МИНОБРНАУКИ РОССИИ
ФГБОУ ВО «СГУ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

АЛГОРИТМЫ АЛГЕБРЫ И ТЕОРИИ ЧИСЕЛ

ЛАБОРАТОРНАЯ РАБОТА №10

студента 4 курса 431 группы
направления 10.05.01 — Компьютерная безопасность
факультета КНиИТ
Никитина Арсения Владимировича

Проверил
доцент ____________ А. С. Гераськин

Саратов 2022

СОДЕРЖАНИЕ

1 Задание лабораторной работы

Осуществить построение большого простого числа с использованием теоремы Поклингтона.

2 Теоретическая часть

Теорема Поклингтона

Пусть $n = q^k R + 1$ где q — простое число, $k \geqslant 1$. Если существует такое целое число a, что $a^{n-1} \equiv 1 \pmod n$ и НОД$(a^{(n-1)/q} - 1, n) = 1$, то каждый простой делитель p числа n имеет вид $p = q^k r + 1$ при некотором натуральном r.

Доказательство

Пусть p — простой делитель числа n. Тогда из условия теоремы вытекает, что $a^{n-1} = 1 \pmod p$ и $a^{(n-1)/q} \neq 1 \pmod p$. Отсюда получаем, что порядок m элемента a по модулю p удовлетворяет условиям: $n-1 = md$, где d — некоторое целое. Допустим, q делит d. В этом случае $(n-1)/q = m(d/q)$, где (d/q) — целое. Следовательно $a^{(n-1)/q} = 1 \pmod p$, что невозможно. Поскольку $n - 1 = md = q^k R$, то m делится на q^k. Однако m должно делить число $p - 1$. Следовательно, $p = q^k r + 1$ при некотором r. Теорема доказана.

Критерий Поклингтона

Пусть n — натуральное число. Пусть число $n - 1$ имеет простой делитель q, причем $q > \sqrt{n} - 1$. Если найдётся такое целое число a, что выполняются следующие два условия:

1. $a^{n-1} \equiv 1 \pmod n$,
2. числа n и $a^{(n-1)/q} - 1$ взаимнопросты,

то n — простое число.

Доказательство

Предположим, что n является составным числом. Тогда существует простое число p — делитель n, причем $p < \sqrt{n}$. Заметим, что $q > p - 1$, следовательно q и $p - 1$ — взаимнопросты. Следовательно, существует некоторое целое число u, такое, что $uq \equiv 1 \pmod{p-1}$. Но в таком случае $a^{(n-1)/q} \equiv a^{uq(n-1)/q} = a^{u(n-1)} \equiv 1 \pmod p$ (в силу условия 1). Но таким образом получено противоречие условию 2. Следовательно, n является простым числом.

3 Практическая часть

3.1 Пример работы алгоритма

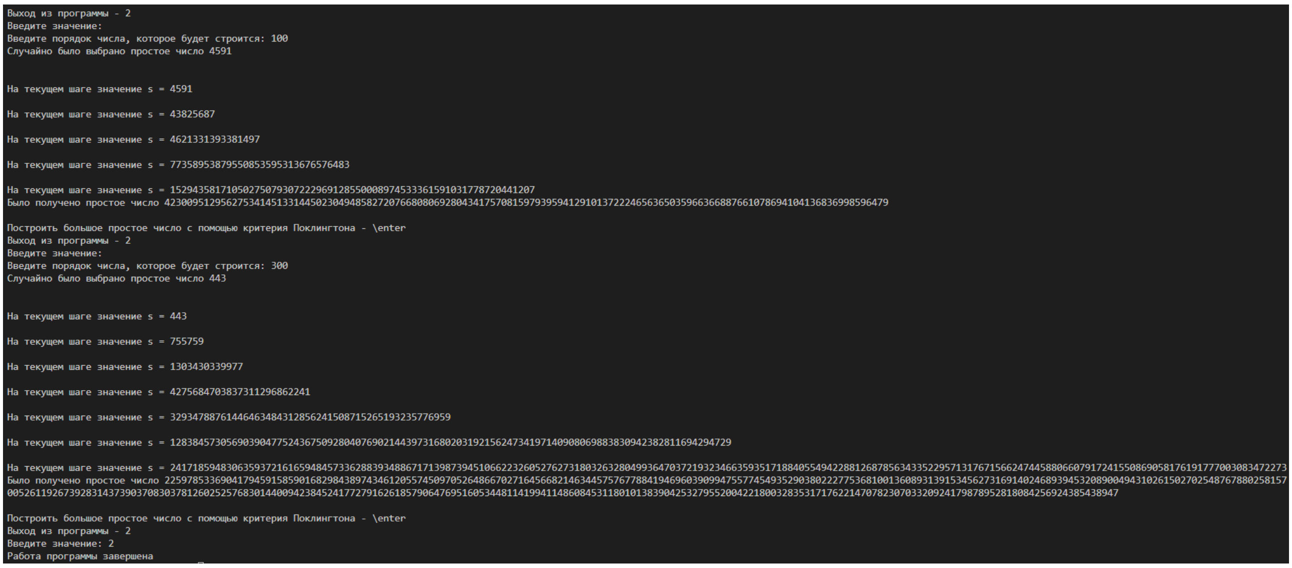

Рисунок 1

3.2 Код программы, реализующей рассмотренный алгоритм

```
from sympy import *
import random


def gcd(a, b):
    if b == 0:
        return a
    else:
        return gcd(b, a % b)


def modula_power(a, power, modula):

    b = 1
    while power:
        if not power % 2:
            power //= 2
            a = (a * a) % modula
        else:
            power -= 1
            b = (b * a) % modula
    return b
```

```python


def pocklington_criterion(n):

    limit = 10 ** n

    primes = list(primerange(10000))

    s = random.choice(primes)
    print(f'Случайно было выбрано простое число {s}\n')

    right = 2 * (2 * s + 1)

    while s < limit:
        print(f'\nНа текущем шаге значение s = {s}')

        while True:

            r = random.randrange(s + 1, right, 2)
            # print(f'Случайно было выбрано четное число {r}') #\u2208 [{s}, {right}]\n')

            n = s * r + 1
            # print(f'На простоту проверяется число {n}\n')

            flag = False
            for prime in primes:
                if not n % prime:
                    flag = True
                    # print(f'Число оказалось не простым, так как имееет '\
                        # f'делитель ({prime})\n')
                    break

            if flag:
                continue

            while True:

                a = random.randint(2, n - 1)
                # print(f'Случайно было выбрано число {a}') #\u2608 [{2}, {n - 1}]\n')
```

```

                if modula_power(a, n - 1, n) != 1:
                    # print(f'Число {n} оказалось непростым, так как '\
                    #     f'{a} ^ {n - 1} \u2241 1 (mod {n})\n')
                    break
                else:
                    pass
                    #  print(f'{a} ^ {n - 1} \u2263 1 (mod {n})\n')

                d = gcd((modula_power(a, r, n) - 1), n)

                if d != n:
                    if d == 1:
                        # print(f'Переход к следующей итерации алгоритма \n')
                        s = n
                        right = 2 * (2 * s + 1)
                        break
                    else:
                        continue
                else:
                    continue

            if s == n:
                 break
    return s


def main():

    while True:

        print('\nПостроить большое простое число с помощью критерия Поклингтона - \enter')

        print('Выход из программы - 2')

        try:
            value = int(input('Введите значение: '))

        except ValueError:
            value = 1
```

```

        if value == 1:
            n = int(input('Введите порядок числа, которое будет строится: '))
            generated_prime = pocklington_criterion(n)
            print(f'Было получено простое число {generated_prime}')

        elif value == 2:
            print('Работа программы завершена')
            return


if __name__ == "__main__":
    main()
```