

Reporte de prácticas primer periodo

Ian Mendoza Jaimes

4 de septiembre de 2016

Índice

1. Introducción	3
2. Alfabetos	4
2.1. Descripción del problema	4
2.2. Código fuente	4
2.3. Pruebas	8
3. Números primos	10
3.1. Descripción del problema	10
3.2. Código	10
3.3. Pruebas	13
4. Palabras que terminan en ere	15
4.1. Descripción del problema	15
4.2. Código	15
4.3. Pruebas	21
5. Paridad	22
6. Protocolo	23
7. Cadenas que terminan en 01	24

1. Introducción

2. Alfabetos

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

2.1. 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,

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

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

2.2. 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("%s\n", "Seleccione_la_modalidad:_\n1.-_Automatico_\n2.-_Manual_\n3.-_Salir");
        scanf("%c", &seleccion);

        if(seleccion == '1' || seleccion == '2'){
            continuar_modalidad = 1;
```

```

        while (continuar_modalidad == 1) {
            if (seleccion == '1') {
                n = 1 + (rand() % 5);
            }
            else {
                printf("%s", "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 tamaño_alfabeto = 2;
    char * alfabeto = NULL;
    alfabeto = (char *) malloc(tamaño_alfabeto * sizeof(char));
    iniciar_alfabeto(&alfabeto, tamaño_alfabeto);

    obtener_cadenas(alfabeto, tamaño_alfabeto, n);

    return 1;
}

```

```

}

int obtener_cadenas(char *alfabeto, int tamaño, 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) > (tamaño - 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 tamaño){
    int i;
    fputs(",_", archivo);
    for(i = 0; i < tamaño; i++){
        fputc(*(alfabeto + *(cadena_temporal + i)) , archivo);
    }
    return 1;
}

```

```

}

int iniciar_alfabeto(char **alfabeto , int tamano){
    int i;
    for(i = 0; i < tamano; 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

```


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

3. Números primos

Un número primo es aquel que solo puede ser dividido entre si mismo o la unidad. A lo largo de la historia mucha gente los a 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.

3.1. 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.

3.2. 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.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.-Manual\n3.-Salir")
        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)
        numero_ceros_unos = contar_ceros_unos(numeros_primos_binarios)

        imprimir_ceros_unos(numero_ceros_unos, numeros_primos)

        archivo.close()

        if seleccion == 1:
            continuar = int(random() * 100) % 2
        else:
            eleccion = ingresar_datos("\nDesea_ingresar_otra_n?_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")

def imprimir_ceros_unos(numero_ceros_unos, numeros_primos):

```

```

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

main()

```

Archivo: metodos.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):
        es_primo = True

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

        if es_primo:
            numeros_primos.append(x)

    return numeros_primos

def convertir_primos_a_binarios(numeros_primos, archivo):
    numeros_primos_binarios = []
    for x in numeros_primos:
        numeros_primos_binarios.append(bin(x)[2:])
        archivo.write(" ,_" + bin(x)[2:])

    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

```

3.3. Pruebas

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

Para la selección de modo automático:

```

numeros_primos — Python main.py — 136x41
Selecciona el modo en que se ejecutará el programa:
1.- Automático
2.- Manual
3.- Salir
1

Fue seleccionado un n = 239
[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]
{ numero primo, numero de unos }
{ 2, 1 }, { 3, 2 }, { 5, 2 }, { 7, 3 }, { 11, 3 }, { 13, 3 }, { 17, 2 }, { 19, 3 }, { 23, 4 }, { 29, 4 }, { 31, 5 }, { 37, 3 }, { 41, 3 }, { 43, 4 }, { 47, 5 }, { 53, 4 }, { 59, 5 }, { 61, 5 }, { 67, 3 }, { 71, 4 }, { 73, 3 }, { 79, 5 }, { 83, 4 }, { 89, 4 }, { 97, 3 }, { 101, 4 }, { 103, 5 }, { 107, 5 }, { 109, 5 }, { 113, 4 }, { 127, 7 }, { 131, 3 }, { 137, 3 }, { 139, 4 }, { 149, 4 }, { 151, 5 }, { 157, 5 }, { 163, 4 }, { 167, 5 }, { 173, 5 }, { 179, 5 }, { 181, 5 }, { 191, 7 }, { 193, 3 }, { 197, 4 }, { 199, 5 }, { 211, 5 }, { 223, 7 }, { 227, 5 }, { 229, 5 }, { 233, 5 }, { 239, 7 },

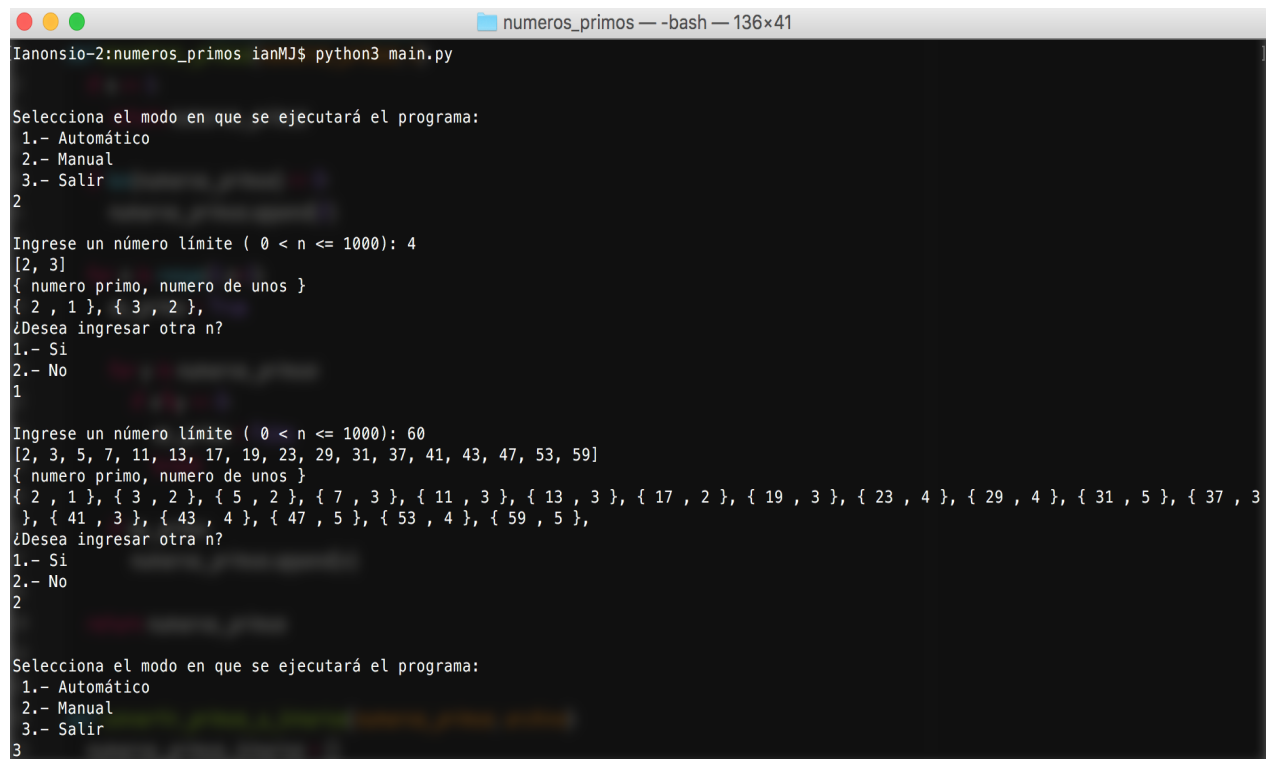
Fue seleccionado un n = 556
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 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]
{ numero primo, numero de unos }
{ 2, 1 }, { 3, 2 }, { 5, 2 }, { 7, 3 }, { 11, 3 }, { 13, 3 }, { 17, 2 }, { 19, 3 }, { 23, 4 }, { 29, 4 }, { 31, 5 }, { 37, 3 }, { 41, 3 }, { 43, 4 }, { 47, 5 }, { 53, 4 }, { 59, 5 }, { 61, 5 }, { 67, 3 }, { 71, 4 }, { 73, 3 }, { 79, 5 }, { 83, 4 }, { 89, 4 }, { 97, 3 }, { 101, 4 }, { 103, 5 }, { 107, 5 }, { 109, 5 }, { 113, 4 }, { 127, 7 }, { 131, 3 }, { 137, 3 }, { 139, 4 }, { 149, 4 }, { 151, 5 }, { 157, 5 }, { 163, 4 }, { 167, 5 }, { 173, 5 }, { 179, 5 }, { 181, 5 }, { 191, 7 }, { 193, 3 }, { 197, 4 }, { 199, 5 }, { 211, 5 }, { 223, 7 }, { 227, 5 }, { 229, 5 }, { 233, 5 }, { 239, 7 }, { 241, 5 }, { 251, 7 }, { 257, 2 }, { 263, 4 }, { 269, 4 }, { 271, 5 }, { 277, 4 }, { 281, 4 }, { 283, 5 }, { 293, 4 }, { 307, 5 }, { 311, 6 }, { 313, 5 }, { 317, 6 }, { 331, 5 }, { 337, 4 }, { 347, 6 }, { 349, 6 }, { 353, 4 }, { 359, 6 }, { 367, 7 }, { 373, 6 }, { 379, 7 }, { 383, 8 }, { 389, 4 }, { 397, 5 }, { 401, 4 }, { 409, 5 }, { 419, 5 }, { 421, 5 }, { 431, 7 }, { 433, 5 }, { 439, 7 }, { 443, 7 }, { 449, 4 }, { 457, 5 }, { 461, 6 }, { 463, 7 }, { 467, 6 }, { 479, 8 }, { 487, 7 }, { 491, 7 }, { 499, 7 }, { 503, 8 }, { 509, 8 }, { 521, 3 }, { 523, 4 }, { 541, 5 }, { 547, 4 },

Fue seleccionado un n = 630
[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]
{ numero primo, numero de unos }
{ 2, 1 }, { 3, 2 }, { 5, 2 }, { 7, 3 }, { 11, 3 }, { 13, 3 }, { 17, 2 }, { 19, 3 }, { 23, 4 }, { 29, 4 }, { 31, 5 }, { 37, 3 }, { 41, 3 }, { 43, 4 }, { 47, 5 }, { 53, 4 }, { 59, 5 }, { 61, 5 }, { 67, 3 }, { 71, 4 }, { 73, 3 }, { 79, 5 }, { 83, 4 }, { 89, 4 }, { 97, 3 }, { 101, 4 }, { 103, 5 }, { 107, 5 }, { 109, 5 }, { 113, 4 }, { 127, 7 }, { 131, 3 }, { 137, 3 }, { 139, 4 }, { 149, 4 }, { 151, 5 }, { 157, 5 }, { 163, 4 }, { 167, 5 }, { 173, 5 }, { 179, 5 }, { 181, 5 }, { 191, 7 }, { 193, 3 }, { 197, 4 }, { 199, 5 }, { 211, 5 }, { 223, 7 }, { 227, 5 }, { 229, 5 }, { 233, 5 }, { 239, 7 }, { 241, 5 }, { 251, 7 }, { 257, 2 }, { 263, 4 }, { 269, 4 }, { 271, 5 }, { 277, 4 }, { 281, 4 }, { 283, 5 }, { 293, 4 }, { 307, 5 }, { 311, 6 }, { 313, 5 }, { 317, 6 }, { 331, 5 }, { 337, 4 }, { 347, 6 }, { 349, 6 }, { 353, 4 }, { 359, 6 }, { 367, 7 }, { 373, 6 }, { 379, 7 }, { 383, 8 }, { 389, 4 }, { 397, 5 }, { 401, 4 }, { 409, 5 }, { 419, 5 }, { 421, 5 }, { 431, 7 }, { 433, 5 }, { 439, 7 }, { 443, 7 }, { 449, 4 }, { 457, 5 }, { 461, 6 }, { 463, 7 }, { 467, 6 }, { 479, 8 }, { 487, 7 }, { 491, 7 }, { 499, 7 }, { 503, 8 }, { 509, 8 }, { 521, 3 }, { 523, 4 }, { 541, 5 }, { 547, 4 },

```

Figura 4: El resultado de seleccionar la modalidad automática

Para la selección manual:



```
numeros_primos — -bash — 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

Ingresa un número límite ( 0 < n <= 1000): 4
[2, 3]
{ numero primo, numero de unos }
{ 2 , 1 }, { 3 , 2 },
¿Desea ingresar otra n?
1.- Si
2.- No
1

Ingresa un número límite ( 0 < n <= 1000): 60
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59]
{ numero primo, numero de unos }
{ 2 , 1 }, { 3 , 2 }, { 5 , 2 }, { 7 , 3 }, { 11 , 3 }, { 13 , 3 }, { 17 , 2 }, { 19 , 3 }, { 23 , 4 }, { 29 , 4 }, { 31 , 5 }, { 37 , 3 }, { 41 , 3 }, { 43 , 4 }, { 47 , 5 }, { 53 , 4 }, { 59 , 5 },
¿Desea ingresar otra n?
1.- Si
2.- No
2

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

Figura 5: El resultado de seleccionar la modalidad manual

4. 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.

4.1. 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 línea se encuentran. Para la realización de este programa, se debe de utilizar un autómata finito determinístico.

4.2. Código

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

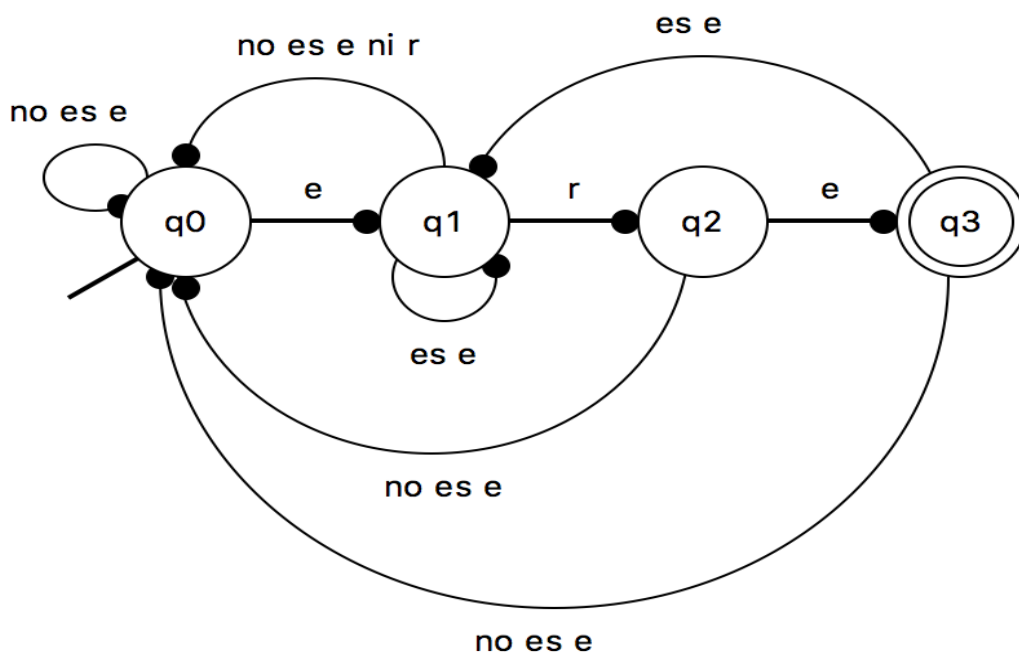


Figura 6: El autómata utilizado para este problema

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

Archivo: main.py

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

def main():
    while seguir:
        try:
            palabras_aceptadas = []
            texto = ""
            archivo = None

            opcion = imprimir_menu()

            if opcion == 1:
                texto = input("\n\nIngrese un texto: \n")
                palabras_aceptadas = leer_texto(texto)
            elif opcion == 2:
                texto = input("\n\nIngrese el nombre de un archivo: \n")
                archivo = open(texto, "r")
                palabras_aceptadas = leer_archivo(archivo)
            elif opcion == 3:
                print("Sera utilizado el archivo heart.txt")
                archivo = open("heart.txt", "r")
                palabras_aceptadas = leer_archivo(archivo)
            elif opcion == 4:
                graficar_automata()
            else:
                return 0

            imprimir_palabras_aceptadas(palabras_aceptadas, opcion)

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

        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.-\n_\nIngresar texto\n2.-\n_Ingresar un archivo\n3.-\n_Automatico\n4.-\n_Ver grafico del automata\n5.-\n_Salir\n"))
```



```

        return opcion
    except Exception as e:
        print("Por_favor ,_ingrese_un_dato_valido.")

def leer_texto(texto):
    texto += "_"
    aceptadas = obtener_palabras(texto)
    return aceptadas

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

    archivo.close()
    return palabras_aceptadas

def imprimir_palabras_aceptadas(palabras_aceptadas, seleccion):
    print("\n\n\n")
    if seleccion == 1:
        print("Palabras_aceptadas:_", palabras_aceptadas)
    else:
        contador = 0
        for x in palabras_aceptadas:
            if len(x[1]) > 0:
                print("Linea_", x[0], "_palabras_aceptadas:_", x[1])

main()

```

Archivo:

```

from tkinter import *
import time

def obtener_palabras(texto):
    estado = 0
    palabras_aceptadas = []
    temporal = ''
    letra_auxiliar = ''
    for x in texto:
        letra_auxiliar = x
        letra_auxiliar.lower()

```

```

    estado = automata(estado, letra_auxiliar)

    if estado == -1:
        estado = 0
        temporal = ''
    elif estado == 4:
        palabras_aceptadas.append(temporal)
        estado = 0
        temporal = ''
    else:
        temporal += x

    return palabras_aceptadas

def automata(estado, letra_auxiliar):
    alfabeto = [97, 122]
    if estado == 0:
        return estado_cero(alfabeto, letra_auxiliar)
    elif estado == 1:
        return estado_uno(alfabeto, letra_auxiliar)
    elif estado == 2:
        return estado_dos(alfabeto, letra_auxiliar)
    elif estado == 3:
        return estado_tres(alfabeto, letra_auxiliar)
    else:
        return -1

def estado_cero(alfabeto, letra):
    if (ord(letra) >= alfabeto[0] and ord(letra) <= alfabeto[1]):
        if letra == 'e':
            return 1
        else:
            return 0
    else:
        return -1

def estado_uno(alfabeto, letra):
    if (ord(letra) >= alfabeto[0] and ord(letra) <= alfabeto[1]):
        if letra == 'e':
            return 1
        elif letra == 'r':
            return 2
        else:
            return 0
    else:
        return -1

```

```

def estado_dos(alfabeto, letra):
    if(ord(letra) >= alfabeto[0] and ord(letra) <= alfabeto[1]):
        if letra == 'e':
            return 3
        else:
            return 0
    else:
        return -1

def estado_tres(alfabeto, letra):
    if(ord(letra) >= alfabeto[0] and ord(letra) <= alfabeto[1]):
        if letra == 'e':
            return 1
        else:
            return 0
    else:
        return 4

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_e_ni_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)

```

4.3. Pruebas

5. Paridad

6. Protocollo

7. Cadenas que terminan en 01

Referencias

- [1] HOPCROFT JOHN, MOTWANI RAJEEV, U. J. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, 2008.