НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №3.1
“Реалізація задачі розкладання числа на прості множники (факторизація числа)”

Виконав:
студент групи ІП-84
ІП-8408
Засько Євгеній

Київ 2021

Теоретичні відомості

Факторизації лежить в основі стійкості деяких криптоалгоритмів, еліптичних кривих, алгебраїчній теорії чисел та кванових обчислень, саме тому дана задача дуже гостро досліджується, й шукаються шляхи її оптимізації. На вхід задачі подається число n Є ℕ, яке необхідно факторизувати. Перед виконанням алгоритму слід переконатись в тому, що число не просте. Далі алгоритм шукає перший простий дільник, після чого можна запустити алгоритм заново, для повторної факторизації. В залежності від складності алгоритми факторизації можна розбити на дві групи: Експоненціальні алгоритми (складність залежить експоненційно від довжини вхідного параметру); Субекспоненціальні алгоритми. Існування алгоритму з поліноміальною складністю – одна з найважливіших проблем в сучасній теорії чисел. Проте, факторизація з даною складністю можлива на квантовому комп'ютері за допомогою алгоритма Шора.

Метод факторизації Ферма.

Ідея алгоритму заключається в пошуку таких чисел A і B, щоб факторизоване число n мало вигляд: $n = A^2 – B^2$. Даний метод гарний тим, що реалізується без використання операцій ділення, а лише з операціями додавання й віднімання.

Приклад алгоритму:

Початкова установка: x = [√*n*] – найменше число, при якому різниця x^2 - n невід'ємна. Для кожного значення k Є ℕ, починаючи з k = 1, обчислюємо $(\lceil\sqrt{n}\rceil + k)^2 - n$ і перевіряємо чи не є це число точним квадратом. Якщо не є, то k++ і переходимо на наступну ітерацію. Якщо є точним квадратом, тобто $x^2 - n = (\lceil\sqrt{n}\rceil + k)^2 - n = y^2$, то ми отримуємо розкладання: $n = x^2 - y^2 = (x + y)(x - y) = A * B$, в яких $x = (\lceil\sqrt{n}\rceil + k)$ Якщо воно є тривіальним і єдиним, то n - просте

Модофікований метод факторизації Ферма.

Ідея алгоритму заключається в пошуку таких чисел A і B, щоб факторизоване число n мало вигляд: $n = A^2 – B^2$. Даний метод гарний тим, що реалізується без використання операцій ділення, а лише з операціями додавання й віднімання.

Приклад алгоритму:

1. Початкова установка: x = 2[√*n*] + 1 , y = 1, r = [√*n*] 2 – n.

2. Якщо r = 0, то алгоритм закінчено: $n = (x-y)/2 * (x+y-2)/ 2$

3. Присвоюємо r = r + x, x = x + 2.

4. Присвоюємо r = r - y, y = y +

5. Якщо r > 0, повертаємось до кроку 4, інакше повертаємось до кроку 2.

Умови завдання для варіанту бригади

Розробити програма для факторизації заданого числа методом Ферма. Реалізувати користувацький інтерфейс з можливістю вводу даних.

Лістинг програми із заданими умовами завдання

```
package ua.kpi.comsys.lab3_1

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.core.content.ContextCompat
import androidx.core.widget.addTextChangedListener
import kotlinx.coroutines.*
import ua.kpi.comsys.lab3_1.databinding.ActivityMainBinding
import java.lang.Exception
import kotlin.math.ceil
import kotlin.math.pow
import kotlin.math.sqrt

class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
    }

    override fun onResume() {
        super.onResume()

        binding.edittext.addTextChangedListener {
```

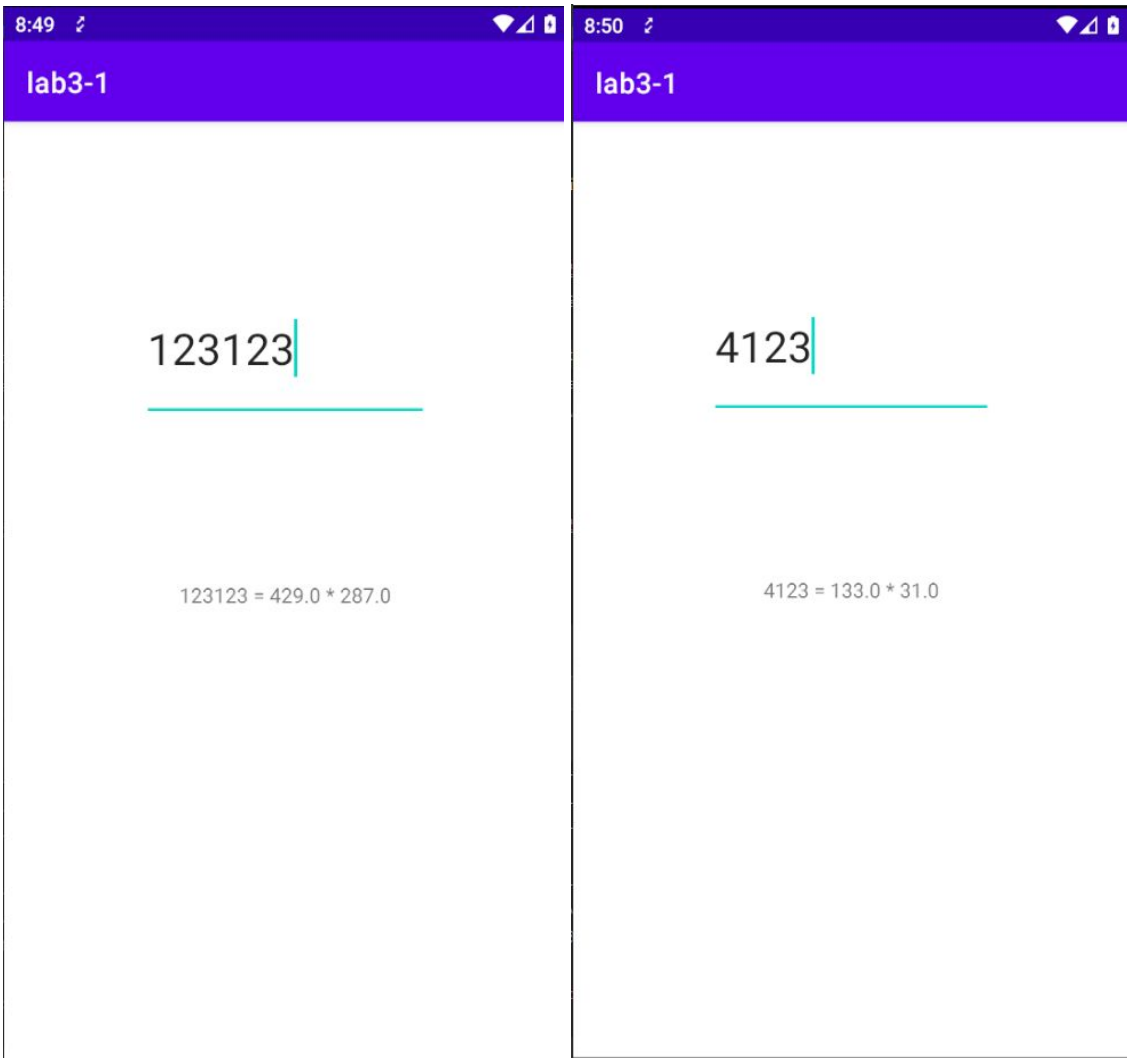
```
            try {
                binding.textView.text = "Calculating..."
                GlobalScope.launch {
                    val num = it.toString().toLong()
                    val (a, b) = factorization(num)
                    withContext(Dispatchers.Main) {
                        binding.textView.text = "$num = $a * $b"
                    }
                }
            } catch(e: Exception) {
                binding.textView.text = "Invalid arguments"
            }
        }
    }

    private fun factorization(n: Long): Pair<Double, Double>{
        val k = ceil(sqrt(n.toDouble()))

        for (a in k.toInt()..n) {
            val b = a.toDouble().pow(2) - n
            val q = sqrt(b)
            if (q.isWhole()) return a + q to a - q
        }

        return 1.toDouble() to n.toDouble()
    }

    private fun Double.isWhole() = this.toLong() - this == 0.0
}
```

Результати виконання кожної програми

```
package ua.kpi.comsys.lab3_1

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.core.content.ContextCompat
import androidx.core.widget.addTextChangedListener
import kotlinx.coroutines.*
import ua.kpi.comsys.lab3_1.databinding.ActivityMainBinding
import java.lang.Exception
import kotlin.math.ceil
import kotlin.math.pow
import kotlin.math.sqrt

class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
    }

    override fun onResume() {
        super.onResume()

        binding.edittext.addTextChangedListener {
            if (!it.isNullOrBlank()) {
                try {
                    binding.textView.text = "Calculating..."
                    val num = it.toString().toLong()
```

```
                GlobalScope.launch {
                    val (a, b) = factorization(num)
                    withContext(Dispatchers.Main) {
                        binding.textView.text = "$num = $a * $b"
                    }
                }
            } catch(e: Exception) {
                binding.textView.text = "Invalid arguments"
            }
        }
    }

    private fun factorization(n: Long): Pair<Double, Double>{
        val k = ceil(sqrt(n.toDouble()))

        for (a in k.toInt()..n) {
            val b = a.toDouble().pow(2) - n
            val q = sqrt(b)
            if (q.isWhole()) return a + q to a - q
        }

        return 1.toDouble() to n.toDouble()
    }

    private fun Double.isWhole() = this.toLong() - this == 0.0
}
```

Результати виконання кожної програми



Висновки щодо виконання лабораторної роботи

В ході виконання лабораторної роботи ознайомився з принципом роботи алгоритмів, що розкладають числа на прості множники, а саме метод перебору, факторизації Ферма та модифікований факторизації Ферма. Розроблено застосунок, що приймає число та повертає його в вигляді двох множників.