

Instituto Politécnico Nacional
Escuela Superior de Cómputo

Teoría Computacional

Programa 01:
Universo de las cadenas binarias Σ^*

Alumna: Solis Hernández Viridiana

Fecha de entrega:
31/10/2020

Juárez Martínez Genaro

Introducción

Universo de cadenas

Sea Σ un alfabeto, se define Σ^* como el conjunto de todas las cadenas de Σ donde:

Alfabeto

Conjunto finito de elementos (un elemento de un alfabeto es llamado símbolo).

$$\Sigma = \{a,b,c\} \qquad \Sigma = \{0,1\}$$

Cadena

Secuencia o concatenación de símbolos de Σ

$$\Sigma = \{a,b,c\} \quad aabb, bc, cba$$

$$\Sigma = \{0,1\} \quad \varepsilon, 0, 1, 01, 110, 111 \text{ (donde } \varepsilon \text{ es una cadena que no tiene símbolos)}$$

Longitud de Cadena

Si $W = W_1 W_2 \dots W_n$ es una cadena de Σ se define la longitud de W como:

$$|W| = n \quad |01| = 2$$

Sea Σ un alfabeto y $k \in \mathbb{N} \cup \{0\}$

Σ^k es el conjunto de cadenas de longitud k

Sea $\Sigma = \{0,1\}$ un alfabeto $\Sigma^0 = \{\varepsilon\}$ $\Sigma^1 = \{0,1\}$ $\Sigma^2 = \{0,1,00,01,10,11\}$

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots$$

Planteamiento del problema

Se requiere un programa que tenga como salida el universo de las cadenas binarias Σ^k . Dada una "k" que introduzca el usuario o que el programa lo determine automáticamente. El rango de "k" debe de estar en el intervalo de $[0,1000]$.

1. El programa debe de preguntar si quiere calcular otra "k" o no.
2. La salida, expresada en notación de conjunto, debe ir a un archivo de texto.
3. Una segunda salida (archivo de texto) debe concatenar todas las cadenas calculadas en una sola cadena, quitar las llaves, comas y cualquier otro símbolo que no sean 0s y 1s.
4. Del primer archivo de salida, graficar el número de 1s de cada cadena. El eje de las x es la cadena y el eje de las y el número de 1s que tiene esa cadena. Específicamente, calcular y graficar cuando $k=23$. Al mismo tiempo, calcular la gráfica pero calculando su logaritmo en base 2 y 10 respectivamente.

5. Del segundo archivo de salida, particionar la cadena en subcadenas de longitud 32 y graficar la cantidad de unos de esas subcadenas. Al mismo tiempo, calcular la gráfica pero calculando su logaritmo en base 2.

Implementación

Se muestra un menú el cual tendrá como opciones:

1. Insertar valor de k: el usuario ingresa un entero k que se encuentren el intervalo [0,1000].
2. Generar valor aleatorio de k: el programa utiliza la función rand() para generar un número entero aleatorio k que se encuentre en el intervalo [0,1000].
3. Salir: finaliza el programa.

(En caso de no seleccionar una opción valida el programa volverá a solicitar una opción).

Una vez obtenido un número entero k ya sea de manera aleatoria o no, se crean dos matrices dinámicas, cada una tendrá como tamaño $2^k/2$ filas y k columnas.

Se genera la mitad de la tabla de verdad de 2^k (por método de la división) y se guarda en matriz1 mientras la negación se guarda en la matriz2.(Figura1)

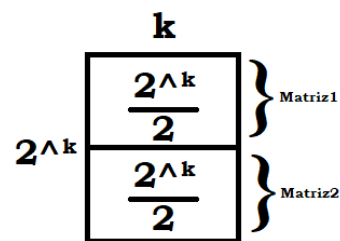


Figura1.

Se abre un primer archivo de texto donde imprimiremos Σ^* en notación de conjunto al mismo tiempo se abre un archivo extra donde se mostrará el número de unos por cadena (el cual nos ayudará para graficar).

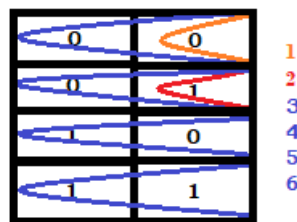


Figura2.

(Así es como se fueron imprimiendo las cadenas en el archivo utilizando ciclos for).

Se abre un segundo archivo de texto donde imprimiremos la concatenación de todas las cadenas del primer archivo, al mismo tiempo se particiona la cadena del segundo archivo en cadenas de longitud 32 y se abre un segundo archivo extra donde mostrara el número de unos por cadena (el cual nos ayudara para graficar).

Una vez terminado el cálculo muestra un segundo menú el cual tiene como opciones:

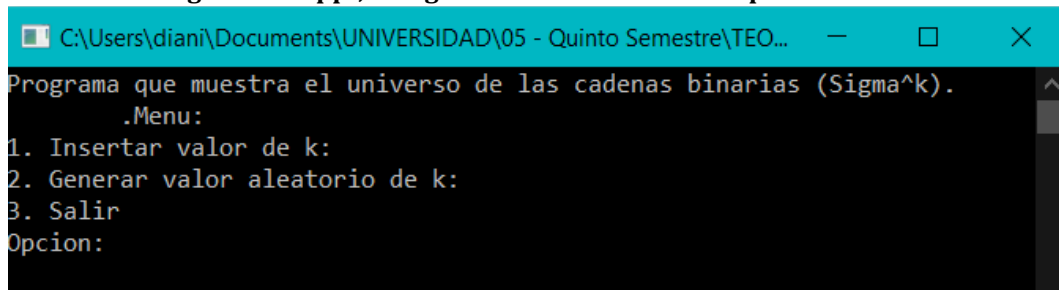
1. Ingresar una nueva k: lleva al menú principal.
2. Salir: finaliza el programa.

Se utilizó el software GNUplot para graficar lo solicitado.

4

Funcionamiento

Ejecución de "Programa01.cpp", "Programa01.exe" Menú Principal

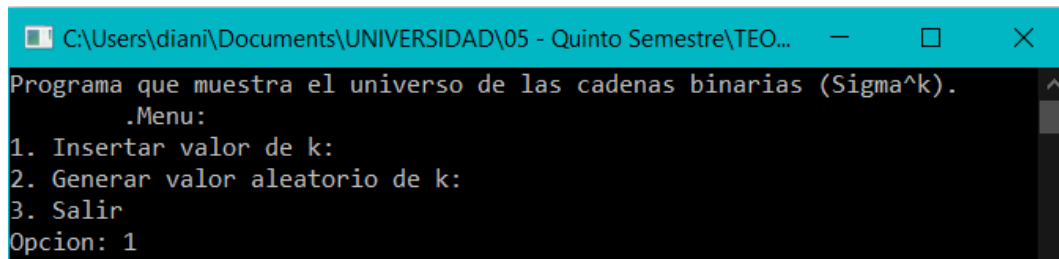


```
C:\Users\diani\Documents\UNIVERSIDAD\05 - Quinto Semestre\TEO...
Programa que muestra el universo de las cadenas binarias (Sigma^k).
.Menu:
1. Insertar valor de k:
2. Generar valor aleatorio de k:
3. Salir
Opcion:
```

Figura3.

PRUEBA [1]

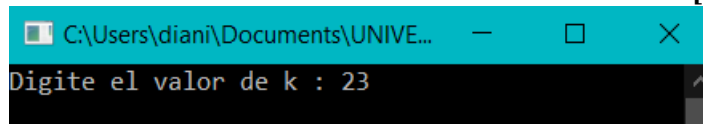
El usuario selecciona la Opción 1 del Menú Principal.



```
C:\Users\diani\Documents\UNIVERSIDAD\05 - Quinto Semestre\TEO...
Programa que muestra el universo de las cadenas binarias (Sigma^k).
.Menu:
1. Insertar valor de k:
2. Generar valor aleatorio de k:
3. Salir
Opcion: 1
```

Figura4.

El usuario ingresa un entero k = 23 el cual se encuentra en el intervalo [0,1000]



```
C:\Users\diani\Documents\UNIVE...
Digite el valor de k : 23
```

Figura5.

Menú Secundario

Una vez finalizados todos los cálculos con $k = 23$ aparece el menú secundario

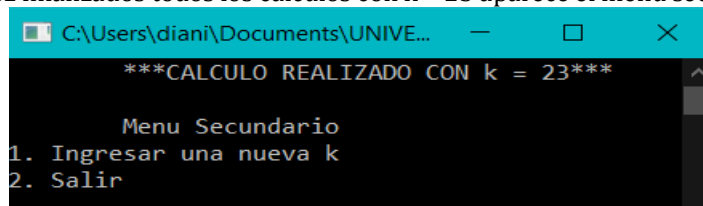


Figura6.

Archivo Salida 1: 'conjunto universo.txt'.

Muestra \sum^k en notación de conjunto.

```
S*=e, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, 0000, 0001, 0010,
0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 00000,
00001, 00010, 00011, 00100, 00101, 00110, 00111, 01000, 01001, 01010, 01011, 01100,
01101, 01110, 01111, 10000, 10001, 10010, 10011, 10100, 10101, 10110, 10111, 11000,
11001, 11010, 11011, 11100, 11101, 11110, 11111, 000000, 000001, 000010, 000011, 000100,
000101, 000110, 000111, 001000, 001001, 001010, 001011, 001100, 001101, 001110, 001111,
010000, 010001, 010010, 010011, 010100, 010101, 010110, 010111, 011000, 011001, 011010,
011011, 011100, 011101, 011110, 011111, 100000, 100001, 100010, 100011, 100100, 100101,
100110, 100111, 101000, 101001, 101010, 101011, 101100, 101101, 101110, 101111, 110000,
110001, 110010, 110011, 110100, 110101, 110110, 110111, 111000, 111001, 111010, 111011,
111100, 111101, 111110, 111111, 0000000, 0000001, 0000010, 0000011, 0000100, 0000101,
0000110, 0000111, 0001000, 0001001, 0001010, 0001011, 0001100, 0001101, 0001110,
0001111, 0010000, 0010001, 0010010, 0010011, 0010100, 0010101, 0010110, 0010111,
0011000, 0011001, 0011010, 0011011, 0011100, 0011101, 0011110, 0011111, 0100000,
0100001, 0100010, 0100011, 0100100, 0100101, 0100110, 0100111, 0101000, 0101001,
0101010, 0101011, 0101100, 0101101, 0101110, 0101111, 0110000, 0110001, 0110010,
0110011, 0110100, 0110101, 0110110, 0110111, 0111000, 0111001, 0111010, 0111011,
0111100, 0111101, 0111110, 0111111, 1000000, 1000001, 1000010, 1000011, 1000100,
1000101, 1000110, 1000111, 1001000, 1001001, 1001010, 1001011, 1001100, 1001101,
1001110, 1001111, 1010000, 1010001, 1010010, 1010011, 1010100, 1010101, 1010110,
1010111, 1011000, 1011001, 1011010, 1011011, 1011100, 1011101, 1011110, 1011111,
1100000, 1100001, 1100010, 1100011, 1100100, 1100101, 1100110, 1100111, 1101000,
1101001, 1101010, 1101011, 1101100, 1101101, 1101110, 1101111, 1110000, 1110001,
1110010, 1110011, 1110100, 1110101, 1110110, 1110111, 1110111, 1111000, 1111001,
1111010, 1111011, 1111100, 1111101, 1111110, 1111111, 00000000, 00000001, 00000010, 00000011,
00000100, 00000101, 00000110, 00000111, 00001000, 00001001, 00001010, 00001011,
00001100, 00001101, 00001110, 00001111, 00010000, 00010001, 00010010, 00010011,
00010100, 00010101, 00010110, 00010111, 00011000, 00011001, 00011010, 00011011,
00011100, 00011101, 00011110, 00011111, 00100000, 00100001, 00100010, 00100011,
00100100, 00100101, 00100110, 00100111, 00101000, 00101001, 00101010, 00101011,
00101100, 00101101, 00101110, 00101111, 00110000, 00110001, 00110010, 00110011,
00110100, 00110101, 00110110, 00110111, 00111000, 00111001, 00111010, 00111011,
00111100, 00111101, 00111110, 00111111, 01000000, 01000001, 01000010, 01000011,
01000100, 01000101, 01000110, 01000111, 01001000, 01001001, 01001010, 01001011,
01001100, 01001101, 01001110, 01001111, 01010000, 01010001, 01010010, 01010011,
01010100, 01010101, 01010110, 01010111, 01011000, 01011001, 01011010, 01011011,
01011100, 01011101, 01011110, 01011111, 01100000, 01100001, 01100010, 01100011,
01100100, 01100101, 01100110, 01100111, 01101000, 01101001, 01101010, 01101011,
01101100, 01101101, 01101110, 01101111, 01110000, 01110001, 01110010, 01110011,
01110100, 01110101, 01110110, 01110111, 01111000, 01111001, 01111010, 01111011,
01111100, 01111101, 01111110, 01111111, 10000000, 10000001, 10000010, 10000011,
10000100, 10000101, 10000110, 10000111, 10001000, 10001001, 10001010, 10001011,
10001100, 10001101, 10001110, 10001111, 10010000, 10010001, 10010010, 10010011,
10010100, 10010101, 10010110, 10010111, 10011000, 10011001, 10011010, 10011011,
10011100, 10011101, 10011110, 10011111, 10100000, 10100001, 10100010, 10100011,
10100100, 10100101, 10100110, 10100111, 10101000, 10101001, 10101010, 10101011,
10101100, 10101101, 10101110, 10101111, 10110000, 10110001, 10110010, 10110011,
```

Figura 7.

Archivo Salida 2: 'cadena universo.txt'.

Concatena todas las cadenas de Σ^k en una sola cadena

[illegible]

Figura8.

Para realizar las gráficas se requirió de la salida de dos archivos de texto extra 'unos cadena.txt' y 'unos conjunto.txt'. **Archivo Extra de Salida 1: 'unos conjunto.txt'.**

Número de 1s de cada cadena que pertenece a Σ^k

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

Figura9.

Archivo Extra de Salida 2: "unoscadena.txt".

Número de 1s de subcadenas de 32 símbolos de cadena concatenada Σ^k

```

0 15
1 12
2 20
3 9
4 14
5 16
6 18
7 23
8 7
9 10
10 15
11 12
12 16
13 20
14 12
15 16
16 20
17 17
18 22
19 25
20 6
21 9
22 9
23 13
24 11
25 14
26 18
27 11
28 13
29 14
30 17
31 16
32 22

```

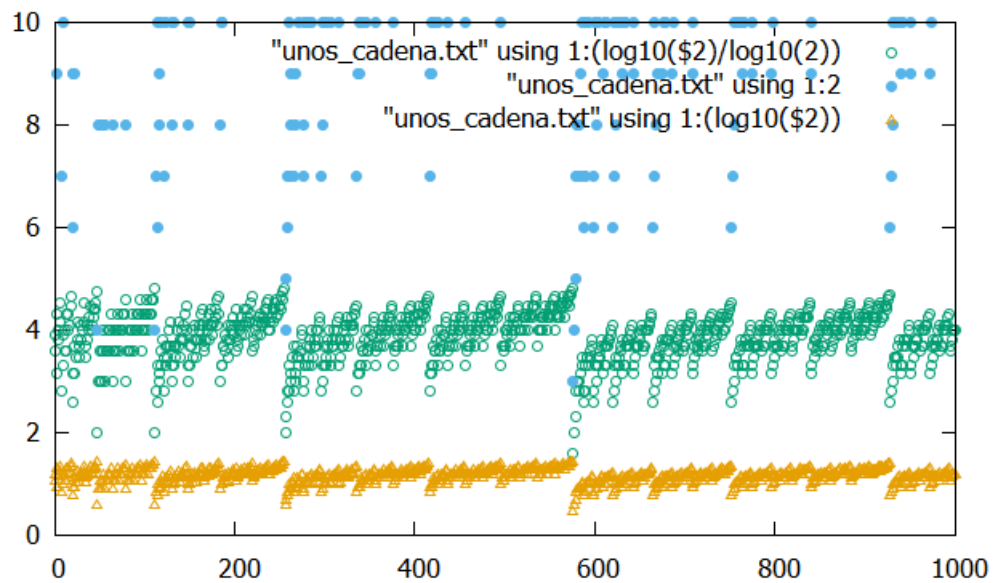
Figura10.

Grafica 'unos cadena.txt':

(AZUL) Número de 1s de cada cadena.

(VERDE) Logaritmo base 2 de número de 1s de cada cadena.

(AMARILLO) Logaritmo base 10 de número de 1s de cada cadena.

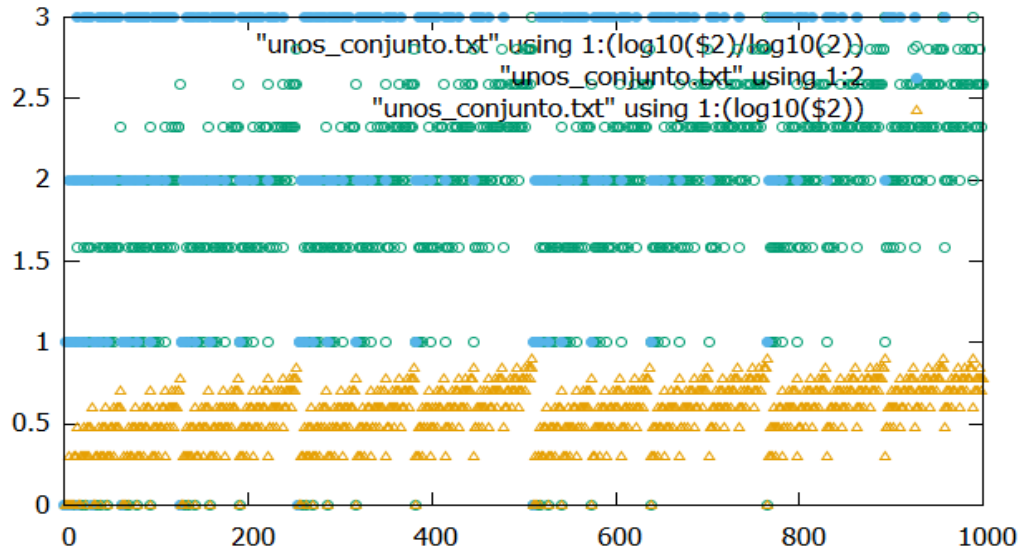


Grafica 'unos conjunto.txt':

(COLOR) Número de 1s de cada cadena.

(COLOR) Logaritmo base 2 de número de 1s de cada cadena.

(COLOR) Logaritmo base 10 de número de 1s de cada cadena.



PRUEBA [2]

El programa genera un número entero aleatorio k = que se encuentra en el intervalo $[0,1000]$

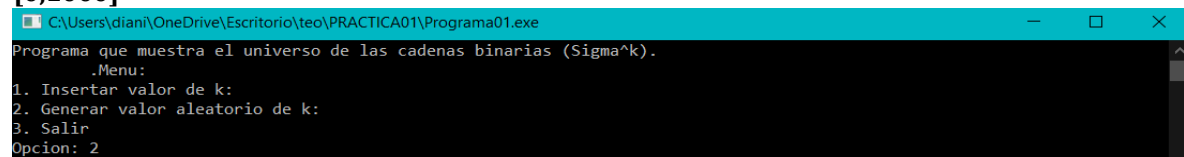


Figura13

Menú Secundario

Una vez finalizados todos los cálculos con $k = 9$ aparece el menú secundario

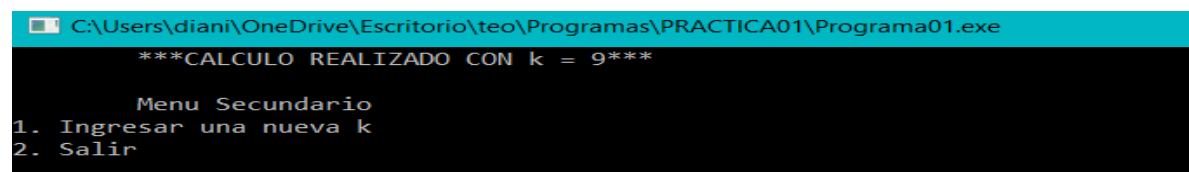


Figura14

Archivo Salida 1: 'conjunto universo.txt'.

Muestra \sum^k en notación de conjunto.

S*={e, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 00000, 00001, 00010, 00011, 00100, 00101, 00110, 00111, 01000, 01001, 01010, 01011, 01100, 01101, 01110, 10000, 10001, 10010, 10011, 10100, 10101, 10110, 10111, 11000, 11001, 11010, 11011, 11100, 11101, 11110, 11111, 000000, 000001, 000010, 000011, 000100, 000101, 000110, 000111, 001000, 001001, 001010, 001011, 001100, 001101, 001110, 001111, 010000, 010001, 010010, 010011, 010100, 010101, 010110, 010111, 011000, 011001, 011010, 011011, 011100, 011101, 011110, 011111, 100000, 100001, 100010, 100011, 100100, 100101, 100110, 100111, 101000, 101001, 101010, 101011, 101100, 101101, 101110, 101111, 110000, 110001, 110010, 110011, 110100, 110101, 110110, 110111, 111000, 111001, 111010, 111011, 111100, 111101, 111110, 111111, 0000000, 0000001, 0000010, 0000011, 0000100, 0000101, 0000110, 0000111, 0000100, 0001000, 0001001, 0001010, 0001011, 0001100, 0001101, 0001110, 0001111, 0001100, 0001101, 0001110, 0001111, 0010000, 0010001, 0010010, 0010011, 0010100, 0010101, 0010110, 0010111, 0011000, 0011001, 0011010, 0011011, 0011100, 0011101, 0011110, 0011111, 0100000, 0100001, 0100010, 0100011, 0100100, 0100101, 0100110, 0100111, 0101000, 0101001, 0101010, 0101011, 0101100, 0101101, 0101110, 0101111, 0110000, 0110001, 0110010, 0110011, 0110100, 0110101, 0110110, 0110111, 0111000, 0111001, 0111010, 0111011, 0111100, 0111101, 0111110, 0111111, 1000000, 1000001, 1000010, 1000011, 1000100, 1000101, 1000110, 1000111, 1001000, 1001001, 1001010, 1001011, 1001100, 1001101, 1001110, 1001111, 1010000, 1010001, 1010010, 1010011, 1010100, 1010101, 1010110, 1010111, 1011000, 1011001, 1011010, 1011011, 1011100, 1011101, 1011110, 1011111, 1100000, 1100001, 1100010, 1100011, 1100100, 1100101, 1100110, 1100111, 1101000, 1101001, 1101010, 1101011, 1101100, 1101101, 1101110, 1101111,

Figura15

Archivo Salida 2: 'cadena universo.txt'.

Concatena todas las cadenas de Σ^k en una sola cadena

```

00100011011000001001111001011101110000000100100011010000110110011110
0010011010101111001101110111000000001000100000110010000010100110001
1101000010010101001011000110101110011110000100001001010011101001
0101101010111100011001110101110111101111011110111100000000000100
0010000011000100000101000110000111001000001001001010001011001000011
010011100001110100000100010100100100101010100010101010100101101100
01100101001010110110111000111010111100001000010001010001010001110
01001001011001101001110100010100110101010101101100101101101101011
1111000011000111001011001111010011010110110111110001110001110101
111011111001110111101111100000000000010000010000011000010000
01010000110000011100010000001001000101000001010001100000110100011100
001111001000000100010010010001001100101000010100010110001011001100
000110010011010001101001110000111010011110001111010000001000010100
010010001101001000100101010011001001110101000010001010101001010101
011000101010101110010111011000001100010110010011001101010100010101
011011001101101110000110010110100110101110001110101111001111
111000000100000110000101000011000010010001010100011010001111001000100
10011001010100101110010010011010011110100111101000010100010100101
01001110101001010101010101010101110100010100110101010101110110110
01011101011101011111000001100001100010100011100100110010100101100
1101100111101000101001110101010101011101001101010110101110101111
100001110001110010111001111010011101011101011110111111000111001
111101011101111111001111011111101111100000000000000010000001000
000011000001000000010100000110000011100001000000100100001010000010
110000110000001010000111000001110001000000010001000100100000100100
01010000010101000101100001011000110000001100100010100000101000111
0000011101000111100001111001000000010000100100010001000100100000
1001010010011000100111001010000010100101010001010100101100001011
01001011100010111001100000011000100110010001100110011010000101010
11011000101100111000000110010011010001010100110100111000011101010
110110001011001110000001100100110100010101001110100111000011101011

```

Figura16

Para realizar las gráficas se requirió de la salida de dos archivos de texto extra 'unos cadena.txt' y 'unos conjunto.txt'. **Archivo Extra de Salida 1:'unos conjunto.txt'.**

Número de 1s de cada cadena que pertenece a \sum^k

-	0	0
	1	1
	2	0
	3	1
	4	1
	5	2
	6	0
	7	1
	8	1
	9	2
	10	1
	11	2
	12	2
	13	3
	14	0
	15	1
	16	1
	17	2
	18	1
	19	2
	20	2
	21	3
	22	1
	23	2
	24	2
	25	3
	26	2
	27	3
	28	3
	29	4
	30	0
	31	1

Figura17

Archivo Extra de Salida 2: “unoscadena.txt”.

Número de 1s de subcadenas de 32 símbolos de cadena concatenada \sum^k

0	15
1	12
2	20
3	9
4	14
5	16
6	18
7	23
8	7
9	10
10	15
11	12
12	16
13	20
14	12
15	16
16	20
17	17
18	22
19	25
20	6
21	9
22	9
23	13
24	11
25	14
26	18
27	11
28	13
29	14
30	17
31	16
32	18
33	22

Figura18.

Conclusión

Durante la práctica 1 se obtuvo el conocimiento de cómo obtener las cadenas del alfabeto Σ^k .

Además de como utilizar las tablas de verdad imprimir el universo de cadenas binarias de Σ^k

Durante el desarrollo de la práctica se presentaron dificultades en cuanto a la memoria que utilizaba el programa al momento de trasladarse a archivos. Utilice el software "EmEditor" el cual permite abrir archivos pesados pero los datos de $k = 23$ fueron más grandes de lo soportado. Por inducción comprobamos que si funciona para $k = n$ funciona para $k = n+1$.

Anexo

Programa01.cpp

```
#include<iostream>
#include<time.h>      //seed de numero random
#include<stdlib.h>
#include<math.h>      //Funcion pow
#include<stdio.h>
#include<fstream>     //ARCHIVOS

using namespace std;
//Variables Globales
int k;
int **parte1;
int **parte2;
long int filas;
//Prototipos de funciones
void MenuPrincipal();
int Numero_Aleatorio();
void Binario(int);
void Conjunto();
void Cadena();
void SubMenu();
void LiberarMemoria();

//Funcion principal
int main(){
    cout<<"Programa que muestra el universo de las cadenas binarias
(Sigma^k)."<<endl;
    MenuPrincipal();
    LiberarMemoria();
    return 0;
}

//Menú
void MenuPrincipal(){
    int opcion=0;
    do{
        cout<<"\tMenu: "<<endl;
        cout<<"1. Insertar valor de k: "<<endl;
        cout<<"2. Generar valor aleatorio de k: "<<endl;
        cout<<"3. Salir "<<endl;
        cout<<"Opcion: ";
```

```

        cin>>opcion;
        system("cls");
        switch(opcion){
        case 1:
            cout<<"Digite el valor de k : ";
            cin>>k;
            //cout<<"2^"<<k<<" = "<<pow(2,k)<<endl;
            Binario(k);
            Conjunto();
            Cadena();
            system("cls");
            cout<<"\t***CALCULO REALIZADO CON k =
" <<k<<"***\n\n";

            LiberarMemoria();
            SubMenu();

        break;
        case 2:
            k = Numero_Aleatorio();
            //cout<<"\nk = " <<k<<endl;
            //cout<<"2^"<<k<<" = "<<pow(2,k)<<endl;
            Binario(k);
            Conjunto();
            Cadena();
            system("cls");
            cout<<"\t***CALCULO REALIZADO CON k =
" <<k<<"***\n\n";

            LiberarMemoria();
            SubMenu();

        break;
        }
        if(opcion > 3){
            cout<<"\nOpcion invalida. " <<endl;
            system("cls");
        }
    }while(opcion > 3);
}
//Genera un numero aleatorio entre 1 - 1000
int Numero_Aleatorio(){
    srand((unsigned)time(NULL));
    return (rand()%10)+1;
}

void Binario(int n){

    int resultado_div;

```

```

        filas = (pow(2,n))/2;
        //Reservando memoria para filas y columnas de la 1a mitad de las
combinaciones
        parte1 = new int*[filas];
        for(int i = 0; i < filas; i++){
            parte1[i] = new int[n];
        }
        //Reservando memoria para filas y columnas de la 2da mitad de las
combinaciones
        parte2 = new int*[filas];
        for(int i = 0; i < filas; i++){
            parte2[i] = new int[n];
        }
        //Utilizar metodo de la division para conseguir combinaciones
        for(int i=0; i<filas; i++)
        {
            resultado_div=i;
            for(int j=n-1; j>=0; j--)
            {
                if(resultado_div%2==0)
                {
                    parte1[i][j]=0;
                    parte2[i][j]=1;
                }
                else{
                    parte1[i][j]=1;
                    parte2[i][j]=0;
                }
                resultado_div = resultado_div/2;
            }
        }
    }
}
//Imprimir en un archivo de texto el universo en forma de conjunto
void Conjunto(){
    int potencia,z,unos,contador=0;
    //Imprimir archivo
    ofstream conjunto;
    ofstream unosConjunto;
    conjunto.open("conjunto_universo.txt",ios::out); //Abrir archivo
    unosConjunto.open("unos_conjunto.txt",ios::out);

    if(conjunto.fail())
    {
        cout<<"ERROR al abrir archivo"<<endl;
        exit(1);
    }
}

```

```

    }
    if(unosConjunto.fail())
    {
        cout<<"ERROR al abrir archivo"<<endl;
        exit(1);
    }

    conjunto<<"S*={e, ";
    unos = 0;
    contador=0;
    for(z=1;z<k;z++){
        potencia = pow(2,z);
        for(int i=0; i<potencia; i++)
        {
            for(int j=0; j<z; j++)
            {
                conjunto<<parte1[i][j+k-z];
                if(parte1[i][j+k-z] == 1)
                {
                    unos += 1;
                }
            }
            unosConjunto<<contador<<" "<<unos<<endl;
            contador +=1 ;
            unos = 0;
            conjunto<<" ";
        }
    }

    for(int i=0; i<filas; i++)
    {
        for(int j=0; j<k; j++)
        {
            conjunto<<parte1[i][j];
            if(parte1[i][j] == 1)
            {
                unos += 1;
            }
        }
        unosConjunto<<contador<<" "<<unos<<endl;
        contador +=1 ;
        unos = 0;
        conjunto<<" ";
    }
}

```

```

unos=0;
for(int i=filas-1; i>=0; i--)
{
    for(int j=0; j<k; j++)
    {
        conjunto<<parte2[i][j];
        if(parte2[i][j] == 1)
        {
            unos += 1;
        }
    }
    if(i>0)
    {
        conjunto<<" ";
    }
    unosConjunto<<contador<<" "<<unos<<endl;
    contador +=1 ;
    unos = 0;
}
conjunto<<" }";

//Cerrar archivo
conjunto.close();
unosConjunto.close();
}

//Imprimir en un archivo de texto el universo en forma de cadena
void Cadena(){
    int potencia,z,longitud,unos,cadenas;
    //Imprimir archivo
    ofstream cadena;
    ofstream unosCadena;
    cadena.open("cadena_universo.txt",ios::out); //Abrir archivo
    unosCadena.open("unos_cadena.txt",ios::out);
    if(cadena.fail())
    {
        cout<<"ERROR al abrir archivo"<<endl;
        exit(1);
    }
    if(unosCadena.fail())
    {
        cout<<"ERROR al abrir archivo"<<endl;
        exit(1);
    }
}

```



```

    }
    longitud = 0;
    cadenas = 0;
    unos = 0;
    for(z=1;z<k;z++){
        potencia = pow(2,z);
        for(int i=0; i<potencia; i++)
        {
            for(int j=0; j<z; j++)
            {

                if(longitud == 32){

                    longitud=0;
                    unosCadena<<cadenas<<"

" << unos << endl;

                    cadenas += 1;
                    unos=0;
                }
                if(parte1[i][j+k-z] == 1)
                {
                    unos += 1;
                    //cout<<"\nunos: " << unos;
                }
                cadena<<parte1[i][j+k-z];
                longitud += 1;
                //cout<<"\nlong: " << longitud;
            }
            if(longitud == 0)
            {
                //cout<<"\ncadenas: " << cadenas;
                unosCadena<<cadenas<<" " << unos << endl;
                cadenas += 1;
            }
        }
    }

    for(int i=0; i<filas; i++)
    {

        for(int j=0; j<k; j++)

            if(longitud == 32){

```

```

        longitud=0;
        unosCadena<<cadenas<<" "<<unos<<endl;
        cadenas += 1;
        unos=0;
    }
    if(parte1[i][j] == 1)
    {
        unos += 1;
        //cout<<"\nunos: "<<unos;
    }
    cadena<<parte1[i][j];
    longitud += 1;
    //cout<<"\nlong: "<<longitud;
}
}

for(int i=filas-1; i>=0; i--)
{
    for(int j=0; j<k; j++)
    {
        if(longitud == 32){
            longitud=0;
            unosCadena<<cadenas<<" "<<unos<<endl;
            cadenas += 1;
            unos=0;
        }
        if(parte2[i][j] == 1)
        {
            unos += 1;
            //    cout<<"\nunos: "<<unos;
        }
        cadena<<parte2[i][j];
        longitud += 1;
        //cout<<"\nlong: "<<longitud;
    }
    if(i == 0)
    {
        //cout<<"\ncadenas: "<<cadenas;
        unosCadena<<cadenas<<" "<<unos<<endl;
        cadenas += 1;
    }
}

```

```

    }
}
//Cerrar archivo
cadena.close();
unosCadena.close();

}

//Submenu
void SubMenu(){
    int opcion=0;
    do{
        cout<<"\tMenu Secundario "<<endl;
        cout<<"1. Ingresar una nueva k "<<endl;
        cout<<"2. Salir "<<endl;
        cin>>opcion;
        switch(opcion){
            case 1:
                system("cls");
                MenuPrincipal();
                break;
            case 2:
                exit(1);
                break;
        }
        if(opcion > 2){
            cout<<"\nOpcion invalida. "<<endl;
            system("pause");
        }
    }while(opcion > 2);
}

//Liberar memoria
void LiberarMemoria(){
    for(int i=0; i<k; i++)
    {
        free(parte1[i]);
        free(parte2[i]);
    }
    free(parte1);
    free(parte2);
}

```