

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

СОДЕРЖАНИЕ

1	Задание лабораторной работы	3
2	Теоретическая часть	4
3	Практическая часть.....	5
3.1	Пример работы алгоритма.....	5
3.2	Код программы, реализующей рассмотренный алгоритм	5

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, $k \geq 1$. Если существует такое целое число a , что $a^{n-1} \equiv 1 \pmod{n}$ и $\text{НОД}(a^{(n-1)/q} - 1, n) = 1$, 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 $a^{n-1} \equiv 1 \pmod{p}$ и $a^{(n-1)/q} \not\equiv 1 \pmod{p}$. 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 $a^{(n-1)/q} \equiv 1 \pmod{p}$, 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

```
Выход из программы - 2
Введите значение:
Введите порядок числа, которое будет строиться: 100
Случайно было выбрано простое число 4591

На текущем шаге значение s = 4591
На текущем шаге значение s = 43825687
На текущем шаге значение s = 4621331393381497
На текущем шаге значение s = 77358953879550853595313676576483
На текущем шаге значение s = 152943581710502750793072229691285500808974533361591031778720441207
Было получено простое число 42300951295627534145131445023049485827207668080692804341757081597939594129101372224656365035966366887661078694104136836998596479
Построить большое простое число с помощью критерия Полинтона - \enter
Выход из программы - 2
Введите значение:
Введите порядок числа, которое будет строиться: 300
Случайно было выбрано простое число 443

На текущем шаге значение s = 443
На текущем шаге значение s = 755759
На текущем шаге значение s = 1303430339977
На текущем шаге значение s = 4275684703837311296862241
На текущем шаге значение s = 32934788761446463484312856241508715265193235776959
На текущем шаге значение s = 1283845730569039047752436750928040769021443973168020319215624734197140908069883830942382811694294729
На текущем шаге значение s = 241718504830635937216165048457336288393488671713987394510662732605276273180326328049936470372193234663593517188405549422881268785634335229571317671566247445880660791724155086905817619177003083472273
Было получено простое число 22597853369041794591585981682984389743461205574509705264866702716456682146344575767788419469603909947557745493529038022277536810013608931391534562731691402468939453208900494310261502702548767880258157
005261192673928314373903708303781260252576830144009423845241772791626185790647695160534481141994114860845311801013839042532795520042218003283531717622147078230703320924179878952818084256924385438947
Построить большое простое число с помощью критерия Полинтона - \enter
Выход из программы - 2
Введите значение: 2
Работа программы завершена
```

Рисунок 1

3.2 Код программы, реализующей рассмотренный алгоритм

```
1 from sympy import *
2 import random
3
4
5 def gcd(a, b):
6     if b == 0:
7         return a
8     else:
9         return gcd(b, a % b)
10
11
12 def modula_power(a, power, modula):
13
14     b = 1
15     while power:
16         if not power % 2:
17             power //= 2
18             a = (a * a) % modula
19         else:
20             power -= 1
21             b = (b * a) % modula
22     return b
```

```

23
24
25 def pocklington_criterion(n):
26
27     limit = 10 ** n
28
29     primes = list(primerange(10000))
30
31     s = random.choice(primes)
32     print(f'Случайно было выбрано простое число {s}\n')
33
34     right = 2 * (2 * s + 1)
35
36     while s < limit:
37         print(f'\nНа текущем шаге значение s = {s} ')
38
39         while True:
40
41             r = random.randrange(s + 1, right, 2)
42             # print(f'Случайно было выбрано четное число {r}') #\u2208 [{s},
43             ↪ {right}]\n')
44
45             n = s * r + 1
46             # print(f'На простоту проверяется число {n}\n')
47
48             flag = False
49             for prime in primes:
50                 if not n % prime:
51                     flag = True
52                     # print(f'Число оказалось не простым, так как имеет '\
53                     # f'делитель ({prime})\n')
54                     break
55
56             if flag:
57                 continue
58
59             while True:
60
61                 a = random.randint(2, n - 1)
62                 # print(f'Случайно было выбрано число {a}') #\u2608 [{2}, {n -
63                 ↪ 1}]\n')

```

```

62
63         if modula_power(a, n - 1, n) != 1:
64             # print(f'Число {n} оказалось непростым, так как '\
65             #       f'{a} ^ {n - 1} \u2241 1 (mod {n})\n')
66             break
67         else:
68             pass
69             # print(f'{a} ^ {n - 1} \u2263 1 (mod {n})\n')
70
71     d = gcd((modula_power(a, r, n) - 1), n)
72
73     if d != n:
74         if d == 1:
75             # print(f'Переход к следующей итерации алгоритма \n')
76             s = n
77             right = 2 * (2 * s + 1)
78             break
79         else:
80             continue
81     else:
82         continue
83
84     if s == n:
85         break
86     return s
87
88
89 def main():
90
91     while True:
92
93         print('\n Построить большое простое число с помощью критерия
94         ↪ Поклингтона - \n enter')
95
96         print('Выход из программы - 2')
97
98         try:
99             value = int(input('Введите значение: '))
100
101         except ValueError:
102             value = 1

```

```
102
103     if value == 1:
104         n = int(input('Введите порядок числа, которое будет строиться: '))
105         generated_prime = pocklington_criterion(n)
106         print(f'Было получено простое число {generated_prime}')
107
108     elif value == 2:
109         print('Работа программы завершена')
110         return
111
112
113 if __name__ == "__main__":
114     main()
```