Алгоритм AKS проверки чисел на простоту и поиск констант генераторов псевдослучайных чисел

Л. Бараш*

Институт теоретической физики им. Л. Д. Ландау РАН, 142432, Черноголовка, Россия

Аннотация

В работе излагается и анализируется недавно предложенный алгоритм AKS проверки чисел на простоту. Также анализируется возможность применения этого алгоритма для эффективного поиска новых констант генераторов случайных чисел, обсуждается поиск Мерсеновских экспонент и примитивных триномов. На сегодня алгоритм AKS не дает существенных преимуществ, тем не менее он может оказаться перспективным.

1. Алгоритм AKS проверки на простоту

С древних времен математики были очарованы задачами, имеющими отношение к простым числам. Простые числа интенсивно исследовались еще со времен Евклида и Эратосфена (пример см. ниже). Несмотря на то, что они просты для описания и понимания, многие их важные свойства ускользали от людей, а кое-что неизвестно до сих пор.

С 70-х годов двадцатого века дополнительно выяснилось, что многие свойства простых чисел важны и в криптографии. Например, самая известная реализация широко используемого в мире алгоритма шифрования RSA [3] основана на алгоритмической сложности задачи разложения больших составных чисел на простые множители. Если бы были известны эффективные алгоритмы разложения больших чисел на множители, большая часть электронных коммуникаций по всему миру могла бы стать незащищенной.

Естественно, развитие новых направлений криптографии стимулировало интерес ученых к алгоритмическим проблемам теории чисел.

Как известно, теория сложности вычислений – математическая дисциплина, которая классифицирует задачи по степени сложности их решения. Пусть x – входное слово из n бит, L – некоторое подмножество множества Σ всех конечных строк из нулей и единиц (такие множества $L \subset \Sigma$ в теории сложности называют языками). Задача определения истинности $x \in L$ относится к классу P (вычислима за полиномиальное время), если число операций, необходимое для ее решения, ограничено некоторой степенью числа n (более строго, класс P – это класс всех языков, для которых существует машина Тьюринга, решающая задачу определения истинности $x \in L$ за число шагов, ограниченное числом $f(n)$, где f – некоторый полином). Язык L относится к классу NP, если существует предикат $P(x, y) : \Sigma \times \Sigma \to \{0, 1\}$, вычислимый за полиномиальное время, такой что для всякого входного слова длины n можно угадать некоторую строку полиномиальной от n длины и затем с помощью предиката убедиться в правильности догадки. Другими словами, класс NP – это класс задач, решаемых за полиномиальное время на машине Тьюринга, дополненной гипотетической способностью «угадывания». Ясно, что P$\subseteq$NP. Является ли это включение строгим – одна из самых известных нерешенных задач математики. Большинство специалистов считают, что оно строгое (так называемая гипотеза P$\neq$NP).

В криптографических алгоритмах одной из важных задач является проверка на простоту, т.е. умение быстро отличить простое число от составного. Задача проверки на простоту легче, чем задача разложения на множители, и почти всегда считалось, что она лежит в классе P, несмотря на то, что детерминированного алгоритма, работающего за полиномиальное время до последнего времени известно не было.

Кстати, эта задача – одна из фундаментальных задач, имеющих отношение к простым числам. Самый древний известный верный алгоритм проверки на простоту – решето Эратосфена (240 г. до н.э.). Конечно, он работает за экспоненциальное время. В 17 веке Ферма доказал следующее утверждение:

Утверждение (Малая теорема Ферма). Если p – простое, a не делится на p, то

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

*E-mail: barash@itp.ac.ru

Доказательство: Это сразу же следует из свойств группы $\mathbb{Z}_p^*$, состоящей из чисел $1, 2, \ldots, p-1$ с операцией умножения по модулю p. Действительно, порядок любого элемента конечной группы должен делить порядок группы.

Проверка (1) для заданного числа p не требует больших вычислений, но, к сожалению, утверждение, обратное Малой теореме Ферма, неверно. Более того, имеются составные числа p, обладающие свойством (1) для любого целого a с условием НОД$(a, p) = 1$. Они называются числами Кармайкла, и в 1994 году было показано, что таких чисел бесконечно много [7].

Однако Малая теорема Ферма является отправным пунктом при построении нескольких очень эффективных проверок на простоту. В 1976 году Миллер, используя это свойство, придумал детерминированный алгоритм проверки на простоту, имеющий сложность $O(\log^3 n)$. Однако справедливость его результата зависит от недоказанной в настоящее время расширенной гипотезы Римана [6], т.е. правильность работы этого алгоритма не является доказанной.

Самые эффективные алгоритмы проверки на простоту, известные сегодня, являются вероятностными (в отличие от алгоритмов, обсуждаемых выше, которые детерминированы). Это означает, что за любое конечное время либо алгоритм выявит, что число n составное, либо не выявит. Вероятность того, что составное число не будет выявлено за время t, обычно не превосходит $e^{-\alpha t}$. Таким образом, с помощью вероятностного алгоритма нельзя получить доказательство простоты числа n. Знать, что n простое, можно лишь с любой, заданной заранее, близкой к единице, вероятностью. Именно такие алгоритмы и используются на практике.

В августе 2002 года появился первый алгоритм проверки чисел на простоту, работающий детерминированно, за полиномиальное время и правильность работы которого полностью доказана (M.Agrawal, N.Kayal, N.Saxena, см. [1]). Это – важный прорыв. Дополнительным достоинством их подхода является простота и элегантность их работы. Недостатком является то, что этот результат имеет в основном теоретическое значение. Практического распространения он пока не получил, поскольку работает намного медленнее лучших вероятностных алгоритмов.

Рассмотрим подробнее основные идеи и принципы, лежащие в основе алгоритма AKS.

В основе этого теста на простоту лежит следующее тождество:

Утверждение. Пусть НОД$(a, n) = 1$. Тогда n – простое тогда и только тогда, когда

$$(x - a)^n \equiv (x^n - a) \pmod{n} \tag{2}$$

Доказательство: $(x - a)^n = \sum_{i=0}^{n} C_n^i x^i (-a)^{n-i}$, однако мы знаем, что для простого n выполняется $C_n^i \equiv 0 \pmod{n}$ при $1 \leqslant i \leqslant n - 1$. Кроме того, $a^n \equiv a \pmod{n}$. Отсюда получаем (2).

Если же n составное, а q – его простой делитель (q^k входит в разложение n на простые множители), то коэффициент при x^q в левой части выражения (2), не равен нулю в $\mathbb{Z}_n$ (он равен $C_n^q(-a)^{n-q} \pmod{n}$, а C_n^q не делится на q^k).

Конечно, простая проверка выполнения (2) для заданного числа n займет экспоненциальное время. Действительно, в некоторых случаях для этого придется посчитать n коэффициентов в левой части (2). Однако, можно существенно упростить вычисления, если рассмотреть остаток от деления выражения (2) на некоторый полином вида $x^r - 1$:

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

Запись $\pmod{n, x^r - 1}$ означает, что мы рассматриваем кольцо, состоящее из многочленов вида $a_0 + a_1 x + \ldots + a_{r-1} x^{r-1}$, где все $a_i \in \mathbb{Z}_n$. Очевидно, из (2) следует, что для простых чисел n равенство (3) всегда выполняется, однако некоторые составные n тоже могут удовлетворять (3) для нескольких значений параметров a и r. Оказывается, справедлива следующая

Теорема AKS. Пусть $n \geqslant 2$; n – целое; q, r – простые числа, причем [1]

а) $\forall m \in \{1, 2, \ldots, r\} : \text{НОД}(m, n) = 1$

б) $q \mid (r - 1)$

в) $q \geqslant 4\sqrt{r} \log n$

г) $n^{(r-1)/q} \not\equiv 1 \pmod{r}$

д) $\forall a \in \{1, \ldots, \lfloor 2\sqrt{r} \log n \rfloor + 1\} : (x - a)^n \equiv (x^n - a) \pmod{n, x^r - 1}$

Тогда n – степень простого числа.

Эта теорема приводит к искомому алгоритму (см. рис. 1). В качестве q достаточно взять наибольший простой делитель $(r - 1)$.

---

[1] Ниже в тексте статьи $\log \equiv \log_2$. Обозначение $b \mid a$ (также как и $a \vdots b$), где a и b – целые числа, означает, что a делится на b без остатка.

```
r=2;
while(r<n){
    if(НОД(r,n)≠ 1) output составное;
    if(r - простое число, r > 2){
       q = наибольший простой делитель у (r − 1);
       if((q ⩾ 4√r log n) and ( n^{(r−1)/q} ≢ 1 (mod r))) break;
    }
    r ← r + 1;
}
for a = 1 to (⌊2√r log n⌋ + 1)
    if((x − a)^n ≡ (x^n − a) (mod x^r − 1, n) ) output составное;
if(n = a^b;  a, b - целые; a, b ⩾ 2) output составное;
    else output простое;
```

Рис. 1. Алгоритм AKS

Возникает естественный вопрос, найдется ли такое r, которое удовлетворяет всем условиям теоремы AKS и, если найдется, долго ли мы его будем искать. Опираясь на известные из теории чисел оценки числа простых чисел, меньших n и числа простых чисел $p < n$, таких что $p-1$ имеет не слишком маленькие простые делители, AKS легко доказали следующее

Утверждение. Существуют константы $c_2 > c_1 > 0$ такие, что для любого достаточно большого n существует простое r, такое что

а) $c_1 \log^6 n < r < c_2 \log^6 n$

б) $(r-1)$ имеет простой делитель q, где $q \geqslant 4\sqrt{r}\log n$

в) $n^{(r-1)/q} \not\equiv 1 \pmod r$

Отсюда видно, что цикл while благополучно завершается, в худшем случае за $O(\log^6 n)$ итераций.

Итак, новое, что придумали AKS, и что позволило воплотить в жизнь их алгоритм – это теорема AKS и ее доказательство. Как же получилось, что эта теорема оказалась верна? Рассмотрим более подробно основные идеи ее доказательства, которое представляет собой, наверное, самую элегантную часть статьи AKS.

Вкратце напомним необходимые сведения из алгебры. Пусть $h(x)$ – неприводимый полином степени d, лежащий в кольце $\mathbb{Z}_p[x]$ (как известно, $\mathbb{Z}_p[x]$ содержит все многочлены с коэффициентами, лежащими в $\mathbb{Z}_p$, а неприводимость полинома означает, что его нельзя представить в этом кольце в виде произведения двух множителей меньшей степени). Рассмотрим факторкольцо $\mathbb{Z}_p[x]/(h(x))$. Напомним, что достаточно считать, что оно содержит все многочлены вида $a_0 + a_1 x + \ldots + a_{d-1}x^{d-1}$, где все $a_i \in \mathbb{Z}_p$, причем умножение многочленов осуществляется по модулю многочлена $h(x)$. Легко видеть, что если $h(x)$ – неприводимый полином, а p – простое число, то $\mathbb{Z}_p[x]/(h(x))$ – поле. Очевидно, оно содержит p^d элементов. Это поле называется полем Галуа. Несложно показать (см., например, [9]), что любое конечное поле по сути является полем Галуа.

В поле Галуа, как и в любом поле из многочленов, можно утверждать, что любой многочлен степени d имеет единственное разложение на неприводимые многочлены (и имеет не более d корней). Легко видеть, что в случае многочленов, лежащих просто в кольце, а не в поле, это утверждение неверно.

Можно показать, что если p, r – простые числа, а $d = o_r(p)$ – порядок числа p по модулю r (т.е. d – наименьшее число, такое что $p^d \equiv 1 \pmod r$), то в поле $\mathbb{Z}_p$ многочлен $(x^r-1)/(x-1)$ распадается на неприводимые полиномы каждый степени d (см. [1]).

Итак, пусть все, указанное в условии теоремы AKS, верно, и пусть $l = \lfloor 2\sqrt{r}\log n \rfloor + 1$. Тогда

Лемма. У числа n существует простой множитель p такой что $q \mid d = o_r(p)$.

Доказательство: Из условия теоремы, $n^{(r-1)/q} \not\equiv 1 \pmod r$, НОД$(n, r) = 1$; из Малой теоремы Ферма, $n^{r-1} \equiv 1 \pmod r$. Следовательно, $(r-1)$ делится на $o_r(n)$, но $\frac{r-1}{q}$ не делится на $o_r(n)$. Таким образом, $q \mid o_r(n)$.

Пусть $n = p_1^{k_1} \ldots p_m^{k_m}$ – разложение числа n на простые множители, пусть $f =$НОК$(o_r(p_i))$, тогда $\forall i : f = \alpha_i o_r(p_i)$, где $\alpha_i \in \mathbb{N}$, поэтому

$$n^f = p_1^{k_1\alpha_1 o_r(p_1)} \ldots p_m^{k_m\alpha_m o_r(p_m)} \equiv 1 \pmod r \;\Rightarrow f \vdots o_r(n) \vdots q \Rightarrow \exists i : o_r(p_i) \vdots q$$

Итак, искомое $p = p_i$ найдено и лемма доказана. Мы знаем, что существует неприводимый полином $h(x)$ степени $d = o_r(p)$, входящий в разложение $x^r - 1$ на неприводимые многочлены в $\mathbb{Z}_p$. При этом из условия $(x-a)^n \equiv$

$(x^n-a)\ (\mathrm{mod}\ x^r-1,n)$ следует условие $(x-a)^n \equiv (x^n-a)\ (\mathrm{mod}\ h(x),p)$. Таким образом, мы можем рассмотреть все полиномы в поле $\mathbb{Z}_p[x]/(h(x))$.

Лемма. Пусть $h(x)$ – неприводимый в поле $\mathbb{Z}_p$ полином степени $d = o_r(p)$. Тогда в поле $\mathbb{Z}_p[x]/(h(x))$, группа G, генерируемая l многочленами $(x-a)$, где $a \in \{1,\dots,l\}$, циклическая и количество ее элементов больше, чем $\left(\frac{d}{l}\right)^l$.

Доказательство: G – циклическая группа, т.к. G – подгруппа циклической группы $(\mathbb{Z}_p[x]/(h(x)))^*$ (мультипликативная группа любого конечного поля циклическая (см. [9])). По условию $r > q > l$, кроме того, из условия а) теоремы следует, что $p > r$, поэтому все $a \in \{1,2,\dots,l\}$ различны по модулю p. Поскольку в поле $\mathbb{Z}_p[x]$ разложение любого полинома на неприводимые множители единственно, то все элементы вида $(x-1)^{\alpha_1}\dots(x-l)^{\alpha_l}$, где $\alpha_i \geqslant 0 \forall i$, различны в $\mathbb{Z}_p[x]$. Те из них, степень которых меньше d, т.е. для которых $\alpha_1+\ldots+\alpha_l \leqslant (d-1)$, будут различимы и в $\mathbb{Z}_p[x]/(h(x))$. Сколько всего таких элементов? Рассмотрим диаграмму, состоящую из $d-1$ точек и l перегородок между ними. α_i – число точек между i-й и $(i+1)$-ой перегородкой. Точки левее первой перегородки, если таковые есть, не учитываются. Так, для $d=8, l=3$ диаграмма « $\cdots||\cdots|\cdot$ » означает, что $\alpha_1=0, \alpha_2=3, \alpha_3=1$. Каждой диаграмме соответствует набор чисел $\{\alpha_1\dots\alpha_l\}$ такой, что $\sum\alpha_i \leqslant (d-1)$, и наоборот. Всего число мест, на которых находятся точки или перегородки, равно $l+d-1$, поэтому всего существует ровно C^l_{l+d-1} разных диаграмм. Поскольку $C^l_{l+d-1} = \frac{(l+d-1)(l+d-2)\dots d}{l!} > \left(\frac{d}{l}\right)^l$, то лемма доказана.

Поскольку $d \vdots q \geqslant 2l$, то $d \geqslant 2l$, т.е. число элементов в группе G больше, чем $2^l = n^{2\sqrt{r}}$. Пусть $g(x)$ – генерирующий элемент в G, порядок которого равен числу элементов группы. Пусть

$$I_{g(x)} = \{m | g(x)^m \equiv g(x^m)\ (\mathrm{mod}\ x^r-1,p)\} \tag{4}$$

Лемма. Множество $I_{g(x)}$ замкнуто относительно умножения

Доказательство: Действительно, $m_1, m_2 \in I_{g(x)} \Rightarrow g(x^{m_1})^{m_2} \equiv g(x^{m_1 m_2})\ (\mathrm{mod}\ x^{m_1 r}-1,p) \Rightarrow g(x^{m_1})^{m_2} \equiv g(x^{m_1 m_2})\ (\mathrm{mod}\ x^r-1,p) \Rightarrow g(x)^{m_1 m_2} \equiv (g(x)^{m_1})^{m_2} \equiv g(x^{m_1})^{m_2} \equiv g(x^{m_1 m_2})\ (\mathrm{mod}\ x^r-1,p) \Rightarrow m_1 m_2 \in I_{g(x)}$.

Лемма. Пусть порядок элемента $g(x)$ в $\mathbb{Z}_p[x]/(h(x))$ равен o_g. Пусть $m_1, m_2 \in I_{g(x)}$. Тогда $m_1 \equiv m_2\ (\mathrm{mod}\ r) \Rightarrow\ m_1 \equiv m_2\ (\mathrm{mod}\ o_g)$.

Доказательство: Пусть $m_2 = m_1 + kr$, $k \geqslant 0$. Поскольку $x^{m_2} = x^{m_1}x^{kr} \equiv x^{m_1}\ (\mathrm{mod}\ x^r-1)$, то $x^{m_2} \equiv x^{m_1}\ (\mathrm{mod}\ h(x))$. Учтем теперь, что $m_2 \in I_{g(x)}$. Все последующие соотношения верны лишь в поле $\mathbb{Z}_p[x]/(h(x))$: $g(x)^{m_1}g(x)^{kr} \equiv g(x)^{m_2} \equiv g(x^{m_2}) \equiv g(x^{m_1}) \equiv g(x)^{m_1}$. Поскольку наше множество – поле, и $g(x) \neq 0$, то отсюда следует, что $g(x)^{kr} \equiv 1 \Rightarrow\ kr \equiv 0\ (\mathrm{mod}\ o_g) \Rightarrow\ m_1 \equiv m_2\ (\mathrm{mod}\ o_g)$.

Лемма. $p \in I_{g(x)}$

Доказательство: Достаточно показать, $f(x) = a_0 + \ldots + a_d x^d$ – произвольный полином с целыми коэффициентами, а p – простое число, то $f(x)^p \equiv f(x^p)\ (\mathrm{mod}\ p)$. Действительно, рассмотрим коэффициент при x^i в выражении $f(x)^p = \underbrace{f(x)\dots f(x)}_{p\ \text{множителей}}$. Он равен

$$\sum_{\substack{i_1+2i_2+\ldots+di_d=i \\ i_0+i_1+\ldots+i_d=p}} a_0^{i_0}a_1^{i_1}\dots a_d^{i_d}\frac{p!}{i_0!\dots i_d!}$$

Во всех случаях, когда $\forall j : i_j \neq p$, эта сумма делится на p. Если же $i_j = p$, то $i = pj$, т.е. коэффициент при x^i равен $a_j^p \equiv a_j\ (\mathrm{mod}\ p)$. Это дает нам то, что требовалось: $f(x)^p \equiv a_0 + a_1x^p + \ldots + a_dx^{dp}\ (\mathrm{mod}\ p)$.

Доказательство теоремы: Из условия д) следует, что $g(x)$ (который всего лишь является произведением l многочленов вида $(x-a)$) удовлетворяет соотношению $g(x)^n \equiv g(x^n)\ (\mathrm{mod}\ x^r-1,p)$. Это означает, что $n \in I_{g(x)}$. Чуть выше мы показали, что $p \in I_{g(x)}$. Очевидно, $1 \in I_{g(x)}$. Рассмотрим множество $E = \{n^i p^j | 0 \leqslant i,j \leqslant \lfloor\sqrt{r}\rfloor\} \subset I_{g(x)}$. Число элементов в E: $|E| = (1+\lfloor\sqrt{r}\rfloor)^2 > r$, поэтому существуют два различных элемента $n^{i_1}p^{j_1}$ и $n^{i_2}p^{j_2}$ такие, что $n^{i_1}p^{j_1} \equiv n^{i_2}p^{j_2}\ (\mathrm{mod}\ r) \Rightarrow\ n^{i_1}p^{j_1} \equiv n^{i_2}p^{j_2}\ (\mathrm{mod}\ o_g) \Rightarrow\ n^{i_1-i_2} \equiv p^{j_1-j_2}\ (\mathrm{mod}\ o_g)$. Здесь $o_g \geqslant n^{2\sqrt{r}}$, $n^{|i_1-i_2|} < n^{\sqrt{r}}$, $p^{|j_1-j_2|} < n^{\sqrt{r}}$, поэтому равенство по модулю o_g превращается в настоящее равенство $n^{i_1-i_2} = p^{j_1-j_2}$. Это означает, что $n = p^k$ для некоторого $k \in \mathbb{N}$. Теорема доказана.

Какова же вычислительная сложность алгоритма AKS? Действительно ли он всегда работает за полиномиальное время?

Утверждение. Сложность алгоритма AKS – не более $O(\log^{19} n)$.

Доказательство: Как мы выяснили, цикл while повторяется максимум $O(\log^6 n)$ раз.

- Известно, что алгоритм Евклида вычисления НОД занимает $O(\log^3 n)$ операций умножения.
- Проверку числа r на простоту производим следующим образом – для каждого числа $p \in \{1,2,\dots,\sqrt{r}\}$ мы находим «школьным методом» $r\ (\mathrm{mod}\ p)$ за $O(\log^2 r)$ операций. Следовательно, проверка r на простоту занимает $O(\sqrt{r}\log^2 r)$ операций.

- Таким же способом находим за $O(\sqrt{r}\log^2 r)$ операций наибольший простой делитель у числа $(r-1)$.
- Вычисление $a^b \pmod r$ занимает $O(\log b \log^2 r + \log^2 n)$ операций. Действительно, вычисление $a \pmod r$ занимает $\log^2 n$ операций, где $n = \max(a, r)$; возведение в квадрат числа, меньшего r, занимает $\log^2 r$ операций; повторять возведение в квадрат и редукцию по модулю r мы будем $\log b$ раз.
- В частности, вычисление $n^{(r-1)/q} \pmod r$ занимает $O(\log^2 n + \log^3 r)$ операций
- Итак, каждая итерация цикла while займет $O(\log^3 n + \sqrt{r}\log^2 r) = O(\log^3 n(\log\log n))$ операций. Весь цикл while имеет сложность $O(\log^9 n(\log\log n))$
- Итерация цикла for занимает $O(r^2 \log^3 n)$ операций. Действительно, вычисление «столбиком» произведения двух чисел, меньших n, занимает $\log^2 n$ операций. Чтобы умножить друг на друга два многочлена степени, меньшей r, по модулю $(x^r - 1, n)$, нужно произвести $O(r^2)$ простых умножений. Возводить в квадрат по модулю $(x^r - 1, n)$ нужно $\log n$ раз.
- Следовательно, весь цикл for занимает $O(r^{5/2}\log^4 n) = O(\log^{19} n)$ операций.
- Чтобы выяснить, выполняется ли для каких-нибудь целых $a, b \geqslant 2$ равенство $n = a^b$, мы для каждого $b = 2, 3, \ldots, \log n$ вычисляем $n^{1/b}$ методом деления отрезка пополам. Это означает, что нам придется $O(\log^2 n)$ раз вычислить a^b, т.е. произвести $O(\log^2 n \log b) = O(\log^2 n \log\log n)$ возведений в квадрат. Всего будет $O(\log^4 n \log\log n)$ операций.

Итак, сложность алгоритма AKS составляет $O(\log^{19} n)$. Можно показать, что использование улучшенных алгоритмов для умножения целых чисел и быстрого преобразования Фурье для умножения полиномов позволяет уменьшить сложность алгоритма вплоть до $O(\log^{12} n(\log\log n)^c)$, где c – некоторая константа.

А еще AKS сформулировали очень интересную гипотезу. Ее истинность позволила бы намного улучшить эффективность алгоритма (она станет равной $O(\log^3 n \log\log n)$) и сделать его практическое применение существенно более реальным.

Гипотеза. Если НОД$(n, r) = 1$ и $(x-1)^n \equiv (x^n - 1) \pmod{x^r - 1, n}$, то либо n – степень простого числа, либо $n^2 \equiv 1 \pmod r$.

Удивительно, что до сих пор не известно, истинно ли столь простое на первый взгляд утверждение о многочленах. Можно, кстати, переформулировать эту гипотезу в более «наглядно-школьной» формулировке:

Утверждение. По кругу написано r чисел. В начальный момент все эти числа равны нулю, кроме 1 и -1, стоящих рядом. На каждом шаге вместо чисел $(a_0, a_1, \ldots, a_{r-1})$, стоящих по кругу, записывают разности $(a_{r-1} - a_0, a_0 - a_1, \ldots, a_{r-2} - a_{r-1})$. При этом все числа берутся по модулю n. Известно, что на n-ом шаге вновь все числа равны нулю, кроме двух, равных 1 и -1 (-1 стоит на том же месте, что и в начале). Показать, что 1 и -1 вновь стоят рядом.

Проверено, что эта гипотеза выполняется для $r < 100$ и $n < 10^{10}$. Следовало бы проверить ее и для больших значений r, но в сети пока таких результатов нет. В статье [11] доказано несколько утверждений, которые несомненно полезны для понимания смысла гипотезы и различных путей и способов ее возможного доказательства. Кроме утверждений, взятых из статьи [11], автору известны еще несколько полезных и верных фактов, имеющих прямое отношение к данной гипотезе. Впрочем, до полного доказательства тут далеко – возможно, гипотеза и неверна. Гипотеза была сформулирована AKS за много месяцев до появления статьи [1]. Похоже, именно сложность нахождения ее доказательства привела к поиску обходного пути и к теореме AKS.

2. Генераторы псевдослучайных чисел

При проведении широкомасштабных численных экспериментов (и вообще любых вычислений методом Монте-Карло) необходимо иметь возможность выборки случайных чисел, равномерно распределенных между 0 и 1. Для этих целей используются генераторы случайных чисел (Random Number Generators). Конечно, RNG – это простая программа, которая выдает некоторую последовательность чисел. Эта последовательность выглядит случайной и обладает важными свойствами случайной последовательности, но на самом деле произведена детерминистским способом и является *псевдослучайной*.

В настоящее время наиболее распространены генераторы псевдослучайных чисел двух классов: линейно-конгруэнтные генераторы и генераторы на сдвиговых регистрах.

Линейно-конгруентный метод (ЛКМ) был предложен Лемером в 1948 году. Последовательность псевдослучайных чисел $(x_0, x_1, \ldots, x_n, \ldots)$ вычисляется после задания начального значения x_0 по формуле

$$x_{n+1} = (ax_n + c) \pmod M \tag{5}$$

Здесь a – множитель, c – приращение, M – модуль. Очевидно, максимально возможный период такой последовательности не может превосходить модуля M. Оказывается, выполняется следующее утверждение:

Теорема. Длина периода линейной конгруэнтной последовательности равна M тогда и только тогда, когда

- c и M взаимно просты
- $p \mid (a-1)$ для любого простого $p \mid M$
- $4 \mid (a-1)$, если $4 \mid M$

Линейно-конгруэнтный метод имеет два существенных недостатка. Первый, это алгоритмическое ограничение на максимальный период, который не может превосходить машинную длину целого числа (при этом последовательность длины $2^{32} \approx 4 \cdot 10^9$ исчерпывается на современных рабочих станциях в течение нескольких секунд). Второй недостаток был обнаружен сравнительно недавно. Он заключается в том, что при использовании ЛКМ генераторов, например, для случайного выбора точки в k-мерном пространстве (k последовательных случайных чисел задают ее координаты), точки заполняют не все k-мерное пространство, а лишь некоторое семейство $(k-1)$-мерных гиперповерхностей (см. [10]).

Второй класс генераторов случайных чисел – это генераторы, в основе которых лежат сдвиговые регистры. Как мы увидим, сдвиговые регистры не имеют таких ограничений по длине периода, как линейно-конгруэнтные алгоритмы. Состоянием сдвигового регистра называется набор из r бит $(a_{n-r}, a_{n-r+1}, \ldots, a_{n-1})$. В следующий момент времени состоянием сдвигового регистра будет $(a_{n-r+1}, a_{n-r+2}, \ldots, a_n)$. При этом

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_r a_{n-r} \pmod 2 \tag{6}$$

Здесь $c_i \in \{0,1\}, i \in \{1,2,\ldots,r\}$; r выбрано так, что $c_r = 1$. Ясно, что последовательность состояний сдвигового регистра периодична, причем ее период $p \leqslant (2^r - 1)$ (число возможных состояний сдвигового регистра, кроме состояния, состоящего из одних нулей, равно $2^r - 1$).

Определение. Генерирующей функцией сдвигового регистра (6) называется ряд $G(x) = \sum_{n=0}^{\infty} a_n x^n$. Характеристическим полиномом сдвигового регистра (6) называется многочлен $f(x) = 1 - \sum_{i=1}^{r} c_i x^i$.

Легко показать (см., напр., [12]), что если начальные условия задать в виде $a_{-r} = 1, a_{1-r} = a_{2-r} = \ldots = a_{-1} = 0$, то $G(x) = \frac{1}{f(x)}$.

Утверждение. Если начальные условия $a_{-r} = 1, a_{1-r} = a_{2-r} = \ldots = a_{-1} = 0$, то периодом последовательности на сдвиговом регистре будет минимальное $p \in \mathbb{N}$, такое что $f(x) \mid (1 - x^p) \pmod 2$, где $f(x)$ – характеристический полином.

Доказательство: Если p – период последовательности $\{a_i\}$, то при указанных начальных условиях $\frac{1}{f(x)} = G(x) = (a_0 + a_1 x + \ldots + a_{p-1}x^{p-1})(1 + x^p + x^{2p} + \ldots) = (a_0 + a_1 x + \ldots + a_{p-1}x^{p-1})/(1 - x^p) \Rightarrow f(x)(a_0 + a_1 x + \ldots + a_{p-1}x^{p-1}) = 1 - x^p$. Во-первых, отсюда следует, что $a_{p-r} = 1, a_{p-r+1} = \ldots = a_{p-1} = 0$ (так и должно быть, ведь p – период); во вторых, то, что требовалось: $f(x) \mid (1 - x^p) \pmod 2$.

Наоборот, если $1 - x^p$ делится на $f(x)$, то, записав частное в виде $(\alpha_0 + \alpha_1 x + \ldots + \alpha_{p-r}x^{p-r})$, получаем $G(x) = 1/f(x) = (\alpha_0 + \alpha_1 x + \ldots + \alpha_{p-r}x^{p-r})/(1 - x^p) = (\alpha_0 + \alpha_1 x + \ldots + \alpha_{p-r}x^{p-r}) + x^p(\alpha_0 + \alpha_1 x + \ldots + \alpha_{p-r}x^{p-r}) + \ldots$, т.е. p делится на период последовательности $\{a_i\}$. Если p – минимальное целое, такое, что $f(x) \mid (1 - x^p) \pmod 2$, то p – период.

Верно ли это утверждение, когда начальные условия другие? Для произвольных начальных условий будет выполняться $G(x) = \frac{g(x)}{f(x)}$, где числитель $g(x)$ имеет степень, меньшую, чем r. Таким образом, если последовательность имеет период p, то $f(x)(a_0 + a_1 x + \ldots + a_{p-1}x^{p-1}) = (1 - x^p)g(x)$. Мы приходим к следующему утверждению:

Утверждение. Если $f(x)$ неприводим в $\mathbb{Z}_2$, то период не зависит от начальных ненулевых условий.

Определение. Говорят, что последовательность на сдвиговом регистре имеет максимальную длину, если ее период $p = 2^r - 1$.

Утверждение. Если последовательность $\{a_i\}$ имеет максимальную длину, то ее характеристический полином неприводим в $\mathbb{Z}_2$.

Доказательство: Поскольку период максимален, то в какой-то момент времени состоянием сдвигового регистра будет $(100\ldots0)$. Если $f(x)$ приводим, то $f(x) = s(x)t(x)$, где порядок $s(x)$ равен r_s, а порядок $t(x)$ равен r_t. Ясно, что $r_s + r_t = r$, и можно считать, что НОД$(s(x), t(x)) = 1$. Таким образом,

$$\frac{1}{f(x)} = \frac{\alpha(x)}{s(x)} + \frac{\beta(x)}{t(x)} \tag{7}$$

Слева стоит степенной ряд, период последовательности коэффициентов которого равен $2^r - 1$. Периоды в коэффициентах степенных рядов, равных $\frac{\alpha(x)}{s(x)}$ и $\frac{\beta(x)}{t(x)}$ не больше, чем $2^{r_s} - 1$ и $2^{r_t} - 1$ соответственно. Таким образом, $2^r - 1 \leqslant$ НОК$(2^{r_s} - 1, 2^{r_t} - 1) \leqslant (2^{r_s} - 1)(2^{r_t} - 1) \leqslant 2^{r_s + r_t} - 3 = 2^r - 3$. Противоречие.

Утверждение. В $\mathbb{Z}_2$ любой неприводимый полином степени $r > 1$ делит $(1 - x^{2^r-1})$.

Доказательство: В мультипликативной группе $(\mathbb{Z}_2[x]/(P(x)))^*$, имеющей $2^r - 1$ элементов, порядок элемента x должен делить порядок группы.

Определение. Полином $P(x)$ степени $r > 1$ называется примитивным, если $P(x)$ неприводим и $x^j \not\equiv 1 \pmod{P(x)}$ $\forall j \in \{1, 2, \ldots, 2^r - 2\}$.

Отсюда ясно, что если характеристический полином примитивен, то последовательность на сдвиговом регистре имеет максимальную длину. Т.е. каждый примитивный полином степени r соответствует сдвиговому регистру максимальной длины!

Определение. Число r называют Мерсеновской экспонентой, если $(2^r - 1)$ простое.

Очевидно, если r – Мерсеновская экспонента, то, в частности, само r должно быть простым числом.

Утверждение. В случае, если r – Мерсеновская экспонента, каждый неприводимый полином $f(x)$ степени r является примитивным.

Доказательство: Действительно, пусть p – минимальное число, такое что $f(x) \mid (1 - x^p) \pmod 2$. Мы знаем, что $f(x) \mid (1 - x^{2^r-1}) \pmod 2$ и что $f(x)$ – неприводим. Если $(2^r - 1)$ делится на p, то $p = 2^r - 1$, (поскольку r – Мерсеновская экспонента), в этом случае все доказано. Если же это не так, то $(2^r - 1) = u + v$, где $v = 2^r - 1 \pmod p > 0$. Тогда $f(x) \mid (1 - x^u) \pmod 2 \Rightarrow\ f(x) \mid (1 - x^v) \pmod 2$. При этом условие $v < p$ противоречит минимальности p.

Из доказательства видно, что если p – период последовательности на сдвиговом регистре, характеристический полином которого неприводим (но не обязательно примитивен), то $(2^r - 1)$ делится на p.

Что мы получили? Превосходный способ создать быстрый и высококачественный генератор случайных чисел с гигантским периодом $2^r - 1$. Он у нас в кармане, если мы найдем достаточно большую Мерсеновскую экспоненту и неприводимые полиномы степени, равной этой Мерсеновской экспоненте. Обычно желательно использование примитивных полиномов с малым количеством ненулевых слагаемых – в этом случае эффективность генератора случайных чисел возрастает. В частности, можно использовать примитивные *триномы* – многочлены вида $x^r + x^s + 1$, где $r > s > 0$.

3. Поиск Мерсеновских экспонент

Как же выяснить, является ли число r Мерсеновской экспонентой? Ведь для этого нужно проверить число $2^r - 1$ на простоту! Какой тест на простоту следует использовать? На практике для этого используют следующую теорему.

Теорема Люка-Лемера. (1930) Пусть $M_p = 2^p - 1,\ S_1 = 4,\ S_{k+1} = S_k^2 - 2$. Тогда для того, чтобы M_p было простым, необходимо и достаточно, чтобы $S_{p-1} \equiv 0 \pmod{M_p}$.

Доказательство достаточности: Пусть $\begin{cases} w=2+\sqrt{3} \\ v=2-\sqrt{3} \end{cases} \Rightarrow \begin{cases} wv=1 \\ w+v=4 \end{cases}$. Легко видеть (по индукции), что $S_n = w^{(2^{n-1})} + v^{(2^{n-1})}$.

Пусть $S_{p-1} \equiv 0 \pmod{M_p} \Rightarrow \exists R \in \mathbb{N} : w^{(2^{p-2})} + v^{(2^{p-2})} = RM_p$. Умножая обе части этого равенства на $w^{(2^{p-2})}$ и вычитая единицу, получаем

$$w^{(2^{p-1})} = RM_p w^{(2^{p-2})} - 1 \tag{8}$$

Предположим, M_p – составное, т.е. M_p делится на q, где q – простое число и $q \leqslant \sqrt{M_p}$. Рассмотрим группу $G = \mathbb{Z}_q[\sqrt{3}]^*$, состоящую из всех чисел $(a + b\sqrt{3}) \neq 0$ таких, что $a, b \in \{0, 1, \ldots, q - 1\}$ и что каждое из них имеет обратный элемент такого же вида $(c + d\sqrt{3})$ (умножение целых чисел производится по модулю q). Ясно, что группа G непустая, т.к. $w, v \in G$. Ясно, что число элементов в G не может быть больше, чем $q^2 - 1$. Из (8) следует, что в группе G выполняется $w^{(2^{p-1})} = -1 \Rightarrow\ w^{(2^p)} = 1$. Таким образом, порядок элемента w в группе G в точности равен 2^p. Порядок элемента не может быть больше порядка группы, поэтому $2^p \leqslant (q^2 - 1) < M_p \Rightarrow 2^p < (2^p - 1)$. Противоречие. Следовательно, M_p должно быть простым.

Итак, с помощью теста Люка-Лемера можно проверить на простоту любое число вида $n = 2^p - 1$. Какова сложность этого алгоритма? Легко видеть, что даже без использования быстрого преобразования Фурье в алгоритме умножения чисел, сложность алгоритма окажется равной $O(\log^3 n)$. Более эффективные реализации теста Люка-Лемера имеют сложность $O(\log^2 n \log\log n)$. Этот тест исключительно быстр на бинарном компьютере еще и потому, что он не требует операций деления. Как видим, несмотря на то, что доказательство теоремы Люка-Лемера потрясающе простое, их тест во много миллиардов раз быстрее любых тестов на простоту общего назначения, таких как AKS. К сожалению, тесты на простоту такой эффективности, как тест Люка-Лемера, известны лишь для чисел вида $2^p - 1$.

Именно с помощью теста Люка-Лемера найдены самые большие известные простые числа.

На сегодняшний день известно 42 Мерсеновских экспоненты. Самая большая из них была найдена в 2005 году. Она соответствует простому числу $2^{25964951} - 1$. Восемь самых больших известных сегодня Мерсеновских

экспонент найдено в рамках одного и того же проекта GIMPS [2]. Смысл проекта GIMPS заключается в том, чтобы использовать для поиска потенциал тысяч небольших персональных компьютеров. Любой желающий может принять участие в проекте GIMPS, запустив их программу на своем компьютере.

4. Поиск примитивных триномов

Мы уже знаем, что в $\mathbb{Z}_2$ любой неприводимый полином степени $r > 1$ делит $(x^{2^r-1}+1)$. Верным оказывается и несколько более общее утверждение:

Теорема. Пусть $\Phi_{d,1}, \Phi_{d,2}, \ldots$ – неприводимые полиномы степени d в $\mathbb{Z}_2[x]$. Тогда для $n \in \mathbb{N}$:

$$\prod_{d|n} \prod_{j} \Phi_{d,j}(x) = x^{(2^n)} + x \tag{9}$$

Другими словами, произведение всех неприводимых в $\mathbb{Z}_2[x]$ полиномов, степень которых делит n, равно $x^{(2^n)} + x$ (см. [9]).

Пусть r – Мерсеновская экспонента, тогда, как мы уже знаем, все неприводимые полиномы являются примитивными.

Итак, предположим, нам нужно проверить на неприводимость некоторый трином $P(x) = x^r + x^s + 1$. Проверка полинома на неприводимость напоминает проверку числа на простоту. Можно выиграть довольно много времени, если для начала проверить, делится ли $P(x)$ на какой-нибудь неприводимый полином маленькой степени n. Этот процесс называется отсеиванием. Достаточно посчитать $G = \text{НОД}(x^r + x^s + 1, x^k + 1)$, где $k = 2^n - 1$. Часто эффективнее вычислять $G = \text{НОД}(P(x), (x^{(2^n)} \bmod P(x)) + x)$. Отсеивание производится для $n = 2, 3, 4, \ldots$ до тех пор, пока не будет выполнено одно из следующих условий:

- Найден нетривиальный (не равный единице) НОД, что означает, что $P(x)$ не является неприводимым.
- $nT_s(n) \geqslant T_f(r)$, где $T_s(n)$ – оценка времени вычисления очередного НОД, $T_f(r)$ – оценка полного времени теста

На практике (для числа r=3021377) [13] в большинстве случаев отсеивание происходит для $n \leqslant 24$, занимает 8% полного времени теста и отсеивает где-то 93% триномов.

Итак, пусть наш полином не был выброшен на этапе отсеивания, т.е. для него требуется настоящая проверка на неприводимость.

Следующие утверждения являются простыми следствиями теоремы, приведенной выше.

Утверждение. Пусть $P(x) \in \mathbb{Z}_2[x]$, $\deg P = r > 1$. Тогда для того, чтобы $P(x)$ был неприводим, необходимо и достаточно, чтобы

- $x^{(2^r)} \equiv x \pmod{P(x)}$
- $\forall d \in \{1, 2, \ldots, r-1\}, d \mid r : \text{НОД}(x^{2^d} + x, P(x)) = 1$.

Следствие. Пусть $r > s > 0$, где r – простое. В этом случае $P(x) = x^r + x^s + 1 \in \mathbb{Z}_2[x]$ неприводим тогда и только тогда, когда $x^{(2^r)} \equiv x \pmod{P(x)}$.

Очевидно, это приводит к реализации проверки на неприводимость, содержащей блок, который состоит из возведения многочлена $A(x)$ в квадрат и редукции его по модулю $P(x)$. Блок выполняется r раз.

В реализации этого алгоритма блок выполняется за $O(r)$ бит-операций и требует $O(r)$ бит в памяти.

Заметим, что если $A(x) = a_0 + a_1 x + \ldots + a_d x^d$, то при возведении в квадрат в $\mathbb{Z}_2$, удвоенные произведения обращаются в нуль: $A(x)^2 = a_0 + a_1 x^2 + \ldots + a_d x^{2d}$.

Brent, Larvala, Zimmerman [13] усовершенствовали этот алгоритм так, что он стал работать примерно в два раза быстрее. Дело в том, что в простой реализации проверки на неприводимость слишком много бит-операций производится с битами, про которые заранее известно, что они должны быть равны нулю. Основная идея BLZ заключается в том, чтобы усовершенствовать метод вычисления $(A(x)^2 \bmod P(x))$. Пусть, например, $r = 7, s = 3$. Мы храним $A(x) = a_0 + \ldots + a_6 x^6$ в виде семи бит $(a_0, a_1, a_2, a_3, a_4, a_5, a_6)$. Тогда операцию возведения в квадрат и редукции по модулю $x^7 + x^3 + 1$ можно записать в виде $(a_0, a_1, a_2, a_3, a_4, a_5, a_6) \leftarrow (a_0, a_4 + a_6, a_1, a_5, a_2 + a_4 + a_6, a_6, a_3 + a_5)$. Алгоритм BLZ в этом случае состоит из двух шагов:

а) $(a_0, a_1, a_2, a_3, a_4, a_5, a_6) \leftarrow (a_0, a_1, a_2 + a_4 + a_6, a_3 + a_5, a_4 + a_6, a_5, a_6)$.

б) $(a_0, a_2, a_4, a_6, a_1, a_3, a_5) \leftarrow (a_0, a_1, a_2, a_3, a_4, a_5, a_6)$.

Этот алгоритм легко обобщается для любых r и s (см. [13]).

В настоящее время для всех Мерсеновских экспонент с $r \leqslant 6972593$ поиск примитивных триномов завершен (см. табл.1).

Таблица 1. Примитивные триномы степени, равной Мерсеновской экспоненте.

##	r	s	##	r	s
1	2	1	22	9941	-
2	3	1	23	11213	-
3	5	2	24	19937	881, 7083, 9842
4	7	1, 3	25	21701	-
5	13	-	26	23209	1530, 6619, 9739
6	17	3, 5, 6	27	44497	8575, 21034
7	19	-	28	86243	-
8	31	3, 6, 7, 13	29	110503	25230, 53719
9	61	-	30	132049	7000, 33912, 41469, 52549, 54454
10	89	38	31	216091	-
11	107	-	32	756839	215747, 267428, 279695
12	127	1, 7, 15, 30, 63	33	859433	170340, 288477
13	521	32, 48, 158, 168	34	1257787	-
14	607	105, 147, 273	35	1398269	-
15	1279	216, 418	36	2976221	-
16	2203	-	37	3021377	361604, 1010202
17	2281	715, 915, 1029	38	6972593	3037958
18	3217	67, 576	39	13466917	?
19	4253	-	40	20996011	?
20	4423	271, 369, 370, 649, 1393, 1419, 2098	41	24036583	?
21	9689	84, 471, 1836, 2444, 4187	42	25964951	?

5. Заключение

Итак, поиск констант эффективных и высококачественных генераторов псевдослучайных чисел на сдвиговых регистрах сводится к проверке чисел на простоту и полиномов на неприводимость. Существующие способы реализации этой программы замечательно работают и намного более предпочтительны из-за уникальности чисел вида $2^r - 1$. Алгоритм AKS же является универсальным для всех чисел (а также детерминированным, первым, работающим за полиномиальное время и имеющим простое и красивое доказательство). Он может оказаться применимым в этой области (и не только в ней) в том случае, если он будет усовершенствован и будет доказана правильность его более эффективного варианта. Очень перспективной выглядит гипотеза AKS, однако выяснение ее истинности оказалось не столь простым.

Автор благодарен Л. Н. Щуру за привлечение внимания к обсуждаемой проблематике и полезные обсуждения. Данная работа частично поддержана грантами РФФИ 02-02-17377, 01-07-90119, 02-07-90421.

Список литературы

[1] Agrawal M., Kayal N., Saxena N., *Primes in P*, 2002, http://www.cse.iitk.ac.in/news/primality.pdf

[2] GIMPS, *The Great Internet Mersenne Prime Search*, http://www.mersenne.org

[3] Rivest R. L., Shamir A., Adleman L., *A method for obtaining digital signatures and public key cryptosystems*, Commun. ACM. V.21, No.2, 1978, P.120-126

[4] Ященко В. В., *Введение в криптографию* (Москва, МЦНМО, 1998)

[5] Винберг Э. Б., *Курс алгебры* (Москва, Факториал, 1999, 2001)

[6] Карацуба А. А., *Основы аналитической теории чисел* (М.: Наука, 1972)

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

[8] Кнут Д., *Искусство программирования, том 2 / Получисленные алгоритмы, 3-е изд* (Москва, Вильямс, 2000)

[9] Лидл Р., Ниддеррайтер Г., *Конечные поля* (Москва, Мир, 1988)

[10] Tezuka S., *Uniform Random Numbers: Theory and Practice* (Kluwer, Boston et al., 1995)

[11] Kayal N., Saxena N., *Towards a deterministic polynomial-time primality test. Technical report*, IIT Kanpur, 2002, http://www.cse.iitk.in/research/btp2002/primality.html

[12] Golomb S. W., *Shift Register Sequences* (Holden-Day, San Francisco, 1967)

[13] Brent R. P., Larvala S., Zimmerman P., *A fast algorithm for testing reducibility of trinomials mod 2 and some new primitive trinomials of degree 3021377*, http://web.comlab.ox.ac.uk/oucl/work/richard.brent/pub/pub199.html