МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Проверка чисел на простоту

ОТЧЁТ

ПО ДИСЦИПЛИНЕ

«ТЕОРЕТИКО-ЧИСЛОВЫЕ МЕТОДЫ В КРИПТОГРАФИИ»

студента 5 курса 531 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Алексеева Александра Александровича

Преподаватель

профессор, д.ф.-м.н. ___________ В. А. Молчанов

подпись, дата

Саратов 2023

СОДЕРЖАНИЕ

1 Цель работы и порядок её выполнения

Цель работы – изучение методов проверки простоты чисел и их программная реализация.

Порядок выполненных работы:

1. Рассмотреть тест Ферма проверки чисел на простоту и привести его программную реализацию.
2. Рассмотреть тест Соловея-Штрассена проверки чисел на простоту и привести его программную реализацию.
3. Рассмотреть тест Миллера-Рабина и привести его программную реализацию.

2 Теория

Вероятностный алгоритм проверки числа n на простоту использует необходимое условие простоты $P(a)$:

1) выбирается случайным образом $1 < a < n$ и проверяется выполнимость теста $P(a)$ – некоторого условия алгоритма;

2) если тест не проходит, т.е. $P(a)$ не выполняется, то вывод «число n составное»,

3) если тест проходит, т.е. $P(a)$ выполняется, то вывод «число n, вероятно, простое».

Если событие A – «число n простое» имеет вероятность $P(A) > \frac{1}{2}$, то вероятность ошибки – получить для составного числа n вывод «число n возможно простое» $P(\bar{\bar{A}}) < \frac{1}{2}$ и при t повторах теста вероятность ошибки $P(\bar{\bar{A}}^t) < \frac{1}{2^t} \approx 0$.

2.1 Тест Ферма проверки чисел на простоту

Малая теорема Ферма. Если р – простое число, то для любого $a \in \boldsymbol{Z}_p^*$ выполняется свойство $F_p(a) = (a^{p-1} \equiv 1 \pmod p)$.

$$p \text{ – простое число} \Rightarrow F_p^+ = \boldsymbol{Z}_n^*,$$

где $F_p^+ = \{a \in \boldsymbol{Z}_p^* : F_p(a)\}$ – множество истинности предиката $F_p(a)$.

Определение. Нечётное число n называется *числом Кармайкла*, если $F_n^+ = Z_n^*$ (и, значит, вероятность успеха теста Ферма будет $P_0 = 1 - \frac{\varphi(n)}{n-1}$.

Лемма. Для любого числа Кармайкла n справедливы утверждения:

1) $n = p_1 p_2 \dots p_k$ для $k \geq 3$ простых различных чисел $p_1, p_2, \dots, p_k$;

2) $(\forall\ p$ – простое$)$ $p \mid n \Rightarrow p - 1 \mid n - 1$.

Плотность распределения чисел Кармайкла:

$1 - 10^5$ – 16 чисел: 561, 1105, 1729, …;

$1 - 2{,}5 \bullet 10^{10}$ – 2163 чисел.

Алгоритм – тест простоты на основе малой теоремы Ферма:

Вход: нечётное число $n > 5$.

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

Шаг 1. Выбрать случайно $a \in \{1, 2, \dots, n-1\}$ и вычислить d = НОД(a, n). Если $d > 1$, то ответ «Число n составное».

Шаг 2. Если $d = 1$, То проверить условие

$$F_n(a) = (a^{n-1} \equiv 1 \pmod n).$$

Если оно не выполнено, то ответ «Число n составное». В противном случае ответ «Число n, вероятно, простое».

Псевдокод теста Ферма проверки чисел на простоту

Вход: нечётное число $n > 5$, количество раундов $k > 0$.

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

Шаг 1. Цикл по i от 0 до k:

Шаг 1.1.Сгенерировать случайное $a \in \{1, 2, \dots, n-1\}$.

Шаг 1.2. Вычислить d = НОД(a, n). Если $d > 1$, то ответ «Число n составное».

Шаг 1.3. Проверить условие $a^{n-1} \equiv 1 \pmod n$. Если оно не выполнено, то ответ «Число n составное».

Шаг 2. Вернуть ответ «Число n, вероятно, простое».

2.2 Тест Соловея-Штрассена проверки чисел на простоту

Критерий Эйлера. Нечётное число n является простым тогда и только тогда, когда для любого $a \in \boldsymbol{Z}_n^*$ выполняется свойство

$$E_n(a) = \left(a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \pmod n\right).$$

$$n \text{ – простое число} \Leftrightarrow E_n^+ = \boldsymbol{Z}_n^*, \text{ где}$$

$$E_n^+ = \{a \in \boldsymbol{Z}_n^* | E_n(a)\}.$$

Алгоритм – тест простоты Соловея-Штрассена на основе критерия Эйлера:

Вход: нечётное число $n > 5$.

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

Шаг 1. Выбрать случайно $a \in \{1, 2, \ldots, n-1\}$ и вычислить d = НОД(a, n). Если $d > 1$, то ответ «Число n составное».

Шаг 2. Если $d = 1$, то проверить условие $E_n(a)$. Если оно не выполнено, то ответ «Число n составное». В противном случае ответ «Число n, вероятно, простое».

Псевдокод теста Соловея-Штрассена проверки чисел на простоту

Вход: нечётное число $n > 5$, количество раундов $k > 0$.

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

Шаг 1. Цикл по i от 0 до k:

Шаг 1.1.Сгенерировать случайное $a \in \{1, 2, \ldots, n-1\}$.

Шаг 1.2. Вычислить d = НОД(a, n). Если $d > 1$, то ответ «Число n составное».

Шаг 1.3. Проверить условие $a^{\frac{n-1}{2}} \equiv (\frac{a}{n}) (\mathrm{mod}\ n)$. Если оно не выполнено, то ответ «Число n составное».

Шаг 2. Вернуть ответ «Число n, вероятно, простое».

2.3 Тест Миллера-Рабина проверки чисел на простоту

Теорема (Критерий Миллера). Пусть n – нечётное число и $n - 1 = 2^s t$ для нечётного t. Тогда n является простым в том и только том случае, если для любого $a \in Z_n^*$ выполняется свойство

$$M_n(a) = (a^t \equiv 1 (\mathrm{mod}\ n) \lor (\exists\ 0 \le k < s)(a^{2^k t} \equiv -1 (\mathrm{mod}\ n))).$$

$$n \text{ – простое число} \Leftrightarrow M_n^+ = Z_n^*, \text{ где}$$

$$M_n^+ = \{a \in Z_n^* | M_n(a)\}.$$

Алгоритм – тест простоты Миллера-Рабина на основе критерия Миллера:

Вход: нечётное число $n > 5$.

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

Шаг 1. Выбрать случайно $a \in \{1, 2, \ldots, n-1\}$ и вычислить d = НОД(a, n). Если $d > 1$, то ответ «Число n составное».

Шаг 2. Если $d = 1$, то вычислить $r_k = a^{2^k t}$ для значений $k \in \{0, 1, 2, \ldots, s-1\}$. Если $r_0 \equiv 1 \pmod n$ или $r_k \equiv -1 \pmod n$ для некоторого $0 \le k < s$, то ответ «Число n, вероятно, простое». В противном случае ответ «Число n составное».

Псевдокод теста Миллера-Рабина проверки чисел на простоту

Вход: нечётное число $n > 5$, количество раундов $k > 0$.

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

Шаг 1. Вычислить $n - 1 = 2^s t$.

Шаг 2. Цикл по i от 0 до k:

Шаг 1.1.Сгенерировать случайное $a \in \{1, 2, \ldots, n-1\}$.

Шаг 1.2. Вычислить d = НОД(a, n). Если $d > 1$, то ответ «Число n составное».

Шаг 1.3. Вычислить $r = a^t \pmod n$. Если $r \pmod n = 1$ или $r \pmod n = -1$, то перейти к следующей итерации.

Шаг 1.4. Цикл по j от 1 до s:

Шаг 1.4.1. Вычислить $r = r * r (\mathrm{mod}\, n)$.

Шаг 1.4.2. Если $r = n - 1$, то перейти к следующей итерации цикла по i.

Шаг 1.4.3. Выдать ответ «Число n составное».

Шаг 2. Вернуть ответ «Число n, вероятно, простое».

3 Результаты работы

3.1 Оценки сложности рассмотренных алгоритмов

Тест Ферма проверки числа на простоту – $O(\log^3 n)$;

Тест Соловея-Штрассена проверки числа на простоту – $O(\log^3 n)$;

Тест Миллера-Рабина проверки числа на простоту – $O(\log^3 n)$.

3.2 Результаты тестирования программ

```
        Проверка числа n на простоту с помощью теста Ферма, Соловея-Штрассена и Миллера-Рабина
Введите n > 5: 561
Введите число проверок k > 0: 3

Тест Ферма: число 561 составное
Тест Соловея-Штрассена: число 561 составное
Тест Миллера-Рабина: число 561 составное

        Проверка числа n на простоту с помощью теста Ферма, Соловея-Штрассена и Миллера-Рабина
Введите n > 5: 561
Введите число проверок k > 0: 2

Тест Ферма: число 561 составное
Тест Соловея-Штрассена: число 561, вероятно, простое
Тест Миллера-Рабина: число 561 составное

        Проверка числа n на простоту с помощью теста Ферма, Соловея-Штрассена и Миллера-Рабина
Введите n > 5: 561
Введите число проверок k > 0: 1

Тест Ферма: число 561, вероятно, простое
Тест Соловея-Штрассена: число 561 составное
Тест Миллера-Рабина: число 561 составное
```

```
        Проверка числа n на простоту с помощью теста Ферма, Соловея-Штрассена и Миллера-Рабина
Введите n > 5: 1729
Введите число проверок k > 0: 10

Тест Ферма: число 1729 составное
Тест Соловея-Штрассена: число 1729 составное
Тест Миллера-Рабина: число 1729 составное

        Проверка числа n на простоту с помощью теста Ферма, Соловея-Штрассена и Миллера-Рабина
Введите n > 5: 1729
Введите число проверок k > 0: 5

Тест Ферма: число 1729 составное
Тест Соловея-Штрассена: число 1729 составное
Тест Миллера-Рабина: число 1729 составное

        Проверка числа n на простоту с помощью теста Ферма, Соловея-Штрассена и Миллера-Рабина
Введите n > 5: 1729
Введите число проверок k > 0: 1

Тест Ферма: число 1729, вероятно, простое
Тест Соловея-Штрассена: число 1729 составное
Тест Миллера-Рабина: число 1729 составное
```

```
        Проверка числа n на простоту с помощью теста Ферма, Соловея-Штрассена и Миллера-Рабина
Введите n > 5: 999983
Введите число проверок k > 0: 10

Тест Ферма: число 999983, вероятно, простое
Тест Соловея-Штрассена: число 999983, вероятно, простое
Тест Миллера-Рабина: число 999983, вероятно, простое

        Проверка числа n на простоту с помощью теста Ферма, Соловея-Штрассена и Миллера-Рабина
Введите n > 5: 999983
Введите число проверок k > 0: 5

Тест Ферма: число 999983, вероятно, простое
Тест Соловея-Штрассена: число 999983, вероятно, простое
Тест Миллера-Рабина: число 999983, вероятно, простое

        Проверка числа n на простоту с помощью теста Ферма, Соловея-Штрассена и Миллера-Рабина
Введите n > 5: 999983
Введите число проверок k > 0: 1

Тест Ферма: число 999983, вероятно, простое
Тест Соловея-Штрассена: число 999983, вероятно, простое
Тест Миллера-Рабина: число 999983, вероятно, простое
```

3.3 Код программы

```
#include "iostream"
#include "vector"

using namespace std;

vector <long long> deg2(long long el, long long n) {//Раскладываем число на
степени двойки
	vector <long long> res;
	while (n != 0) {
		if (n / el == 1) {
			res.push_back(el);
			n -= el;
			el = 1;
		}
		else
			el *= 2;
	}
	return res;
}


long long multMod(long long n, long long mod, vector <pair <long long, long
long>> lst) {//Умножаем число по модулю
	if (lst.size() == 1) {
		long long res = 1;
		for (short i = 0; i < lst[0].second; i++)
			res = res * lst[0].first % mod;
		return res;
	}
	else if (lst[0].second == 1) {
		long long el = lst[0].first;
		lst.erase(lst.begin());
		return (el * multMod(n, mod, lst)) % mod;
	}
	else {
		for (short i = 0; i < lst.size(); i++)
			if (lst[i].second > 1) {
```

```
                                lst[i].first = (lst[i].first * lst[i].first) %
mod;
                                lst[i].second /= 2;
                        }
                return multMod(n, mod, lst);
        }
}


long long powClosed(long long x, long long y, long long mod) {//Возводим число в
степени по модулю
        if (y == 0)
                return 1;

        vector <long long> lst = deg2(1, y);
        vector <pair <long long, long long>> xDegs;
        for (short i = 0; i < lst.size(); i++)
                xDegs.push_back(make_pair(x, lst[i]));

        long long res = multMod(x, mod, xDegs);
        return res;
}


int binaryEuclid(int a, int b) {
        if (a < 0 || b < 0)
                throw string{ "Выполнение невозможно: a < 0 или b < 0" };

        if (a == 0)
                return b;
        else if (b == 0 || a == b)
                return a;
        else if (a == 1 || b == 1)
                return 1;
        else if ((a & 1) == 0 && (b & 1) == 0)
                return binaryEuclid(a >> 1, b >> 1) << 1;
        else if ((a & 1) == 0 && (b & 1) == 1)
                return binaryEuclid(a >> 1, b);
        else if ((a & 1) == 1 && (b & 1) == 0)
                return binaryEuclid(a, b >> 2);
        else {
                if (b > a)
                        return binaryEuclid((b - a) >> 1, a);
                else
                        return binaryEuclid((a - b) >> 1, b);
        }
}


int symbolLegendre(int a, int p) {
        if (a == 0)
                return 0;
        int res = powClosed(a, (p - 1) / 2, p);
        return res == 1 ? 1 : -1;
}


bool ferma(int n, short k) {
        for (short i = 0; i < k; i++) {
                int a = rand() % (n - 2) + 2;

                int d = binaryEuclid(a, n);
                if (d > 1)
```

```
                        return false;

                if (powClosed(a, n - 1, n) != 1)
                        return false;
        }
        return true;
}


bool soloveyStrassen(int n, short k) {
        int halfed = n >> 1;
        for (short i = 0; i < k; i++) {
                int a = rand() % (n - 2) + 2;

                int d = binaryEuclid(a, n);
                if (d > 1)
                        return false;

                int pow = powClosed(a, halfed, n);
                int symLegendre = symbolLegendre(a, n);
                if ((pow == 1 && symLegendre != 1) || (pow == n - 1 &&
symLegendre != -1))
                        return false;
        }
        return true;
}


bool millerRabin(int n, short k) {
        int t = n - 1;
        int s = 0;
        while (t % 2 == 0) {
                s++;
                t = t / 2;
        }

        for (short i = 0; i < k; i++) {
                int a = rand() % (n - 2) + 2;

                int d = binaryEuclid(a, n);
                if (d > 1)
                        return false;

                long long r = powClosed(a, t, n);
                if (r == 1 || r == n - 1)
                        continue;

                bool isSimple = false;
                for (short j = 1; j < s; j++) {
                        r = (r * r) % n;
                        if (r == n - 1) {
                                isSimple = true;
                                break;
                        }
                }
                if (!isSimple)
                        return false;
        }
        return true;
}


int main() {
```

```
	setlocale(LC_ALL, "ru");
	for (;;) {
		srand(time(0));
		cout << "\tПроверка числа n на простоту с помощью теста Ферма, 
Соловея-Штрассена и Миллера-Рабина \nВведите n > 5: ";
		int n;
		cin >> n;
		cout << "Введите число проверок k > 0: ";
		short k;
		cin >> k;

		if (n < 6 || k < 0) {
			cout << "Incorrect. Try again \n\n";
			continue;
		}

		if (ferma(n, k))
			cout << "\nТест Ферма: число " << n << ", вероятно, 
простое";
		else
			cout << "\nТест Ферма: число " << n << " составное";

		if (soloveyStrassen(n, k))
			cout << "\nТест Соловея-Штрассена: число " << n << ", 
вероятно, простое";
		else
			cout << "\nТест Соловея-Штрассена: число " << n << " 
составное";

		if (millerRabin(n, k))
			cout << "\nТест Миллера-Рабина: число " << n << ", 
вероятно, простое";
		else
			cout << "\nТест Миллера-Рабина: число " << n << " 
составное";

		cout << "\n\n";
	}
	return 0;
}
```

ЗАКЛЮЧЕНИЕ

В ходе данной работы были разобраны вероятностные алгоритмы проверки чисел на простоту: тест Ферма, тест Соловея-Штрассена на основе критерия Эйлера и тест Миллера-Рабина на основе критерия Миллера.