

# Reporte de prácticas del primer periodo

Ian Mendoza Jaimes

2CM4  
Teoría Computacional  
Profesor Genaro Juaréz Martínez

# Índice

<b>1. Universo de palabras</b>	<b>3</b>
1.1. Descripción del problema . . . . .	3
1.2. Código fuente . . . . .	3
1.3. Pruebas . . . . .	7
<b>2. Números primos</b>	<b>9</b>
2.1. Descripción del problema . . . . .	9
2.2. Código . . . . .	9
2.3. Pruebas . . . . .	13
<b>3. Palabras que terminan en ere</b>	<b>16</b>
3.1. Descripción del problema . . . . .	16
3.2. Código . . . . .	16
3.3. Pruebas . . . . .	22
<b>4. Paridad de ceros y unos</b>	<b>25</b>
4.1. Descripción del problema . . . . .	25
4.2. Código . . . . .	25
4.3. Pruebas . . . . .	31
<b>5. Protocolo</b>	<b>33</b>
5.1. Descripción del problema . . . . .	33
5.2. Código . . . . .	33
5.3. Pruebas . . . . .	39
<b>6. Cadenas que terminan en 01</b>	<b>42</b>
6.1. Descripción del problema . . . . .	42
6.2. Código . . . . .	42
6.3. Pruebas . . . . .	47
<b>7. Referencias</b>	<b>49</b>

# 1. Universo de palabras

Los alfabetos se definen como un conjunto finito, no vacío de símbolos. Comúnmente se denotan con la letra griega  $\Sigma$ . Al definir un alfabeto, podemos tener acceso a seleccionar un conjunto finito de estos símbolos, a esto se le llama cadena o string. [1]

## 1.1. Descripción del problema

Se necesita realizar un programa, que dado un alfabeto binario  $\Sigma = \{0, 1\}$  en este caso, sea capaz de calcular e imprimir en un archivo de texto todas las palabras que puedan ser formadas por un alfabeto binario, es decir,

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

Sin embargo, tiene limitaciones, pues es imposible calcular algo infinito, el programa deberá imprimir todas las combinaciones de las palabras con la longitud:  $0 \leq n \leq 1000$ .

## 1.2. Código fuente

El programa para este problema fue escrito en el lenguaje C. Esto debido a su alta velocidad de procesamiento, la cual es bastante necesaria si la longitud de las palabras es grande. El código utilizado para la resolución del problema se muestra a continuación:

Archivo: main.c

```
#include "alfabeto.h"

int main(int argc, char const *argv[]) {
    char seleccion = '1';
    int continuar = 1;
    int continuar_modalidad = 1;
    int n = 0;
    char seleccion_tiro = '_';

    srand(time(NULL));

    while(continuar == 1){
        printf("%\n", "Seleccione la modalidad:\n1.- Automatico\n2.- Manual\n3.- Salir");
        scanf(" %", &seleccion);

        if(seleccion == '1' || seleccion == '2'){
            continuar_modalidad = 1;
            while(continuar_modalidad == 1) {
                if(seleccion == '1'){
                    n = 1 + (rand() %5);
                }
                else{
                    printf("%", "Ingrese un n:");
                    scanf("%d", &n);
                }
            }
        }
    }
}
```

```

        iniciar_programa(n);

        if(seleccion == '2'){
            printf("%s\n", "Desea_ingresar_otra_n?:s/n");
            scanf("%c", &seleccion_tiro);
            if(seleccion_tiro == 's' || seleccion_tiro == 'S'){
                continuar_modalidad = 1;
            }
            else{
                continuar_modalidad = 0;
            }
        }
        else{
            continuar_modalidad = rand() %2;
            printf("%d\n", continuar_modalidad);
        }
    }
    else{
        return 1;
    }
}

return 0;
}
}

```

Archivo: alfabeto.c

```

#include "alfabeto.h"

int iniciar_programa(int n){
    int tamano_alfabeto = 2;
    char * alfabeto = NULL;
    alfabeto = (char *)malloc(tamano_alfabeto * sizeof(char));
    iniciar_alfabeto(&alfabeto, tamano_alfabeto);

    obtener_cadenas(alfabeto, tamano_alfabeto, n);

    return 1;
}

int obtener_cadenas(char *alfabeto, int tamano, int n){
    int i;
    int j;
    int * cadena_temporal = NULL;
    int salir = 0;

    FILE *archivo = NULL;

    archivo = fopen("cadenas.txt", "w");
    if (archivo == NULL) {

```

```

        printf("%s\n", "Error_al_abrir_el_archivo");
        exit(0);
    }

fputs("= ", archivo);

for(i = 1; i <= n; i++){
    cadena_temporal = (int*)calloc(i, sizeof(int));
    while(salir == 0){
        escribir_palabra(archivo, cadena_temporal, alfabeto, i);
        for(j = i - 1; j > -1; j--){
            *(cadena_temporal + j) = *(cadena_temporal + j) + 1;
            if( *(cadena_temporal + j) > (tamanio -1)){
                *(cadena_temporal + j) = 0;
            }
            else {break;}
        }
        if(j == -1){
            free(cadena_temporal);
            break;
        }
    }
    printf("Va_en_n=%d\n", i);
}
fputs("} ", archivo);
fclose(archivo);

return 1;
}

int escribir_palabra(FILE *archivo, int * cadena_temporal, char * alfabeto, int tamanio
){
    int i;
    fputs(", ", archivo);
    for(i = 0; i < tamanio; i++){
        fputc(*(alfabeto + *(cadena_temporal + i)), archivo);
    }
    return 1;
}

int iniciar_alfabeto(char **alfabeto, int tamanio){
    int i;
    for(i = 0; i < tamanio; i++){
        *(*alfabeto + i) = i + '0';
    }
    return 1;
}
}

```

Archivo: alfabeto.h

```

#ifndef _ALFABETO_H_
#define _ALFABETO_H_

#include <stdio.h>
#include <string.h>

```

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

int iniciar_programa();
int iniciar_alfabeto(char **, int);
int escribir_palabra(FILE *, int *, char*, int);
int obtener_cadenas(char *, int , int);

#endif
```

### **1.3. Pruebas**

En cuanto a las pruebas, a continuación se mostrarán una serie de imágenes capturadas al momento de ejecutar el programa. Los resultados arrojados por el programa anterior son:

Para la selección en modo automático:

```
[Ianonsio-2:alfabetos ianMJ$ ./a.out
Seleccione la modalidad:
1.- Automatico
2.- Manual
3.- Salir
1
Se ha seleccionado un n = 24
Va en n = 1
Va en n = 2
Va en n = 3
Va en n = 4
Va en n = 5
Va en n = 6
Va en n = 7
Va en n = 8
Va en n = 9
Va en n = 10
Va en n = 11
Va en n = 12
Va en n = 13
Va en n = 14
Va en n = 15
Va en n = 16
Va en n = 17
Va en n = 18
Va en n = 19
Va en n = 20
Va en n = 21
Va en n = 22
Va en n = 23
Va en n = 24
0
Seleccione la modalidad:
1.- Automatico
2.- Manual
3.- Salir
```

Figura 1: Selección de un  $n = 24$  de forma automática.

Figura 2: Texto producido por un  $n = 24$ .

Para la selección en modo manual:

```
Seleccione la modalidad:  
1.- Automatico  
2.- Manual  
3.- Salir  
2  
Ingrese un n:  15  
Va en n = 1  
Va en n = 2  
Va en n = 3  
Va en n = 4  
Va en n = 5  
Va en n = 6  
Va en n = 7  
Va en n = 8  
Va en n = 9  
Va en n = 10  
Va en n = 11  
Va en n = 12  
Va en n = 13  
Va en n = 14  
Va en n = 15  
Desea ingresar otra n?: s/n  
s  
Ingrese un n:  4  
Va en n = 1  
Va en n = 2  
Va en n = 3  
Va en n = 4  
Desea ingresar otra n?: s/n  
n  
Seleccione la modalidad:  
1.- Automatico  
2.- Manual  
3.- Salir
```

Figura 3: El resultado de escoger manualmente a n.

```
00, 11111110000101, 11111110000110, 11111110000111, 111111100001000, 111111100001001,
11, 11111110001100, 11111110001101, 11111110001110, 11111110001111, 111111100010000,
10, 11111110010011, 11111110010100, 11111110010101, 11111110010110, 11111110010111,
01, 11111110011010, 11111110011011, 11111110011100, 11111110011101, 11111110011110,
00, 11111110100001, 11111110100010, 11111110100011, 111111101000100, 111111101000101,
11, 11111110101000, 11111110101001, 11111110101010, 11111110101011, 11111110101100,
10, 11111110101111, 11111110110000, 11111110110001, 11111110110010, 11111110110011,
01, 11111110110110, 11111110110111, 11111110111000, 11111110111001, 11111110111010,
00, 11111110111101, 11111110111110, 11111110111111, 11111111000000, 11111111000001,
11, 1111111000100, 1111111000101, 1111111000110, 1111111000111, 1111111001000,
10, 1111111001011, 1111111001100, 1111111001101, 1111111001110, 1111111001111,
01, 1111111010010, 1111111010011, 1111111010100, 1111111010101, 1111111010110,
00, 1111111011001, 1111111011010, 1111111011011, 1111111011100, 1111111011101,
11, 1111111000000, 1111111000001, 1111111000010, 1111111000011, 1111111001000,
10, 1111111001011, 1111111010000, 1111111010001, 1111111010100, 1111111010111,
01, 11111110110110, 11111110110111, 11111111110000, 11111111110001, 11111111110010,
00, 11111111110101, 11111111110110, 11111111110111, 11111111111000, 11111111111001,
11, 11111111111100, 11111111111110, 11111111111111, 11111111111111, 0000000000000000,
0010, 00000000000011, 000000000000100, 000000000000101, 000000000000110,
1000, 00000000001001, 00000000001010, 00000000001011, 00000000001100,
1110, 00000000001111, 000000000010000, 000000000010001, 000000000010010,
0100, 000000000010101, 000000000010110, 000000000010111, 000000000011000,
1010, 000000000011011, 000000000011100, 000000000011101, 000000000011110,
0000, 000000000010001, 0000000000100010, 0000000000100011, 0000000000100100,
0110, 000000000010011, 0000000000101000, 0000000000101001, 0000000000101010,
1100, 000000000010101, 000000000010110, 000000000010111, 0000000000110000,
0010, 000000000011001, 0000000000110100, 0000000000110101, 0000000000110110,
1000, 0000000000111001, 0000000000111010, 0000000000111011, 0000000000111100,
1110, 0000000000111111, 0000000000100000, 0000000000100001, 0000000000100010,
```

Figura 4: Texto producido por  $n = 15$ .

## 2. Números primos

Un número primo es aquel que solo puede ser dividido entre sí mismo o la unidad. A lo largo de la historia mucha gente los ha estudiado y han propuesto numerosas maneras de encontrarlos. Algunos ejemplos de ello son la criba de Eratóstenes o la criba de Euler. En esta sección presentaremos un programa capaz de calcular todos los números primos en un intervalo dado.

### 2.1. Descripción del problema

Realizar un programa capaz de encontrar todos los números primos dentro de un intervalo dado por:  $0 \leq n \leq 1000$ . Además, deberá convertir dichos números de su representación decimal a su representación binaria y guardarlos en un archivo de texto. Después, se debe proceder a graficar la cantidad de ceros y unos que aparecen dependiendo de  $n$ .

### 2.2. Código

El programa, en este caso, fue escrito en Python. Fue seleccionado este lenguaje por la facilidad que presenta al manejar estructuras de datos como listas, pilas, etc. A continuación se presenta el código utilizado:

Archivo: main\_primos.py

```
from metodos import *
from random import random

def main():
    archivo = open("primos.txt", "w")
    archivo.write("")
    archivo.close()
    seguir = True
    while seguir:
        print("\n\nSelecciona el modo en que se ejecutara el programa:\n1.-\nAutomatico\n2.-\nManual\n3.-\nSalir")
        try:
            seleccion = int(input())
            if seleccion > 0 and seleccion <= 2:
                iniciar_programa(seleccion)
            elif seleccion == 3:
                return 1;
            else:
                raise Exception()
        except Exception as e:
            print("Por favor ingrese un dato valido.")

def iniciar_programa(seleccion=1):
    numeros_primos = []
    numeros_primos_binarios = []
    numero_ceros_unos = []
```

```

n = 0
continuar = True
archivo = None

while continuar:
    if seleccion == 2:
        n = ingresar_datos("\nIngrese un numero limite (0 < n <= 1000): ")
    else:
        n = int(random() * 1000)
        print("\nFue seleccionado un n = ", n)

    archivo = open("primos.txt", "a")

    numeros_primos = encontrar_primos(numeros_primos, n)
    numeros_primos_binarios = convertir_primos_a_binarios(numeros_primos, archivo, n)
    numero_ceros_unos = contar_ceros_undos(numeros_primos_binarios)

    archivo.close()

    imprimir_ceros_undos(numero_ceros_undos, numeros_primos, n)

    if seleccion == 1:
        continuar = int(random() * 100) % 2
    else:
        eleccion = ingresar_datos("\nDesea ingresar otra n? \n1. - Si \n2. - No \n")
        if eleccion != 1:
            continuar = False


def ingresar_datos(texto):
    while True:
        try:
            dato_n = int(input(texto))
            if dato_n > 0 and dato_n <= 1000:
                return dato_n
            else:
                raise Exception()
        except Exception as e:
            print("Por favor ingrese un dato valido", e)


def imprimir_ceros_undos(numero_ceros_undos, numeros_primos, n):
    if n == 1:
        return -1

    contador = 0
    i = 0
    print('[ ', end=' ')
    try:
        while numeros_primos[i] <= n:
            print(str(numeros_primos[i]) + ', ', end=' ')
            i += 1
    except Exception as e:
        pass

```

```

print('')

print("[_numero_primo, _numero_de_unos, _numero_de_ceros_]")
while contador < len(numero_ceros_unos):
    print("[" , numeros_primos[contador], " , " , numero_ceros_unos[contador][1] , " , " ,
          numero_ceros_unos[contador][0] , " ]" , end=" , ")
    contador += 1

main()

```

Archivo: metodos\_primos.py

```

def encontrar_primos(numeros_primos, n):
    if n == 1:
        return numeros_primos

    if len(numeros_primos) == 0:
        numeros_primos.append(2)

    #   for x in range(2,n+1):
    x = 2
    if n >= 2:
        for y in numeros_primos:
            if x%y == 0:
                es_primo = False
                break

    x = 3
    while x <= n:
        es_primo = True

        for y in numeros_primos:
            if x%y == 0:
                es_primo = False
                break

        if es_primo:
            numeros_primos.append(x)

        x += 2

    return numeros_primos

def convertir_primos_a_binarios(numeros_primos, archivo, n):
    numeros_primos_binarios = []
    for x in numeros_primos:
        if x <= n:
            numeros_primos_binarios.append(bin(x)[2:])
            archivo.write(", " + bin(x)[2:])
        else:
            break

    return numeros_primos_binarios

```

```
def contar_ceros_unos(numeros_primos_binarios):
    numero_ceros_unos = []
    contador_ceros = 0
    contador_unos = 0
    for x in numeros_primos_binarios:
        contador_unos = 0
        contador_ceros = 0
        for y in x:
            if y == '0':
                contador_ceros += 1
            else:
                contador_unos += 1
        numero_ceros_unos.append([contador_ceros, contador_unos])
    return numero_ceros_unos
```

## 2.3. Pruebas

A continuación, se mostrarán las imágenes de los resultados de ejecutar el programa en consola.

Para la selección de modo automático:

```
numeros_primos — Python main.py — 136x41

Selección el modo en que se ejecutará el programa:
1.- Automático
2.- Manual
3.- Salir
1

Fue seleccionado un n = 913
[ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 1

[ numero primo, numero de unos, numero de ceros ]

[ 2, 1, 1 ], [ 3, 2, 0 ], [ 5, 2, 1 ], [ 7, 3, 0 ], [ 11, 3, 1 ], [ 13, 3, 1 ], [ 17, 2, 3 ], [ 19, 3, 2 ], [ 23, 4, 1 ], [ 29, 4, 1 ], [ 31, 5, 0 ], [ 37, 3, 3 ], [ 41, 3, 3 ], [ 43, 4, 2 ], [ 47, 5, 1 ], [ 53, 4, 2 ], [ 59, 5, 1 ], [ 61, 5, 1 ], [ 67, 3, 4 ], [ 71, 4, 3 ], [ 73, 3, 4 ], [ 79, 5, 2 ], [ 83, 4, 3 ], [ 89, 4, 3 ], [ 97, 3, 4 ], [ 103, 5, 2 ], [ 107, 5, 2 ], [ 109, 5, 2 ], [ 113, 4, 3 ], [ 127, 7, 0 ], [ 131, 3, 5 ], [ 137, 3, 5 ], [ 139, 4, 4 ], [ 149, 4, 4 ], [ 151, 5, 3 ], [ 157, 5, 3 ], [ 163, 4, 4 ], [ 167, 5, 3 ], [ 173, 5, 3 ], [ 179, 5, 3 ], [ 181, 5, 3 ], [ 191, 7, 1 ], [ 193, 3, 5 ], [ 197, 4, 4 ], [ 199, 5, 3 ], [ 211, 5, 3 ], [ 223, 7, 1 ], [ 227, 5, 3 ], [ 229, 5, 3 ], [ 233, 5, 3 ], [ 239, 7, 1 ], [ 241, 5, 3 ], [ 251, 7, 1 ], [ 257, 2, 7 ], [ 263, 4, 5 ], [ 269, 4, 5 ], [ 271, 5, 5 ], [ 277, 4, 5 ], [ 281, 4, 5 ], [ 283, 5, 4 ], [ 293, 4, 5 ], [ 307, 5, 4 ], [ 311, 6, 3 ], [ 313, 5, 4 ], [ 331, 5, 4 ], [ 337, 4, 5 ], [ 347, 6, 3 ], [ 349, 6, 3 ], [ 353, 4, 5 ], [ 359, 6, 3 ], [ 367, 7, 2 ], [ 373, 6, 6 ], [ 379, 7, 2 ], [ 383, 8, 1 ], [ 389, 4, 5 ], [ 397, 5, 4 ], [ 401, 4, 5 ], [ 409, 5, 4 ], [ 419, 5, 4 ], [ 421, 5, 4 ], [ 431, 7, 2 ], [ 433, 5, 4 ], [ 439, 7, 2 ], [ 443, 7, 2 ], [ 449, 4, 5 ], [ 457, 5, 4 ], [ 461, 6, 3 ], [ 463, 7, 2 ], [ 467, 6, 3 ], [ 479, 8, 1 ], [ 487, 7, 2 ], [ 491, 7, 2 ], [ 499, 7, 2 ], [ 503, 8, 1 ], [ 509, 8, 1 ], [ 511, 7, 2 ], [ 523, 4, 6 ], [ 541, 5, 5 ], [ 547, 4, 6 ], [ 557, 5, 5 ], [ 563, 5, 5 ], [ 569, 5, 5 ], [ 571, 6, 4 ], [ 577, 7, 2 ], [ 579, 6, 4 ], [ 587, 5, 5 ], [ 593, 4, 6 ], [ 599, 6, 4 ], [ 601, 5, 5 ], [ 607, 7, 3 ], [ 613, 5, 5 ], [ 617, 5, 5 ], [ 619, 6, 4 ], [ 631, 7, 3 ], [ 641, 3, 7 ], [ 643, 4, 6 ], [ 647, 5, 5 ], [ 653, 5, 5 ], [ 659, 5, 5 ], [ 661, 5, 5 ], [ 673, 7, 2 ], [ 677, 5, 5 ], [ 683, 6, 4 ], [ 691, 6, 4 ], [ 701, 7, 3 ], [ 709, 5, 5 ], [ 719, 7, 3 ], [ 727, 7, 3 ], [ 733, 7, 3 ], [ 739, 6, 4 ], [ 743, 7, 3 ], [ 751, 8, 2 ], [ 757, 7, 3 ], [ 761, 7, 3 ], [ 769, 3, 7 ], [ 773, 4, 6 ], [ 787, 5, 5 ], [ 797, 6, 4 ], [ 809, 5, 5 ], [ 811, 6, 4 ], [ 821, 6, 4 ], [ 823, 7, 3 ], [ 827, 7, 3 ], [ 829, 7, 3 ], [ 839, 6, 4 ], [ 853, 6, 4 ], [ 857, 6, 4 ], [ 859, 7, 3 ], [ 863, 8, 2 ], [ 877, 7, 3 ], [ 881, 6, 4 ], [ 883, 7, 3 ], [ 887, 8, 2 ], [ 907, 6, 4 ], [ 911, 7, 3 ],
```

Figura 5: El resultado de seleccionar la modalidad automática.

```
, 10, 11, 101, 111, 1011, 1101, 10001, 10011, 10111, 11101, 11111, 100101, 101001,  
101011, 101111, 110101, 111011, 111101, 1000011, 1000111, 1001001, 1001111, 1010011,  
1011001, 1100001, 1100101, 1100111, 1101011, 1101101, 1110001, 1111111, 10000011,  
10001001, 10001011, 10010101, 10010111, 10011101, 10100011, 10100111, 10101101,  
10110011, 10110101, 10111111, 11000001, 11000101, 11000111, 11010011, 11011111,  
11100011, 11100101, 11101001, 11101111, 11110001, 11111011, 100000001, 100000111,  
100001001, 100001111, 100010101, 100011001, 100011011, 100100101, 100110011,  
100110111, 100111001, 100111101, 101001011, 101010001, 101011011, 101011101,  
101100001, 101100111, 101101111, 101110101, 101111011, 101111111, 110000101,  
110001101, 110010001, 110011001, 110100011, 110100101, 110101111, 110110001,  
110110111, 110111011, 111000001, 111001001, 111001101, 111001111, 111010001,  
111011001, 111011101, 111100011, 111110101, 1000001001, 1000010011,  
1000011011, 1000101011, 1000111001, 1001001011, 1001011001, 1001101001, 1001110011,  
1001111011, 1001111101, 1010000001, 1010000011, 1010000111, 1010001111, 1010010011,  
1010011001, 1010011101, 1010100001, 1010100011, 1010100101, 1010101011, 1010110011,  
1010111001, 1011001111, 1011010111, 1011011101, 1011100011, 1011100111, 1011101111,  
1011110011, 1011111001, 1100000001, 1100000011, 1100000101, 1100001101, 1100011101,  
1100101011, 1100110011, 1100111011, 1100111101, 1101000011, 1101000111, 1101001011,  
1101011001, 1101011101, 1101011111, 1101100101, 1101110001, 1101111001, 1101111011,  
1110001011, 1110001111
```

Figura 6: El resultado en texto de seleccionar la modalidad automática.

Para la selección manual:

```
Ianonsio-2:numeros_primos ianMJ$ python3 main.py
numeros_primos — Python main.py — 136x41

[ Ianonsio-2:numeros_primos ianMJ$ python3 main.py

Selecciona el modo en que se ejecutará el programa:
1.- Automático
2.- Manual
3.- Salir
2

Ingrese un número límite ( 0 < n <= 1000): 3
[ 2, 3, ]
[ numero primo, numero de unos, numero de ceros ]
[ 2, 1, 1 ], [ 3, 2, 0 ],
¿Desea ingresar otra n?
1.- Si
2.- No
1

Ingrese un número límite ( 0 < n <= 1000): 100
[ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ]
[ numero primo, numero de unos, numero de ceros ]
[ 2, 1, 1 ], [ 3, 2, 0 ], [ 5, 2, 1 ], [ 7, 3, 0 ], [ 11, 3, 1 ], [ 13, 3, 1 ], [ 17, 2, 3 ], [ 19, 3, 2 ], [ 23, 4, 1
], [ 29, 4, 1 ], [ 31, 5, 0 ], [ 37, 3, 3 ], [ 41, 3, 3 ], [ 43, 4, 2 ], [ 47, 5, 1 ], [ 53, 4, 2 ], [ 59, 5, 1 ], [ 61
, 5, 1 ], [ 67, 3, 4 ], [ 71, 4, 3 ], [ 73, 3, 4 ], [ 79, 5, 2 ], [ 83, 4, 3 ], [ 89, 4, 3 ], [ 97, 3, 4 ],
¿Desea ingresar otra n?
1.- Si
2.- No
n
Por favor ingrese un dato válido invalid literal for int() with base 10: 'n'
¿Desea ingresar otra n?
1.- Si
2.- No
2
```

Figura 7: El resultado de seleccionar la modalidad manual.

```
[ primos.txt ▾
, 10, 11, 10, 11, 101, 111, 1011, 1101, 10001, 10011, 10111, 11101, 11111, 100101,
101001, 101011, 101111, 110101, 111011, 111101, 1000011, 1000111, 1001001, 1001111,
1010011, 1011001, 1100001
```

Figura 8: El resultado en texto de seleccionar la modalidad manual.

Se graficó la relación entre ceros y unos de los números primos cuando estos eran convertidos a binario. El resultado de evaluar un  $n = 1000$ , es el siguiente:

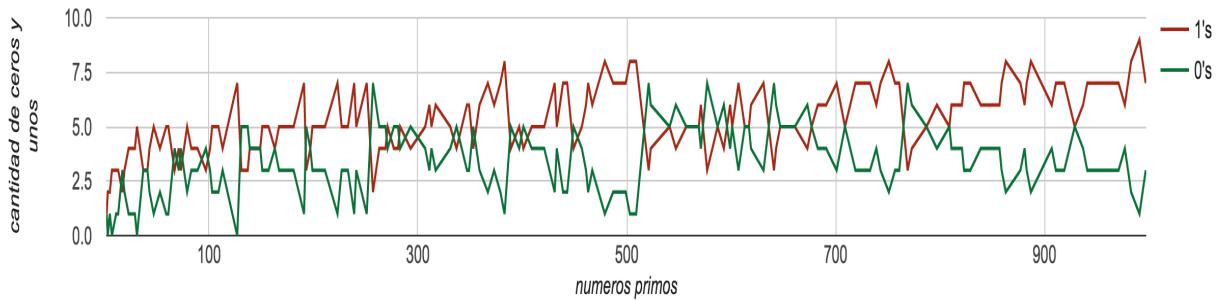


Figura 9: Gráfico de ceros y unos.

### 3. Palabras que terminan en ere

Los autómatas son una forma de evaluar cadenas a través de una serie de estados. En concreto los autómatas determinísticos utilizan estados que solo pueden ser seguidos por otro estado. En esta sección se plantea un problema en el cual es muy conveniente utilizar un autómata para resolverlo y sirve como una introducción a esta teoría tan amplia.

#### 3.1. Descripción del problema

Se tiene que elaborar un programa que pueda evaluar un texto y determinar cuales son las palabras que terminan con *ere*. Además, deberá decir en que linea se encuentran. Para la realización de este programa, se debe de utilizar un autómata finito determinístico.

#### 3.2. Código

El modelo del autómata utilizado para resolver este programa es el siguiente:

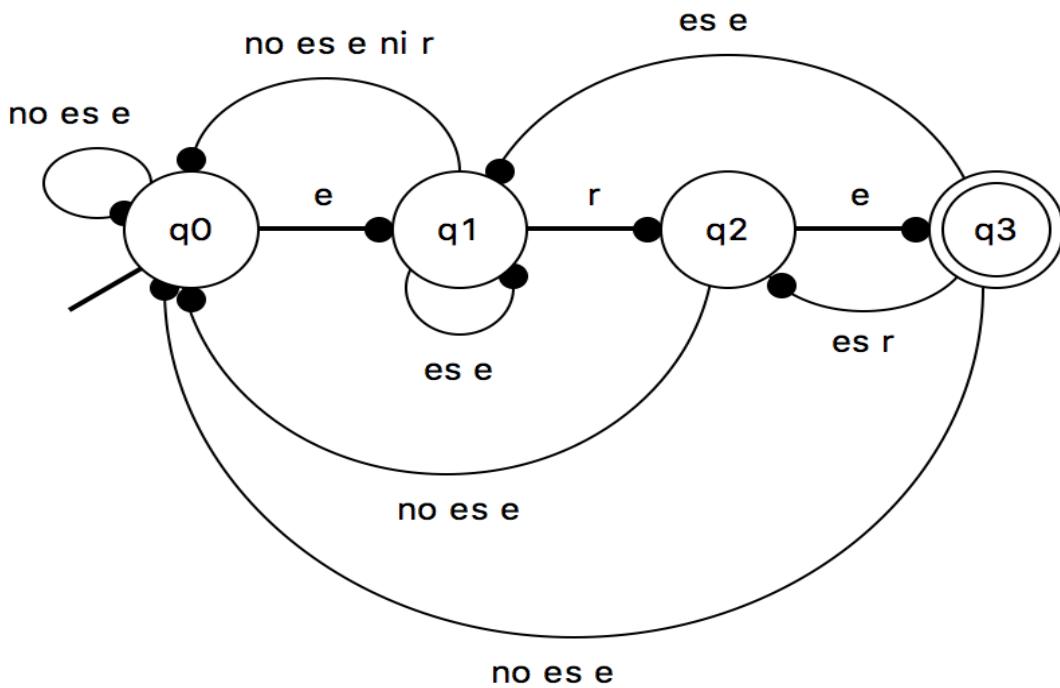


Figura 10: El autómata utilizado para este problema.

El lenguaje utilizado para este programa fue Python en su versión 3.5. El código para resolver el problema es el siguiente:

Archivo: main\_ere.py

```
from automata import obtener_palabras, graficar_automata
from ctypes import *

def main():
    seguir = True
    archivo = open('historial_ere.txt', 'w')
    archivo.close()
    while seguir:
        try:
            palabras_aceptadas = []
            texto = ""
            archivo = None

            opcion = imprimir_menu()

            if opcion == 1:
                while True:
                    print("\n\nIngrese un texto (dos veces la tecla enter para salir):\n")
                    tem = ''
                    t = ''
                    texto = []
                    contador = 0
                    while True:
                        t = input()
                        if t == '':
                            break
                        texto.append(t)

                    palabras_aceptadas = leer_texto(texto)
                    imprimir_palabras_aceptadas(palabras_aceptadas, opcion)

                    texto = input("\n\nDesea ingresar otra cosa? [s/n]: ")
                    if (texto != 's') and (texto != 'S'):
                        break

            elif opcion == 2:
                texto = input("\n\nIngrese el nombre de un archivo:\n")
                archivo = open(texto, "r")
                palabras_aceptadas = leer_texto(archivo)
                archivo.close()
                imprimir_palabras_aceptadas(palabras_aceptadas, opcion)

            elif opcion == 3:
                print("Sera utilizado el archivo heart.txt")
                archivo = open("heart.txt", "r")
                palabras_aceptadas = leer_texto(archivo)
                imprimir_palabras_aceptadas(palabras_aceptadas, opcion)

            elif opcion == 4:
                graficar_automata()

        else:
            return 0

    except Exception as e:
        print("Uuups!, parece que tuvimos un problema:", e)
```

```

    return 1

def imprimir_menu():
    seguir = True
    while seguir:
        try:
            opcion = int(input("\n\nIngrese la opcion que desea:\n1.- Ingresar texto\n2.- Ingresar un archivo\n3.- Automatico\n4.- Ver grafico del automata\n5.- Salir\n"))
            return opcion
        except Exception as e:
            print("Por favor, ingrese un dato valido.")

def leer_texto(texto):
    aceptadas = None
    palabras_aceptadas = []
    contador = 1
    for linea in texto:
        aceptadas = obtener_palabras(linea)
        palabras_aceptadas.append([contador, aceptadas])
        contador += 1

    return palabras_aceptadas

def imprimir_palabras_aceptadas(palabras_aceptadas, seleccion):
    print("\n\n")
    contador = 0
    for x in palabras_aceptadas:
        if len(x[1]) > 0:
            print("")
            print("Linea ", str(x[0]), ", palabras_aceptadas:", end=' ')
            for palabra in x[1]:
                print(palabra[0], '(No. palabra:' + str(palabra[1]) + ') ', end=' , ')
    print()

main()

```

### Archivo: automata\_ere.py

```

from tkinter import *
import time

def obtener_palabras(texto):
    archivo = open('historial_ere.txt', 'a')
    estado = 0
    palabras_aceptadas = []
    temporal = ''
    letra_auxiliar = ''
    ascii_axiliar = 0
    contador = 0

```

```

estado_axiliar = ''
for x in texto:
    letra_auxiliar = x
    letra_auxiliar = letra_auxiliar.lower()
    ascii_axiliar = ord(letra_auxiliar)
    if (ascii_axiliar >= 97 and ascii_axiliar <= 122) or (ascii_axiliar >= 224 and
        ascii_axiliar <= 255):
        print(letra_auxiliar)
        estado_axiliar = 'Delta(q'+str(estado)+', '+x+')-->'
        archivo.write(estado_axiliar)
        estado = automata(estado, letra_auxiliar)
        temporal += x
    else:
        contador += 1
        estado_axiliar = 'q'+str(estado)+'\n\n'
        archivo.write(estado_axiliar)
        if estado == 3:
            palabras_aceptadas.append([temporal, contador])
        temporal = ','
        estado = 0

    estado_axiliar = 'q'+str(estado)+'\n\n'
    archivo.write(estado_axiliar)
    archivo.close()
    print(estado)
    if estado == 3:
        contador += 1
        palabras_aceptadas.append([temporal, contador])

return palabras_aceptadas

def automata(estado, letra_auxiliar):
    if estado == 0:
        return estado_cero(letra_auxiliar)
    elif estado == 1:
        return estado_uno(letra_auxiliar)
    elif estado == 2:
        return estado_dos(letra_auxiliar)
    elif estado == 3:
        return estado_tres(letra_auxiliar)
    else:
        return -1

def estado_cero(letra):
    if (ord(letra) >= 232 and ord(letra) <= 235) or letra == 'e':
        return 1
    else:
        return 0

def estado_uno(letra):
    if (ord(letra) >= 232 and ord(letra) <= 235) or letra == 'e':
        return 1

```

```

elif letra == 'r':
    return 2
else:
    return 0

def estado_dos(letra):
    if (ord(letra) >= 232 and ord(letra) <= 235) or letra == 'e':
        return 3
    else:
        return 0

def estado_tres(letra):
    if (ord(letra) >= 232 and ord(letra) <= 235) or letra == 'e':
        return 1
    elif letra == 'r':
        return 2
    else:
        return 0

def graficar_automata():
    raiz = Tk()
    raiz.title('Automata')
    raiz.geometry('500x350')
    canvas = Canvas(raiz, width=600, height=410, bg='white')
    canvas.place(x=0,y=0)
    canvas.pack(expand=YES, fill=BOTH)

    x = 90
    y = 100
    r = 50
    numero_circulos = 4
    opciones = [ 'e' , 'r' , 'e' ]
    canvas.create_line(x-20, y+r*1.2, x+.2*r, y+.8*r, width=2, fill='black')
    for i in range(numero_circulos):
        dibujos_especificos(canvas, i, x, y)
        x += r+50

    x = 90
    for i in range(numero_circulos):
        canvas.create_oval(x, y, x+r, y+r, width=1, fill='white')
        widget = Label(canvas, text='q'+str(i), fg='black', bg='white')
        widget.pack()
        canvas.create_window(x+.5*r, y+.5*r, window=widget)

        if i < numero_circulos-1:
            canvas.create_line(x+r, y+(r/2), x+r+50, y+(r/2), width=2, fill='black')
            widget = Label(canvas, text=opciones[i], fg='black', bg='white')
            widget.pack()
            canvas.create_window(x+r+25, y+(r/2)-15, window=widget)
            canvas.create_oval(x+r+40, y+(r/2)-5, x+r+50, y+(r/2)+5, width=1, fill='black')

```

```

if i == numero_circulos -1:
    canvas.create_oval(x+5, y+5, x+r-5, y+r-5, width=1, fill='white')
x += r+50

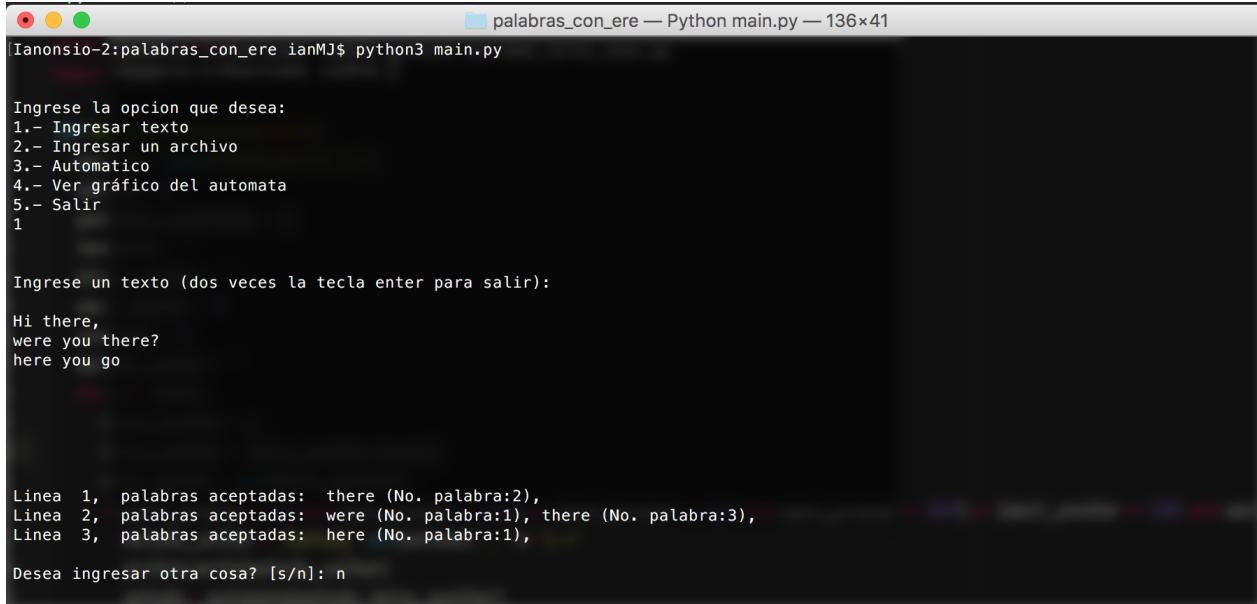
raiz.mainloop()

def dibujos_especificos(canvas, i, x, y):
    if i == 0:
        xy = x+10, y-20, x+30, y+10
        canvas.create_oval(x-30,y-10, x+10,y+20, width=1)
        canvas.create_oval(x-5, y+13, x+5, y+23, width=1, fill='black')
        widget = Label(canvas, text='no_es_e', fg='black', bg='white')
        widget.pack()
        canvas.create_window(x-20, y-25, window=widget)
    elif i == 1:
        xy = x-75, y-40, x+25, y+40
        canvas.create_arc(xy, start=0, extent=180, style='arc')
        canvas.create_oval(x-80, y-10, x-70, y, width=1, fill='black')
        canvas.create_oval(x+5,y+30, x+45,y+70, width=1)
        widget = Label(canvas, text='no_es_eni_r', fg='black', bg='white')
        widget.pack()
        canvas.create_window(x-20, y-55, window=widget)
        canvas.create_oval(x+40,y+40, x+50,y+50, width=1, fill='black')
        widget = Label(canvas, text='es_e', fg='black', bg='white')
        widget.pack()
        canvas.create_window(x+25, y+85, window=widget)
    elif i == 2:
        xy = x-180, y-70, x+20, y+130
        canvas.create_arc(xy, start=0, extent=-180, style='arc')
        widget = Label(canvas, text='no_es_e', fg='black', bg='white')
        widget.pack()
        canvas.create_window(x-75, y+145, window=widget)
        canvas.create_oval(x-180,y+50, x-170,y+60, width=1, fill='black')
    elif i == 3:
        xy = x-285, y-100, x+20, y+200
        canvas.create_arc(xy, start=0, extent=-180, style='arc')
        canvas.create_oval(x-290,y+45, x-280,y+55, width=1, fill='black')
        widget = Label(canvas, text='no_es_e', fg='black', bg='white')
        widget.pack()
        canvas.create_window(x-140, y+215, window=widget)
        xy = x-170, y+120, x+20, y-50
        canvas.create_arc(xy, start=0, extent=180, style='arc')
        canvas.create_oval(x-165,y+5, x-155,y-5, width=1, fill='black')
        widget = Label(canvas, text='es_e', fg='black', bg='white')
        widget.pack()
        canvas.create_window(x-80, y-65, window=widget)
        xy = x-65, y, x+16, y+60
        canvas.create_arc(xy, start=0, extent=-180, style='arc')
        widget = Label(canvas, text='es_r', fg='black', bg='white')
        widget.pack()
        canvas.create_window(x-24, y+73, window=widget)
        canvas.create_oval(x-60,y+44, x-50,y+54, width=1, fill='black')

```

### 3.3. Pruebas

Los siguientes, son los resultados que se obtuvieron del programa anterior.



```
Ianonsio-2:palabras_con_ere ianMJ$ python3 main.py

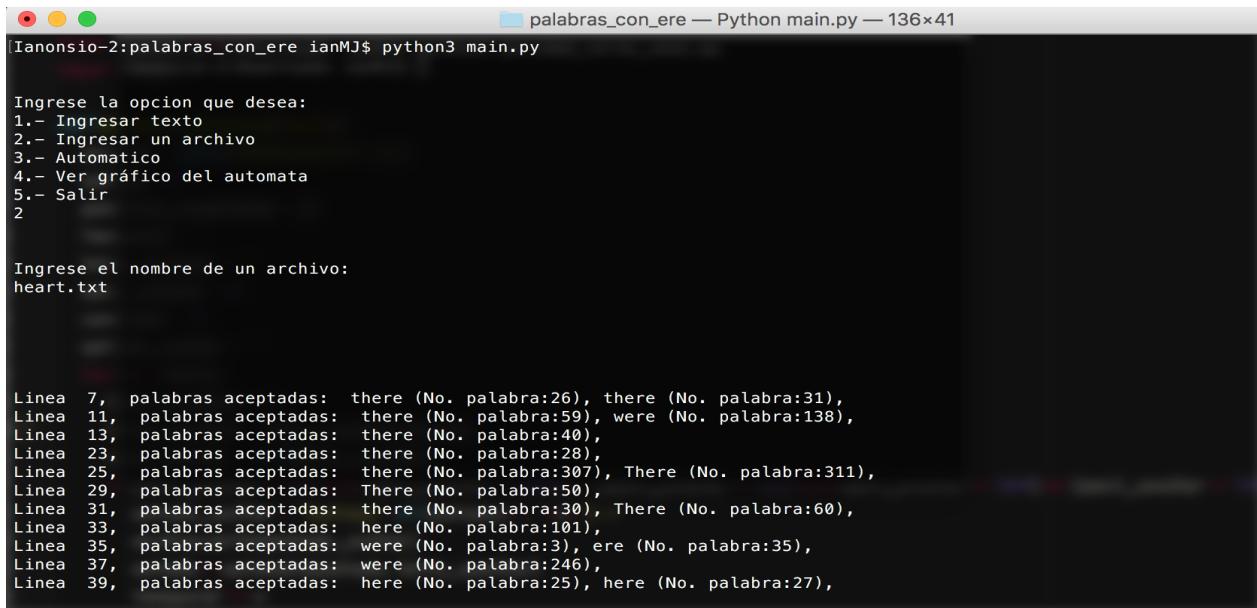
Ingrese la opcion que desea:
1.- Ingresar texto
2.- Ingresar un archivo
3.- Automatico
4.- Ver grafico del automata
5.- Salir
1

Ingrese un texto (dos veces la tecla enter para salir):
Hi there,
were you there?
here you go

Linea 1, palabras aceptadas: there (No. palabra:2),
Linea 2, palabras aceptadas: were (No. palabra:1), there (No. palabra:3),
Linea 3, palabras aceptadas: here (No. palabra:1),

Desea ingresar otra cosa? [s/n]: n
```

Figura 11: Un texto ingresado de forma manual.



```
Ianonsio-2:palabras_con_ere ianMJ$ python3 main.py

Ingrese la opcion que desea:
1.- Ingresar texto
2.- Ingresar un archivo
3.- Automatico
4.- Ver grafico del automata
5.- Salir
2

Ingrese el nombre de un archivo:
heart.txt

Linea 7, palabras aceptadas: there (No. palabra:26), there (No. palabra:31),
Linea 11, palabras aceptadas: there (No. palabra:59), were (No. palabra:138),
Linea 13, palabras aceptadas: there (No. palabra:40),
Linea 23, palabras aceptadas: there (No. palabra:28),
Linea 25, palabras aceptadas: there (No. palabra:307), There (No. palabra:311),
Linea 29, palabras aceptadas: There (No. palabra:50),
Linea 31, palabras aceptadas: there (No. palabra:30), There (No. palabra:60),
Linea 33, palabras aceptadas: here (No. palabra:101),
Linea 35, palabras aceptadas: were (No. palabra:3), ere (No. palabra:35),
Linea 37, palabras aceptadas: were (No. palabra:246),
Linea 39, palabras aceptadas: here (No. palabra:25), here (No. palabra:27),
```

Figura 12: Un archivo ingresado de forma manual.

```

palabras_con_ere — Python main.py — 136x41
Ianonsio-2:palabras_con_ere ianMJ$ python3 main.py

Ingrese la opcion que desea:
1.- Ingresar texto
2.- Ingresar un archivo
3.- Automatico
4.- Ver gráfico del automata
5.- Salir
3
Sera utilizado el archivo heart.txt

:
:

Linea 7, palabras aceptadas: there (No. palabra:26), there (No. palabra:31),
Linea 11, palabras aceptadas: there (No. palabra:59), were (No. palabra:138),
Linea 13, palabras aceptadas: there (No. palabra:40),
Linea 23, palabras aceptadas: there (No. palabra:28),
Linea 25, palabras aceptadas: there (No. palabra:307), There (No. palabra:311),
Linea 29, palabras aceptadas: There (No. palabra:50),
Linea 31, palabras aceptadas: there (No. palabra:30), There (No. palabra:60),
Linea 33, palabras aceptadas: here (No. palabra:101),
Linea 35, palabras aceptadas: were (No. palabra:3), ere (No. palabra:35),
Linea 37, palabras aceptadas: were (No. palabra:246),
Linea 39, palabras aceptadas: here (No. palabra:25), here (No. palabra:27),

```

Figura 13: Un archivo analizado de forma automática por el programa.

```

historial.txt ▾
Delta(q0, H)-->Delta(q0, i)-->q0

Delta(q0, t)-->Delta(q0, h)-->Delta(q0, e)-->Delta(q1, r)-->Delta(q2, e)-->q3
q0

Delta(q0, w)-->Delta(q0, h)-->Delta(q0, e)-->Delta(q1, r)-->Delta(q2, e)-->q3
Delta(q0, y)-->Delta(q0, o)-->Delta(q0, u)-->q0

Delta(q0, t)-->Delta(q0, h)-->Delta(q0, e)-->Delta(q1, r)-->Delta(q2, e)-->q3
q0

Delta(q0, h)-->Delta(q0, e)-->Delta(q1, r)-->Delta(q2, e)-->q3
Delta(q0, y)-->Delta(q0, o)-->Delta(q0, u)-->q0

Delta(q0, g)-->Delta(q0, o)-->q0

Delta(q0, T)-->Delta(q0, H)-->Delta(q0, E)-->q1

Delta(q0, T)-->Delta(q0, E)-->Delta(q1, L)-->Delta(q0, L)-->q0

Delta(q0, T)-->Delta(q0, A)-->Delta(q0, L)-->Delta(q0, E)-->q1

Delta(q0, H)-->Delta(q0, E)-->Delta(q1, A)-->Delta(q0, R)-->Delta(q0, T)-->q0
q0

```

Figura 14: El historial del programa.

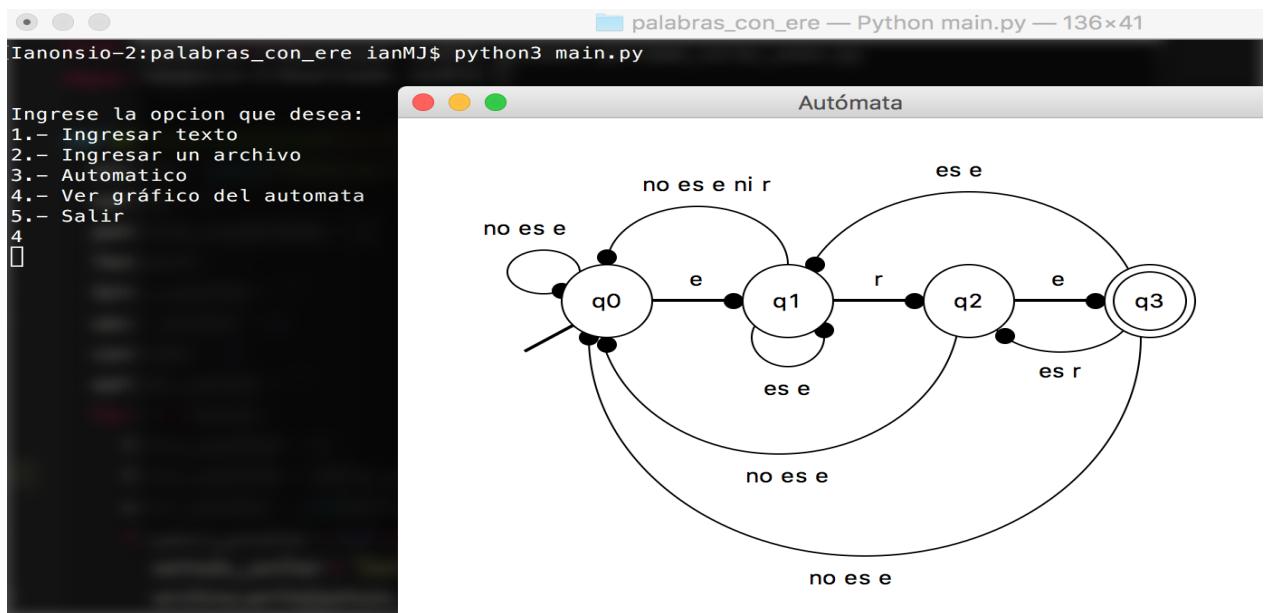


Figura 15: El programa imprimiendo el autómata.

## 4. Paridad de ceros y unos

Los autómatas, son una manera de poder evaluar cadenas y determinar si cumplen con ciertas características que nosotros deseamos. En este caso se plantea un problema para practicar esta funcionalidad.

### 4.1. Descripción del problema

Se tiene que crear un programa que determine si una cadena de números binarios tiene una cantidad par de ceros y de unos. Para la resolución de este problema, se tiene que utilizar un autómata finito determinístico.

### 4.2. Código

El modelo de autómata que se utilizo para resolver este problema es el siguiente:

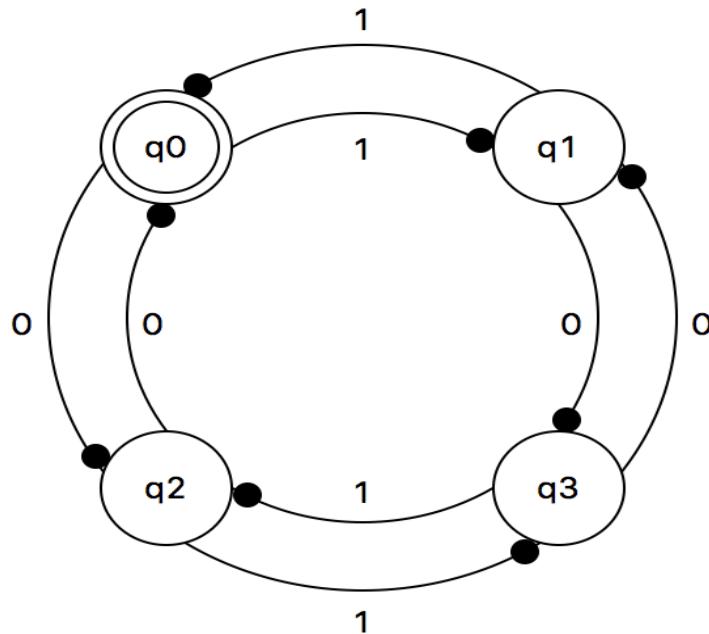


Figura 16: El autómata utilizado para este problema.

El lenguaje utilizado para este programa fue Python en su versión 3.5. El código para resolver el problema es el siguiente:

Archivo: main\_paridad.py

```
from metodos import obtener_paridades, ejecuta_diagrama
from random import random
```

```

def main():
    archivo = open("historial_paridad.txt", "w")
    try:
        while True:
            opcion = imprimir_menu()

            if opcion == 1:
                ejecuta_manual()
            elif opcion == 2:
                ejecutaAutomatico()
            elif opcion == 3:
                ejecutaDiagrama()
            else:
                return 1

    except Exception as e:
        print('Uups, parece que hubo un problema:', e)

def ejecuta_manual():
    while True:
        texto = input('\n\nIntroduzca un numero binario: ')
        es_permitida = obtener_paridades(texto)
        opcion = ''

        if es_permitida:
            print('La palabra ', texto, ' es permitida')
        else:
            print('La palabra ', texto, ' no es permitida')

        opcion = input("Quiere ingresar otro numero? [s/n] ")
        if opcion != 's' and opcion != 'S':
            return 1

def ejecutaAutomatico():
    limite = 0
    aleatorio = 0
    numero = ''
    i = 0
    seguir = True
    es_permitida = False
    continuar = 0

    while seguir:
        limite = int(random() * 2000) % 1001
        i = 0
        while i < limite:
            aleatorio = int(random() * 10) % 2
            if aleatorio == 0:
                numero += '0'

```

```

        else:
            numero += '1'
            i += 1

    print( '\n\nSe _utilizara _la _palabra: ', numero)
    es_permitida = obtener_paridades(numero)
    if es_permitida:
        print('La _palabra _', numero, '_es _permitida')
    else:
        print('La _palabra _', numero, '_no _es _permitida')

    continuar = int(random() * 10) %2
    if continuar == 0:
        seguir = False
    else:
        seguir = True

def imprimir_menu():
    while True:
        try:
            opcion = int(input("\n\n\nEscoja _lo _que _desea: \n1._ Manual\n2._ Automatico\n3._ Ver _diagrama\n4._ Salir\n\n"))
            if opcion >= 1 and opcion <= 4:
                return opcion
            else:
                raise Exception
        except Exception as e:
            print('Por _favor , _intrudusca _un _dato _valido ')
    main()

```

Archivo: metodos\_paridad.py

```

from tkinter import *

def obtener_paridades(texto):
    archivo = open("historial_paridad.txt", 'a')
    historial = ''
    palabras_permitidas = []
    temporal = ''
    estado = 0
    for x in texto:
        historial = '(q'+str(estado)+', '+x+')-->'
        print(historial, end='')
        archivo.write(historial)
        estado = automata(estado, x)

    historial = 'q'+str(estado)
    print(historial, end=' ')
    archivo.write(historial)
    archivo.write('\n')
    archivo.close()
    if estado == 0:

```

```

        return True
else:
    return False

def automata(estado, letra):
    if estado == 0:
        return estado_cero(letra)
    elif estado == 1:
        return estado_uno(letra)
    elif estado == 2:
        return estado_dos(letra)
    else:
        return estado_tres(letra)

def estado_cero(letra):
    if letra == '1':
        return 1
    elif letra == '0':
        return 2
    else:
        return -1

def estado_uno(letra):
    if letra == '1':
        return 0
    elif letra == '0':
        return 3
    else:
        return -1

def estado_dos(letra):
    if letra == '1':
        return 3
    elif letra == '0':
        return 0
    else:
        return -1

def estado_tres(letra):
    if letra == '1':
        return 2
    elif letra == '0':
        return 1
    else:
        return -1

def ejecuta_diagrama():
    ventana = Tk()
    ventana.title("Automata")
    ventana.geometry("500x350")

```

```

canvas = Canvas(ventana, width=600, height=410, bg='white')
canvas.place(x=0,y=0)
canvas.pack(expand=YES, fill=BOTH)

x = 140
y = 70
r = 50

dibujos_especificos(x,y, canvas, r)

for i in range(2):
    canvas.create_oval(x, y, x+r, y+r, width=1, fill="white")
    canvas.create_oval(x, y+r+100, x+r, y+2*r+100, width=1, fill="white")

    widget = Label(canvas, text='q'+str(i), fg='black', bg='white')
    widget.pack()
    canvas.create_window(x+25, y+(r/2), window=widget)

    widget = Label(canvas, text='q'+str(i+2), fg='black', bg='white')
    widget.pack()
    canvas.create_window(x+25, y+2*r+75, window=widget)

if i == 0:
    canvas.create_oval(x+5, y+5, x+r-5, y+r-5, width=1)

x += r + 100

ventana.mainloop()

def dibujos_especificos(x, y, canvas, r):
    canvas.create_oval(x-20, y-20, x+r+170, y+2*r+120)
    canvas.create_oval(x+10, y+10, x+r+140, y+2*r+90)

    canvas.create_oval(x+32, y-7, x+42, y+3, width=1, fill="black")
    canvas.create_oval(x+140, y+17, x+150, y+27, width=1, fill="black")
    canvas.create_oval(x+173, y+140, x+183, y+150, width=1, fill="black")
    canvas.create_oval(x+50.5, y+173, x+60.5, y+183, width=1, fill="black")
    canvas.create_oval(x+17.5, y+50, x+27.5, y+60, width=1, fill="black")
    canvas.create_oval(x-7.3, y+156, x+2.7, y+166, width=1, fill="black")
    canvas.create_oval(x+156.5, y+198, x+166.5, y+208, width=1, fill="black")
    canvas.create_oval(x+197.5, y+33, x+207.5, y+43, width=1, fill="black")

    widget = Label(canvas, text='1', fg='black', bg='white')
    widget.pack()
    canvas.create_window(x+100, y-31, window=widget)
    widget = Label(canvas, text='1', fg='black', bg='white')
    widget.pack()
    canvas.create_window(x+100, y+26, window=widget)

    widget = Label(canvas, text='1', fg='black', bg='white')
    widget.pack()
    canvas.create_window(x+100, y+177, window=widget)
    widget = Label(canvas, text='1', fg='black', bg='white')

```

```
widget.pack()
canvas.create_window(x+100, y+234, window=widget)

widget = Label(canvas, text='0', fg='black', bg='white')
widget.pack()
canvas.create_window(x-30, y+103, window=widget)
widget = Label(canvas, text='0', fg='black', bg='white')
widget.pack()
canvas.create_window(x+20, y+103, window=widget)

widget = Label(canvas, text='0', fg='black', bg='white')
widget.pack()
canvas.create_window(x+180, y+103, window=widget)
widget = Label(canvas, text='0', fg='black', bg='white')
widget.pack()
canvas.create_window(x+230, y+103, window=widget)
```

### **4.3. Pruebas**

A continuación se presentan imágenes de los resultados del programa corriendo:

```
[Ianonsio-2:paridad ianMJ$ python3 main.py

Escoja lo que desea:
1.- Manual
2.- Automático
3.- Ver diagrama
4.- Salir

1

Introduzca un número binario: 01010011
(q0, 0)-->(q2, 1)-->(q3, 0)-->(q1, 1)-->(q0, 0)-->(q2, 0)-->(q0, 1)-->(q1, 1)-->q0 La palabra 01010011 es permitida
Quiere ingresar otro número? [s/n] s

Introduzca un número binario: 0111
(q0, 0)-->(q2, 1)-->(q3, 1)-->(q2, 1)-->q3 La palabra 0111 no es permitida
Quiere ingresar otro número? [s/n]
```

Figura 17: Números ingresados manualmente.

Figura 18: Modo automático.

Figura 19: El historial del programa.

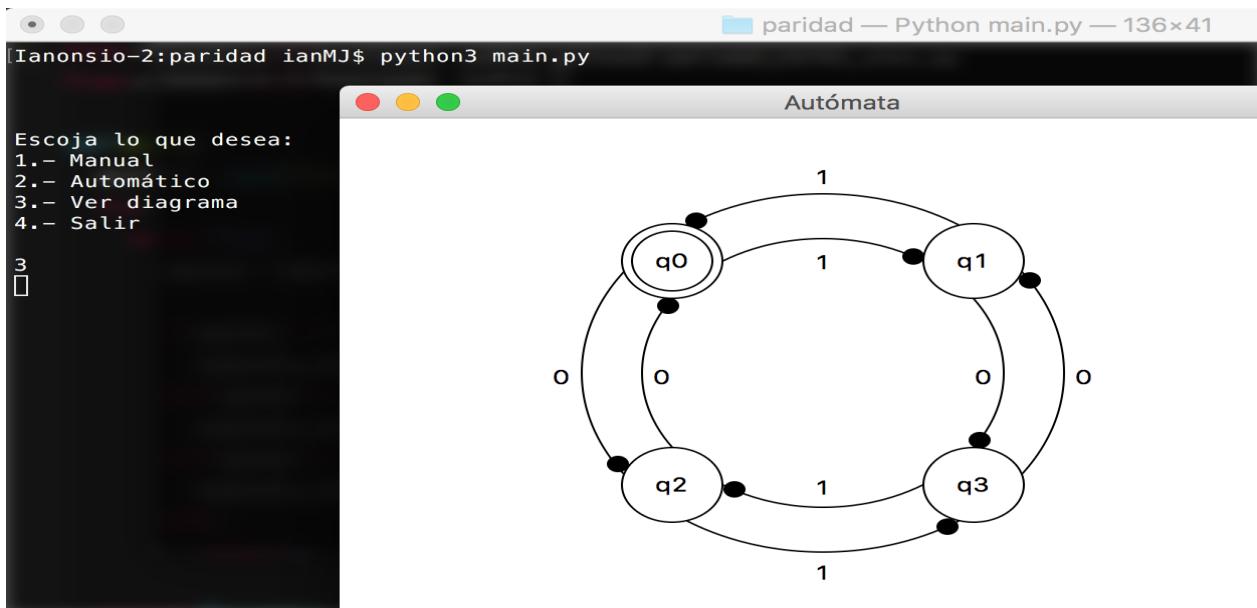


Figura 20: El programa imprimiendo el autómata.

## 5. Protocolo

Un protocolo es una serie de reglas para facilitar un fin. En este caso se presenta un ejemplo de un bosquejo de un protocolo para la transferencia de datos utilizando autómatas.

### 5.1. Descripción del problema

Se necesita crear un programa que genere 50 cadenas aleatoriamente para que después de tres segundos, que sería un tiempo de espera suficiente en este caso, comience a filtrar las cadenas que son validas de acuerdo a un criterio establecido. Para este problema, el criterio sera la paridad de ceros y unos, exactamente el mismo autómata de la sección anterior.

### 5.2. Código

El diagrama del protocolo es el siguiente [2]

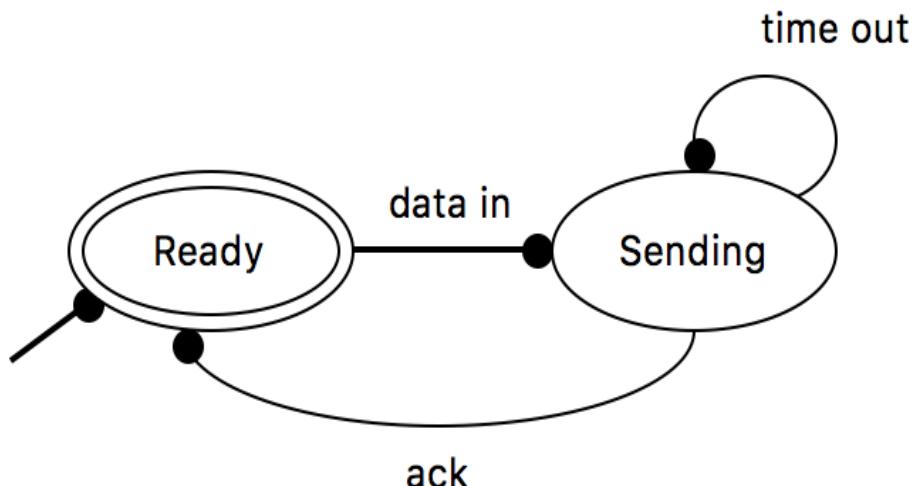


Figura 21: El autómata utilizado para este problema.

El lenguaje utilizado para este programa fue Python en su versión 3.5. El código para resolver el problema es el siguiente:

Archivo: main\_protocolo.py

```

from protocolo import *
from grafico_automata import dibujar_grafico_automata
from random import random

def main():
    try:
        while True:
            opcion = imprimir_menu()

            if opcion == 1:
                nombre_archivo = "palabras_generadas.txt"
                archivo = open(nombre_archivo, 'w')
                archivo.close()

                while True:
                    estado_encendido = int(random() * 100) % 3
                    estado = 0
                    if estado_encendido == 1:
                        print('\nEstado de la maquina: Encendido')
                    else:
                        print('\nEstado de la maquina: Apagado')
                    return 1
                while True:
                    estado = automata_protocolo(estado)
                    if estado == 0:
                        break
            elif opcion == 2:
                dibujar_grafico_automata()
            else:
                return 1

    except Exception as e:
        print('Uups, parece que hubo un problema:', e)

def imprimir_menu():
    seguir = True
    while seguir:
        try:
            opcion = int(input("\n\nIngrese la opcion que desea: \n1.- Iniciar \n2.- Ver grafico \n3.- Salir \n"))
            return opcion
        except Exception as e:
            print("Por favor, ingrese un dato valido.")

main()

```

Archivo: protocolo.py

```

from random import random
import time

```

```

from tkinter import *

def automata_protocolo(estado):
    if estado == 0:
        return crear_palabras()
    elif estado == 1:
        return enviar_datos()
    else:
        return -1

def enviar_datos():
    retrazar(3)
    return evaluar('palabras_generadas.txt')

def crear_palabras():
    nombre_archivo = 'palabras_generadas.txt'
    archivo = open(nombre_archivo, "a")
    longitud = 32
    aleatorio_numero = 0
    for x in range(0, 50):
        i = 0
        par = int(random() * 10) % 2
        while i < longitud:
            aleatorio_numero = int(random() * 10) % 2
            if par == 0:
                archivo.write(str(aleatorio_numero)*2)
                i += 2
            else:
                archivo.write(str(aleatorio_numero))
                i += 1
        archivo.write("\n")
    archivo.close()

    return 1

def retrazar(tiempo):
    time.sleep(tiempo)

def evaluar(nombre_archivo):
    archivo = open(nombre_archivo, 'r')
    aceptadas = open("palabras_aceptadas.txt", "w")
    historial = open("historial_protocolo.txt", "w")
    historial_auxiliar = ''
    palabras_permitidas = []
    temporal = ''
    estado = 0
    letra = ''
    while True:
        letra = archivo.read(1)
        if not letra:
            break

```

```

if letra != '_':
    historial_auxiliar = 'Delta(q'+str(estado)+','+letra+')-->'
    estado = automata(estado, letra)
    temporal += letra
else:
    historial_auxiliar = 'q'+str(estado)+'\n\n\n'
    if estado == 0:
        palabras_permitidas.append(temporal)
        aceptadas.write(temporal)
        aceptadas.write('\n')
    estado = 0
    temporal = ''

historial.write(historial_auxiliar)

archivo.close()
aceptadas.close()
historial.close()
print("Palabras_aceptadas:\n",palabras_permitidas)
return 0

def automata(estado, x):
    if estado == 0:
        return estado_cero(x)
    elif estado == 1:
        return estado_uno(x)
    elif estado == 2:
        return estado_dos(x)
    elif estado == 3:
        return estado_tres(x)
    else:
        return -1

def estado_cero(letra):
    if letra == '1':
        return 1
    elif letra == '0':
        return 2
    else:
        return -1

def estado_uno(letra):
    if letra == '1':
        return 0
    elif letra == '0':
        return 3
    else:
        return -1

def estado_dos(letra):
    if letra == '1':

```

```

        return 3
    elif letra == '0':
        return 0
    else:
        return -1

def estado_tres(letra):
    if letra == '1':
        return 2
    elif letra == '0':
        return 1
    else:
        return -1

```

Archivo: grafico\_automata\_protocolo.py

```

from tkinter import *

def dibujar_grafico_automata():
    raiz = Tk()
    raiz.title('Automata')
    raiz.geometry('450x250')
    canvas = Canvas(raiz, width=600, height=410, bg='white')
    canvas.place(x=0,y=0)
    canvas.pack(expand=YES, fill=BOTH)

    x = 90
    y = 100
    r = 50
    numero_circulos = 2
    opciones = ['Ready', 'Sending']
    canvas.create_line(x-20, y+r*1.2, x+.2*r, y+.8*r, width=2, fill='black')
    canvas.create_oval(x+.2*r - 8, y+.8*r +7, x+.2*r +2 , y+.8*r-3, width=1, fill='black')

    for i in range(numero_circulos):
        if i == 1:
            canvas.create_oval(x+r, y+10, x+2*r, y-30, width=1, fill='white')
            canvas.create_oval(x+r-3, y-10, x+r+7, y, width=1, fill='black')
            widget = Label(canvas, text='time_out', fg='black', bg='white')
            widget.pack()
            canvas.create_window(x+2*r, y-r+5, window=widget)
            xy = x-2*r-30, y+80, x+r, y+r-30
            canvas.create_arc(xy, start=0, extent=-180, style='arc')
            widget = Label(canvas, text='ack', fg='black', bg='white')
            widget.pack()
            canvas.create_window(x-r+10, y+95, window=widget)
            canvas.create_oval(x-2*r-33, y+50, x-2*r-23, y+60, width=1, fill='black')

            canvas.create_oval(x, y, x+2*r, y+r, width=1, fill='white')
            widget = Label(canvas, text=opciones[i], fg='black', bg='white')
            widget.pack()
            canvas.create_window(x+r, y+.5*r, window=widget)

```

```
if i < numero_circulos-1:  
    canvas.create_line(x+2*r, y+(r/2), x+2*r+70, y+(r/2), width=2, fill='black')  
    )  
    widget = Label(canvas, text='data_in', fg='black', bg='white')  
    widget.pack()  
    canvas.create_window(x+2*r+35, y+(r/2)-15, window=widget)  
    canvas.create_oval(x+2*r+60, y+(r/2)-5, x+2*r+70, y+(r/2)+5, width=1, fill='black')  
  
if i == 0:  
    canvas.create_oval(x+5, y+5, x+2*r-5, y+r-5, width=1, fill='white')  
  
x += 2*r+70  
  
raiz.mainloop()
```

### **5.3. Pruebas**

A continuación, se presentan las imágenes de los resultados que el programa arroja.

Figura 22: El programa corriendo el protocolo.

Figura 23: Las palabras generadas por el programa.

```

001011110111100100111101000001101
11110000111100111100000011001100
0011000000000011111110000111111
00110000110000001111000011001111
11000000000000001100000000000000
10110010101100100000001111001001
00001111000011110011001111001111
110011000000000011110011001100
10000001011100001111011000011010
000000110011111110110011001111
0110110000100100101011110100010
001110011111110100011101111110
110011111111100000001111111100
00110000001111000011110000001111
1100110011111100001111111111100
0001011101001000110010001010101
1011010101110001111100101010000
110000001111001100001111000001100
0000000010010001101100010100110
11111100111000001100110011001111
111111111000000011110000110000
111100010000101111000001000010
00001111000000111100111111001100
001000001110111100100010010011
0011001111001111111111110011100
000000111111000011110001111111
11111111001111110000111100000000
10101101000001011010100111101100
00111100001100111100110011111100
00110011001111001100110011111100

```

Figura 24: Las palabras aceptadas según el criterio de paridad.

```

Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 1)-->Delta(q1, 0)-->Delta(q3, 1)-->Delta(q2, 1)-->Delta(q3, 1)-->Delta(q2, 1)-->Delta(q3, 1)-->Delta(q1, 1)-->Delta(q0, 1)-->Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 1)-->Delta(q1, 1)-->Delta(q0, 1)-->Delta(q0, 0)-->Delta(q2, 1)-->Delta(q3, 0)-->Delta(q1, 0)-->Delta(q3, 0)-->Delta(q1, 0)-->Delta(q3, 1)-->Delta(q2, 1)-->Delta(q3, 0)-->Delta(q1, 1)-->q0

Delta(q0, 1)-->Delta(q1, 1)-->Delta(q0, 1)-->Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 0)-->Delta(q0, 1)-->Delta(q1, 1)-->Delta(q0, 1)-->Delta(q1, 1)-->Delta(q0, 1)-->Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 1)-->Delta(q1, 1)-->Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 1)-->Delta(q1, 1)-->Delta(q0, 0)-->Delta(q2, 0)-->q0

Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 1)-->Delta(q1, 1)-->Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 0)-->Delta(q0, 1)-->Delta(q2, 0)-->Delta(q0, 1)-->Delta(q1, 1)-->Delta(q0, 1)-->Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 1)-->Delta(q1, 1)-->Delta(q0, 1)-->Delta(q1, 1)-->Delta(q0, 1)-->Delta(q0, 1)-->Delta(q1, 1)-->q0

Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 1)-->Delta(q1, 1)-->Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 0)-->Delta(q0, 1)-->Delta(q2, 0)-->Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 1)-->Delta(q1, 1)-->Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 1)-->Delta(q1, 1)-->Delta(q0, 1)-->Delta(q0, 0)-->Delta(q2, 0)-->Delta(q0, 1)-->Delta(q1, 1)-->q0

```

Figura 25: El historial de estados del autómata de paridad usado para validar las palabras.

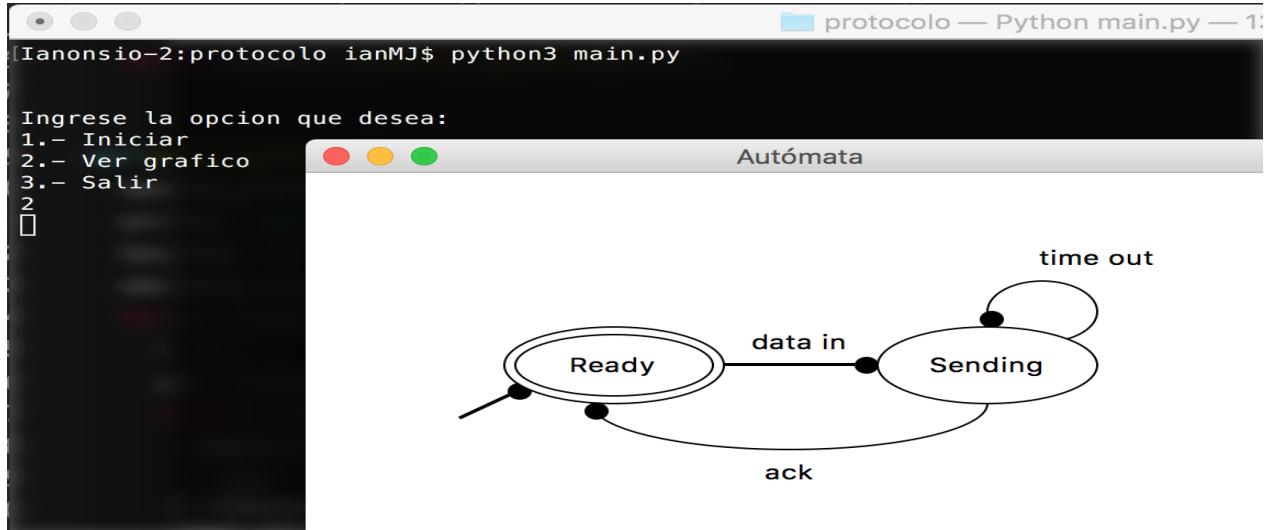


Figura 26: El programa imprimiendo el autómata.

## 6. Cadenas que terminan en 01

Los autómatas no determinísticos (NFA) son otro tipo de autómatas cuya característica es que cada estado puede ser seguido de un conjunto de estados, a diferencia de los autómatas determinísticos que son lineales. Otra característica importante es que los NFA es que tienen menos caminos que los DFA, el siguiente problema, ilustra la utilidad de este tipo de autómatas.

### 6.1. Descripción del problema

Se necesita crear un programa que sea capaz de evaluar si una cadena binaria termina en 01. Además, tiene que imprimir una tabla con los estados por los que ha pasado la cadena a evaluar.

### 6.2. Código

El autómata utilizado para este problema fue:

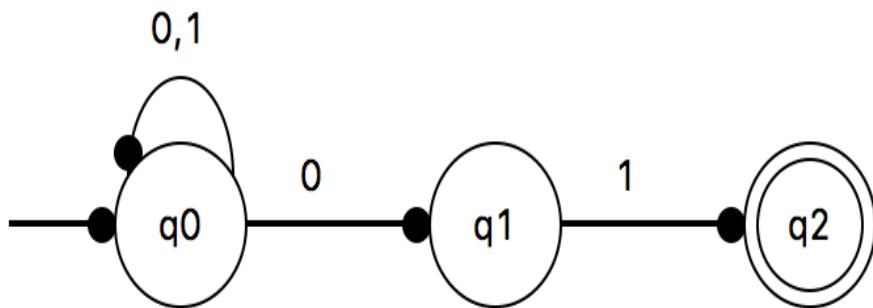


Figura 27: El autómata utilizado para este problema.

El lenguaje utilizado para este programa fue Python en su versión 3.5. El código para resolver el problema es el siguiente:

Archivo: main\_cero\_uno.py

```

from metodos import *

def main():
    try:
        historial = open('historial_cero_uno.txt', 'w')
        historial.close()

        while True:
            opcion = imprimir_menu()
            texto = ''
            tabla_estados = []
            if opcion == 1:
                while True:
                    texto = input('\n\nIngrese una palabra binaria para evaluar:')
                    tabla_estados = evaluar_cadena(texto)
                    imprimir_estados(tabla_estados)

                    opcion = input("\n\nQuiere ingresar otro numero? [s/n]")
                    if opcion != 's' and opcion != 'S':
                        break

            elif opcion == 2:
                aleatorio = 0
                while True:
                    texto = generar_palabra()
                    print('\n\nSe evaluara la palabra:', texto)
                    tabla_estados = evaluar_cadena(texto)
                    imprimir_estados(tabla_estados)
                    aleatorio = int(random() * 10) % 2
                    if aleatorio == 0:
                        break

            elif opcion == 3:
                graficar_automata()
            else:
                return 1

    except Exception as e:
        print('Uups, parece que hubo un problema,', e)

def imprimir_menu():
    seguir = True
    while seguir:
        try:
            opcion = int(input("\n\nIngrese la opcion que desea:\n1.- Ingresar texto\n2.- Automatico\n3.- Ver grafico del automata\n4.- Salir\n"))
            return opcion
        except Exception as e:
            print("Por favor, ingrese un dato valido.")


main()

```

Archivo: metodos\_cero\_uno.py

```

from random import random
from tkinter import *

def evaluar_cadena(texto):
    tabla_estados = [[0]]
    estado = []
    contador = 0
    for x in texto:
        for i in range(len(tabla_estados)):
            estado = automata(tabla_estados[i][contador], x)
            if len(estado) == 2:
                tabla_estados.append([0] * (contador+1))
                tabla_estados[(len(tabla_estados)-1)].append(1)
                tabla_estados[i].append(0)
            elif estado[0] == 2:
                tabla_estados[i].append(2)
            elif estado[0] == -1:
                tabla_estados[i].append(-1)
            else:
                tabla_estados[i].append(0)

        contador += 1

    return tabla_estados


def automata(estado, letra):
    if estado == 0:
        return estado_cero(letra)
    elif estado == 1:
        return estado_uno(letra)
    elif estado == 2:
        return estado_dos(letra)
    else:
        return [-1]

def estado_cero(letra):
    if letra == '0':
        return [0,1]
    elif letra == '1':
        return [1]
    else:
        return [-1]

def estado_uno(letra):
    if letra == '1':
        return [2]
    else:
        return [-1]

```

```

def estado_dos(letra):
    if letra != '':
        return [-1]
    else:
        return [2]

def imprimir_estados(tabla_estados):
    archivo = open('historial_cero_uno.txt', 'a')
    auxiliar = ''
    for x in range(0, len(tabla_estados[0])):
        for j in range(len(tabla_estados)):
            if tabla_estados[j][x] == -1:
                archivo.write('xx_')
                print('xx', end='_')
            else:
                auxiliar = 'q'+str(tabla_estados[j][x])+'_'
                archivo.write(auxiliar)
                print(auxiliar, end=' ')
        archivo.write('\n')
        print("")

    if tabla_estados[len(tabla_estados)-1][len(tabla_estados[0])-1] == 2:
        archivo.write('La_palabra_es_valida\n\n')
        print("La_palabra_es_valida\n\n")
    else:
        archivo.write('La_palabra_no_es_valida\n\n')
        print("La_palabra_no_es_valida\n\n")

    archivo.close()

def graficar_automata():
    raiz = Tk()
    raiz.title('Automata')
    raiz.geometry('500x250')
    canvas = Canvas(raiz, width=600, height=410, bg='white')
    canvas.place(x=0,y=0)
    canvas.pack(expand=YES, fill=BOTH)

    x = 120
    y = 100
    r = 50
    numero_circulos = 3
    opciones = ['0', '1']
    canvas.create_line(x-40, y+(r/2), x, y+(r/2), width=2, fill='black')
    canvas.create_oval(x-10, y+(r/2)-5, x, y+(r/2)+5, width=1, fill='black')

    xy = x+5, y-20, x+45, y+40
    canvas.create_arc(xy, start=0, extent=180, style='arc')
    canvas.create_oval(x, y+(r/2)-27, x+10, y+(r/2)-17, width=1, fill='black')
    widget = Label(canvas, text='0,1', fg='black', bg='white')
    widget.pack()

```

```

canvas.create_window(x+25, y-35, window=widget)

for i in range(numero_circulos):
    canvas.create_oval(x, y, x+r, y+r, width=1, fill='white')
    widget = Label(canvas, text='q'+str(i), fg='black', bg='white')
    widget.pack()
    canvas.create_window(x+.5*r, y+.5*r, window=widget)

    if i < numero_circulos-1:
        canvas.create_line(x+r, y+(r/2), x+r+70, y+(r/2), width=2, fill='black')
        widget = Label(canvas, text=opciones[i], fg='black', bg='white')
        widget.pack()
        canvas.create_window(x+r+25, y+(r/2)-15, window=widget)
        canvas.create_oval(x+r+60, y+(r/2)-5, x+r+70, y+(r/2)+5, width=1, fill='
            black')

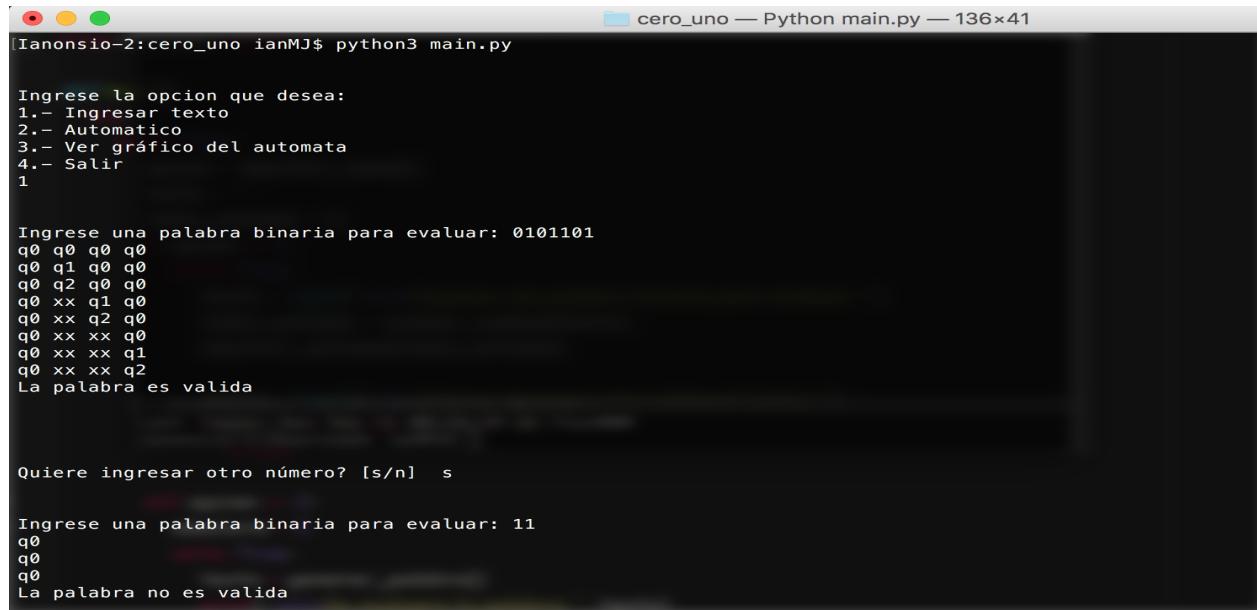
    if i == numero_circulos -1:
        canvas.create_oval(x+5, y+5, x+r-5, y+r-5, width=1, fill='white')
    x += r+70

raiz.mainloop()

```

### 6.3. Pruebas

En este apartado se muestran las imágenes de los resultados del autómata.



```
cero_uno — Python main.py — 136x41
| Ianonsio-2:cero_uno ianMJ$ python3 main.py

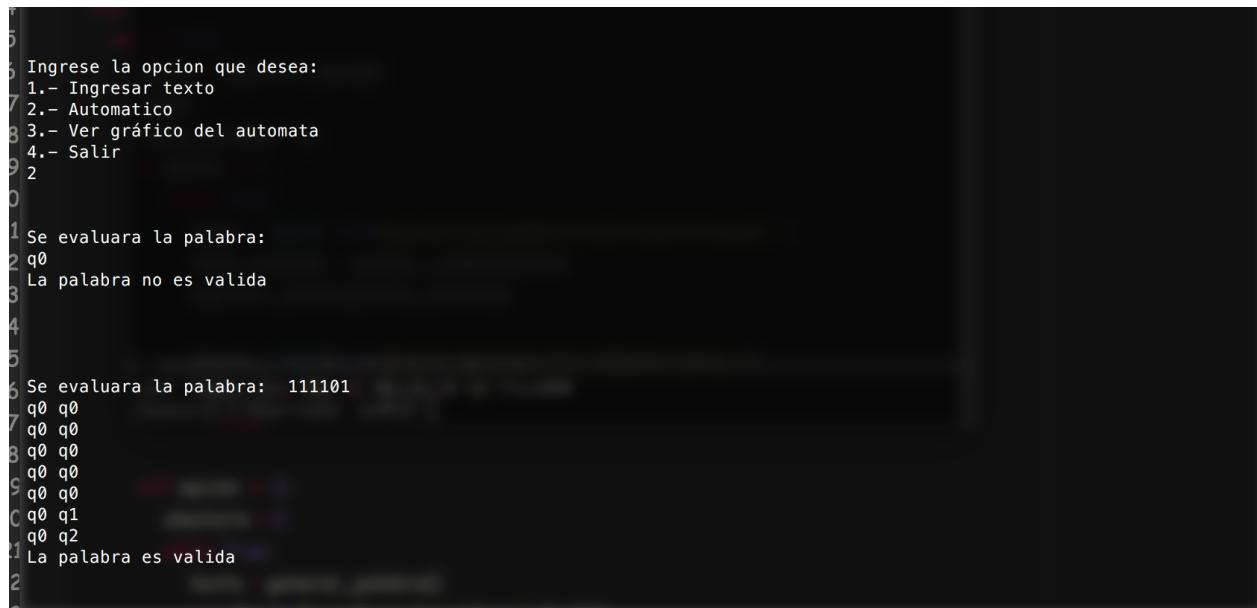
Ingrese la opcion que desea:
1.- Ingresar texto
2.- Automatico
3.- Ver grafico del automata
4.- Salir
1

Ingrese una palabra binaria para evaluar: 0101101
q0 q0 q0 q0
q0 q1 q0 q0
q0 q2 q0 q0
q0 xx q1 q0
q0 xx q2 q0
q0 xx xx q0
q0 xx xx q1
q0 xx xx q2
La palabra es valida

Quiere ingresar otro numero? [s/n] s

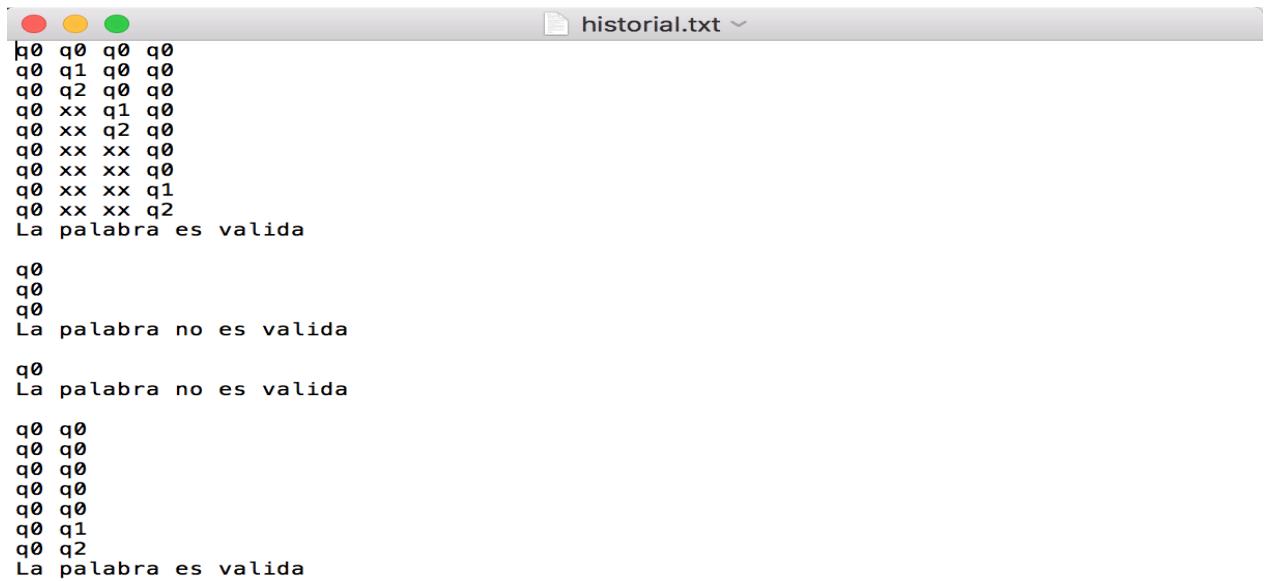
Ingrese una palabra binaria para evaluar: 11
q0
q0
q0
La palabra no es valida
```

Figura 28: Números ingresados manualmente.



```
5
6 Ingrese la opcion que desea:
7 1.- Ingresar texto
8 2.- Automatico
9 3.- Ver grafico del automata
10 4.- Salir
11 2
12
13 Se evaluara la palabra:
14 q0
15 La palabra no es valida
16
17
18 Se evaluara la palabra: 111101
19 q0 q0
20 q0 q0
21 q0 q0
22 q0 q0
23 q0 q1
24 q0 q2
25 La palabra es valida
26
```

Figura 29: Modo automático.



```

 historial.txt ~
q0 q0 q0 q0
q0 q1 q0 q0
q0 q2 q0 q0
q0 xx q1 q0
q0 xx q2 q0
q0 xx xx q0
q0 xx xx q0
q0 xx xx q1
q0 xx xx q2
La palabra es valida

q0
q0
q0
La palabra no es valida

q0
La palabra no es valida

q0 q0
q0 q0
q0 q0
q0 q0
q0 q0
q0 q1
q0 q2
La palabra es valida

```

Figura 30: El historial de correr el programa.

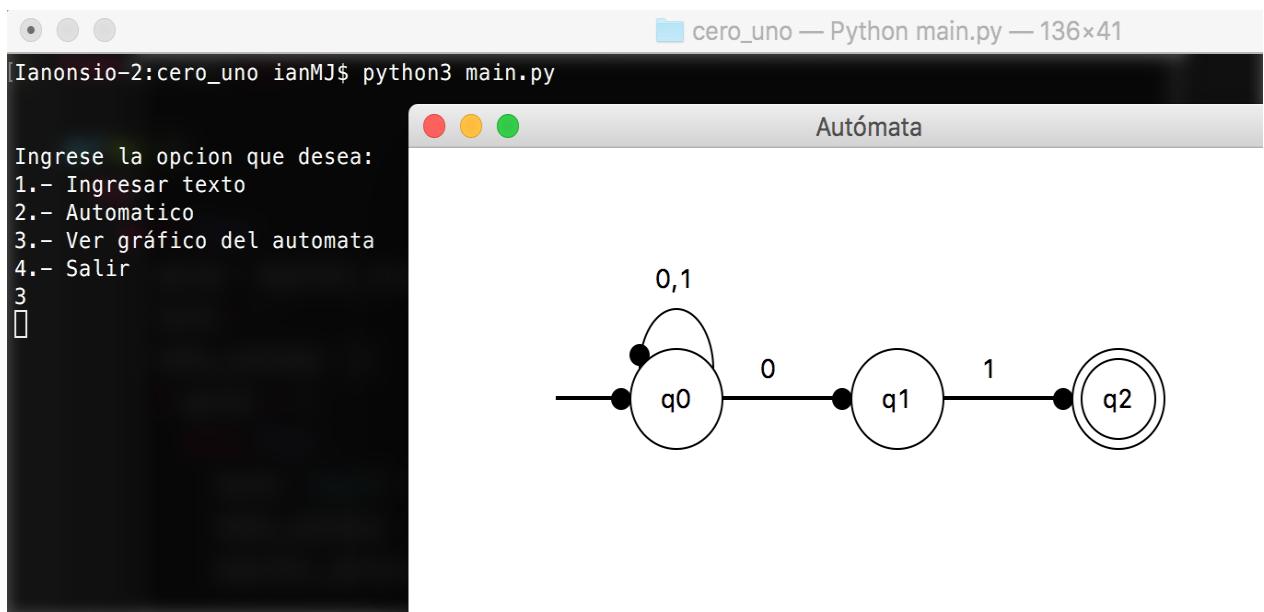


Figura 31: El programa imprimiendo el autómata.

## **7. Referencias**

- [1] HOPCROFT JOHN, MOTWANI RAJEEV, U. J. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, 2008.
- [2] ULLMAN, J. D. Finite automata. url`http://infolab.stanford.edu/~ullman/ialc/spr10/slides/fa1.pdf` 2010. Accedido: 2016-09-10.