Теоретико-числовые алгоритмы. Семинар 7. 21 марта 2019 г.

Подготовил: Горбунов Э.

Ключевые слова: алгоритм Евклида, расширенный алгоритм Евклида, модульная арифметика, линейные диофантовые уравнения, китайская теорема об остатках, теорема Эйлера, малая теорема Ферма, дискретный логарифм, протокол Диффи-Хеллмана, схема RSA, цифровая подпись, квадратичный вычет, символ Лежандра, символ Якоби, тест Ферма, тест Соловея-Штрассена, тест Миллера-Рабина

Литература: [Кормен 1, Глава 33], [Кормен 2, Глава 31], [ДПВ, Глава 2], [Виноградов]

Мотивировка

Долгое время было принято считать теорию чисел совершенно бесполезной областью чистой математики. Годфрид Харди — известный специалист по теории чисел — написал в своей книге «Апология математика»: *«Я никогда не делал чего-нибудь «полезного». Ни одно мое открытие не принесло и не могло бы принести, явно или неявно, к добру или ко злу, ни малейшего изменения в благоустройстве этого мира.»* Он бы очень удивился, если бы узнал, что теоремы теории чисел найдут непосредственное применение, например, в банковских операциях. Сейчас теоретико-числовые алгоритмы широко используются в криптографических схемах, в которых, например, важно находить большие простые числа.

Договоримся, что в данном разделе мы будем использовать битовую сложность (то есть размер входа для нас — количество битов, использованных для записи чисел, число операций теоретико-числового алгоритма мы будем измерить в битовых операциях).

Алгоритм Евклида

Начнём мы с одного из самых древних алгоритмов — алгоритма Евклида. Кратко напомню, что алгоритм Евклида предназначен для того, чтобы для пары целых чисел (a, b), где $a \geqslant b$, найти НОД (a, b).

```
procedure Euclid(a, b)
    if b = 0 then
        return a
    else
        return Euclid(b, a mod b)
    end if
end procedure
```

Этот алгоритм основан на простом свойстве наибольшего общего делителя: НОД $(a, b) =$ НОД $(a, a - b)$. Время работы процедуры Euclid(a, b) пропорционально глубине рекурсии. Пусть F_k — k-е число Фибоначчи. Тогда справедлива следующая лемма.

Лемма 1. Пусть $a > b \geqslant 0$. Если процедура Euclid(a, b) во время работы вызывает себя k раз ($k \geqslant 1$), то $a \geqslant F_{k+2}$ и $b \geqslant F_{k+1}$.

Доказательство. Доказательство проведём индукцией по k.

База индукции. Пусть $k = 1$. Если происходит хотя бы один рекурсивный вызов, то $b > 0$, а значит, $b \geqslant 1 = F_2 \Longrightarrow a \geqslant 2 = F_3$. База индукции доказана.

Шаг индукции. Пусть утверждение выполнено для $k - 1$ вызовов. На первом шаге процедура Euclid(a, b) вызывает процедуру Euclid$(b, a \mod b)$, внутри которой происходит $k-1$ рекурсивный вызов. Применим предположение индукции для Euclid$(b, a \mod b)$: $b \geqslant F_{k+1}$ и $(a \mod b) \geqslant F_k$. Из неравенства $a > b > 0$ следует, что $\left\lfloor \frac{a}{b} \right\rfloor \geqslant 1$ и $b + (a \mod b) = b + \left(a - \left\lfloor \frac{a}{b} \right\rfloor \cdot b\right) \leqslant a$, так что $a \geqslant b + (a \mod b) \geqslant F_{k+1} + F_k = F_{k+2}$, что и требовалось доказать. □

Из доказанной леммы получаем следующую теорему.

Теорема 1. (Ламе). Пусть k — целое положительное число. Если $a > b \geqslant 0$ и $b < F_{k+1}$, то процедура $\text{Euclid}(a, b)$ выполняет менее k рекурсивных вызовов.

Поскольку F_k примерно равно $\frac{\varphi^k}{\sqrt{5}}$, где $\varphi = \frac{1+\sqrt{5}}{2}$ — «золотое сечение», то число рекурсивных вызовов для процедуры $\text{Euclid}(a, b)$ составляет $O(\log b)$ при $a > b \geqslant 0$. Если процедура Euclid применяется к двум m-битовым числам, то ей приходится выполнять $O(m)$ арифметических операций, а значит, $O(m^3)$ битовых операций. На самом деле эта оценка достаточно грубая. Можно показать, что число битовых операций равняется $O(m^2)$.

Расширенный алгоритм Евклида

Допустим, кто-то утверждает, что нашёл нибольший общий делитель d чисел a и b. Как он может убедить нас в этом? Не заново же нам искать НОД. Оказывается, что в качестве подтверждения достаточно предъявить представление числа d в виде суммы вида $ax + by$.

Лемма 2. Если d делит оба числа a и b, а также $d = ax+by$ для некоторых целых чисел x и y, то $d = \text{НОД}\,(a, b)$

Доказательство. Так как d — общий делитель чисел a и b, то он не превосходит наибольшего общего делителя по определению: $d \leqslant \text{НОД}\,(a, b)$. С другой стороны, $\text{НОД}\,(a, b)$ делит a и b, а значит, делит и $ax + by = d$, то есть $\text{НОД}\,(a, b) \leqslant d$. Из этих двух неравенств получаем, что $d = \text{НОД}\,(a, b)$. ☐

Оказывается, что можно немного видоизменить процедуру $\text{Euclid}(a, b)$ так, чтобы она возвращала и коэффициенты x и y указанной линейной комбинации.

procedure $\textsc{Extended-Euclid}(a, b)$
    if $b = 0$ **then**
        return $(1, 0, a)$
    end if
    $(x', y', d) \leftarrow \textsc{Extended-Euclid}(b, a \mod b)$
    return $(y', x' - \left\lfloor \frac{a}{b} \right\rfloor y', d)$
end procedure

Лемма 3. Для произвольных неотрицательных чисел a и b ($a \geqslant b$) расширенный алгоритм Евклида возвращает целые числа x, y, d, для которых $\text{НОД}\,(a, b) = d = ax + by$.

Доказательство. Во-первых, если выкинуть из алгоритма x и y, то получим обычный алгоритм Евклида, поэтому он действительно вычисляет $d = \text{НОД}\,(a, b)$. Оставшееся утверждение докажем инукцией по b.
База индукции для $b = 0$ очевидна.
Шаг индукции. Воспользуемся предположением индукции для рекурсивного вызова $\text{Extended-Euclid}(b, a \mod b)$ (это корректно, ведь $a \mod b < b$):

$$\text{НОД}\,(b, a \mod b) = bx' + (a \mod b)y'.$$

Перепишем $(a \mod b)$ как $\left(a - \left\lfloor \frac{a}{b} \right\rfloor \cdot b\right)$ и получим:

$$\begin{aligned} d &= \text{НОД}\,(a, b) = \text{НОД}\,(b, a \mod b) = bx' + (a \mod b)y' \\ &= bx' + \left(a - \left\lfloor \tfrac{a}{b} \right\rfloor \cdot b\right) y' = ay' + b\left(x' - \left\lfloor \tfrac{a}{b} \right\rfloor \cdot y'\right). \end{aligned}$$

Итак, $d = ax + by$ при $x = x'$ и $y = x' - \left\lfloor \frac{a}{b} \right\rfloor \cdot y'$, что и требовалось доказать. ☐

Вспоминаем высшую алгебру

1. **Группы $\mathbb{Z}_n$ и $\mathbb{Z}_n^*$.** Будем обозначать через $(\mathbb{Z}_n, +_n)$ — аддитивную (по сложению) группу[1] вычетов по модулю n, а через $(\mathbb{Z}_n^*, \cdot_n)$ — мультипликативную (по умножению) группу вычетов по модулю n (далее будем опускать индекс n при обозначении групповой операции, а сами группы обозначать $\mathbb{Z}_n$ и $\mathbb{Z}_n^*$ соответственно). Отметим, что аддитивная группа вычетов содержит вычеты $[0]_n, [1]_n, \ldots, [n-1]_n$ (то есть все вычеты), а мультипликативная группа состоит только из вычетов, взаимно простых с n. Количество элементов в группе $(\mathbb{Z}_n^*, \cdot_n)$ определяется *функцией Эйлера*, которая обозначается через $\varphi(n)$ (то есть $\varphi(n)$ — это количество чисел, которые меньше n и взаимно просты с n). Можно доказать, что

$$\varphi(n) = n\left(1 - \frac{1}{p_1}\right) \cdot \ldots \cdot \left(1 - \frac{1}{p_s}\right),$$

где $p_1, p_2, \ldots, p_s$ — список всех (различных) простых делителей числа n. Это делается в три шага:

   1) показать, что для простых n выполняется $\varphi(n) = n - 1$;
   2) показать, что для всех n вида $n = p^k$, где p — простое число, выполняется $\varphi(n) = p^k - p^{k-1}$;
   3) показать, что для взаимно простых чисел m и n выполняется $\varphi(mn) = \varphi(m)\varphi(n)$.

2. **Порядок подгруппы.** Далее мы ещё вернёмся к функции Эйлера, а пока что вспомним важный факт о порядке[2] подгруппы[3].

   Теорема 2. (Лагранж). Если $(G', \circ)$ является подгруппой группы $(G, \circ)$, то $|G'|$ является делителем числа $|G|$.

   На доказательстве останавливаться не будем. Напомню лишь, что делается это при помощи рассмотрения *левых смежных классов* подгруппы G' (то есть множеств вида $x \circ G' = \{x \circ y \mid y \in G'\}$ для каждого x). Отметим важное следствие теоремы Лагранжа.

   Следствие 1. Если G' является собственной подгруппой[4] конечной группы G, то $|G'| < \frac{|G|}{2}$.

3. **Порядок элемента. Образующие.** Пусть $x \in G$, где G — конечная группа. Будем применять групповую операцию $\circ$ к элементам e и x, затем к x и x, потом к $x \circ x$ и x и так далее. В результате мы получим последовательность

$$e,\ x,\ x \circ x,\ x \circ x \circ x, \ldots\ .$$

   Для $k \in \mathbb{N}$ введём обозначения: $x^{(k)} = \underbrace{x \circ x \circ \ldots \circ x}_{k\ \text{штук}}$ и $x^{(-k)} = \left(x^{-1}\right)^{(k)} = \left(x^{(k)}\right)^{-1}$. Кроме того, будем считать, что $x^{(0)} = e$. Тогда указанная выше последовательность может быть записана следующим образом:

$$x^{(0)},\ x^{(1)},\ x^{(2)},\ x^{(3)}, \ldots\ .$$

   Так как группа G конечна, то найдутся такие i и j, что $x^{(i)} = x^{(j)}$, что эквивалентно тому, что найдётся такое целое неотрицательное число m, что $x^{(m)} = x^{(0)} = e$. Понятно, что таких чисел много (если $x^{(m)} = e$, то и $x^{(2m)} = e, x^{(3m)} = e$ и так далее. Нас же будет интересовать *наименьшее* целое неотрицательное число m, что $x^{(m)} = e$. Из курса высшей алгебры мы знаем, что это число называют *порядком* элемента

[1] Напомню, что пара $(G, \circ)$ называется *группой*, если G — это некоторое непустое множество с бинарной операцией $\circ$, определённой на этом множестве, которая удовлетворяет следующим свойствам.

0. $\forall x, y \in G \hookrightarrow x \circ y \in G$ (в таких случаях говорят, что множество G замкнуто относительно операции $\circ$).
1. Существует элемент $e \in G$, называемый нейтральным элементом группы, такой что $\forall x \in G \hookrightarrow x \circ e = e \circ x = x$.
2. Для всякого элемента $x \in G$ существует элемент $y \in G$, такой что $x \circ y = y \circ x = e$. Обычно y обозначают через x^{-1}.
3. Выполняется свойство ассоциативности: $\forall x, y, z \in G \hookrightarrow x \circ (y \circ z) = (x \circ y) \circ z$.

В рамках этого раздела мы будем работать только с конечными группами.

[2] Порядок конечной группы G — количество элементов в группе G. Обозначается через $|G|$.

[3] Пару $(G', \circ)$ называют подгруппой группы $(G, \circ)$, если $(G', \circ)$ — группа и $G' \subseteq G$.

[4] Подгруппа G' группы G называется собственной, если она не совпадает со всей группой G.

x (и обозначают $m = \mathrm{ord}\,(x)$). Более того, элементы $e, x^{(1)}, x^{(2)}, \ldots, x^{(m-1)}$ образуют подгруппу группы G. Такую подгруппу называют *подгруппой, порождённой элементом* x и обозначают $\langle x \rangle$. Элемент x называют *образующим* подгруппы[5] $\langle x \rangle$. Из всего написано ранее следуют три простых факта.

Теорема 3. Пусть $(G, \circ)$ — конечная группа. Если $x \in G$, то $|\langle x \rangle| = \mathrm{ord}\,(x)$.

Следствие 2. Для любого $x \in G$, где G — конечная группа, последовательность $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$ имеет период $m = \mathrm{ord}\,(x)$, то есть $x^{(i)} = x^{(j)}$ тогда и только тогда, когда $i \equiv j \pmod{m}$.

Следующий факт получается из теоремы Лагранжа и теоремы 3.

Следствие 3. Пусть G — конечная группа. Тогда для любого $x \in G$ выполняется $x^{(|G|)} = e$.

4. **Степени элементов в $\mathbb{Z}_n^*$.** Оказывается, что если применить следствие 3 для группы $\mathbb{Z}_n^*$, то получим следующий факт, известный многим со школьных олимпиад.

   Теорема 4. (Эйлер). Если $n > 1$ — целое число, то для любого целого числа x, взаимно простого с n, выполнено
   $$x^{\varphi(n)} \equiv 1 \pmod{n}.$$

   Если n — просто число, то получаем ещё один известный многим со школы факт.

   Теорема 5. (Малая Теорема Ферма). Если $p > 1$ — простое число, то для любого целого числа x, взаимно простого с p, выполнено
   $$x^{p-1} \equiv 1 \pmod{p}.$$

   Теперь предположим, что в группе $\mathbb{Z}_n^*$ существует некоторый элемент g, такой что $\mathrm{ord}\,(g) = |\mathbb{Z}_n^*| = \varphi(n)$. Это означает, что всю группу $\mathbb{Z}_n^*$ можно получить, возводя элемент g в степени $0, 1, 2, \ldots, \varphi(n)-1$, то есть все элементы $\mathbb{Z}_n^*$ являются степенями g. Если такой элемент g существует, то его называют *первообразным корнем по модулю* n. Первообразный корень существует не всегда.

   Теорема 6. В группе $\mathbb{Z}_n^*$ существует первообразный корень тогда и только тогда, когда n равно 2, 4, имеет вид p^k или $2p^k$, где $p > 2$ — простое число, а k — натуральное число.

   Если g — первообразный корень в группе $\mathbb{Z}_n^*$, то для всякого $a \in \mathbb{Z}_n^*$ сущесвтует x, для которого $g^x \equiv a \pmod{n}$. Такое x называют *дискретным логарифмом* или *индексом* элемента $a \in \mathbb{Z}_n^*$ по основанию g и обозначают $\mathrm{ind}_{n,g}\,(a)$. Из теоремы 3 вытекает следующий факт.

   Теорема 7. (О дискретном логарифме) Пусть g — первообразный корень по модулю n. Тогда сравнение $g^x \equiv g^y \pmod{n}$ равносильно сравнению $x \equiv y \pmod{\varphi(n)}$.

   Из этой теоремы видно, что понятие индекса $\mathrm{ind}_{n,g}\,(a)$ определено с точностью до слагаемого, кратного $\varphi(n)$.

5. **Кольцо вычетов по модулю n.** При работе с вычетами не хочется ограничивать себя либо только сложением, либо только умножением (да и зачем?). Напомню, что множество $\mathbb{Z}_n$, операция сложения "$+$" и операция умножения "$\cdot$" образуют коммутативное кольцо с единицей[6]. Грубо говоря, при работе с вычетами по некоторому модулю n можно их складывать и перемножать как обычные числа, причём промежуточные результаты можно заменять на остатки по модулю n. Тем не менее, алгебраический подход очень продуктивен и позволяет получить простые доказательства некоторых утверждений.

[5] Напомню, что группы, порождённые некоторым элементов, называют *циклическими*.

[6] *Кольцом* называют такое множество R, на котором заданы две бинарные операции: $+$ (сложение) и $\times$ (умножение), для которых выполняются следующие свойства для любых $a, b, c \in R$:

1) множество R с операцией $+$ образует *коммутативную* (то есть $\forall a, b \in R \hookrightarrow a + b = b + a$) группу;
2) ассоциативность умножения: $(a \times b) \times c = a \times (b \times c)$;
3) дистрибутивность: $a \times (b + c) = a \times b + a \times c$ и $(b + c) \times a = b \times a + c \times a$.

Кольцо с единицей — это кольцо, в котором есть нейтральный элемент по умножению, обозначаемый обычно через 1: $\forall a \in R \hookleftarrow 1 \times a = a \times 1 = a$. Коммутативное кольцо — это кольцо, у которого операция умножения является коммутативной: $\forall a, b \in R \hookrightarrow a \times b = b \times a$.

6. **Конечные поля.** Как мы помним из курса высшей алгебры конечное поле[7] может иметь размер только p^n, где p — простое число, n — некоторое натуральное число. Для обозначения конечного поля размера p^n используют $GF(p^n)$. Все конечные поля размера $GF(p^n)$ изоморфны факторкольцу $\mathbb{Z}_p[x]$ (кольцо многочленов над полем $\mathbb{Z}_p$) по некоторому неприводимому многочлену $f(x) \in \mathbb{Z}_p[z]$ степени n (такой всегда существует): $GF(p^n) \cong \mathbb{Z}_p[x]_{/(f(x))}$. Например, поле $GF(4)$ можно представить как множество $\{0, 1, \alpha, \alpha+1\}$, где α — корень многочлена x^2+x+1 над $\mathbb{Z}_2$, т.е. $\alpha^2 = -\alpha-1 = \alpha+1$, что стоит учитывать при операциях с элементами этого поля. Грубо говоря, $GF(4)$ строится по $GF(2)$ добавлением нового числа α, которое в квадрате даёт $\alpha+1$. Полезно держать в голове аналогию с полем комплексных чисел: оно строится по полю $\mathbb{R}$ добавлением нового числа i, которое в квадрате даёт -1. Чуть более подробно про конечные поля можно почитать, например, в Википедии.

Модулярная арифметика: сложение и умножение

Многие теоретико-числовые алгоритмы основаны на арифметических операциях по некоторому модулю n. Как анализировать такие алгоритмы? Хотелось бы уметь так же, как мы это делали раньше, измерять временную сложность числом битовых операций на машине Тьюринга, чтобы оставаться в той же модели вычислений, в которой мы находимся и относительно которой мы изучали сложностные классы алгоритмов. Начнём мы с малого: научимся складывать и умножать числа по модулю n в нашей модели вычислений.

Сложение двух чисел a и b по модулю n происходит следующим образом. Мы складываем a и b обычным способом. Так как оба числа лежат в между 0 и $n-1$, то полученная сумма будет лежать между 0 и $2(n-1)$. Дальше нам достаточно сравнить $a+b$ с n и, если $a+b < n$, то мы нашли сумму по модулю n, а если $a+b > n$, то достаточно вычесть n из $a+b$ и мы получим $a+b \mod n$. В результате мы делаем в худшем случае два сложения и одно сравнение. И ту, и другую операции мы умеем выполнять за полиномиальное от $\log n$ время на МТ.

Умножение двух чисел a и b по модулю n делается похожим образом. Сначала мы перемножаем числа a и b, как обычные числа, а потом берём остаток по модулю n. Произведение чисел a и b не превосходит $(n-1)^2$. Двоичная запись числа $(n-1)^2$ занимает не более $2\log_2 n$ бит, поскольку $\log_2(n-1)^2 = 2\log_2(n-1) \leqslant 2\log_2 n$. Остаток по модулю n можно найти алгоритмом деления. И умножение, и деление требуют полиномиального по $\log_2 n$ времени.

Модулярная арифметика: деление. Решение линейных диофантовых уравнений

Деление по модулю n — это решение линейного сравнения $ax \equiv b \pmod n$ относительно x. Такие сравнения ещё называют *линейными диофантовыми уравнениями*. Оказывается, что такие сравнения можно решать при помощи расширенного алгоритма Евклида, о котором мы говорили ранее. Для начала докажем несколько вспомогательных утверждений.

Теорема 8. Для любых положительных целых чисел a, n и d, таких что $d = \text{НОД}\,(a, n)$, выполняется

$$\langle a\rangle = \langle d\rangle = \left\{0, d, 2d, \ldots, \left(\frac{n}{d}-1\right)d\right\}$$

и

$$|\langle a\rangle| = \frac{n}{d}.$$

Доказательство. Алгоритм Extended-Euclid(a, n) вернёт тройку (d, x', y'), для которой $d = \text{НОД}\,(a, n)$ и $ax' + ny' = d$. Тогда $ax' \equiv d \pmod n$ и поэтому $d \in \langle a\rangle$. С другой стороны, d — это делитель a, а значит, $a \in \langle d\rangle$. Следовательно, $\langle a\rangle = \langle d\rangle = \left\{0, d, 2d, \ldots, \left(\frac{n}{d}-1\right)d\right\}$ и $|\langle a\rangle| = \frac{n}{d}$. ☐

[7] *Полем* называют такое множество F, на котором заданы две бинарные операции: $+$ (сложение) и $\times$ (умножение), для которых выполняются следующие свойства:

1) множество F с операцией $+$ образует коммутативную группу с нейтральным 0;
2) множество $F \setminus \{0\}$ (все элементы, кроме нейтрального по сложению) с операцией $\times$ образует коммутативную группу;
3) дистрибутивность: для любых $a, b, c \in F$ выполняется $a \times (b+c) = a \times b + a \times c$ и $(b+c) \times a = b \times a + c \times a$.

Следствие 4. Уравнение $ax \equiv b \pmod n$ разрешимо относительно x тогда и только тогда НОД (a, n) является делителем числа b.

Доказательство. Сравнение $ax \equiv b \pmod n$ имеет решение тогда и только тогда, когда $b \in \langle a \rangle = \langle d \rangle = \left\{0, d, 2d, \ldots, \left(\frac{n}{d} - 1\right) d\right\} \iff d$ является делителем числа b, где $d = \text{НОД}\,(a, n)$. $\square$

Иными словами, «делить число b на число a по модулю n» можно тогда и только тогда, когда НОД (a, n) — это делитель числа b. В частности, если НОД $(a, n) = 1$, то делить на a можно всегда. Но это и понятно: если НОД $(a, n) = 1$, то $a \in \mathbb{Z}_n^* \implies \exists a^{-1} \mod n$. Тем не менее «разделить» на a можно даже и в случае, когда a и n не взаимно просты. Например, сравнение $2x \equiv 4 \pmod 6$ имеет решения $x = 2$ и $x = 5$. Заметим, что в первом случае решение (в $\mathbb{Z}_n$) единственно, а во втором — решений два. Следующий факт даёт ответ на вопрос о числе решений в общем случае.

Следствие 5. Уравнение $ax \equiv b \pmod n$ имеет $d = \text{НОД}\,(a, n)$ различных решений в $\mathbb{Z}_n$ или не имеет их вовсе.

Доказательство. Здесь полезно смотреть на $\mathbb{Z}_n$ как на группу. Вспомним, как мы определяли $\langle a \rangle$: мы рассматривали последовательность[8] $0, a, 2a, 3a, \ldots$. Мы доказали, что эта последовательность имеет период $|\langle a \rangle| = \frac{n}{d}$. Так как $b \in \langle a \rangle$, то вычет b встретится ровно один раз среди первых $\frac{n}{d}$ членов последовательности $0, a, 2a, 3a, \ldots$. В силу периодичности последовательности, элемент b встретится ещё ровно один раз среди вторых $\frac{n}{d}$ элементов указанной последовательности и так далее. Тогда всего он встретится d раз. Каждому вхождению b в последовательность $0, a, 2a, 3a, \ldots, (n-1)a$ соответствует свой вычет x, а значит, всего таких значений x ровно d штук. $\square$

Следующие две теоремы проливают свет на то, как находить решение линейного диофантового уравнения при помощи расширенного алгоритма Евклида.

Теорема 9. Пусть $d = \text{НОД}\,(a, n) = ax' + ny'$, где x' и y' — целые числа (например, они могут быть получены процедурой Extended-Euclid(a, n)). Если $d \mid b$ (d делит b), то число $x_0 = x' \cdot \frac{b}{d} \mod n$ является решением уравнения $ax \equiv b \pmod n$.

Доказательство. Из $d = ax' + ny'$ следует, что $ax' \equiv d \pmod n$, поэтому $ax_0 \equiv ax' \cdot \frac{b}{d} \equiv d \cdot \frac{b}{d} \equiv b \pmod n$. $\square$

Теорема 10. Пусть уравнение $ax \equiv b \pmod n$ разрешимо и x_0 является его решением. Тогда уравнение имеет $d = \text{НОД}\,(a, n)$ решений в $\mathbb{Z}_n$, задаваемых формулой $x_i = x_0 + \frac{i \cdot n}{d}$, где $i = 0, 1, 2, \ldots, d-1$.

Доказательство. Из доказательства следствия 5 мы знаем, что решения сравнения $ax \equiv b \pmod n$ соответствуют d числам из последовательности $0, a, 2a, \ldots, (n-1)a$, причём эти числа расположены с периодом $\frac{n}{d}$. Но это как раз и означает, что все числа вида $x_0 + i \cdot \frac{n}{d}$, где $i = 0, 1, \ldots, d-1$, являются решениями сравнения $ax \equiv b \pmod n$. Кроме того, их ровно d штук, причём они попарно различны по модулю n в силу того, что $x_0 + (d-1) \cdot \frac{n}{d} - x_0 = (d-1) \cdot \frac{n}{d} < n$. Значит, это в точности все решения сравнения. $\square$

Следующая процедура по целым числам a, b и $n > 0$ даёт все решения уравнения $ax \equiv b \pmod n$.

```
procedure Modular-Linear-Equation-Solver(a, b, n)
    (d, x', y') ← Extended-Euclid(a, n)
    if d | b then
        x0 ← x' · b/d  mod n
        for i ← 0 to d − 1 do
            print (x0 + i · n/d)  mod n
        end for
    else
        print «нет решений»
    end if
end procedure
```

[8] В качестве операции "$\circ$"здесь выступает "$+$".

Процедура Modular-Linear-Equation-Solver(a, b, n) выполнит $O(\log n)$ арифметических операций в строке 2 и $O(\text{НОД}\,(a, n))$ операций в остальных строках, то есть всего $O(\log n + \text{НОД}\,(a, n))$. Но если нам достаточно найти все решения, а не печатать их на экран, то число арифметических операций будет $O(\log n)$, а битовых операций будет $O(\log^3 n)$, т.к. процедура Extended-Euclid(a, n) делает $O(\log n)$ делений целых чисел длины $O(\log n)$ с остатком, что делается за $O(\log^2 n)$.

Отметим ещё два важных следствия.

Следствие 6. Пусть $n > 1$. Если $\text{НОД}\,(a, n) = 1$, то уравнение $ax \equiv b \pmod n$ имеет единственное решение в $\mathbb{Z}_n$. В частности, если $b = 1$ и $\text{НОД}\,(a, n) = 1$, то уравнение $ax \equiv 1 \pmod n$ имеет единственное решение в $\mathbb{Z}_n$. При $\text{НОД}\,(a, n) > 1$ уравнение $ax \equiv 1 \pmod n$ решения не имеет.

Итак, деление по модулю n и вычисление обратного по модулю n, как частный случай, требуют $O(\log^3 n)$ битовых операций, то есть так же полиномиальное время.

Решение систем линейных сравнений. Китайская теорема об остатках

Развивая тему предыдущего раздела и намерено отходя немного в сторону от модулярной арифметики, рассмотрим систему линейных сравнений вида:

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ x &\equiv a_2 \pmod{n_2} \\ &\dots \\ x &\equiv a_k \pmod{n_k}, \end{aligned}$$

где $n_1, n_2, \dots, n_k$ — попарно взаимно простые числа. Наша цель — найти такой вычет a по модулю $n = n_1 n_2 \cdot \ldots \cdot n_k$, что a — это решение указанной линейной системы сравнений, то есть $x \equiv a \pmod n$. Китайская теорема об остатках утверждает, что $\mathbb{Z}_n$ устроено как произведение колец вычетов $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \times \ldots \times \mathbb{Z}_{n_k}$ (с покомпонентным сложением и умножением). Это соответствие полезно с алгоритмической точки зрения, так как бывает проще выполнить операции во всех множествах $\mathbb{Z}_{n_i}$, чем в кольце $\mathbb{Z}_n$.

Теорема 11. (Китайская теорема об остатках). Пусть $n = n_1 n_2 \cdot \ldots \cdot n_k$, причём числа $n_1, n_2, \dots, n_k$ попарно взаимно просты. Рассмотрим соответствие

$$a \leftrightarrow (a_1, a_2, \dots, a_k),$$

где $a \in \mathbb{Z}_n, a_i \in \mathbb{Z}_{n_i}$ и $a_i \equiv a \pmod{n_i}$ при $i = 1, 2, \dots, k$. Тогда данное соответствие являтся взаимно однозначным между $\mathbb{Z}_n$ и $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \times \ldots \times \mathbb{Z}_{n_k}$. При этом операциям сложения, вычитания и умножения в $\mathbb{Z}_n$ соответствуют покомпонентные операции над k-элементными кортежами: если

$$a \leftrightarrow (a_1, a_2, \dots, a_k)$$

и

$$b \leftrightarrow (b_1, b_2, \dots, b_k),$$

то

$$\begin{aligned} (a+b) \mod n &\leftrightarrow ((a_1+b_1) \mod n_1, \dots, (a_k+b_k) \mod n_k)\,, \\ (ab) \mod n &\leftrightarrow ((a_1 b_1) \mod n_1, \dots, (a_k b_k) \mod n_k)\,. \end{aligned}$$

Доказательство. Начнём с того, что отметим, что отображение из $\mathbb{Z}_n$ в декартово произведение задано корректно: если два числа сравнимы по модулю n, то их разность кратна n, и потому эти числа дают одинаковые остатки при делении на любое из n_i, так как n_i — делитель числа n.

Построим обратное отображение. Положим $m_i = \frac{n}{n_i}$ для всех n_i. Отсюда следует, что $m_i \equiv 0 \pmod{n_j}$ при $i \neq j$. Положим

$$c_i = m_i \left(m_i^{-1} \mod n_i\right)$$

для всех $i = 1, 2, \dots, k$. Тогда $c_i \equiv 1 \pmod{n_i}$ и $c_i \equiv 0 \pmod{n_j}$ при $j \neq i$, и числу c_i поэтому соответствует набор с одной единицей на i-м месте:

$$c_i \leftrightarrow (0, 0, \dots, 0, \underset{i}{1}, 0, \dots, 0).$$

Тем самым, положив
$$a \equiv (a_1c_1 + a_2c_2 + \ldots + a_kc_k) \pmod n,$$
мы получим число соответствующее набору $(a_1, a_2, \ldots, a_k)$. Следовательно, для каждого набора можно найти соответствующий элемент. Далее заметим, что если a и a' дают одинаковые остатки при делении на все n_i, то $a - a'$ делится на все n_i, а значит, и на n в силу взаимной простоты чисел n_i. Следовательно, отображение взаимно однозначно. □

Следствие 7. Если $n_1, n_2, \ldots, n_k$ попарно взаимно просты и $n = n_1n_2 \ldots n_k$, то система сравнений
$$x \equiv a_i \pmod{n_i}$$
относительно x (где $i = 1, \ldots, k$) имеет единственное решение по модулю n.

Следствие 8. Если $n_1, n_2, \ldots, n_k$ попарно взаимно просты, $n = n_1n_2 \ldots n_k$ и x и a — целые числа, то свойство
$$x \equiv a \pmod n$$
равносильно выполнению сравнений
$$x \equiv a \pmod{n_i}$$
при всех $i = 1, \ldots, k$.

Пример. Рассмотрим систему сравнений
$$\begin{aligned} a &\equiv 2 \pmod 5 \\ a &\equiv 3 \pmod{13}. \end{aligned}$$
Здесь $a_1 = 2, a_2 = 3, n_1 = m_2 = 5, n_2 = m_1 = 13, n = 65$, если сохранять обозначения теоремы 11. Чтобы восстановить a по модулю $n = 65$, нам нужно найти числа c_i. Так как $13^{-1} \equiv 2 \pmod 5$ и $5^{-1} \equiv 8 \pmod{13}$ (как мы выяснили в предыдущем разделе, при помощи расширенного алгоритма Евклида мы можем находить обратный элемент (если он существует) за $O(\log^3 n)$ битовых операций), мы получаем
$$\begin{aligned} c_1 &= 13 \cdot (2 \mod 5) = 26 \\ c_2 &= 5 \cdot (8 \mod 13) = 40, \end{aligned}$$
откуда
$$\begin{aligned} a &\equiv 2 \cdot 26 + 3 \cdot 40 \pmod{65} \\ &\equiv 52 + 120 \pmod{65} \\ &\equiv 42 \pmod{65}. \end{aligned}$$

Модулярная арифметика: быстрое возведение в степень

В работе теоретико-числовых алгоритмов часто нужно уметь возводить некоторое число в степень по некоторому модулю n. Причём хочется делать это за полиномиальное время. На первом семинаре мы касались алгоритма быстрого возведения в степень. Напомню основные моменты. Пусть мы хотим вычислить a^m. Для этого представим число m в двоичной системе счисления: $m = (\overline{m_km_{k-1}\ldots m_0})_2 = m_k \cdot 2^k + m_{k-1} \cdot 2^{k-1} + \ldots + m_1 \cdot 2 + m_0$, где $m_i \in \{0, 1\}$. Тогда
$$a^m = a^{((\ldots((m_k \cdot 2 + m_{k-1}) \cdot 2 + m_{k-2}) \cdot 2 \ldots) \cdot 2 + m_1) \cdot 2 + m_0} = ((\ldots(((a^{m_k})^2 \cdot a^{m_{k-1}})^2 \ldots)^2 \cdot a^{m_1})^2 \cdot a^{m_0}.$$
Тогда последовательность действий следующая: изначально у нас текущее число равно a (что соответствует числу $a^{m_k} = a$) и $i = k$; затем мы возводим текущее число в квадрат и если $m_{i-1} = 1$, то домножаем результат на a, а если $m_i = 0$, то оставляем как есть; уменьшаем счётчик: $i := i - 1$. При таком подходе происходит $O(\log m)$ умножений при вычислении a^m.

Быстрое возведение в степень по модулю n основано на той же самой идее, но теперь после каждого умножения нужно ещё делить результат на n, чтобы найти остаток по модулю n. Алгоритм приведён ниже.

```
procedure Modular-Exponentiation(a, m, n)
    c ← 0
    d ← 1
    пусть (m_k m_{k-1} ... m_0)_2 — двоичная запись числа m
    for i ← k downto 0 do
        c ← 2c
        d ← (d · d) mod n
        if b_i = 1 then
            c ← c + 1
            c ← (d · a) mod n
        end if
    end for
    return d
end procedure
```

Если a, m и n имеют двоичную запись длины не более β, то число арифметических операций есть $O(\beta)$, а число битовых — $O(\beta^3)$.

Квадратичный вычет. Символ Лежандра. Символ Якоби.

Определение. Целое число a называется *квадратичным вычетом* по модулю m, если разрешимо сравнение $x^2 \equiv a \pmod m$. В противном случае число a называется *квадратичным невычетом.*

Теорема 12. (Критерий Эйлера). Пусть p — простое нечётное число. Число a, взаимно простое с p, является квадратичным вычетом по модулю p тогда и только тогда, когда

$$a^{\frac{p-1}{2}} \equiv 1 \pmod p$$

и является квадратичным невычетом тогда и только тогда, когда

$$a^{\frac{p-1}{2}} \equiv -1 \pmod p.$$

Для случая квадратичных вычетов по простому модулю Лежандр придумал красивое обозначение.

Определение. Пусть p — нечётное простое число и a — целое число. Тогда выражение $\left(\frac{a}{p}\right)$ называется *символом Лежандра* и определяется следующим образом:

- $\left(\frac{a}{p}\right) = 0$, если a делится на p;
- $\left(\frac{a}{p}\right) = 1$, если a — квадратичный вычет по модулю p;
- $\left(\frac{a}{p}\right) = -1$, если a — квадратичный невычет по модулю p.

Перечислим некоторые свойства квадратичных вычетов.

1. **Мультипликативность.** $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right) \cdot \left(\frac{b}{p}\right)$.
2. **Периодичность.** Если $a \equiv b \pmod p$, то $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$.
3. $\left(\frac{1}{p}\right) = 1$.
4. $\left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}}$.
5. $\left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}}$.

6. **Квадратичный закон взаимности.** $\left(\frac{p}{q}\right) = (-1)^{\frac{p-1}{2}\cdot\frac{q-1}{2}}\left(\frac{q}{p}\right)$, где p и q — неравные нечётные простые числа.
7. **Формула Эйлера.** $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod p$.

На первый взгляд кажется, что символ Лежандра оптимально вычислять при помощи быстрого возведения в степень по модулю. Такая процедура потребует $O(\log^3 p)$ битовых операций. Однако если ввести в рассмотрение более общую конструкцию, называемую *символом Якоби*, то символ Лежандра можно будет вычислять за $O(\log^2 n)$.

Определение. Пусть P — нечётное число, большее единицы и $P = p_1 p_2 \cdot \ldots \cdot p_k$ — его разложение на простые множители (среди $p_1, \ldots, p_k$ могут быть и равные). Тогда для произвольного целого числа a *символ Якоби* определяется равенством

$$\left(\frac{a}{P}\right) = \left(\frac{a}{p_1}\right)\left(\frac{a}{p_2}\right)\cdot\ldots\cdot\left(\frac{a}{p_k}\right),$$

где $\left(\frac{a}{p_i}\right)$ — символы Лежандра. Кроме того, по определению будем считать, что $\left(\frac{a}{1}\right) = 1$ для всех a.

Перечислим некоторые свойства символа Якоби.

1. **Мультипликативность.** $\left(\frac{ab}{P}\right) = \left(\frac{a}{P}\right)\cdot\left(\frac{b}{P}\right)$.
2. **Периодичность.** Если $a \equiv b \pmod P$, то $\left(\frac{a}{P}\right) = \left(\frac{b}{P}\right)$.
3. $\left(\frac{1}{P}\right) = 1$.
4. $\left(\frac{-1}{P}\right) = (-1)^{\frac{P-1}{2}}$.
5. $\left(\frac{2}{P}\right) = (-1)^{\frac{P^2-1}{8}}$.
6. **Аналог квадратичного закона взаимности.** $\left(\frac{P}{Q}\right) = (-1)^{\frac{P-1}{2}\cdot\frac{Q-1}{2}}\left(\frac{Q}{P}\right)$, где P и Q — нечётные взаимно простые числа.

Пример (вычисление символа Лежандра при помощи символа Якоби). Из аналога квадратичного закона взаимности:

$$\left(\frac{219}{383}\right) = -\left(\frac{383}{219}\right) = -\left(\frac{164}{219}\right) = -\underbrace{\left(\frac{4}{219}\right)}_{1}\left(\frac{41}{219}\right) = -\left(\frac{219}{41}\right) = -\left(\frac{14}{41}\right) - \left(\frac{2}{41}\right)\left(\frac{7}{41}\right) = -\left(\frac{7}{41}\right) = -\left(\frac{41}{7}\right) = -\left(\frac{-1}{7}\right) = 1.$$

Вычисление рекуррент по простому модулю

В этом разделе мы немного затронем вопрос, как вычислять рекурренту по простому модулю. Более подробно вы столкнётесь с этим вопросом в задании.

Упражнение. Пусть F_k — k-е число Фибоначчи. Рассмотрим новую последовательность: $F_{k,p} = F_k \mod p$ — последовательность остатков от деления чисел Фибоначчи на простое число p. Пусть $p = 11$. Предложить алгоритм вычисления $F_{k,p}$, который делает полиномиальное по $\log k$ число арифметических операций с числами, не превышающими p^2 (эти ограничения вполне естественны, если на вход мы получаем 2 двоичных числа k и p, а нужно за полином вычислить $F_{k,p}$).

Решение. Первое, что приходит в голову — просто посчитать k-число Фибоначчи, используя рекурренту, а потом взять остаток от деления на p. У этого подхода сразу видим проблему — полученные числа в ходе вычислений могут получаться больше, чем p^2. Исправим это: заметим, что $F_{k,p} = (F_{k-1,p} + F_{k-2,p}) \mod p$. Используя эту формулу, мы гарантируем, что мы не будем проделывать арифметические операции с числами, превышающими $2p$, а значит, и p^2. Однако у этого подхода есть другой недостаток (который, кстати говоря, был и у первоначального подхода) — количество арифметических операций для вычисления $F_{k,p}$ таким способом равно $O(k)$, что экспоненциально по $\log_2 k$. Что же делать?

Вспоминаем, что для чисел Фибоначчи есть явная формула, которая называется формулой Бине: $F_k = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^k - \left(\frac{1-\sqrt{5}}{2}\right)^k}{\sqrt{5}}$. Эту формулу можно, например, вывести при помощи характеристического многочлена и начальных условий (а можно даже просто по индукции доказать, что, конечно, чуть более трудрёмко и не даёт ответа, как эта формула получена). Забудем на секунду, что в этой формуле присутствуют иррациональные числа и применим алгоритм быстрого возведения в степень для подсчёта степеней в числителе. Тогда нам потребуется сделать $O(\log k)$ арифметических операций с иррациональными числами, чтобы найти $F_{k,p}$. Формула Бине гарантирует, что в результате мы получим целое число, а поэтому в конце останется просто найти остаток при делении на $p = 11$. Проблемы тут две: нужно делать арифметические операции с иррациональными числами и числа, которые будут получаться в ходе вычислений, могут оказаться больше p^2 (причём не просто больше, чем p^2, а больше, скажем, чем $\left(\frac{1+\sqrt{5}}{2}\right)^k$, если k достаточно большое, а такое число требует $\Omega(k)$ бит, чтобы его записать). Хочется сделать примерно то же самое, что мы сделали в первом варианте решения этой задачи, когда боролись с тем, что получаемые числа могут быть слишком большими, а именно, хочется как-то научиться работать с вычетами, вместо иррациональных чисел, которые к тому же могут быть большими. Что же делать?×2

Не зря мы ввели понятие квадратичного вычета. Заметим, что $4^2 = 16 = 5 \mod 11$, то есть 5 — квадратичный вычет по модулю 11. Заметим, что если в формуле $F_k = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^k - \left(\frac{1-\sqrt{5}}{2}\right)^k}{\sqrt{5}} = \sqrt{5}\frac{\left(\frac{1+\sqrt{5}}{2}\right)^k - \left(\frac{1-\sqrt{5}}{2}\right)^k}{5}$ раскрыть скобки по биному Ньютона, а затем привести все подобные слагаемые, то получится целое число, ведь в левой части равенства написано целое число. Это означает, что все нечётные степени $\sqrt{5}$ после упрощений, использующих $\left(\sqrt{5}\right)^2 = 5$, в сумме дадут $0 \cdot \sqrt{5} = 0$. С таким же успехом мы могли бы вместо $\sqrt{5}$ подставить некоторое число x (например, $x = 4$) раскрыть скобки, учесть всюду, что $x^2 = 5$, и получить выражение, в котором x уже не встречается. Следовательно, мы можем в исходную формулу вместо x подставить 4, а дальше воспользоваться стандартными приёмами модулярной арифметики, которые мы обсудили ранее (то есть после каждой арифметической операции брать остаток от деления на $p = 11$, а также, числа возводить в степень при помощи быстрого возведения в степень по модулю, что мы уже обсудили ранее). Учтём также, что $5^{-1} = 9 \mod 11$ и $2^{-1} = 6 \mod 11$, и получим

$$F_{k,p} = 9 \cdot 4 \left((6(1+4))^k - (6(1-4))^k\right) \mod 11 = 3\left(8^k - 4^k\right) \mod 11.$$

Последнее выражение можно вычислить за $O(\log k)$ арифметических операций с числами, не превосходящими $11^2 = 121$ при помощи быстрого возведения в степень по модулю.

Упражнение. Пусть F_k — k-е число Фибоначчи. Рассмотрим новую последовательность: $F_{k,p} = F_k \mod p$ — последовательность остатков от деления чисел Фибоначчи на простое число p. Пусть $p = 17$. Предложить алгоритм вычисления $F_{k,p}$, полиномиальный по $\log k$.

Решение. Заметим, что в прошлый раз нам крупно повезло, что 5 оказалось квадратичным вычетом по модулю $p = 11$. По модулю $p = 17$ число 5 является квадратичным невычетом, поэтому «просто извлечь корень» уже не получится. Что делать?×3

Мы не случайно при решении прошлого упражнения заметили, что вместо $\sqrt{5}$ можно подставить некоторое x, такое что $x^2 = 5$. Что это напоминает? А напоминает это просто работу в поле $\mathbb{Z}_{17}[x]_{/(x^2-5)}$ (отметим, что здесь очень важно, что мы знаем, что 5 — квадратичный невычет, ибо мы пользуемся неприводимостью многочлена x^2-5 над полем $\mathbb{Z}_{17}$). Действительно, если мы заменим $\sqrt{5}$ на x, учтём, что $2^{-1} = 9 \mod 17$ и $5^{-1} = 7 \mod 17$ (а значит, эти равенства верны и в $\mathbb{Z}_{17}[x]_{/(x^2-5)}$), и рассмотрим получившееся выражение

$$F_{k,p} = 7x\left((9+9x))^k - (9-9x))^2\right)$$

как элемент $\mathbb{Z}_{17}[x]_{/(x^2-5)}$, то получим (после всех преобразований раскрытия скобок, учёта $x^2 = 5$ и того факта, что коэффициенты — это вычеты по модулю 17) в итоге, что $F_{k,p}$ — это просто некоторый вычет по модулю 17. Таким образом, мы можем делать все преобразования в поле $\mathbb{Z}_{17}[x]_{/(x^2-5)}$, чтобы получить правильный ответ. Кроме того, стоит учесть, что порядок мультипликативной группы поля $\mathbb{Z}_{17}[x]_{/(x^2-5)}$ равен $17^2 - 1 = 288$ (17 вариантов выбрать первый коэффициент, 17 вариантов выбрать второй коэффициент, но нужно исключить элемент 0), а значит, $(9 \pm 9x)^{288} = 1$ в поле $\mathbb{Z}_{17}[x]_{/(x^2-5)}$. Следовательно, $F_{k,p} = F_{k \mod 288,p}$ (операция взятия остатка числа по модулю 288 делается за полином). Далее можно сказать, что достаточно посчитать для данного фиксированного $p = 17$ первые 288 элементов $F_{k,p}$ в поле $\mathbb{Z}_{17}[x]_{/(x^2-5)}$. Такой алгоритм остаётся полиномиальным по k. Однако можно поступить чуть более хитро, если k очень большое число и если мы знаем разложение 288 на простые множители. Из КТО мы знаем, что $\mathbb{Z}_{288} \cong \mathbb{Z}_9 \times \mathbb{Z}_{32}$. Допустим $k = 2008^{10^{100000000}}$. Понятно, что двоичная запись такого k содержит больше $10^{100000000}$ и привидённый нами алгоритм остаётся полиномиальным относительно этого безумно большого числа. Однако, такие числа можно эффективно задать, например, указав только показатели и основания степеней и порядок, в котором нужно возвести эти числа в степень (например, передать числа $2008, 10, 100000000$ с какими-нибудь специальными метками, поясняющими, что нужно делать). И вот в некоторых специальных случаях это может заметно упростить вычисление $F_{k,p}$. Заметим, что $2008 = 280 \mod 288$. Кроме того, из изоморфизма, установленного в теореме 11, получаем, что 280 при таком изоморфизме переходит в $(1, -8) \in \mathbb{Z}_9 \times \mathbb{Z}_{32}$ (будем сокращённо писать $288 \leftrightarrow (1, -8)$). Тогда $280^2 \leftrightarrow (1, -8)^2 = (1, 64) = (1, 0) \leftrightarrow 64$. Отсюда следует, что 280 в любой чётной степени тоже даёт 64 по модулю 288: $280^{2m} \leftrightarrow (1, 0)^m = (1, 0) \leftrightarrow 64$. Отсюда следует, что $F_{k,p} = F_{64,p}$, причём мы обошлись без взятия остатка k по модулю 288. Дальше остаётся посчитать $(9 \pm 9x)^{64}$ при помощи быстрого возведения в степень, например. Опустим выкладки, каждый их и сам может проделать: в результате получится $F_{k,p} = 13$.

Схемы с открытым ключом. Протокол Диффи-Хеллмана*

С современной точки зрения задачи проверки простоты и факторизации гипотетически принципиально различны, и предположительно для последней вообще нельзя построить эффективного алгоритма[9]. Грубо говоря, ничего лучше, чем решето Эратосфена и придумать нельзя, хотя, конечно, во всяком переборном алгоритме крайне важны константы; прочитать об этих процедурах можно, например, **[Кормен 1, §33.9]**. Задача факторизации имеет многочисленные криптографические приложения, где фактически используется в качестве примитива, одно из которых мы сейчас и рассмотрим.

Начнем с модельного примера выбора секретного ключа с использованием публичных каналов обмена информацией. Могут ли Алиса и Боб[10] договориться, например, *по телефону (который может прослушиваться)* о некотором *секретном ключе*, который они будут использовать в дальнейшем?

Сначала рассмотрим следующую **СХЕМУ 1**.

(i) Алиса и Боб выбирают большое простое число p и некоторое $1 < g < p$. Эта информация публичная и может быть перехвачена "нехорошим человеком" Евой.

(ii) Затем Алиса *секретно* выбирает число n, а Боб *секретно* выбирает число m.

(iii) Затем Алиса открыто передает Бобу число $A = ng \pmod p$, а Боб открыто сообщает Алисе число $B = mg \pmod p$.

(iv) Теперь оба могут легко вычислить "секретный ключ" $s = nmg \pmod p$.

Легко убедиться, что можно быстро (за полиномиальное время при помощи расширенного алгоритма Евклида находим n, зная A и g; после этого остаётся только перемножить два числа по модулю) найти s, зная p, g, A, B. В криптографических терминах это значит, что **СХЕМА 1** является ненадежной[11].

Рассмотрим также **СХЕМУ 2** выбора секретного ключа или схему обмена ключами Диффи и Хеллмана.[12] Хронологически эта схема предшествовала (и являлась мотивацией для) схемы RSA (см. ниже). Она очень похожа на первый вариант.

(i) Алиса и Боб выбирают большое простое число p и g — некоторый первообразный корень $\pmod p$. Эта информация публичная и может быть перехвачена "нехорошим человеком" Евой.

[9] Следует отметить, что если изменить правила пересчета вероятности на те, которые (экспериментально) выполняются в микромире, и рассмотреть т. н. квантовые алгоритмы, то задачу факторизации удается быстро решить. Рекомендуем прочитать об этом в книге **[К-Ш-В]** (*Китаев А., Шень А., Вялый М.* Классические и квантовые вычисления. М.: МНЦМО-ЧеРо, 1999.), хотя бы для того, чтобы правильно реагировать на многочисленные досужие рассуждения на эту тему, которые периодически появляются даже в таблоидах.

[10] Это стандартные персонажи криптографических протоколов. Им обычно противостоит "нехороший человек" Ева.

[11] На самом деле, свойство криптографической ненадежности гораздо слабее, чем существование полиномиального алгоритма, поскольку Алису и Боба также не устроит результат, когда Ева может достаточно **часто** дешифровывать сообщения. Обсуждению этих вопросов посвящена обширная литература.

[12] Предложена в статье (Diffie Whitfield; Hellman Martin E. New directions in cryptography. IEEE Trans. Information Theory IT-22 (1976), N 6, 644–654), где было определено само понятие криптографии с открытым ключом.

(ii) Затем Алиса *секретно* выбирает число n, а Боб *секретно* выбирает число m.

(iii) Затем Алиса открыто передает Бобу число $g^n \pmod p$, а Боб открыто сообщает Алисе число $g^m \pmod p$.

(iv) Теперь оба могут легко вычислить "секретный ключ" $s = g^{mn} = (g^n)^m = (g^m)^n \pmod p$.

В настоящее время СХЕМА 2 считается надежной, поскольку Еве для дешифровки ключа предположительно нужно уметь быстро вычислять *дискретный логарифм*, т. е. решать уравнение $g^x = a \pmod p$, чего пока никто не умеет[13].

Криптосистема RSA*

Но если Ева может не только подслушивать, но и выступать активным агентом, то дела Алисы и Боба осложняются. Например, если Ева может перехватывать и подменять сообщения, то она может послать Бобу *от имени Алисы* некоторое $g^t \pmod p$ и получить секретный ключ $g^{tm} \pmod p$ для дешифровки сообщений Боба. Такую же операцию Ева может провести и в отношении Алисы. Таким образом, возникает проблема идентификации участников. Эта задача решается, например, в очень распространенной схеме шифрования с *открытым ключом* **RSA**[14]. Состоит она в следующем.

(i) Боб выбирает *модуль*– число $n = pq$, равное произведению двух больших простых чисел.

(ii) Потом Боб выбирает *секретный ключ*– d (он известен только ему).

(iii) Затем Боб вычисляет *открытый ключ* $e = d^{-1} \pmod{(p-1)(q-1)}$.

(vi) Информация о (e, n)— публичная (например, Боб помещает ее в сеть).

(v) Если Алиса хочет послать секретное сообщение x Бобу, то она проводит шифровку (e{ncrypts}) $x \to e(x) = x^e \pmod n$ и посылает $e(x)$ *по открытому каналу.*

(vi) Боб легко дешифрует d{ecrypts}) сообщение с помощью секретного ключа $d(e(x)) \to (e(x))^d \pmod n = x \pmod n$.

Считается, что "нехороший человек" Ева не сможет прочитать сообщение, поскольку для этого ей нужно найти делители n.

Схема RSA позволяет также создавать защищенные электронные подписи. Пусть открытый ключ Боба (e, n). Если он хочет электронно "подписать" свое сообщение A, то должен послать сообщение $B = A^{\text{секр. ключ Боба}} \pmod n$ (для того чтобы идентифицировать "подпись" Боба его сообщение нужно преобразовать $B^e \pmod n$).

Резюме. *RSA — это асимметричная схема (для шифрования и дешифрования применяются разные процедуры), которая характеризуется следующими параметрами: $n = pq$, где p, q — различные большие простые числа; открытый ключ (e, n), где e взаимно просто с $\varphi(n) = (p-1)(q-1)$; секретный ключ (d, n), где d обратно к e по модулю $\varphi(n)$. Пусть M — остаток по модулю n. Тогда процедура шифрования сообщения M выглядит так: $P(M) = M^e \mod n$, а процедура дешифрования сообщения C выглядит так: $S(C) = C^d \mod n$. Криптоустойчивость схемы основана на предполагаемой сложности задачи* **факторизации** *(число n всем известно, но не понятно, как по нему вычислить $\varphi(n)$, если нам не известно разложение n на множители).*

Вероятностные тесты, проверяющие простоту числа. Тест Ферма

Мы увидели, что в криптосистеме RSA очень важно уметь генерировать большие простые числа. Возникает резонный вопрос: как это делать? Одна из простых идей, что приходят на ум: взять большое случайное число и проверить, является ли оно простым. Если нет, то взять новое случайное большое число и так далее. Как мы понимаем, такой способ пригоден только в том случае, если простые числа не слишком редки. К счастью, это действительно так.

[13]Хотя к утверждению о надежности нужно относится с известной долей скепсиса. Во всяком случае на сайтах, посвященных криптографии, буквально заклинают не использовать схемы "из книжек", поскольку они довольно успешно взламываются.

Кроме того, обратите внимание, что здесь мы, как и многие авторы криптографических текстов, слегка передернули. Формально перед Евой стоит не задача вычисления дискретного логарифма, а проблема вычисления ключа по известным g^n и g^m, а эта задача может оказаться проще, чем вычисление дискретного логарифма.

[14]Названной в честь авторов R{ivest}- S{hamir}- A{dleman}. По непроверенным данным RSA прочно удерживает первое место в мире по числу проданных патентов.

Теорема 13. (асимптотический закон распределения простых чисел). Пусть $\pi(n)$ — это количество простых чисел, не превосходящих n. Тогда

$$\lim_{n\to\infty} \pi(n) \cdot \frac{\ln n}{n} = 1,$$

то есть простых чисел на отрезке $[1, n]$ асимптотически столько: $\frac{n}{\ln n}$.

Доказательство этой теоремы можно прочитать, например, тут или здесь.

С тем, что простые числа достаточно часто встречаются, мы разобрались (или приняли на веру). Осталось научиться быстро проверять, что число простое. И тут на помощь приходят вероятностные алгоритмы.

Первый такой вероятностный алгоритм, который мы рассмотрим, называется **тестом Ферма**. А основан он на простом факте: нечётное целое число n является простым тогда и только тогда, когда для всех a, взаимно простых с n, выполняется сравнение Ферма: $a^{n-1} \equiv 1 \pmod n$. Соответственно, тест Ферма — это вероятностный алгоритм типа Монте-Карло, то есть работающий, пока не получит противоречие, либо не превысит допустимое число итераций, который на каждой итерации генерирует случайное целое число a из отрезка $[1, n-1]$ (каждый из вычетов может появиться с одной и той же вероятностью). Если на какой-то итерации $a^{n-1} \not\equiv 1 \pmod n$, то алгоритм останавливается и выдаёт ответ "нет". В противном случае, он дальше генерирует новые числа a, пока не получит противоречие со сравнением Ферма, либо не отработает заданное число итераций.

Определение. Пусть n — нечётное простое число и $a \in \{1, 2, \ldots, n-1\}$. Число a называют *свидетелем* Ферма, если $a^{n-1} \not\equiv 1 \pmod n$, и *лжесвидетелем Ферма*, если $a^{n-1} \equiv 1 \pmod n$. Кроме того, a называют *тривиальным свидетелем Ферма*, если НОД$(a, n) > 1$ (отсюда, конечно же, следует, что $a^{n-1} \not\equiv 1 \pmod n$), и *нетривиальным свидетелем Ферма*, если $a^{n-1} \not\equiv 1 \pmod n$ и НОД$(a, n) = 1$.

Следующую теорему вам предлагается доказать в домашнем задании.

Теорема 14. Пусть n — нечётное составное число. Если n имеет нетривиальный свидетель Ферма, то

$$\frac{\left|\left\{a \in \{1, 2, \ldots, n-1\} \mid a^{n-1} \not\equiv 1 \pmod n\right\}\right|}{n-1} > \frac{1}{2}.$$

Данная теорема означает, что если число n — нечётное составное и имеет хотя бы один нетривиальный свидетель Ферма, то вероятность того, что тест Ферма через t шагов на входе n не найдёт свидетеля Ферма, меньше чем $\frac{1}{2^t}$. Иными словами, на таком входе n не более чем за t шагов тест Ферма определит, что это число составное, с вероятностью большей $1 - \frac{1}{2^t}$, что является очень близким к единице числом даже для $t = 10$ (вероятность получается примерно $99,9\%$).

Однако не всё так хорошо для данной вероятностной процедуры. Оговорка о том, что у числа n существует нетривиальный свидетель Ферма, здесь не случайна. Дело в том, что существуют *числа Кармайкла*, у которых все свидетели Ферма тривиальны. Самое маленькое число Кармайкла равно 561. Более того, доказано, что чисел Кармайкла бесконечно много, что является серьёзным недостатком теста Ферма.

Тест Соловея-Штрассена

В этом разделе мы рассмотрим другую вероятностную процедуру под названием *тест Соловея-Штрассена*, для которой уже нет таких проблем, которые были связаны с существованием чисел Кармайкла для теста Ферма.

Оказывается формулу Эйлера можно превратить в критерий простоты числа, если перейти от символа Лежандра к символу Якоби.

Теорема 15. Нечётное число $n > 1$ является простым тогда и только тогда, когда для всякого $a \in \{1, \ldots, n-1\}$ выполняется *сравнение Эйлера* $a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \pmod n$.

Именно это сравнение и проверяет тест Соловея-Штрассена: принцип работы абсоютно аналогичен, но проверяется сравнение Эйлера, а не сравнение Штрассена.

Определение. Пусть $n > 1$ — нечётное число, $a \in \{1, 2, \ldots, n-1\}$. Число a называют *свидетелем Эйлера*, если НОД$\,(a, n) > 1$ или НОД$\,(a, n) = 1$ и $a^{\frac{n-1}{2}} \not\equiv \left(\frac{a}{n}\right) \pmod n$, и *лжесвидетелем Эйлера*, если НОД$\,(a, n) = 1$ и $a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \pmod n$.

Теорема 16. (Соловей-Штрассен, 1977 г.) Пусть n — нечётное и составное, тогда

(i) существует число $a \in \{1, 2, \ldots, n-1\}$, такое что НОД$\,(a, n) = 1$ и $a^{\frac{n-1}{2}} \not\equiv \left(\frac{a}{n}\right) \pmod n$;

(ii) выполнено неравенство

$$\frac{\left|\left\{a \in \{1, 2, \ldots, n-1\} \mid \text{НОД}\,(a, n) > 1 \text{ или } \left(\text{НОД}\,(a, n) = 1 \text{ и } a^{\frac{n-1}{2}} \not\equiv \left(\frac{a}{n}\right) \pmod n\right)\right\}\right|}{n-1} > \frac{1}{2}.$$

Данную теорему я предлагаю доказать в качестве дополнительной задачи в задании. Отметим, что тест Соловея-Штрассена на любом нечётном составном числе за не более чем t шагов даст ответ «нет» с вероятностью большей, чем $1 - \frac{1}{2^t}$. При этом исключений в стиле чисел Кармайкла, как было для теста Ферма, для теста Соловея-Штрассена уже нет. Проверка сравнения Эйлера требует $O(\log^3 n)$ операций, как и проверка сравнения Ферма.

Тест Миллера-Рабина

Следующий алгоритм будет основываться тоже на некотором утверждении, которое вытекает из сравнения Ферма и ещё некоторых групповых свойств.

Теорема 17. Пусть $n > 2$ — нечётное число и пусть $n = 2^l k$, где $l \geqslant 1$ и k — нечётное число. Тогда число n является простым тогда и только тогда, когда для любого $a \in \{1, 2, \ldots, n-1\}$ выполнено одно из условий:

(i) $a^k \equiv 1 \pmod n$;

(ii) существует $0 \leqslant i \leqslant l-1$, такое что $a^{2^i k} \equiv -1 \pmod n$.

На этом факте основывается вероятностный тест Миллера-Рабина.

Определение. Пусть $n > 2$ — нечётное число и $a \in \{1, 2, \ldots, n-1\}$. Тогда число a называется *лжесвидетелем Миллера-Рабина*, если последовательность $(a^k, a^{2k}, a^{4k}, \ldots, a^{2^{l-1}k})$ по модулю n равна либо последовательности, у которой на первом месте стоит единица, либо последовательности, у которой на каком-то месте стоит (-1), и *свидетелем Миллера-Рабина* в противном случае.

Оказывается, что верна следующая теорема.

Теорема 18. Пусть $n > 2$ — нечётное составное число. Тогда

$$\frac{\text{количество свидетелей Миллера-Рабина}}{n-1} \geqslant \frac{3}{4}.$$

Другими словами тест Миллера-Рабина не более, чем через t шагов, на числе n, являющимся нечётным составным, выдаёт ответ «нет» с вероятностью не меньше $1 - \frac{1}{4^t}$.