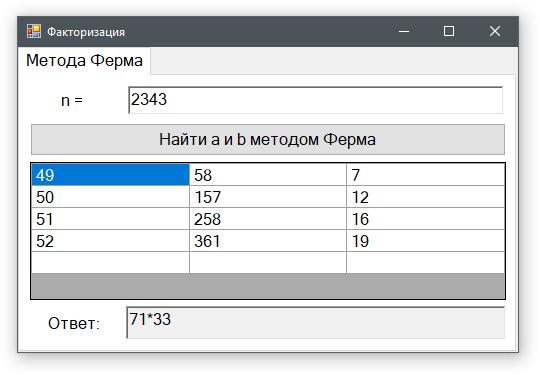
Лабораторная работа № 4. Метод Ферма и Метод Диксона

Десятов Александр, 09-641



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Numerics;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Метода\_Ферма\_и\_Диксона

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

if(bW\_Ferma.IsBusy)

{

bW\_Ferma.WorkerSupportsCancellation = true;

bW\_Ferma.CancelAsync();

}

if(bW\_Dikson.IsBusy)

{

bW\_Dikson.WorkerSupportsCancellation = true;

bW\_Dikson.CancelAsync();

}

}

// Ферма

DataTable dataTable\_Ferma;

private void btn\_Ferma\_Click(object sender, EventArgs e)

{

rTB\_answer\_Ferma.Text = "";

btn\_Ferma.Enabled = false;

dGV\_Ferma.Columns.Clear();

bW\_Ferma.RunWorkerAsync();

}

private void bW\_Ferma\_DoWork(object sender, DoWorkEventArgs e)

{

// Готовим таблицу

if (dataTable\_Ferma != null) dataTable\_Ferma.Clear();

dataTable\_Ferma = new DataTable();

// Заголовки

dataTable\_Ferma.Columns.Add("x", typeof(BigInteger));

dataTable\_Ferma.Columns.Add("z", typeof(BigInteger));

dataTable\_Ferma.Columns.Add("y", typeof(BigInteger));

// Считывание n

if (!BigInteger.TryParse(rTB\_n\_Ferma.Text, out BigInteger n))

{

MessageBox.Show("Некорректное число n");

return;

}

if (n < 2)

{

MessageBox.Show("n должно быть больше 1");

return;

}

if (n > 1 && n < 4)

{

rTB\_answer\_Ferma.Text = "Простое";

return;

}

if(n.IsEven)

{

rTB\_answer\_Ferma.Text = "" + (n/2) + "\*2";

return;

}

// На данном этапе n - нечетное натуральное и больше 3

BigInteger x = Sqrt(n);

if (x\*x == n)

{

rTB\_answer\_Ferma.Text = "" + x + "\*" + x;

return;

}

BigInteger z = x \* x - n;

do

{

x++;

if (x == (n + 1) / 2)

{

rTB\_answer\_Ferma.Text = "Простое";

return;

}

else

{

z += 2 \* x - 1;

BigInteger y = Sqrt(z);

try { dataTable\_Ferma.Rows.Add(x, z, y); } catch (Exception) {; }

if (y \* y == z)

{

rTB\_answer\_Ferma.Text = "" + (x + y).ToString() + "\*" + (x - y).ToString();

return;

}

}

} while (true);

}

private void bW\_Ferma\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

btn\_Ferma.Enabled = true;

try

{

if (dataTable\_Ferma != null)

{

dGV\_Ferma.ColumnCount = dataTable\_Ferma.Columns.Count;

dGV\_Ferma.RowCount = dataTable\_Ferma.Rows.Count + 1;

}

}

catch(Exception)

{

MessageBox.Show("В таблице не хватило строк");

}

}

private void dGV\_Ferma\_CellValueNeeded(object sender, DataGridViewCellValueEventArgs e)

{

if (e.RowIndex < 0 || e.RowIndex >= dataTable\_Ferma.Rows.Count)

return;

e.Value = dataTable\_Ferma.Rows[e.RowIndex][e.ColumnIndex];

}

// Sqrt для BigInteger есть в java, но нет .Net

private BigInteger Sqrt(BigInteger n)

{

if (n == 0) return 0;

if (n > 0)

{

int bitLength = Convert.ToInt32(Math.Ceiling(BigInteger.Log(n, 2)));

BigInteger root = BigInteger.One << (bitLength >> 1);

while (!IsSqrt(n, root))

{

root += n / root;

root >>= 1;

}

return root;

}

throw new ArithmeticException("NaN");

}

private bool IsSqrt(BigInteger n, BigInteger root)

{

BigInteger lowerBound = root \* root;

return n >= lowerBound && n <= lowerBound + (root << 1);

}

}

}

**Метод Диксона**

import numpy as np

from tmp import findsol

import itertools

import math

def gcd(a, b):

while b:

a, b = b, a % b

return abs(a)

def factorization(n\_in\_f):

f = open('primes1.txt', 'r')

all\_prime\_str = f.read()

f.close()

all\_prime\_arr = [int(i) for i in all\_prime\_str.split()]

p\_wiki = math.sqrt(math.exp(math.sqrt(math.log(n\_in\_f) \* math.log(math.log(n\_in\_f)))))

S = [i for i in all\_prime\_arr if i < p\_wiki]

k = len(S)

t = k + 1

while True:

print("t =", t)

V = []

A = []

B = []

while len(V) < t:

sqrt\_n = math.sqrt(n\_in\_f)

a = np.ceil(sqrt\_n + np.random.random() \* (n\_in\_f - sqrt\_n) - 1)

A.append(a)

b = (a\*a) % n\_in\_f

if b == 0:

continue

B.append(b)

print("a =", a, "b =", b, "len(V) =", len(V), "t =", t)

v\_row = np.zeros(k)

for i\_s, s in enumerate(S):

while b % s == 0:

b /= s

v\_row[i\_s] = not v\_row[i\_s]

# Если удалось b\_f разложить на множители из S, то есть b\_f является p(k) гладкой

if b == 1:

V.append(v\_row)

V = np.array(V)

zero\_arr\_k = np.array(np.zeros(k))

# cV = 0

# Все бинарные числа в двоичном списке

С = np.array(list(itertools.product([0, 1], repeat=t)))

for c in С:

# Если c является решением уравнения cV = 0

print("c =", c)

if np.array\_equal(c@V % 2, zero\_arr\_k):

x = b = 1

for i\_val in enumerate(c):

if i\_val[1] == 1:

b \*= B[i\_val[0]]

x = (x\*A[i\_val[0]]) % n\_in\_f

y = int(np.sqrt(b))

print("x =", x, "y =", y)

if (not x == y % n\_in\_f) and (not x == (n\_in\_f - y) % n\_in\_f):

answer = gcd(x + y, n\_in\_f)

if not answer == 1:

return answer

t += 1

# n = float(input("n = "))

n = 1022117

print(factorization(n))