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=352358118079150493187099355141629527101749106167997255509619020

528333722352217,

Второе число:

n2=487382235506648564840107199192413681867587239828653545406477588

822435405764775982788867290488175802195211272137279254075449003024

560839618292390535501407609050238639766598383900922297610271959554

752478259182785991806370135230647201939344097195516435438603272584

360734684335554669921859642314670438294621187340324196732562602497

408907459047858027418101536161257362484646707984136223986248632841

895430142248772166147518572443486015665264246735010720702697306531

5203

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[,](file:///C:\\Users\\прямо%20за%20мечтой\\Downloads\\report.doc" \l "page10) 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[,](file:///C:\\Users\\прямо%20за%20мечтой\\Downloads\\report.doc" \l "page10) который реализует в себе целый ряд алгоритмов факторизации с общим названием «Общий метод решета числового поля»[.](file:///C:\Users\прямо%20за%20мечтой\Downloads\report.doc#page10) 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 С#)

using System;

using System.Numerics;

using System.IO;

using System.Linq;

namespace factor

{

class Program

{

static void Main(string[] args)

{

string num = "4873822355066485648401071991924136818675872398286535454064775888224354057647759827888672904881758021952112721372792540754490030245608396182923905355014076090502386397665983839009222976102719595547524782591827859918063701352306472019393440971955164354386032725843607346843355546699218596423146704382946211873403241967325626024974089074590478580274181015361612573624846467079841362239862486328418954301422487721661475185724434860156652642467350107207026973065315203";

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

14405905820671020421886650123043155292335770715573280262727989699114055951235303387066907474754040785330880967285100099533415405314587176692757470320812639

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.

* 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