Національний технічний університет України
«Київський політехнічний інститут»
Факультет інформатики і обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №1
З теорії ймовірностей

Виконав:
Студент групи ІО-32
Довгаль Д.С.
Залікова книжка №3211
Перевірив:
Марковський О. П.

м. Київ
2014 р.

1. Варіант завдання:

Тип генератора	Розрядність	Об'єм вибірки	Складність
T(p12)	**6+**	**20000**	**л**

2. Лістинги класів проекту:

```
import java.util.Random;

public class WorkClass {

    private static final long SELECTION = 20000;
    private static final int LFSRNUM = 12;
    private static final int STARTPOINT = 6;
    private static final int WINDOWSIZE = 5;
    private static byte[] randomCode = new byte[(int)SELECTION];

    public static void main(String[] args) throws Exception{

        long time= System.currentTimeMillis();
        byte[] table= new byte[(int)Math.pow(2,LFSRNUM)];
        Lfsr[] raid= new Lfsr[LFSRNUM];


        //create raid
        try{
            for (int i=0;i<LFSRNUM;i++){
                raid[i]= new Lfsr(Polynoms.getPolynom(i+STARTPOINT));
            }
        }catch (Exception e){
            System.out.println("Don't have in database polynom for this LFSR length");
            return;
        }

        //create random table
        Random r= new Random();
        for (int j=0;j<table.length;j++){
            boolean tmp= r.nextBoolean();
            if (tmp) table[j]= 1;
        }

        //create answer array
        for (int i=0;i<SELECTION;i++){
            //get cell num
            long cell=0;
            for (int j=0;j<LFSRNUM;j++){
                if (raid[j].shift()==1) cell+=Math.pow(2,j);
            }

            //get bit from table and put it
            randomCode[i]= table[(int)cell];
        }

        System.out.println("Time for generating: "+
(double)(System.currentTimeMillis()-time)/1000+ " sec\n");

        //-----------Start Testing-----------
        firstTest();
        secondTest();
        thirdTest(WINDOWSIZE);
        System.out.println("\nForth test result of linear complexity: "+ fourthTest()+
" попугаев\n");
        System.out.println("Time for generating and testing: "+
(double)(System.currentTimeMillis()-time)/1000+ " sec");
    }
```

```
    private static void firstTest(){
        int oneNum=0;
        for (int i=0;i< randomCode.length;i++){
            if (randomCode[i]==1) oneNum++;
        }
        System.out.println("First test: "+ oneNum+ "/"+ randomCode.length+ " ("+
(double)oneNum/ randomCode.length+ ")\n");
    }

    private static void secondTest(){
        byte[] tmpArr= new byte[randomCode.length-1];
        for (int i=0;i< randomCode.length-1;i++){
            tmpArr[i]= (byte)(randomCode[i]^ randomCode[i+1]);
        }
        int oneNum=0;
        for (int i=0;i<tmpArr.length;i++){
            if (tmpArr[i]==1) oneNum++;
        }
        System.out.println("Second test: "+ oneNum+ "/"+ tmpArr.length+ " ("+
(double)oneNum/tmpArr.length+ ")\n");
    }

    private static void thirdTest(int windowSize){
        //каждая ячейка этого масива хранит инф. сколько раз повторяется бин код номера
этой ячейки
        int[] tmpArr= new int[(int)Math.pow(2,windowSize)];
        //столько раз будеу ходить окном
        for (int i=0;i< randomCode.length-windowSize+1;i++){
            int cell=0;
            //перевожу код в окне в десятичную
            for (int j=0;j<windowSize;j++){
                if (randomCode[i+j]==1) cell+=Math.pow(2,windowSize-j-1);
            }
            tmpArr[cell]++; //нашел число номера ячейки - там увеличиваю количество
найденых разов
        }
        //идет подсчет и инва черпается из tmpArr
        System.out.println("Third test with window size "+ windowSize+ ":");
        for (int i=0;i<tmpArr.length;i++){
            System.out.println(Integer.toBinaryString(i)+ " - "+ tmpArr[i]+ "/"+
randomCode.length+ " ("+ (double)tmpArr[i]/randomCode.length+ ")");
        }
    }

    private static int fourthTest(){

        int length= randomCode.length;
        int[] BD = new int[length]; //вспомогательный полином
        int[] CD = new int[length]; //полином для L-розрядного LFSR
        int[] t = new int[length];
        BD[0] = 1;
        CD[0] = 1;
        int L = 0; //текущее значение сложности
        int p = -1; //соответствует X

        //номер текущего бита - i
        for (int i = 0; i < length; i++){
            int d = 0;
            for (int j = 0; j <= L; j++) d ^= (CD[j] * randomCode[i - j]); //та
странная фр-ла
            if (d == 1){
                System.arraycopy(CD, 0, t, 0, CD.length);
                int ip = i - p;
                for (int k = 0; k < length - ip; k++) CD[ip + k] ^= BD[k]; //коррекция
полинома
                if (L <= i / 2){
```

```
                    L = i + 1 - L;
                    p = i;
                    System.arraycopy(t, 0, BD, 0, t.length);
                    }
                }
            }
        return L;
        }

}


import java.util.Random;

public class Lfsr {

    private boolean[] srcArr;
    private byte[] polinomical;

    Lfsr(byte[] arr){
        srcArr= new boolean[arr[0]];
        polinomical= arr;

        //create start random info
        Random r= new Random();
        for (int i=0;i<srcArr.length;i++){
            int tmp= r.nextInt();
            if (tmp%2==0) srcArr[i]= true;
        }
    }

    byte shift (){
        //save first
        boolean first= srcArr[0];

        //xor with other cells, that in polynom
        for (int i=1;i<polinomical.length;i++){
            srcArr[srcArr.length-polinomical[i]]^=first;
        }

        //shift and first goes last
        shiftLeft(srcArr);
        srcArr[srcArr.length-1]= first;

        if (first) return 1;
        else  return 0;
    }

    private void shiftLeft(boolean[] srcArr){
        int size= srcArr.length;
        boolean[] tmpArr= srcArr;
        for (int i=0;i<size-1;i++){
            srcArr[i]= tmpArr[i+1];
        }
    }

}


public class Polynoms {
    static byte[] getPolynom(int size)throws Exception{
        byte[] answer={};
        switch (size){
            case 6: answer= new byte[]{6,1};
            break;
            case 7: answer= new byte[]{7,3};
            break;
```

```
            case 8: answer= new byte[]{8,4,3,2};
            break;
            case 9: answer= new byte[]{9,4};
            break;
            case 10: answer= new byte[]{10,3};
            break;
            case 11: answer= new byte[]{11,2};
            break;
            case 12: answer= new byte[]{12,6,4,1};
            break;
            case 13: answer= new byte[]{13,4,3,1};
            break;
            case 14: answer= new byte[]{14,10,6,1};
            break;
            case 15: answer= new byte[]{15,1};
            break;
            case 16: answer= new byte[]{16,12,3,1};
            break;
            case 17: answer= new byte[]{17,3};
            break;
            default: throw new Exception();
        }
        return answer;
    }
}
```