*Національний технічний університет України*

*«Київський політехнічний інститут»*

***Факультет інформатики та обчислювальної техніки***

Лабораторна робота №1

*з курсу "****Теорія ймовірностей****"*

**Тема: “Генератори псевдовипадкових двійкових послідовностей”**

***Виконав:***

*Долинний О.В.*

***Група*** *ІО-31*

***Перевірив:***

*Марковський О.П.*

***Київ - 2014р.***

**Тема: генератори псевдовипадкових двійкових послідовностей.**

**Варіант: Т(р8) 6+ 10000 Л**

**Клас LFSR**

**package** lab\_1;

**import** java.util.Random;

**public** **class** LFSR {

**int**[] factors;

**int** output;

**int**[] polynom;

**int** capacity;

**public** LFSR(**int**[] polynom) {

**this**.factors = **new** **int**[polynom.length];

capacity = **this**.factors.length-1;

Random rand = **new** Random();

**for** (**int** i = 0; i <= capacity; i++) {

**this**.factors[i]=rand.nextInt(2);

}

**int** bufer = **this**.factors[capacity];

**for** (**int** i = capacity; i > 0; i--) {

**if** (polynom[i - 1] == 1) {

**this**.factors[i] = *xor*((**this**.factors[i - 1]), bufer);

}

**else** {

**this**.factors[i] = **this**.factors[i - 1];

}

}

**this**.factors[0] = bufer;

**this**.polynom = polynom;

}

**public** **static** **int** xor(**int** a, **int** b) {

**if** ((a + b) == 2) {

**return** 0;

}

**if**((a + b) == 1) {

**return** 1;

}

**return** 0;

}

**public** **int** generateStep() {

**int** bufer = **this**.factors[capacity];

**for** (**int** i = capacity; i > 0; i--) {

**if** (**this**.polynom[i - 1] == 1) {

**this**.factors[i] = *xor*((**this**.factors[i - 1]), bufer);

} **else** {

**this**.factors[i] = **this**.factors[i - 1];

}

}

**this**.factors[0] = bufer;

output = bufer;

**return** output;

}

}

**Клас Table**

**package** lab\_1;

**import** java.util.ArrayList;

**import** java.util.Random;

**public** **class** Table {

**private** **int**[] table = **new** **int**[256];

**private** **int** position = 0;

**final** **int** NUMBER\_LFSR = 8;

**public** Table() {

**int** number0=0;

**int** number1=0;

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

Random rand = **new** Random();

table[i] = rand.nextInt(2);

**if** (table[i]==0) { number0++;}

**else** { number1++;}

**if** (number0 == table.length / 2) {

**for** ( **int** j=i; j<table.length; j++) {

table[j] = 1;

}

**break**;

}

**if** (number1 == table.length / 2) {

**for** ( **int** j=i; j<table.length; j++) {

table[j] = 0;

}

**break**;

}

}

}

**public** **int** generate(ArrayList<LFSR> lfsr) {

**int** output = 0;

position = 0;

**int**[] number = **new** **int**[NUMBER\_LFSR];

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

number[i] = lfsr.get(i).generateStep();

}

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

**if** (number[i] == 1) {

position = position + (**int**)(Math.*pow*(2, i));

}

}

output = table[position];

**return** output;

}

}

**Клас Tester**

**package** lab\_1;

**public** **class** Tester {

**public** **static** **double** frequencyTest(**int**[] arr) {

**int** output = 0;

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

**if**(arr[i] == 1) {

output++;

}

}

**return** ((**double**) output/arr.length);

}

**public** **static** **double** xorFrequencyTest(**int**[] arr) {

**int** output = 0;

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

**if** ((arr[i] ^ arr[i + 1]) == 1) {

output++;

}

}

**return** ((**double**) output/(arr.length - 1));

}

**public** **static** **int**[] rankTest(**int**[] arr, **int** width){

**if** (width <= 0 || width > 10) {

**throw** **new** IllegalArgumentException("Width must be range [2;10]");

}

**int**[] output = **new** **int**[(**int**)Math.*pow*(2, width)];

**for** (**int** i = 0; i < arr.length - width; i++){

**int** bufer = 0;

**for** (**int** j = 0; j < width; j++) {

bufer = bufer << 1;

bufer = bufer | arr[i + j];

}

output[bufer]++;

}

**return** output;

}

**public** **static** **void** complexityTest(**int**[] arr) {

**final** **int** N = arr.length;

**int**[] b = **new** **int**[N];

**int**[] c = **new** **int**[N];

**int**[] t = **new** **int**[N];

b[0] = 1;

c[0] = 1;

**int** l = 0;

**int** m = -1;

**for** (**int** n = 0; n < N; n++) {

**int** d = 0;

**for** (**int** i = 0; i <= l; i++) {

d ^= c[i] \* arr[n - i];

}

**if** (d == 1) {

System.*arraycopy*(c, 0, t, 0, N);

**int** NiM = n - m;

**for** (**int** j = 0; j < N - NiM; j++) {

c[NiM + j] ^= b[j];

}

**if** (l <= n / 2) {

l = n + 1 - l;

m = n;

System.*arraycopy*(t, 0, b, 0, N);

}

}

}

System.***out***.println(l);

}

}

**Клас Working**

**package** lab\_1;

**import** java.util.ArrayList;

**public** **class** Working {

**public** **static** **void** main(String[] args) {

**int**[] selection = **new** **int** [10000];

//int[] selection1 = {1,0,1,0,1,1,0,0,1,0,0,0,1,1,1};

**int**[] polynom0 = {1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13};

**int**[] polynom1 = {1, 0, 0, 0, 0, 1};

**int**[] polynom2 = {0, 0, 1, 0, 0, 0, 1};

**int**[] polynom3 = {0, 1, 1, 1, 0, 0, 0, 1};

**int**[] polynom4 = {0, 0, 0, 1, 0, 0, 0, 0, 1};

**int**[] polynom5 = {0, 0, 1, 0, 0, 0, 0, 0, 0, 1};

**int**[] polynom6 = {0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1};

**int**[] polynom7 = {1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1};

**int**[] polynom8 = {1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1};

ArrayList<LFSR> lfsr = **new** ArrayList<LFSR>();

lfsr.add(0, **new** LFSR(polynom1));

lfsr.add(1, **new** LFSR(polynom2));

lfsr.add(2, **new** LFSR(polynom3));

lfsr.add(3, **new** LFSR(polynom4));

lfsr.add(4, **new** LFSR(polynom5));

lfsr.add(5, **new** LFSR(polynom6));

lfsr.add(6, **new** LFSR(polynom7));

lfsr.add(7, **new** LFSR(polynom8));

Table table = **new** Table();

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

**int** bufer;

bufer = table.generate(lfsr);

selection[i] = bufer;

System.***out***.print(bufer + ",");

}

System.***out***.println("");

System.***out***.println("Tests: ");

System.***out***.println("1. Frequency: " + Tester.*frequencyTest*(selection));

System.***out***.println("2. XOR: " + Tester.*xorFrequencyTest*(selection));

**int**[] ranks = Tester.*rankTest*(selection, 4);

System.***out***.println("3. Rank: ");

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

System.***out***.println(Integer.*toBinaryString*(i)+" "+(**double**)ranks[i]/\*/selection.length\*/);

}

System.***out***.println("4. Complexity: ");

Tester.*complexityTest*(selection);

}

}