Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Фізико-технічний інститут

**ЛАБОРАТОРНА РОБОТА №1**

З дисципліни«Криптографія»

«Побудова регістрів зсуву з лінійним зворотним зв’язком та дослідження їх властивостей»

Виконали:

студенти 3 курсу ФТІ

групи ФБ-73  
Дем’яненко Д.

Проноза А.

Перевірив:

Чорний О.

**Мета роботи**

Ознайомлення з деякими принципами побудови криптосистем на лінійних регістрах зсуву; практичне освоєння програмної реалізації лінійних регістрів зсуву (ЛРЗ); ознайомлення з методом кореляційного аналізу криптосистем на прикладі генератора Джиффі.

**Порядок виконання роботи**

1. Уважно прочитали методичні вказівки до виконання комп’ютерного практикуму.

2. За даними характеристичними многочленами р1(x), р2(x) склали лінійні рекурентні співвідношення для ЛРЗ, що задаються цими характеристичними многочленами.

3. Написати програми роботи кожного з ЛРЗ , .

4. За допомогою цих програм згенерували імпульсні функції для кожного з ЛРЗ і підрахували їх періоди.

5. За отриманими результатами зробити висновки щодо влавстивостей кожного з характеристичних многочленів р1(x), р2(x)

6. Для кожної з двох імпульсних функцій обчислили розподіл k-грам на періоді, k≤, де - степінь полінома (x), і=1,2 а також значення функції автокореляції А(d) для 0≤ d ≤ 10.

**Варіант 7**

P1(X)= X^23+X^20+X^7+X^6+X^5+X^2+1

P2(X)= X^20 + X^13 + X^12 + X^10 + X^6 + X^5 + X^4 + X^3 + X^2 + X + 1

Хід роботи

**Довжини періодів:**

|  |  |
| --- | --- |
| P1: 8388607 | P1 є примітивним поліномом, так як його період дорівнює максимальному періоду (T=q^n-1) |
| P2: 349525 | P2 є непримітивним та незвідним поліномом над даним полем, так як його період кратний q^n-1 |

**Розподіл k-грам на періодах імпульсних функцій ,**

|  |  |  |
| --- | --- | --- |
|  | 2-грами | "00" = 0,2499999404  "11" = 0,2500000596  "01" = 0,2499999404  "10" = 0,2499999404 |
| 3-грами | "000" = 0,1249999255  "011" = 0,1250000447  "110" = 0,1250000447  "001" = 0,1250000447  "100" = 0,1249999255  "111" = 0,1250000447  "101" = 0,1249999255  "010" = 0,1249999255 |
| 4-грами | "1110" = 0,0625000298  "1000" = 0,0624999106  "1011" = 0,0625000298  "0010" = 0,0625000298  "1010" = 0,0624999106  "0101" = 0,0624999106  "0000" = 0,0624999106  "0011" = 0,0625000298  "1101" = 0,0625000298  "0110" = 0,0625000298  "0001" = 0,0625000298  "0100" = 0,0624999106  "1001" = 0,0625000298  "0111" = 0,0625000298  "1100" = 0,0625000298  "1111" = 0,0625000298 |
| 5-грами | "00101" = 0,0312500186  "00000" = 0,0312498994  "00001" = 0,0312500186  "00100" = 0,0312500186  "01111" = 0,0312500186  "01110" = 0,0312500186  "01011" = 0,0312500186  "01010" = 0,0312498994  "10011" = 0,0312500186  "11100" = 0,0312500186  "10110" = 0,0312500186  "11001" = 0,0312500186  "10010" = 0,0312500186  "11000" = 0,0312500186  "11101" = 0,0312500186  "10111" = 0,0312500186  "11010" = 0,0312500186  "01101" = 0,0312500186  "00011" = 0,0312500186  "00110" = 0,0312500186  "01001" = 0,0312500186  "00111" = 0,0312500186  "01100" = 0,0312500186  "01000" = 0,0312498994  "00010" = 0,0312500186  "10000" = 0,0312498994  "11111" = 0,0312500186  "11011" = 0,0312500186  "11110" = 0,0312500186  "10101" = 0,0312498994  "10001" = 0,0312500186  "10100" = 0,0312498994 |
|  | 2-грами | "00" = 0,2507324247  "11" = 0,2497539511  "01" = 0,2497568121  "10" = 0,2497539511 |
| 3-грами | "000" = 0,1251219519  "011" = 0,1248759023  "110" = 0,1248759023  "001" = 0,1256111901  "100" = 0,1256083291  "111" = 0,1248787633  "101" = 0,1241463366  "010" = 0,1248787633 |
| 4-грами | "1000" = 0,0624366993  "1011" = 0,0620733459  "1110" = 0,0624395603  "0010" = 0,0628057747  "1010" = 0,0620733459  "0101" = 0,0620733459  "0000" = 0,0626827496  "0011" = 0,0628029137  "0110" = 0,0624366993  "1101" = 0,0620733459  "0001" = 0,0624395603  "0100" = 0,0628057747  "0111" = 0,0624395603  "1100" = 0,0628029137  "1001" = 0,0631719892  "1111" = 0,0624395603 |
| 5-грами | "00101" = 0,0312198695  "00000" = 0,0316461672  "00001" = 0,0310367617  "00100" = 0,0315860850  "01111" = 0,0312198695  "01011" = 0,0310367617  "01110" = 0,0312198695  "01010" = 0,0310367617  "11100" = 0,0314029772  "11001" = 0,0315860850  "10011" = 0,0315860850  "10110" = 0,0310367617  "10010" = 0,0315860850  "11000" = 0,0312170084  "10111" = 0,0310367617  "11101" = 0,0310367617  "11010" = 0,0310367617  "00011" = 0,0312170084  "00110" = 0,0314001162  "01101" = 0,0310367617  "00111" = 0,0314029772  "01001" = 0,0315860850  "01100" = 0,0314001162  "01000" = 0,0312198695  "00010" = 0,0312198695  "10000" = 0,0310339007  "11111" = 0,0312198695  "11011" = 0,0310367617  "11110" = 0,0312198695  "10101" = 0,0308536540  "10100" = 0,0312198695  "10001" = 0,0314029772 |

**Значення автокореляції для ,**

|  |  |
| --- | --- |
|  | d = 1 : 4194304  d = 2 : 4194304  d = 3 : 4194304  d = 4 : 4194304  d = 5 : 4194304  d = 6 : 4194304  d = 7 : 4194304  d = 8 : 4194304  d = 9 : 4194304 |
|  | d = 1 : 174592  d = 2 : 175104  d = 3 : 174592  d = 4 : 174592  d = 5 : 174592  d = 6 : 174592  d = 7 : 174592  d = 8 : 175104  d = 9 : 174592 |

**Висновок:**

Ознайомилися з принципами побудови регістрів зсуву з лінійним зворотним зв’язком; практично освоїли принципи їх програмної реалізації; дослідили властивості лінійних рекурентних послідовностей та їх залежності від властивостей характеристичного полінома регістра.

Код

import java.io.\*;

import java.text.DecimalFormat;

import java.util.\*;

public class Main {

public static final int[] coefficientsP1 = {1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0};

public static final int[] coefficientsP2 = {1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0};

public static ArrayList<Integer> sequenceP1 = new ArrayList<>();

public static ArrayList<Integer> sequenceP2 = new ArrayList<>();

public static void main(String[] args) {

System.out.println("var7:\n" +

"p1(x)= x^23 + x^20 + x^7 + x^6 + x^5 + x^2 + 1\n" +

"p2(x)= x^20 + x^13 + x^12 + x^10 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1");

int[] impulseFunctionP1 = new int[coefficientsP1.length];

impulseFunctionP1[coefficientsP1.length - 1] = 1;

int[] impulseFunctionP2 = new int[coefficientsP2.length];

impulseFunctionP2[coefficientsP2.length - 1] = 1;

System.out.println("impulse function for p1: " + Arrays.toString(impulseFunctionP1));

System.out.println("impulse function for p2: " + Arrays.toString(impulseFunctionP2));

int periodP1 = linearShiftRegister(coefficientsP1, impulseFunctionP1, sequenceP1, "sequenceP1.txt");

int periodP2 = linearShiftRegister(coefficientsP2, impulseFunctionP2, sequenceP2, "sequenceP2.txt");

System.out.println("Згенеровані послідовності ви можете переглянути у файлах sequenceP1.txt та sequenceP2.txt");

System.out.println("Period P1: " + periodP1);

System.out.println("Period P2: " + periodP2);

int maxPeriodP1 = (int) Math.pow(2, coefficientsP1.length) - 1;

int maxPeriodP2 = (int) Math.pow(2, coefficientsP2.length) - 1;

detectPolinomType(periodP1, maxPeriodP1, "P1");

detectPolinomType(periodP2, maxPeriodP2, "P2");

// HERE WE CALCULATE FREQUENCY N-GRAM AND WRITE RESULT IN THE FILE (THIS PROCESS ENOUGH LONG, SO IT'S COMMENTED,

// BECAUSE FILES WAS ALREADY CREATED AND IT WILL BE JUST REWRITED IF WE UNCOMMENTED THIS)

// for (int i = 2; i < coefficientsP1.length + 1; i++) {

// createNgramFrequency(readFile("sequenceP1.txt"), i, "P1");

// }

// for (int i = 2; i < coefficientsP2.length + 1; i++) {

// createNgramFrequency(readFile("sequenceP2.txt"), i, "P2");

// }

System.out.println("Calculate auto correlation coefficients: ");

System.out.println("For P1:");

for (int i =1 ; i< 10 ; i++){

System.out.println("d = " + i + " : " + calculateAutoCor(sequenceP1, periodP1, i));

}

System.out.println("\nFor P2:");

for (int i =1 ; i< 10 ; i++){

System.out.println("d = " + i + " : " + calculateAutoCor(sequenceP2, periodP2, i));

}

}

public static void showLinearRecurrenceRelations(String num, int... coef) {

System.out.println("Лінійне рекурентне співвідношення для p" + num + "(x):\n" +

"s(i+n)=");

for (int i = coef.length - 1; i >= 0; i--) {

if (coef[i] == 1) {

System.out.print("1\*s(i+" + i + ")");

}

}

}

public static int linearShiftRegister(int[] coefficients, int[] impulseFunction, ArrayList<Integer> sequence, String fileName) {

int[] register = new int[impulseFunction.length];

System.arraycopy(impulseFunction, 0, register, 0, impulseFunction.length);

int period = 0;

try (FileWriter writer = new FileWriter(fileName, true)) {

cleanFile(fileName);

do {

period++;

int tempBit = shiftSequenceOneTact(coefficients, register);

writer.write(String.valueOf(tempBit));

sequence.add(tempBit);

} while (!Arrays.equals(register, impulseFunction));

} catch (IOException ex) {

ex.printStackTrace();

}

return period;

}

private static void cleanFile(String fileName) {

try (FileWriter writer = new FileWriter(fileName, false)) {

} catch (IOException ex) {

ex.printStackTrace();

}

}

public static int shiftSequenceOneTact(int[] coefficients, int[] register) {

int firstBit = register[0];

int generetedBit = 0;

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

if (coefficients[i] != 0) {

generetedBit += register[i];

}

}

for (int i = 0; i < register.length - 1; i++) {

register[i] = register[i + 1];

}

generetedBit = generetedBit % 2;

register[register.length - 1] = generetedBit;

return firstBit;

}

public static void detectPolinomType(int realPeriod, int maxPeriod, String name) {

if (realPeriod == maxPeriod) {

System.out.println(name + " є примітивним поліномом, так як його період дорівнює макс.періоду(T=q^n-1)");

} else if (maxPeriod % realPeriod == 0) {

System.out.println(name + " є непримітивним та незвідним поліномом над даним полем, так як його період кратний q^n-1");

} else {

System.out.println(name + " є непримітивним і звідним поліномом над даним полем");

}

}

public static StringBuffer readFile(String fileName) {

StringBuffer text = new StringBuffer();

FileReader fr;

try {

fr = new FileReader(fileName);

int symbol;

while ((symbol = fr.read()) != -1) {

if (symbol == 48) {

text.append("0");

} else if (symbol == 49) {

text.append("1");

}

}

} catch (Exception e) {

e.printStackTrace();

}

return text;

}

private static int countAmountNgram(StringBuffer fileData, int step) {

return fileData.length() - step + 1;

}

private static void createNgramFrequency(StringBuffer fileData, int n, String version) {

int total = countAmountNgram(fileData, n);

Map<String, Integer> alphabet = new HashMap<>();

for (int i = 0; i < fileData.length() - n; i++) {

String ngram = fileData.substring(i, i + n);

int temp = alphabet.getOrDefault(ngram, 0);

temp++;

alphabet.put(ngram, temp);

}

String filename = "..//розподіл n-грам//" + n + "-gram frequency" + version + ".txt";

try (FileWriter writer = new FileWriter(filename, true)) {

cleanFile(filename);

for (Map.Entry<String, Integer> e : alphabet.entrySet()) {

String frequency = new DecimalFormat("#0.0000000000").format((double) e.getValue() / total);

writer.write("\"" + e.getKey() + "\" = " + frequency + System.getProperty("line.separator"));

}

} catch (IOException ex) {

ex.printStackTrace();

}

}

public static int calculateAutoCor(ArrayList<Integer> sequence, int period, int step) {

int coefficient = 0;

for (int i = 0; i < sequence.size(); i++) {

coefficient += (sequence.get(i) + sequence.get((i + step) % period)) % 2;

}

return coefficient;

}

}