Московский авиационный институт (Национальный исследовательский университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Криптография».

Студент: Алексюнина Ю.В.
Преподаватель: А. В. Борисов
Группа: М8О-307Б
Дата:
Оценка:
Подпись:

Москва, 2020

Вариант №1

Задача:

Разложить каждое из чисел представленных ниже на нетривиальные сомножители.

Первое число:

n1=352358118079150493187099355141629527101749106167997255509619020528333722352217,

Второе число:

n2=48738223550664856484010719919241368186758723982865354540647758882243540576477598278886729048817580219521127213727925407544900302456083961829239053550140760905023863976659838390092229761027195955475247825918278599180637013523064720193934409719551643543860327258436073468433555466992185964231467043829462118734032419673256260249740890745904785802741810153616125736248464670798413622398624863284189543014224877216614751857244348601566526424673501072070269730653155203

Выходные данные:

Для каждого числа необходимо найти и вывести все его множители - простые числа.

Описание

Процесс разложения числа на его простые множители называется факторизацией. Для решения этой задачи существует множество алгоритмов, позволяющих находить множители, используя свойства простых чисел.

Для данной решения задачи я изначально выбрала алгоритм Полларда, как один из наиболее простых и эффективных. Однако эффективен он, как я узнала позднее, для чисел меньшего порядка, чем в моем задании, поэтому я решила обратиться к другим алгоритмам.

Так как преподаватель не ограничил нас в выборе средств для решения задачи, я решила прибегнуть к готовому решению, реализованному профессионалами - программному продукту msieve, который реализует в себе целый ряд алгоритмов факторизации с общим названием «Общий метод решета числового поля». Этот метод считается одним из самых эффективных современных алгоритмов факторизации. Он справился с поставленной задачей для 1-го числа менее чем за 1 минуту, что является впечатляющим результатом.

Однако 2 число имеет более 400 квадратичных знаков, факторизация которого на обычном компьютере за разумное время практически невозможна ни одним из ныне существующих алгоритмов. Но я узнала, что один из множителей этого числа определяется как наибольший общий делитель с одним из чисел другого варианта. Поэтому я написала программу, перебирающую все числа других вариантов и определяющую их НОД с числом моего варианта и выводящую его, если он > 1. Второе же число определяется как результат деления числа моего варианта на НОД (по свойству делителя). Для работы с большими числами и отыскания делителя в этой программе я использовала класс BigInteger.

Реализация(на языке C#)

```
using System;
using System.Numerics;
using System.IO;
using System.Linq;

namespace factor
{
    class Program
    {
        static void Main(string[] args)
        {
            string num =
"4873822355066485648401071991924136818675872398286535454064775888224354
0576477598278886729048817580219521127213727925407544900302456083961829 2
3905355014076090502386397665983839009222976102719595547524782591827859 9
1806370135230647201939344097195516435438603272584360734684335554669921 8
5964231467043829462118734032419673256260249740890745904785802741810153 6
1612573624846467079841362239862486328418954301422487721661475185724434 8
6015665264246735010720702697306531520 3";
            BigInteger GCD = 1, r, res = 1;
            BigInteger.TryParse(num, out r);
            BigInteger[] nums = new BigInteger[40];
            string[] lines =
File.ReadAllLines("GCD.txt").Take(40).ToArray();
            for (int i = 0; i < 40; i++)
            {
                BigInteger.TryParse(lines[i], out nums[i]);
                GCD = BigInteger.GreatestCommonDivisor(r, nums[i]);
                if (GCD > 1)
                {
                    res = BigInteger.Divide(r, GCD);
                    break;
                }
            }

            using (StreamWriter outputFile = new
StreamWriter("res.txt"))
            {
                outputFile.WriteLine(GCD);
                outputFile.WriteLine(res);
            }
        }
    }
}
```

Ответ

Разложение первого числа:

562068224324090847465842314308226273321

626895637985717823820028254946860326577

Разложение второго числа:

338321131328863932564818680296074563081532769076856609725860402543148614331548252714137193761514829808776013335385989807884956317300479543405309911233314952878929489767681129617464167512042524219275287655118056649888395988512738187341200216598691518616681800731333973241210979683235278330374664946737167178077

144059058206710204218866501230431552923357707155732802627279896991140559512353033870669074747540407853308809672851000995334154053145871766927574703208126 39

Выводы

Благодаря первой лабораторной работе по данному курсу, я напрямую познакомилась с новой для себя темой - факторизацией больших чисел.

Эта работа сама по себе не является очень трудной для выполнения, однако я столкнулась со сложностью в выборе нужного метода для поиска множителей. Ведь, например, метод простого перебора будет выполнять данную работу на моем компьютере дольше сотни лет(и не факт, что выполнит), а метод решета также требует больше ресурсов, чем я могу предоставить, для этой задачи. Это стало мне уроком и показало насколько важно выбрать оптимальный алгоритм для решения задачи.

Я рада, что мне пришлось позаниматься этой задачей в курсе криптографии, и надеюсь, что в дальнейшем меня будут ждать такие же интересные задачи.

Список литературы

Метод квадратичного решета
URL: https://ru.wikipedia.org/wiki/Метод_квадратичного_решета (дата обращения: 03.03.2020).

Библиотека Msieve
URL: https://github.com/radii/msieve (дата обращения: 03.03.2020).

BigInteger
URL: https://docs.microsoft.com/ru-ru/dotnet/api/system.numerics.biginteger?view=netframework-4.8 (дата обращения: 08.03.2020).

Алгоритм Полларда
URL: https://ru.wikipedia.org/wiki/Ро-алгоритм_Полларда (дата обращения: 03.03.2020).

Общий метод решета числового поля
URL: https://ru.wikipedia.org/wiki/Общий_метод_решета_числового_поля (дата обращения: 03.03.2020