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 №3-2

З дисципліни «Методи оптимізації та планування»

# ДОСЛІДЖЕННЯ ГЕНЕТИЧНОГО АЛГОРИТМУ

ВИКОНАВ:

Студент ІІ курсу ФІОТ

Групи IO-93 Ященко Є.В. - 9331

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) {

== y)

if (a \* perm[0] + b \* perm[1] + c \* perm[2] + d \* perm[3] 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");

%").append("\n");

out.append("Найкращий відсоток за часом: "). append(Math.*round*(bestTime.getFraction() \* 100)).append("

}

textview\_output\_result.setTextColor(activity.getResources().getColor(R.color.*green*)); textview\_output\_result.setText(out);

}

};

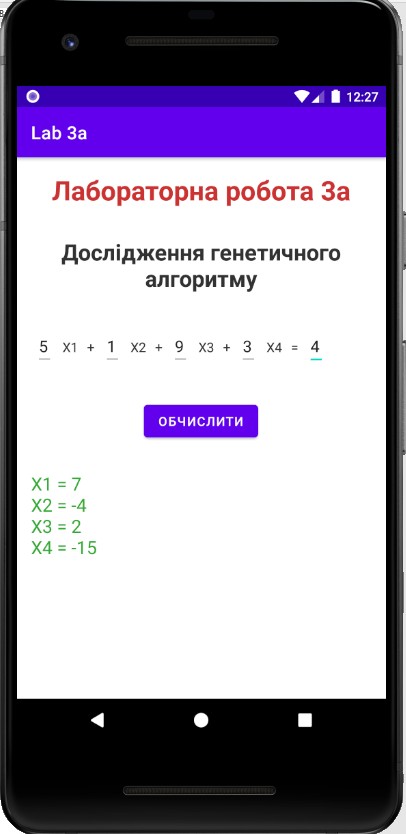
button\_count.setOnClickListener(onButtonCountClick);

}

}

}

# 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.