Лабораторная работа № 1 по курсу: криптография

Выполнил студент группы М8О-308Б-17 МАИ *Милько Павел.*

Задача

Разложить каждое из чисел n1 и n2 на нетривиальные сомножители.
Вариант 12.
n_1 :
313230894596513941163065516500542159481861849753982064716706926040955753912601
n_2 :
693532252693819955253847249043757529180324494073823494547591843582136805148466227647256127350410778190683007539432807696698056465273137755794986981751541604145575912570331718107924355720576988204493545005230906033105693432028755478062000534784352638264990268794303011750110045088373921373348481833734839258579645479000424548585936336555233860123720335766312774151565346595485784249448792904040188292802408882242972670662804668689397345518513975596694683183631239 3

Решение

Для поиска простых делителей я пытался использовать алгоритм факторизации Ферма. Метод основан на поиске чисел x и y, которые удовлетворяют соотношению $x^2 - y^2 = n$, что приводится к разложению $(x - y)(x + y) = n$. Так же уравнение равносильно следующему: $x^2 - n = y^2$, то есть $x^2 - n$ – квадрат. Для начала алгоритма выбирается наименьшее x, такое что $x^2 > n$. Для каждого значения последующего значения вычисляют $(x + k)^2 - n$ и проверяют, не является ли это число точным квадратом. Если оно является точным квадратом, то получено разложение: $n = x^2 - y^2 = (x - y)(x + y)$. Так же можно проверить число на простоту: если один из сомножителей единица, то исходное число – простое. Метод работает быстро, если n явлеется произведением двух близких к друг другу сомножителей.

Программа обрабатывала первое число более 12 часов, но так и не завершила вычисления. Поэтому я воспользовался готовой реализацией метода решета числового поля – msieve.

msieve.log

```
Tue Feb 25 01:27:23 2020
Tue Feb 25 01:27:23 2020
Tue Feb 25 01:27:23 2020  Msieve v. 1.53 (SVN unknown)
Tue Feb 25 01:27:23 2020  random seeds: 31d94c7b 069b18a6
```

```
Tue Feb 25 01:27:23 2020  factoring 2041 (4 digits)
Tue Feb 25 01:27:23 2020  p2 factor: 13
Tue Feb 25 01:27:23 2020  p3 factor: 157
Tue Feb 25 01:27:23 2020  elapsed time 00:00:00
Tue Feb 25 01:27:30 2020
Tue Feb 25 01:27:30 2020
Tue Feb 25 01:27:30 2020  Msieve v. 1.53 (SVN unknown)
Tue Feb 25 01:27:30 2020  random seeds: d45d5401 c30b9837
Tue Feb 25 01:27:30 2020  factoring
    313230894596513941163065516500542159481861849753982064716706926040955753912601 (78
    digits)
Tue Feb 25 01:27:30 2020  no P-1/P+1/ECM available, skipping
Tue Feb 25 01:27:30 2020  commencing quadratic sieve (78-digit input)
Tue Feb 25 01:27:30 2020  using multiplier of 1
Tue Feb 25 01:27:30 2020  using generic 32kb sieve core
Tue Feb 25 01:27:30 2020  sieve interval: 12 blocks of size 32768
Tue Feb 25 01:27:30 2020  processing polynomials in batches of 17
Tue Feb 25 01:27:30 2020  using a sieve bound of 999269 (39162 primes)
Tue Feb 25 01:27:30 2020  using large prime bound of 99926900 (26 bits)
Tue Feb 25 01:27:30 2020  using trial factoring cutoff of 27 bits
Tue Feb 25 01:27:30 2020  polynomial 'A' values have 10 factors
Tue Feb 25 01:29:17 2020  39510 relations (20731 full + 18779 combined from 211459 partial),
     need 39258
Tue Feb 25 01:29:17 2020  begin with 232190 relations
Tue Feb 25 01:29:17 2020  reduce to 55927 relations in 2 passes
Tue Feb 25 01:29:17 2020  attempting to read 55927 relations
Tue Feb 25 01:29:17 2020  recovered 55927 relations
Tue Feb 25 01:29:17 2020  recovered 42152 polynomials
Tue Feb 25 01:29:17 2020  attempting to build 39510 cycles
Tue Feb 25 01:29:17 2020  found 39510 cycles in 1 passes
Tue Feb 25 01:29:17 2020  distribution of cycle lengths:
Tue Feb 25 01:29:17 2020     length 1 : 20731
Tue Feb 25 01:29:17 2020     length 2 : 18779
Tue Feb 25 01:29:17 2020  largest cycle: 2 relations
Tue Feb 25 01:29:17 2020  matrix is 39162 x 39510 (5.7 MB) with weight 1174885 (29.74/col)
Tue Feb 25 01:29:17 2020  sparse part has weight 1174885 (29.74/col)
Tue Feb 25 01:29:17 2020  filtering completed in 3 passes
Tue Feb 25 01:29:17 2020  matrix is 26985 x 27049 (4.3 MB) with weight 901416 (33.33/col)
Tue Feb 25 01:29:17 2020  sparse part has weight 901416 (33.33/col)
Tue Feb 25 01:29:17 2020  saving the first 48 matrix rows for later
Tue Feb 25 01:29:17 2020  matrix includes 64 packed rows
Tue Feb 25 01:29:17 2020  matrix is 26937 x 27049 (2.8 MB) with weight 662930 (24.51/col)
Tue Feb 25 01:29:17 2020  sparse part has weight 476210 (17.61/col)
Tue Feb 25 01:29:17 2020  commencing Lanczos iteration
Tue Feb 25 01:29:17 2020  memory use: 2.9 MB
Tue Feb 25 01:29:23 2020  lanczos halted after 428 iterations (dim = 26933)
Tue Feb 25 01:29:23 2020  recovered 13 nontrivial dependencies
Tue Feb 25 01:29:23 2020  p39 factor: 537228079155448813380781027030896715807
Tue Feb 25 01:29:23 2020  p39 factor: 583050117352260532679885280778162124743
Tue Feb 25 01:29:23 2020  elapsed time 00:01:53
```

Получанные числа действительно являются сомножителями n1 и при перемножении получается исходное число.

Со вторым числом такой фокус не прошёл, потому что *msieve* н обрабатывает числа длиннее 311 символов (Второе число состоит из 464 символов)

По совету старшекурсников, сомножители второго числа я искал как НОД с чис-

лом другого варианта. Тут мой код заработал как надо.

find.py

```
#!/usr/bin/env python

import sys

if len(sys.argv) != 2:
    print("expected filename")
    sys.exit(1)

FILE = open(sys.argv[1], "r")


def read_one(file=FILE):
    try:
        n1 = int(file.readline()[3:])
        n2 = int(file.readline()[3:])
        return (n1, n2)
    except:
        return (0, 0)


def NOD(a, b):
    while a != 0 and b != 0:
        if a > b:
            a %= b
        else:
            b %= a

    return a+b


def main():
    nums = []
    num_variant = 12

    while True:
        n1, n2 = read_one()
        if n1 == 0:
            break

        nums.append((n1, n2))

    most = nums[num_variant]

    for idx, num in enumerate(nums):
```

```
        n1 = NOD(most[1], num[0])
        n2 = NOD(most[1], num[1])
        for id_res, num_nod in enumerate((n1, n2)):
            if num_nod != 1:
                print(
                    f"variant={idx}\nNOD:\n{num_nod}\norig:\n{most
                        [1]}\nsecond:\n{num[id_res]}\n")
                return


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

test.res.format

```
variant=8
NOD:
214992603100076870820660222843230578058396066737712299070761057956973043100812446817837921430169694167529918778664297620499905048114418802939952640576941130
orig:
693532252693819955253847249043757529180324494073823494547591843582136805148466227647256127350410778190683007539432807696698056465273137755794986981751541604145575912570331718107924355720576988204493545005230906033105693432028755478062000534784352638264990268794303011750110045088373921373348481833734839258579645479000424548585936336555233860123720335766312774151565346595485784249448792904040188292802408882242972670662804668689397345518513975596694683183631239308
second:
420485970579145597958630635373365683138770428963133761653041725352287186214112063193243399009198794940697115014334627159749150070564594917809326213430731801530785419469293337080958121008766495807246629051017536955212049514499180340005448669590071737087784561583796585755051767068376460240691698197991002936873606774141343074347557185123133244906507991775680591075097199618814327703549705991448048413575595104347540154458778200555527823373791889640981452517485410 7
```