Лабораторная работа №5

Математические основы защиты информации и информационной безопасности

Минов Кирилл Вячеславович | НПМмд-02-23

Содержание

# 1 Цель работы

Реализовать на языке программирования вероятностные алгоритмы проверки чисел на простоту.

# 2 Теоретическое введение

**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).

# 3 Выполнение лабораторной работы

Тест Ферма реализуем по следующей схеме:

а вход подается нечетное целое число n>= 5;

1)Выбрать случайное целое число a 2<= a <= n - 2;

2)Вычислить r <- a^(n-1) (mod n)

3)При r = 1 результат: "Число, вероятно, простое". В противном случае результат: "Число составное"

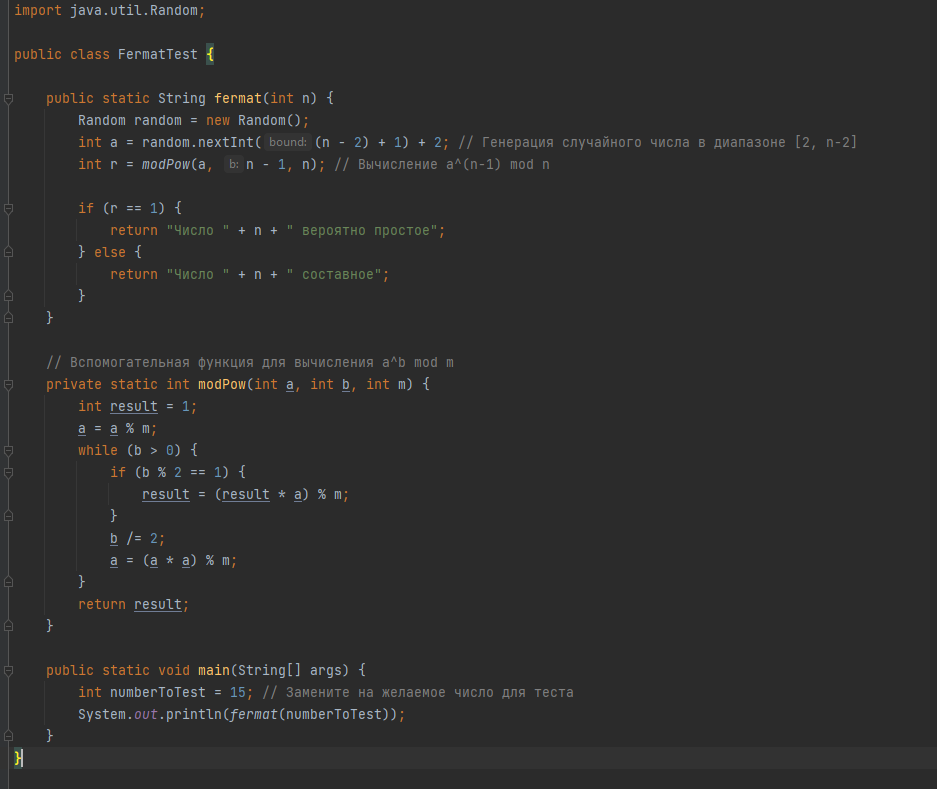


Рис. 1: Тест Ферма

Вычисление символа Якоби реализуем по следующей схеме:

Инициализация: Создание класса JacobiSymbol. Определение статической функции jacobiSymbol, которая принимает два целых числа a и n и возвращает символ Якоби (a/n). В функции main происходит тестирование алгоритма для заданных значений a и n.

Базовые случаи: Если n не положительное нечетное число или n четное, выбрасывается исключение IllegalArgumentException. Если a равно 0, возвращается 1, если n равно 1, и 0 в противном случае. Если a равно 1, возвращается 1.

Свойства символа Якоби: Если a отрицательно, устанавливается знак в зависимости от значения n % 4. Если a четное, вычисляется знак в зависимости от значения n % 8. Если a нечетное и неотрицательное, применяется критерий взаимной простоты.

Критерий взаимной простоты: Если a % 4 == 3 и n % 4 == 3, меняется знак. Рекурсивный вызов функции jacobiSymbol с аргументами n % a и a.

Тестирование: Задание значений a и n. Вызов функции jacobiSymbol с заданными значениями. Вывод результата. Если возникает исключение IllegalArgumentException, выводится сообщение об ошибке

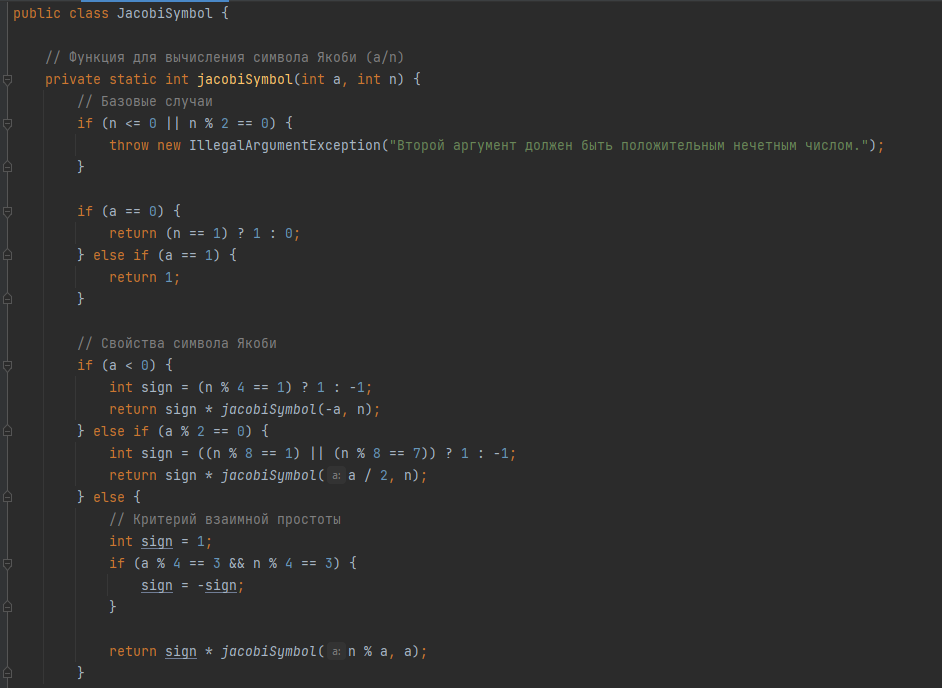


Рис. 2: Вычисление символа Якоби

Тест Соловэя-Штрассена реализуем по следующей схеме:

Инициализация: Создание класса SolovayStrassenTest. Определение статических функций jacobiSymbol и modularExponentiation для вычисления символа Якоби и модульного возведения в степень соответственно.

Вычисление символа Якоби: Функция jacobiSymbol(a, n) вычисляет символ Якоби для целых чисел a и n. Рекурсивные вызовы и базовые случаи рассматриваются в соответствии с определением символа Якоби.

Модульное возведение в степень: Функция modularExponentiation(a, exponent, n) использует алгоритм быстрого возведения в степень.

Тест Соловея-Штрассена: Функция solovayStrassenTest(n, iterations) проверяет простоту числа n. Проверяется, что n больше 2, и в случае, если n равно 2, возвращается true. Выполняется цикл с заданным количеством итераций: Генерируется случайное целое число a в интервале [2, n-2]. Вычисляется символ Якоби jacobi Проверяется условие теста Соловея-Штрассена. Если не выполняется, возвращается false. Если условие прошло для всех итераций, возвращается true, что означает, что n вероятно простое число.

Тестирование: В функции main выбирается число n, которое нужно проверить на простоту, и указывается количество итераций. Вызывается функция solovayStrassenTest для проверки простоты n. Выводится результат проверки. Если число вероятно простое, выводится "вероятно простое число", в противном случае - "составное число".

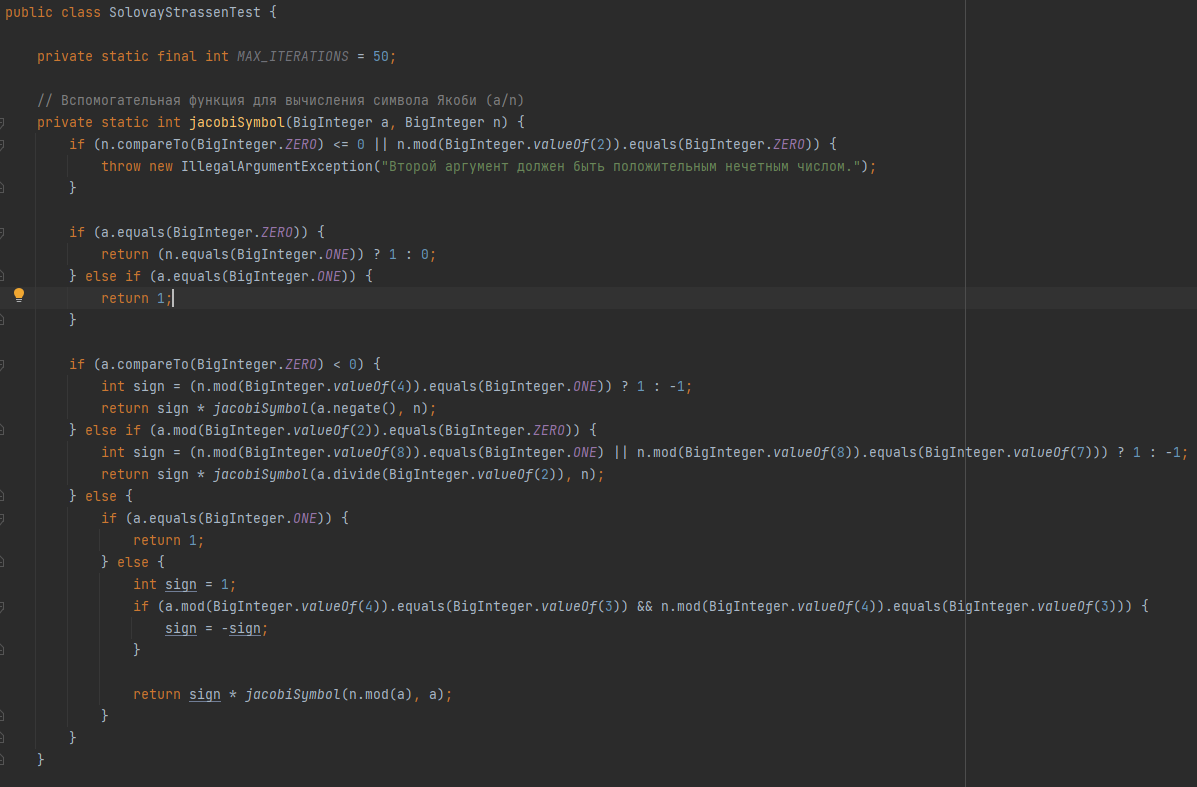


Рис. 3: Тест Соловэя-Штрассена



Рис. 4: Тест Соловэя-Штрассена



Рис. 5: Тест Соловэя-Штрассена

Тест Миллера-Рабина реализуем по следующей схеме:

Инициализация:

Создание класса MillerRabinTest. Определение статической функции power Определение статической функции millerRabinTest для проверки простоты числа.

Вычисление степени по модулю: Функция power(a, b, m) вычисляет с использованием алгоритма быстрого возведения в степень.

Функция millerRabinTest(n, iterations) проверяет простоту числа n. Проверяется, что n больше 2. Если n равно 2, возвращается true (2 - простое число). Если n четное (кроме 2), возвращается false, так как четные числа (кроме 2) не являются простыми.

Выполняется цикл с заданным количеством итераций: Выбирается случайное число a в интервале [2, n-2]. Проверяется условие Миллера-Рабина. Если выполняется, переход к следующей итерации. Если условия не выполняются, возвращается false (n - составное число). Если все итерации прошли успешно, возвращается true (n вероятно простое).

Тестирование: В функции main выбирается число n, которое нужно проверить на простоту, и указывается количество итераций. Вызывается функция millerRabinTest для проверки простоты n. Выводится результат проверки. Если число вероятно простое, выводится "вероятно простое число", в противном случае - "составное число".

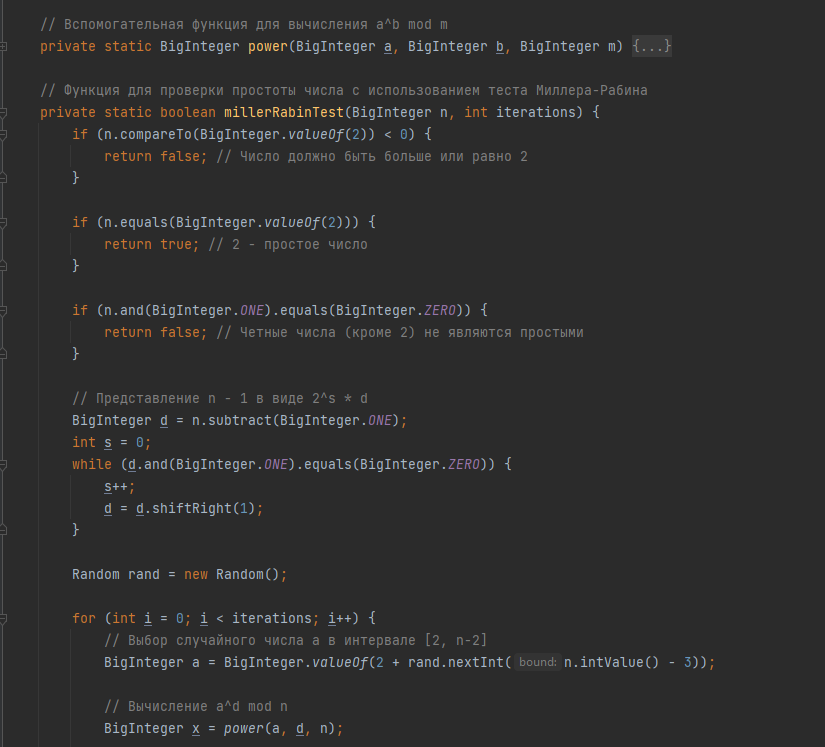


Рис. 6: Тест Миллера-Рабина

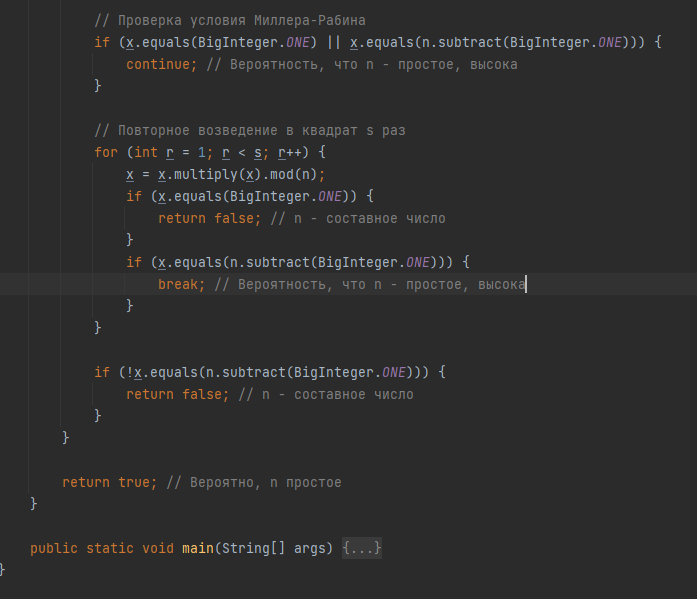


Рис. 7: Тест Миллера-Рабина

# 4 Выводы

В ходе выполнения данной лабораторной работы были реализованы вероятностные алгоритмы проверки чисел на простоту.