquad b = b_1 + 2^k b_2,$$

где a_1, a_2, b_1, b_2 — числа, длина которых в двоичной записи не превосходит k. Справедливо равенство

$$ab = a_1 b_1 + 2^k((a_1 + a_2)(b_1 + b_2) - (a_1 b_1 + a_2 b_2)) + 2^{2k} a_2 b_2. \tag{1.11}$$

Числа $a_1 + a_2$ и $b_1 + b_2$ могут иметь в двоичной записи длину $k + 1$, но их можно представить в виде

$$a_1 + a_2 = \alpha + 2a_3, \qquad b_1 + b_2 = \beta + 2b_3,$$

где числа a_3 и b_3 имеют длину не больше, чем k, а числа α и β суть нули или единицы. Стало быть, (1.11) можно переписать в виде

$$ab = a_1 b_1 + 2^k(\alpha\beta + 2\alpha b_3 + 2\beta a_3 + 4a_3 b_3 - (a_1 b_1 + a_2 b_2)) + 2^{2k} a_2 b_2. \tag{1.12}$$

Таким образом, исходная задача при помощи нескольких операций сложения, вычитания и домножения на степень двойки (которое является просто–напросто

приписыванием соответствующего числа нулей) сводится к вычислению трех произведений a_1b_1, a_2b_2 и a_3b_3, каждое из которых является произведением чисел длины не более, чем $k = 2^{n-1}$. К этим трем произведениям рекурсивно применяем описанные рассуждения.

Обозначим через L_n количество битовых операций, необходимое для вычисления данным методом произведения двух произвольных чисел, длина которых в двоичной записи не превосходит 2^n. Тогда найдется такая константа C, что $L_0 \leqslant C$ и

$$L_n \leqslant 3L_{n-1} + 2^n C, \qquad n \geqslant 1.$$

Отсюда по индукции находим, что

$$L_n \leqslant C(3^{n+1} - 2^{n+1}).$$

Действительно, в предположении, что $L_{n-1} \leqslant C(3^n - 2^n)$, получаем:

$$L_n \leqslant 3C(3^n - 2^n) + C2^n = C(3^{n+1} - 2^{n+1}).$$

Учитывая теперь, что $2^{n-1} < [\log_2 a] + 1$, при $a \geqslant b \geqslant 2$ заключаем

$$L_n \leqslant C \cdot 3^{n+1} = 3C \cdot (2^n)^{\log_2 3} < 3C \cdot (2[\log_2 a] + 2)^{\log_2 3} = 9C \cdot ([\log_2 a] + 1)^{\log_2 3}.$$

Таким образом, если a и b — натуральные числа и $a \geqslant b$, то перемножить их можно не более, чем за $9C \cdot m^{\log_2 3}$ битовых операций, где $m = [\log_2 a] + 1$ - количество цифр в двоичной записи числа a.

В 1971г. Шёнхаге и Штрассен придумали алгоритм умножения двух больших целых чисел, требующий не более $C_1 m \log_2 m \log_2 \log_2 m$ битовых операций, где m - количество цифр в двоичной записи большего из сомножителей, а C_1 - некоторая абсолютная постоянная, см. [8]. В 2007г. М. Фюрер, см. [5], предложил усовершенствование, позволившее заменить второй логарифм в оценке сложности алгоритма функцией, растущей существенно медленнее. Ещё одно усовершенствование было предложено в работе [3]. Предполагается, что существует алгоритм умножения с битовой сложностью порядка $m \log_2 m$. Найти такой алгоритм - это известная открытая математическая проблема.

1.6 Вероятностные методы отсеивания составных чисел.

Рассмотренные ранее алгоритмы относятся к разряду детерминированных. Так называют алгоритмы, в которых результат каждого шага однозначно определяется предшествующими вычислениями. Однако, существуют и другие типы алгоритмов, так называемые вероятностные алгоритмы, весьма удобные на практике.

Пусть N — натуральное число, вообще говоря, большое. Оно может быть либо составным, либо простым. Соответственно, можно рассмотреть две важные в связи с поисками или построением больших простых чисел задачи.
1. N — составное число, и требуется доказать это.
2. N — простое число, и требуется доказать это.

В настоящем параграфе будет рассматриваться первая из задач. Мы покажем, что существуют достаточно быстрые вероятностные алгоритмы, решающие ее и не использующие при этом разложение N на множители.

Выберем целое число $a, 1 < a < N$. Справедливы следующие утверждения:
1) Если наибольший общий делитель $(a, N) > 1$, то N — составное число.

Условие $(a, N) > 1$ легко проверить, вычислив (a, N) с помощью алгоритма Евклида.

2) Если $(a, N) = 1$ и $a^{N-1} \not\equiv 1 \pmod N$, то N — составное число.

Это утверждение следует из малой теоремы Ферма, и его также легко проверить с помощью алгоритма возведения в степень.

К сожалению, утверждений 1)-2) не достаточно для доказательства того, что испытываемое число N — составное. Существуют составные числа, для которых выполняется сравнение $a^{N-1} \equiv 1 \pmod N$ при любом целом $a, (a, N) = 1$.

Определение 1.2. *Составное число N называется числом Кармайкла, если при любом $a, (a, N) = 1$, выполняется сравнение*

$$a^{N-1} \equiv 1 \pmod N. \tag{1.13}$$

Пример. $N = 561 = 3 \cdot 11 \cdot 17$.

Для каждого целого $a, (a, 561) = 1$, имеем $(a, 3) = (a, 11) = (a, 17) = 1$ и по малой теореме Ферма выполняются сравнения

$$a^2 \equiv 1 \pmod 3, \qquad a^{10} \equiv 1 \pmod{11}, \qquad a^{16} \equiv 1 \pmod{17}.$$

Так как 560 делится на 2, 10 и 16, то число $a^{560}-1$ делится на каждую из разностей $a^2 - 1, a^{10} - 1, a^{16} - 1$. Значит, $a^{560} - 1$ делится на 3, 11, 17 и потому делится на произведение этих чисел, т.е. делится на 561. Число 561 есть число Кармайкла.

В 1994г. было доказано, что множество чисел Кармайкла бесконечно, см. [2]. Существование чисел Кармайкла показывает необходимость уточнения свойств 1)-2) для доказательства непростоты чисел. Следующее утверждение позволяет достаточно эффективно решать рассматриваемую задачу.

Тест 1 (Миллер - Рабин, [6]). *Пусть $N > 2$ — нечетное натуральное число. Определим целые числа s, t равенством $N - 1 = 2^s t$, где t нечетно. Выберем целое число $a, a > 1$.*

1) Если $(a, N) > 1$, *то* N *составное число.*
2) Если $(a, N) = 1$ *и выполнены условия*

$$a^t \not\equiv 1 \pmod{N}, \quad a^{2^k t} \not\equiv -1 \pmod{N}, \quad k = 0, 1, \ldots, s-1, \tag{1.14}$$

то N *составное число.*

Справедливо разложение на множители

$$a^{N-1} - 1 = (a^t - 1)(a^t + 1)(a^{2t} + 1) \cdots (a^{2^{s-1}t} + 1).$$

Если N — простое число, то по малой теореме Ферма обе части последнего равенства должны делиться на N. Поскольку N — простое, то хотя бы один из сомножителей в правой части этого равенства должен делиться на N. Значит, хотя бы одно из условий (1.14) будет нарушено. Это доказывает справедливость второго утверждения теста Миллера - Рабина.

Рассмотрим множество $M(N)$ состоящее из чисел $a \in \mathbb{Z},\ 1 \leqslant a < N,\ (a, N) = 1$, для которых нарушается хотя бы одно из условий (1.14).

Если N — простое число, то $\#M(N) = N - 1$. Это следует из справедливости второго утверждения теста Миллера - Рабина. Для составных N множество $M(N)$ будет достаточно малым. Точнее, выполняется доказанное в 1984г. Рабином утверждение.

Теорема 1.5. *Пусть* N *— нечетное составное число,* $N \neq 9$. *Тогда* $\#M(N) \leqslant \frac{1}{4}\varphi(N)$.

Мы не приводим здесь доказательство этой теоремы. Оно элементарно, но длинно, см. книгу [1].

Пример. *Для* $N = 9$ *имеем* $M(9) = \{1,\ 8\}$ *и* $\#M(9) = 2 = \frac{1}{3}\varphi(9)$.

Приведённый ниже вероятностный алгоритм, удостоверяет, что заданное число — составное, если оно таковым является. Алгоритм основан на тесте Миллера-Рабина.

Алгоритм 1.3. Данные: *Нечетное число* $N > 9$ *повидимому составное.*
Доказать: *Число* N *составное.*

1) Вычислить натуральные числа s, t *такие, что* $N - 1 = 2^s t$ *и* t *нечетно.*
2) Выбрать случайным образом число $a, 1 < a < N$ *и проверить выполнимость условий 1) и 2) теста Миллера - Рабина.*
3) Если хотя бы одно из условий теста выполнено, то число N *составное; СТОП.*
4) Если оба условия теста нарушаются, то перейти в пункт 2.

Для оценки сложности этого вероятностного алгоритма, что нужно для сравнения эффективности различных алгоритмов, можно использовать среднее время работы алгоритма, оно же иногда называется математическое ожидание времени работы. Мы оценим эту сложность для приведенного алгоритма на следующей лекции.

Конец четвёртой лекции

Лекция 5.

В конце прошлой лекции был описан вероятностный алгоритм, свидетельствующий, что данное нам число N, якобы являющееся составным, действительно таковым является. Докажем сейчас, что среднее количество арифметических операций, необходимое для выполнения работы этого алгоритма, т.е. его сложность, не превосходит $c_0 \cdot \ln N$.[1]

Напомним определение множества $M(N)$, количество его элементов оценивается в теореме 1.5. Согласно определению это множество состоит из всех чисел $a \in \mathbb{Z},\ 1 \leqslant a < N,\ (a, N) = 1$, для которых нарушается хотя бы одно из условий (1.14). Таким образом, любое целое число a, $1 \leqslant a < N$, не принадлежащее этому множеству, подтвердит с помощью теста Миллера - Рабина, что число N составное. Значит при случайном выборе a мы с вероятностью

$$\rho = \frac{N - 1 - \#M(N)}{N - 1} \geqslant 1 - \frac{1}{4}\frac{\varphi(N)}{N-1} \geqslant \frac{3}{4}$$

попадаем на хорошее a (доказывающее непростоту N). Здесь использовалась оценка сверху $\#M(N) \leqslant \frac{1}{4}\varphi(N)$ количества элементов в множестве $M(N)$, справедливая для всех нечетных составных чисел $N > 9$, см. теорему 1.5.

Пусть A_k — событие, состоящее в том, что при k испытаниях $k-1$ раз попадались плохие a и в k-й раз попалось хорошее. Тогда $p(A_k) = (1-\rho)^{k-1}\rho$. Так как тест Миллера – Рабина использует только вычисление наибольшего общего делителя двух чисел и возведение в степень по модулю N, то при фиксированном a для проверки условий теста 2 требуется не более $c_1 \ln N$ арифметических операций. Для проверки же принадлежности $a \in A_k$ потребуется не более $c_1 k \ln N$ арифметических операций. Среднее количество арифметических операций в алгоритме не превосходит

$$c_1 \ln N \cdot \sum_{k=1}^{\infty} k(1-\rho)^{k-1}\rho = \rho^{-1} \cdot c_1 \ln N \leqslant 4c_1 \ln N = c_0 \ln N.$$

Здесь в вычислениях использовалось при $x = 1 - \rho$ тождество

$$\sum_{k=1}^{\infty} kx^k = \frac{x}{(1-x)^2}, \qquad 0 \leqslant x < 1. \tag{1.15}$$

Оно помещено среди задач, но мы разберём здесь его доказательство.

[1] Здесь и далее буквой c с индексами обозначаются различные абсолютные постоянные, т.е. постоянные, не зависящие ни от каких параметром алгоритма, например $3, \pi$ или e^2.

Сначала докажем, что при любом действительном $x \geqslant 0$ и любом целом $n \geqslant 0$ справедливо неравенство

$$(1+x)^n \geqslant 1 + nx + \frac{n(n-1)}{2}x^2. \tag{1.16}$$

Для этого воспользуемся методом математической индукции. При $n = 0, 1, 2$ неравенство (1.16) превращается в равенство и потому выполняется. Предположим, что $n \geqslant 2$ и неравенство (1.16) справедливо. Умножив его на положительное число $1 + x$, получим

$$(1+x)^{n+1} \geqslant (1+x)(1+nx+\frac{n(n-1)}{2}x^2) = 1+(n+1)x+\frac{(n+1)n}{2}x^2+\frac{n(n-1)}{2}x^3 > \\ > 1 + (n+1)x + \frac{(n+1)n}{2}x^2.$$

В силу аксиомы индукции это доказывает неравенство (1.16) при любом $n \geqslant 0$..

Теперь докажем, что для любого $x, 0 \leqslant x < 1$, последовательность nx^n стремится к нулю при $n \to +\infty$. Для $x = 0$ утверждение, очевидно, выполняется. Для положительных x сделаем замену $x = \frac{1}{1+y}$. Тогда $y = \frac{1}{x} - 1 > 0$ и, пользуясь (1.16), находим

$$0 \leqslant nx^n = \frac{n}{(1+y)^n} \leqslant \frac{n}{1 + ny + \frac{n(n-1)}{2}y^2} < \frac{2}{(n-1)y^2}.$$

Правое выражение при любом фиксированном y и $n \to \infty$ стремится к нулю. Это доказывает нужное утверждение, а также в силу неравенств $0 \leqslant x^n \leqslant nx^n$ и стремление к нулю последовательности x^n.

Наконец, для доказательства (1.15) обозначим

$$s_n = x + 2x^2 + 3x^3 + \cdots + nx^n.$$

Тогда

$$(1-x)s_n = x + 2x^2 + 3x^3 + \cdots + nx^n - (x^2 + 2x^3 + 3x^4 + \cdots + nx^{n+1}) = \\ = x+x^2+x^3+\ldots+x^n-nx^{n+1} = \frac{1-x^{n+1}}{1-x} - 1 - nx^{n+1} = \frac{x}{1-x} - x^n\frac{x}{1-x} - x\cdot nx^n.$$

С увеличением числа слагаемых, т.е. при $n \to +\infty$, правая часть имеет предел, равный $\frac{x}{1-x}$. Поэтому и последовательность s_n имеет предел, равный $\frac{x}{(1-x)^2}$.

На практике алгоритм 1.3 является очень важной составляющей общей стратегии доказательства простоты чисел. Если заданное число N, о котором не известно простое оно или составное, прошло достаточно много шагов алгоритма 1.3, то оно наверное будет простым. Ведь вероятность составному числу N выдержать d испытаний не превосходит 4^{-d}. Например, при $d = 100$ она ничтожно мала 4^{-100}. Таким образом, остается только вопрос, как доказать, что это число простое?

1.7 Построение больших простых чисел.

В следующем параграфе мы обсудим алгоритмы, позволяющие доказывать простоту чисел. Здесь же я укажу одно утверждение, итерации которого позволяют строить большие простые числа. Доказано оно будет в следующем параграфе.

Предложение 1.1. *Пусть F – простое нечётное число, R – чётное число из промежутка $F \leqslant R \leqslant 4F+2$ и $N = FR+1$. Если N удовлетворяет условиям*

$$b^{N-1} \equiv 1 \pmod{N}, \qquad \bigl(b^R - 1,\, N\bigr) = 1. \tag{1.17}$$

с некоторым b из промежутка $1 < b < N$, то N – простое число.

Применяя это утверждение, чётное число R из промежутка $F \leqslant R \leqslant 4F+2$ можно выбирать случайным образом, причём так, чтобы число $N = FR+1$ не имело малых простых делителей и проходило бы достаточно большое число раз тест Миллера Рабина.

Заметим, что выбирая при найденном R случайным способом число b на промежутке $1 < b < N$ и проверяя для него выполнимость условий (1.17), в случае успеха можно найти большее простое число. Построенное таким способом простое число N будет удовлетворять неравенству $N > F^2$, т.е. будет записываться вдвое большим количеством цифр, чем исходное простое число F. Заменив теперь число F на найденное простое число N и повторив с этим новым F все указанные выше действия, можно построить еще большее простое число. Начав с какого-нибудь простого числа, скажем, записанного 10 десятичными цифрами (простоту его можно проверить, например, делением на маленькие табличные простые числа), и повторив указанную процедуру достаточное число раз, можно построить простые числа нужной величины.

Конец пятой лекции.

Лекция 6.

1.8 Доказательство простоты чисел.

Мы будем рассматривать здесь лишь так называемые $(N-1)$ - методы доказательства простоты чисел. Они позволяют, используя некоторую информацию о свойствах простых делителей числа $N-1$, заключить, что N - простое число.

Теорема 1.6 (Лемер, Поклингтон). *Пусть N нечетно, $N-1 = F \cdot R$, причем для каждого простого делителя q числа F с некоторым целым b выполнены условия*

$$b^{N-1} \equiv 1 \pmod{N}, \qquad \left(b^{(N-1)/q} - 1,\, N\right) = 1. \tag{1.18}$$

Тогда любой простой делитель p числа N удовлетворяет сравнению

$$p \equiv 1 \pmod{F}$$

Приведём два следствия теоремы, связанные с доказательством простоты чисел. Если известна достаточно большая часть разложения $N-1$ на простые сомножители, то иногда можно сделать заключение о простоте N.

Следствие 1.1. *Если в условиях теоремы 1.6 выполнено неравенство*

$$R \leqslant F + 1$$

то N — простое.

Доказательство. Согласно теореме 1.6 каждое простое $p|N$ удовлетворяет сравнению $p \equiv 1 \pmod{F}$ и, значит, удовлетворяет неравенству $p \geqslant 1 + F$. Предположим, что N — составное. Тогда

$$(F+1)^2 \leqslant N = FR + 1 \leqslant F^2 + F + 1.$$

Получившееся противоречие завершает доказательство. □

Следствие 1.2. *Пусть F – простое нечётное число, R – чётное число из промежутка $F \leqslant R \leqslant 4F + 2$ и $N = FR + 1$. Если N удовлетворяет условиям*

$$b^{N-1} \equiv 1 \pmod{N}, \qquad \left(b^{R} - 1,\, N\right) = 1. \tag{1.19}$$

с некоторым b из промежутка $1 < b < N$, то N – простое число.

Доказательство. Согласно теореме 1.6 при $q = F$ каждое простое p, делящее N удовлетворяет сравнению $p \equiv 1 \pmod F$. Кроме того, p нечетно, так что $p \equiv 1 \pmod{2F}$ и $p \geqslant 2F + 1$. Для составного N имеем неравенства

$$(2F + 1)^2 \leqslant N = FR + 1 \leqslant 4F^2 + 2F + 1,$$

что неверно. Значит, N — простое число. □

В следующем далее доказательстве теоремы 1.6 используется символ $\nu_q(a)$, обозначающий для любого целого a и простого q кратность, с которой q входит в разложение a на простые сомножители.

Доказательство. Пусть p — простой делитель N, q — простой делитель F и b — число, удовлетворяющее (1.18). Первое из сравнений (1.18) означает, что b не делится на p. Обозначим $a = b^R$. Тогда a не делится на p и по малой теореме Ферма выполняется сравнение $a^{p-1} \equiv 1 \pmod p$. Обозначим буквой d наименьшее натуральное число с условием $a^d \equiv 1 \pmod p$. Разделим $p-1$ на d с остатком: $p - 1 = ud + v$, где $0 \leqslant v < d$. Тогда

$$1 \equiv a^{p-1} = (a^d)^u \cdot a^v \equiv a^v \pmod p.$$

Если $v > 0$, приходим к противоречию с определением d. Поэтому $v = 0$,

$$d|(p - 1) \quad \text{и} \quad \nu_q(d) \leqslant \nu_q(p - 1). \tag{1.20}$$

Из (1.18) следует, что

$$a^F \equiv 1 \pmod p, \qquad a^{F/q} \not\equiv 1 \pmod p.$$

Отсюда точно так же, как и (1.20) выводятся свойства $d|F$ и $d \nmid F/q$, что возможно лишь в случае, когда

$$\nu_q(F) = \nu_q(d). \tag{1.21}$$

Из (1.20) и (1.21) находим

$$\nu_q(F) = \nu_q(d) \leqslant \nu_q(p - 1).$$

Так как последнее неравенство справедливо для любого простого делителя q числа F, то $F|p - 1$ или $p \equiv 1 \pmod F$. □

Если условия (1.18) нарушаются, нужно выбрать другое значение b и повторять эти операции до тех пор, пока такое число не будет обнаружено.

Предположим теперь, что построенное число N действительно является простым. Зададимся вопросом, сколь долго придется перебирать числа b, пока не будет найдено такое, для которого выполнены условия (1.18). Заметим, что для

простого числа N первое условие (1.18), согласно малой теореме Ферма, будет выполняться всегда. Те же числа b, для которых нарушается второе условие (1.18), удовлетворяют сравнению $b^R \equiv 1 \pmod N$. Как известно, уравнение $x^R = 1$ в поле вычетов $\mathbb{Z}/N\mathbb{Z}$ имеет не более R решений. Одно из них — $x = 1$. Поэтому на промежутке $1 < b < N$ имеется не более $R - 1$ чисел, для которых не выполняются условия (1.18). Это означает, что, выбирая случайным образом число b на промежутке $1 < b < N$, при простом N можно с вероятностью большей, чем $1 - F^{-1}$, найти число b, для которого будут выполнены условия следствия 1.2, и тем доказать, что N действительно является простым числом.

Алгоритм 1.4. Данные: *Большое положительное целое число M. Алгоритм строит простое число N, превосходящее границу M.*

1) Определить, например, с помощью таблиц простое число F, превосходящее 10^5. Определить счётчик $S1$ и положить $S1 = 0$.
2) Выбрать случайным образом чётное число R из промежутка $F \leqslant R \leqslant 4F+2$ и положить $N = FR + 1$. Если НОД$[N, 15015] > 1$, повторить пункт 2.
3) Вычислить натуральные числа s, t такие, что $N - 1 = 2^s t$ и t нечетно.
4) Выбрать случайным образом число $a, 1 < a < N$. Если
• $(a, N) > 1$ *или*
• $(a, N) = 1$ *и*

$$a^t \not\equiv 1 \pmod N, \quad a^{2^k t} \not\equiv -1 \pmod N, \quad k = 0, 1, \ldots, s-1, \tag{1.22}$$

то N составное число. Перейти в пункт 2). В противном случае увеличить счётчик $S1$ на 1 и, если $S1 < 20$, перейти в п. 4. Если же $S1 = 20$, положить $S1 = 0$ и $S2 = 0$, перейти в п. 5).
5) Выбрать случайным образом b из промежутка $1 < b < N$. Если N не удовлетворяет хотя бы одному из условий

$$b^{N-1} \equiv 1 \pmod N, \qquad \left(b^R - 1,\, N\right) = 1, \tag{1.23}$$

то при $S2 < 10$ увеличить $S2$ на 1 и повторить пункт 5). В случае же $S2 = 10$, перейти в пункт 4.
6) Если N удовлетворяет обоим условиям (1.23) и $N < M$ положить $F = N$ и перейти в пункт 2.
7) Если $N \geqslant M$ алгоритм останавливается, нужное простое число построено.

Конец шестой лекции.

Литература

[1] Герман О.Н., Нестеренко О.Н., Теоретико-числовые методы в криптографии, Москва, ИЦ Академия, 2012.

[2] W.R. Alford, A. Granville, C. Pomerance, There are infinitely many Carmichael numbers, Ann. Math., 1994, v.140, p.703-722.

[3] A. De, P. Kurur, Ch. Saha, R. Saptharishi, Fast Integer Multiplication Using Modular Arithmetic. Symposium on Theory of Computation (STOC) 2008. arXiv:0801.1416

[4] W. Diffie, M. Hellman. New directions in cryptography. IEEE Trans. Inform. Theory, vol.22, pp.644–654, 1976.

[5] M. Fürer. Faster integer multiplication. Proceedings of the 39th ACM STOC 2007, pp. 57–66.

[6] M. Rabin, Probabilistic algorithms for testing primality, J. Number Theory, 1980, v.12, p. 128-138.

[7] R. Rivest, A. Shamir, L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Comm. ACM, vol.21, pp.120–126, 1978.

[8] A. Schönhage, V. Strassen. Schnelle Multiplikation grosser Zahlen. Computing (Arch. Elektron. Rechnen), vol.7, pp.281–292, 1971.