**ЛАБОРАТОРНА РОБОТА № 6**

Метою даної роботи є написання програми, яка обчислює дискретний логарифм за модулем простого числа .

Нехай є деякий елемент з та припустимо, що вам дано у таке, що , де . Ваше завдання знайти . Тобто, на вхід програми подається та на виході отримаємо .

Використані позначення: , , – найбільший спільний дільник (greater common divisor). Якщо - просте число, то .  
Тривіальний алгоритми для розв’язку цієї задачі це перебирати усі 240 варіантів значень , поки не буде знайдено правильне значення, тобто таке, що у . Такий підхід вимагає 240 піднесень до степеня по модулю. Але в даному завданні ви повинні реалізувати алгоритм, який використовує приблизно піднесень, реалізовуючи при цьому атаку «зустріч посередині» (a meet in the middle attack).   
  
Нехай . Оскільки менше, ніж можна записати невідомий як , де беруться з проміжку . Тоді у .

Перетворюючи даний вираз отримаємо

у .

Змінні в цьому рівнянні , усе інше – відомі величини: дано та . Оскільки в даному рівнянні змінні стоять з різних сторін рівняння, то можна знайти рішення методом «зустрічі посередині»:

1. Спочатку будуємо хеш таблицю усіх можливих значень лівої частини рівняння для усіх значень .

2. Тоді для усіх значень перевіряємо чи значення правої частини наявне у цій хеш-таблиці. Якщо так, то знайдено рішення , з якого легко обчислити значення .

Загальний обсяг обчислень: 220 піднесень до степеня для побудови хеш таблиці і, в найгіршому випадку, 220 пошуків у таблиці.

Дані для обчислень:

   p=134078079299425970995740249982058461274793658205923933 \

77723561443721764030073546976801874298166903427690031 \

858186486050853753882811946569946433649006084171

g=11717829880366207009516117596335367088558084999998952205 \

59997945906392949973658374667057217647146031292859482967 \

5428279466566527115212748467589894601965568

h=323947510405045044356526437872806578864909752095244 \

952783479245297198197614329255807385693795855318053 \

2878928001494706097394108577585732452307673444020333  
  
Кожне з цих чисел має приблизно 153 цифри. Знайдіть таке, що у .

Щоб зробити це завдання використайте середовище, яке підтримує модулярну арифметику та арифметику довільної точності. У Python можна використати модулі [gmpy2](http://readthedocs.org/docs/gmpy2/en/latest/mpz.html#mpz-methods) або [numbthy](http://www.math.umbc.edu/%7Ecampbell/Computers/Python/numbthy.py). В С можна використати [GMP](http://gmplib.org/). У Java використовуйте клас BigInteger.

**ВИХІДНИЙ КОД РЕАЛІЗАЦІЇ МОВОЮ JAVASCRIPT**

**var bigInt = require("big-integer");**

**const extendedEuklid = (fi, e) => {**

**console.log('START COMPUTE EXTENDED EUKLIDIAN');**

**const table = [[fi, e], [bigInt(-1), bigInt(-1)], [bigInt(1), bigInt(0)], [bigInt(0), bigInt(1)]];**

**let r = bigInt(0);**

**let i = bigInt(2);**

**do {**

**r = table[0][i - 2].mod(table[0][i - 1]);**

**const q = table[0][i - 2].divide(table[0][i - 1]);**

**const l = table[2][i - 2] - (q \* table[2][i - 1])**

**const b = table[3][i - 2] - (q \* table[3][i - 1])**

**table[0][i] = r;**

**table[1][i] = q;**

**table[2][i] = l;**

**table[3][i] = b;**

**i++;**

**} while (!r.equals(1));**

**const d = table[3][--i];**

**return bigInt(d).greater(0) ? d : d + fi;**

**};**

**const computeLeft = (x, B, p, e) => {**

**console.log('START FOUND AN ANSWER')**

**for (let i = 1; i < x; i++) {**

**const c = bigInt(e).pow(i);**

**const l = bigInt(B).multiply(c);**

**const r = l.mod(p);**

**if (right[r]) {**

**console.log('ANSWER R = ', r);**

**console.log('ANSWER X = ', i);**

**return;**

**}**

**}**

**};**

**const right = {};**

**const computeRight = (x, B, p, g) => {**

**console.log('START COMPUTE RIGHT PART')**

**for (let i = 1; i < x; i++) {**

**const c = bigInt(g).pow(B).pow(i);**

**const r = c.mod(p);**

**right[r] = i;**

**}**

**};**

**const computeB = (p) => {**

**console.log('START COMPUTE B');**

**let i = 0;**

**while (bigInt(2).pow(i).lesser(p)) {**

**i++;**

**}**

**const bresponse = bigInt(2).pow(bigInt(i).divide(2));**

**console.log('FOUND B ', bresponse);**

**return bresponse;**

**};**

**const g = bigInt('7');**

**const p = bigInt('683');**

**const h = bigInt('510');**

**const B = computeB(p);**

**const e = extendedEuklid(p, g);**

**//console.log(extendedEuklid(p, g))**

**//const left = bigInt(7).modPow(B, p);**

**// computeLeft(32, B, p, e);**

**computeRight(B, B, p, g);**

**computeLeft(B, h, p, e);**

**ВИХІДНИЙ КОД РЕАЛІЗАЦІЇ МОВОЮ PYTHON**

def tmpFun (a, c, p):

r = 1

while 0<c:

r = (r\*a)%p

c-=1

return r

def funLen(p):

i=0

while p > (2\*\*i):

i=i+1

return i

def fun1(a, l, p):

if l.get(a,-1) != -1:

#for i in l:

#if i == a:

return (a, l.get(a))

return (-1, -1)

def gcdex(a, b):

if b == 0:

return a, 1, 0

else:

d, x, y = gcdex(b, a % b)

return d, y, x - y \* (a // b)

p=13407807929942597099574024998205846127479365820592393377723561443721764030073546976801874298166903427690031858186486050853753882811946569946433649006084171

g=11717829880366207009516117596335367088558084999998952205599979459063929499736583746670572176471460312928594829675428279466566527115212748467589894601965568

h=3239475104050450443565264378728065788649097520952449527834792452971981976143292558073856937958553180532878928001494706097394108577585732452307673444020333

a,b,d = gcdex(g,p)

c = funLen(p)

coun = int(c/2)

l = {}

l2= []

#c2=2\*\*coun

c2=2\*\*20

g1=tmpFun(g,c2,p)

t=1

for c in range(c2):

l.update({(t\*h)%p:c})

#print((t\*h)%p,)

t=(t\*b)%p

#l.append(((h\*((b\*\*c)%p))%p))

t=1

for x0 in range(c2):

#print (x0)

x,x1 = fun1(t, l, p)

#x,x1 = fun1(((g1\*\*x0)%p), l, p)

if -1 != x:

break

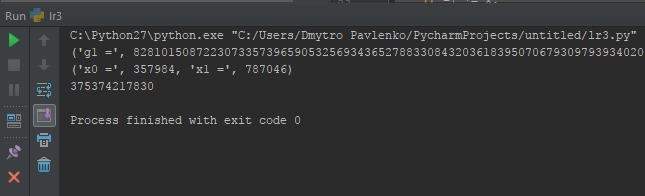
t=(t\*g1)%p

print ("g1 =",g1)

print ("x0 =",x0, "x1 =",x1)

print ((c2\*x0+x1)%p)

**РЕЗУЛЬТАТ ВИКОНАННЯ**



Х = 375374217830