Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №3

з дисципліни
«Інтелектуальні вбудовані системи»

на теми
«РЕАЛІЗАЦІЯ ЗАДАЧІ РОЗКЛАДАННЯ ЧИСЛА НА ПРОСТІ МНОЖНИКИ (ФАКТОРИЗАЦІЯ ЧИСЛА)
ДОСЛІДЖЕННЯ НЕЙРОННИХ МЕРЕЖ. МОДЕЛЬ PERCEPTRON.
ДОСЛІДЖЕННЯ ГЕНЕТИЧНОГО АЛГОРИТМУ»

Виконав:

студент групи ІП-84
Сімонов Павло
номер залікової книжки: 8421

Перевірив:

викладач
Регіда Павло Геннадійович

Київ 2021

Основні теоретичні відомості

3.1

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.

Розглянемо принципи роботи найпростіших алгоритмів факторизації.

Метод перебору можливих дільників.

Один з найпростіших і найочевидніших алгоритмів заключається в тому, щоб послідовно ділити задане число n на натуральні числа від 1 до $\lfloor\sqrt{n}\rfloor$. Формально, достатньо ділити лише на прості числа в цьому інтервалі, але для цього необхідно знати їх множину. На практиці складається таблиця простих чисел і на вхід подаються невеликі числа (до 2^{16}), оскільки даний алгоритм має низьку швидкість роботи.

Приклад алгоритму:

1. Початкова установка: t = 0, k = 0, n = N (t,k,n такі, що n = N / $p_1 \ldots p_n$ і n не мають простих множників, менших за d_k).
2. Якщо n = 1, закінчуємо алгоритм.
3. Присвоюємо q = [n / d_k], r = n mod d_k.
4. Якщо r ≠ 0, переходимо на крок 6.
5. Присвоюємо t++, $p_t = d_k$, n = q і повертаємось на крок 2.
6. Якщо q > d_k → k++ і повертаємось на крок 3.
7. Присвоїти t++, p_t = n і закінчити виконання алгоритму.

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: $x = 2[\sqrt{n}] + 1$, y = 1, $r = [\sqrt{n}]^2 - n$.
2. Якщо r = 0, то алгоритм закінчено: $n = \frac{x-y}{2} * \frac{x+y-2}{2}$
3. Присвоюємо r = r + x, x = x + 2.
4. Присвоюємо r = r - y, y = y + 2.
5. Якщо r > 0, повертаємось до кроку 4, інакше повертаємось до кроку 2.

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: $x = [\sqrt{n}]$ – 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 $x^2 - n = \left(\lceil\sqrt{n}\rceil + k\right)^2 - n = y^2$, 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

3.2

Важливою задачеюяку система реального часу має вирішувати є отриманнянеобхідних для обчислень параметрів, її обробка та виведення результату у встановлений дедлайн. З цього постає проблема отримання водночас точних та швидких результатів. Модель Перцпептрон дозволяє покроково наближати початкові значення.

Розглянемо приклад: дано дві точки A(1,5), B(2,4), поріг спрацювання P = 4, швидкість навчання δ = 0.1. Початкові значення ваги візьмемо нульовими W1 = 0, W2 = 0. Розрахунок вихідного сигналу у виконується за наступною формулою:

x1 * W1 + x2 * W2 = y

Для кожного кроку потрібно застосувати дельта-правило, формула для розрахунку похибки:

Δ = P – y

де y – значення на виході.

Для розрахунку ваги, використовується наступна формули:

W1(i+1) = W1(i) + W2 * x11

W2(i+1) = W1(i) + W2 * x12

де i – крок, або ітерація алгоритму.

Розпочнемо обробку:

1 ітерація:

Використовуємо формулу обрахунку вихідного сигналу:

0 = 0 * 1 + 0 * 5 значення не підходить, оскільки воно менше зазначеного порогу. Вихідний сигнал повинен бути строго більша за поріг.

Далі, рахуємо Δ:

Δ = 4 – 0 = 4

За допомогою швидкості навчання δ та минулих значень ваги, розрахуємо нові значення ваги:

W1 = 0 + 4 * 1 * 0,1 = 0,4

W2 = 0 + 4 * 5 * 0.1 = 2

Таким чином ми отримали нові значення ваги. Можна побачити, що результат

змінюється при зміні порогу.

2 ітерація:

Виконуємо ті самі операції, але з новими значеннями ваги та для іншої точки.

8,8 = 0,4 * 2 + 2 * 4 , не підходить, значення повинно бути менше порогу.

Δ = -5 , спрощуємо результат для прикладу.

W1 = 0,4 + 5 * 2 * 0,1 = -0,6

W2 = 2 - 5 * 4 * 0.1 = 0

3 ітерація:

Дано тільки дві точки, тому повертаємось до першої точки та нові значення ваги

розраховуємо для неї.

-0,6 = -0,6 * 1 + 0 * 5 , не підходить, значення повинно бути більше порогу.

Δ = 5 , спрощуємо результат для прикладу.

W1 = -0,6 + 5 * 1 * 0,1 = -0,1

W2 = 0 + 5 * 5 * 0.1 = 2,5

По такому самому принципу рахуємо значення ваги для наступних ітерацій, поки не

отримаємо значення, які задовольняють вхідним даним.

На восьмій ітерації отримуємо значення ваги W1 = -1,8 та W2 = 1,5.

5,7 = -1,8 * 1 + 1,5 * 5, більше за поріг, задовольняє

2,4 = -1,8 * 2 + 1,5 * 4, менше за поріг, задовольняє

Отже, бачимо, що для заданого прикладу, отримано значення ваги за 8 ітерацій.

При розрахунку значень, потрібно враховувати дедлайн. Дедлайн може бути в вигляді

максимальної кількості ітерацій або часовий.

3.3

Генетичні алгоритми служать, головним чином, для пошуку рішень в багатовимірних просторах пошуку.

Можна виділити наступні етапи генетичного алгоритму:

(Початок циклу)

Розмноження (схрещування)

Мутація

Обчислити значення цільової функції для всіх особин

Формування нового покоління (селекція)

Якщо виконуються умови зупинки, то (кінець циклу), інакше (початок циклу).

Розглянемо приклад реалізації алгоритму для знаходження цілих коренів діофантового рівняння a+b+2c=15.

Згенеруємо початкову популяцію випадковим чином, але з дотриманням умови – усі згенеровані значення знаходяться у проміжку від

одиниці до y/2, тобто на відрізку [1;8] (узагалі, границі випадкового генерування можна вибирати на свій розсуд):

(1,1,5); (2,3,1); (3,4,1); (3,6,4)

Отриманий генотип оцінюється за допомогою функції пристосованості (fitness function). Згенеровані значення підставляються у рівняння, після чого обраховується різниця отриманої правої частини з початковим y. Після цього рахується ймовірність вибору генотипу для ставання батьком – зворотня дельта ділиться на сумму сумарних дельт усіх генотипів.

$1+1+2\cdot5=12 \quad \Delta=3 \quad \frac{\frac{1}{3}}{\frac{27}{24}} = 0{,}7$

$2+3+2\cdot1=7 \quad \Delta=8 \quad \frac{\frac{1}{8}}{\frac{27}{24}} = 0{,}11$

$3+4+2\cdot1=9 \quad \Delta=6 \quad \frac{\frac{1}{6}}{\frac{27}{24}} = 0{,}15$

$3+6+2\cdot4=17 \quad \Delta=2 \quad \frac{\frac{1}{2}}{\frac{27}{24}} = 0{,}44$

Наступний етап включає в себе схрещування генотипів по методу кросоверу – у якості дітей виступають генотипи, отримані змішуванням коренів – частина йде від одного з батьків, частина від іншого, наприклад:

(3 | 6,4)
(1 | 1,5)
→
(3,1,5)
(1,6,4)

Лінія кросоверу може бути поставлена в будь-якому місці, кількість потомків також може вибиратися. Після отримання нових генотипів вони перевіряються функцією пристосованості та створюють власних потомків, тобто виконуються дії, описані вище.

Ітерації алгоритму відбуваються, поки один з генотипів не отримає Δ=0, тобто його значення будуть розв’язками рівняння.

Завдання на лабораторну роботу

3.1

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

3.2

Поріг спрацювання: P = 4

Дано точки: A(0,6), B(1,5), C(3,3), D(2,4).

Швидкості навчання: δ = {0,001; 0,01; 0,05; 0.1; 0.2; 0,3}

Дедлайн: часовий = {0.5c; 1c; 2c; 5c}, кількість ітерацій = {100;200;500;1000}

Обрати швидкість навчання та дедлайн. Налаштувати Перцептрон для даних точок. Розробити відповідний мобільний додаток і вивести отримані значення.
Провести аналіз витрати часу та точності результату за різних параметрах навчання.
3.3
Налаштувати генетичний алгоритм для знаходження цілих коренів діофантового рівняння ax1+bx2+cx3+dx4=y. Розробити відповідний мобільний додаток вивести отримані значення. Провести аналіз витрат часу на розрахунки.

Лістинг програми

MainAktivity.kt

```
package ua.kpi.comsys.lab3system

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.EditText
import android.widget.Spinner
import android.widget.TextView
import kotlin.math.absoluteValue
import kotlin.math.pow

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

    }

    fun calculate(view: View){
        val editText = findViewById<EditText>(R.id.editTextTextPersonName)
        val textView = findViewById<TextView>(R.id.id_answer)
        val number = editText.text.toString().toInt()
        val ans = Fermat(number)
        textView.apply {
            text = ans
        }
```

```
}

fun calculate2(view: View){
    val spinner1 = findViewById<Spinner>(R.id.spinner1)
    val spinner2 = findViewById<Spinner>(R.id.spinner2)
    val spinner3 = findViewById<Spinner>(R.id.spinner3)
    val selected1 = spinner1.selectedItem.toString().toDouble()
    val selected2 = spinner2.selectedItem.toString().toDouble()
    val selected3 = spinner3.selectedItem.toString().toInt()
    val textView = findViewById<TextView>(R.id.id_answer2)

    val answer = Perceptron(selected1,selected2,selected3)

    textView.apply {
        text = answer
    }
}

fun calculate3(view: View){
    val editText1 = findViewById<EditText>(R.id.editTextNumber)
    val editText2 = findViewById<EditText>(R.id.editTextNumber2)
    val editText3 = findViewById<EditText>(R.id.editTextNumber3)
    val editText4 = findViewById<EditText>(R.id.editTextNumber4)
    val editText5 = findViewById<EditText>(R.id.editTextNumber5)
    val textView = findViewById<TextView>(R.id.id_answer3)
    val x1 = editText1.text.toString().toDouble()
    val x2 = editText2.text.toString().toDouble()
    val x3 = editText3.text.toString().toDouble()
    val x4 = editText4.text.toString().toDouble()
    val y = editText5.text.toString().toDouble()
    val ans = Evo(x1,x2,x3,x4,y)
    textView.apply {
        text = ans
    }
}

private fun Fermat(number : Int) : String
{
    //number = ((a + b) * (a - b))) = a^2 - b^2        30
```

```
        if (number % 2 == 0){
            return "Please enter odd number!"
        }
        val numberSqrt = kotlin.math.sqrt(number.toDouble()).toInt() + 1
//5,449287429383 + 1 = 6
        var k = 0
        var iter = 0
        var bPow = 0
        while (k == 0)
        {
            bPow = ((numberSqrt + iter).toDouble()).pow(2.00).toInt() - number  //6 ->
36  -> 6     //7 -> 49 -> 19
            if((kotlin.math.sqrt(bPow.toDouble()))% 1.0 == 0.0){
                k = iter
                break
            } else iter++
        }
        val a : Int = numberSqrt + k
        val b : Int = kotlin.math.sqrt(bPow.toDouble()).toInt()
        return mutableListOf(a, b).toString()
    }

    private fun Evo(x1_base: Double,
                    x2_base: Double,
                    x3_base: Double,
                    x4_base: Double,
                    y_base: Double) : String{

        val populationZero: MutableList<MutableList<Double>> = mutableListOf()
        val population: MutableList<MutableList<Double>> = mutableListOf()
        val listOfFitnesses: MutableList<Double> = mutableListOf()
        val populationOfChild: MutableList<MutableList<Double>> =
mutableListOf()
        var bestPopulation: MutableList<Double> = mutableListOf()

        fun fitness(population: MutableList<Double>): Double {
            val fitness: Double = y_base -
                    population[0] * x1_base -
                    population[1] * x2_base -
```

```
            population[2] * x3_base -
            population[3] * x4_base
    return fitness.absoluteValue
}

fun populationZeroFind() {
    for (i in 0..3) {
        populationZero.add(mutableListOf())
        for (j in 0..3) {
            populationZero[i].add((1..8).random().toDouble())
        }
    }
}

fun findFitnessOfPopulation() {
    listOfFitnesses.clear()
    if (population.isEmpty()) {
        populationZero.mapTo(population) { it }
    }
    for (i in 0..3) {
        listOfFitnesses.add(fitness(population[i]))
    }
}

fun findRoulette() {
    populationOfChild.clear()
    var roulette = 0.00
    val roulettePercent: MutableList<Double> = mutableListOf()
    val circleRoulette: MutableList<Double> = mutableListOf()
    listOfFitnesses.forEach { roulette += 1 / it }
    for (i in 0..3) {
        roulettePercent.add(1 / listOfFitnesses[i] / roulette)
    }

    for (i in 0..3) {
        if (i == 0) {
            circleRoulette.add(roulettePercent[i])
        } else {
            circleRoulette.add(circleRoulette[i - 1] + roulettePercent[i])
```

```
        }
    }

    var i = 0
    populationOfChild.clear()
    while (i < 4) {
        val piu: Double = (1..100).random().toDouble() / 100
        var thisChild = 0
        for (k in 0..3) {
            if (piu >= circleRoulette[k]) {
                thisChild = k
            }
        }
        populationOfChild.add(population[thisChild])
        i++
    }
}

fun crossingOver() {
    population.clear()
    for (p in 0..3) {
        val c: MutableList<Double> = mutableListOf()
        c.clear()
        for (j in 0..3) {
            if (p % 2 == 0) {
                if (j < 2) {
                    c.add(populationOfChild[p][j])
                } else c.add(populationOfChild[p + 1][j])
            } else
                if (j < 2) {
                    c.add(populationOfChild[p][j])
                } else c.add(populationOfChild[p - 1][j])
        }
        population.add(c)
    }
}

fun bestFitnessFind(): Boolean {
    findFitnessOfPopulation()
```

```
    listOfFitnesses.forEach { if (it == 0.0) return true }
    return false
}

fun life() {
    var q = 0
    while (!bestFitnessFind() && q < 10) {
        findFitnessOfPopulation()
        findRoulette()
        crossingOver()
        q++
    }
}

fun result(): MutableList<Double> {
    populationZeroFind()
    life()
    while ((!listOfFitnesses.contains(0.0)) &&
            population[0] == populationOfChild[0] &&
            population[1] == populationOfChild[1] &&
            population[2] == populationOfChild[2] &&
            population[3] == populationOfChild[3]
    ) {
        populationZero.clear()
        population.clear()
        listOfFitnesses.clear()
        populationOfChild.clear()
        populationZeroFind()
        life()
    }
    for (i in 0..3) {
        if (listOfFitnesses[i] == 0.0) {
            bestPopulation = population[i]
        }
    }
    return bestPopulation
}

val answer : MutableList<Double>  =  result();
```

```
        if (answer.isEmpty()) {
            return "No possible answer in range [1;y/2]";
        }
        return answer.toString();
    }

    private fun Perceptron (speed: Double, time: Double, iterations: Int): String{
        var W1 = 0.00
        var W2 = 0.00
        val P = 4.00
        val points = arrayListOf(Pair(0.00, 6.00), Pair(1.00, 5.00), Pair(3.00, 3.00),
Pair(2.00, 4.00))

        fun check(): Boolean {
            for (i in 0 .. 3){
                var y = W1 * points[i].first + W2 * points[i].second
                if ((i < 2 && y < P) || (i >= 2 && y > P) ) return false
            }
            return true
        }

        fun result(): Pair<Double, Double> {
            val startTime = System.currentTimeMillis()
            for (i in 0..iterations) {
                if ((System.currentTimeMillis() - startTime) <= time * 1000) {
                    for (k in 0 until points.size) {
                        val y = W1 * points[k].first + W2 * points[k].second
                        val delta = P - y
                        W1 += delta * points[k].first * speed
                        W2 += delta * points[k].second * speed
                        if (check()) {
                            return Pair(W1, W2)
                        }
                    }
                }
            }
            return Pair(W1, W2)
        }
```

```
        if (W1.isNaN() || W1 == Double.NEGATIVE_INFINITY || W1 ==
Double.POSITIVE_INFINITY || W2.isNaN() || W2 ==
Double.NEGATIVE_INFINITY || W2 == Double.POSITIVE_INFINITY){
            return ("W1 is $W1, W2 is $W2 => ERROR")
        }
        return result().toString();
    }
}
```

aktivity_main.kt

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/id_answer"
        android:layout_width="match_parent"
        android:layout_height="35dp"
        android:layout_marginTop="8dp"
        android:gravity="center"
        android:text="Your Answer for Lab 3.1!"
        android:textAlignment="center"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button1" />

    <TextView
        android:id="@+id/id_lab1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
```

```
    android:text="Lab 3.1 \nEnter number"
    android:textAlignment="center"
    android:gravity="center"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />


<TextView
    android:id="@+id/id_lab2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="36dp"
    android:gravity="center"
    android:text="Lab 3.2"
    android:textAlignment="center"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/id_answer" />

<Spinner
    android:id="@+id/spinner1"
    android:layout_width="115dp"
    android:layout_height="50dp"
    android:layout_marginTop="11dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/id_lab2"
    android:layout_marginLeft="20dp"
    android:entries = "@array/learningSpeed"/>

<Spinner
    android:id="@+id/spinner3"
    android:layout_width="115dp"
    android:layout_height="50dp"
    android:layout_marginTop="11dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/id_lab2"
    android:layout_marginRight="20dp"
```

```
    android:entries = "@array/maxIteration"/>

<Spinner
    android:id="@+id/spinner2"
    android:layout_width="115dp"
    android:layout_height="50dp"
    android:layout_marginTop="11dp"
    app:layout_constraintEnd_toStartOf="@+id/spinner3"
    app:layout_constraintStart_toEndOf="@+id/spinner1"
    app:layout_constraintTop_toBottomOf="@+id/id_lab2"
    android:entries = "@array/deadline"/>

<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:layout_weight="1"
    android:onClick="calculate2"
    android:text="LAB 3.2 CALCULATE"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/spinner2" />

<TextView
android:id="@+id/id_answer2"
android:layout_width="match_parent"
android:layout_height="35dp"
android:layout_marginTop="8dp"
android:gravity="center"
android:text="Your Answer for Lab 3.2!"
android:textAlignment="center"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.0"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/button2" />
```

```
<TextView
    android:id="@+id/id_lab3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="212dp"
    android:gravity="center"
    android:text="Lab 3.3"
    android:textAlignment="center"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/id_lab2" />

<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:layout_weight="1"
    android:onClick="calculate"
    android:text="LAB 3.1 CALCULATE"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editTextTextPersonName" />

<EditText
    android:id="@+id/editTextTextPersonName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:gravity="center"
    android:inputType="number"
    android:textAlignment="center"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/id_lab1" />

<EditText
    android:id="@+id/editTextNumber"
```

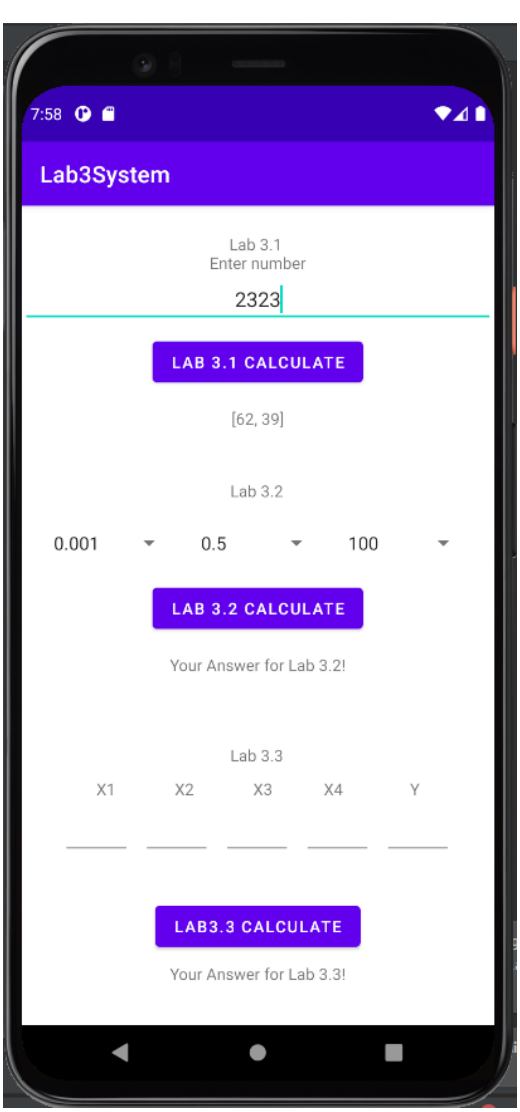
```
    android:layout_width="60dp"
    android:layout_height="40dp"
    android:layout_marginStart="35dp"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="40dp"
    android:ems="10"
    android:inputType="number"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/id_lab3" />

<EditText
    android:id="@+id/editTextNumber2"
    android:layout_width="60dp"
    android:layout_height="40dp"
    android:layout_marginStart="10dp"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="40dp"
    android:ems="10"
    android:inputType="number"
    app:layout_constraintStart_toEndOf="@+id/editTextNumber"
    app:layout_constraintTop_toBottomOf="@+id/id_lab3" />

<EditText
    android:id="@+id/editTextNumber3"
    android:layout_width="60dp"
    android:layout_height="40dp"
    android:layout_marginStart="10dp"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="40dp"
    android:ems="10"
    android:inputType="number"
    app:layout_constraintStart_toEndOf="@+id/editTextNumber2"
    app:layout_constraintTop_toBottomOf="@+id/id_lab3" />

<EditText
    android:id="@+id/editTextNumber4"
    android:layout_width="60dp"
    android:layout_height="40dp"
    android:layout_marginStart="10dp"
```

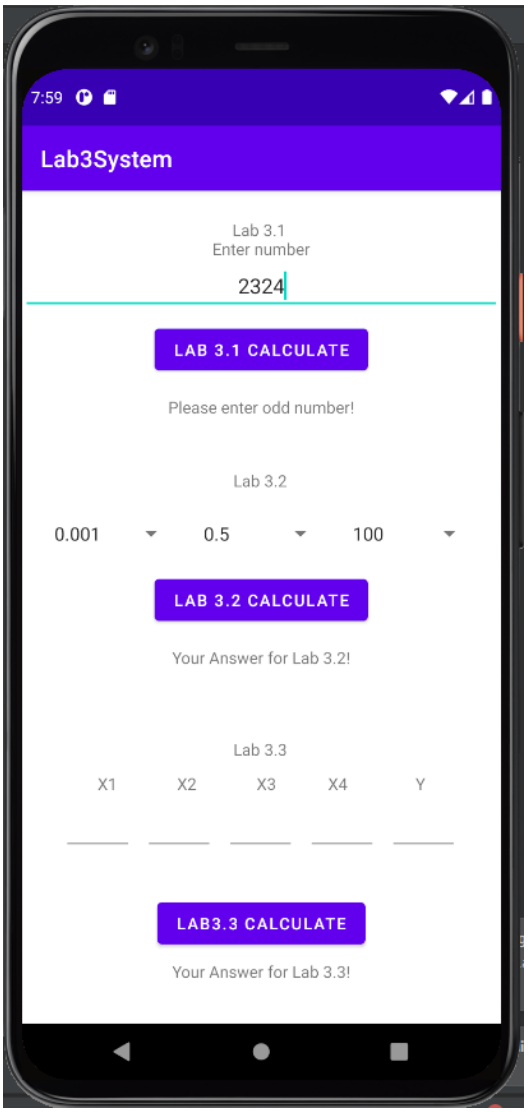
```
    android:layout_marginLeft="10dp"
    android:layout_marginTop="40dp"
    android:ems="10"
    android:inputType="number"
    app:layout_constraintStart_toEndOf="@+id/editTextNumber3"
    app:layout_constraintTop_toBottomOf="@+id/id_lab3" />

<EditText
    android:id="@+id/editTextNumber5"
    android:layout_width="60dp"
    android:layout_height="40dp"
    android:layout_marginStart="10dp"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="40dp"
    android:ems="10"
    android:inputType="number"
    app:layout_constraintStart_toEndOf="@+id/editTextNumber4"
    app:layout_constraintTop_toBottomOf="@+id/id_lab3" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="match_parent"
    android:layout_height="25dp"
    android:text="X1          X2          X3          X4          Y"
    android:textAlignment="center"
    android:gravity="center"
    app:layout_constraintBottom_toTopOf="@+id/editTextNumber3"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/id_lab3" />

<Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="36dp"
    android:onClick="calculate3"
    android:text="Lab3.3 Calculate"
    app:layout_constraintEnd_toEndOf="parent"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editTextNumber3" />

    <TextView
        android:id="@+id/id_answer3"
        android:layout_width="match_parent"
        android:layout_height="35dp"
        android:gravity="center"
        android:text="Your Answer for Lab 3.3!"
        android:textAlignment="center"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button3" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

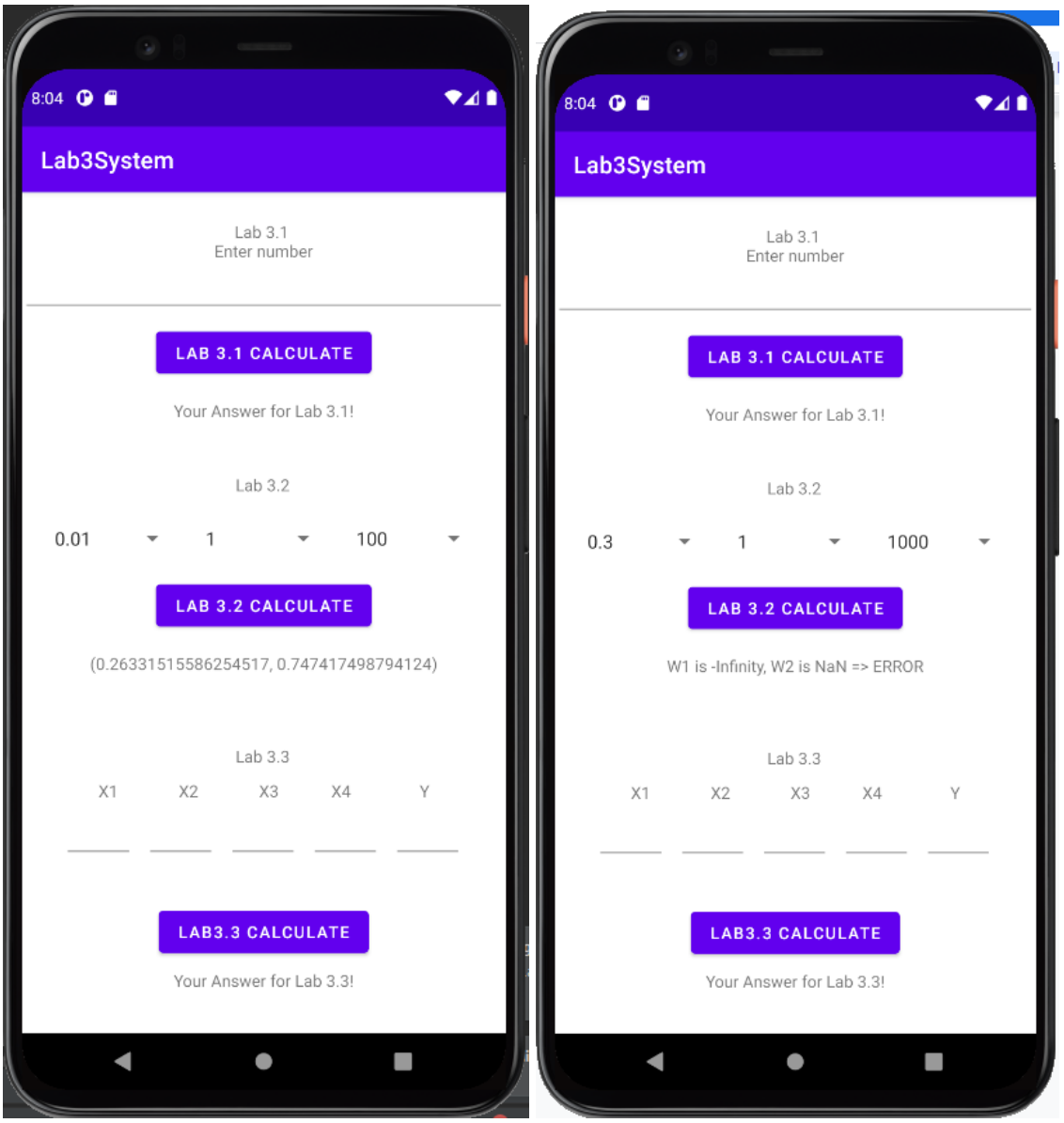
Результати роботи програми

3.1

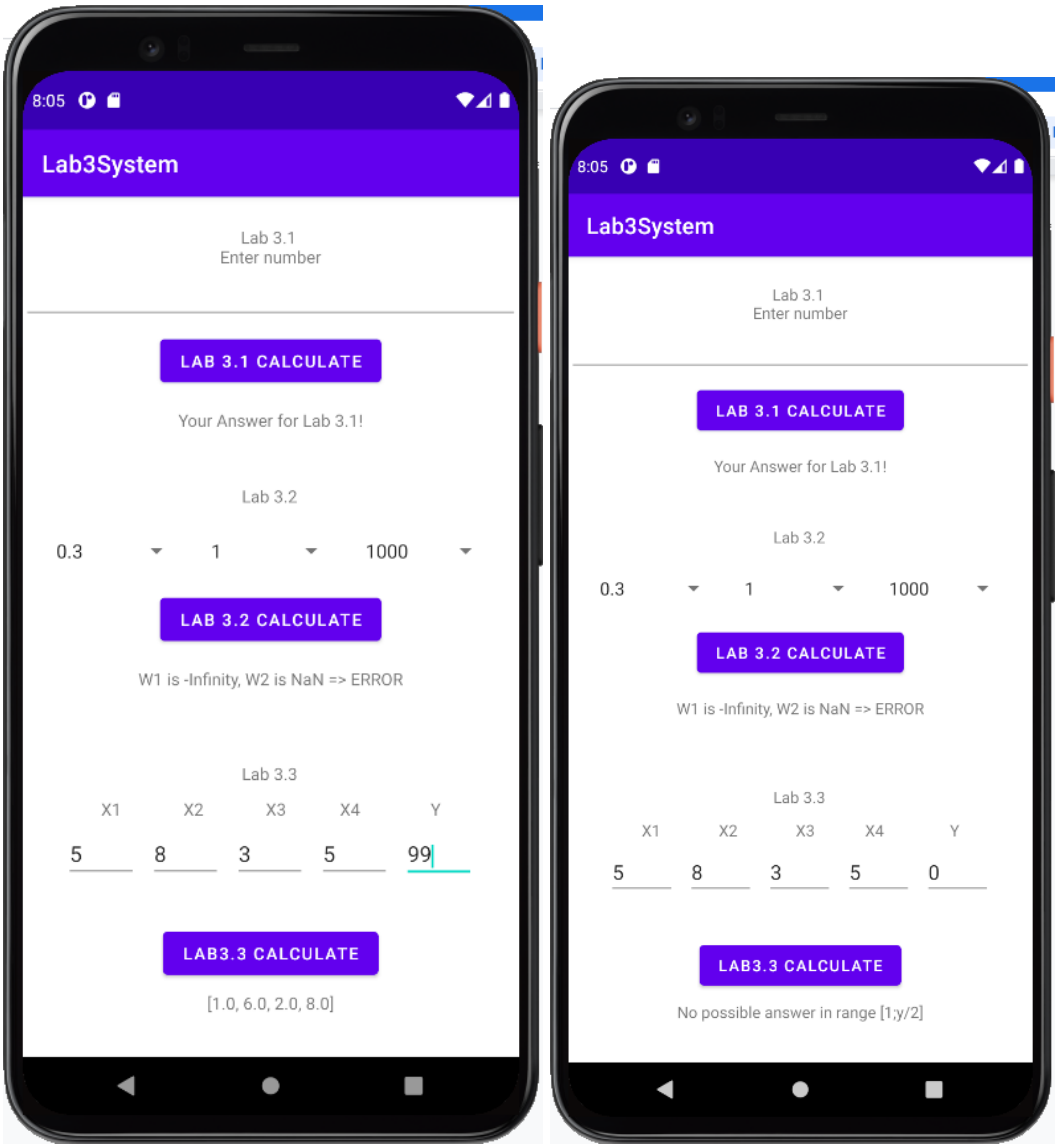

7:58
Lab3System
Lab 3.1
Enter number
2323
LAB 3.1 CALCULATE
[62, 39]
Lab 3.2
0.001
0.5
100
LAB 3.2 CALCULATE
Your Answer for Lab 3.2!
Lab 3.3
X1
X2
X3
X4
Y
LAB3.3 CALCULATE
Your Answer for Lab 3.3!


7:59
Lab3System
Lab 3.1
Enter number
2324
LAB 3.1 CALCULATE
Please enter odd number!
Lab 3.2
0.001
0.5
100
LAB 3.2 CALCULATE
Your Answer for Lab 3.2!
Lab 3.3
X1
X2
X3
X4
Y
LAB3.3 CALCULATE
Your Answer for Lab 3.3!

3.2


8:04
Lab3System
Lab 3.1
Enter number
LAB 3.1 CALCULATE
Your Answer for Lab 3.1!
Lab 3.2
0.01
1
100
LAB 3.2 CALCULATE
(0.26331515586254517, 0.747417498794124)
Lab 3.3
X1
X2
X3
X4
Y
LAB3.3 CALCULATE
Your Answer for Lab 3.3!
8:04
Lab3System
Lab 3.1
Enter number
LAB 3.1 CALCULATE
Your Answer for Lab 3.1!
Lab 3.2
0.3
1
1000
LAB 3.2 CALCULATE
W1 is -Infinity, W2 is NaN => ERROR
Lab 3.3
X1
X2
X3
X4
Y
LAB3.3 CALCULATE
Your Answer for Lab 3.3!

3.3



Висновки

У ході виконання лабораторної роботи проведено ознайомлення з основними принципами розкладання числа на прості множники 3 використанням різних алгоритмів факторизації.

Ознайомлення з принципами машинного навчання за допомогою математичної моделі сприйняття інформації Перцептрона. Змодельовано роботу нейронної мережі та досліджено вплив параметрів на час виконання та точність результату.

Ознайомлення з принципами реалізації генетичного алгоритму, досліджено особливості даного алгоритму з використанням засобів моделювання і сучасних програмних оболонок.