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

**Лабораторна робота №3-3**

З дисципліни «Методи оптимізації та планування»

# ДОСЛІДЖЕННЯ ГЕНЕТИЧНОГО АЛГОРИТМУ

ВИКОНАВ:
Студент II курсу ФІОТ
Групи ІО-93
Мудрий Юрій
Номер залікової - №9323

ПЕРЕВІРИВ:
Регіда П.Г.

Київ 2021 р.

**Мета роботи** - ознайомлення з принципами реалізації генетичного алгоритму,
вивчення та дослідження особливостей даного алгоритму з використанням засобів моделювання і сучасних програмних оболонок.

**Завдання на лабораторну роботу**

Налаштувати генетичний алгоритм для знаходження цілих коренів діофантового рівняння ax1+bx2+cx3+dx4=y. Розробити відповідний мобільний додаток і вивести отримані значення. Провести аналіз витрат часу на розрахунки.

**Код програми:**

```
package com.lab3a.activity;

import android.annotation.SuppressLint;
import android.os.Build; import
android.view.View; import
android.widget.Button; import
android.widget.EditText; import
android.widget.TextView;

import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import com.lab3a.R;
import com.lab3a.execution.EquationSolver; import
com.lab3a.utils.Permissions;
import com.lab3a.utils.exception.ItersExceededException;

import java.util.Arrays;

public class MainActivityInflater {

    @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
    public static void inflate(AppCompatActivity activity) {
        EditText edittext_input_a = activity.findViewById(R.id.edittext_input_a);
        EditText edittext_input_b = activity.findViewById(R.id.edittext_input_b);
        EditText  edittext_input_c  =  activity.findViewById(R.id.edittext_input_c);
EditText edittext_input_d = activity.findViewById(R.id.edittext_input_d);

        EditText edittext_input_y = activity.findViewById(R.id.edittext_input_y);
        Button button_count = activity.findViewById(R.id.button_count);



        TextView textview_output_result =
activity.findViewById(R.id.textview_output_result);
        @SuppressLint("SetTextI18n") View.OnClickListener onButtonCountClick = v -> {
            String string_a = String.valueOf(edittext_input_a.getText());
            String string_b = String.valueOf(edittext_input_b.getText());
            String string_c = String.valueOf(edittext_input_c.getText());
String string_d = String.valueOf(edittext_input_d.getText());
```

```
            String string_y = String.valueOf(edittext_input_y.getText());
```

```
            if (
                    string_a.trim().equals("") || string_b.trim().equals("") ||
string_c.trim().equals("") || string_d.trim().equals("")
||
                            string_y.trim().equals("")
) {

textview_output_result.setTextColor(activity.getResources().getColor(R.color.red));
textview_output_result.setText("Не введені дані");
            } else if (
                    string_a.trim().equals("-") || string_b.trim().equals("-") ||
string_c.trim().equals("-") || string_d.trim().equals("-
") ||
                            string_y.trim().equals("-")
) {

textview_output_result.setTextColor(activity.getResources().getColor(R.color.red));
textview_output_result.setText("Неправильно введені дані");
             } else {
                  long a =
 Long.parseLong(string_a);                long b =
 Long.parseLong(string_b);                long c =
 Long.parseLong(string_c);                long d =
 Long.parseLong(string_d);

                long y = Long.parseLong(string_y);
                EquationSolver solver = new EquationSolver();

                solver.setCoefficients(new long[] {a, b, c, d});
solver.setY(y);

                try {

solver.solve();
                    long[]          roots          =
solver.getRoots();

                    Permissions permissions = new Permissions(roots);

                    long[][] perms = permissions.getPerms();
                     for (long[] perm : perms)
{
                         if (a * perm[0] + b * perm[1] + c * perm[2] + d * perm[3]
== y)                                roots = perm;

                    }

                    StringBuilder out = new StringBuilder();
                     for (int i = 0; i < roots.length; i++) {
out.append("X").append(i + 1).append(" =
").append(roots[i]).append("\n");
                    }
```

```
textview_output_result.setTextColor(activity.getResources().getColor(R.color.green));
textview_output_result.setText(out);



                } catch (ItersExceededException exception) {

textview_output_result.setTextColor(activity.getResources().getColor(R.color.red));
textview_output_result.setText("Перевищена максимальна кількість ітерацій
(1000000)");

                }

            }

        };
        View.OnClickListener onButtonCountWithDifferentFractionsClick = v -> {
            String string_a = String.valueOf(edittext_input_a.getText());
            String string_b = String.valueOf(edittext_input_b.getText());
              String string_c = String.valueOf(edittext_input_c.getText());
        String string_d = String.valueOf(edittext_input_d.getText());

            String string_y = String.valueOf(edittext_input_y.getText());

            if (
                    string_a.trim().equals("") || string_b.trim().equals("") ||
string_c.trim().equals("") || string_d.trim().equals("") ||

                            string_y.trim().equals("")
) {

textview_output_result.setTextColor(activity.getResources().getColor(R.color.red));
textview_output_result.setText("Не введені дані");
            } else if (
                    string_a.trim().equals("-") || string_b.trim().equals("-") ||
string_c.trim().equals("-") || string_d.trim().equals("-
") ||
                            string_y.trim().equals("-")
) {

textview_output_result.setTextColor(activity.getResources().getColor(R.color.red));
textview_output_result.setText("Неправильно введені дані");
             } else {
                  long a =
 Long.parseLong(string_a);                long b =
 Long.parseLong(string_b);                long c =
 Long.parseLong(string_c);                long d =
 Long.parseLong(string_d);

                long y = Long.parseLong(string_y); EquationSolver[]
                solvers = new EquationSolver[10];

                boolean[] solved = new boolean[10];
```

```
for (int i = 0; i < solvers.length; i++) {
```

```
                    solvers[i] = new EquationSolver(0.1 * (i + 1));
                    solvers[i].setCoefficients(new long[] {a, b, c, d});
solvers[i].setY(y);
                    try {
                        solvers[i].solve();
                        solved[i] = true;
                    } catch (ItersExceededException exception) {
                        solved[i] = false;
                    }
                }
                EquationSolver bestIters =
solvers[0]; EquationSolver bestTime = solvers[0];
                for (int i = 0; i < solvers.length; i++) {
                    if (solved[i]) {
                        if (solvers[i].getIters() < bestIters.getIters()) bestIters =
solvers[i];
                        if (solvers[i].getTime() < bestTime.getTime()) bestTime =
solvers[i];
                    }
                }
                StringBuilder out = new StringBuilder("Обчислення з різними
відсотками мутації:\n\n");
                 for (int i = 0; i < solvers.length; i++)
{
                    out.append("Відсоток
мутації:
").append(Math.round(solvers[i].getFraction() * 100)).append(" %").append("\n");
                    if (solved[i]) {
                        long[] roots = solvers[i].getRoots(); Permissions
                        permissions = new Permissions(roots); long[][]
                        perms =
permissions.getPerms();
                        for (long[] perm : perms) {
                            if (a * perm[0] + b * perm[1] + c * perm[2] + d * perm[3]
== y)
                                roots = perm;
```

```
                    }
```

```
                        out.append("Обчислено: Так").append("\n");
                        out.append("Корені:
").append(Arrays.toString(roots)).append("\n");
out.append("Кількість ітерацій:
").append(solvers[i].getIters()).append("\n");
                        out.append("Час обчислення: ").append(solvers[i].getTime()
/ 1000000000.0).append(" с").append("\n");

                    } else {

                        out.append("Обчислено: Ні").append("\n");
                    }

                    out.append("\n");

                }

                boolean any = solved[0];

                for (int i = 1; i < solved.length; i++) any |= solved[i];

                out.append("\n");

                if (any) {
                     out.append("Найкращий відсоток за кількістю ітерацій: ").
append(Math.round(bestIters.getFraction() * 100)).append("
%").append("\n\n");
                    out.append("Найкращий відсоток за часом: ").
append(Math.round(bestTime.getFraction() * 100)).append(" %").append("\n");

                }


textview_output_result.setTextColor(activity.getResources().getColor(R.color.green));
textview_output_result.setText(out);
            }

        };

        button_count.setOnClickListener(onButtonCountClick);

    }

}

}
```

**Результат роботи програми:**

**Висновок:**

Виконавши цю роботу, я ознайомився з принципами реалізації генетичного алгоритму, вивчив та дослідив особливості даного алгоритму з використанням засобів моделювання і сучасних програмних оболонок.