Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет: “Информационные технологии и прикладная математика”
Кафедра: 806 “Вычислительная математика и программирование”

Отчет по лабораторной работе №1

по курсу «Криптография»

Студент: Полей-Добронравова Амелия Вадимовна

Группа: М8О-307Б, № по списку 16.

Преподаватель: Борисов А.В.

Дата: 08.05.2021

Итоговая оценка:

Подпись преподавателя:

Москва, 2021

Условие

Разложить каждое из чисел n1 и n2 на нетривиальные сомножители.

Вариант №16
n1=24258741345568931180594169758210354434344402573793060972812930301130760182355 1,

n2=15109385843025147460686876803591387120848268695317498338161525361070299566943782 2
866501448480999328468063646504533658467000651269248205716885880525173052241243557553704763875918384943786116958217435310061676086144208333891116298297801865460907348745561834472564647434110644877018611946543743680554031457390231514801060564296939903623927
999086648137755263103834503833267130046044915082613304759940295270222043813232424080148048305599685013560938061277308857626493 9

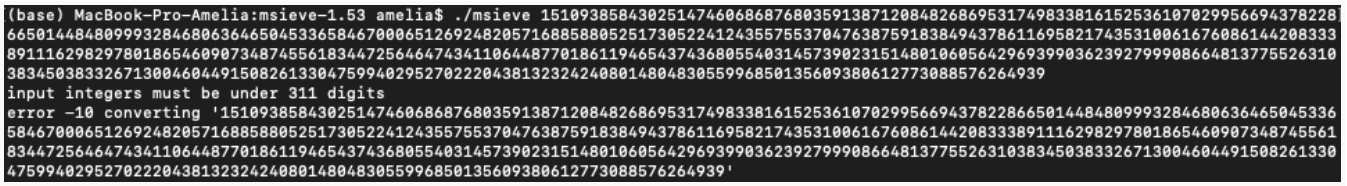
Описание

Факторизацией натурального числа называется его разложение в произведение простых множителей. Существование и единственность такого разложения следует из основной теоремы арифметики.

Существует ρ-алгоритм Полларда, но его сложность $O(n^{1/4})$. Число n1 имеет в своей записи 78 цифр, число n2 содержит 463 цифры. Таким образом, число n1 приблизительно равно 10^{77}. Подставляя его в формулу сложности найденного алгоритма, получаем 10^{19}. Язык C++ выполняет приблизительно 10^9 операций в секунду, и C++ считается одним из самых быстрых языков программирования. Следовательно для факторизации числа n1 данным алгоритмом потребуется приблизительно 10^{10} секунд, а это 2,7 миллиона часов. Остальные похожие алгоритмы имеют приблизительно такую же сложность, на которую не хватит физически времени и машинных мощностей.

На помощь здесь приходит библиотека msieve, которую можно скачать в интернете. Это библиотека на языке Си, реализующая набор алгоритмов для факторизации больших целых чисел и включает в себя реализации общего метода решета числового поля и квадратичного решета.
Используем её через вызов в командной строке. Библиотека msieve смогла факторизовать первое число за **2 минуты 49 секунд,** это отображено в файле msieve.log.

Второе же число имеет 463 цифры. Факторизовать такое большое число за короткое время невозможно. Например, даже библиотека msieve принимает на вход числа максимум из 311 цифр

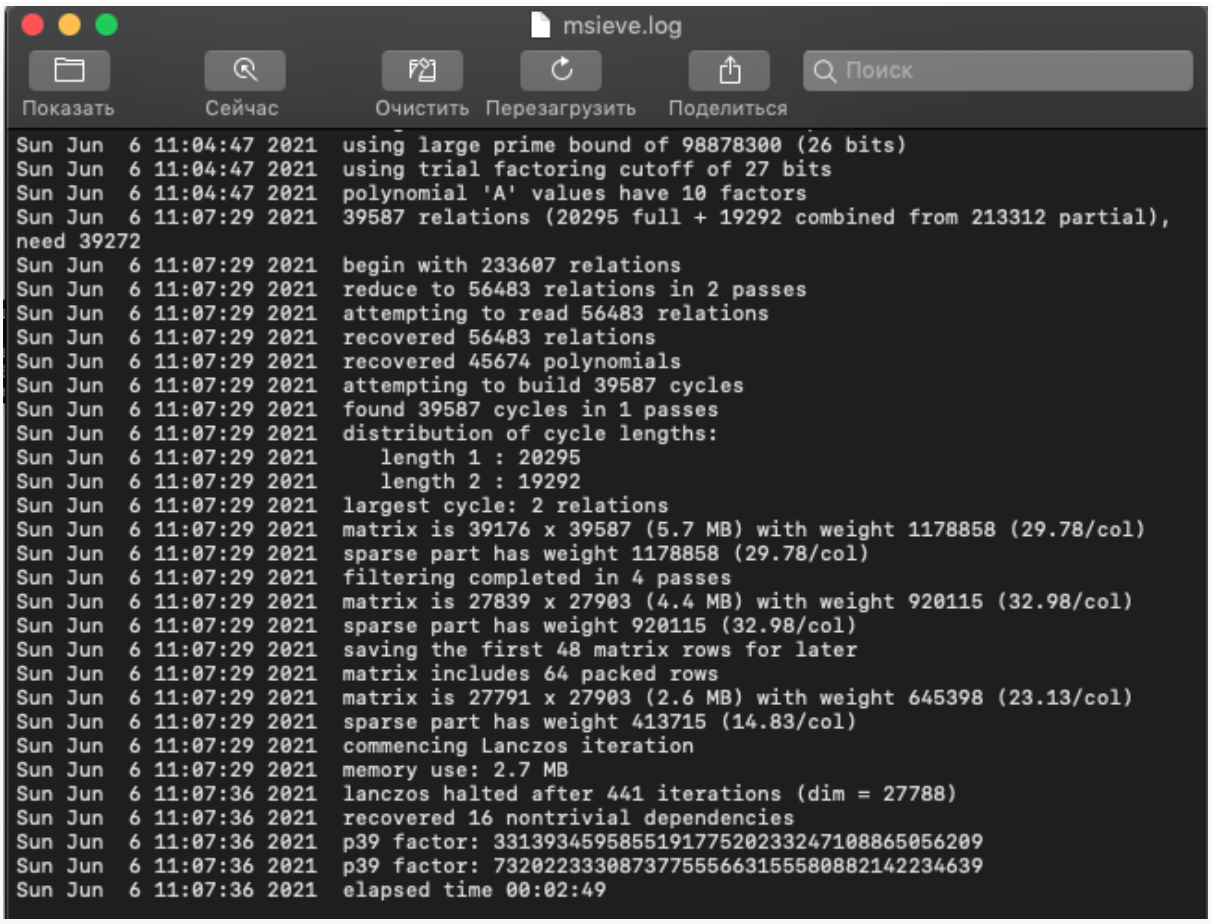


Оказалось, что один из множителей числа n2 можно найти как НОД с n2 из 15 варианта. Второй множитель легко найти, разделив данное число на полученный НОД.
Для выполнения этой части задания напишу небольшой скрипт на Python, использующий библиотеку gcd для поиска НОД.

Ход выполнения работы

1) Установить библиотеку msieve.
   Для этого, скачав её архив, разархивирую. Затем добавлю в Makefile по рекомендации add 'NO_ZLIB=1' if you don't have zlib. Далее пропишу в терминале make all.
2) Запуск программы, передавая ей число n1 в качестве аргумента

3) Результат отображается в файле msieve.log



Найдено два нетривиальных сомножителя для n1:

- 331393459585519177520233247108865056209
- 732022333087377555663155580882142234639

4) Напишу скрипт для работы со вторым числом

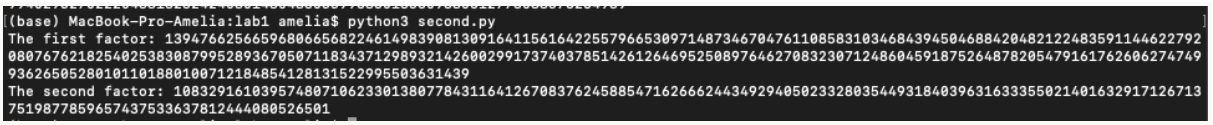
```
from math import gcd

n2 =
    151093858430251474606868768035913871208482686953174983381615253610702995669437822866501448480999
    328468063646504533658467000651269248205716885880525173052241243557553704763875918384943786116958
    217435310061676086144208333891116298297801865460907348745561834472564647434110644877018611946543
    743680554031457390231514801060564296939903623927999086648137755263103834503833267130046044915082
    613304759940295270222043813232424080148048305599685013560938061277308857626493 9
n3 =
    141690844477193411432723606433569517503385556872451472327609090923890224945076116311617929837009
    763773660959874697853968119080617502373943782494979020311419544728762119216205286391137003028125
    331158247702385902798481867910823926760076341189111357818193897834136876367785553468541342743729
    023927657307836543731689119550558446364266971611293672837308855338590286435921893375062744052147
    047677417879341309775432810687681000908343262821328867219442075462092054885112 9

b = gcd(n2, n3)
a = n2//b

print('The first factor:', b)
print('The second factor:', a)
```

5) Результат его работы



Получаем нетривиальные сомножители для числа n2:

- 139476625665968066568224614983908130916411561642255796653097148734670476110858310346843945046884204821224835911446227920807676218254025383087995289367050711834371298932142600299173740378514261264695250897646270832307124860459187526487820547916176260627474993626505280101101880100712184854128131522995503631439

- 108329161039574807106233013807784311641267083762458854716266624434929405023328035449318403963163335502140163291712671375198778596574375336378124440805265 01

Затраченное время на факторизацию первого числа: 2 минуты 49 секунд
Затраченное время на факторизацию второго числа: меньше 1 секунды

Для второго числа мы используем только поиск НОД и деление одного числа на другое. Сложность алгоритма Евклида для поиска НОД это $O(\log\min(a,b))$, получаем, что для поиска НОД чисел из 463 цифр потребуется только 1536 операций, что по времени почти мгновенно.

Характеристики компьютера, на котором производился расчёт:
Процессор: 2,3 GHz Intel Core i7
Память: 8 ГБ 1600 MHz DDR3
Операционная система: macOS Mojave 10.14.6

Вывод

Факторизация чисел - первое применение квантового компьютера, которое вспоминают. Длительное время, необходимое для вычисления факторизации больших целых чисел - основа существования RSA алгоритма.
Система RSA используется для защиты программного обеспечения и в схемах цифровой подписи.
Также она используется в открытой системе шифрования PGP и иных системах шифрования (к примеру, DarkCryptTC и формат xdc) в сочетании с симметричными алгоритмами.

Данная лабораторная работа стала поводом для небольшого исследования в данном вопросе, и вывод положительный: на данный момент использование RSA безопасно. Но как только квантовый компьютер будет доработан, нужно будет пересматривать текущую систему и искать новые способы шифрования.