

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

**ЗВІТ**  
з лабораторної роботи №3.3 з дисципліни  
«Інтелектуальні вбудовані системи»

Виконав:  
ІП-83  
Сергійчук Н. С.

Київ 2020

## 1. Завдання

Налаштувати генетичний алгоритм для знаходження цілих коренів діофантового рівняння  $ax_1 + bx_2 + cx_3 + dx_4 = y$ . Розробити відповідний мобільний додаток і вивести

2. отримані значення. Провести аналіз витрат часу на розрахунки. Лістинг програми

```
import 'dart:math';

abstract class Phenotype {
  int fitness();
  List<Phenotype> crossover(Random rng, Phenotype other);
  Phenotype mutate(Random rng);
}
```

```
class CachedPhenotype {
  Phenotype ph;
  int fitness;
```

```
  CachedPhenotype(this.ph, this.fitness);
}
```

```
class WeightedPhenotype {
  Phenotype ph;
  double weight;
```

```
  WeightedPhenotype(this.ph, this.weight);
```

```
  String toString() => "Wp($weight)";
```

```
}
```

```
List<WeightedPhenotype> weight(List<CachedPhenotype> ph) {  
    double sum =  
    ph.map((w) => 1.0 / w.fitness).fold(0, (prev, next) => prev + next);
```

```
    var weighted =  
    ph.map((c) => WeightedPhenotype(c.ph, 1.0 / c.fitness / sum)).toList();
```

```
    for (int i = 1; i < weighted.length; i++) {  
        weighted[i].weight += weighted[i - 1].weight;  
    }  
    return weighted;  
}
```

```
Phenotype chooseWeighted(Random rng, List<WeightedPhenotype> ph) {  
    var random = rng.nextDouble();  
    for (var w in ph) {  
        if (random < w.weight) {  
            return w.ph;  
        }  
    }  
    throw Unreachable();  
}
```

```
class Unreachable extends Error {}
```

```
class Simulator {
```

```
List<Phenotype> _population;
int maxIters;
int iters = 0;
Random rng = Random();
Stopwatch stopwatch = Stopwatch();
```

```
Simulator(this._population, {this.maxIters});
```

```
Duration get elapsed => stopwatch.elapsed;
```

```
Phenotype run() {
stopwatch.start();
while (true) {
var sorted = _population
.map((ph) => CachedPhenotype(ph, ph.fitness()))
.toList()
..sort((a, b) => a.fitness.compareTo(b.fitness));
assert(sorted.first.fitness >= 0);
```

```
if (sorted.first.fitness == 0 || iters == maxIters) {
stopwatch.stop();
return sorted.first.ph;
}
```

```
var weights = weight(sorted);
var nextPopulation = <Phenotype>[];
for (int i = 0; i < _population.length / 2; i++) {
var mom = chooseWeighted(rng, weights);
```

```

var dad = chooseWeighted(rng, weights);
var child = mom.crossover(rng, dad);
nextPopulation.addAll(child);
}
for (var ph in nextPopulation) {
ph.mutate(rng);
}
_population = nextPopulation;
iters++;
}
}
}

```

```

class DiophantineEquation {
List<int> coeffs;
int goal;

```

```

DiophantineEquation(this.coeffs, this.goal);
}

```

```

class Diophantine extends Phenotype {
List<int> solutions;
DiophantineEquation equation;

```

```

Diophantine(this.equation, this.solutions);

```

```

@Override
List<Phenotype> crossover(Random rng, Phenotype other) {

```

```

var dad = other as Diophantine;
var len = solutions.length;
var mid = rng.nextInt(len);
var alice = Diophantine(equation, List<int>.filled(len, 0));
var bob = Diophantine(equation, List<int>.filled(len, 0));
for (int i = 0; i < len; i++) {
    if (i < mid) {
        alice.solutions[i] = solutions[i];
        bob.solutions[i] = dad.solutions[i];
    } else {
        alice.solutions[i] = dad.solutions[i];
        bob.solutions[i] = solutions[i];
    }
}
return [alice, bob];
}

```

```

@Override
int fitness() {
    int sum = 0;
    for (int i = 0; i < solutions.length; i++) {
        sum += solutions[i] * equation.coeffs[i];
    }
    sum -= equation.goal;
    return sum < 0 ? -sum : sum;
}

```

```

@Override

```

```

Phenotype mutate(Random rng) {
var index = rng.nextInt(solutions.length);
var delta = rng.nextBool() ? 1 : -1;
solutions[index] += delta;
return this;
}

```

```

String toString() {
var result = "";
var plus = "";
for (int i = 0; i < solutions.length; i++) {
result += "$plus${solutions[i]} * ${equation.coeffs[i]}";
plus = " + ";
}
result += " = ${equation.goal}";
return result;
}
}

```

```

// void main() {
// var rng = Random();
// var coeffs = List<int>.generate(10, (index) => index);
// var equation = DiophantineEquation(coeffs, 10);
// var start = List<Diophantine>.generate(10, (index) {
// var solutions = List<int>.generate(10, (index) => rng.nextInt(5));
// return Diophantine(equation, solutions);
// });
// var simulation = Simulator(start, maxIters: 100);

```

```
// var result = simulation.run();  
// print(result);  
// }
```

```
/=====
```

```
import 'dart:math';  
  
import 'package:flutter/material.dart';  
import 'package:flutter/services.dart';  
import 'package:lab3_3/genetic.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,  
      ),  
      home: MyHomePage(title: 'Flutter Demo Home Page'),  
    );  
  }  
}
```



```
);  
}  
}
```

```
class MyHomePage extends StatefulWidget {  
  MyHomePage({Key key, this.title}) : super(key: key);  
  final String title;  
  @override  
  _MyHomePageState createState() => _MyHomePageState();  
}
```

```
class _MyHomePageState extends State<MyHomePage> {  
  List<int> result = [0, 0, 0, 0];  
  List<int> coeffs = [1, 1, 1, 1, 1];  
  String stats = "";  
  Random rng = Random();  
  final int populationSize = 10;  
  final int eqautionSize = 4;  
  final int maxIters = 100;
```

```
void _runSimulation() {  
  setState(() {  
    var cfs = coeffs.take(eqautionSize).toList();  
    var goal = coeffs.last;  
    var eqaution = DiophantineEquation(cfs, eqautionSize);  
    var start = List<Diophantine>.generate(populationSize, (index) {  
    var solutions =  
    List<int>.generate(eqautionSize, (index) => rng.nextInt(goal));
```

```

return Diophantine(eqaution, solutions);
});

var simulation = Simulator(start, maxIters: maxIters);
var res = simulation.run() as Diophantine;
result = res.solutions;
var time = simulation.elapsed.inMilliseconds;
var iters = simulation.iters;
stats = "Took: ${time}ms\nIterations: $iters";
});
}

```

```

@override

Widget build(BuildContext context) {
  // This method is rerun every time setState is called, for instance as
done

  // by the _incrementCounter method above.

  //

  // The Flutter framework has been optimized to make rerunning build
methods

  // fast, so that you can just rebuild anything that needs updating rather
// than having to individually change instances of widgets.

  return Scaffold(
    appBar: AppBar(
      // Here we take the value from the MyHomePage object that was created by
      // the App.build method, and use it to set our appbar title.
      title: Text(widget.title),
    ),
    body: Container(
      padding: const EdgeInsets.all(20.0),

```

```

child: Column(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: List<Widget>.generate(
        5,
        (index) => _buildInputField(index),
      ),
    ),
    OutlinedButton(
      child: Text("Run simulation"),
      onPressed: () => _runSimulation(),
    ),
    Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: List<Widget>.generate(
        result.length, (i) => Text("x${i + 1} : ${result[i]}")),
    ),
    Text(
      stats,
      textAlign: TextAlign.left,
    ),
  ],
),
);
}

```

```

Widget _buildInputField(int index) {

```

```

var label = index < 4 ? "a${index + 1}" : "b";
return Expanded(
  child: Container(
    child: TextField(
      decoration: InputDecoration(labelText: label),
      keyboardType:
        TextInputType.numberWithOptions(signed: true, decimal: false),
      onChanged: (String value) => coeffs[index] = int.tryParse(value) ?? 1,
      inputFormatters: [
        FilteringTextInputFormatter(RegExp("^-?[0-9]{0,4}"), allow: true)
      ],
    ),
    padding: const EdgeInsets.symmetric(horizontal: 10),
  ),
);
}
}

```

### 3. Результати виконання кожної програми.



