
Teoría computacional

- Lenguajes -

Práctica 1

2CM5/Vargas Romero Erick Efraín
Prof. Juárez Martínez Genaro

Instituto Politécnico Nacional
Escuela Superior de Cómputo
Juan de Dios Bátiz, nueva industrial Vallejo
07738 ciudad de México

0.1 Creación de un universo

Si Σ es un alfabeto podemos, expresar el conjunto de todas las cadenas de una determinada longitud de dicho alfabeto utilizando una notación exponencial. Definimos Σ^k para que sea el conjunto de las cadenas de longitud k , tales que cada uno de los símbolos de las mismas pertenece a Σ .

Siendo así debemos de tener en cuenta que $\Sigma^0 = \epsilon$ esto ocurre en cualquier alfabeto, en otras palabras ϵ es la única cadena cuya longitud es 0.

Siendo así $\Sigma = \{0,1\}$ en nuestro caso este conjunto es nuestro alfabeto, entonces si $\Sigma^1 = \{0,1\}$, si continuamos con este mismo orden observamos que $\Sigma^2 = \{00,01,10,11\}$ y si continuásemos $\Sigma^3 = \{000,001,010,011,100,101,110,111\}$ y así sucesivamente.

Siendo así se procede a realizar un programa que genere un "universo" de diversas cadenas a partir del lenguaje previamente descrito el cual es $\Sigma = \{0,1\}$ siendo así el usuario ingresará el valor de k y el programa irá concatenando las diversas cadenas que genere el programa a partir de nuestro alfabeto, por ejemplo $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots$ donde Σ^* es el conjunto de todas las cadenas de un alfabeto. También podemos ver lo antes mencionado de la siguiente manera:

$\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, \dots\}$

Finalmente, todas las cadenas que se generen mediante este programa serán guardadas en un archivo de texto, además, el programa mostrará una gráfica con la cantidad de unos que se han generado desde $k=0$ hasta $k=20$

0.1.1 Pruebas

Al ejecutar el programa podemos observar nada más y nada menos que un pequeño menú el cual nos permite elegir entre alguna de las opciones que tenemos para ejecutar este programa, ya sea automático o manual, siendo así al elegir el modo automático, un usuario ingresará el valor de k que para nuestra primer prueba se ha ingresado un valor de $k=23$.

Una vez hecho lo anterior el programa realizará las diferentes combinaciones que se generen dependiendo del valor de k , posteriormente se muestra en consola si se ha guardado la información en un fichero.

```
> > >Menu< < <
Elige una opcion
1. Manual
2. Automatico
3. Salir
1
Escribe un numero entre 1 y 1,000|
23
Todos los numeros se han guardado en Universe.txt
```

Figure 1: Prueba en manual

Como ya se mencionó anteriormente, el programa puede gráficar hasta que k tenga un valor de 20 ($k=20$) en la figura 2 podemos ver el gráfico, y además que su comportamiento es exponencial.

[illegible]

Una vez mostrado lo anterior podemos proceder a mostrar el modo automático del programa, este modo tiene un funcionamiento muy sencillo, solamente se hace un random

para obtener un número que será el que utilizaremos para generar las cadenas en base a nuestro alfabeto, para esta prueba se ha establecido con un random cuyo máximo valor es 25

Como podemos observar en la figura 5, es similar a el menú antes mostrado en la figura 1, pero muestra el número que ha elegido el sistema en base al random, para este caso el número elegido fue 22.

Además al igual que el programa en manual, muestra la gráfica con la cantidad de unos que hay hasta $k=20$ y también genera un fichero el cual tiene almacenado todas l

Finalmente en el caso de este programa mostraremos el código que se ha creado, primeramente el de nuestra clase llamada MymMain

```
> > >Menu< < <
Elige una opcion

1. Manual
2. Automatico
3. Salir
2
El sistema eligio el numero 22
Todos los numeros se han guardado en Universe.txt
```

Figure 5: Prueba en automático

```

1 package universe;
2
3 import Graph.Graph;
4 import java.io.FileWriter;
5 import java.math.BigInteger;
6 import java.util.Random;
7 import java.util.Scanner;
8 import javax.swing.JFrame;
9 import javax.swing.JLabel;
10 import javax.swing.JPanel;
11 import org.jfree.data.xy.XYSeries;
12
13 public class MyMain {
14
15     void display() {
16         System.out.println("\n\n\n\n\n\n\n\n\n\n\n> >>Menu< <<\n");
17         System.out.println("Elige una opcion\n");
18
19         System.out.println("1. Manual");
20         System.out.println("2. Automatico");
21         System.out.println("3. Salir");
22
23         switch (new Scanner(System.in).nextInt()) {
24             //Manual option
25             case 1:
26                 manual(new BigInteger("1000"));
27                 break;
28             //Automatic option
29             case 2:
30                 automatic(new BigInteger("1000"));
31                 break;
32             case 3:
33                 break;
34             default:
35                 System.out.println("Esa opcion no existe.");
36                 display();
37                 break;
38         }

```

```

39 }
40
41 void manual(BigInteger limit) {
42     System.out.println("Escribe un numero entre 1 y 1,000");
43     XYSeries oneSeries = new XYSeries("Unos");
44     try {
45         FileWriter fileWriter = new FileWriter("Universe.txt");
46         fileWriter.write("\u03A3 = {\u03B5}");
47         int size = new Scanner(System.in).nextInt();
48         if (size > 0 && size <= 1000) {
49             for (int i = 1; i < size + 1; i++) {
50                 Number number = new Matrix().fillMatrix(i);
51                 new Files().saveUniverse(number, fileWriter);
52                 if (i <= 20)
53                     oneSeries.add(i, number.getOnes());
54                 if (i == 20 && size >= 20)
55                     showGraph(oneSeries);
56                 if (size < 20 && i == size)
57                     showGraph(oneSeries);
58             }
59             fileWriter.write("}");
60             fileWriter.close();
61             System.out.println("Guardado en Universe.txt");
62         } else {
63             System.out.println("Fuera de rango");
64             manual(limit);
65         }
66     } catch (Exception e) {
67         System.err.println("Error! " + e);
68     } finally {
69
70         display();
71     }
72 }
73
74 void automatic(BigInteger limit) {
75     try {
76         XYSeries oneSeries = new XYSeries("Unos");
77         FileWriter fileWriter = new FileWriter("Universe.txt");
78         fileWriter.write("\u03A3 = {\u03B5}");
79         int potence = new Random().nextInt(25);
80         System.out.println("El sistema eligio el numero "
81 + (potence + 1));
82         for (int i = 1; i < potence + 2; i++) {
83             Number number = new Matrix().fillMatrix(i);
84             new Files().saveUniverse(number, fileWriter);
85             if (i <= 20)
86                 oneSeries.add(i, number.getOnes());
87             if (i == 20 && potence + 1 >= 20)
88                 showGraph(oneSeries);
89             if (potence + 1 < 20 && i == potence + 1)
90                 showGraph(oneSeries);
91         }
92         fileWriter.write("}");
93         fileWriter.close();
94         System.out.println("Guardado en Universe.txt");
95     } catch (Exception e) {
96         System.err.println("Error! " + e);
97     } finally {
98         display();
99     }

```

```

100     }
101
102     void showGraph(XYSeries onesSerie) {
103         JPanel jpanel;
104         JLabel jlabel;
105         JFrame jframe;
106         jpanel = new JPanel();
107         jlabel = new JLabel();
108         jframe = new JFrame();
109         jframe.setSize(1300, 700);
110         jpanel.setSize(1300, 700);
111         jlabel.setIcon(new Graph(jpanel.getSize(), onesSerie));
112         jlabel.setText("");
113         jpanel.add(jlabel);
114         jframe.add(jpanel);
115         jframe.setVisible(true);
116     }
117 }

```

Ahora se muestra el código contenido en la clase Matrix

```

1 package universe;
2
3 import java.math.BigInteger;
4 import java.util.ArrayList;
5
6 public class Matrix {
7
8     BigInteger ones = new BigInteger("0");
9     //This function receive the number of columns
10    //If we raise to power two we get the number of rows
11    Number fillMatrix(int k) {
12        BigInteger base = new BigInteger("2");
13        BigInteger alternance = new BigInteger("0");
14        ArrayList<ArrayList<String>> matrix = new ArrayList<>();
15        //Here I go over all the columns one by one
16        //Mathematically our columns is represented by k word
17        //Also our number of rows is get with 2^k or Math.pow(2,k);
18        //The alternance of each column is get with 2^k-1
19        //Remember that I go over each column from right to left <- <-
20        for (BigInteger columns = new BigInteger("1");
21             columns.compareTo(new BigInteger(Integer.toString(k + 1))) ==
22                 -1;
23             columns = columns.add(BigInteger.ONE)) {
24            //Here I go over each row the number of row ever is the
25            //maximum number of combinations Math.pow(2,k)
26            //Mathematically is 2^k
27            ArrayList<String> column = new ArrayList<>();
28            for (BigInteger rows = new BigInteger("0");
29                 rows.compareTo(base.pow(k)) == -1;
30                 rows = rows.add(BigInteger.ONE)) {
31                if (alternance.compareTo(base.pow(columns.subtract(new
32                    BigInteger("1")).intValueExact())) == -1) {
33                    alternance = alternance.add(BigInteger.ONE);
34                    column.add("0");
35                } else if ((alternance.compareTo(base.pow(columns.subtract(
36                    new BigInteger("1")).intValueExact())) == 0)
37                    || alternance.compareTo(base.pow(columns.subtract(new
38                        BigInteger("1")).intValueExact())) == 1) {
39                    alternance = alternance.add(BigInteger.ONE);
40                    column.add("1");

```

```

37         if (alternance.compareTo(base.pow(columns.intValue())) ==
38             0) {
39             alternance = new BigInteger("0");
40         }
41         ones = ones.add(BigInteger.ONE);
42     }
43     //Here I add a column in our ArrayList of ArrayList
44     matrix.add(column);
45 }
46 return new Number(matrix, ones);
47 }
48 }

```

Aquí tenemos el código de una clase llamada Number.

```

1 package universe;
2
3 import java.math.BigInteger;
4 import java.util.ArrayList;
5
6 public class Number {
7
8     private ArrayList<ArrayList<String>> table;
9     private BigInteger ones;
10
11     public Number(ArrayList<ArrayList<String>> table, BigInteger ones){
12         this.table = table;
13         this.ones = ones;
14     }
15
16     public ArrayList<ArrayList<String>> getTable(){
17         return this.table;
18     }
19
20     public BigInteger getOnes(){
21         return this.ones;
22     }
23
24 }

```

Ahora se muestra la clase llamada Files, con la cual escribimos en un fichero las cadenas que se han generado anteriormente.

```

1 package universe;
2
3 import java.io.FileWriter;
4 import java.math.BigInteger;
5
6 public class Files {
7
8     void saveUniverse(Number number, FileWriter fileWriter) throws Exception{
9
10         try{
11             for(BigInteger i = new BigInteger("0");
12                 i.compareTo(new BigInteger(String.valueOf(number.getTable
13                     ().get(0).size()))) == -1;
14                 i = i.add(BigInteger.ONE)){
15                 fileWriter.write(", ");
16                 for(BigInteger j = new BigInteger("0");
17                     j.compareTo(new BigInteger(String.valueOf(number.
18                         getTable().size()))) == -1;

```



```

17         j = j.add(BigInteger.ONE)){
18             String output = number.getTable().get(j.intValueExact
               ()).get(i.intValueExact());
19             fileWriter.write(output);
20         }
21     }
22 } catch (Exception e) {
23     System.err.println("Ha ocurrido un error: " + e);
24 }
25 }
26 }

```

Finalmente, la clase principal donde tenemos contenido al método main

```

1 package universe;
2
3 public class Universe {
4
5     public static void main(String[] args) {
6         new MyMain().display();
7     }
8 }

```

0.2 Definición de lenguajes mediante descripciones de conjuntos

La descripción de este programa es muy simple, hay que generar un conjunto de números binarios cuyo valores un número primo, como ejemplo tenemos el siguiente conjunto {10,11,101,111,1011,...} respectivamente en decimal es {2,3,5,7,11,...}

0.2.1 Pruebas

Una vez dicho lo anterior procederemos con las pruebas de este programa, el cual es prácticamente idéntico al programa anterior en cuanto al menú que se ha creado. En el menú se elige el modo en el cual se desea ejecutar el programa, sea automático (haciendo un random) o manual. Para la primer prueba se eligió el modo manual, ingresando el valor máximo permitido en el programa que es de 100,000 tal y como se muestra en la Figura 6. Además se muestra un mensaje para confirmar si se ha guardado o no en el fichero el conjunto de números primos generado.

```

> > >Menu< < <

Elige una opcion

1. Manual
2. Automatico
3. Salir
1
Escribe un numero entre 1 y 100,000
100000
guardado en Primes.txt

```

Figure 6: Prueba en manual

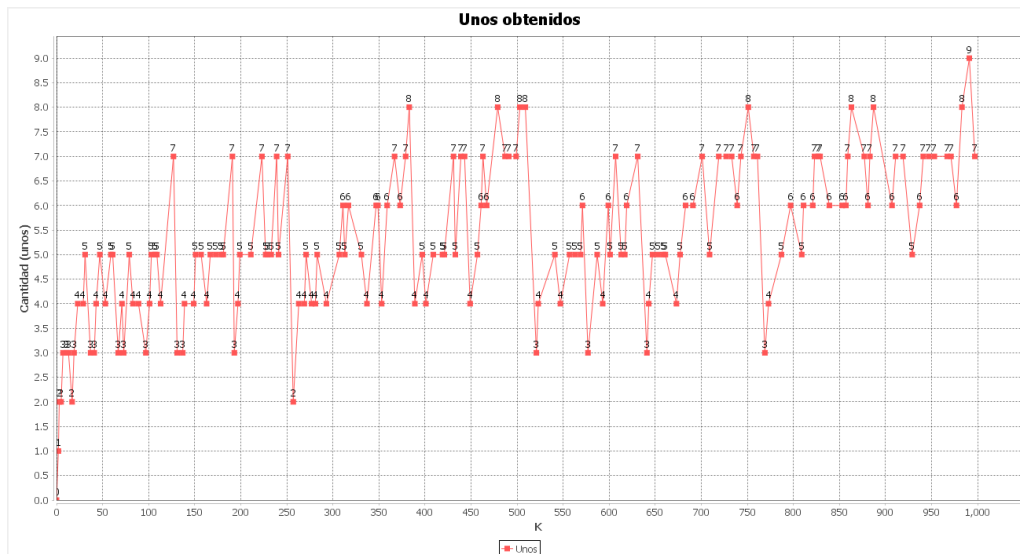


Figure 7: Gráfica generada

Como podemos observar en la Figura 7, también se ha generado un gráfico mostrando la cantidad de unos existentes en cada cadena hasta el número 1,000.


 Primes 19/08/2017 11:08 ... Documento de tex... 3 KB

Figure 8: Fichero generado

En la Figura 8 al igual que en la primer prueba es mostrado el fichero generado.

```

K = {10, 11, 101, 111, 1011, 1101, 10001, 10011, 10111, 11101, 11111, 100101, 101001, 1010
1001, 1000111011, 1001000001, 1001001011, 1001010001, 1001010111, 1001011001, 1001011111, :
0011, 10001101001, 10001111111, 10010000001, 10010001011, 10010010011, 10010011101, 100101
010101, 11010111001, 11010111011, 11011000101, 11011001101, 11011010011, 11011011001, 110
00101, 100100000111, 100100011101, 100100100011, 100100100101, 100100101011, 10010010111,
101, 101100111111, 101101000111, 101101010001, 101101010111, 101101011101, 101101100101, 1
1, 110110110111, 110110111101, 110111000111, 110111001001, 110111001101, 110111010011, 110
11111111011, 111111111101, 1000000000011, 1000000001111, 1000000011111, 1000000100001, 10
0110111, 1001001000001, 1001001000111, 1001001010011, 1001001011111, 1001001110001, 100100
001, 1010010011111, 1010010100001, 1010010110001, 1010010110111, 1010010111101, 1010011001
1011011011001, 1011011011011, 1011011100001, 1011011100101, 1011011101011, 1011011101101, :
100011011, 1100100110001, 1100100110011, 1100101000101, 1100101001001, 1100101010001, 1100
10011, 1101101111111, 1101110000011, 1101110010001, 1101110011101, 1101110100111, 11011101
1, 1110111111001, 1111000000001, 1111000000111, 1111000001011, 1111000010011, 111100001011
001100011, 10000001100101, 10000001101001, 10000001110111, 10000001111101, 10000010001001,
010011101, 10001010011111, 10001010100011, 10001010110111, 10001010111101, 10001011011011,
0110111001, 10010011011101, 10010011011111, 10010011110101, 10010011110111, 10010011110111,
100110101 10011100110111 10011101001101 100111010011 10011101010101 10011101011111

```

Figure 9: Contenido del fichero (Binario)

Finalmente, en la figura 9 podemos ver el contenido del fichero (En binario)

```

Σ = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83,
, 1249, 1259, 1277, 1279, 1283, 1289, 1291, 1297, 1301, 1303, 1307, 1319, 1321, 1327, 1361, 1
51, 2557, 2579, 2591, 2593, 2609, 2617, 2621, 2633, 2647, 2657, 2659, 2663, 2671, 2677, 2683,
3929, 3931, 3943, 3947, 3967, 3989, 4001, 4003, 4007, 4013, 4019, 4021, 4027, 4049, 4051, 405
, 5419, 5431, 5437, 5441, 5443, 5449, 5471, 5477, 5479, 5483, 5501, 5503, 5507, 5519, 5521, 5
83, 6899, 6907, 6911, 6917, 6947, 6949, 6959, 6961, 6967, 6971, 6977, 6983, 6991, 6997, 7001,
8447, 8461, 8467, 8501, 8513, 8521, 8527, 8537, 8539, 8543, 8563, 8573, 8581, 8597, 8599, 860
, 9967, 9973, 10007, 10009, 10037, 10039, 10061, 10067, 10069, 10079, 10091, 10093, 10099, 10
383, 11393, 11399, 11411, 11423, 11437, 11443, 11447, 11467, 11471, 11483, 11489, 11491, 1149
9, 12743, 12757, 12763, 12781, 12791, 12799, 12809, 12821, 12823, 12829, 12841, 12853, 12889,
14159, 14173, 14177, 14197, 14207, 14221, 14243, 14249, 14251, 14281, 14293, 14303, 14321, 14
541, 15551, 15559, 15569, 15581, 15583, 15601, 15607, 15619, 15629, 15641, 15643, 15647, 1564
7, 16993, 17011, 17021, 17027, 17029, 17033, 17041, 17047, 17053, 17077, 17093, 17099, 17107,
18379, 18397, 18401, 18413, 18427, 18433, 18439, 18443, 18451, 18457, 18461, 18481, 18493, 18
919, 19927, 19937, 19949, 19961, 19963, 19973, 19979, 19991, 19993, 19997, 20011, 20021, 2002
7, 21379, 21383, 21391, 21397, 21401, 21407, 21419, 21433, 21467, 21481, 21487, 21491, 21493,
22777, 22783, 22787, 22807, 22811, 22817, 22853, 22859, 22861, 22871, 22877, 22901, 22907, 22
179, 24181, 24197, 24203, 24223, 24229, 24239, 24247, 24251, 24281, 24317, 24329, 24337, 2435
3, 25771, 25793, 25799, 25801, 25819, 25841, 25847, 25849, 25867, 25873, 25889, 25903, 25913,
27259, 27271, 27277, 27281, 27283, 27299, 27329, 27337, 27361, 27367, 27397, 27407, 27409, 27
729, 28751, 28753, 28759, 28771, 28789, 28793, 28807, 28813, 28817, 28837, 28843, 28859, 2886
7, 30313, 30319, 30323, 30341, 30347, 30367, 30389, 30391, 30403, 30427, 30431, 30449, 30467,

```

Figure 10: Contenido del fichero (Decimal)

Y en la Figura 10 se muestra el mismo fichero pero visto en su parte decimal. Ahora

```

> > >Menu< < <

Elige una opcion

1. Manual
2. Automatico
3. Salir
1
Escribe un numero entre 1 y 100,000
100000
guardado en Primes.txt

```

Figure 11: Prueba en automático

se procede de la misma forma a trabajar en el modo automático, tal y como se muestra en la Figura 11. El resto de lo ocurrido es idéntico a la prueba en automático, desde mostrar la gráfica hasta la generación del fichero que contiene nuestras cadenas en binario y decimales.

Finalmente, se mostrará el código utilizado en este programa, del cual tenemos primeramente el código de la clase llamada MyMain

```

1 package prime . numbers ;
2
3 import Graph . Graph ;
4 import java . io . FileWriter ;
5 import java . math . BigInteger ;
6 import java . util . Random ;
7 import java . util . Scanner ;
8 import javax . swing . JFrame ;
9 import javax . swing . JLabel ;
10 import javax . swing . JPanel ;
11 import org . jfree . data . xy . XYSeries ;
12
13 public class MyMain {
14
15     void display () {
16         System.out.println("\n\n\n\n\n\n\n\n\n\n\n> >>Menu< <<\n");
17         System.out.println("Elige una opcion\n");
18
19         System.out.println("1. Manual");
20         System.out.println("2. Automatico");
21         System.out.println("3. Salir");
22
23         switch (new Scanner(System.in).nextInt()) {
24             //Manual option
25             case 1:
26                 manual(new BigInteger("100000"));
27                 break;
28             //Automatic option
29             case 2:
30                 automatic(new BigInteger("100000"));
31                 break;
32             case 3:
33                 break;
34             default :
35                 System.out.println("Esa opcion no existe.");
36                 display ();
37                 break;
38         }
39     }
40
41     void manual(BigInteger limit) {
42         String naturalNumbers = "\u03A3 = {" ;
43         System.out.println("Escribe un numero entre 1 y 100,000");
44         XYSeries oneSeries = new XYSeries("Unos");
45         oneSeries.add(0,0);
46         try {
47             FileWriter fileWriter = new FileWriter("Primes.txt");
48             fileWriter.write("\u03A3 = {" );
49             int size = new Scanner(System.in).nextInt();
50             if (size > 0 && size <= 100000) {
51                 for (int i = 0; i < size; i++) {
52                     Number number = new Number(new BigInteger(String.valueOf(i)));
53                     if(number.getIsPrime()){
54                         if(i!=2){
55                             naturalNumbers = naturalNumbers + ", ";
56                             fileWriter.write(", ");
57                         }
58                         fileWriter.write(number.getBinary());
59                         if (i <= 1000)

```

```

60         oneSeries.add(i, number.getOnes());
61         naturalNumbers = naturalNumbers + number.getNumber();
62     }
63 }
64 showGraph(oneSeries);
65 fileWriter.write("}");
66 fileWriter.write("\r\n");
67 fileWriter.write("\r\n");
68 fileWriter.write(naturalNumbers);
69 fileWriter.write("}");
70 fileWriter.close();
71 System.out.println("guardado en Primes.txt");
72 } else {
73     System.out.println("Fuera de rango establecido");
74     manual(limit);
75 }
76 } catch (Exception e) {
77     System.err.println("Error! " + e);
78 } finally {
79
80     display();
81 }
82 }
83
84 void automatic(BigInteger limit) {
85     try {
86         String naturalNumbers = "\u03A3 = {";
87         XYSeries oneSeries = new XYSeries("Unos");
88         oneSeries.add(0,0);
89         FileWriter fileWriter = new FileWriter("Primes.txt");
90         fileWriter.write("\u03A3 = {");
91         int potence = new Random().nextInt(limit.intValueExact());
92         System.out.println("El sistema eligio el numero " + (potence + 1)
93             );
94         for (int i = 0; i < potence + 2; i++) {
95             Number number = new Number(new BigInteger(String.valueOf(
96                 i)));
97             if(number.getIsPrime()){
98                 if(i!=2){
99                     naturalNumbers = naturalNumbers + ", ";
100                     fileWriter.write(", ");
101                 }
102                 fileWriter.write(number.getBinary());
103                 if (i <= 1000)
104                     oneSeries.add(i, number.getOnes());
105                 naturalNumbers = naturalNumbers + number.getNumber();
106             }
107             showGraph(oneSeries);
108             fileWriter.write("}");
109             fileWriter.write("\r\n");
110             fileWriter.write("\r\n");
111             fileWriter.write(naturalNumbers);
112             fileWriter.write("}");
113             fileWriter.close();
114             System.out.println("Guardado en Primes.txt");
115         } catch (Exception e) {
116             System.err.println("Error! " + e);
117         } finally {
118             display();
119         }
120     }
121 }

```

```

119     }
120
121     void showGraph(XYSeries onesSerie) {
122         JPanel jpanel;
123         JLabel jlabel;
124         JFrame jframe;
125         jpanel = new JPanel();
126         jlabel = new JLabel();
127         jframe = new JFrame();
128         jframe.setSize(1300, 700);
129         jpanel.setSize(1300, 700);
130         jlabel.setIcon(new Graph(jpanel.getSize(), onesSerie));
131         jlabel.setText("");
132         jpanel.add(jlabel);
133         jframe.add(jpanel);
134         jframe.setVisible(true);
135     }
136
137 }

```

Ahora tenemos la clase llamada Graph, la cual se utilizó para crear nuestra gráfica generada a partir de la cantidad de unos que hay por cadena.

```

1 package Graph;
2
3 import java.awt.Color;
4 import java.awt.Dimension;
5 import java.awt.image.BufferedImage;
6 import javax.swing.ImageIcon;
7 import org.jfree.chart.ChartFactory;
8 import org.jfree.chart.JFreeChart;
9 import org.jfree.chart.labels.StandardXYItemLabelGenerator;
10 import org.jfree.chart.labels.XYItemLabelGenerator;
11 import org.jfree.chart.plot.PlotOrientation;
12 import org.jfree.chart.plot.XYPlot;
13 import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
14 import org.jfree.data.xy.XYDataset;
15 import org.jfree.data.xy.XYSeries;
16 import org.jfree.data.xy.XYSeriesCollection;
17
18 public class Graph extends ImageIcon {
19
20     public Graph(Dimension d, XYSeries xyseries) {
21         XYDataset xydataset = xyDataset(xyseries);
22         JFreeChart jfreechart = ChartFactory.createXYLineChart(
23             "Unos obtenidos", "K", "Cantidad (unos)",
24             xydataset, PlotOrientation.VERTICAL, true, true, false);
25         XYPlot xyplot = (XYPlot) jfreechart.getPlot();
26         xyplot.setBackgroundPaint(Color.white);
27         xyplot.setDomainGridlinePaint(Color.BLACK);
28         xyplot.setRangeGridlinePaint(Color.BLACK);
29         XYLineAndShapeRenderer xylineandshaperenderer = (
30             XYLineAndShapeRenderer) xyplot.getRenderer();
31         xylineandshaperenderer.setBaseShapesVisible(true);
32         XYItemLabelGenerator xy = new StandardXYItemLabelGenerator();
33         xylineandshaperenderer.setBaseItemLabelGenerator(xy);
34         xylineandshaperenderer.setBaseItemLabelsVisible(true);
35         xylineandshaperenderer.setBaseLinesVisible(true);
36         xylineandshaperenderer.setBaseItemLabelsVisible(true);
37         BufferedImage bufferedImage = jfreechart.createBufferedImage(d.width,
38             d.height);

```

```

37         this.setImage(bufferedImage);
38     }
39
40     private XYDataset xyDataset(XYSeries xyseries)
41     {
42
43         XYSeriesCollection xyseriescollection = new XYSeriesCollection();
44         xyseriescollection.addSeries(xyseries);
45         return xyseriescollection;
46     }
47
48 }

```

Otra clase utilizada y la cual es parte fundamental es la llamada Number, la cual contiene diversos métodos setters.

```

1 package prime.numbers;
2
3 import java.math.BigInteger;
4
5 public class Number {
6
7     private BigInteger number, ones;
8     private boolean prime;
9     private String binary;
10
11     public Number(BigInteger number){
12         this.number = number;
13         this.prime = isPrime(number);
14         this.binary = number.toString(2);
15         this.ones = countOnes(binary);
16     }
17
18     public BigInteger getNumber(){
19         return this.number;
20     }
21
22     public boolean getIsPrime(){
23         return this.prime;
24     }
25
26     public String getBinary(){
27         return this.binary;
28     }
29
30     public BigInteger getOnes(){
31         return ones;
32     }
33
34     private boolean isPrime(BigInteger number){
35         BigInteger count = new BigInteger("0");
36         for(int i = 1; i<= number.intValueExact(); i++){
37             if(number.intValueExact() % i == 0)
38                 count = count.add(BigInteger.ONE);
39         }
40         if(count.intValueExact() != 2)
41             return false;
42         else return true;
43     }
44
45     private BigInteger countOnes(String binaryString){

```

```
46         BigInteger numberOnes = new BigInteger("0");
47         for (int i=0; i<binaryString.length(); i++){
48             if (binaryString.charAt(i) == '1')
49                 numberOnes = numberOnes.add(BigInteger.ONE);
50         }
51         return numberOnes;
52     }
53
54 }
```

Finalmente, pero no menos importante, esta la clase PrimeNumbers, que solamente contiene al método main.

```
1 package prime.numbers;
2
3 public class PrimeNumbers {
4
5     public static void main(String[] args) {
6         new MyMain().display();
7     }
8
9 }
```