Documentación práctica 2.

Interfaz

- Las sucesiones que servirán de parámetros para todos los métodos (excepto LFSR) se pueden introducir a través de la interfaz, en la casilla correspondiente u opcionalmente en el fichero de entrada 'entrada.txt'. El programa comprobará primero si se encuentran estas en las casillas de la interfaz y si no es así intentará leerlas del fichero propiamente dicho. Las salidas de los métodos se mostrarán en la interfaz.

- Para el método LFSR la semilla y el polinomio de conexión se deben especificar en la interfaz, no en el fichero de entrada. Las salida se escribirán en el fichero de salida 'salida.txt'. En este método se puede especificar el tamaño de la sucesión que se desea obtener especificándolo en la casilla 'Tamaño salida' de la interfaz.

- Para los dos métodos de mezcla: a través de la interfaz solamente se pueden introducir dos sucesiones que deberán tener el mismo tamaño. Si se desea introducir más de dos se debe hacer a través del fichero de entrada 'entrada.txt' poniendo estas en lineas separadas y siendo estas del mismo tamaño.

Constructor

Practica2CRIP

```
public Practica2CRIP()
```

Métodos

is_periodic

```
public static int is_periodic(java.lang.String suc) throws
java.lang.ArrayIndexOutOfBoundsException
```

Método que dada una sucesión de bits, determina si es o no periódica, y en caso afirmativo, determina la longitud del periodo (para que sea periódica, el periodo debe aparecer en la sucesión al menos dos veces).

Parámetros:

suc- Sucesion de bits que se va a comprobar.

Returns:

La longitud del periodo si es periódica o -1 si no lo es.

Throws:

java.lang.ArrayIndexOutOfBoundsException

---

equal

```
public static boolean equal(java.lang.String suc1,java.lang.String suc2)
                    throws java.lang.ArrayIndexOutOfBoundsException
```

Método que dadas dos sucesiones de bits determina si son iguales, es llamado desde 'is_periodic(String suc)'

Parámetros:

suc1- Sucesiòn de bits que se va a comprobar.

suc2- La otra sucesión de bits que se va a comprobar.

Returns:

True si las dos sucesiones son iguales o False si son distintas.

Throws:

java.lang.ArrayIndexOutOfBoundsException

LFSR

```
public static java.lang.String LFSR(java.lang.String semSuc,
                                    java.lang.String pol,
                                    int sizeOutput)
```

Método que construye la sucesión pseudo-aleatoria que generaría un LFSR, con polinomio de conexión p(x), y semilla la sucesión de n bits dada. La salida debe ser guardada en un fichero de texto, y debe contener, al menos, dos veces el periodo.

Parámetros:

semSuc- Semilla, la sucesion de n bits dada

polConex- Polinomio p(x) perteneciente a Z2[x] de grado n, y tal que p(0) = 1

Ejemplo:

$c(D) = D^4 + D^1 + 1$ -> 1001 y s0 = 1, s1 = 0, s2 = 0, s3 = 1 -> 1001 100100011110101 100100011110101 100100011110101.....

---

berlekamp_massey

```
public static java.lang.String berlekamp_massey(int[] array)
```

Algoritmo de Berlekamp-Massey que dada una sucesión de bits periódica, determina la complejidad lineal de dicha sucesión, y el polinomio de conexión que la genera.

Parámetros:

suc- sucesión de bits periódica.

Returns:

un string con la la complejidad lineal y el polinomio de conexión que la genera.

Ejemplo:

$c(D) = D^4 + D^1 + 1$ y s0 = 1, s1 = 0, s2 = 0, s3 = 1

100100011110101 100100011110101 100100011110101.....

$c(D) = D^3 + D^1 + 1$ y s0 = 1, s1 = 0, s2 = 1 1010011 1010011 1010011....

---

mixing_function_AND

```
public static java.lang.String
mixing_function_AND(java.util.List<java.lang.String> listSuc)
```

Método que a partir de dos o mas sucesiones genera una sucesión por medio de la operación AND de las sucesiones dadas.

Parámetros:

listSuc- lista con las sucesiones dadas, debe de haber dos o mas.

Returns:

la nueva sucesión

---

mixing_function_XOR

```
public static java.lang.String
mixing_function_XOR(java.util.List<java.lang.String> listSuc)
```

Método que a partir de dos o mas sucesiones genera una sucesión por medio de la operación XOR de las sucesiones dadas.

Parámetros:

listSuc- lista con las sucesiones dadas, debe de haber dos o mas.

Returns:

la nueva sucesión

---

reader

```
public static java.io.BufferedReader reader()
                                    throws java.io.IOException
```

Método usado para las entradas desde ficheros de texto

Returns:
un objeto BufferedReader el cual sera leido desde el metodo que llame a este o null si el fichero esta vacio

Throws:
java.io.IOException

---

writer

```
public static void writer(java.lang.String sucesión)
                  throws java.io.IOException
```

Método usado para las salidas a ficheros de texto

Parámetros:
sucesión - la secuencia de unos y ceros que se escribe en el fichero

Throws:
java.io.IOException

---

main

```
public static void main(java.lang.String[] args)
```

Parámetros:
args- the command line arguments