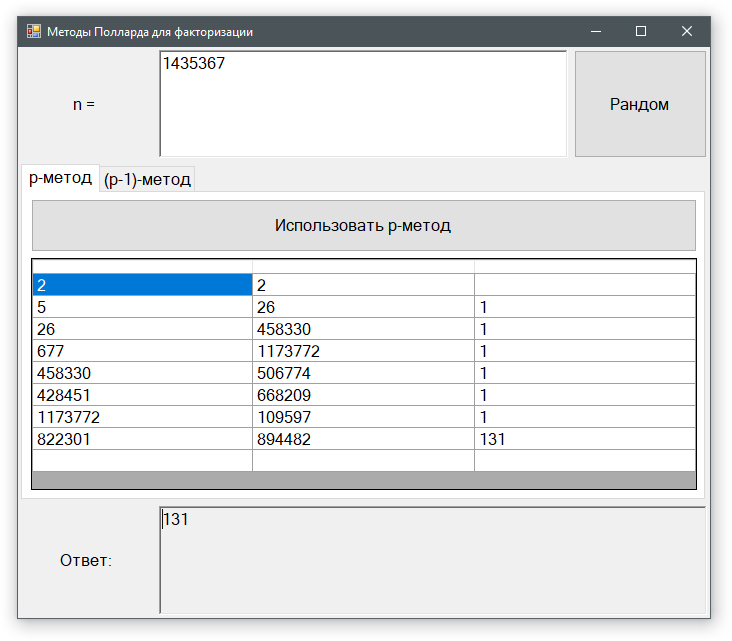
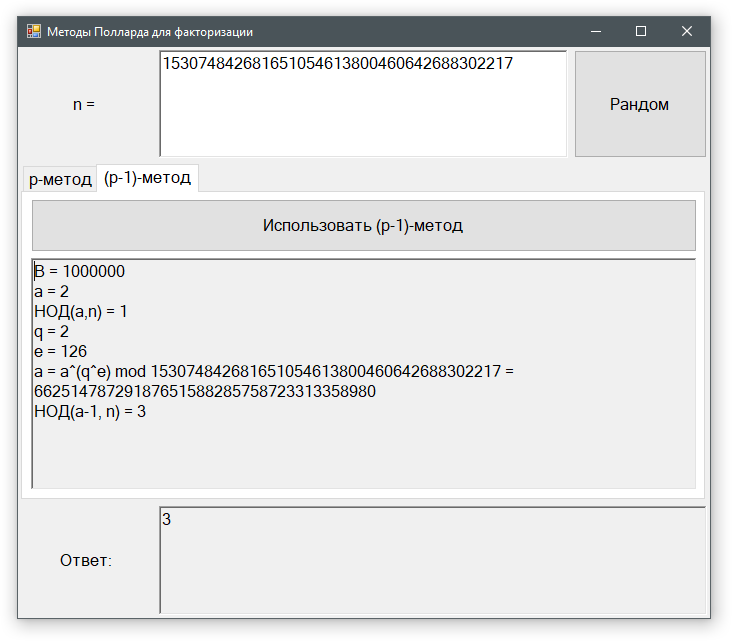
Лабораторная работа № 3. Методы Полларда для факторизации.

Десятов Александр, 09-641



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Numerics;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ро\_метод\_Полларда\_факторизация

{

public partial class FormPollardFact : Form

{

DataTable dataTable;

public FormPollardFact()

{

InitializeComponent();

}

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

{

Random random = new Random();

int length = 16;// Длина в байтах рандомного числа

byte[] data = new byte[length];

random.NextBytes(data);

rTB\_n.Text = BigInteger.Abs((new BigInteger(data))).ToString();

}

private void FormPollardFact\_Load(object sender, EventArgs e)

{

rnd = new Random((int)DateTime.Now.Ticks);

}

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

{

if (bW\_ro.IsBusy)

{

bW\_ro.WorkerSupportsCancellation = true;

bW\_ro.CancelAsync();

}

if (bW\_p\_1.IsBusy)

{

bW\_p\_1.WorkerSupportsCancellation = true;

bW\_p\_1.CancelAsync();

}

}

// Ро-метод

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

{

rTB\_answer.Text = "";

btn\_ro.Enabled = false;

dGV\_ro.Columns.Clear();

bW\_ro.RunWorkerAsync();

}

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

{

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

{

MessageBox.Show("Некорректное число n");

return;

}

if (n <= 0) return;

if (n > 0 && n < 4)

{

rTB\_answer.Text = n.ToString();

return;

}

// Готовим таблицу

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

dataTable = new DataTable();

// Заголовки

dataTable.Columns.Add("a", typeof(BigInteger));

dataTable.Columns.Add("b", typeof(BigInteger));

dataTable.Columns.Add("d", typeof(BigInteger));

BigInteger a = 2, b = 2, d;

BigInteger a\_temp = 1, b\_temp = 1;

dataTable.Rows.Add(a, b);

BigInteger attempts\_a\_b = 5; // Столько попыток будет, чтобы найти разные 'a' и 'b'

BigInteger attempts\_in\_main\_cicle = 100; // Попыток в главном цикле (без этого, если простое, то не выходит)

BigInteger i = 1;

do

{

BigInteger cons = 0; // константа внутри функции

BigInteger a\_prev = a, b\_prev = b;

// Поиск разных 'a' и 'b'

do

{

cons++;

a = Func\_Ro(a, n, cons);

b = Func\_Ro(Func\_Ro(b, n, cons), n, cons);

} while (a == b && cons < attempts\_a\_b);

// Если не удалось найти разные 'a' и 'b'

if(a == b)

{

dataTable.Rows.Add(a, b);

MessageBox.Show("Отказ");

return;

}

// НОД(|a-b|, n)

d = BigInteger.GreatestCommonDivisor(BigInteger.Abs(a - b), n);

// Нет смысла рассматривать a и b, которые равны предыдущим a и b, поэтому проводим новую инициализацию

if (a == a\_prev && b == b\_prev)

{

a = a\_temp;

// Если все возможные 'a' перебраны

if (a\_temp++ > n)

{

a\_temp = 1;

b = b\_temp;

// Если все возможные 'b' перебраны

if (b\_temp++ > n)

{

MessageBox.Show("Все 'a' перебраны. Множители числа " + n + " не найдены");

return;

}

}

continue;

}

// Запись в таблицу

dataTable.Rows.Add(a, b, d);

// Проверка на количество попыток

/\*if (i++ > attempts\_in\_main\_cicle)

{

MessageBox.Show("За " + attempts\_in\_main\_cicle + " попыток не удалось найти множитель числа " + n);

return;

}\*/

} while (d == 1);

rTB\_answer.Text = d.ToString();

}

private BigInteger Func\_Ro(BigInteger x, BigInteger n, BigInteger cons)

{

return (x\*x + cons) % n;

}

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

{

if(dataTable != null)

{

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

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

}

btn\_ro.Enabled = true;

btn\_p\_1.Enabled = true;

}

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

{

if (e.RowIndex < 0 || e.RowIndex >= dataTable.Rows.Count) return;

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

}

// (p-1)-метод

string text\_to\_rTB = "";

private void btn\_p\_1\_Click(object sender, EventArgs e)

{

rTB\_p\_1.Text = "";

rTB\_answer.Text = "";

btn\_p\_1.Enabled = false;

bW\_p\_1.RunWorkerAsync();

}

private void bW\_p\_1\_DoWork(object sender, DoWorkEventArgs e)

{

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

{

MessageBox.Show("Некорректное число n");

return;

}

if (n <= 0) return;

if (n > 0 && n < 4)

{

rTB\_answer.Text = n.ToString();

return;

}

// Этот текст будет в rTB

text\_to\_rTB = "";

BigInteger B = 1000000; text\_to\_rTB += "B = " + B + "\n";

string[] prime\_arr\_str;

// чтение из файла

using (StreamReader sr = new StreamReader(@"primes1.txt"))

{

prime\_arr\_str = sr.ReadToEnd().Split();

}

List <BigInteger> prime\_arr = new List<BigInteger>();

foreach(var prime in prime\_arr\_str)

{

prime\_arr.Add(BigInteger.Parse(prime));

}

// Случайное 'a'. Начнем с '2', потом можно задавать случайно или все перебрать

BigInteger a\_tmp = 2, a = 2; text\_to\_rTB += "a = " + a + "\n";

// НОД(a, n)

BigInteger d = BigInteger.GreatestCommonDivisor(a, n); text\_to\_rTB += "НОД(a,n) = " + d + "\n";

if (d > 1)

{

rTB\_answer.Text = d.ToString();

return;

}

BigInteger min\_B\_n = BigInteger.Min(B, n) + 1;

for (int i = 0; i < prime\_arr.Count && prime\_arr[i] < min\_B\_n; i++)

{

text\_to\_rTB += "q = " + prime\_arr[i] + "\n";

BigInteger \_e = new BigInteger(BigInteger.Log(n) / BigInteger.Log(prime\_arr[i])); text\_to\_rTB += "e = " + \_e + "\n";

a = BigInteger.ModPow(a, BigInteger.ModPow(prime\_arr[i], \_e, n), n); text\_to\_rTB += "a = a^(q^e) mod " + n + " = " + a + "\n";

if (a == 1)

{

/\*a = a\_tmp++; text\_to\_rTB += "a = " + a + "\n";

if (a\_tmp > n - 2)

break;\*/

a = GetRandomBigInteger(3,n-2); text\_to\_rTB += "a = " + a + "\n"; // Рандомно

i = -1; // (потом будет i++)

continue;

}

d = BigInteger.GreatestCommonDivisor(a - 1, n); text\_to\_rTB += "НОД(a-1, n) = " + d + "\n";

if (d == 1)

{

B++; text\_to\_rTB += "B = " + B + "\n";

}

else

{

rTB\_answer.Text = d.ToString();

return;

}

}

MessageBox.Show("Множители числа " + n + " не найдены");

}

private void bW\_p\_1\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

rTB\_p\_1.Text = text\_to\_rTB;

btn\_p\_1.Enabled = true;

}

#region Тесты на простоту

private int Test\_Millera\_Rabina(BigInteger n)

{

if (n < 2)

return -1; // не простое и не составное

else

{

int s = 0;

BigInteger d = n - 1;

while (d.IsEven)

{

d /= 2;

s++;

}

BigInteger[] prime\_numbers\_50 = {

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,

73, 79, 83, 89, 97, 101, /\*103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173,

179, 181, 191, 193, 197, 199, 211, 223, 227, 229\*/};

// Сколько раз проверять?

BigInteger r = 50; // log2(n)

// Если число n-2 меньше, чем r, то нужно проверять не r раз, а (n-2) раз

foreach (BigInteger a in prime\_numbers\_50)

//r = n - 2 < r ? n - 2 : r;

//for (BigInteger a = 2; a <= r + 1; a++) // a должно по простым ходить...

{

if (a > n - 1)

break;

BigInteger x\_current = BigInteger.ModPow(a, d, n);

if (x\_current == 1 || x\_current == n - 1)

continue; // a - свидетель простоты

else

{

bool svidetel\_neprostoti = true;

for (int i = 1; i < s; i++)

{

x\_current = BigInteger.ModPow(x\_current, 2, n);

if (x\_current == n - 1)

{

svidetel\_neprostoti = false;

break; // a - свидетель простоты

}

}

if (svidetel\_neprostoti)

return 0; // составное

}

}

return 1; // вероятно простое

}

}

private int Test\_Soloveya\_Shtrassena(BigInteger n)

{

if (n < 2)

return -1; // не простое и не составное

else if (n == 2)

return 1; // простое

else if (n.IsEven)

return 0; // составное

int k = 10;

while (k-- > 0)

{

// Случайным образом выбирается число a<n

BigInteger a = GetRandomBigInteger(2, n);

if (BigInteger.GreatestCommonDivisor(a, n) > 1)

{

return 0; // Составное

}

else

{

BigInteger res1 = BigInteger.ModPow(a, (n - 1) / 2, n);

BigInteger res2 = Yakobi\_symbol(a, n);

while (res2 < 0)

res2 += n;

if (res1 != res2)

return 0; // Составное

}

}

return 1; // Вероятно простое

}

private int Yakobi\_symbol(BigInteger a, BigInteger b)

{

if (BigInteger.GreatestCommonDivisor(a, b) != 1)

return 0;

int r = 1;

if (a < 0)

{

a = -a;

if (b % 4 == 3)

r = -r;

}

while (a != 0)

{

BigInteger t = 0;

while (a.IsEven)

{

t++;

a >>= 1; //деление на 2

}

if (!t.IsEven)

{

BigInteger.DivRem(b, 8, out BigInteger ostatok\_b\_na\_8);

if (ostatok\_b\_na\_8 == 3 || ostatok\_b\_na\_8 == 5)

r = -r;

}

BigInteger.DivRem(a, 4, out BigInteger ostatok\_a\_na\_4);

BigInteger.DivRem(b, 4, out BigInteger ostatok\_b\_na\_4);

if (ostatok\_a\_na\_4 == ostatok\_b\_na\_4 && ostatok\_a\_na\_4 == 3)

r = -r;

BigInteger c = a;

BigInteger.DivRem(b, c, out a);

b = c;

}

return r;

}

// Получить рандомное BigInteger в диапазоне

Random rnd;

private BigInteger GetRandomBigInteger(BigInteger min, BigInteger max)

{

if (min > max)

{

var buff = min;

min = max;

max = buff;

}

max -= min;

int deg = (int)BigInteger.Log(max, 2);

// Рандомная длина

deg = rnd.Next(deg + 1);

BigInteger tmp = 1;

while (deg-- > 0)

{

tmp <<= 1; //умножаем на 2

tmp += rnd.Next(2);

}

// Если вдруг больше max получилось

BigInteger.DivRem(tmp, max, out BigInteger result);

return result + min;

}

#endregion

}

}