МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский Авиационный Институт»

(Национальный Исследовательский Университет)

Институт: №8 «Информационные технологии и прикладная математика» Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа № 3 по курсу

«Криптография»

Группа: М8О-306Б-20

Студент: И. П. Попов

Преподаватель: А. В. Борисов

Оценка:

Дата:

Москва, 2022

# **Факторизация числа**

Задача:

Разложить число на нетривиальные сомножители.

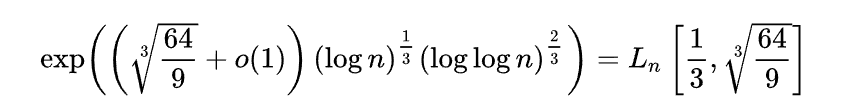
Вариант выбрать следующим образом: свое ФИО подать на вход в хеш-функцию, являющуюся стандартом, выход хеш-функции представить в шестнадцатеричном виде и рассматривать младший байт как номер варианта. В отчете привести подробности процесса вычисления номера варианта.

# **Применение в криптографии**

Предполагаемая большая [вычислительная сложность](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C) задачи факторизации лежит в основе [криптостойкости](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D1%82%D0%BE%D0%B9%D0%BA%D0%BE%D1%81%D1%82%D1%8C) некоторых [алгоритмов шифрования с открытым ключом](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81_%D0%B0%D1%81%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D1%8B%D0%BC_%D0%BA%D0%BB%D1%8E%D1%87%D0%BE%D0%BC), таких как [RSA](https://ru.wikipedia.org/wiki/RSA). Более того, если известен хотя бы один из параметров ключей RSA, то система взламывается однозначно, кроме того, существует множество алгоритмов восстановления всех ключей в системе, обладая какими-то данными.

# **Общий метод решета числового поля**

**Общий метод решета числового поля** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *general number field sieve*, GNFS) — метод [факторизации целых чисел](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D1%86%D0%B5%D0%BB%D1%8B%D1%85_%D1%87%D0%B8%D1%81%D0%B5%D0%BB). Является наиболее эффективным алгоритмом факторизации чисел длиной более 110 десятичных знаков. Сложность алгоритма оценивается эвристической формулой



Метод является обобщением специального метода решета числового поля: тогда как последний позволяет факторизовать числа только некоторого специального вида, общий метод работает на множестве целых чисел, за исключением степеней простых чисел (которые факторизуются тривиально извлечением корней).

## Гладкие числа

В [теории чисел](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D1%87%D0%B8%D1%81%D0%B5%D0%BB) **гладким числом** называется [целое число](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE), все [простые делители](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%81%D1%82%D0%BE%D0%B9_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B8%D1%82%D0%B5%D0%BB%D1%8C) которого малы. Поскольку понятие «делители малы» может быть истрактовано вольно, чаще всего гладким числом называют такое, чьи простые делители не превосходят 10 (то есть, по сути равны 2, 3, 5 или 7).

[Натуральное число](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%82%D1%83%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE) называется *B*-**гладким**, если все его простые делители не превосходят *B*.

Число 2000 имеет следующее разложение на множители: 24 × 53. Поэтому 2000 — это 5-гладкое число, а также 6-гладкое число и так далее, но не 4-гладкое.

## Суть метода

Метод решета числового поля (как специальный, так и общий) можно представить как усовершенствование более простого метода — метода рационального решета либо метода квадратичного решета. Подобные им алгоритмы требуют нахождения [гладких чисел](https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D0%B0%D0%B4%D0%BA%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE) порядка . Размер этих чисел экспоненциально растёт с ростом n. Метод решета числового поля, в свою очередь, требует нахождения гладких чисел субэкспоненциального относительно n размера. Благодаря тому, что эти числа меньше, вероятность того, что число такого размера окажется гладким, выше, что и является причиной эффективности метода решета числового поля. Для достижения ускорения вычислений в рамках метода проводятся в [числовых полях](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE%D0%B2%D0%BE%D0%B5_%D0%BF%D0%BE%D0%BB%D0%B5), что усложняет алгоритм, по сравнению с более простым рациональным решетом.

## Основные принципы

* [Метод факторизации Ферма](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8_%D0%A4%D0%B5%D1%80%D0%BC%D0%B0) для факторизации натуральных нечетных чисел *n*, состоящий в поиске таких целых чисел *x* и *y*, что  что ведет к разложению 
* Нахождение подмножества множества целых чисел, произведение которых — квадрат
* Составление факторной базы: набора , где *pi* — простые числа, такие, что *pi* <= *B* для некоторого *B*.
* Просеивание выполняется подобно [решету Эратосфена](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%88%D0%B5%D1%82%D0%BE_%D0%AD%D1%80%D0%B0%D1%82%D0%BE%D1%81%D1%84%D0%B5%D0%BD%D0%B0) (откуда метод и получил своё название). Решетом служат простые числа факторной базы и их степени. При просеивании число не «вычеркивается», а делится на число из решета. Если в результате число оказалось единицей, то оно *B*-гладкое.
* Основная идея состоит в том, чтобы вместо перебора чисел и проверки, делятся ли их квадраты по модулю *n* на простые числа из факторной базы, перебираются простые числа из базы и сразу для всех чисел вида  проверяется, делятся ли они на это простое число или его степень.

# **Ход работы**

Для вычисление номера своего варианта я подал свое ФИО на вход в хеш-функцию, являющуюся стандартом языка Java, выход хеш-функции представил в шестнадцатеричном виде

*import java.util.Objects;*

*public class CR\_lab2 {*

*private String str;*

*@Override*

*public boolean equals(Object o) {*

*if (this == o) return true;*

*if (o == null || getClass() != o.getClass()) return false;*

*CR\_lab2 cr\_lab2 = (CR\_lab2) o;*

*return Objects.equals(str, cr\_lab2.str);*

*}*

*@Override*

*public int hashCode() {*

*return Objects.hash(str);*

*}*

*public void setStr(String str) {*

*this.str = str;*

*}*

*public static void main(String[] args) {*

*CR\_lab2 cr\_lab2 = new CR\_lab2();*

*cr\_lab2.setStr("Попов Илья Павлович");*

*int hash = cr\_lab2.hashCode();*

***System.out.println(hash);***

***System.out.println(Integer.toHexString(hash));***

*}*

*}*

D:\w\_dev\jdk-19.0.2\bin\java.exe "-javaagent:D:\w\_dev\IntelliJ IDEA 2022.3.2\lib\idea\_rt.jar=62545:D:\w\_dev\IntelliJ IDEA 2022.3.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath "D:\w\_dev\java projects\untitled\out\production\untitled" CR\_lab2

**-464930166**

**e449ba8a**

# 

Первой строкой вывода я распечатал сам хэш-код моей строки, а второй - этот хэш-код, представленный в 16-ричной системе.

Младшие разряды - 8a => вариант:

**n(8a)=470159777969277756545806729184364929095652721684184499136482560232872665495718735408182744373048785161383519747237403915113362306454610013529906923361700377416302467197972392758257181515051095757893304064613238956668135666009448930868653279737748581275896969378506326643807790360971638736424181744421156622531**

Следующий:

**n(8b)=599660627714014218191479082190875958082244100774598479292922074303241589048878390573221723515646241641177891270007882991802618514532888656736287309284426331738378502922927526108529093739996883775411450825397598242674462098425922591579456689745923228206381033637208676121493023263770299958459503311720216805007**

Прямая факторизация настолько больших чисел невозможна, поэтому была применена следующая хитрость: был найден НОД (число А) чисел из моего варианта и следующего за ним. После этого мое число было поделено на этот самый НОД и частным оказалось второе число(число B).

Числа А и В были проверены на простоту тестом Ферма и оба оказались простыми. Следовательно, они и будут являться ответом задачи.

## Код на python:

**from math import gcd**

**import random**

**def bin\_pow(a, n):**

**res = 1**

**while n > 0:**

**if n % 2 == 1:**

**res = res \* a**

**a = a \* a**

**n //= 2**

**return res**

**def is\_prime(num, test\_count):**

**for i in range(test\_count):**

**rnd = random.randint(1, num - 1)**

**if gcd(num, rnd) != 1:**

**return False**

**return True**

**myVar = 470159777969277756545806729184364929095652721684184499136482560232872665495718735408182744373048785161383519747237403915113362306454610013529906923361700377416302467197972392758257181515051095757893304064613238956668135666009448930868653279737748581275896969378506326643807790360971638736424181744421156622531**

**nextVar = 599660627714014218191479082190875958082244100774598479292922074303241589048878390573221723515646241641177891270007882991802618514532888656736287309284426331738378502922927526108529093739996883775411450825397598242674462098425922591579456689745923228206381033637208676121493023263770299958459503311720216805007**

**d1 = gcd(myVar, nextVar)**

**d2 = myVar // d1**

**print("Проверка на простоту")**

**print("Первый делитель:", is\_prime(d1, 100))**

**print("Второй делитель:", is\_prime(d2, 100))**

**print("\nВывод делителей")**

**print("Первый делитель:", d1)**

**print("Второй делитель:", d2)**

**Вывод в терминал:**

Проверка на простоту

A: True

B: True

Ответ:

A: 217451550740883846561507857729403011932884658895029895125998624235271671431311

B: 2162135778601653030788482770592016485599709367908319834911961901953710520584365748562332750593721290665201743488404082958580998366475524246444509132125378784366844996632330863616116763429443638317975311580302162165174589438937915021

# **Выводы**

В ходе выполнения данной лабораторной работы я узнал границы работы алгоритмов факторизации и, изучил метод Ферма для проверки числа на простоту, асимптотика которого O(k \* log n), где k – число тестов, n – проверяемое число.