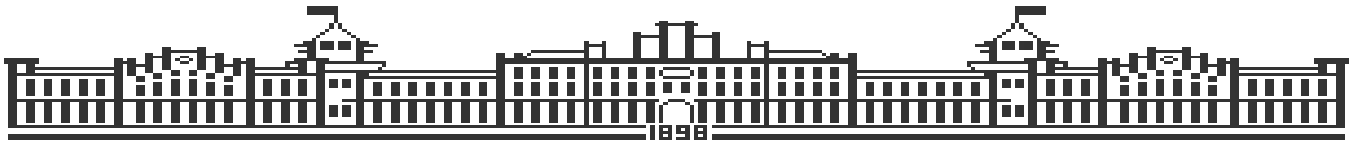


Міністерство освіти і науки України
Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

ЗВІТ

про виконання лабораторної роботи №3.3

з дисципліни

«Інтелектуальні вбудовані системи»

на тему:

«ДОСЛІДЖЕННЯ ГЕНЕТИЧНОГО АЛГОРИТМУ»

Перевірив:
асистент кафедри ОТ
Регіда П. Г

ВИКОНАВ:
студент 3 курсу
групи ІП-83, ФІОТ
Мінченко В.Ю.
Залікова книжка №8315
Варіант – 18

Київ 2021

Завдання на лабораторну роботу

Налаштувати генетичний алгоритм для знаходження цілих коренів діофантового рівняння $ax_1+bx_2+cx_3+dx_4=y$. Розробити відповідний мобільний додаток і вивести отримані значення. Провести аналіз витрат часу на розрахунки

Програмний код:

Genetic_algorithm.dart

```
import 'dart:math';

String geneticAlgorithm(
  List<String> inputEquation,
  int numberPopulations,
  int maxIterations
) {
  final stopwatch = Stopwatch()..start();
  final inputCoefficients = List.generate(inputEquation.length,
    (index) => int.parse(inputEquation[index]));
  final yValue = inputCoefficients.removeLast();
  final maxCoefficient = inputCoefficients.reduce(max);
  final maxGeneValue = (yValue/maxCoefficient).ceil();
  var currentPopulation = generateStartPopulation(
    numberPopulations,
    inputCoefficients.length,
    maxGeneValue
  );
  var iterations = maxIterations;
  var iterationCounter = 0;
  while(iterations == 0 || iterations > 0) {
    iterationCounter++;
    List<int> result;
    final deltasFitness = currentPopulation.map<int>((chromosome) {
      final delta = calcFitness(inputCoefficients, chromosome, yValue);
      if (delta == 0) result = chromosome;
      return delta;
    }).toList();
    if (result != null) return result.toString() + '\niterations: $iterationCounter \ntim
e: ${stopwatch.elapsedMilliseconds / 1000}';
    final probabilities = calcProbability(deltasFitness);
    final rouletteElements = currentPopulation
      .asMap()
      .map((index, element) {
        Map<String, dynamic> chromosomeExt = Map();
        chromosomeExt['chromosome'] = element;
        chromosomeExt['probability'] = probabilities[index];
        return MapEntry(index, chromosomeExt);
      })
      .values
      .toList();
    currentPopulation = [];
    for (var i = 0; i < numberPopulations/2; i++) {
      final selectedGenes = calcRoulette(rouletteElements);
      final mixedGenes = mixChromosomesGene(selectedGenes);
```

```
      final mutatedGenes = mixedGenes.map(
        (gene) => calcMutation(gene, maxGeneValue));
      currentPopulation..addAll(mutatedGenes);
    }
    if (maxIterations != 0)
    iterations--;
  }
}

int generateRandomValue(int max) => Random().nextInt(max);

List<List<int>> generateStartPopulation(int numberPopulations, int varNumber, int yMax) =
>
  List.generate(numberPopulations, (index) => List.generate(varNumber, (index) => generat
eRandomValue(yMax)));

int calcFitness(List<int> inputCoefficients, List<int> chromosome, int yValue) {
  var sum = 0;
  chromosome.asMap().forEach((index, gene) => sum += gene * inputCoefficients[index]);
  return (yValue - sum).abs();
}

double calcInvertedSumDeltas(List<int> deltas) => deltas.fold<double>(0, (previousValue,
currentValue) => previousValue + 1 / currentValue);
List<double> calcProbability(List<int> deltas) =>
  deltas.map<double>((delta) => 1 / delta / calcInvertedSumDeltas(deltas)).toList();

List<List<int>> calcRoulette(List<Map<String, dynamic>> elements, {numWins = 2}) =>
  List.generate(numWins, (index) => selectRandom(elements));

List<int> selectRandom(List<Map<String, dynamic>> elements) {
  var randomValue = Random().nextDouble();
  List<List<int>> result = [];
  elements
    .forEach((element) =>
      (randomValue -
= element['probability']) < 0 ? result.add(element['chromosome']) : null);
  return result[0];
}

List<List<int>> mixChromosomesGene(List<List<int>> parents) {
  final parentFirst = parents[0];
  final parentSecond = parents[1];
  final index = (parentFirst.length / 2).floor();
  return [
    [...parentFirst.sublist(0, index), ...parentSecond.sublist(index)],
    [...parentSecond.sublist(0, index), ...parentFirst.sublist(index)]
  ];
}

List<int> calcMutation(List<int> chromosome, int maxGene, {double thresholdProbability =
0.1}) {
  final random = Random().nextDouble();
  final value = Random().nextInt(maxGene);
```

```
  final i = Random().nextInt(chromosome.length);
  if (thresholdProbability >= random) {
    return chromosome.asMap().map((index, gene) =>
      MapEntry(index, i == index ? value : gene)
    )
    .values
    .toList();
  }
  return chromosome;
}

// void main() {
//   print(geneticAlgorithm(['1', '1', '2', '4', '45'], 4, 0));
// }
```

Genetic_algorithm_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:lab3_mobile/helpers/genetic_algorithm.dart';

class GeneticAlgorithm extends StatefulWidget {
  @override
  _GeneticAlgorithmState createState() => _GeneticAlgorithmState();
}

class _GeneticAlgorithmState extends State<GeneticAlgorithm> {
  final allControllers = List<TextEditingController>.generate(5, (index) => TextEditingCo
ntroller());
  bool _offstage = true;
  String resultValue = '';

  @override
  void dispose() {
    allControllers.forEach((controller) => controller.dispose());
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Container(
      child: Padding(
        padding: const EdgeInsets.all(50.0),
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  _buildElementQuation(0),
                  Text('x1 + '),
                  _buildElementQuation(1),
                  Text('x2 + '),
```

```
                _buildElementQuation(2),
                Text('x3 + '),
                _buildElementQuation(3),
                Text('x4 = '),
                _buildElementQuation(4),
              ],
            ),
            Offstage(
              offstage: _offstage,
              child: Padding(
                padding: const EdgeInsets.all(25.0),
                child: Text(
                  resultValue,
                  style: TextStyle(
                    color: Colors.orange
                  )
                ),
              ),
            ),
            _offstage ? SizedBox(height: 8.0) : SizedBox(height: 0.0),
            ElevatedButton(
              child: const Text('Calculate'),
              style: ElevatedButton.styleFrom(
                primary: Colors.purple,
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.all(Radius.circular(10.0))
                ),
              ),
              onPressed: () {
                setState(() {
                  _offstage = false;
                  resultValue = geneticAlgorithm(
                    allControllers.map((controller) => controller.text).toList(),
                    4,
                    0
                  );
                });
              },
            ),
          ],
        ),
      ),
    ),
  );
}

Widget _buildElementQuation(int controllerIndex) =>
  Container(
    width: 20,
    child: TextField(
      controller: allControllers[controllerIndex],
      keyboardType: TextInputType.number,
      inputFormatters: [FilteringTextInputFormatter.digitsOnly],
      textAlign: TextAlign.center,
```

```
        style: TextStyle(
          color: Colors.red,
        ),
      ),
    );
}
```

Bottom_navigation.dart

```
import 'package:flutter/material.dart';

class TabItem {
  TabItem({this.label, this.title, this.icon, this.backgroundColor});
  final String label;
  final String title;
  final icon;
  final Color backgroundColor;
}

List<TabItem> allTabItems = <TabItem>[
  TabItem(
    icon: Icon(Icons.miscellaneous_services),
    label: 'factorization',
    title: 'Fermat`s factorization example',
    backgroundColor: Colors.redAccent[400]),
  TabItem(
    icon: Icon(Icons.mediation),
    label: 'perceptron',
    title: 'Perceptron example',
    backgroundColor: Colors.tealAccent[400]),
  TabItem(
    icon: Icon(Icons.developer_board),
    label: 'genetic',
    title: 'Genetic algorithm for diaphantine equation',
    backgroundColor: Colors.deepPurpleAccent[400]),
];

class BottomNavigation extends StatelessWidget {
  BottomNavigation({@required this.currentIndex, @required this.onSelectTab});

  final int currentIndex;
  final ValueChanged<int> onSelectTab;

  @override
  Widget build(BuildContext context) {
    return BottomNavigationBar(
      type: BottomNavigationBarType.shifting,
      items: allTabItems
        .map((TabItem tabItem) => BottomNavigationBarItem(
            icon: tabItem.icon,
            backgroundColor: tabItem.backgroundColor,
            label: tabItem.label))
        .toList(),
      currentIndex: currentIndex,
```

```
      selectedItemColor: Colors.white,
      unselectedItemColor: Colors.grey,
      onTap: onSelectTab,
      iconSize: 30,
    );
  }
}
```

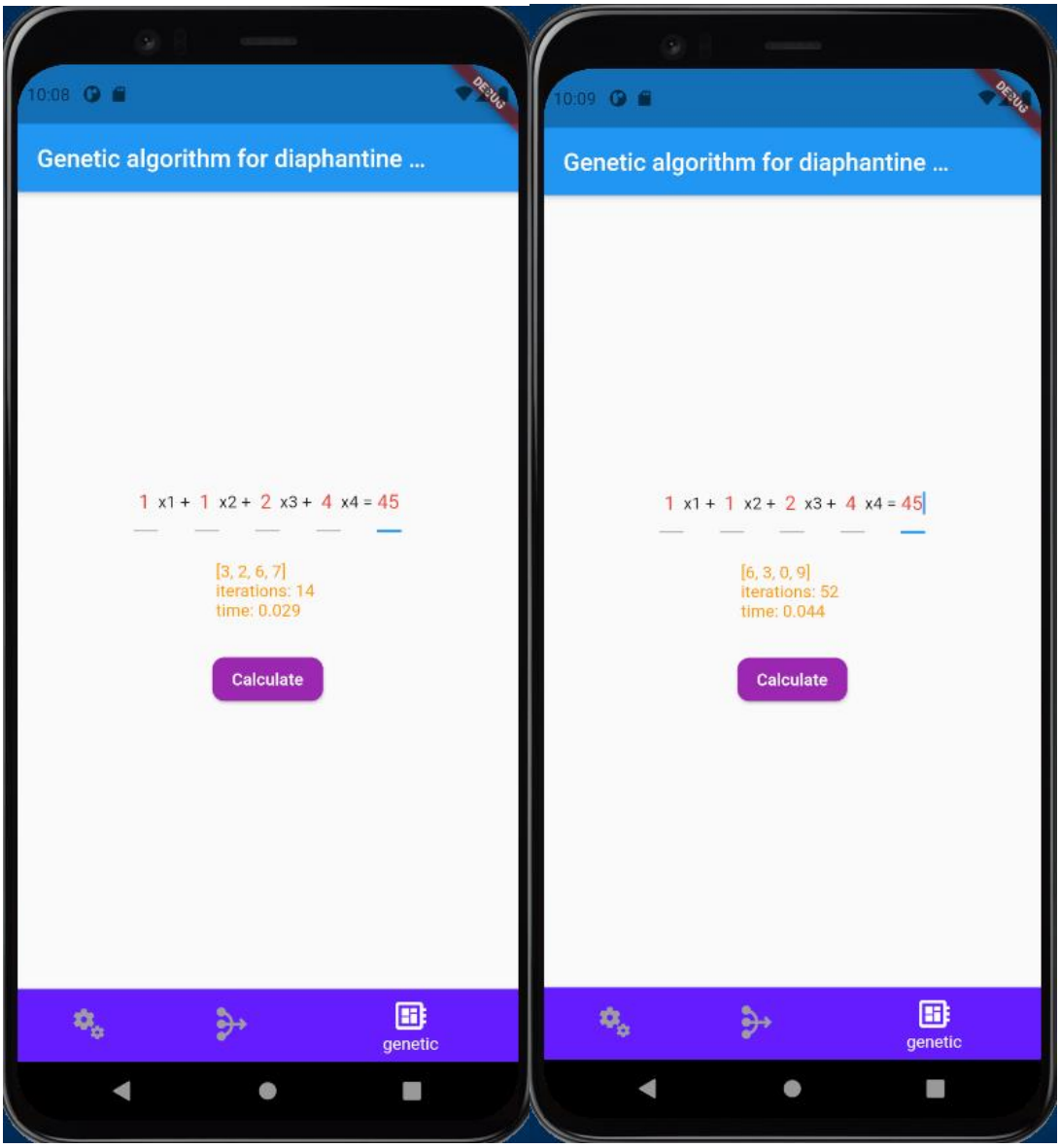
Main.dart

```dart
import 'package:flutter/material.dart';
import 'screens/main_screen.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: MainScreen(),
    );
  }
}
```

Результати роботи програми:



Висновки:
Отже, в ході лабораторної роботи, ми отримали навички з розв’язування діафантового рівняння методом генетичного алгоритму з програмною реалізацію для мобільних додатків.
Результати наведено в звіті та врепозиторії. Кінцеву мету було досягнуто.