Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования

"Национальный исследовательский университет

"Высшая школа экономики"

Московский институт электроники и математики им. А.Н.Тихонова

Департамент прикладной математики

ОТЧЕТ по работе по дисциплине

«Теоретико-числовые методы в криптографии»

Проверка чисел на простоту:

полиномиальный алгоритм AKS

Выполнили:

Щелканов Густав СКБ181

Ротай Николай СКБ181

Руководитель:

Нестеренко А. Ю.

Москва, 2022 г.

Введение

В 2002 г. индийским математиком Агравалом (Agrawal) и двумя его студентами Кайялом (Kayal) и Саксеной (Saxena) был предложен произведший ошеломляющее впечатление алгоритм проверки простоты чисел [1]. Этот алгоритм, формально опубликованный в 2004 г. [2], стали называть алгоритмом AKS. Авторы оригинально модифицировали существующие тесты и создали первый алгоритм, который удовлетворяет одновременно трём требованиям:

- детерминированный — всегда находит корректный ответ;
- безусловный — не опирается на какие-либо недоказанные гипотезы, такие, как обобщенная гипотеза Римана;
- полиномиальный — асимптотическое время выполнения полиномиально.

Большая часть идей, используемых в алгоритме AKS, была в широком обращении, но до указанных авторов никто на их основе не преуспел в построении алгоритма с такими свойствами. Более ранние тесты на простоту могли удовлетворять двум из этих свойств, но не всем трём.

Из предшествующих детерминированных алгоритмов быстрейшие (напри-мер, Адлемана-Померанса-Румели, см. [3]) имеют время выполнения порядка $O\ (\log n)^{c \log \log n\ 2)}$. Это экспоненциальное время, однако степень $c \log \log n$ стремится к бесконечности с ростом $\log n$ довольно медленно. Так, при $d < 10^{38}$ значение $\log \log d$ не превышает 7.

Теоретическая часть

1. Теоремы Эйлера и Ферма:
   - Если $m > 1$ и НОД$(a,m) = 1$, то $a^{\phi(m)} \equiv 1 \pmod m$, где $\phi(r)$ — функция Эйлера (теорема Эйлера).
   - Если p — простое число, то $a^p \equiv a \pmod p$ (малая теорема Ферма).

Из последней теоремы при простом p получаем $(x + y)^p \equiv x + y \pmod p$ и $x^p + y^p \equiv x + y \pmod p$, откуда следует примечательное тождество

$$(x + y)^p \equiv x^p + y^p \pmod p,$$

иногда называемое детской биномиальной теоремой.

2. Критерий простоты:

Если a и n взаимно просты, то тождество

$$(x + a)^n \equiv (x^n + a) \pmod n$$

выполняется тогда и только тогда, когда n — простое число.

Сравнение понимается так: все коэффициенты многочлена $(x+a)^n-(x^n+a)$ кратны n.

Если n просто, то для любого целого r > 0 *и любого целого a,* $0 < a < n$*, выполняется сравнение*

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

К сожалению, обратное утверждение неверно. Однако желаемый результат был достигнут Агравалом, Кайялом и Саксеной при использовании некоторого определенного r и некоторого множества значений для a. Достаточные условия простоты числа n были сведены к выполнению требований:

(i) значение r должно быть таким, что *order* $r(n) > L$, где L зависит от n;

(ii) соотношение (3) выполняется для любого целого a из интервала $[1,A]$, где A зависит от n и r.
(Иногда вместо $[1,A]$ указывают интервал $[0,A]$. Это то же самое, так как при $a = 0$ условие (3) выполнено.)

В разных изложениях алгоритма, в том числе и разных авторских, значения L и A несколько различаются. Мы будем проводить все доказательства и расчеты для принятых в [5] значений

$$L = 4\log^2 n, \quad A = 2\sqrt{r}\log n.$$

ТЕОРЕМА (AKS). *Если существует r, удовлетворяющее (i), при котором для всех a, удовлетворяющих (ii), выполняется условие* (3)*, то* *n либо простое, либо степень простого.*

Алгоритм:

Алгоритм AKS. На входе — нечётное число n.	
Шаги алгоритма	Оценка числа битовых операций
Определить, будет ли n точной степенью целого числа, $n = m^k$ для какого-то $k > 1$. Если да, возврат СОСТАВНОЕ.	$\tilde{O}(\log^3 n)$ (и даже $\tilde{O}(\log n)$, если применять алгоритм [6]).
Если n — не точная степень, то	
Найти такое целое r, что $order_r(n) > L$. Для этого надо вычислять $n^j \pmod q$ при $j = 1, 2, \ldots, \lfloor L \rfloor$ при каждом целом $q > L$, пока не найдётся первое q, для которого ни один из этих $n^j \pmod q$ не равен 1. После этого принять $r = q$.	$O(r \cdot \log^2 n \cdot \log^2 r)$, где r оценивает количество обращений к q, а $O(\log^2 n \cdot \log^2 r)$ оценивает при фиксированном q трудоёмкость вычисления $n^j \pmod q$ с учетом $q \leqslant r$ и $j \leqslant 4\log^3 n$.
Проверить основное соотношение $$(x+a)^n \equiv (x^n + a) \pmod{x^r - 1, n}$$ для каждого целого $a \in [1, 2\sqrt{r}\log n]$. Если найдётся a, для которого оно выполнится, возврат СОСТАВНОЕ.	$\tilde{O}(r^{\frac{3}{2}} \log^3 n)$. Наиболее трудоёмкая часть: для каждого a нужно $r\,\tilde{O}(\log^2 n)$ битовых операций при использовании бинарного возведения в степень и быстрого преобразования Фурье.
Если не вышли из алгоритма раньше, возврат ПРОСТОЕ.	

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

1) $c_1 \log^6 n < r < c_2 \log^6 n$
2) $(r-1)$ имеет простой делитель q, где $q \geq 4\sqrt{r}\log n$
3) $n^{(r-1)/q} \not\equiv 1 \ (mod\ r)$

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

Реализация и описание алгоритма

Алгоритм реализован на языке Python с использованием Sage.

Код программы:

```
def step1(n):
    b = 2
    while True:
        a = n ** (1 / b)
        b += 1
        if a.is_integer():
            print(f"{n} is composite | Step 1")
            return False
        if a < 2:
            return True

def step2(n):
    L = math.floor(4 * math.log2(n) ** 2)
    condition = True
    r = 1
    while condition:
        r += 1
        condition = False
        j = 0
        while j <= L and not condition:
            j += 1
            if pow(n, j, r) == 1:
                condition = True
    return r

def step3(n, r):
    if n <= r:
        print(f"{n} is prime | Step 3")
        return True
    max_a = floor(2 * math.sqrt(r) * math.log2(n))
    R.<x> = PolynomialRing(Integers(n))
    Z = R.quotient((x^r)-1)
    for a in range(1, max_a + 1):
        q = Z((x + a)^n)
        d = x^(Mod(n, r)) + a
        if (q != d):
            print(f"{n} is composite | Step 3")
            return False
    print(f"{n} is prime | Step 3")
    return True
```

```
    import time
    import matplotlib.pyplot as plt

    def aks(n):
        if n <= 1:
            print("Error: n must be > 1.")
            return -1
        first_step = step1(n)
        if first_step:
            r = step2(n)
            third_step = step3(n, r)
            if not third_step:
                return 0
        else:
            return 0
        return 1

times = []
numbers = []
def main(n_):
    start = time.time()
    aks_res = aks(n_)
    test = is_prime(n_)
    if not ((aks_res == 0 and not test) or (aks_res == 1 and test)):
        print('ERROR')
        return n_, -1
    end = time.time() - start
    print(end)
    print('===' * 20)
    return n_, end

for i in range(2, 2^14):
    number, time_ = main(i)
    numbers.append(number)
    times.append(time_)
```

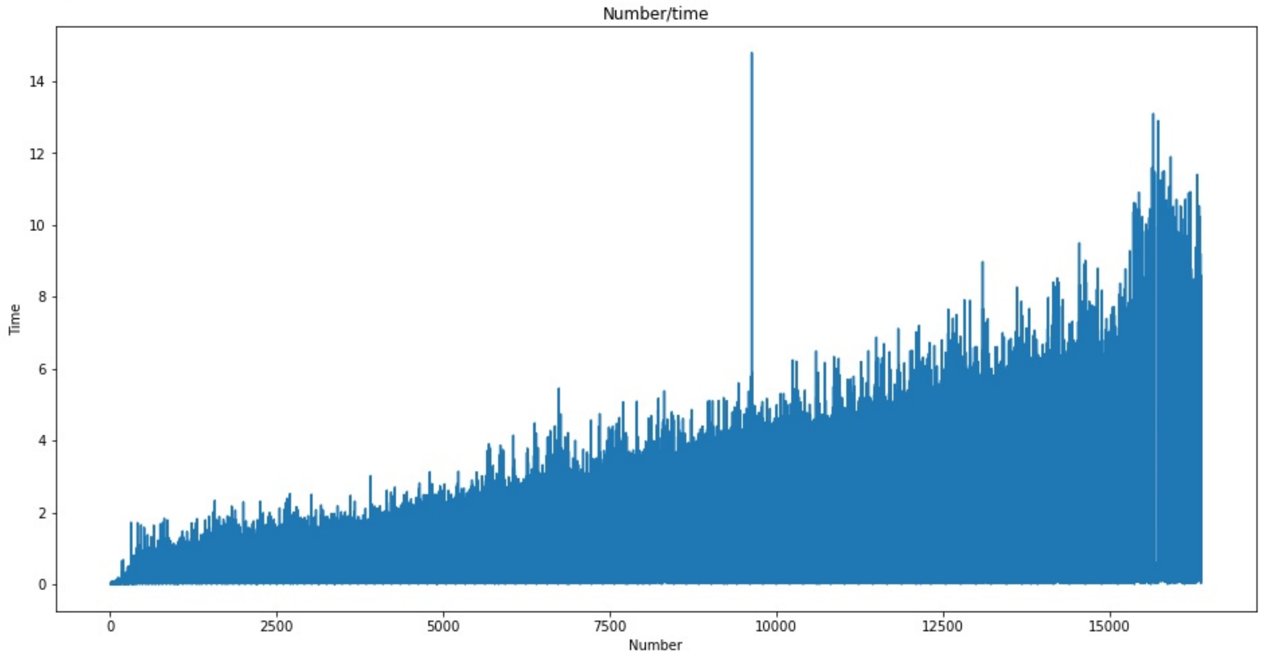
Пример работы программы:

```
============================================================
16135 is composite | Step 3
0.11027717590332031
============================================================
16136 is composite | Step 3
0.10893917083740234
============================================================
16137 is composite | Step 3
0.15419721603393555
============================================================
16138 is composite | Step 3
0.10457873344421387
============================================================
16139 is prime | Step 3
9.700096368789673
============================================================
16140 is composite | Step 3
0.10275530815124512
============================================================
16141 is prime | Step 3
10.707839727401733
```

Зависимость времени от размера входного числа:

```
plt.figure(figsize=(16,8))
plt.plot(numbers, times)
plt.title('Number/time')
plt.xlabel('Number')
plt.ylabel('Time')
```

```
Text(0, 0.5, 'Time')
```



Список литературы

1. PRIMES is in P. Manindra Agrawal, Neeraj Kayal, Nitin Saxena. https://www.cse.iitk.ac.in/users/manindra/algebra/primality_v6.pdf
2. Агафонова И. В. «Проверка чисел на простоту: полиноминальный алгоритм».