

Отчет по лабораторной работе № 5. Вероятностные алгоритмы проверки чисел на простоту

**дисциплина: Математические основы защиты информации и
информационной безопасности**

Наливайко Сергей Максимович

Содержание

1	Цель работы	3
2	Задание	4
3	Выполнение лабораторной работы	5
3.1	Реализация вероятностных алгоритмов	5
4	Выводы	11

1 Цель работы

Научиться реализовывать вероятностные алгоритмы проверки чисел на простоту.

2 Задание

- Реализовать алгоритм, реализующий тест Ферма.
- Реализовать алгоритм вычисления числа Якоби.
- Реализовать алгоритм, реализующий тест Соловея-Штрассена.
- Реализовать алгоритм, реализующий тест Миллера-Рабина.

3 Выполнение лабораторной работы

3.1 Реализация вероятностных алгоритмов

Реализуем вероятностные алгоритмы проверки чисел на простоту на языке программирования C++.

Код алгоритма, реализующего тест Ферма:

```
bool ferma_test(uint_fast64_t n) {
    if (n < 5)
        throw std::invalid_argument("n >= 5!");
    if (n % 2 == 0)
        return false;
    std::srand(std::time(nullptr));
    for(int i = 0; i < TEST_ITERATIONS; ++i) {
        uint_fast64_t a = (std::rand() % (n - 3) + 2);
        uint_fast64_t r = 1;
        for(int j = 1; j < n; ++j) r *= a;
        r %= n;
        if(r == 1)
            return true;
    }
    return false;
}
```

Код алгоритма нахождения числа Якоби:

```

int jacobi(int a, int b) {

    int g;
    if (a >= b) a %= b;
    if (a == 0) return 0;
    if (a == 1) return 1;
    if (a < 0) {
        if ((b-1)/2 % 2 == 0)
            return jacobi(-a,b);
        else
            return -jacobi(-a,b);
    }
    if (a % 2 == 0) {
        if (((b*b - 1)/8) % 2 == 0)
            return jacobi(a/2,b);
        else
            return -jacobi(a/2,b);
    }
    g = euclidean_algorithm(a,b);
    if (g == a)
        return 0;
    else if (g != 1)
        return jacobi(g,b)*jacobi(a/g,b);

    else if (((a-1)*(b-1)/4) % 2 == 0)
        return jacobi(b,a);
    else
        return -jacobi(b,a);
}

```

Код алгоритма, реализующего тест Соловея-Штрассена:

```
bool s_sh_test(ulong n) {
    if (n < 5)
        throw std::invalid_argument("n >= 5!");
    if (n % 2 == 0)
        return false;
    std::srand(std::time(nullptr));
    for(int i = 0; i < TEST_ITERATIONS; ++i) {
        uint_fast64_t a = (std::rand() % (n - 3) + 2);
        if(euclidean_algorithm(a, n) > 1)
            return false;
        uint_fast64_t r = 1;
        for(int j = 1; j <= ((n - 1)/2); ++j)
            r *= a;
        int s = jacobi(a, n);

        if(((r - s) % n) != 0)
            return false;
    }

    return true;
}
```

Код алгоритма, реализующего тест Миллера-Рабина:

```
ulong mulmod(ulong a, ulong b, ulong mod)
{
    ulong x = 0, y = a % mod;
    while (b > 0)
```

```

{
    if (b % 2 == 1)
    {
        x = (x + y) % mod;
    }
    y = (y * 2) % mod;
    b /= 2;
}
return x % mod;
}

```

```

ulong modulo(ulong base, ulong exponent, ulong mod)
{
    ulong x = 1;
    ulong y = base;
    while (exponent > 0)
    {
        if (exponent % 2 == 1)
            x = (x * y) % mod;
        y = (y * y) % mod;
        exponent = exponent / 2;
    }
    return x % mod;
}

```

```

bool m_r_test(ulong p)
{
    if (p < 2)

```



```

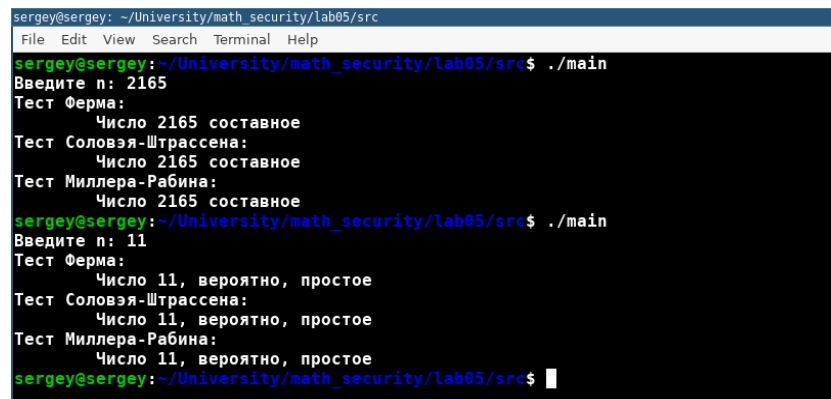
{
    return false;
}
if (p != 2 && p % 2 == 0)
{
    return false;
}
ulong s = p - 1;
while (s % 2 == 0)
{
    s /= 2;
}
for (int i = 0; i < TEST_ITERATIONS; i++)
{
    ulong a = rand() % (p - 1) + 1, temp = s;
    ulong mod = modulo(a, temp, p);
    while (temp != p - 1 && mod != 1 && mod != p - 1)
    {
        mod = mulmod(mod, mod, p);
        temp *= 2;
    }
    if (mod != p - 1 && temp % 2 == 0)
    {
        return false;
    }
}
return true;
}

```

Полный листинг программного кода представлен в файле main.cpp (архив

lab05, директория src).

Скомпилируем и запустим программу fig. 3.1.



```
sergey@sergey: ~/University/math_security/lab05/src
File Edit View Search Terminal Help
sergey@sergey:~/University/math_security/lab05/src$ ./main
Введите n: 2165
Тест Ферма:
    Число 2165 составное
Тест Соловья-Штрассена:
    Число 2165 составное
Тест Миллера-Рабина:
    Число 2165 составное
sergey@sergey:~/University/math_security/lab05/src$ ./main
Введите n: 11
Тест Ферма:
    Число 11, вероятно, простое
Тест Соловья-Штрассена:
    Число 11, вероятно, простое
Тест Миллера-Рабина:
    Число 11, вероятно, простое
sergey@sergey:~/University/math_security/lab05/src$
```

Figure 3.1: Вероятностные алгоритмы проверки чисел на простоту

4 Выводы

В ходе лабораторной работы мы реализовывали вероятностные алгоритмы проверки чисел на простоту.