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

Лабораторна робота №3.3
з дисципліни
“Інтелектуальні вбудовані системи”
на тему
“ДОСЛІДЖЕННЯ ГЕНЕТИЧНОГО АЛГОРИТМУ”

Виконала:
студентка групи ІП-84
Романова Вікторія Андріївна
номер залікової книжки: 8418

Перевірив:
ас. кафедри ОТ
Регіда П. Г.

Київ 2021

Теоретичні відомості

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.

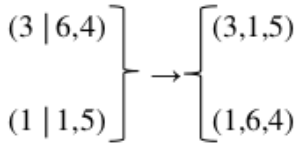
$1+1+2\cdot5=12 \quad \Delta=3 \quad \frac{\frac{1}{3}}{\frac{27}{24}} = 0{,}7$

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

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

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

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:



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 $\Delta=0$, тобто його значення будуть розв'язками рівняння.

Лістинг коду

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity"
    android:layout_margin="15dp">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/ax1_bx2_cx3_dx4_y"
        android:textAlignment="center"
        android:textSize="20sp" />

    <LinearLayout
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

    <EditText
        android:id="@+id/a"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:layout_weight="1"
        android:hint="@string/a"
        android:inputType="number"
        android:textAlignment="center"
        android:textSize="16sp" />

    <EditText
        android:id="@+id/b"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:layout_weight="1"
        android:hint="@string/b"
        android:inputType="number"
        android:textAlignment="center"
        android:textSize="16sp" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >
```

```
    <EditText
        android:id="@+id/c"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:layout_weight="1"
        android:hint="@string/c"
        android:inputType="number"
        android:textAlignment="center"
        android:textSize="16sp" />

    <EditText
        android:id="@+id/d"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:layout_weight="1"
        android:hint="@string/d"
        android:inputType="number"
        android:textAlignment="center"
        android:textSize="16sp" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <EditText
        android:id="@+id/y"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
```

```
        android:layout_margin="5dp"
        android:layout_weight="1"
        android:hint="@string/y"
        android:inputType="number"
        android:textAlignment="center"
        android:textSize="16sp" />
</LinearLayout>

<Button
    android:id="@+id/calculate"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="15dp"
    android:text="@string/calculate" />

<TableLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <TableRow>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:text="@string/best_result"
            android:textSize="16sp" />

        <TextView
            android:id="@+id/result"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
```

```
            android:textSize="16sp" />
        </TableRow>
        <TableRow>
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="5dp"
                android:text="@string/time"
                android:textSize="16sp" />

            <TextView
                android:id="@+id/time_ms"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="5dp"
                android:textSize="16sp" />
        </TableRow>
    </TableLayout>
</LinearLayout>
```

MainActivity.kt

```
package ua.kpi.comsys.lab33

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import ua.kpi.comsys.lab33.databinding.ActivityMainBinding
import kotlin.system.measureTimeMillis

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

    override fun onCreate(savedInstanceState: Bundle?) {
```

```
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        with(binding) {
            calculate.setOnClickListener {
                val argA = if (a.text.isNullOrBlank()) 0 else a.text.toString().toInt()
                val argB = if (b.text.isNullOrBlank()) 0 else b.text.toString().toInt()
                val argC = if (c.text.isNullOrBlank()) 0 else c.text.toString().toInt()
                val argD = if (d.text.isNullOrBlank()) 0 else d.text.toString().toInt()
                val argY = if (y.text.isNullOrBlank()) 0 else y.text.toString().toInt()

                val res: List<Int>?
                val time = measureTimeMillis {
                    res = try {
                        Roulette(argA, argB, argC, argD, argY).run()
                    } catch (e:Exception) {
                        null
                    }
                }

                val resultText = (res ?: "Not found :(").toString()
                val timeMsText = "$time ms"

                result.text = resultText
                timeMs.text = timeMsText
            }
        }
    }
}
```

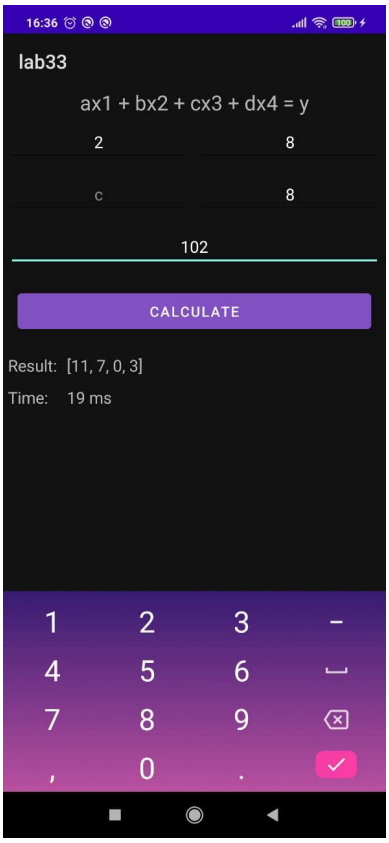
Roulette.kt

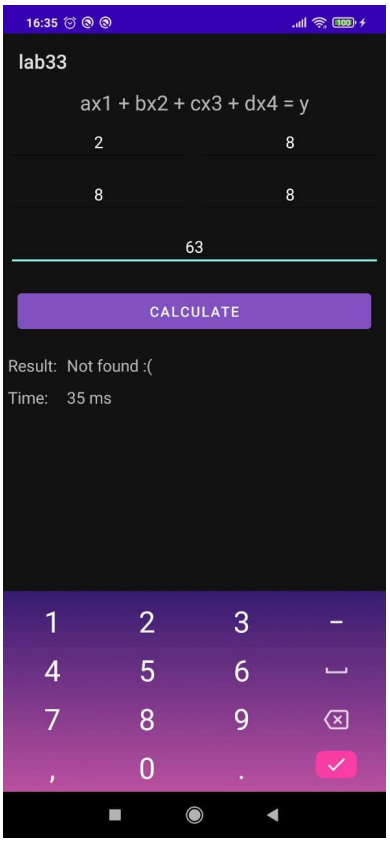
```
package ua.kpi.comsys.lab33

import kotlin.math.abs
import kotlin.random.Random


class Chromosome(private val range: IntRange,
                 private val coefficients: List<Int>,
                 private val y: Int) {
    var data = List(4) { range.random() }
    val fitness: Int by lazy {
        val result = data.mapIndexed { i, v -> coefficients[i] * v }.sum()
        abs(y - result)
    }
}

class Roulette(a: Int, b: Int, c: Int, d: Int, private val y: Int) {
    private val coefficients = listOf(a, b, c, d)
    private val range = 0..y / coefficients.maxOf { it }

    fun run(): List<Int>? {
        var chromosomes = initGeneration()
        for (g in 1..100) {
            chromosomes.forEach { if (it.fitness == 0) return it.data }
                    if (chromosomes.all { it.data == chromosomes.first().data })
chromosomes = initGeneration()

            chromosomes = selection(chromosomes).map(this::crossing).flatten()
            chromosomes.forEach(this::mutation)
        }
        return null
    }
```

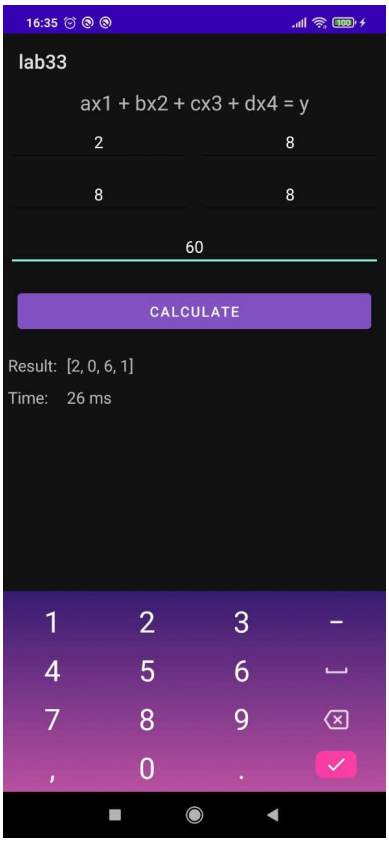
```
private fun initGeneration() = List(4) { Chromosome(range, coefficients, y) }

private fun generator(ch: List<Chromosome>): Chromosome {
    val roulette = ch.map { 1.0 / it.fitness }.sum()
    val chances = ch.map { 1.0 / it.fitness / roulette }
    val rand = Random.nextFloat()
    var a = 0.0
    for (i in chances.indices) {
        if (rand <= chances[i] + a) return ch[i]
        a += chances[i]
    }
    return ch.last()
}

private fun pair(ch: List<Chromosome>): List<Chromosome> {
    val ch1 = generator(ch)
    val ch2 = generator(ch.filterNot { it == ch1 })
    return listOf(ch1, ch2)
}

private fun selection(ch: List<Chromosome>) = List(2) { pair(ch) }

private fun crossing(pair: List<Chromosome>): List<Chromosome> {
    val ch1 = pair[0].data
    val ch2 = pair[1].data
    val point = Random.nextInt(1, ch1.lastIndex + 1)

    val ch3 = Chromosome(range, coefficients, y)
    ch3.data = ch1.subList(0, point) + ch2.subList(point, ch2.lastIndex + 1)

    val ch4 = Chromosome(range, coefficients, y)
```

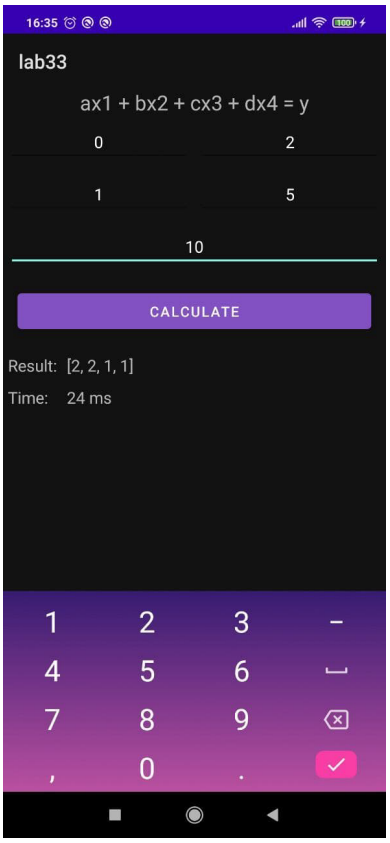
```
        ch4.data = ch2.subList(0, point) + ch1.subList(point, ch1.lastIndex + 1)

        return listOf(ch3, ch4)
    }

    private fun mutation(ch: Chromosome) {
        val mutationProbability = 0.1
        if (Random.nextFloat() <= mutationProbability) {
            val newData = ch.data.toMutableList()
            val index = Random.nextInt(newData.lastIndex + 1)

            val mutation = newData[index] + if (Random.nextBoolean()) 1 else -1
            if (mutation in range) {
                newData[index] = mutation
                ch.data = newData
            }
        }
    }
}
```

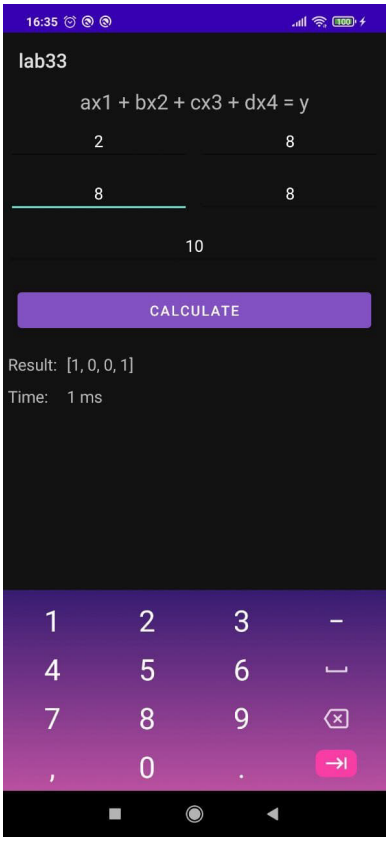
Результати виконання

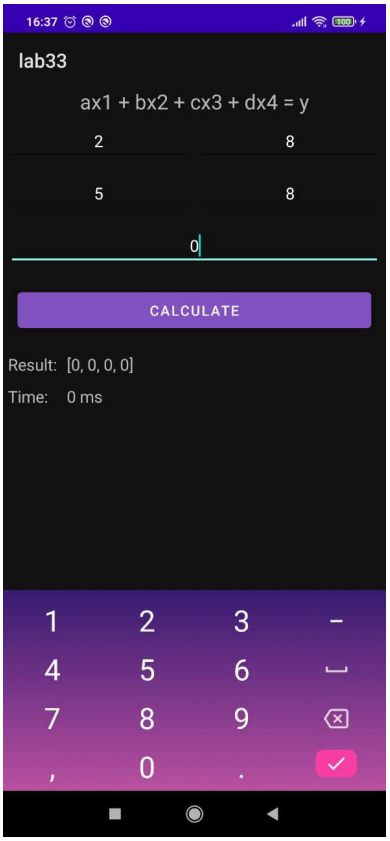


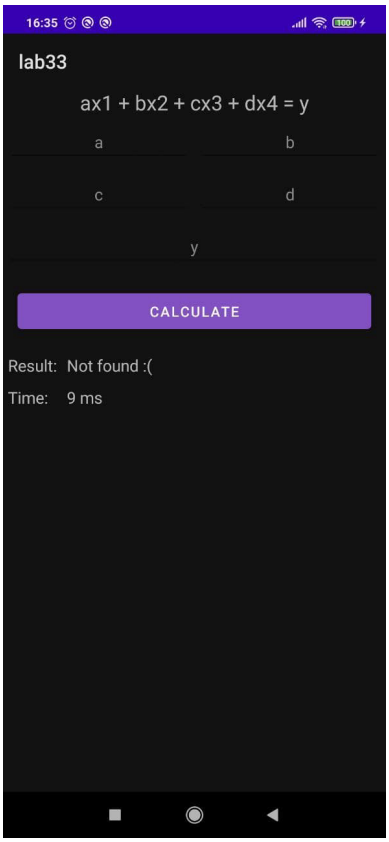







16:35
lab33
ax1 + bx2 + cx3 + dx4 = y
2
8
8
8
10
CALCULATE
Result: [1, 0, 0, 1]
Time: 1 ms


16:37
lab33
ax1 + bx2 + cx3 + dx4 = y
2
8
5
8
0
CALCULATE
Result: [0, 0, 0, 0]
Time: 0 ms


16:35
lab33
ax1 + bx2 + cx3 + dx4 = y
a
b
c
d
y
CALCULATE
Result: Not found :(
Time: 9 ms

Висновок

Було проведено ознайомлення з принципами реалізації генетичного алгоритму, вивчення та дослідження особливостей даного алгоритму з використанням засобів моделювання і сучасних програмних оболонок.