Лекция 7.

2.8. Построение больших простых чисел.

Алгоритмы построения больших простых чисел носят рекурсивный характер. Чтобы построить большое простое число, нужно построить возрастающую последовательность простых чисел. Число нужного размера появляется в конце этой последовательности, члены же её принадлежат последовательности уменьшающихся промежутков, с построения которых начинается алгоритм. Эта конструкция вместе со всеми пояснениями обсуждается в настоящем параграфе.

Сначала мы рассмотрим следующую задачу: Для каждого заданного натурального $N \geqslant 11$ **построить простое число Q, лежащее на промежутке** $2^{N-1} \leqslant Q < 2^N$.

1. Для каждого числа $N \leqslant 18$ требуемое простое число можно найти непосредственно. Например, на промежутке от 2^{17} до 2^{18} их имеется 10749 штук, и каждое составное число этого промежутка делится на какое нибудь простое $p < 2^9 = 512$. Для нахождения простого числа в нужном маленьком промежутке (правый конец не превосходит 2^{18}) можно воспользоваться алгоритмом пробных делений, см. подраздел 2.10.1.

Далее будет найдена последовательность целых чисел a_k, b_k, q_k таких что

$$q_k \text{ - простые}, \quad 2^{a_k} \leqslant q_k < 2^{b_k}, \quad a_k = b_k - 1, \quad 1 \leqslant k \leqslant m, \quad b_m = N.$$

Тогда $Q = q_m$ - искомое простое число.

2. Пусть N - натуральное число, превосходящее 18. Построим последовательность целых чисел, начинающуюся с N, в которой за числом ℓ следует число $\lceil \frac{\ell}{2} \rceil + 1$. Легко видеть, что эта последовательность убывает при $\ell \geqslant 4$. Более того, в ней обязательно встретится целое число из промежутка $11 \leqslant \ell \leqslant 18$. Кроме того, не трудно проверить, что при любом $\ell \geqslant 19$ выполняется неравенство $\lceil \frac{\ell}{2} \rceil + 1 \geqslant 11$.

Например, для $N = 1024$ эта последовательность имеет вид

$$N = 1024, \quad 513, \quad 258, \quad 130, \quad 66, \quad 34, \quad 18.$$

Итак,

с какого бы числа N ни начиналась указанная последовательность, она обязательно содержит единственное число из промежутка $11 \leqslant$

$x \leqslant 18$. *Начиная с этого числа, перенумеруем в обратном порядке все члены последовательности буквами* $b_1, b_2, \ldots, b_m$. *Тогда* $11 \leqslant b_1 \leqslant 18$ *и* $b_m = N$. *Для каждого индекса* k, $1 \leqslant k \leqslant m$, *положим* $a_k = b_k - 1$.

3. Простое число q_1 выбирается так, как это указано в пункте 1, ведь $11 \leqslant b_1 \leqslant 18$ и $2^{a_1} < q_1 < 2^{b_1}$. Например, в случае $b_1 = 11$ на роль q_1 может быть взято любое из 137 простых чисел, лежащих на отрезке от 1024 до 2048, скажем 1933.

Предположим теперь, что $k \geqslant 2$ и простое число q_{k-1} уже построено. Для краткости обозначим

$$\ell = b_k, \quad q = q_{k-1}. \quad \text{Тогда} \quad a_k = \ell - 1, \quad b_{k-1} = \lceil \frac{\ell}{2} \rceil + 1, \quad a_{k-1} = \lceil \frac{\ell}{2} \rceil.$$

Число q_k будет строиться с помощью Следствия (2) из теоремы (7), см. также алгоритм (5). При этом $F = q$, $R = 2t$, где t - некоторое натуральное, оно будет выбираться с помощью испытаний, и

$$c = 2tq + 1 = FR + 1. \tag{2.25}$$

Для того, чтобы число c попало в нужный интервал, т.е. $2^{\ell-1} < c < 2^{\ell}$, на параметр t накладываются условия

$$\frac{2^{\ell-1}}{2q} \leqslant t \leqslant \frac{2^{\ell} - 1}{2q}. \tag{2.26}$$

Действительно, в этих ограничениях имеем

$$c = 2tq + 1 \geqslant 2^{\ell-1} + 1 > 2^{\ell-1} = 2^{a_k}$$

и

$$c = 2tq + 1 < 2^{\ell} = 2^{b_k}.$$

Строгое неравенство выполняется в силу того, что c нечётно, а 2^{ℓ} чётное число. Не трудно проверить, что для любых положительных чисел $u < v$ на интервале $u < t < v$ лежит не менее $v - u - 1$ целых чисел t. Согласно неравенству (2.26) можно утверждать, что количество целых t, для которых число c принадлежит интервалу $2^{\ell-1} < t < 2^{\ell}$ не меньше, чем $\frac{2^{\ell-}}{2q} - 1$. Например, при $b_1 = 11, \ell = b_2 = 19$ и $q = 1933$ на промежутке $2^{18} < c < 2^{19}$ имеется 10 простых чисел вида $2tq + 1$. Самые маленькие из них соответствуют числам $t = 65, 72, 75, 77, 81$. При $t = 77$ имеем $c = 297683$.

4. Как правило, строящиеся большие простые числа должны иметь случайный характер. Поэтому на интервале от $2^{\ell-1}$ до 2^{ℓ} выбирается

случайное число x, а затем от него начанается отсчёт чисел t, соответствующие им числа c проверяются на простоту. Как только попадётся простое c, оно выбирается в качестве следующего простого q_k. Если при очередном t соответствующее число c выйдет за верхний предел заданного промежутка, следующее значение t принимается равным наименьшему возможному значению и вычисления продолжаются.

Выберем псевдослучайное целое число x на промежутке $2^{\ell-1} \leqslant x < 2^{\ell}$ и положим $t = \lceil \frac{x}{2q} \rceil$. Определим также число c равенством (2.25).

Из неравенства $x \geqslant 2^{\ell-1}$ следует (2.25).

5. *Если t оказалось настолько большим, что*

$$2tq + 1 \geqslant 2^{\ell},$$

то полагаем $t = \lceil \frac{2^{\ell-1}}{2q} \rceil$. Если с помощью следствия 2 при $F = q$ и $R = 2t$ выбирая псевдослучайные числа b на промежутке от 2 до $c-2$, удаётся доказать, что число c простое, то выбираем $q_k = c$. Для того, чтобы с помощью следствия 2 можно было доказать простоту числа c, параметры R и F должны быть связаны неравенством $R \leqslant 4F + 2$, которое может быть записано в виде $t \leqslant 2q + 1$. Используя границы для t и q, находим

$$q^2 > 2^{2\lceil \frac{\ell}{2} \rceil} \geqslant 2^{\ell} > 2qt \quad и \quad t < \frac{1}{2}q < 2q + 1,$$

так что следствие 2 применимо.

Если $k = m$, алгоритм завершает свою работу. В противном случае увеличиваем k на единицу и переходим в пункт 3 к следующему промежутку $[a_k, b_k)$.

6. *Если же c оказывается составным, то параметр t увеличивается на 1 и алгоритм переходит в пункт 5.*

Следующий далее алгоритм, по заданным числам L и N строит пары простых чисел P, Q с условиями

$$Q|(P-1), \qquad 2^{L-1} \leqslant P < 2^{L}, \qquad 2^{N-1} \leqslant Q < 2^{N}.$$

Мы ограничимся здесь только формальным описанием его основных моментов.

Алгоритм 7. Даны *натуральные числа L и $N, L > N$.*

Построить *"случайные"простые числа P, Q битовой длины L и N с условием $Q|(P-1)$.*

1. С помощью описанного выше алгоритма построить "случайное" простое число Q с длиной записи N битов.

2. Построить последовательность простых чисел p_k, связанных соотношениями

$$p_k = 2tQp_{k-1} + 1, \quad 1, \ldots, m, \qquad 2^{L-1} \leqslant p_m < 2^L.$$

Для проверки на простоту чисел $c = 2tQp_{k-1} + 1$ использовать следствие 2 с параметрами $F = p_{k-1}$, $R = 2tQ$.

3. Положить $P = p_m$.

2.9. Первообразные корни и дискретное логарифмирование.

Пусть p – простое нечётное число. Множество классов вычетов целых чисел по модулю p составляет поле $\mathbb{F}_p$, а ненулевые элементы этого поля образуют циклическую мультипликативную группу порядка $p-1$. Согласно малой теореме Ферма выполняется сравнение $a^{p-1} \equiv 1 \pmod{p}$. Наименьшее натуральное число d с условием $a^d \equiv 1 \pmod{m}$ называется *показателем* числа a по модулю p. Например, показатель числа 1 по модулю p равен 1, а показатель -1 по нечётному модулю p равен 2. *Целое число a, не делящееся на p, называется первообразным корнем по модулю p, если его показатель равен $p-1$.* Например, показатель числа 5 по модулю 23 равен 22. Мы проверим это позже. Число 5 есть первообразный корень по модулю 23. Его класс вычетов по модулю 23 порождает мультипликативную группу всех ненулевых классов вычетов по модулю 23.

В 1801 году К.Ф. Гаусс опубликовал два доказательства следующей теоремы.

Теорема 8. *Для каждого простого нечетного числа p существуют первообразные корни по модулю p. Количество не сравнимых друг с другом по модулю p первообразных корней равно $\varphi(p-1)$, где $\varphi(n)$ – функция Эйлера.*

Для практического нахождения первообразных корней удобно пользоваться следующим утверждением.

Теорема 9. *Целое число g, не делящееся на простое нечетное p, будет первообразным корнем по модулю p в том и только том случае, если для любого простого числа q, делящего $p-1$, выполняется*

$$g^{\frac{p-1}{q}} \not\equiv 1 \pmod{p}.$$

Доказательство. Обозначим буквой d показатель числа g по модулю p. Разделив $p-1$ на d с остатком, получим

$$p-1 = d\cdot u + v, \qquad 0 \leqslant v < d.$$

Неравенство v>0 в силу малой теоремы Ферма и сравнений

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

противоречит определению d. Значит $v=0$ и $d|(p-1)$. Если $d<p-1$, то найдется такое простое число q, что $d|\frac{p-1}{q}$. Но тогда $g^{\frac{p-1}{q}} = (g^d)^{\frac{p-1}{qd}} \equiv 1 \pmod{p}$, вопреки условию. Значит, $d=p-1$, и это завершает доказательство теоремы. □

Если известны все простые делители числа $p-1$, то проверка условий теоремы 9 выполняется достаточно быстро с помощью алгоритма, изложенного в параграфе 2.3.

Справедливы сравнения по модулю 23

$$5^{11} = 5\cdot 25^5 \equiv 5\cdot 2^5 \equiv -1 \pmod{23}$$

и $5^2 \equiv 2 \pmod{23}$. Применяя теорему 9 к простому числу $p=23$ и $c=5$, заключаем, что 5 есть первообразный корень по модулю 23.

Множество первообразных корней при заданном p достаточно велико, поэтому выбирая числа g случайным образом на промежутке $0<g<p$, можно с большой вероятностью попасть на первообразный корень и доказать это с помощью теоремы 9.

Если g – первообразный корень по модулю p, то числа

$$1, g, g^2, \ldots, g^{p-2} \tag{2.27}$$

имеют разные наименьшие неотрицательные остатки. Действительно, если бы нашлись целые числа $0 \leqslant u < v \leqslant p-2$, для которых g^u и g^v имеют одинаковые остатки, то $g^u \equiv g^v \pmod{p}$. В силу взаимной простоты чисел g и p мы получили бы

$$1 \equiv g^{v-u} \pmod{p}.$$

Но это невозможно, так как $0<v-u<p-1$, а показатель g равен $p-1$.

Множество (2.27) состоит из $p-1$ чисел. Поэтому для каждого целого числа a, не делящегося на p, найдется единственное целое k, удовлетворяющее условиям

$$a \equiv g^k \pmod{p}, \qquad 0 \leqslant k < p-1. \tag{2.28}$$

Число k называют *индексом* a по модулю p при основании g. Это определение и формулируемые ниже свойства индексов напоминают свойства обычных логарифмов действительных чисел. Поэтому иногда в современной литературе индексы называют дискретными логарифмами, а процесс их нахождения – дискретным логарифмированием. Используются также обозначения $\operatorname{ind}_g a$, $Log_g a$. Указание на первообразный корень иногда опускается. Итак, имеем

$$a \equiv g^{\operatorname{ind}_g a} \pmod{p}, \qquad 0 \leqslant \operatorname{ind}_g a < p-1. \tag{2.29}$$

Свойства индексов описывает следующие простые утверждения.

1. Если целые числа a, b не делятся на p, то

$$\operatorname{ind}_g ab \equiv \operatorname{ind}_g a + \operatorname{ind}_g b \pmod{p-1}.$$

2. Если a, b – первообразные корни по модулю p, то для любого числа c, не делящегося на p, выполняется сравнение

$$\operatorname{ind}_b c \equiv \operatorname{ind}_b a \cdot \operatorname{ind}_a c \pmod{p-1}.$$

Сформулируем теперь задачу дискретного логарифмирования по простому нечетному модулю.

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

$$b^x \equiv a \pmod{p}. \tag{2.30}$$

Решение этой задачи очень трудоемко в вычислительном отношении. Не случайно в конце практически всех учебников по элементарной теории чисел приводятся таблицы индексов (дискретных логарифмов).

Лучшие из известных алгоритмов дискретного логарифмирования по простому модулю p, использующие вычисления в полях алгебраических чисел, требуют $O(e^{2(\ln p)^{1/3}(\ln\ln p)^{2/3}})$ арифметических операций. Впрочем, эта оценка условна, ибо опирается на ряд недоказанных, но весьма правдоподобных гипотез теории чисел.

Рекордный по величине простого числа p результат в задаче дискретного логарифмирования был установлен в 2016 году. Группе европейских математиков и программистов удалось сосчитать дискретный логарифм по модулю простого числа, записываемого 232 десятичными цифрами (768 битами). Применённый ими алгоритм требует достаточно глубоких познаний в теории алгебраических чисел, и мы его здесь не объясняем.

Излагаемый здесь метод решения сравнения (2.30) требует $O(p^{1/2}\ln p)$ арифметических операций. Он был придуман в 1962г. А.О.Гельфондом и

в русскоязычной литературе назывется "метод согласования". Аналогичный метод был предложен в 1971г. Д.Шенксом. Его английское название может быть переведено как "метод больших и малых шагов".

Лемма 1. *Пусть p — простое нечетное число и пусть $H = [\sqrt{p}] + 1$. Тогда для каждого целого числа d, $1 \leqslant d < p$, найдутся целые числа u, v, удовлетворяющие условиям*

$$1 \leqslant u, v \leqslant H, \qquad Hu - v = d.$$

Доказательство. Положим

$$u = \left[\frac{d}{H}\right] + 1, \qquad v = Hu - d.$$

Тогда

$$\frac{d}{H} < u \leqslant \frac{d}{H} + 1,$$

откуда видим, что, во–первых,

$$0 < u < \frac{p}{\sqrt{p}} + 1 = \sqrt{p} + 1,$$

а во–вторых, $0 < Hu - d \leqslant H$. Остается воспользоваться тем, что числа u и v целые. □

Алгоритм 8 (Алгоритм Гельфонда)**.** Данные: *Простое число $p \geqslant 3$, первообразный корень g по модулю p, число $b \in \mathbb{Z}$, $p \nmid b$.*
Найти: *Решение сравнения $g^x \equiv b$.*
1. Вычислить $H = [\sqrt{p}] + 1$.
2. Положить $c \equiv g^H \bmod p$, $1 \leqslant c < p$.
3. Составить два набора чисел

$$S_1 = \{c^u \bmod p : 1 \leqslant u \leqslant H\}, \quad S_2 = \{bg^v \bmod p : 1 \leqslant v \leqslant H\}.$$

4. Упорядочить по возрастанию оба набора S_1 и S_2. Найти совпавшие элементы этих наборов, то есть такие числа u, v, для которых

$$c^u \equiv bg^v \pmod{p}. \tag{2.31}$$

5. Положить

$$\operatorname{ind}_g b = Hu - v. \tag{2.32}$$

Решение заданного сравнения имеет вид $x \equiv \operatorname{ind}_g b \pmod{p-1}$.

Пересечение множеств S_1 и S_2 не пусто, так как сравнение (2.31) равносильно представлению (2.32), а последнее, согласно лемме 1, существует всегда.

Вычисления в пунктах 1 и 2 требуют $O(\ln p)$ арифметических операций. Вычисления в пункте 3 требуют $O(H \ln p) = O(\sqrt{p} \ln p)$ арифметических операций. Для упорядочения каждого из множеств S_1, S_2 нужно $O(H \ln H) = O(\sqrt{p} \ln p)$ арифметических операций. Для нахождения одинаковых элементов в упорядоченных множествах S_1, S_2 нужно $O(H) = O(\sqrt{p})$ арифметических операций. Общее же количество операций в алгоритме Гельфонда равно $O(\sqrt{p} \ln p)$.

Пример. Решить сравнение

$$2^x \equiv 23 \pmod{37}. \tag{2.33}$$

Решение. Так как $36 = 2^2 \cdot 3^2$ и

$$2^{18} \equiv -1 \not\equiv 1 \pmod{37}, \qquad 2^{12} \equiv 26 \not\equiv 1 \pmod{37},$$

то 2 - первообразный корень по модулю 37 и, значит данное сравнение разрешимо. В соответствии с алгоритмом вычисляем $H = [\sqrt{37}]+1 = 7$ и $c = 2^7 \equiv 17 \pmod{37}$. Множества S_1 и S_2 состоят из чисел $c^n \pmod{37}$, и $b \cdot a^n \pmod{37}$ при $n = 1, 2, \ldots, 7$, т.е.

$$S_1 = \{17, 30, 29, 12, 19, 27, 15\}, \qquad S_2 = \{9, 18, 36, 35, 33, 29, 21\}.$$

Эти два множества имеют общий элемент 29, который стоит на $u = 3$ месте в первом множестве и на $v = 6$ месте во втором множестве. Таким образом, $\mathrm{ind}_2 23 = 7 \cdot 3 - 6 = 15$, а решение сравнения (2.33) имеет вид $x \equiv 15 \pmod{37}$.

Если в сравнении (2.30) число b не является первообразным корнем по модулю p, то это сравнение можно переписать в равносильном виде

$$x \cdot \mathrm{ind}_g b \equiv \mathrm{ind}_g a \pmod{p-1}. \tag{2.34}$$

Индексы чисел a, b можно вычислить с помощью алгоритма Гельфонда. После этого можно решить линейное сравнение (2.34), см. параграф **??**. Все найденные числа x составят решения (2.30). Если сравнение (2.34) не имеет решений, то не будет их иметь и сравнение (2.30).

Алгоритм Гельфонда, конечно, быстрее полного перебора, но он работает недопустимо медленно, чтобы решать задачу дискретного логарифмирования для простых чисел размером в 250 десятичных цифр.

Конец седьмой лекции.

Лекция 8.

2.10. Задача разложения целых чисел на множители.

В этой главе будут рассматриваться простейшие методы разложения целых чисел на простые сомножители, т.е. методы поиска для заданного целого $N > 1$ простых чисел $p_1, \ldots, p_r$ таких, что

$$N = p_1^{k_1} \cdots p_r^{k_r}, \qquad k_j \geqslant 1,\ k_j \in \mathbb{Z}.$$

При этом будет предполагаться, что разлагаемое число N составное, в чем можно убедиться с помощью тестов из параграфа 2.5.

Достаточно уметь решать более простую задачу о разложении целого числа на два меньших множителя, т.е. задачу о решении в целых числах $a > 1, b > 1$ уравнения $N = ab$. Действительно, в этом случае выполняются неравенства $a < N, b < N$, можно разложить на два меньших множителя каждое из чисел a, b, и продолжать далее эту процедуру, пока такое разложение будет возможным, т.е. до тех пор, пока все сомножители не станут простыми.

Существующие алгоритмы разложения чисел на множители могут быть распределены на группы в зависимости от количества арифметических операций, которые алгоритм требует для своей работы.

1) *Экспоненциальные* алгоритмы используют $O(N^c)$ арифметических операций. Здесь c — положительная постоянная.

2) *Субэкспоненциальные* алгоритмы требуют $O(e^{c(\ln N)^\alpha (\ln \ln N)^\beta})$ арифметических операций. Здесь α, β, c — положительные постоянные, $\alpha + \beta = 1$. Заметим, что при $\alpha = 1$, т.е. $\beta = 0$, оценка совпадает с оценкой сложности экспоненциальных алгоритмов.

Для наиболее быстрого из субэкспоненциальных алгоритмов, так называемого *метода решета числового поля*, или метода просеивания в числовых полях имеем $\alpha = \frac{1}{3}, \beta = \frac{2}{3}$.

Алгоритмы полиномиальной сложности, $\alpha = 0, \beta = 1$, для задачи факторизации не известны, и весьма вероятно, что их не существует.

Деятельность по разложению чисел на множители сочетает в себе черты инженерной науки, поскольку во многом опирается на допущения, основанные на опыте и не имеющие теоретических обоснований, а с другой стороны она сродни искусству, так как зачастую продолжительность работы алгоритма и результат зависят от удачного выбора параметров.

2.10.1. Алгоритм пробных делений.

Пусть $d_1 < d_2 < \ldots$ — последовательность целых чисел, содержащая все простые числа. Алгоритм пробных делений состоит в последовательном делении N на числа $d_1, d_2, \ldots$, не превосходящие $\sqrt{N}$. Если N составное число, то оно имеет простой делитель $p \leqslant \sqrt{N}$ и потому будет разложено на множители.

Этот алгоритм часто используется для нахождения всех простых делителей числа N, не превосходящих некоторой заданной границы B.

Последовательность d_i может совпадать с множеством простых чисел. Но в этом случае необходим алгоритм, строящий все простые числа до заданной границы. Иногда бывает проще использовать последовательности, содержащие и составные числа, но менее сложные в реализации.

Пример 1. Каждое простое число $p > 6$ удовлетворяет одному из двух сравнений $p \equiv 1 \pmod 6$ или $p \equiv 5 \pmod 6$. Поэтому последовательность

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 25, \ldots$$

содержащая все числа вида $6n \pm 1, n \in \mathbb{N}$, включает в себя и все простые числа. Но не только их. Например, она содержит 25 и 91, и многие другие составные числа. Но простое правило порождения этой последовательности облегчает поиск делителей N:

$$d_1 = 2,\ d_2 = 3,\ d_3 = 5, \quad d_{2k} = d_{2k-1} + 2, \quad d_{2k+1} = d_{2k} + 4, \quad k \geqslant 2.$$

При таком выборе d_i алгоритм вообще говоря требует $O(N^{1/2})$ арифметических операций и $O(1)$ памяти. Конечно, он очень медленный, но позволяет быстро отсеивать составные числа, имеющие малые простые делители.

Пример 2. В качестве модуля в сравнениях вместо 6 можно взять число $m = 2 \cdot 3 \cdot 5 = 30$. Все простые числа $p > 5$ будут содержаться в восьми прогрессиях с разностью 30, начинающихся с чисел $1, 7, 11, 13, 17, 19, 23, 29$, т.е.

$$30k + 7, 30k + 11, 30k + 13, 30k + 17, 30k + 19, 30k + 23, 30k + 29, 30k + 31.$$

Соответствующая последовательность d_i порождается формулами

$$\begin{aligned}
d_1 = 2, d_2 = 3, d_3 = 5, &\qquad d_{8k+4} = d_{8k+3} + 2,\\
d_{8k+5} = d_{8k+4} + 4, &\qquad d_{8k+6} = d_{8k+5} + 2,\\
d_{8k+7} = d_{8k+6} + 4, &\qquad d_{8k+8} = d_{8k+7} + 2,\\
d_{8k+9} = d_{8k+8} + 4, &\qquad d_{8k+10} = d_{8k+9} + 6,\\
d_{8k+11} = d_{8k+10} + 2, &\qquad d_{8k+12} = d_{8k+11} + 6, \quad k \geqslant 0.
\end{aligned}$$

Соответствующая последовательность $\{d_i\}$ будет содержать меньшую долю составных чисел. Если $m = \prod_{p \leqslant r} p$, то количество арифметических прогрессий, составляющих эту последовательность равно $\varphi(m)$, а доля чисел из $\{d_i\}$ в натуральном ряде есть $\dfrac{\varphi(m)}{m} = \prod_{p \leqslant r} \left(1 - \dfrac{1}{p}\right)$. При $m = 6$ она равна $\dfrac{1}{3}$, а при $m = 30$ имеем $\frac{1}{4}$.

2.10.2. ρ — метод Полларда.

Пусть $f(x)$ — "достаточно случайный"многочлен. Выберем "случайно" $x_1 \in \mathbb{Z}, 1 < x_1 < N$, и рассмотрим последовательность

$$x_{k+1} \equiv f(x_k) \pmod{N}, \qquad 0 \leqslant x_{k+1} < N, \quad k \geqslant 1, \tag{2.35}$$

Так как количество классов вычетов по модулю N не превосходит N, то существуют такие индексы $i, j, 0 \leqslant i, j < N$, что $x_i \equiv x_j \pmod{N}$ и значит, последовательность $x_k \pmod{N}$ зацикливается. Период этой последовательности не всегда начинается с самого начала, она может иметь предпериодическую часть. Символически такую последовательность можно изобразить в виде греческой буквы ρ. Подходящая снизу ножка этой буквы соответствует предпериодической части последовательности, она переходит в замкнутую петлю, соответствующую периодической части. Эта аналогия и дала название алгоритму.

Полларда обнаружил, что для простых p, как правило, длина периода и предпериодическая часть последовательности

$$x_{k+1} \equiv f(x_k) \pmod{p}, \qquad k \geqslant 1,$$

ограничены сверху величиной $c\sqrt{p}$, где c — некоторая константа.

Идея алгоритма состоит в том, чтобы последовательно вычислять наибольшие общие делители $(x_{2i} - x_i, N)$, $i = 1, 2, 3, \ldots$. Если p — простой делитель N, ℓ — длина периода и a — длина предпериодической части последовательности $x_{k+1} \equiv f(x_k) \pmod{p}$, то для номера i, удовлетворяющего условиям

$$i > a, \qquad \ell | i, \tag{2.36}$$

числа x_i и $x_{i+\ell}$ сравнимы друг с другом по модулю p. Мы не знаем чисел ℓ, i, но знаем, что разность $2i - i = i$ делится на ℓ, поэтому $x_{2i} \equiv x_i \pmod{p}$, так что $(x_{2i} - x_i, N) > 1$. Ясно, что существует число i, удовлетворяющее условиям (2.36) и неравенству $i \leqslant a + \ell = O(\sqrt{p})$. Конечно, может случиться, что $N | x_{2i} - x_i$. Тогда нужно выбирать иное начальное значение x_1.

Если p — наименьший простой делитель N, то $p \leqslant N^{1/2}$, так что $i = O(N^{1/4})$.

В следующем ниже алгоритме вычисляются пары $\{x_i, x_{2i}\}, i \geqslant 1$. Для нахождения следующей пары $\{x_{i+1}, x_{2i+2}\}$ нужно вычислить

$$x_{i+1} \equiv f(x_i) (\mathrm{mod}\, N), \quad x_{2i+1} \equiv f(x_{2i}) (\mathrm{mod}\, N),$$
$$x_{2i+2} \equiv f(x_{2i+1}) (\mathrm{mod}\, N),$$

т.е. выполнить $O(1)$ арифметических операций.

Алгоритм 9. Дано: *составное число* N. Найти: *нетривиальный делитель* N .

1. Выбрать случайно $x_1 \in \mathbb{Z}, 1 < x_1 < N$, *и положить*

$$x = f(x_1) \pmod N, \qquad y = f(x) \pmod N.$$

2. Вычислить $d = (y - x, N)$. *Если* $1 < d < N$ *алгоритм останавливается, нетривиальный делитель* d *числа* N *найден.*

3. Если $d = N$, *перейти в п. 1.*

4. Положить

$$x = f(x) \pmod N, \quad z = f(y) \pmod N, \quad y = f(z) \pmod N$$

и перейти в пункт 2 алгоритма.

Если все удачно сложится, то нетривиальный делитель числа N будет найден за $O(p^{1/2})$ арифметических операций, где p — наименьший простой делитель N, т.е. за $O(N^{1/4})$ арифметических операций.

В силу оценки сложности $O(p^{1/2})$ алгоритм удобен для нахождения не очень больших делителей p числа N, если они есть.

В качестве $f(x)$ обычно выбираются многочлены вида $x^2 + a$. Например, можно взять $f(x) = x^2 + 1$ или $f(x) = x^2 - 1$. В то же время выбор $f(x) = x^2 - 2$ и $x_1 = 2$ не очень удачен, так как в этом случае последовательность x_k имеет период 1.

Приведем теперь некоторые правдоподобные соображения в пользу того, что длина периода и предпериодическая часть последовательности $x_{k+1} \equiv f(x_k) \pmod p$ оцениваются сверху величиной $O(p^{1/2})$.

2.10.3. Парадокс дней рождения.

Пусть $\mathcal{R}$ - множество, состоящее из r элементов. Из принципа ящиков Дирихле следует, что в любой последовательности $z_1, z_2, \ldots z_{r+1}$, $z_i \in \mathcal{R}$, обязательно найдутся два одинаковых элемента. Но можно проверить,

что выбирая случайным образом множество $z_1, z_2, \ldots, z_m$ с теми же условиями на z_i при достаточно большом, но всё же существенно меньшем r значении m, можно с вероятностью, большей $\frac{1}{2}$ попасть на набор, имеющий два одинаковых элемента. Этот кажущийся парадокс носит название "парадокс дней рождения" по причине, о которой мы расскажем немного позже, а сейчас докажем, при некоторых естественных предположениях, указанную оценку на вероятность "хорошей" выборки.

Будем выбирать случайным образом наборы

$$\overline{z} = \{z_1, \ldots, z_m\}, \qquad z_i \in \mathcal{R}, \tag{2.37}$$

предполагая, что все они равновероятны. Какова вероятность того, что выбранная таким образом совокупность $z_1, z_2, \ldots, z_m$, содержит по крайней мере два одинаковых элемента?

Каждое значение z_i может равняться одному из r элементов множества $\mathcal{R}$. Значит количество последовательностей вида (2.40) равно r^m. Сколько же среди них последовательностей с различными элементами. Попробуем построить такие последовательности. На первом месте может стоять любое из возможных r значений для z_1. На втором месте может стоять лишь $r-1$ значений, отличных от стоящего на первом месте. На третьем месте может стоять лишь $r-2$ значения, отличных от первых двух. Действуя так далее мы сможем построить $r(r-1)(r-2)\cdots(r-m+1)$ последовательностей, состоящих из m различных значений. Легко видеть, что каждая такая последовательность будет учтена в этом процессе. Следовательно, доля последовательностей с различными значениями z_i будет равна

$$\mu = \frac{r(r-1)...(r-m+1)}{r^m}.$$

Для оценки μ сверху перепишем

$$\mu = \prod_{k=1}^{m-1}\left(1-\frac{k}{r}\right) \quad \text{и} \quad \ln\mu = \sum_{k=1}^{m-1}\ln(1-\frac{k}{r}). \tag{2.38}$$

При любом действительном x, $0 \leqslant x < 1$ справедливо неравенство

$$\ln(1-x) \leqslant -x. \tag{2.39}$$

Для его доказательства рассмотрим непрерывную на промежутке $[0;1)$ функцию $g(x) = x + \ln(1-x)$. Так как производная $g'(x) = \frac{-x}{1-x}$ отрицательна на интервале $(0;1)$, то $g(x)$ убывает на множестве $0 \leqslant x < 1$, и, значит, на этом множестве выполняется неравенство (2.39).

Взяв $x = \frac{k}{r}$, находим $\ln(1 - \frac{k}{r}) < -\frac{k}{r}$, $1 \leqslant k < m$ и из (2.38)

$$\ln \mu < -\sum_{k=1}^{m-1} \frac{k}{r} = -\frac{m^2 - m}{2r} < -\frac{(m-1)^2}{2r}.$$

Выберем теперь $m = \lceil \sqrt{2r \ln 2} \rceil + 1$. Тогда $(m-1)^2 = \lceil \sqrt{2r \ln 2} \rceil^2 \geqslant 2r \ln 2$ и $\ln \mu < -\ln 2$. Так доказано неравенство $\mu < \frac{1}{2}$. Заметим, что при $r \geqslant 5$ выполняется $m < r$.

Если увеличить константу $\sqrt{2 \ln 2}$ в определении m, оценка μ уменьшится и, значит, вероятность попасть на набор, имеющий по крайней мере два равных элемента, станет больше $\frac{1}{2}$. Например, в случае $m = \lceil 4\sqrt{r} \rceil$ вероятность попасть на выборку, имеющую 2 или более одинаковых элементов, превосходит $0,9996\ldots$.

Вернёмся теперь к последовательности (2.35) и возьмём постоянную в определении m большую, чем $\sqrt{2 \ln 2}$. Из сказанного выше следует, что для "хорошего" многочлена $f(x)$ и быть может после нескольких случайных выборов числа x_1 мы за $O(\sqrt{p} \ln p)$ арифметических операций сможем найти последовательность $x_1, x_2, \ldots, x_m$, имеющую два равных числа. Длины предпериода a и периода ℓ этой последовательности удовлетворяют неравенству $a + \ell \leqslant M$. Действительно, a есть номер первого из двух равных чисел, и $a + \ell$ - номер второго из них чисел. Это объясняет наблюдение Полларда о длине предпериода и периода последовательности в ρ - методе Полларда.

Конечно, это рассуждение не является доказательством, но оно дает правдоподобное объяснение, почему ρ — метод Полларда достаточно быстро на практике находит небольшие простые делители составных чисел.

Заметим также, что скорость работы этого алгоритма может быть увеличена, если наибольший общий делитель $(x_{2i} - x_i, N)$ вычислять не на каждом шаге. Например, можно для последовательности $i = 0, d, 2d, \ldots$ вычислять $(\prod_{k=i+1}^{i+d}(x_{2k} - x_k), N)$, выбрав d не очень большим.

Теперь мы обсудим алгоритм, также основанный на парадоксе дней рождения и позволяющий найти коллизию у любой хеш-функции $H(x)$. Он носит вероятностный характер. Обозначим буквой r количество всех различных значений функции $H(\overline{x})$, положим также $m = [\sqrt{2r}] + 1$. Не трудно проверить, что при $r \geqslant 6$ выполняется неравенство $m < r$. Выберем случайно различные аргументы $x_1, \ldots, x_m$ функции $H(\overline{x})$ и вычислим хеш-значения

$$H(x_1), \quad \ldots, \quad H(x_m). \tag{2.40}$$

Если среди этих значений есть одинаковые, то коллизия построена. При выбранном m справедливы неравенства $(m-1)^2 > \lceil\sqrt{2r}\rceil^2 > 2r$ и $\ln\mu < -1$, так что в противном случае доля последовательностей без одинаковых элементов не превосходит e^{-1}. Поэтому $\mu < e^{-1} = 0,367...$, и количество последовательностей с различными хеш-значениями существенно меньше половины всех последовательностей. Это доказывает, что если все значения хеш-функции равновероятны, то при случайном выборе набора аргументов $x_1, x_2, \ldots, x_M$ вероятность получить последовательность значений, имеющую равные элементы, превосходит $1 - e^{-1} > \frac{1}{2}$. Если все элементы выбранной последовательности хеш-значений оказались различными, можно выбрать вторую последовательность. Если опять не повезло, можно выбрать третью последовательность. Вероятность выбрать последовательность с равными элементами после k шагов не меньше $1 - e^{-k}$. С ростом k она очень быстро приближается к 1. К сожалению, этот гарантированно работающий алгоритм требует слишком много времени. Например, в случае $r = 2^{256}$ имеем $m = \lceil\sqrt{2r}\rceil + 1 > 2^128$. Последовательность из 2^{128} хеш-значений имеет огромную длину.

Рассмотрим ещё одну ситуацию, котороая собственно и дала название корневому понижению оценки числа арифметических опереций при нахождении в последовательности равных элементов. Представим себе, что студенческая группа или просто группа знакомых собралась на вечеринку по какому-то поводу. В группе 23 человека и требуется определить, какова вероятность, что среди приглашенных есть двое, дни рождения которых приходятся на одну дату. На первый взгляд кажется, что эта вероятность очень маленькая. Но рассмотрим ситуацию чуть подробнее. Если год, в котором происходит это событие не високосный, т.е. состоит из $p = 365$ дней, то вероятность выбрать 23 различные даты из 365 равна

$$\mu = \prod_{k=1}^{22}\left(1 - \frac{k}{365}\right) = 0,492703\ldots = \\ = \frac{36997978566217959340182499134166757044383351847256064}{75091883268515350125426207425223147563269805908203125}.$$

Последнее значение точное. Таким образом, любой случайный набор из 23 дат невисокосного года с вероятностью большей $\frac{1}{2}$ содержит две одинаковых даты.

Конец восьмой лекции.