ЛАБОРАТОРНАЯ РАБОТА №5

Вероятностные алгоритмы проверки чисел на простоту

Пусть a – целое число. Числа $\pm 1, \pm a$ называются *тривиальными делителями* числа a.

Целое число $p \in Z/\{0\}$ называется *простым*, если оно не является делителем единицы и не имеет других делителей, кроме тривиальных. В противном случае число $p \in Z/\{-1, 0, 1\}$ называется *составным*.

Например, числа $\pm 2, \pm 3, \pm 5, \pm 7, \pm 11, \pm 13, \pm 17, \pm 19, \pm 23, \pm 29$ являются простыми.

Пусть $m \in N, m > 1$. Целые числа a и b называются сравнимыми по модулю m (обозначается $a \equiv b\ (mod\ m)$) если разность $a - b$ делится на m. Также эта процедура называется нахождением остатка от целочисленного деления a на b.

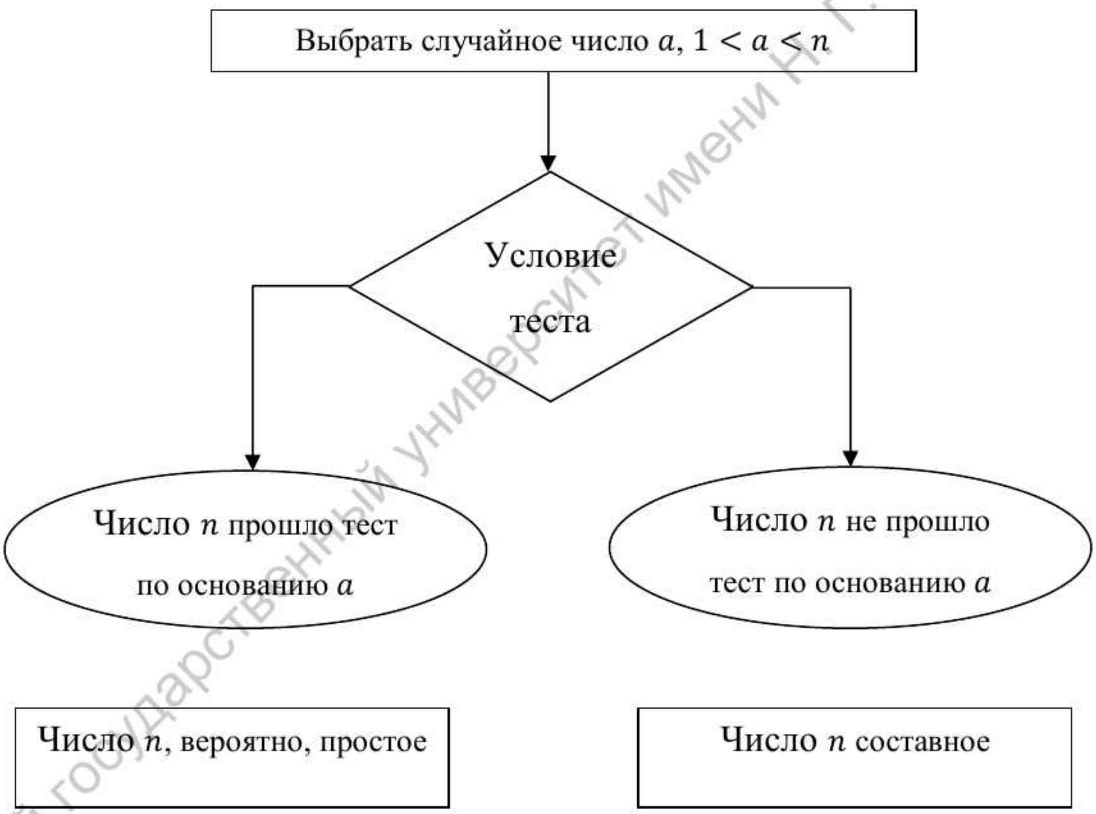
Проверка чисел на простоту является составной частью алгоритмов генерации простых чисел, применяемых в криптографии с открытым ключом. Алгоритмы проверки на простоту можно разделить на вероятностные и детерминированные.

Детерминированный алгоритм всегда действует по одной и той же схеме и гарантированно решает поставленную задачу (или не дает никакого ответа). *Вероятностный* алгоритм использует генератор случайных чисел и дает не гарантированно точный ответ. Вероятностные алгоритмы в общем случае не менее эффективны, чем детерминированные (если используемый генератор случайных чисел всегда дает набор одних и тех же чисел, зависящих от входных данных, то вероятностный алгоритм становится детерминированным).

Для проверки на простоту числа n вероятностным алгоритмом выбирают случайной число a $(1 < a < n)$ и проверяют условия алгоритма. Если число n не проходит тест по основанию a, то алгоритм выдает результат «Число n составное», и число n действительно является составным.

Если же n проходит тест по основанию a, ничего нельзя сказать о том, действительно ли число n является простым. Последовательно проведя ряд проверок таким тестом для разных a и получив для каждого из них ответ «Число n, вероятно, простое», можно утверждать, что число n является простым с вероятностью, близкой к 1. После t независимых выполнений теста вероятность того, что составное число n будет t раз объявлено простым (вероятность ошибки), не превосходит $\frac{1}{2^t}$.

Схема вероятностного алгоритма проверки числа на простоту



Тест Ферма основан на малой теореме Ферма: для простого числа p и произвольного числа a, $1 \le a \le p-1$, выполняется сравнение

$$a^{p-1} \equiv 1 \ (mod \ p).$$

Следовательно, если для нечетного n существует такое целое a, что $1 \le a < n$, НОД$(a,n) = 1$ и $a^{n-1} \neq 1 \ (mod \ n)$, то число n составное. Отсюда получаем следующий вероятностный алгоритм проверки числа на простоту.

1. Алгоритм, реализующий тест Ферма.

Вход. Нечетное целое число $n \geq 5$.

Выход. «Число n, вероятно, простое» или «Число n составное».

1. Выбрать случайное целое число a, $2 \leq a \leq n-2$.
2. Вычислить $r \leftarrow a^{n-1} \ (mod\ n)$.
3. При $r = 1$ результат: «Число n, вероятно, простое». В противном случае результат: «Число n составное».

На шаге 1 мы не рассматривали числа $a = 1$ и $a = n-1$, поскольку $1^{n-1} \equiv 1 \ (mod\ n)$ для любого целого n и $(n-1)^{n-1} \equiv (-1)^{n-1} \equiv 1 \ (mod\ n)$ для любого нечетного n.

Тест Соловэя-Штрассена. Основан на критерии Эйлера: нечетное число n является простым тогда и только тогда, когда для любого целого числа a, $1 \leq a \leq n-1$, взаимно простого с n, выполняется сравнение:

$$a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \ (mod\ n),$$

где $\left(\frac{a}{n}\right)$ – *символ Якоби*.

Пусть $m, n \in Z$, где $n = p_1 p_2 \dots p_r$ и числа $p_i \neq 2$ простые (не обязательно различные). *Символ Якоби* $\left(\frac{m}{n}\right)$ определяется равенством

$$\left(\frac{m}{n}\right) = \left(\frac{m}{p_1}\right)\left(\frac{m}{p_2}\right) \dots \left(\frac{m}{p_r}\right).$$

2. Алгоритм вычисления символа Якоби.

Вход. Нечетное целое число $n \geq 3$, целое число a, $0 \leq a < n$.

Выход. Символ Якоби $\left(\frac{a}{n}\right)$.

1. Положить $g \leftarrow 1$.

2. При $a = 0$ результат: 0.
3. При $a = 1$ результат: g.
4. Представить a в виде $a = 2^k a_1$, где число a_1 нечетное.
5. При четном k положить $s \leftarrow 1$, при нечетном k положить $s \leftarrow 1$, если $n \equiv \pm 1 \ (mod\ 8)$; положить $s \leftarrow -1$, если $n \equiv \pm 3 \ (mod\ 8)$.
6. При $a_1 = 1$ результат: $g \cdot s$.
7. Если $n \equiv 3 \ (mod\ 4)$ и $a_1 \equiv 3 \ (mod\ 4)$, то $s \leftarrow -s$.
8. Положить $a \leftarrow n \ (mod\ a_1)$, $n \leftarrow a_1$, $g \leftarrow g \cdot s$ и вернуться на шаг 2.

3. Алгоритм, реализующий тест Соловэя-Штрассена.

Вход. Нечетное целое число $n \geq 5$.

Выход. «Число n, вероятно, простое» или «Число n составное».

1. Выбрать случайное целое число a, $2 \leq a < n - 2$.
2. Вычислить $r \leftarrow a^{\frac{n-1}{2}} \ (mod\ n)$.
3. При $r \neq 1$ и $r \neq n - 1$ результат: «Число n составное».
4. Вычислить *символ Якоби* $s \leftarrow \left(\frac{a}{n}\right)$.
5. При $r \equiv s \ (mod\ n)$ результат: «Число n составное». В противном случае результат: «Число n, вероятно, простое».

На сегодняшний день для проверки чисел на простоту чаще всего используется тест Миллера-Рабина, основанный на следующем наблюдении. Пусть число n нечетное и $n - 1 = 2^s r$, где r – нечетное. Если n простое, то для любого $a \geq 2$, взаимно простого с n, выполняется условие $a^{p-1} \equiv 1 \ (mod\ p)$.

4. Алгоритм, реализующий тест Миллера-Рабина.

Вход. Нечетное целое число $n \geq 5$.

Выход. «Число n, вероятно, простое» или «Число n составное».

1. Представить $n - 1$ в виде $n - 1 = 2^s r$, где число r нечетное.
2. Выбрать случайное целое число a, $2 \leq a < n - 2$.

3. Вычислить $y \leftarrow a^r (mod\ n)$.
4. При $y \neq 1$ и $y \neq n - 1$ выполнить следующие действия.
   4.1. Положить $j \leftarrow 1$.
   4.2. Если $j \leq s - 1$ и $y \neq n - 1$, то
      4.2.1. Положить $y \leftarrow y^2\ (mod\ n)$.
      4.2.2. При $y = 1$ результат: «Число n составное».
      4.2.3. Положить $j \leftarrow j + 1$.
   4.3. При $y \neq n - 1$ результат: «Число n составное».
5. Результат: «Число n, вероятно, простое».

Задания к лабораторной работе

Реализовать все рассмотренные алгоритмы программно.