**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 № 2   
по курсу «Криптография»

Группа: М8О-306Б-22

Студент: А. Д. Галкин

Преподаватель: А. В. Борисов

Оценка:

Дата: 10.04.2025

Москва 2025

**ОГЛАВЛЕНИЕ**

[1 Тема 3](#_Toc3628)

[2 Задание 3](#_Toc6043)

[3 Теория 4](#_Toc5100)

[4 Ход лабораторной работы 4](#_Toc30475)

[5 Выводы 13](#_Toc8106)

[6 Список используемой литературы 14](#_Toc24902)

# **Тема**

Факторизация больших целых чисел.

# **Задание**

Строку в которой записано своё ФИО подать на вход в качестве аргумента хеш-функции ГОСТ Р 34.11-2012 (Стрибог). Младшие 8 бит выхода интерпретировать как число, которое в дальнейшем будет номером варианта от 0 до 255. В отчёт включить снимок экрана с выбором номера варианта, а также описать шаги решения задачи.

Задача: разложить каждое из чисел `a` и `b` на нетривиальные сомножители

# **Теория**

Факторизация целых чисел — это процесс разложения числа N на нетривиальные множители, то есть на натуральные числа (больше 1 и меньше N), произведение которых даёт исходное число. Эта задача играет ключевую роль в теории чисел и широко применяется в криптографии, например, в алгоритме RSA.

Согласно основной теореме арифметики, любое целое число N > 1 можно единственным образом представить в виде произведения простых чисел. Однако для составных чисел (не являющихся простыми) нахождение такого разложения может быть вычислительно сложной задачей, особенно если N очень большое.

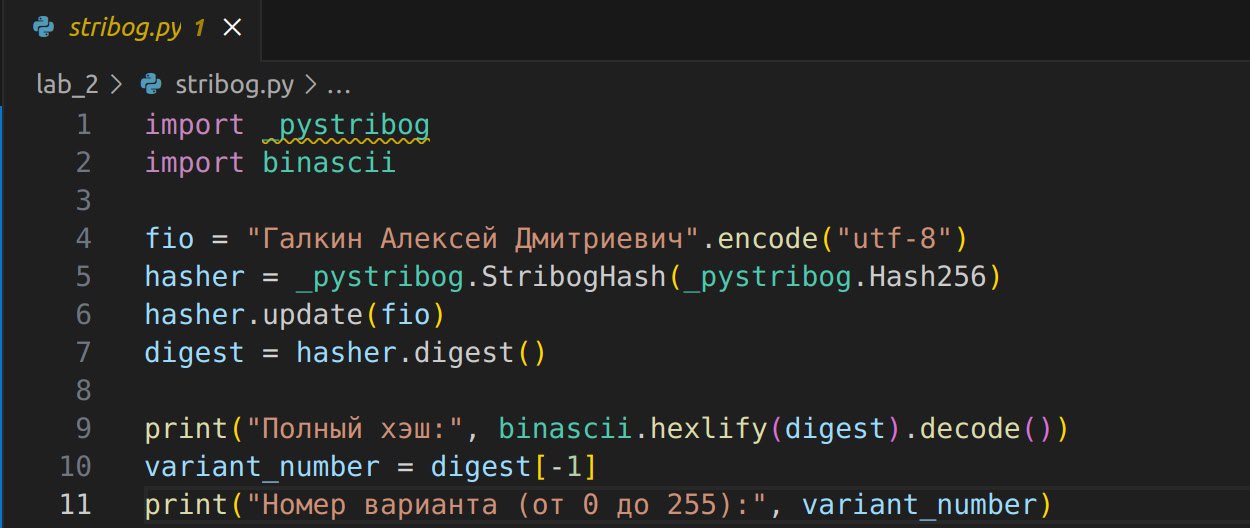
Простейший метод факторизации — перебор делителей — имеет экспоненциальную сложность относительно количества цифр в числе, что делает его неэффективным для больших чисел. Поэтому на практике применяют более продвинутые алгоритмы, такие как:

1. Метод Полларда "ро" (Pollard’s rho)
2. Квадратичное решето
3. Общий метод решета числового поля (GNFS)

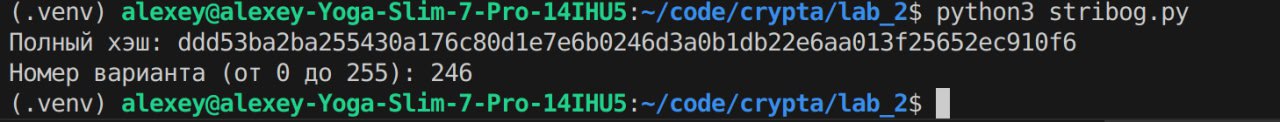
Другой способ — поиск наибольшего общего делителя (НОД) числа N и некоторого другого числа, что в некоторых случаях позволяет быстрее находить нетривиальные делители.

# **Ход лабораторной работы**

Для выбора варианта необходимо было передать в качестве аргумента свое ФИО хеш-функции ГОСТ Р 34.11-2012 (Стрибог), а также выделить младшие 8 бит и интерпретировать соответствующее число в свой номер варианта. Для этого была написана следующая программа на Python, с использованием библиотеки pystribog

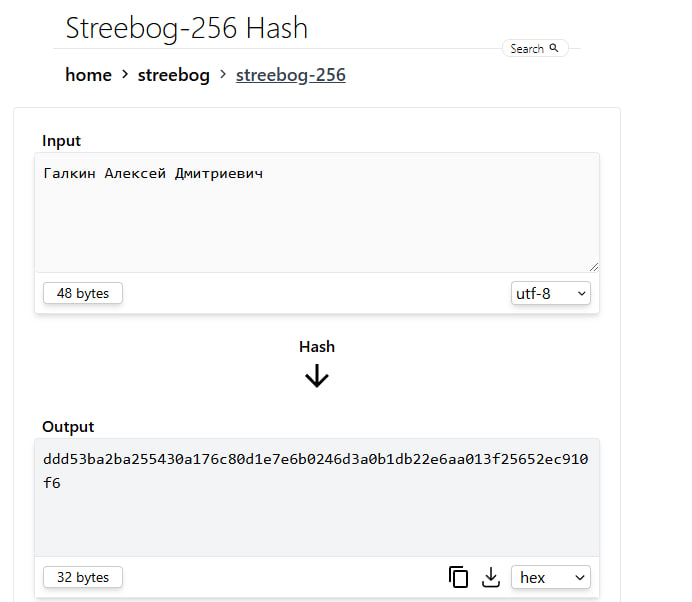


Результат работы программы представлен ниже, соответственно, мой вариант - 2**46**.



Чтобы сверить рассчёты был использован следующий веб-сайт - <https://hashing.tools/streebog/streebog-256>. Результат правильный, хэш совпадает:

Теперь переходим к заданию. Для задания А нужно было факторизовать следующее число:

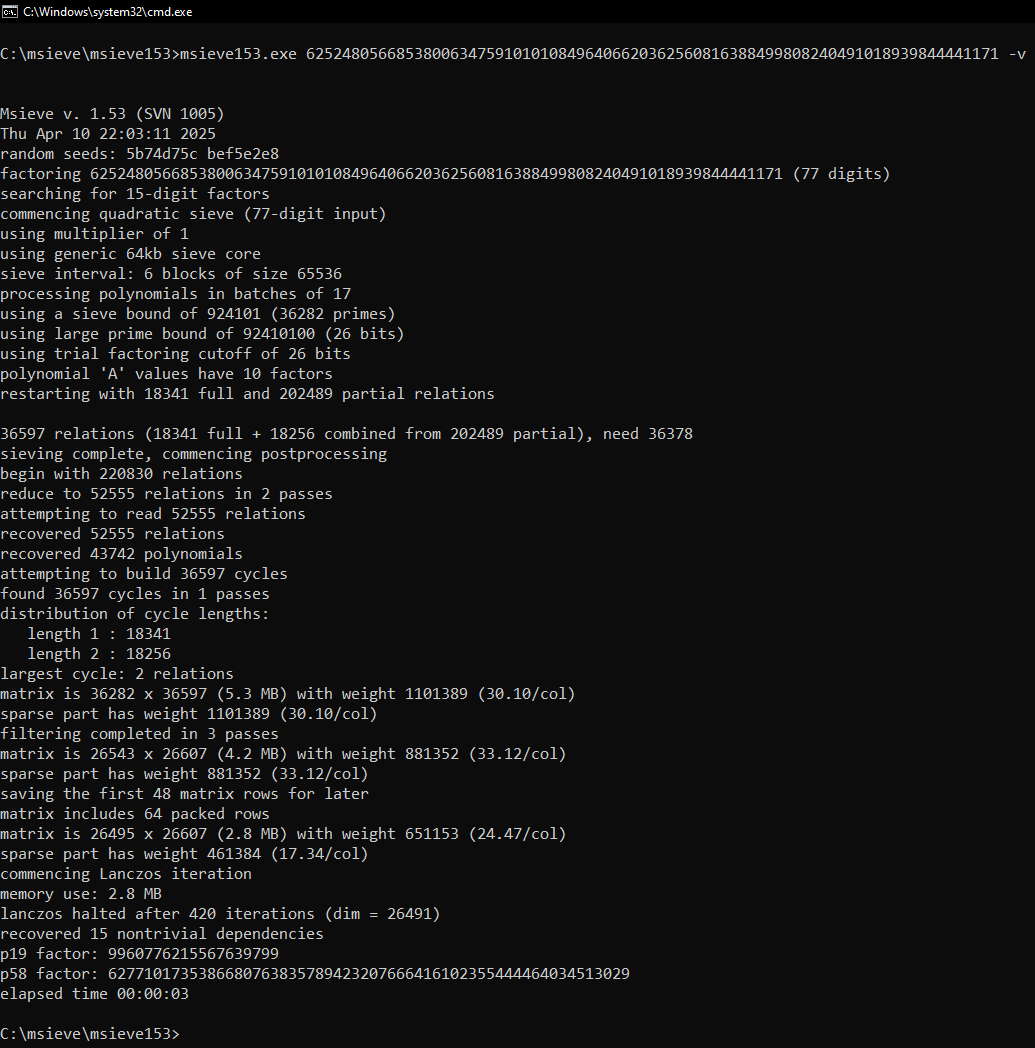


a[246]=62524805668538006347591010108496406620362560816388499808240491018939844441171

Для этого были использованы готовые средства в виде библиотеки msieve. msieve — это библиотека для факторизации целых чисел, которая объединяет в себе сразу несколько алгоритмов. Общая стратегия работы такова, что сначала выполняется поиск малых делителей, а если их не удаётся найти, то для более сложных (больших) чисел используются «тяжёлые» алгоритмы. Более детальная схема работы msieve:

1. Предварительный поиск малых делителей: проверка на деление, Pollard Rho, методы ECM (в том числе P–1/P+1) – этот этап эффективен для чисел с малыми делителями и обычно для чисел примерно до 25 цифр.
2. Далее применяется самоинициализирующееся квадратичное сито (SIQS), которое является быстрым и оптимизированным для чисел диапазона примерно до 110–120 цифр.
3. При необходимости факторизации очень больших чисел: может быть использована реализация GNFS, встроенная в msieve, которая способна справляться с числами значительно большего размера.

Для факторизации заданного числа была использована программа msieve, установленная в рабочую среду Windows. После скачивания и распаковки архива с программой была открыта командная строка, выполнено перемещение в директорию с программой с помощью команды cd. Для запуска процесса факторизации использовалась команда msieve153.exe -v с указанием факторизуемого числа, где флаг -v активирует подробный режим вывода информации. В этом режиме программа отображает детальные сведения о ходе вычислений: этапы выполнения, применяемые алгоритмы и промежуточные результаты. Это позволяет отслеживать процесс факторизации и анализировать работу алгоритмов.



Программа довольно быстро смогла найти два единственных простых сомножителя для числа из задачи А:

1. **9960776215567639799**
2. **6277101735386680763835789423207666416102355444464034513029**

Для задания Б мне нужно факторизовать следующее огромное целое число:

b[246]=32317006071311007300714876688669951960444102669715484032130345427524655138867890893197201411522913463688717960921898019494119559150490921095088152852154660741055510013902155562081361962081595981911420549243824209510848251746533244000107548822959940247783543758121002469477638485548747439577077807719695549325118927778354644932972475948282546066354431328951111490357896748592181747880280860229387786352173196204698411196098855229174987078109775516711215169207291754767289629699943437953970024590148152458155990164703123080559051765953877965319207279108861098880815297029014831152784536829471762315781781832781690092801

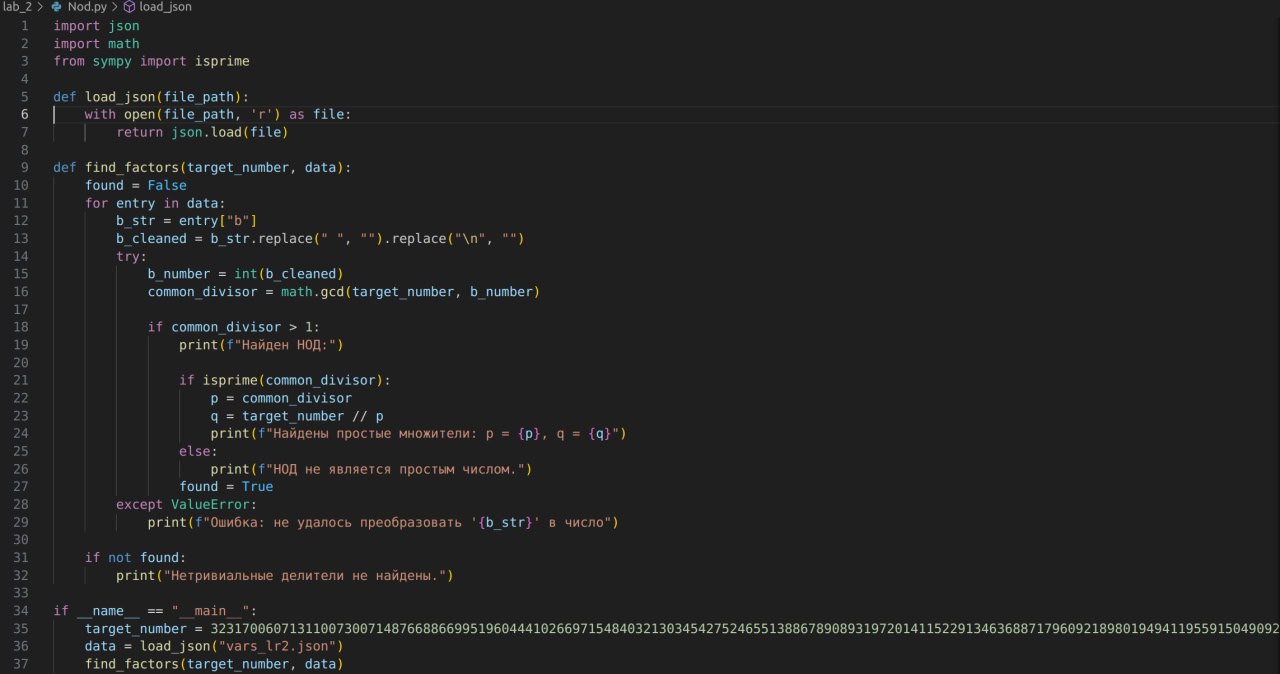
В данном случае использование программы msieve может оказаться неэффективным – либо не даст результата вовсе, либо потребует неприемлемо большого времени вычислений. Гораздо более эффективно будет искать НОД для заданного числа, используя числа из смежных вариантов до тех пор, пока мы не сможем найти простый делитель.

Почему это эффективно и зачем нам числа из смежных вариантов? Дело в том, что эффективный алгоритм Евклида позволяет быстро вычислить наибольший общий делитель двух чисел. При факторизации часто используется подход, при котором для большого числа N перебираются кандидаты X из некоторого набора, и вычисляется gcd⁡(N,X). Если результат оказывается нетривиальным делителем, то найден первый сомножитель, а второй можно найти, разделив N на полученный делитель. Благодаря логарифмической сложности алгоритма Евклида даже для больших чисел операция нахождения НОД остаётся быстрой. Легко заметить, что смежные варианты имеют структурно похожие друг на друга числа, что идеально подходит для составления некоторого набора X.

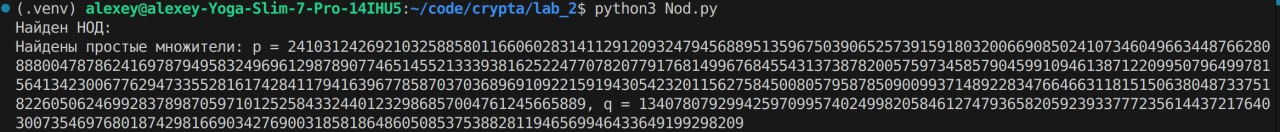
Алгоритм Евклида основан на свойстве, что наибольший общий делитель двух чисел a и b совпадает с наибольшим общим делителем числа b и остатка от деления a на b:

**gcd(a,b)=gcd(b, a  mod  b)**

Это утверждение позволяет последовательно уменьшать размер чисел, заменяя пару чисел на новую пару, где одно из чисел – это уже меньшее число, а второе — остаток от деления. Каждый шаг алгоритма существенно уменьшает размеры чисел, что ведёт к быстрой сходимости.



Имея вышеописанную информацию я составил следующую программу на Python. Необходимо задать исходное число для факторизации и содержимое файла целиком, кроме условия и чисел из своего варианта. Далее с помощью регулярного выражения содержимое файла обрабатывается таким образом, чтобы получить все b-числа без указаний вариантов. Собираем все числа в список и перебираем для поиска общего делителя до тех пор, пока он не будет найден. НОД вычисляем с помощью gcd из библиотеки math. Так же имеется проверка чисел на простоту с помощью isprime из библиотеки sympy.



Программа быстро смогла найти два единственных простых сомножителя для числа из задачи Б:

1. **2410312426921032588580116606028314112912093247945688951359675039065257391591803200669085024107346049663448766280888004787862416978794958324969612987890774651455213339381625224770782077917681499676845543137387820057597345857904599109461387122099507964997815641342300677629473355281617428411794163967785870370368969109221591943054232011562758450080579587850900993714892283476646631181515063804873375182260506246992837898705971012525843324401232986857004761101650501**
2. **1340780792994259709957402499820584612747936582059239337772356144372176403007354697680187429816690342769003185818648605085375388281194656994643364945847590**

# **Выводы**

В ходе лабораторной работы была решена задача факторизации больших целых чисел на нетривиальные сомножители. Для решения задачи А использовалась библиотека msieve, включающая в себя совокупность различных готовых решений от самых тривиальных до самых эффективных и сложных. Однако для решения задачи Б пришлось написать собственную программу поиска НОД для своего огромного целого числа, а также вспомнитть алгоритм Евклида и понять, почему это работает эффективно.

# **Список используемой литературы**

1. [https://ru.wikipedia.org/wiki/Факторизация\_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)
2. <https://github.com/ddulesov/pystribog>
3. [https://ru.wikipedia.org/wiki/Алгоритм\_Евклида](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%95%D0%B2%D0%BA%D0%BB%D0%B8%D0%B4%D0%B0)
4. <https://hashing.tools/streebog/streebog-256>
5. <https://github.com/upiter/msieve>