# Алгоритмы поиска длин циклов в последовательностях и их приложения

**А. Ю. НЕСТЕРЕНКО**
*Московский государственный институт электроники и математики*
e-mail: nesterenko_a_y@mail.ru

УДК 511.178+003.26

**Ключевые слова:** поиск циклов в последовательностях, факторизация, дискретное логарифмирование, эллиптические кривые.

**Аннотация**

В работе рассматриваются алгоритмы поиска длин циклов в последовательностях. Приводится обоснование изложенных алгоритмов, сравнение оценок их трудоёмкости, а также результаты их практического применения для решения задачи дискретного логарифмирования в группе точек эллиптической кривой.

**Abstract**

*A. Yu. Nesterenko, Cycle detection algorithms and their applications*, Fundamentalnaya i prikladnaya matematika, vol. 16 (2010), no. 6, pp. 109—122.

The paper considers several cycle detection algorithms. Proofs of their correctness are given, bounds for complexity are obtained, some number theory applications like the factorization of integers and the discrete log problem are examined.

Рассмотрим конечное множество $\mathcal{M}$, состоящее из $m$ элементов. Зафиксируем некоторое отображение $f\colon \mathcal{M} \to \mathcal{M}$ множества $\mathcal{M}$ в себя и произвольный элемент $a_0 \in \mathcal{M}$ и рассмотрим последовательность элементов $a_0, a_1, a_2, \ldots$, определяемую равенством

$$a_{n+1} = f(a_n), \quad n = 0, 1, \ldots . \tag{1}$$

Поскольку множество $\mathcal{M}$ конечно, то начиная с некоторого момента последовательность (1) зациклится. Цикл может начинаться с произвольного элемента $a_\lambda$, так что выполнено равенство

$$a_n = a_{n+\tau} \ \text{ для всех } \ n \geqslant \lambda \geqslant 0 \ \text{ и } \ \tau \geqslant 1. \tag{2}$$

На протяжении всей статьи мы будем называть величину $\lambda$ длиной подхода к циклу, а величину $\tau$ длиной цикла. Задача заключается в определении величины $\tau$ при известных отображении $f$ и начальном элементе $a_0$.

В настоящее время известно несколько алгоритмов решения данной задачи. Первый и самый известный алгоритм был предложен в 1968 году Робертом

*Фундаментальная и прикладная математика*, 2010, том 16, № 6, с. 109—122.

Флойдом (см. [10, п. 3.1, задача 6b]). Годом позже Дональдом Кнутом был опубликован (см. [10, п. 3.1, задача 7b]) алгоритм Ричарда Брента. Однако наиболее эффективный способ решения поставленной задачи заключается в использовании алгоритма, предложенного в 1972 году Биллом Госпером (см. [3, п. 132]).

Несмотря на свою эффективность, алгоритм Госпера не является широко известным. Единственное его описание на русском языке может быть найдено в переводной книге Генри Уоррена [2, п. 5.4]. Возможно, в силу простоты алгоритма Госпер не дал его строгого математического обоснования. Более того, на настоящий момент автору не известно ни одного опубликованного обоснования данного алгоритма.

Эта статья написана для того, чтобы исправить это маленькое недоразумение и привлечь внимание к алгоритму Госпера. Мы предлагаем оригинальное обоснование алгоритма и примеры его приложений к решению ряда практических задач.

Стоит также отметить ещё два алгоритма. Первый был предложен Робертом Седжвиком и Томасом Шиманским в 1981 году, оценки времени его работы приведены в статье [14]. Алгоритм Седжвика—Шиманского обладает наименьшей трудоёмкостью среди всех известных алгоритмов поиска длин циклов в последовательностях. Вместе с тем алгоритм использует столь большой объём памяти для хранения промежуточных элементов последовательности, что не применяется на практике.

Второй алгоритм предложен Габриэлом Нивашем в 2004 году [11] и основан на очень красивой идее поиска минимального элемента, лежащего внутри цикла. Алгоритм имеет сравнимую с методом Госпера трудоёмкость и объём используемой памяти. Однако данный алгоритм может быть использован только для тех множеств $\mathcal{M}$, на которых можно ввести отношение упорядоченности элементов. В статье мы приведём подробное описание алгоритма Ниваша.

В заключение мы опишем несколько примеров применения рассматриваемых алгоритмов к решению некоторых теоретико-числовых задач и приведём результаты практических вычислений на ЭВМ.

# 1. Алгоритм Флойда

Алгоритм Флойда является самым простым и хорошо известным алгоритмом, решающим рассматриваемую задачу. Он основан на выполнении следующего условия: если выполнено равенство

$$a_n = a_{2n}, \tag{3}$$

то величина $\tau$ делит $n$. Для вычисления $\tau$ можно использовать следующий алгоритм.

**Алгоритм 1**: алгоритм Флойда

**Вход**: Отображение $f\colon \mathcal{M} \to \mathcal{M}$ и начальный элемент $a_0$.
**Выход**: Значение длины цикла $\tau$.
1 Определить начальные значения $a \leftarrow a_0$ и $b \leftarrow f(a)$.
2 **Пока** $a \neq b$ **выполнять** $a \leftarrow f(a)$, $c \leftarrow f(b)$, $b \leftarrow f(c)$.
3 Определить $b \leftarrow f(a)$ и $\tau \leftarrow 1$.
4 **Пока** $a \neq b$ **выполнять** $b \leftarrow f(b)$ и $\tau \leftarrow \tau + 1$.

Второй шаг алгоритма выполняется до тех пор, пока не будет достигнуто равенство (3), при этом точное значение индекса $n$ не вычисляется. Поскольку задача поиска всех делителей числа $n$ в общем случае является весьма трудоёмкой, то на четвёртом шаге алгоритма мы вычисляем точное значение $\tau$ непосредственным перебором.

Легко убедиться, что минимально возможное значение индекса $n$, при котором выполняется равенство $a_n = a_{2n}$, равно

$$\tau \left\lceil \frac{\lambda}{\tau} \right\rceil .$$

Таким образом, трудоёмкость алгоритма Флойда равна

$$\tau \left( 3 \left\lceil \frac{\lambda}{\tau} \right\rceil + 1 \right)$$

операций вычисления функции $f$.

## 2. Алгоритм Брента

Прежде чем описывать алгоритм вычисления длины цикла последовательности (1), мы докажем теорему, утверждения которой служат его обоснованием. При доказательстве теоремы мы будем следовать идеям, изложенным в монографии [6, п. 8.2.2]. Определим функцию целочисленного аргумента

$$l(n) = 2^{\lfloor \log_2 n \rfloor}, \quad n = 1, 2, \ldots,$$

которая возвращает максимальное целое число, являющееся степенью двойки и не превосходящее числа $n$. Из определения функции $l(n)$ следует, что $l(n) \leqslant n < 2l(n)$. Определим параметр $k$ равенством

$$k = \lceil \log_2 \max\{\lambda + 1, \tau\} \rceil ,$$

где $\tau$ обозначает длину цикла, а $\lambda$ — длину подхода к циклу в последовательности, порождённой элементом $a_0$. Из определения параметра $k$ следуют неравенства $\tau \leqslant 2^k$ и $\lambda \leqslant 2^k - 1$. Определим индекс $n_0$ равенством

$$n_0 = 2^k + \tau \left\lceil \frac{l(\lambda) + 1}{\tau} \right\rceil - 1. \tag{4}$$

**Теорема 1.** *Справедливы следующие утверждения.*

1. *Верны неравенства* $2^k \leqslant n_0 < 2^{k+1}$.
2. *Для индекса* $n_0$ *выполнено равенство* $a_{n_0} = a_{l(n_0)-1}$.
3. *Выполнено неравенство* $\frac{3}{2} l(n_0) \leqslant n_0 < 2l(n_0)$.

**Доказательство.** Начнём с доказательства первого утверждения теоремы. Поскольку длина цикла $\tau$ является целым числом, то $\tau \geqslant 1$. Поскольку неравенство

$$\left\lceil \frac{l(\lambda)+1}{\tau} \right\rceil \geqslant 1$$

выполнено по определению, то неравенство $2^k \leqslant n_0$ очевидно. Для оценки сверху заметим, что нам достаточно показать, что

$$\tau \left\lceil \frac{l(\lambda)+1}{\tau} \right\rceil \leqslant 2^k,$$

тогда $n_0 \leqslant 2 \cdot 2^k - 1 < 2^{k+1}$.

1. Сначала рассмотрим случай, когда $\tau > l(\lambda)$. Тогда выполнены неравенства $\tau \geqslant l(\lambda)+1$ и $\frac{l(\lambda)+1}{\tau} \leqslant 1$. Следовательно,

   $$\left\lceil \frac{l(\lambda)+1}{\tau} \right\rceil = 1, \quad \tau \left\lceil \frac{l(\lambda)+1}{\tau} \right\rceil = \tau \leqslant 2^k.$$

   Последнее неравенство выполнено в силу выбора параметра $k$.
2. Рассмотрим второй случай: $\tau \leqslant l(\lambda)$. Тогда выполнено

   $$\left\lceil \frac{l(\lambda)+1}{\tau} \right\rceil = \frac{l(\lambda)+1+\delta}{\tau} \leqslant \frac{l(\lambda)+\tau}{\tau} \leqslant \frac{2l(\lambda)}{\tau}$$

   при некотором натуральном $\delta < \tau$. Полученное неравенство позволяет записать

   $$\tau \left\lceil \frac{l(\lambda)+1}{\tau} \right\rceil \leqslant 2l(\lambda) = 2^{\lfloor \log_2 \lambda \rfloor + 1} = 2^{\lceil \log_2 (\lambda+1) \rceil} \leqslant 2^k$$

   (мы пользуемся равенством $\lfloor \log_2 \lambda \rfloor + 1 = \lceil \log_2 (\lambda+1) \rceil$, которое выполнено при натуральных значениях $\lambda$).

Мы получили, что в обоих случаях выполнено необходимое неравенство. Таким образом, первое утверждение теоремы доказано.

Для доказательства второго утверждения теоремы заметим, что из первого утверждения следует, что

$$k \leqslant \log_2 n_0 < k+1, \quad l(n_0) = 2^k.$$

Тогда разность

$$n_0 - (l(n_0) - 1) = \tau \left\lceil \frac{l(\lambda)+1}{\tau} \right\rceil$$

кратна длине цикла $\tau$, т. е. равенство $a_{n_0} = a_{l(n_0)-1}$ действительно имеет место.

Нам осталось доказать последнее утверждение теоремы. Оценка сверху тривиально вытекает из определения функции $l(n)$. Для получения нижней оценки заметим, что справедливы неравенства

$$\tau \left\lceil \frac{l(\lambda)+1}{\tau} \right\rceil - 1 \geqslant \tau - 1 = 2^{\log_2 \tau} - 1 \geqslant 2^{\lceil \log_2 \tau \rceil - 1}, \tag{5}$$

$$\tau \left\lceil \frac{l(\lambda)+1}{\tau} \right\rceil - 1 \geqslant l(\lambda) = 2^{\lceil \log_2 \lambda + 1 \rceil - 1}. \tag{6}$$

Следовательно, выполнено неравенство

$$\tau \left\lceil \frac{l(\lambda)+1}{\tau} \right\rceil - 1 \geqslant 2^{k-1},$$

откуда следует соотношение

$$n_0 = 2^k + \tau \left\lceil \frac{l(\lambda)+1}{\tau} \right\rceil - 1 \geqslant 2^k + 2^{k-1} = \frac{3l(n_0)}{2}$$

и доказательство последнего утверждения теоремы. □

Доказанная нами теорема в явном виде задаёт значение индекса $n_0$, которое необходимо вычислить для определения длины цикла. К сожалению, на практике значение $n_0$ нам неизвестно.

Основываясь на утверждениях доказанной теоремы, Р. Брент предложил следующую идею: для поиска значения, кратного величине $\tau$, необходимо воспользоваться тем фактом, что $a_{n_0} = a_{2^k-1}$ при некотором натуральном значении $k$, таком что $3 \cdot 2^{k-1} \leqslant n_0 < 2^{k+1}$.

**Алгоритм 2**: алгоритм Брента

**Вход**: Отображение $f\colon \mathcal{M} \to \mathcal{M}$ и начальный элемент $a_0$.
**Выход**: Значение длины цикла $\tau$.

1 Определить начальные значения $c \leftarrow a_0$, $a \leftarrow f(a_0)$, $n \leftarrow 1$ и $t \leftarrow 1$.
2 **Если** $a = c$, **то** вернуть значение $\tau \leftarrow 1$ и завершить алгоритм.
3 **Если** $n = t$, **то** положить $c = a$ и вычислить $t = 2t$.
4 Определить $a \leftarrow f(a)$ и вычислить $n \leftarrow n + 1$.
5 **Если** $n \geqslant 3t/4$, **то** проверить, выполняется ли равенство $a = c$. Если нет, то вернуться на шаг 3.
6 Определить $\tau \leftarrow 1$ и $a \leftarrow f(c)$.
7 **Пока** $a \neq c$ **выполнять** $a \leftarrow f(a)$ и $\tau \leftarrow \tau + 1$.

Алгоритм Брента, как и алгоритм Флойда, не позволяет в явном виде определить значение длины цикла $\tau$. На пятом шаге приведённого алгоритма мы находим совпадение $a_{n_0} = a_{2^k-1}$ для некоторого натурального числа $k$. Последние два шага приведённого алгоритма позволяют определить значение $\tau$ в явном виде. Как и ранее, мы не вычисляем значение $n_0 + 1 - 2^k$, кратное величине $\tau$, а находим искомую величину простым перебором.

Для оценки трудоёмкости алгоритма Брента заметим, что из третьего утверждения теоремы 1 и определения величины $k$ следует оценка снизу

$$n_0 \geqslant \frac{3l(n_0)}{2} = \frac{3}{2} \cdot 2^k \geqslant \frac{3}{2} \max\{\lambda + 1, \tau\}$$

для числа шагов, необходимых для поиска совпадения на пятом шаге алгоритма. Таким образом, общая трудоёмкость алгоритма Брента составит не менее

$$\tau + \frac{3}{2} \max\{\lambda + 1, \tau\}$$

операций вычисления функции $f$.

# 3. Алгоритм Госпера

В алгоритме Госпера для поиска двух совпадающих элементов последовательности (1)

$$a_{n+1} = f(a_n), \quad n = 0, 1, \ldots,$$

производится сравнение $a_n$ с элементами некоторого множества $M(n)$. Фиксируем значение $n > 0$ и поместим в множество $M(n)$ элементы $a_{n_0}, a_{n_1}, \ldots$ последовательности (1), такие что

$$n_k = \max_{r<n}\{r \mid \nu_2(r+1) = k\} \tag{7}$$

для всех возможных значений $k = 0, 1, \ldots$. Напомним, что функция $\nu_2(r+1)$ возвращает наибольшую степень двойки, делящую величину $r+1$. Из определения следует, что множество $M(n)$ конечно, содержит не более $\lfloor \log_2 n \rfloor + 1$ чисел и отличается от множества $M(n+1)$ лишь одним элементом.

**Теорема 2.** *Пусть заданы параметры $\lambda$ и $\tau$, определяющие длину подхода к циклу и длину цикла последовательности* (1)*. Тогда найдутся натуральные индексы $r$ и $n = r + \tau$, такие что*

1) *элемент $a_r$ принадлежит множеству $M(n)$ и выполнено равенство $a_n = a_r$;*
2) $\lambda + \tau \leqslant n < \lambda + 2\tau$.

**Доказательство.** Определим в качестве параметра $k$ целое число, удовлетворяющее неравенствам $2^k \leqslant \tau < 2^{k+1}$, и рассмотрим целые числа

$$r = 2^k \left\lceil \frac{\lambda + 1}{2^k} \right\rceil - 1, \quad n = r + \tau.$$

Очевидно, что $r < n$ для любого целого $\tau \geqslant 1$. Рассмотрим представление

$$\left\lceil \frac{\lambda + 1}{2^k} \right\rceil = 2^l s,$$

где $l \geqslant 0$ — целое число и $s = 2h + 1$ — нечётное целое число. Тогда $r$ имеет вид

$$r = 2^{k+l}s - 1 = 2^{k+l}(2h+1) - 1 = 2^{k+l} - 1 + h2^{k+l+1}, \tag{8}$$

т. е. $r \equiv 2^{k+l} - 1 \pmod{2^{k+l+1}}$, и мы получаем, что $\nu_2(r+1) = k+l$. Поскольку выполнено неравенство

$$r + 2^{k+l+1} \geqslant r + 2^{k+1} > r + \tau = n,$$

мы получаем, что индекс $r$ принадлежит множеству $M(n)$. Учитывая, что $\tau$ — период последовательности (1), имеем $a_r = a_{r+\tau} = a_n$. Первое утверждение теоремы доказано.

Для доказательства второго утверждения теоремы получим оценки снизу и сверху на величину $r$. Воспользовавшись неравенством

$$x \leqslant \lceil x \rceil < x + 1,$$

выполненным для любого действительного $x > 0$, получим, что

$$\lambda = 2^k \left( \frac{\lambda+1}{2^k} \right) - 1 \leqslant 2^k \left\lceil \frac{\lambda+1}{2^k} \right\rceil - 1 = r < \\ < 2^k \left( \frac{\lambda+1}{2^k} + 1 \right) - 1 = \lambda + 2^k \leqslant \lambda + \tau,$$

т. е. справедливо неравенство $\lambda \leqslant r < \lambda + \tau$, из которого следует утверждение теоремы. □

Утверждение теоремы в явном виде задаёт нам множество $M(n)$, в котором содержится элемент $a_r$, такой что $a_r = a_n$. Более того, теорема позволяет получить оценку сверху на максимальное число элементов последовательности (1), которые необходимо вычислить для нахождения указанного равенства.

Мы можем построить множество $M(n)$ следующим образом. Разобьём последовательность (1) на несколько подпоследовательностей так, что первая подпоследовательность содержит все элементы с такими индексами $i$, что $i+1$ нечётно, вторая — элементы с такими индексами $i$, что $i+1$ делится в точности на двойку, третья — элементы с такими индексами $i$, что $i+1$ делится в точности на четвёрку и т. д. Тогда в множество $M(n)$ входит по одному элементу из каждой подпоследовательности с максимальным индексом, не превосходящим $n$. Например, для $n = 16$ множество $M(16)$ имеет вид

$$M(16) = \{a_{14}, a_{13}, a_{11}, a_7, a_{15}\}.$$

Мы храним множество $M(n)$ в массиве $T$, содержащем элемент с максимальным индексом из каждой подпоследовательности. Элемент $T[0]$ содержит элемент первой подпоследовательности, $T[1]$ — элемент второй подпоследовательности и так далее.

Неизвестное значение $\tau$ мы определяем на четвёртом шаге алгоритма. Из (7) и (8) следует, что при завершении работы алгоритма

$$h = \left\lfloor \frac{n - (2^k - 1)}{2^{k+1}} \right\rfloor, \quad r = 2^k - 1 + h2^{k+1}.$$

Поскольку $\tau = n - r$, то мы получаем равенство, приведённое выше.

**Алгоритм 3**: алгоритм Госпера

**Вход**: Отображение $f\colon \mathcal{M} \to \mathcal{M}$ и начальный элемент $a_0$.
**Выход**: Значение длины цикла $\tau$.
1 Определить начальные значения: $a \leftarrow a_0$, $n \leftarrow 1$, $t \leftarrow 1$ и $T[0] \leftarrow a_0$.
2 Вычислить $a \leftarrow f(a)$.
3 **Для всех** $i = 0, \ldots, t-1$ **выполнять**
4 если выполнено равенство $T[i] = a$, то завершить алгоритм с результатом $\tau \leftarrow n - 2^k\bigl(1 + 2\bigl\lfloor \frac{n-2^k+1}{2^{k+1}} \bigr\rfloor\bigr) + 1$ при $k = i$.
5 **конец**
6 Вычислить $n \leftarrow n+1$ и $k \leftarrow \nu_2(n)$.
7 **Если** $k = t$, **то** вычислить $t = t+1$.
8 Определить $T[k] = a$ и вернуться на шаг 2.

# 4. Алгоритм Ниваша

Предложенный в 2004 году алгоритм Ниваша основан на следующей очень простой идее. Пусть на множестве $\mathcal{M}$ задано отношение упорядоченности, т. е. для любых двух элементов $a, b \in \mathcal{M}$ определена такая функция $h\colon \mathcal{M} \to \{-1, 0, 1\}$, что

$$h(a,b) = \begin{cases} -1, & \text{если } a < b, \\ 0, & \text{если } a = b, \\ 1, & \text{если } a > b. \end{cases}$$

В этом случае в цикле последовательности (1) может быть определён наименьший элемент $a_l$, такой что $h(a_l, a_n) = -1$ для всех индексов $n = \lambda, \ldots, \lambda + \tau - 1$ при $n \neq l$. Идея Ниваша заключается в том, что минимальный элемент может быть определён при первом проходе цикла, тогда на втором проходе цикла будет найдено совпадение.

Для реализации поиска минимального элемента нам потребуется массив $T$, элементами которого будут являться пары $(a_n, n)$, содержащие значение элемента последовательности и его индекс. Мы будем обозначать через $T[i].a$ значение элемента последовательности, содержащееся в $i$-й ячейке массива. Аналогично $T[i].n$ будет означать индекс сохранённого в ячейке элемента последовательности.

Легко убедиться, что количество вычислений функции $f$ в приведённом алгоритме не превышает величины $\lambda + 2\tau$, т. е. длины подхода и двух циклов последовательности (1). Массив $T$ реализуется как стек, в котором хранятся элементы последовательности, отсортированные по возрастанию. Каждый новый вырабатываемый элемент $a$ помещается в стек. Если в нём есть элементы, большие чем $a$, они удаляются.

**Алгоритм 4**: алгоритм Ниваша

**Вход**: Отображение $f\colon \mathcal{M} \to \mathcal{M}$ и начальный элемент $a_0$.
**Выход**: Значение длины цикла $\tau$.
1 Определить значения $a \leftarrow a_0$, $n \leftarrow 0$, $k \leftarrow 1$ и $T[0] \leftarrow (a_0, 0)$.
2 Определить $a \leftarrow f(a)$, $n \leftarrow n+1$ и $i \leftarrow k$.
3 **Выполнять** $i = i-1$ **пока** $(i \geqslant 0$ и $h(a, T[i].a) = -1)$.
4 **Если** $h(a, T[i].a) = 0$, **то** определить $\tau = n - T[i].n$ и завершить алгоритм.
5 **Если** $h(a, T[i].a) = 1$, **то** определить новое значение элемента массива $T[i+1].a = a$, $T[i+1].n = n$ и $k = i+2$.
6 Вернуться на шаг 2.

Если предположить, что элементы последовательности (1) являются реализацией некоторой случайной равномерно распределённой на множестве $\mathcal{M}$ величины, то, как показано в [9, п. 1.2.10], с ростом индекса $n$ размер стека растёт как величина порядка $O(\log_2 n)$. Таким образом, мы можем считать, что величина стека в алгоритме Ниваша составляет $\lceil \log_2(\lambda + 2\tau) \rceil$ элементов, каждый из которых хранит элемент последовательности (1) и его индекс.

# 5. Сравнительный анализ алгоритмов и приложения

Для сравнения сведём в одну таблицу различные характеристики описанных выше алгоритмов. Во второй колонке мы приводим оценку трудоёмкости алгоритма, измеряемую как количество вычислений функции $f$. В третьей колонке содержится количество ячеек памяти, необходимых для вычисления длины цикла $\tau$.

| Алгоритм | Трудоёмкость | Объём памяти |
|---|---|---|
| Флойда | $\tau\bigl(3\lceil \frac{\lambda}{\tau} \rceil + 1\bigr)$ | 3 |
| Брента | не менее $\tau + \frac{3}{2}\max\{\lambda+1, \tau\}$ | 4 |
| Госпера | не более $\lambda + 2\tau$ | $\lceil \log_2 m \rceil + 4$ |
| Ниваша | не более $\lambda + 2\tau$ | $2\lceil \log_2(\lambda + 2\tau) \rceil$ |

Рассмотрение вопроса о трудоёмкости приведённых в таблице алгоритмов в зависимости от мощности $m$ множества $\mathcal{M}$ сводится к определению значений математического ожидания $E_m(\lambda, f)$ и $E_m(\tau, f)$ для величин $\lambda$ и $\tau$ соответственно при случайном выборе отображения $f$. Хорошо известно (см., например, [1, п. 5.3]), что

$$\lim_{m\to\infty} \frac{E_m(\lambda, f)}{\sqrt{m}} = \lim_{m\to\infty} \frac{E_m(\tau, f)}{\sqrt{m}} = \sqrt{\frac{\pi}{8}}. \tag{9}$$

Таким образом, асимптотическая оценка у всех приведённых алгоритмов одинакова и составляет $O(\sqrt{m})$.

В заключение мы остановимся на некоторых приложениях описанных выше алгоритмов к решению некоторых задач теории чисел: задаче факторизации и задаче дискретного логарифмирования.

## 5.1. Факторизация целых чисел

Вопрос применения алгоритмов поиска длин циклов в последовательностях к решению задачи факторизации в настоящее время хорошо изучен. Идея применить метод Флойда для решения задачи разложения больших целых чисел на множители была высказана в 1975 году Джоном Поллардом [13]. Данный метод получил название $\rho$-метода Полларда или метода Полларда—Флойда. В 1980 году Р. Брент (см. [5]) предложил использовать для этой же цели свой алгоритм.

Пусть $m$ — составное число, для которого требуется найти делитель $p$, и $\mathbb{Z}_m$ — кольцо вычетов по модулю $m$. Идея обоих методов заключается в вычислении последовательности вычетов $a_0, a_1, \ldots$ кольца $\mathbb{Z}_m$ для некоторого полиномиального отображения $f\colon \mathbb{Z}_m \to \mathbb{Z}_m$.

В качестве множества $\mathcal{M}$ выступает кольцо вычетов $\mathbb{Z}_p$. Тогда в случае обнаружения периода выполнено сравнение $a_n \equiv a_{n+\tau} \pmod p$ и делитель $p$ может быть найден путём проверки условия

$$1 < \text{НОД}(m, a_{n+\tau} - a_n) < m. \tag{10}$$

Заметим, что оба алгоритма могут быть реализованы в двух модификациях. Первая работает до тех пор, пока не будет найден нетривиальный делитель $p$. Вторая модификация представляет собой тест, который вычисляет лишь заданное количество элементов последовательности и тем самым проверяет лишь наличие небольших делителей у числа $m$. Действительно, согласно (9) после вычисления $n$ шагов последовательности мы вправе ожидать определения делителей числа $m$, не превосходящих величины $O(n^2)$.

Р. Брент также утверждал (см. [5, с. 177]), что метод Госпера поиска длин циклов не позволит снизить трудоёмкости методов по сравнению с его вариантом. Действительно, проверка условия (10) для метода Госпера превращается в проверку условия

$$1 < \text{НОД}(m, z), \quad \text{где} \quad z = \prod_{a_i \in M(n)} (a_n - a_i),$$

величина $z$ должна быть пересчитана после каждого нового вычисления элемента $a_n$. Последнее обстоятельство существенно усложняет алгоритм и увеличивает его практическую трудоёмкость.

В заключение заметим, что алгоритм Ниваша вообще не может быть применён для решения задачи факторизации. Поскольку нам неизвестен делитель $p$, мы не можем в явном виде определить множество $\mathcal{M}$ и задать на нём отношение упорядоченности.

## 5.2. Решение задачи дискретного логарифмирования

Пусть $G$ — произвольная абелева группа и $a$ — её образующий элемент порядка $m > 1$. Под задачей дискретного логарифмирования мы понимаем задачу определения вычета $x \pmod m$, удовлетворяющего уравнению $a^x = b$ для некоторого заданного $b \in G$. Наиболее известными группами, для которых задача дискретного логарифмирования носит практический интерес, являются мультипликативная группа конечного простого поля и группа точек эллиптической кривой.

Как это ни странно, вопрос об использовании методов поиска длин циклов в последовательностях для решения задачи дискретного логарифмирования изучен значительно хуже, чем для задачи факторизации.

В [13] Дж. Поллард, основываясь на методе Флойда, предложил алгоритм для решения задачи дискретного логарифмирования в мультипликативной группе $\mathbb{F}_p^* = \{1, 2, \ldots, p-1\}$ для произвольного простого числа $p > 2$. Метод Полларда может быть легко модифицирован так, чтобы в нём использовался любой из описанных нами алгоритмов поиска циклов в последовательностях: алгоритм Брента, Госпера или Ниваша. Кроме того, в статье [12] был предложен параллельный вариант метода Полларда—Флойда. В настоящее время для решения задачи дискретного логарифмирования в мультипликативной группе $\mathbb{F}_p^*$ известны более эффективные алгоритмы, чем метод Полларда—Флойда (см. [7, 8]).

Для задачи дискретного логарифмирования на эллиптической кривой произвольного вида подобное утверждение неверно. До сих пор не известен алгоритм, имеющий трудоёмкость ниже, чем метод согласования, предложенный в 1962 году А. О. Гельфондом. Поскольку этот метод накладывает большие требования на объём используемой памяти, он не применяется на практике при больших значениях порядка группы точек. Поэтому использование методов, основанных на поиске длин циклов в последовательностях, является единственным способом находить решение на практике.

Рассмотрим задачу дискретного логарифмирования на эллиптической кривой, заданной в проективной форме сравнением

$$x_2^2 x_3 \equiv x_1^3 + c_4 x_1 x_3^2 + c_6 x_3^3 \pmod p, \quad 4c_4^3 + 27c_6^2 \not\equiv 0 \pmod p. \tag{11}$$

Пусть $P = (x_1 : x_2 : x_3)$ — точка кривой (11), порядок которой равен $m$. Мы будем искать величину $x$, удовлетворяющую равенству

$$xP = \underbrace{P + \ldots + P}_{x \text{ раз}} = Q, \quad 0 \leqslant x < m, \tag{12}$$

для некоторой точки $Q = (y_1 : y_2 : y_3)$, принадлежащей подгруппе, порождённой точкой $P$.

Предложим следующий алгоритм поиска неизвестного $x$, основанный на алгоритме Госпера поиска циклов в последовательностях. В качестве множества $\mathcal{M}$ возьмём подгруппу, порождённую точкой $P$. Разобьём множество $\mathcal{M}$ на $s$ непересекающихся интервалов, $\mathcal{M} = \bigcup\limits_{i=1}^{s} J_i$, и определим отображение $f$

следующим образом:

$$f(R) = R + \gamma_i P + \omega_i Q, \quad \text{если} \quad R \in J_i, \quad 0 < \gamma_i, \omega_i < m,$$

где величины $\gamma_i$, $\omega_i$ — заранее выбранные натуральные константы.

Разбиение множества $\mathcal{M}$ можно производить различными способами. Можно, например, относить к множеству $J_i$ те точки $(x_1 : x_2 : x_3)$ кривой (11), для которых $i \equiv x_1 \pmod{s}$. Число разбиений $s$, в отличие от оригинального алгоритма Полларда, должно быть велико. Для мультипликативной группы это было замечено в работе Эдлин Теске [15], а для эллиптических кривых подтверждено автором при проведении практических экспериментов. Мы считаем, что значение величины $s$ должно быть не менее 500.

Если в качестве начальной точки $R_0$ выбрать $\alpha_0 P + \beta_0 Q$ для некоторых случайных натуральных величин $\alpha_0$, $\beta_0$, то каждый элемент последовательности $R_0, R_1, \ldots$ будет удовлетворять соотношению

$$R_n = f(R_{n-1}), \quad R_n = \alpha_n P + \beta_n Q, \quad n = 1, 2, \ldots,$$

для некоторых натуральных величин $\alpha_n$, $\beta_n$. Когда алгоритм Госпера найдёт два совпадающих элемента последовательности $R_0, R_1, \ldots$, мы, учитывая равенство (12), можем записать

$$\alpha_n P + \beta_n x P = R_n = R_t = \alpha_t P + \beta_t x P$$

для некоторых индексов $n$, $t$. Последнее равенство позволяет выразить неизвестное $x$:

$$x \equiv \frac{\beta_n - \beta_t}{\alpha_t - \alpha_n} \pmod{m}. \tag{13}$$

Для реализации алгоритма требуется массив $T$, каждый элемент которого $T[n]$ хранит в себе координаты точки $R_n$ и коэффициенты $\alpha_n$, $\beta_n$, выражающие эту точку через исходные точки $P$, $Q$.

В ходе подготовки статьи автором совместно с Ф. А. Синицыным была разработана компьютерная программа, позволившая получить ряд численных результатов. Мы рассмотрели кривую

$$x_2^2 x_3 \equiv x_1^3 + 2000087865367749360 8 x_1 x_3^2 + \\ + 1839744934859799462 4 x_3^3 \pmod{2094641975286000090 1}$$

и точки $P = (8965431374621232772, 6559412994221936909, 1)$ и $Q = -P$. Порядок группы точек данной кривой равен 20946419760089834350, порядок точки $P$ равен 56774958479.

В ходе выполнения алгоритма логарифмирования после 435287 итераций вычисления функции $f$ было найдено совпадение двух точек

$$2450565259Q + 35503301414P = 27039682333Q + 3317460009P,$$

что позволило найти неизвестное значение $x = 56774958478$. Время работы программы на ПЭВМ с процессором AMD Athlon X64 составило не более 20 секунд.

**Алгоритм 5**: дискретное логарифмирование методом Госпера

**Вход**: Точки $P$, $Q$ кривой (11), связанные соотношениями (12), $\operatorname{ord} P = m$. Кроме того, определено отображение $f$.

**Выход**: Значение величины $x$, удовлетворяющей (12).

1 Определить значения: $R \leftarrow R_0$, $n \leftarrow 1$, $t \leftarrow 1$ и $T[0] \leftarrow R_0$.
2 Вычислить $R \leftarrow f(R)$.
3 **Для всех** $i = t-1, \ldots, 1, 0$ **выполнять**
4  **Если** *выполнено равенство* $T[i] = R$, **то**
5  определить величины $\alpha_n$, $\beta_n$ для $R$ и $\alpha_t$, $\beta_t$ для $T[i]$;
6  используя (13), определить $x$ и завершить алгоритм.
7  **конец**
8 **конец**
9 Вычислить $n \leftarrow n+1$ и $k \leftarrow \nu_2(n)$.
10 **Если** $k = t$, **то** вычислить $t = t+1$.
11 Определить $T[k] = R$ и вернуться на шаг 2.

Мы рассмотрели не все возможные приложения алгоритмов поиска циклов в последовательностях. Алгоритмы Брента и Госпера, а также алгоритм Ниваша могут быть использованы для поиска коллизий у произвольной функции хеширования, а также для анализа криптографических свойств блочных шифров аналогично тому, как это показано в работе [4]. К сожалению, подробное рассмотрение данных задач и методов их решения выходит за рамки настоящей статьи.

# Литература


[1] Колчин В. Ф. Случайные графы. — М.: Физматлит, 2004.

[2] Уоррен Г. С. Алгоритмические трюки для программистов. — М.: Вильямс, 2003.

[3] Beeler M., Gosper R. W., Schroeppel R. HACMEM. — MIT Artifical Intelligence Laboratory AIM 239, 1972.

[4] Biham E. New techniques for cryptanalysis of hash functions and improved attacks on Snefru // Fast Software Encryption: 15th International Workshop, FSE 2008, Lausanne, Switzerland, 2008.

[5] Brent R. P. An improved Monte Carlo factorization algorithm // BIT. — 1980. — Vol. 20. — P. 176—184.

[6] Cohen H. A Course in Computational Algebraic Number Theory. — Berlin: Springer, 1996.

[7] Gordon D. Discrete logarithms in $F_p$ using the number field sieve // SIAM J. Discrete Math. — 1993. — Vol. 6. — P. 124—138.

[8] Joux A., Lercier R. Improvements to the general number field sieve for discrete logarithms in prime fields // Math. Comp. — 2003. — Vol. 72, no. 242. — P. 953—967.

[9] Knuth D. The Art of Computer Programming. Vol. I. Fundamental Algorithms. — Addison-Wesley, 1969.

[10] Knuth D. The Art of Computer Programming. Vol. II. Seminumerical Algorithms. — Addison-Wesley, 1969.

[11] Nivash G. Cycle detecting using a stack // J. Inform. Process. Letters. — 2004. — Vol. 90, no. 3.

[12] Van Oorschot P. C., Wiener M. J. Parallel collision search with cryptanalytic applications // J. Cryptology. — 1999. — Vol. 12. — P. 1—28.

[13] Pollard J. M. A Monte Carlo method for factorization // BIT. — 1975. — Vol. 15. — P. 331—334.

[14] Sedgewick R., Szymansky T. G., Yao A. C. The complexity of finding cycles in periodic functions // SIAM J. Comput. — 1982. — Vol. 11, no. 2. — P. 376—390.

[15] Teske E. On random walks for Pollard's rho method // Math. Comp. — 2001. — Vol. 70. — P. 809—825.