Primer Entrega de Programas

Aburto Pérez Luis Mario

11 de septiembre de 2016

Índice

1. Universo

El universo es el conjunto que contiene a la unión de todos los conjuntos de un alfabeto.

1.1. Descripción del problema

Se desea elaborar un programa que calcule el universo de un alfabeto binario a partir de un número dado, este número representara el número de caracteres que tendrá cada elemento, es decir, se calcularan las combinaciones que se pueden realizar con cierto número de caracteres, este será dado por el usuario o un numero aleatorio en el programa. El programa tendrá dos opciones: Automático y Manual para indicar la forma que obtendrá el número con el que va a trabajar. La restricción de este número es que este en el rango de 1 a 1000.

1.2. Código fuente

Este programa fue elaborado en el lenguaje C, su código fuente se muestra a continuación:

Archivo: universo.c

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <time.h>

int random(int,int);
void agregarCaracter(int,int,int,char[]);
int guardarPalabra(int);
void opcionAutomatica(void);
void opcionManual(void);
int crearArchivo(void);
int menu(void);
int repetir(int);


int main(void){
        int modo=0;
        int rep=0;
        srand(time(NULL));
        if(crearArchivo()==-1){
                printf("Error al crear el archivo");
        }
        modo=menu();
        if(modo==0){
                return 0;
        }
```

```
        do{
                rep=repetir(modo);
        }while(rep!=1);
        return 1;
}

int repetir(int modo){
        int opcion=0;
        if(modo==1){
                printf("Quieres que se repita? \n 1.Si \n 2. No \n");
                scanf("%d",&opcion);
                if(opcion==1){
                        printf("Has seleccionado que se repita\n");
                        opcionManual();
                        return 0;
                }
                if(opcion==2){
                        printf("No repetir, Adios");
                        return 1;
                }
        }
        if(modo==2){
                printf("Quieres que se repita? \n 1.Si \n 2. No \n");
                opcion=random(1,2);
                if(opcion==1){
                printf("Has seleccionado que se repita\n");
                opcionAutomatica();
                return 0;
                }
                if(opcion==2){
                        printf("No repetir, Adios");
                        return 1;
                }
        }
}

int menu(void){
        int potencia=0;
        int opcion=0;
        printf("PROGRAMA 1\n MENU DE OPCIONES:\n 1. Operacion Manual \n2.
            Operacion automatica\n3. Salir\n===> \n");
        scanf("%d",&opcion);
        if(opcion==1){
                opcionManual();
        }
        if(opcion==2){
                opcionAutomatica();
        }
        if(opcion==3){
                printf("Adios");
                return 0;
        }
```

```
        return opcion;
}

int crearArchivo(void){
        FILE * archivo;
        archivo=fopen("archivo.txt","w");
        fclose(archivo);
        if(archivo==NULL){
                return -1;
        }else{
                return 0;
        }
}

void opcionAutomatica(void){
        int potencia=0;
        printf("Opcion Automatica Seleccionada\n");
                printf("Indique la potencia a usar ===>  ");
                potencia=random(0,1000);
                printf("\nLa potencia seleccionada es %d \n",potencia);
                if(guardarPalabra(potencia)!=0){
                        printf("Error con el archivo");
                }
}

void opcionManual(void){
        int potencia=0;
                printf("Opcion Manual Seleccionada\n");
                printf("Indique la potencia a usar ===>  ");
                scanf("%d",&potencia);
                printf("\nLa potencia seleccionada es %d \n",potencia);
                if(guardarPalabra(potencia)!=0){
                        printf("Error con el archivo");
                }
}

void agregarCaracter(int potencia, int noPalabra,int casilla,char palabra []){
        int x=0;
        if(potencia==0){
                        palabra[casilla]='e';
                }
        for(x=potencia;x>0;x--,casilla++){
                int rep=pow(2,x);
                        if(x==1 && noPalabra%2==1){
                                palabra[casilla]='0';
                        }else{
                                if(x==1 && noPalabra%2==0){
                                        palabra[casilla]='1';
                                }else{
                                                if(noPalabra<=rep/2){
                                                        palabra[casilla]='0';
                                                }else{
```

```
                                                palabra[casilla]='1';
                                                noPalabra=noPalabra-(
                                                    rep/2);
                                                }
                                        }
                                }
        }
}

int guardarPalabra(int potencia){
        FILE * archivo;
        int i=0;
        int x=0;
        int repeticiones=pow(2,potencia);
        int numeroPalabra=1;
        archivo=fopen("archivo.txt","a");
        fprintf(archivo,"A={");
        if(potencia==0){
                potencia++;
                char palabra [potencia];
                potencia--;
        for(i=0;i<repeticiones;i++){
                agregarCaracter(potencia,numeroPalabra,0,palabra);
                        for(x=0;x<=potencia;x++){
                                fprintf(archivo,"%c",palabra[x]);
                        }
                                if(i!=repeticiones-1){
                                fprintf(archivo,",");
                                }
                                numeroPalabra++;
        }
        }else{
                char palabra[potencia];
                for(i=0;i<repeticiones;i++){
                agregarCaracter(potencia,numeroPalabra,0,palabra);
                        for(x=0;x<potencia;x++){
                                fprintf(archivo,"%c",palabra[x]);
                        }
                                if(i!=repeticiones-1){
                                fprintf(archivo,",");
                                }
                                numeroPalabra++;
                }
        }
        fprintf(archivo,"}");
        fclose(archivo);
        return 0;
}
int random(int limite_inf,int limite_sup){
        int num=(rand()%limite_sup) + limite_inf;
        return num;
}
```

1.3. Pruebas

Se realizaran dos pruebas para este programa una en su forma manual y otra en su forma automática.

Modo Automático:

```
PROGRAMA 1
 MENU DE OPCIONES:
 1. Operacion Manual
2.Operacion automatica
3.Salir
===>
2
Opcion Automatica Seleccionada
Indique la potencia a usar ===>
La potencia seleccionada es 4
Quieres que se repita?
 1.Si
 2. No
No repetir, Adios
--------------------------------
```

Figura 1: Selección de un n = 1 y n = 4 de forma automática

De la ejecución anterior se procesó la información dando como resultado:

```
A={0000,0001,0010,0011,0100,0101,0110,0111,1000,1001,1010,1011,1100,1101,1110,1111}
```

Figura 2: Archivo de salida con el universo generado

Modo Manual:

```
PROGRAMA 1
 MENU DE OPCIONES:
 1. Operacion Manual
2.Operacion automatica
3.Salir
===>
1
Opcion Manual Seleccionada
Indique la potencia a usar ===>  0

La potencia seleccionada es 0
Quieres que se repita?
 1.Si
 2. No
1
Has seleccionado que se repita
Opcion Manual Seleccionada
Indique la potencia a usar ===>  3

La potencia seleccionada es 3
Quieres que se repita?
 1.Si
 2. No
2
No repetir, Adios
```

Figura 3: Selección de un n = 0 y n = 3 de forma manual

De la ejecución anterior se procesó la información dando como resultado:

```
A={e}A={000,001,010,011,100,101,110,111}
```

Figura 4: Archivo de salida con el universo generado

2. Números Primos

Los números primos son aquellos que solo cuentan con dos divisores, el 1 y el mismo numero.

2.1. Descripción del problema

Se desea generar un programa que calcule los números primos primero en decimal y luego que represente el mismo grupo de números en su forma binaria. Para el cálculo de las números se dará un numero n que será el límite hasta el cual se generaran todos los números primos menores o iguales a ese número dado. La única restricción para este número es que este en el rengo de 1 a 1000.

2.2. Código fuente

Este programa fue elaborado en el lenguaje C, su código fuente se muestra a continuación:

Archivo: primos.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

int random(int,int);
int elementos(int);
void generarPrimos(int [],int);
void guardarDecimales(int,int []);
int crearArchivo(void);
void convertirBinario(int [],int);
int leerModo(void);
int menu(int);
int opcionAutomatica(void);
int opcionManual(void);
int repetir(int);

int main(void){
        srand(time(NULL));
        int modo=leerModo();
        int limite=menu(modo);
        if(crearArchivo()!=0){
                printf("Error al crear archivo");
        }
        while(limite!=0){
                int noElementos=elementos(limite);
                int primos[noElementos];
                generarPrimos(primos,limite);
                guardarDecimales(noElementos,primos);
                convertirBinario(primos,noElementos);
```

```
                limite=repetir(modo);
                }
}

int repetir(int modo){
        int opc=0;
        int r=0;
        int invalido=0;
        if(modo==1){
                do{
                printf("Quieres que se repita? \n 1.Si \n 2. No \n");
                scanf("%d",&opc);
                if(opc<1||opc>2){
                        printf("Opcion Invalida\n");
                        invalido=1;
                }else{
                        if(opc==1){
                                printf("Has seleccionado que se repita\n");
                                r=opcionManual();
                                return r;
                        }else{
                                if(opc==2){
                                        printf("No repetir , Adios");
                                        return 0;
                                        }
                                }
                        }
                }while(invalido==1);
        }
        if(modo==2){
                printf("Quieres que se repita? \n 1.Si \n 2. No \n");
                opc=random(1,2);
                if(opc==1){
                printf("Has seleccionado que se repita\n");
                r=opcionAutomatica();
                return r;
                }
                if(opc==2){
                        printf("No repetir , Adios");
                        return 0;
                }
        }
}

int leerModo(void){
        int num=0;
        printf("PROGRAMA 1\n MENU DE OPCIONES:\n 1. Operacion Manual \n2.
            Operacion automatica\n3. Salir\n===> \n");
        scanf("%d",&num);
        return num;
}
```

```
int menu(int modo){
        int limite=0;
        if(modo==1){
                limite=opcionManual();
        }
        if(modo==2){
                limite=opcionAutomatica();
        }
        if(modo==3){
                printf("Adios");
        }
        return limite;
}

int opcionAutomatica(void){
        int limite=0;
        printf("Opcion Automatica Seleccionada\n");
                printf("Indique el limite ===>  ");
                limite=random(1,1000);
                printf("\nEl limite seleccionado es %d \n",limite);
                return limite;
}

int opcionManual(void){
        int limite=0;
                printf("Opcion Manual Seleccionada\n");
                printf("Indique el limite ===>  ");
                scanf("%d",&limite);
                printf("\nEl limite seleccionado es %d \n",limite);
                return limite;
}

int elementos(int limite){
        int x=0;
        int i=0;
        int np=0;
        int elementos=0;
                for(x=2;x<=limite;x++){
                        np=0;
                        for(i=1;i<x;i++){
                                if(x%i==0){
                                        np++;
                                }
                        }
                        if(np==1){
                                elementos++;
                        }
                }
        return elementos+1;
}

void guardarDecimales(int nElementos, int primos []){
```

```
        FILE *fp;
        int x=0;
        fp=fopen("archivo.txt","a");
        fprintf(fp,"D={");
        for(x=0;x<nElementos;x++){
        fprintf(fp,"%d",primos[x]);
                if(x!=nElementos-1){
                        fprintf(fp,",");
                }
        }
        fprintf(fp,"}");
}

void generarPrimos(int primos[],int limite){
        primos[0]=1;
        int i=0;
        int x=0;
        int np=0;
        int casilla=1;
        for(x=2;x<=limite;x++){
                np=0;
                for(i=1;i<x;i++){
                        if(x%i==0){
                        np++;
                        }
                }
        if(np==1){
                        primos[casilla]=x;
                        casilla++;
                }
        }
}

void convertirBinario(int primos[],int nElem){
        FILE *fp;
        fp=fopen("archivo.txt","a");
        int res=0;
        int x=0;
        int i=0;
        int c=0;
        char aux[10];
        char cadena[10];
        fprintf(fp,"B={");
        for(c=0;c<nElem;c++){
                x=0;
                do{
                res=primos[c]%2;
                itoa(res,aux,10);
                cadena[x]=aux[0];
                primos[c]=primos[c]/2;
                x++;
                }while(primos[c]!=0);
```

```
                        for(i=x-1;i>=0;i--){
                        fprintf(fp,"%c",cadena[i]);
                        }
                                if(c<nElem-1){
                                fprintf(fp,",");
                                }
        }
        fprintf(fp,"}");
}

int crearArchivo(void){
        FILE *fp;
        fp=fopen("archivo.txt","w");
        fclose(fp);
        if(fp==NULL){
                return -1;
        }else{
                return 0;
        }
}

int random(int limite_inf,int limite_sup){
        int num=(rand()%limite_sup) + limite_inf;
        return num;
}
```

2.3. Pruebas

Se realizaran dos pruebas para este programa una en su forma manual y otra en su forma automática.

Modo Manual:

```
PROGRAMA 1
 MENU DE OPCIONES:
 1. Operacion Manual
2.Operacion automatica
3.Salir
===>
1
Opcion Manual Seleccionada
Indique el limite ===>  30

El limite seleccionado es 30
Quieres que se repita?
 1.Si
 2. No
2
No repetir, Adios
--------------------------------
```

Figura 5: Selección de un n = 30 de forma manual

De la ejecución anterior se procesó la información dando como resultado:

```
D={1,2,3,5,7,11,13,17,19,23,29}
B={1,10,11,101,111,1011,1101,10001,10011,10111,11101}
```

Figura 6: Archivo de salida con los numeros primos hasta el limite en decimal y binario

Modo Automático:

```
PROGRAMA 1
 MENU DE OPCIONES:
 1. Operacion Manual
2.Operacion automatica
3.Salir
===>
2
Opcion Automatica Seleccionada
Indique el limite ===>
El limite seleccionado es 48
Quieres que se repita?
 1.Si
 2. No
No repetir, Adios
```

Figura 7: Selección de un n = 48 de forma automática

De la ejecución anterior se procesó la información dando como resultado:

```
D={1,2,3,5,7,11,13,17,19,23,29,31,37,41,43,47}
B={1,10,11,101,111,1011,1101,10001,10011,10111,11101,11111,100101,101001,101011,101111}
```

Figura 8: Archivo de salida con los numeros primos hasta el limite en decimal y binario

2.4. Gráfica de Ceros y Unos

Se graficó el comportamiento de los ceros y unos en los números primos en su forma binario dando como resultados la siguiente gráfica.

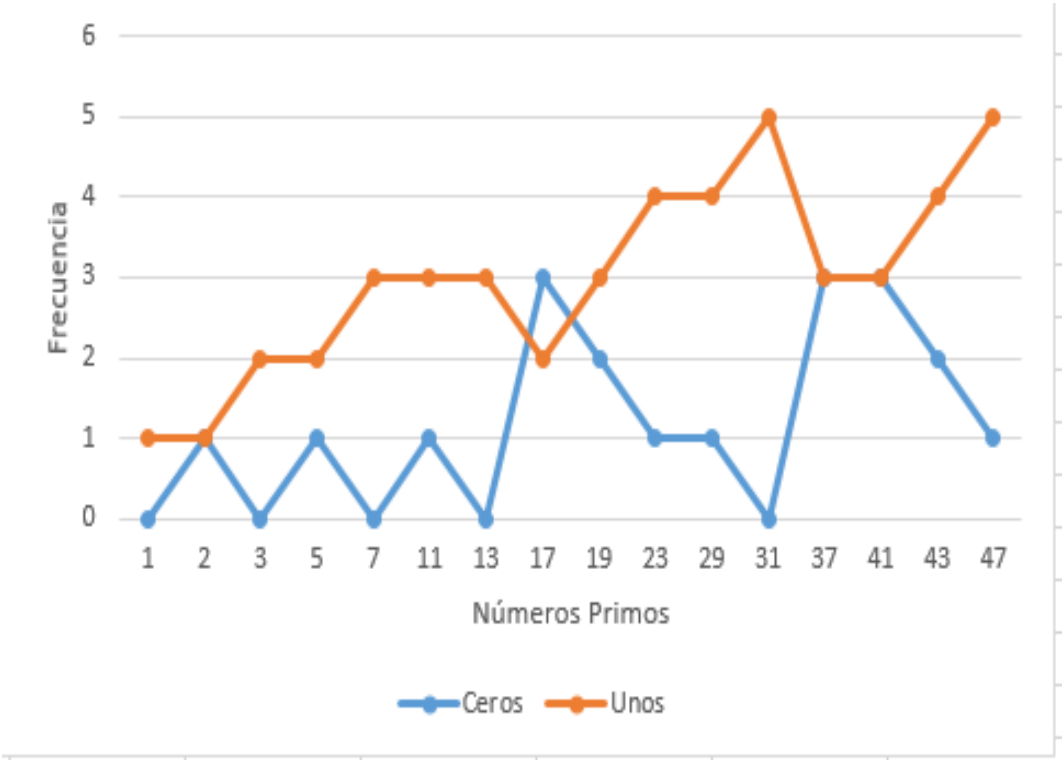

Figura 9: Comportamiento de 0's y 1's

3. Autómata Terminación 'ere'

3.1. Descripción del problema

Se desea generar un programa que contenga un autómata el cual determine si una palabra termina con la sub-cadena 'ere' . El programa tendrá dos modos, automático y manual, en su modo automático el programa leerá un archivo que contenga un texto, del cual sacara las palabras con dicha terminación y las guardara en otro archivo con el número de línea y carácter en el que se encuentra la palabra. En el modo manual el usuario ingresara una cadena para verificar si termina o no en 'ere'. Además contiene una opción que permite ver el grafico del autómata que se empleó en el programa.

3.2. Código fuente

Este programa fue elaborado en el lenguaje Python, su código fuente se muestra a continuación:

Archivo: ere.py

```
from tkinter import *
from random import *

def grafico():
        root = Tk()
        root.title('Automata␣Terminacion␣"ere"␣Grafico') # Nombre de la
            ventana
        canvas=Canvas(root,width=800,height=200)
        canvas.pack()
        canvas.create_arc(80,15,110,50, start=0, extent=180, style='arc') #
            arco 0-0
        canvas.create_arc(290,15,310,50, start=0, extent=180, style='arc') #
            arco 1-1
        canvas.create_arc(130,25,260,115, start=0, extent=180, style='arc') #
            arco 1-0 arriba
        canvas.create_arc(300,25,660,115, start=0, extent=180, style='arc') #
            arco 3-2 arriba
        canvas.create_arc(100,150,500,100, start=0, extent=-180, style='arc')
            #arco 3-1 abajo
        canvas.create_arc(90,200,700,70, start=0, extent=-180, style='arc') #
            arco 4-1 abajo
        canvas.create_arc(500,130,690,70, start=0, extent=-180, style='arc') #
            arco 4-3 abajo
        canvas.create_line(5,75,50,75) #Linea 1
        canvas.create_line(150,75,250,75) #Linea 2
        canvas.create_line(350,75,450,75) #Linea 3
        canvas.create_line(550,75,645,75) #Linea 4
        canvas.create_oval(50, 30, 150, 130, width=1, fill='green') #Primer
            Circulo
```

```
canvas.create_oval(250, 30, 350, 130, width=1, fill='blue') #Segundo
    Circulo
canvas.create_oval(450, 30, 550, 130, width=1, fill='green') #Tercer
    Circulo
canvas.create_oval(650, 30, 750, 130, width=1, fill='blue') #Cuarto
    Circulo
canvas.create_oval(645, 25, 755, 135, width=1) #Cuarto Circulo
canvas.create_oval(40,70,50,80,fill="red") #Flecha 0-0
canvas.create_oval(135,43,145,53,fill="red") #Flecha 1-0
canvas.create_oval(240,70,250,80,fill="red") #Flecha 0-1
canvas.create_oval(440,70,450,80,fill="red") #Flecha 1-2
canvas.create_oval(635,70,645,80,fill="red") #Flecha 0-1
canvas.create_oval(335,40,345,50,fill="red") #Flecha 3-1
canvas.create_oval(100,125,110,135,fill="red") #Flecha 2-0
canvas.create_oval(85,125,95,135,fill="red") #Flecha 3-0
canvas.create_oval(75,25,85,35,fill="red") #Flecha 0-0
canvas.create_oval(285,25,295,35,fill="red") #Flecha 1-1
canvas.create_oval(525,115,535,125,fill="red") #Flecha 3-2
estado=Label(root,text="Estado␣0",bg="green",font=("Helvetica",12),fg=
    "white") #ESTADO CERO
estado.place(x = 60, y = 60)
estado=Label(root,text="Estado␣1",bg="blue",font=("Helvetica",12),fg="
    white") #ESTADO UNO
estado.place(x = 270, y = 60)
estado=Label(root,text="Estado␣2",bg="green",font=("Helvetica",12),fg=
    "white") #ESTADO DOS
estado.place(x = 470, y = 60)
estado=Label(root,text="Estado␣3",bg="blue",font=("Helvetica",12),fg="
    white") #ESTADO TRES
estado.pack()
estado.place(x = 670, y = 60)
letra=Label(root,text="e",font=("Helvetica",12)) #
letra.place(x = 190, y = 60)
letra=Label(root,text="e",font=("Helvetica",12)) #
letra.place(x = 310, y = 5)
letra=Label(root,text="no␣es␣e",font=("Helvetica",12)) # 0-0
letra.place(x = 15, y = 5)
letra=Label(root,text="e",font=("Helvetica",12)) #
letra.place(x = 450, y = 5)
letra=Label(root,text="no␣es␣e␣ni␣r",font=("Helvetica",12)) # 1-0
letra.place(x = 140, y = 5)
letra=Label(root,text="no␣es␣e",font=("Helvetica",12)) # 1-0
letra.place(x = 280, y = 150)
letra=Label(root,text="no␣es␣e␣ni␣r",font=("Helvetica",12)) # 3-0
letra.place(x = 500, y = 180)
letra=Label(root,text="r",font=("Helvetica",12)) # 3-0
letra.place(x = 600, y = 120)
letra=Label(root,text="r",font=("Helvetica",12)) #
letra.place(x = 390, y = 60)
letra=Label(root,text="e",font=("Helvetica",12)) #
letra.pack()
letra.place(x=590,y=60)
```

```
        root.mainloop()

def guardarPalabra(caracter,cadena,linea):
        f=open("cadenas.txt","a")
        temporal=''
        x=caracter
        while(x<len(cadena) and (validacion(ord(cadena[x].lower()))!=0)):
                temporal+=cadena[x]
                x+=1
        f.write("Fila␣"+str(linea)+"␣Caracter␣"+str(caracter+1)+"␣"+temporal+"
            ␣")
        f.close()

def crearArchivos():
        cadenas=open("cadenas.txt","w")
        cadenas.close()
        estados=open("estados.txt","w")
        estados.close()

def automata(cadena):
        f=open("estados.txt","a")
        estado=0
        if(cadena==''):
                return 0
        else:
                for c in cadena:
                        if(estado==0):
                                f.write("Estado␣"+str(estado)+"␣"+ c+"␣")
                                if(c=='e'):
                                        estado=1
                        elif(estado==1):
                                f.write("Estado␣"+str(estado)+"␣"+ c+"␣")
                                if(c=='r'):
                                        estado=2
                                elif(c=='e'):
                                        estado=1
                                else:
                                        estado=0
                        elif(estado==2):
                                f.write("Estado␣"+str(estado)+"␣"+ c+"␣")
                                if(c=='e'):
                                        estado=3
                                else:
                                        estado=0
                        elif(estado==3):
                                f.write("Estado␣"+str(estado)+"␣"+ c+"␣")
                                if(c=='e'):
                                        estado=1
                                elif(c=='r'):
                                        estado=2
                                else:
                                        estado=0
```

```
        if(estado==3):
                f.write("Estado␣"+str(estado)+"␣␣␣")
                return 1
        f.close()

def auto():
        f = open("archivo.html","r")
        linea=""
        noLinea=1
        control=0
        temporal=''
        linea=f.readline()
        while (linea != ''):
                posTemporal=0
                pos=0
                palabra=linea.lower()
                for c in palabra:
                        if(validacion(ord(c))==0):
                                control=1
                                if(automata(temporal)==1):
                                        guardarPalabra(pos,linea,noLinea)
                                temporal=''
                                posTemporal+=1
                                pos=posTemporal
                        else:
                                temporal+=c
                                posTemporal+=1
                                control=0
                linea=f.readline()
                noLinea+=1
        if(control==0):
                if(automata(temporal)==1):
                        guardarPalabra(pos,linea,noLinea)
        f.close()
        f=open("estados.txt","a")
        f.write("\n\n")
        f.close()
        f=open("cadenas.txt","a")
        f.write("\n\n")
        f.close()
        repetir(2)

def menu():
        crearArchivos()
        eleccion=0
        while (eleccion!=4):
                eleccion=input("Selecciona␣una␣opcion\n1.Manual\n2.Automatico\
                    n3.Mostrar␣Grafico\n4.Salir\n===>␣")
                if(eleccion=='1'):
                        print("Opcion␣Manual␣seleccionada")
                        manual()
                elif(eleccion=='2'):
```

```
                print("Opcion Automatica seleccionada")
                auto()
            elif(eleccion=='3'):
                print("Visualizar Grafico")
                grafico()
            elif(eleccion=='4'):
                print("Adios")
                return 0
            else:
                print("Opcion Invalida , Intentalo de nuevo")

def repetir(modo):
    rep=''
    if(modo==1):
        while(rep!='1' and rep!='2'):
            rep=input("Deseas Repetir en este modo\n 1. Si \n 2.
                No\n ==> ")
            if(rep=='1'):
                print("Seleccionaste repetir")
                manual()
            elif(rep=='2'):
                print("Seleccionaste NO repetir")
            else:
                print("Opcion Invalida , Intentalo de nuevo")
    elif(modo==2):
        rep=choice(['1', '2'])
        if(rep=='1'):
            print("Seleccionaste repetir")
            auto()
        elif(rep=='2'):
            print("Seleccionaste NO repetir")

def manual():
    control=0
    pos=0
    posTemporal=0
    temporal=''
    cadena=input("Ingresa la cadena ")
    palabra=cadena.lower()
    for c in palabra:
        if(validacion(ord(c))==0):
            control=1
            if(automata(temporal)==1):
                guardarPalabra(pos,cadena,1)
            temporal=''
            posTemporal+=1
            pos=posTemporal
        else:
            temporal+=c
            posTemporal+=1
            control=0
    if(control==0):
```

```
                if(automata(temporal)==1):
                        guardarPalabra(pos,cadena,1)
        f=open("estados.txt","a")
        f.write("\n\n")
        f.close()
        f=open("cadenas.txt","a")
        f.write("\n\n")
        f.close()
        repetir(1)

def validacion(num):
        if(num>=97 and num<=122):
                return 1
        else:
                return 0

menu()
```

3.3. Pruebas

Se realizaran dos pruebas para este programa una en su forma manual y otra en su forma automática.

Modo Manual:

```
C:\Users\Luis\Desktop>python ere.py
Selecciona una opcion
1.Manual
2.Automatico
3.Mostrar Grafico
4.Salir
===> 1
Opcion Manual seleccionada
Ingresa la cadena where
Deseas Repetir en este modo
 1. Si
 2. No
 ==> 2
Seleccionaste NO repetir
```

Figura 10: Palabra 'where'

De la ejecución con la cadena ingresada se genera el archivo 'cadenas.txt'.

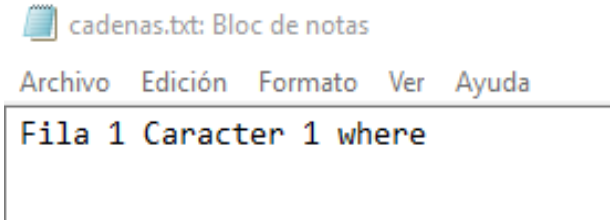

Figura 11: Archivo de salida con las cadenas validas de la ejecución.

Modo Automático: Para el modo automático se necesita tener un archivo el cual será leído por el programa para guardar en otro archivo las cadenas que son válidas.

```
Were music
From Wikipedia, the free encyclopedia

This article does not cite any sources. Please help improve this article
Were musicgere (Yoruba: Wéré) is an indigenous Yoruba music, which, like

Unlike ajisari, were is performed in groups. Usually young men or boys,

In early 1970s, were music genre became popular and forced its way into



Stub icon       This article about a music genre is a stub. You can help
Stub icon       This Nigeria-related article is a stub. You can help Wik
```

Figura 12: Archivo previamente generado para lectura.

Una vez con el archivo ya generado se ejecuta el programa en su modo automático.

```
C:\Users\Luis\Desktop>python ere.py
Selecciona una opcion
1.Manual
2.Automatico
3.Mostrar Grafico
4.Salir
===> 2
Opcion Automatica seleccionada
Seleccionaste NO repetir
Selecciona una opcion
1.Manual
2.Automatico
3.Mostrar Grafico
4.Salir
===> 4
Adios
```

Figura 13: Modo automatico sin repetición

De la ejecución anterior se procesó la información dando como resultado:

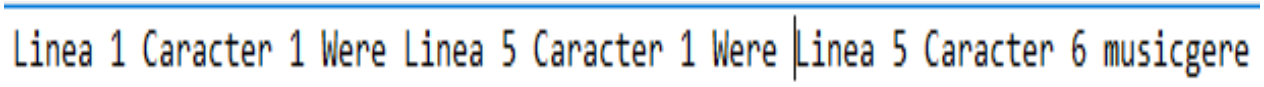

Figura 14: Archivo de salida con las cadenas validas y su ubicación

4. Autómata Paridad 0's y 1's

4.1. Descripción del problema

Se desea generar un programa que utilice un autómata para verificar si una cadena binaria tiene un numero par de 0's como de 1's. Si una cadena binaria cumple con este criterio entonces se guarda en un archivo. El programa tiene dos modos, automático y manual, el primero lee un archivo que contenga cadenas binarias y determina cual cumple con el criterio y cual no, en el segundo el usuario ingresa la cadena y el programa le indica si cumple o no, en caso de cumplir de igual forma se guarda en un archivo. Además contiene una opción que permite ver el grafico del autómata que se empleó en el programa.

4.2. Código fuente

Este programa fue elaborado en el lenguaje Python, su código fuente se muestra a continuación:

Archivo: paridad.py

```
from tkinter import *
from random import *

def guardarPalabra(cadena):
        filtro = open("cadenas.txt","a")
        filtro.write(cadena)
        filtro.close()

def crearArchivos():
        cadenas=open("cadenas.txt","w")
        cadenas.close()
        estados=open("estados.txt","w")
        estados.close()

def automata(cadena):
        estado = 0
        estados=open("estados.txt","a")
        x=0
        while(cadena[x]!=' '):
                if(estado==0):
                        estados.write('estado'+str(estado)+"-"+cadena[x]+" ")
                        if(cadena[x]=='0'):
                                estado=1
                        elif(cadena[x]=='1'):
                                estado=2
                        elif(cadena[x+1]==' '):
                                estado=0
                elif(estado==1):
                        estados.write('estado'+str(estado)+"-"+cadena[x]+" ")
                        if(cadena[x]=='0'):
```

```
                                estado=0
                        elif(cadena[x]=='1'):
                                estado=3
                elif(estado==2):
                        estados.write('estado'+str(estado)+"-"+cadena[x]+"␣")
                        if(cadena[x]=='0'):
                                estado=3
                        elif(cadena[x]=='1'):
                                estado=0
                elif(estado==3):
                        estados.write('estado'+str(estado)+"-"+cadena[x]+"␣")
                        if(cadena[x]=='0'):
                                estado=2
                        elif(cadena[x]=='1'):
                                estado=1
                        x+=1
        estados.write("estado"+str(estado) +"␣")
        if(estado==0):
                return 1
        else:
                return 0

def manual():
        cadena=input("Ingresa␣la␣cadena␣==>␣")
        if(automata(cadena+'␣')==1):
                guardarPalabra(cadena+'␣')
                print ("Cadena␣valida")
        else:
                print("Cadena␣invalida")

def auto():
        f = open("archivo.txt","r")
        cadena=""
        linea=f.readline()
        while(linea != ''):
                x=0
                cad=linea+'\n'
                while(cad[x]!='\n'):
                        if(cad[x]=='1' or cad[x]=='0' or cad[x]=='␣'):
                                cadena+=cad[x]
                                if(cad[x]=='␣' and (cad[x-1]=='0' or cad[x
                                    -1]=='1')):
                                        if(automata(cadena)==1):
                                                guardarPalabra(cadena)
                                        cadena=""
                        else:
                                cadena=""
                        x+=1
                linea=f.readline()
        f.close()

def grafico():
```

```
root = Tk()
root.title('Automata␣Paridad␣Grafico') # Nombre de la ventana
canvas=Canvas(root,width=400,height=400)
canvas.pack()
canvas.create_arc(90,5,320,95, start=0, extent=180, style='arc') #arco
    1-2 arriba
canvas.create_arc(90,10,320,100, start=360, extent=-180, style='arc')
    #arco 1-2 abajo
canvas.create_arc(90,310,320,400, start=0, extent=180, style='arc') #
    arco 2-3 arriba
canvas.create_arc(90,310,320,400, start=360, extent=-180, style='arc')
    #arco 2-3 abajo
canvas.create_arc(310,95,400,320, start=90, extent=-180, style='arc')
    #arco 1-2 derecho
canvas.create_arc(310,95,400,320, start=90, extent=180, style='arc') #
    arco 1-2 izquierdo
canvas.create_arc(10,95,100,320, start=90, extent=-180, style='arc') #
    arco 2-4 derecho
canvas.create_arc(10,95,100,320, start=90, extent=180, style='arc') #
    arco 2-4 izquierdo
canvas.create_oval(10, 10, 100, 100, width=1, fill='green') #Primer
    Circulo
canvas.create_oval(7, 7, 103, 103)
canvas.create_oval(310, 10, 400, 100, width=1, fill='blue') #Segundo
    Circulo
canvas.create_oval(10, 310, 100, 400, width=1, fill='black') #Tercer
    circulo
canvas.create_oval(310, 310, 400, 400, width=1, fill='yellow') #Cuarto
    circulo
canvas.create_oval(95,30,105,40,fill="red") #Flecha 1-0
canvas.create_oval(305,70,315,80,fill="red") #Flecha 0-1
canvas.create_oval(95,330,105,340,fill="red") #Flecha 3-2
canvas.create_oval(305,370,315,380,fill="red") #Flecha 2-3
canvas.create_oval(35,98,45,108,fill="red") #Flecha 2-1
canvas.create_oval(70,305,80,315,fill="red") #Flecha 1-2
canvas.create_oval(335,98,345,108,fill="red") #Flecha 3-1
canvas.create_oval(370,305,380,315,fill="red") #Flecha 1-3
canvas.create_line(0,45,10,45)
canvas.create_oval(3,42,13,52,fill="red") #Flecha 0-0
estado=Label(root,text="Estado␣0",bg="green",font=("Helvetica",12),fg=
    "white") #ESTADO CERO
estado.place(x = 20, y = 40)
estado=Label(root,text="Estado␣1",bg="blue",font=("Helvetica",12),fg="
    white") #ESTADO UNO
estado.place(x = 320, y = 40)
estado=Label(root,text="Estado␣2",bg="black",font=("Helvetica",12),fg=
    "white") #ESTADO DOS
estado.place(x = 20, y = 340)
estado=Label(root,text="Estado␣3",bg="yellow",font=("Helvetica",12),fg
    ="black") #ESTADO TRES
estado.pack()
estado.place(x = 320, y = 340)
```

```
        cero=Label(root,text="0",font=("Helvetica",12)) # CERO 1 ARRIBA
        cero.place(x = 200, y = -2)
        cero=Label(root,text="0",font=("Helvetica",12)) # CERO 1 ABAJO
        cero.place(x = 200, y = 90)
        cero=Label(root,text="0",font=("Helvetica",12)) # CERO 2 ARRIBA
        cero.place(x = 200, y = 300)
        cero=Label(root,text="0",font=("Helvetica",12)) # CERO 1 ABAJO
        cero.pack()
        cero.place(x = 200, y = 380)
        uno=Label(root,text="1",font=("Helvetica",12)) # UNO 1 IZQUIERDA
        uno.place(x = 5, y = 192)
        uno=Label(root,text="1",font=("Helvetica",12)) # UNO 1 DERECHA
        uno.place(x = 90, y = 195)
        uno=Label(root,text="1",font=("Helvetica",12)) # UNO 2 IZQUIERDA
        uno.place(x = 300, y = 195)
        uno=Label(root,text="1",font=("Helvetica",12)) # UNO 2 DERECHA
        uno.pack()
        uno.place(x = 390, y = 195)
        root.mainloop()

def repetir(modo):
        opc=0
        while(opc!=2):
                if(modo==1):
                        opc=int(input("Deseas repetir? 1.Si 2.No \n ==> "))
                        if(opc==1):
                                print("Has seleccionado repetir")
                                manual()
                        elif(opc==2):
                                print ("Adios")
                        else:
                                print("Opcion incorrecta")
                if(modo==2):
                        print("Deseas repetir? 1.Si 2.No \n ==> ")
                        opc=randint(1,2)
                        if(opc==1):
                                print("Has seleccionado repetir")
                                auto()
                        if(opc==2):
                                print ("Adios")

def menu():
        crearArchivos()
        opcion=-1
        while(opcion>2 or opcion<1):
                print ("Selecciona una opcion")
                opcion=int(input("1. Manual\n2. Automatico \n3. Mostrar
                    Grafico \n4. Salir\n ==> "))
                if(opcion==1):
                        print("Opcion Manual Seleccionada")
                        manual()
                        repetir(opcion)
```

```
                elif(opcion==2):
                        print("Opcion␣Automatica␣Seleccionada")
                        auto()
                        repetir(opcion)
                elif(opcion==3):
                        grafico()
                elif(opcion==4):
                        break
                else:
                        print ("Opcion␣Equivocada")

menu()
```

4.3. Pruebas

Se realizaran dos pruebas para este programa uno en su forma manual y otra en su forma automática.

Modo Manual:

```
C:\Users\Luis\Desktop\Programas>python paridad.py
Selecciona una opcion
1. Manual
2. Automatico
3. Mostrar Grafico
4. Salir
 ==> 1
Opcion Manual Seleccionada
Ingresa la cadena ==> 1010
Cadena valida
Deseas repetir? 1.Si 2.No
 ==> 1
Has seleccionado repetir
Ingresa la cadena ==> 101
Cadena invalida
Deseas repetir? 1.Si 2.No
 ==> 2
Adios
```

Figura 15: Ejecución con una cadena valida y otra no.

De la ejecución con la cadena ingresada se genera el archivo 'cadenas.txt'.

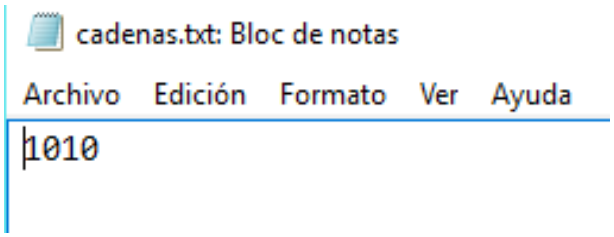

Figura 16: Archivo de salida con las cadenas validas de la ejecución.

Modo Automático: Para el modo automático se necesita tener un archivo el cual será leído por el programa para guardar en otro archivo las cadenas que son válidas.

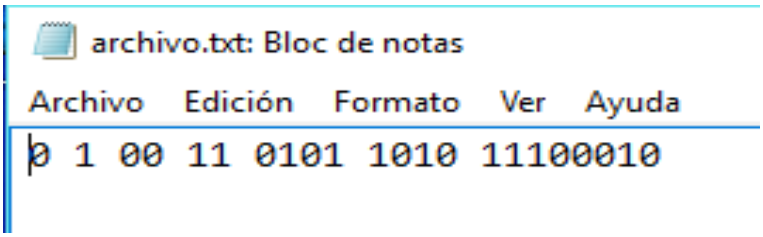

Figura 17: Archivo previamente generado para lectura.

Una vez con el archivo ya generado se ejecuta el programa en su modo automático.

```
C:\Users\Luis\Desktop\Programas>python paridad.py
Selecciona una opcion
1. Manual
2. Automatico
3. Mostrar Grafico
4. Salir
 ==> 2
Opcion Automatica Seleccionada
Deseas repetir? 1.Si 2.No
 ==>
Has seleccionado repetir
Deseas repetir? 1.Si 2.No
 ==>
Has seleccionado repetir
Deseas repetir? 1.Si 2.No
 ==>
Adios
```

Figura 18: Modo automatico con 3 repeticines

De la ejecución anterior se procesó la información dando como resultado:

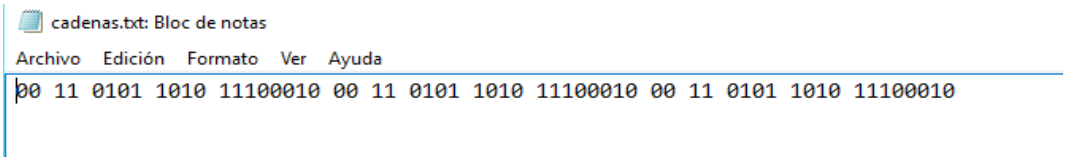

Figura 19: Archivo de salida con las cadenas validas, al ser el mismo archivo valida las mismas cadenas las veces que se repitió el proceso.

5. Protocolo

5.1. Descripción del problema

Se desea simular un protocolo, el programa será automático completamente. Simulara estar 'encendido' o 'apagado' y en caso de estar 'encendido' generara una trama que enviara a un segundo estado de validación y regresara las cadenas validadas. Para la validación utilizaremos el autómata de paridad ya que la trama que se generara será binaria de 4 bytes.

5.2. Código fuente

Este programa fue elaborado en el lenguaje Python, su código fuente se muestra a continuación:

Archivo: protocolo.py

```
from random import *
from time import *
from tkinter import *
def crearTrama():
        f = open("trama.txt","w")
        cadena=""
        f.write("(")
        for i in range(0,50):
                cadena=""
                for c in range(0,32):
                        caracter=str(randint(0,1))
                        cadena+=caracter
                f.write(cadena)
                if(i<49):
                        f.write(",")
        f.write(")")
        f.close()

def interruptor():
        return randint(0,1)

def crearArchivos():
        cadenas=open("cadenasValidas.txt","w")
        cadenas.close()
        archivo=open("estadosA.txt","w")
        archivo.close()

def guardarPalabra(cadena):
        filtro = open("cadenasValidas.txt","a")
        filtro.write(cadena+",")
        filtro.close()

def seccionarArchivo(num):
        filtro = open("cadenasValidas.txt","a")
```

```
        filtro.write("\nTrama " + str(num+1) + "\n")
        filtro.close()

def validar():
        cadena = ""
        estado = 0
        palabra = ""
        f = open("trama.txt","r")
        for cadena in f.readlines():
                for x in range(0,len(cadena)):
                        if(estado==0):
                                if(cadena[x]=='0'):
                                        palabra+=cadena[x]
                                        estado=1
                                elif(cadena[x]=='1'):
                                        palabra=palabra+cadena[x]
                                        estado=2
                                elif(cadena[x]==',' or cadena[x]==')'):
                                        guardarPalabra(palabra)
                                        palabra=""
                                        estado=0
                        elif(estado==1):
                                if(cadena[x]=='0'):
                                        palabra=palabra+cadena[x]
                                        estado=0
                                elif(cadena[x]=='1'):
                                        palabra=palabra+cadena[x]
                                        estado=3
                                elif(cadena[x]==','or cadena[x]==')'):
                                        palabra=""
                                        estado=0
                        elif(estado==2):
                                if(cadena[x]=='0'):
                                        palabra=palabra+cadena[x]
                                        estado=3
                                elif(cadena[x]=='1'):
                                        palabra=palabra+cadena[x]
                                        estado=0
                                elif(cadena[x]==','or cadena[x]==')'):
                                        palabra=""
                                        estado=0
                        elif(estado==3):
                                if(cadena[x]=='0'):
                                        palabra=palabra+cadena[x]
                                        estado=2
                                elif(cadena[x]=='1'):
                                        palabra=palabra+cadena[x]
                                        estado=1
                                elif(cadena[x]==','or cadena[x]==')'):
                                        palabra=""
                                        estado=0
        f.close()
```

```
def grafico():
        f=open("estadosA.txt","a")
        root =  Tk()
        root.title('Protocolo') # Nombre de la ventana
        canvas=Canvas(root,width=600,height=200)
        canvas.pack()
        canvas.create_arc(90,50,320,150, start=0, extent=180, style='arc') #
            arco 1-2 arriba
        canvas.create_arc(90,60,320,160, start=360, extent=-180, style='arc')
            #arco 1-2 abajo
        canvas.create_oval(10, 50, 100, 150, width=1, fill='yellow') #Primer
            Circulo
        canvas.create_oval(7, 47, 103, 153, width=1)
        canvas.create_oval(310, 50, 400, 150, width=1, fill='yellow') #Segundo
             Circulo
        canvas.create_oval(305,75,315,85,fill="cyan") #Flecha 0-1
        canvas.create_oval(90,120,100,130,fill="cyan") #Flecha 1-0
        estado=Label(root,text="Validacion",bg="black",font=("Helvetica",12),
            fg="white") #VALIDACION
        estado.place(x=170,y=150)
        estado=Label(root,text="Estado␣0",bg="yellow",font=("Helvetica",12)) #
            ESTADO CERO
        estado.place(x = 20, y = 80)
        estado=Label(root,text="Estado␣1",bg="yellow",font=("Helvetica",12)) #
            ESTADO UNO
        estado.pack()
        estado.place(x = 320, y = 80)
        encendido=Label(root,text="Encendido",font=("Helvetica",12))
        encendido.pack()
        encendido.place(x = 170, y = 40)
        estado=Label(root,text="Enviando␣Trama␣y␣Validando",font=("Helvetica"
            ,10)) #Enviando Trama
        estado.place(x = 430, y = 75)
        estado=Label(root,text="Interruptor",font=("Helvetica",10)) #
            Interruptor
        estado.pack()
        estado.place(x=430,y=45)
        canvas.create_oval(410,50,425,65,fill="blue") #Encendido/Apagado
        canvas.create_oval(410,80,425,95,width=1,fill="blue") #Enviando Trama
        estado=interruptor()
        crearArchivos()
        rep=0
        sleep(1)
        while(estado==1):
                root.update()
                canvas.create_oval(410,50,425,65,fill="red") #Encendido/
                    Apagado
                f.write("q0=Encendido\n")
                crearTrama()
                seccionarArchivo(rep)
```

```
                validar()
                canvas.create_oval(410,80,425,95,fill="red") #Enviando Trama
                f.write("q1=Generacion y Validacion de Trama\n")
                sleep(1)
                root.update()
                canvas.create_oval(410,80,425,95) #Enviando Trama
                estado=interruptor()
                root.update()
                canvas.create_oval(410,50,425,65,fill="blue") #Encendido/
                    Apagado
                canvas.create_oval(410,80,425,95,width=1,fill="blue") #
                    Enviando Trama
                sleep(1)
                rep+=1
        root.update()
        f.write("q0=Apagado")
        canvas.create_oval(410,50,425,65,fill="black") #Encendido/Apagado
        canvas.create_oval(410,80,425,95,width=1,fill="white") #Enviando Trama
        root.mainloop()

grafico()
```

5.3. Pruebas

Se realizara solo una prueba de este programa ya que todo el proceso es automático.

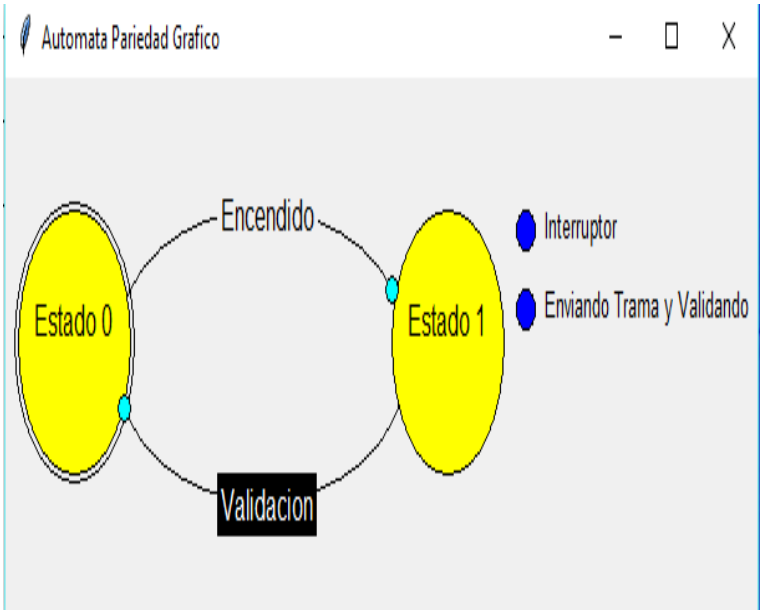

Figura 20: Grafico en estado inicial cuando no se encuentra apagado

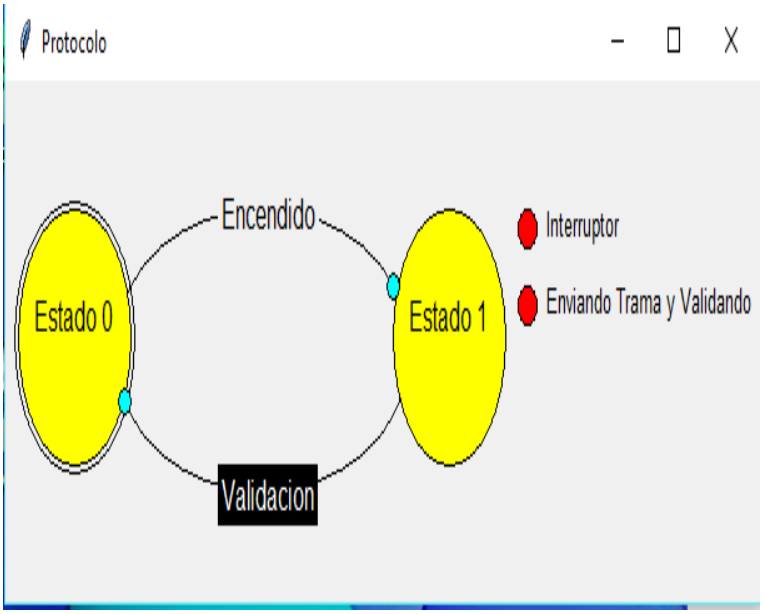

Figura 21: Grafico en estado de proceso.

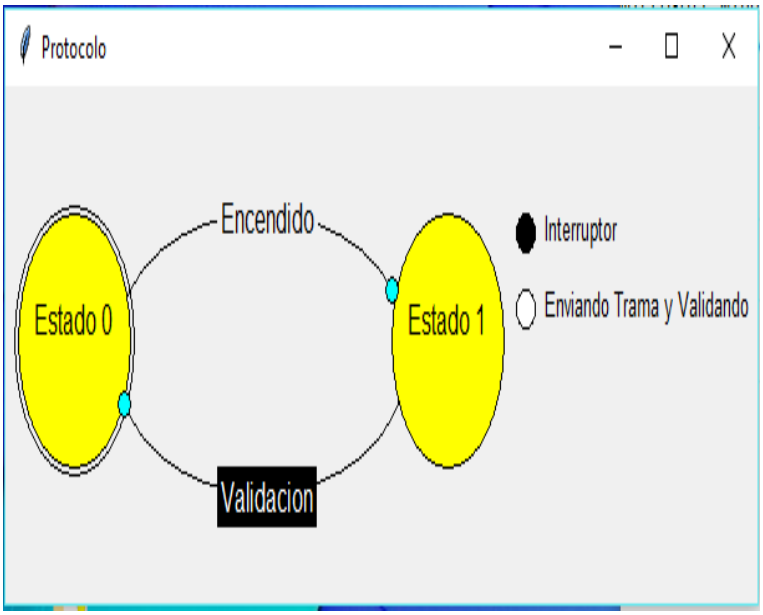

Figura 22: Grafico en estado final, cuando ya está apagado

De la ejecución resultó lo siguiente:

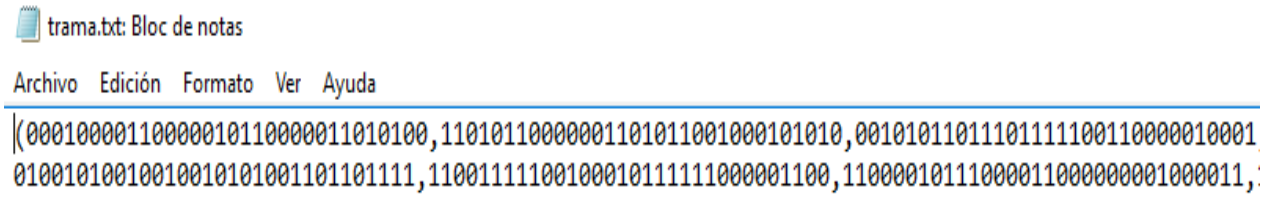

Figura 23: Archivo generado. (Trama de datos)

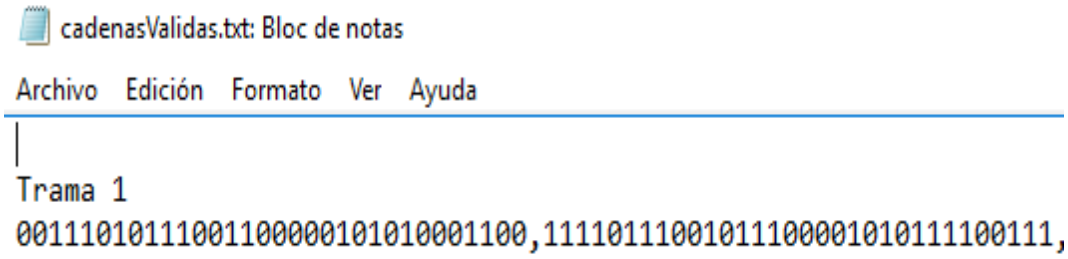

Figura 24: Archivo con las cadenas validas.

6. Autómata No Determinista

6.1. Descripción del problema

Se desea generar un programa que utilice un autómata para verificar si una cadena binaria termina en 01, al ser no determinista genera varios estados a la vez.

6.2. Código fuente

Este programa fue elaborado en el lenguaje Python, su código fuente se muestra a continuación:

Archivo: and.py

```
from random import *
from tkinter import *

def grafico():
        root =  Tk()
        root.title('Automata No Determinista Grafico') # Nombre de la ventana
        canvas=Canvas(root,width=800,height=200)
        canvas.pack()
        canvas.create_line(10,90,635,90) #Linea
        canvas.create_oval(40, 40, 140, 140, width=1, fill='blue') #Primer
            Circulo
        canvas.create_oval(340, 40, 440, 140, width=1, fill='blue') #Segundo
            Circulo
        canvas.create_oval(640, 40, 740, 140, width=1, fill='blue') #Tercer
            Circulo
        canvas.create_oval(635, 35, 745, 145, width=1) #Tercer Circulo
        canvas.create_arc(80,3,100,90, start=0, extent=180, style='arc') #
            Ciclo estado 1
        canvas.create_oval(40,85,30,95,fill='red')#Flecha inical
        canvas.create_oval(340,85,330,95,fill='red')#Flecha de 0-1
        canvas.create_oval(635,85,625,95,fill='red')#Flecha de 1-2
        canvas.create_oval(75,35,85,45,fill='red')#Flecha del ciclo
        estado=Label(root,text="Estado 0",bg="blue",font=("Helvetica",12),fg="
            white") #ESTADO CERO
        estado.place(x = 55, y = 75)
        estado=Label(root,text="Estado 1",bg="blue",font=("Helvetica",12),fg="
            white") #ESTADO UNO
        estado.place(x = 355, y = 75)
        estado=Label(root,text="Estado 2",bg="blue",font=("Helvetica",12),fg="
            white") #ESTADO DOS
        estado.place(x = 655, y = 75)
        numero=Label(root,text="0",font=("Helvetica",12))
        numero.place(x = 205, y = 60)
        numero=Label(root,text="1",font=("Helvetica",12))
        numero.place(x = 505, y = 60)
```

```
        numero=Label(root,text="0,1",font=("Helvetica",12))
        numero.place(x = 110, y = 10)
        root.mainloop()

def automata(cadena):
        lista = []
        lista.append([0])
        for caracter in cadena:
                if(caracter=='0'):
                        entraCero(lista)
                elif(caracter=='1'):
                        entraUno(lista)
                else:
                        return 0;
        imprimirMatriz(lista,cadena)
        if(lista[-1][-1]==2):
                lista=[]
                return 1
        else:
                lista=[]
                return 2

def entraCero(lista):
        listaAux=lista[-1][:]
        while(2 in listaAux):
                listaAux.insert(listaAux.index(2),'x')
                listaAux.pop(listaAux.index(2))
        while(1 in listaAux):
                listaAux.insert(listaAux.index(1),'x')
                listaAux.pop(listaAux.index(1))
        listaAux.append(1)
        lista.append(listaAux)

def entraUno(lista):
        listaAux=lista[-1][:]
        while(2 in listaAux):
                listaAux.insert(listaAux.index(2),'x')
                listaAux.pop(listaAux.index(2))
        while(1 in listaAux):
                listaAux.insert(listaAux.index(1),2)
                listaAux.pop(listaAux.index(1))
        lista.append(listaAux)

def imprimirMatriz(lista,cadena):
        i=0
        e=0
        while(i<len(lista)):
                if(e<len(lista)-1):
                        print(str(cadena[e])+"|"+str(lista[i]))
                else:
                        print("␣|"+str(lista[i]))
                e+=1
```

```
                i+=1

def menu():
        eleccion=0
        while (eleccion!=4):
                eleccion=input("Selecciona una opcion\n1.Manual\n2.Automatico\
                    n3.Mostrar Grafico\n4.Salir\n===> ")
                if(eleccion=='1'):
                        print("Opcion Manual seleccionada")
                        manual()
                elif(eleccion=='2'):
                        print("Opcion Automatica seleccionada")
                        auto()
                elif(eleccion=='3'):
                        print("Visualizar Grafico")
                        grafico()
                elif(eleccion=='4'):
                        print("Adios")
                        return 0
                else:
                        print("Opcion Invalida, Intentalo de nuevo")

def manual():
        cadena=input("Ingresa el numero que deseas validar:\n==> ")
        if(automata(cadena)==1):
                print("La cadena "+cadena+" es Valida")
        else:
                print("La cadena "+cadena+" NO es Valida")
        repetir(1)


def auto():
        cadena=generarCadena()
        if(automata(cadena)==1):
                print("La cadena "+cadena+" es Valida")
        else:
                print("La cadena "+cadena+" NO es Valida")
        repetir(2)

def generarCadena():
    longitud=randint(1,10)
    numero=''
    i=0
    while(i<longitud):
            numero+=choice(['0', '1'])
            i += 1
    print("El numero generado es: "+numero)
    return numero

def repetir(modo):
        rep=''
        if(modo==1):
```

```
                while(rep!='1' and rep!='2'):
                        rep=input("Deseas␣Repetir␣en␣este␣modo\n␣1.␣Si␣\n␣2.␣
                            No\n␣==>␣")
                        if(rep=='1'):
                                print("Seleccionaste␣repetir")
                                manual()
                        elif(rep=='2'):
                                print("Seleccionaste␣NO␣repetir")
                        else:
                                print("Opcion␣Invalida,␣Intentalo␣de␣nuevo")
        elif(modo==2):
                rep=choice(['1', '2'])
                if(rep=='1'):
                        print("Seleccionaste␣repetir")
                        auto()
                elif(rep=='2'):
                        print("Seleccionaste␣NO␣repetir")

menu()
```

6.3. Pruebas

Se realizaran dos pruebas para este programa uno en su forma manual y otra en su forma automática.

Modo Manual:

```
C:\Users\Luis\Desktop>python and.py
Selecciona una opcion
1.Manual
2.Automatico
3.Mostrar Grafico
4.Salir
===> 1
Opcion Manual seleccionada
Ingresa el numero que deseas validar:
==> 101001
1|[0]
0|[0]
1|[0, 1]
0|[0, 2]
0|[0, 'x', 1]
1|[0, 'x', 'x', 1]
 |[0, 'x', 'x', 2]
La cadena 101001 es Valida
```

Figura 25: Ejecución con una cadena valida

Modo Automático: Para el modo automático se genera una cadena aleatoria y se repite n veces segùn el random lo genera.

```
C:\Users\Luis\Desktop>python and.py
Selecciona una opcion
1.Manual
2.Automatico
3.Mostrar Grafico
4.Salir
===> 2
Opcion Automatica seleccionada
El numero generado es: 10100010
1|[0]
0|[0]
1|[0, 1]
0|[0, 2]
0|[0, 'x', 1]
0|[0, 'x', 'x', 1]
1|[0, 'x', 'x', 'x', 1]
0|[0, 'x', 'x', 'x', 2]
 |[0, 'x', 'x', 'x', 'x', 1]
La cadena 10100010 NO es Valida
Seleccionaste NO repetir
Selecciona una opcion
1.Manual
2.Automatico
3.Mostrar Grafico
4.Salir
===> 4
```

Figura 26: Modo automatico