



**INSTITUTO POLITÉCNICO
NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO**



**Reporte de práctica:
Máquina de Turing**

Grupo:
4CM2

Nombre del alumno:
Maldonado Hernández Alexander

Unidad de aprendizaje:
Teoría de la computación

Maestro:
Genaro Juárez Martínez

Fecha de entrega:
22 de junio de 2025

Índice

1. Introducción	1
2. Desarrollo	2
2.1. Definición y representación de la cinta	2
2.2. Animación de la máquina de Turing	3
2.3. Registro de la ejecución	7
3. Conclusiones	8
4. Referencias	9
5. Anexos	10
5.1. Anexo 1. Código fuente del programa	10

1. Introducción

La presente práctica tiene como objetivo la implementación de una Máquina de Turing que reconozca el lenguaje formal definido por $L = \{0^n 1^n \mid n \geq 1\}$, el cual consta de todas las cadenas de ceros seguidos por la misma cantidad de unos. Este lenguaje no es regular ni libre de contexto, lo cual lo hace ideal para ser reconocido mediante una Máquina de Turing, dado su mayor poder computacional. La especificación de la máquina utilizada proviene del libro *Introduction to Automata Theory, Languages, and Computation* de John E. Hopcroft. En el cual se nos especifica la tabla de transiciones del Cuadro 1

Estado	Símbolo				
	0	1	X	Y	B
q_0	(q_1, X, R)	-	-	(q_3, Y, R)	-
q_1	$(q_1, 0, R)$	(q_2, Y, L)	-	(q_1, Y, R)	-
q_2	$(q_2, 0, L)$	-	(q_0, X, R)	(q_2, Y, L)	-
q_3	-	-	-	(q_3, Y, R)	(q_4, B, R)
q_4	-	-	-	-	-

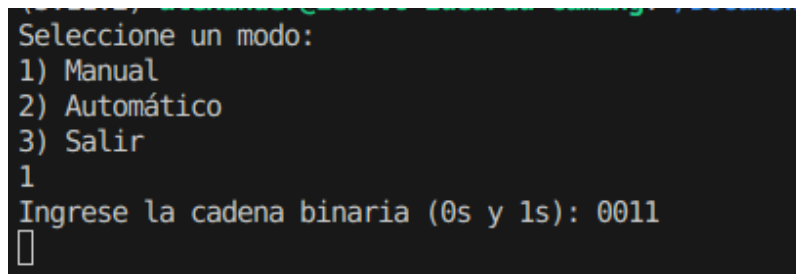
Cuadro 1: Tabla de transición de la máquina de Turing

El programa desarrollado permite ingresar una cadena binaria de dos formas: manualmente por parte del usuario, o bien generándola de manera automática. En ambos casos, se impone una restricción de longitud máxima de 1000 caracteres, para asegurar un tiempo de ejecución razonable y evitar desbordamientos de memoria. Además, se añadió una funcionalidad especial que anima el proceso de ejecución de la máquina mediante una interfaz gráfica utilizando la librería `tkinter`, siempre que la longitud de la cadena sea menor o igual a 10 símbolos. Para cadenas más largas, el programa realiza el procesamiento de forma silenciosa, registrando paso a paso las transiciones en un archivo de texto.

Durante la ejecución, el programa simula el comportamiento de la máquina de Turing de forma detallada, mostrando las descripciones instantáneas de cada transición, incluyendo el estado actual, el símbolo leído, el nuevo estado, el símbolo escrito y el movimiento de la cabeza lectora. También se implementó el manejo dinámico de la cinta, la cual se expande automáticamente hacia la derecha o izquierda en caso de que el cabezal intente acceder a posiciones fuera del rango inicial.

2. Desarrollo

Antes de entrar en el detalle del desarrollo, es importante mencionar cómo se inicia la ejecución del programa y cómo se maneja la cadena de entrada. Al iniciar, el programa solicita al usuario elegir entre ingresar manualmente una cadena binaria o generar una automáticamente. Para ejemplificar el funcionamiento de la máquina de Turing, se utilizará la cadena "0011" como se observa en la Figura 1, ya que esta cumple con la condición del lenguaje $\{0^n 1^n \mid n \geq 1\}$. Esta cadena permitirá mostrar paso a paso cómo la máquina procesa y reconoce correctamente las cadenas válidas.



```
Seleccione un modo:  
1) Manual  
2) Automático  
3) Salir  
1  
Ingrese la cadena binaria (0s y 1s): 0011  
█
```

Figura 1: Insertando la cadena a la máquina de Turing

2.1. Definición y representación de la cinta

En el contexto de una máquina de Turing, la cinta es un componente fundamental, ya que representa la memoria infinita sobre la cual se realiza la lectura y escritura de símbolos. Para esta implementación, la cinta se modela como una lista unidimensional en Python, cuyos elementos corresponden a los símbolos de la cadena de entrada, más un símbolo especial de espacio en blanco ('B') al final, el cual permite detectar el término de la cadena durante el proceso de evaluación.

El cabezal lector-escritor se representa mediante un índice que recorre la lista, comenzando en la primera posición. Cada movimiento hacia la derecha o izquierda corresponde a un incremento o decremento del valor de este índice, respectivamente. En caso de que el puntero se desplace fuera del rango de la cinta actual (por ejemplo, hacia una posición negativa o más allá del final), se extiende la cinta automáticamente agregando un nuevo símbolo blanco ('B') en la dirección correspondiente. Esto simula el comportamiento de una cinta infinita en ambos sentidos, característica típica de las máquinas de Turing teóricas.

Durante la animación gráfica con `tkinter`, cada celda de la cinta se muestra como un rectángulo de tamaño fijo que contiene el símbolo correspondiente. El cabezal se representa visualmente mediante un cambio de color en la celda activa: se utiliza color rojo para indicar la posición actual del cabezal y verde para las demás. Este diseño visual facilita la comprensión paso a paso del funcionamiento de la máquina. La cinta se puede observar en ejecución en la Figura 2.

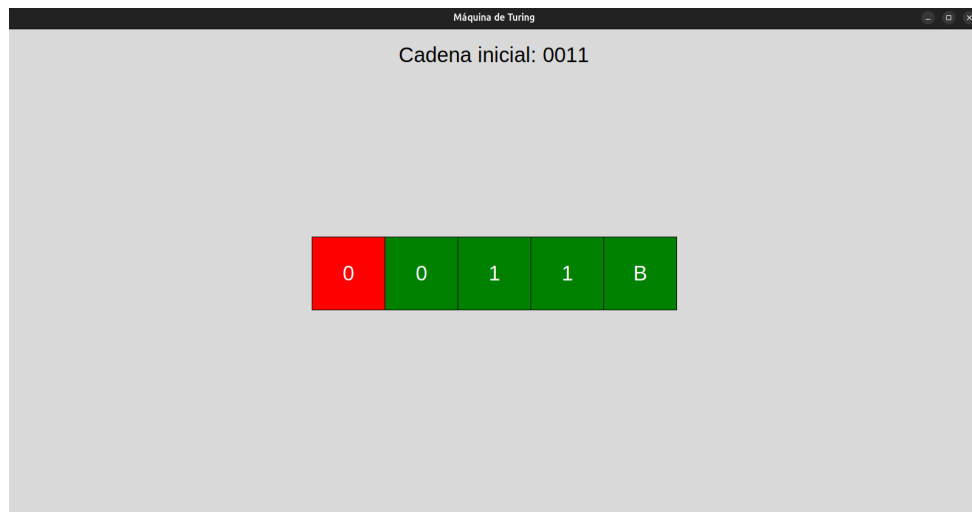


Figura 2: Inicio de la cinta

2.2. Animación de la máquina de Turing

Para comprender el comportamiento de la máquina de Turing implementada, se analiza paso a paso la ejecución con la cadena de entrada 0011, que pertenece al lenguaje $\{0^n 1^n \mid n \geq 1\}$.

Inicialmente, la cinta contiene los símbolos de la cadena seguida por un símbolo blanco ('B') al final, representándose como:

0 0 1 1 B

y el cabezal se posiciona en el primer símbolo, en el estado inicial q_0 como ya se había visto previamente en la Figura 2.

La máquina comienza leyendo el primer símbolo '0' en el estado q_0 . Según la tabla de transiciones 1, la transición correspondiente es:

$$\delta(q_0, 0) = (q_1, X, \rightarrow)$$

Esto significa que el símbolo '0' se reemplaza por 'X', la máquina cambia al estado q_1 , y el cabezal avanza una posición a la derecha. La cinta queda:

X 0 1 1 B

con el cabezal en la segunda posición como se observa en la Figura 3.

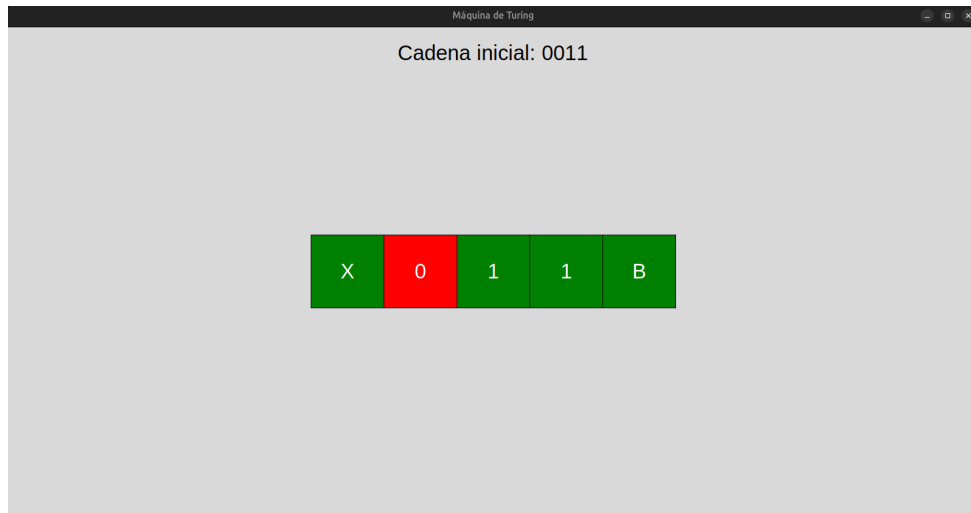


Figura 3: Cinta tras analizar el primer símbolo

En estado q_1 , la máquina sigue moviéndose hacia la derecha, buscando el primer símbolo '1' para marcarlo con una 'Y'. Durante este proceso, si encuentra '0' o 'Y', simplemente se desplaza a la derecha manteniendo el estado q_1 . Cuando el cabezal lee el primer '1' en la tercera posición, la transición aplicada es:

$$\delta(q_1, 1) = (q_2, Y, \leftarrow)$$

Se reemplaza el '1' por 'Y', la máquina cambia al estado q_2 , y el cabezal retrocede una posición. La cinta actualizada se puede ver en la Figura 4 que es:

X 0 Y 1 B

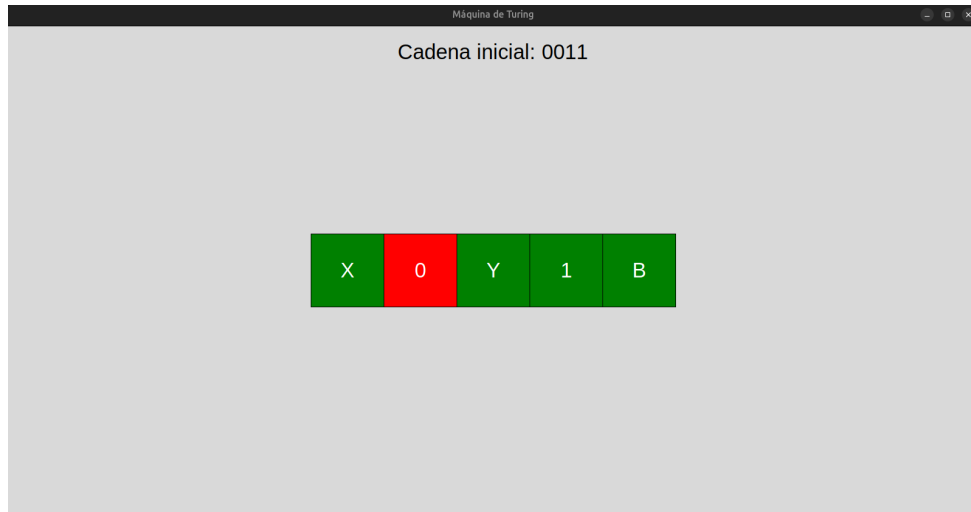


Figura 4: Cinta tras encontrar su primer '1'

El estado q_2 indica que la máquina se mueve hacia la izquierda para regresar al inicio de la cadena, buscando el símbolo marcado 'X' que indica el comienzo del bloque de ceros procesados. Durante este retroceso, si la máquina encuentra símbolos '0' o 'Y', permanece en q_2 y continúa moviéndose a la izquierda. Al leer el símbolo 'X', la transición es:

$$\delta(q_2, X) = (q_0, X, \rightarrow)$$

Esto hace que el cabezal regrese al estado inicial q_0 y avance a la derecha para comenzar otro ciclo de marcado.

Este proceso de marcar un '0' con 'X', buscar el correspondiente '1' para marcarlo con 'Y', y regresar al inicio se repite hasta que todos los ceros y unos han sido marcados como se ve en la Figura 5.

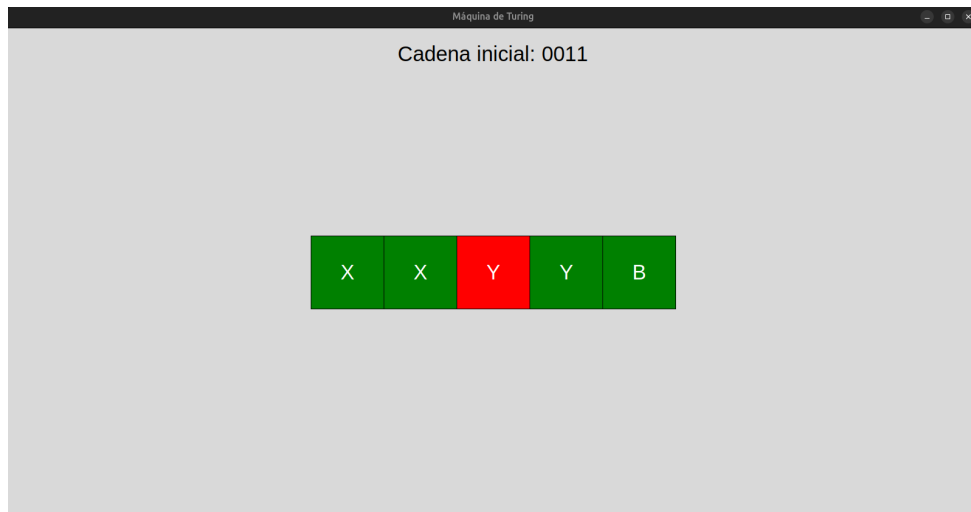


Figura 5: Cinta tras procesar todos los símbolos

Finalmente, cuando no quedan más ceros sin marcar y el cabezal lee únicamente símbolos 'Y', la máquina pasa al estado de aceptación q_4 mediante la transición que maneja el símbolo blanco al final de la cinta como se observa en la Figura 6.

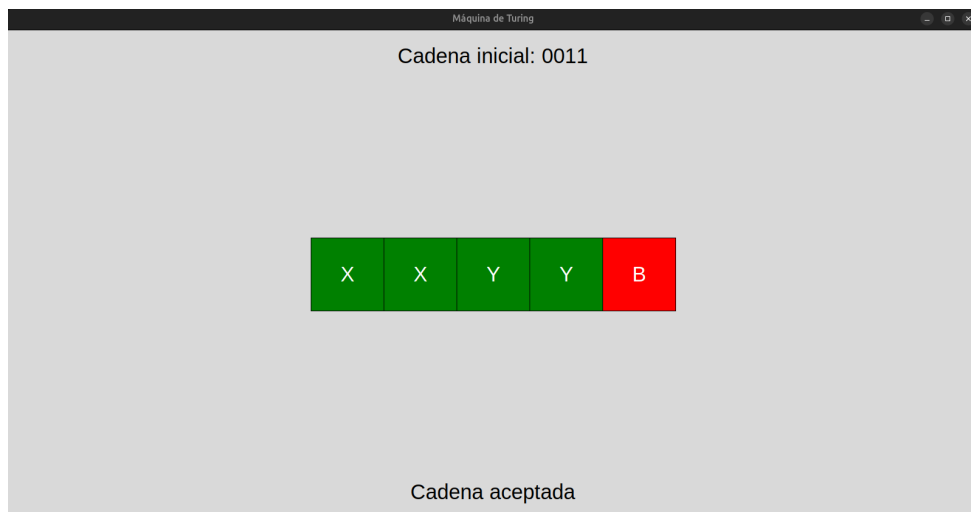


Figura 6: Se llega al estado de aceptación

A lo largo de esta ejecución, cada transición modifica el estado, el símbolo bajo el

cabezal y la posición del mismo, siguiendo estrictamente la tabla de transiciones presentada en la Tabla 1.

2.3. Registro de la ejecución

Con el fin de proporcionar una evidencia clara y detallada del comportamiento de la máquina de Turing durante la evaluación de las cadenas, se implementó un sistema de registro que almacena cada transición realizada en un archivo de texto. Este archivo, nombrado `registro_transiciones.txt`, se genera automáticamente cada vez que se ejecuta el programa y contiene una descripción precisa de las descripciones instantáneas que ocurren en cada paso de la computación.

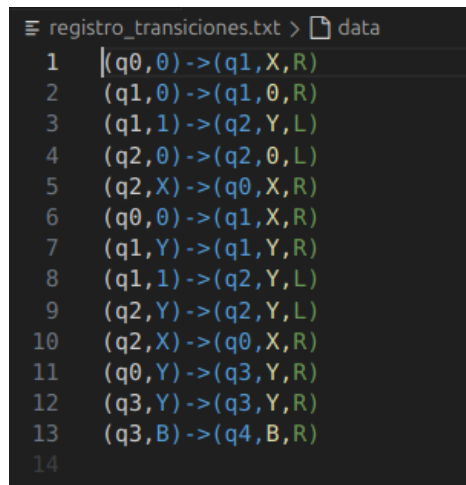
Cada línea del archivo representa una transición de la máquina en el siguiente formato:

$$(\text{estado_actual}, \text{símbolo_leído}) \rightarrow (\text{nuevo_estado}, \text{símbolo_escrito}, \text{dirección})$$

Por ejemplo, la línea:

$$(q_0, 0) \rightarrow (q_1, X, R)$$

indica que la máquina, estando en el estado q_0 y leyendo un símbolo '0', escribió una 'X' en la cinta, se desplazó a la derecha (denotado por 'R') y cambió al estado q_1 . En la Figura 7 se puede apreciar las transiciones realizadas para verificar la cadena 0011.



```
registro_transiciones.txt > data
1  |(q0,0)->(q1,X,R)
2  (q1,0)->(q1,0,R)
3  (q1,1)->(q2,Y,L)
4  (q2,0)->(q2,0,L)
5  (q2,X)->(q0,X,R)
6  (q0,0)->(q1,X,R)
7  (q1,Y)->(q1,Y,R)
8  (q1,1)->(q2,Y,L)
9  (q2,Y)->(q2,Y,L)
10 (q2,X)->(q0,X,R)
11 (q0,Y)->(q3,Y,R)
12 (q3,Y)->(q3,Y,R)
13 (q3,B)->(q4,B,R)
14
```

Figura 7: Transiciones realizadas por la máquina de Turing

Este archivo no solo permite verificar si una cadena ha sido aceptada o rechazada, sino que también sirve como herramienta de análisis para estudiar el comportamiento interno de la máquina. En caso de error o rechazo, es posible revisar el registro para identificar el estado exacto y el símbolo que causó la interrupción de la computación.

Cabe destacar que el registro se mantiene tanto para ejecuciones con animación (cadenas de longitud menor o igual a 10) como para aquellas sin visualización gráfica. En todos los casos, el archivo conserva un historial completo de las acciones realizadas, permitiendo así una trazabilidad completa del proceso.

3. Conclusiones

La realización de esta práctica permitió comprender a profundidad el funcionamiento de una máquina de Turing, no solo desde una perspectiva teórica, sino también desde su implementación computacional. El lenguaje $\{0^n1^n \mid n \geq 1\}$ sirvió como un caso de estudio ideal, al tratarse de un lenguaje no regular que exige una capacidad de memoria y control más allá de la que ofrecen los autómatas finitos o los autómatas con pila.

Uno de los principales logros fue simular correctamente la máquina de Turing definida en la tabla de transiciones referenciada, reproduciendo fielmente su comportamiento mediante estructuras de datos en Python. Se logró manejar de forma adecuada la cinta de longitud dinámica, el desplazamiento del cabezal, el registro de transiciones y la aceptación o rechazo de cadenas.

Además, la incorporación de una interfaz gráfica para la animación de cadenas cortas añadió un valor significativo a la práctica, al facilitar la observación del procesamiento paso a paso y reforzar visualmente los conceptos fundamentales del modelo. Por otro lado, el registro en archivo de texto permitió documentar cada transición realizada, lo cual es útil para la depuración y análisis de resultados.

Finalmente, esta práctica me ayudó a consolidar la diferencia entre los lenguajes reconocibles por distintos modelos de cómputo, y reforzó la importancia de las máquinas de Turing como modelo formal para el estudio de la computabilidad y los límites de lo que es posible calcular mediante algoritmos.

4. Referencias

- Agarwal, V. (2021, diciembre). Implementing a Turing machine easily in Python. <https://medium.com/@vardanagarwal16/implementing-a-turing-machine-easily-in-python-55213fc8d5d5>
- Mateu-Mollá, J. (2024, julio). Máquina de Turing: qué es y cómo funciona. <https://psicologiaymente.com/cultura/maquina-de-turing>
- Taleem, C. (2024, abril). Turing Machine Examples — Top 06 explained. <https://cstaleem.com/turing-machine-with-example>

5. Anexos

5.1. Anexo 1. Código fuente del programa

```
import time
import random as rd
from tkinter import *

MAXIMO_TAMANO = 1000
LIMITE_ANIMACION = 10

def obtener_cadena():
    while True:
        modo = input("Seleccione un modo: \n1) Manual  \n2)
↳ Automático\n3) Salir\n")
        if modo == "1":
            while True:
                cadena = input("Ingrese la cadena binaria (0s y 1s):
↳ ").strip()
                if len(cadena) > MAXIMO_TAMANO:
                    print("Elija una cadena menor a 1000
↳ caracteres.\n")
                    continue
                elif len(cadena) == 0:
                    print("Cadena vacía. Intente con otra cadena.\n")
                elif not set(cadena).issubset({'0','1'}):
                    print("La cadena debe contener solo ceros y
↳ unos.\n")
                else:
                    break
            break
        elif modo == "2":
            ceros = rd.randint(1, 1000)
            unos = rd.randint(0, MAXIMO_TAMANO - ceros)
            cadena = "0" * ceros + "1" * unos
            print("Se ha generado la siguiente cadena
↳ automáticamente:", cadena)
            break
```

```

        elif modo == "3":
            exit()
        else:
            print("Entrada inválida. Presione 1, 2, o 3.\n")
            continue
    return list(cadena) + ["B"] # Espacio en blanco al final

def configurar_interfaz(cinta):
    ventana = Tk()
    ventana.title("Máquina de Turing")
    ventana.geometry("1600x800")

    mostrarCadena = Label(ventana, text="Cadena inicial: " +
        ↪ "".join(cinta[:-1]), font=("Arial", 24))
    mostrarCadena.pack(ipady=20)

    canvas = Canvas(ventana, width=1600, height=632)
    canvas.pack()
    canvas.update_idletasks()

    return ventana, canvas

def dibujar_cinta(canvas, cinta, puntero, ventana):
    canvas.delete("all")
    ancho_celda = 120
    alto_celda = 120

    centroX = canvas.winfo_width() / 2
    x_inicial = centroX - (len(cinta) * ancho_celda) / 2
    centroY = canvas.winfo_height() / 2

    for idx, simbolo in enumerate(cinta):
        color = "red" if idx == puntero else "green"
        x = x_inicial + idx * ancho_celda
        canvas.create_rectangle(x, centroY - alto_celda / 2, x +
            ↪ ancho_celda, centroY + alto_celda / 2, fill=color)
        canvas.create_text(x + ancho_celda / 2, centroY,
            ↪ text=simbolo, fill="white", font=("Arial", 24))

```

```

ventana.update()
time.sleep(1.2)

def mostrar_mensaje(mensaje, ventana):
    print(mensaje)
    mostrarMensaje = Label(ventana, text=mensaje, font=("Arial", 24))
    mostrarMensaje.pack(ipady=20)

def procesar_maquina(cinta, ventana, canvas, archivo_log,
    ↪ animar=True):
    estado = "q0"
    puntero = 0

    transiciones = {
        "q0": {"0": ("q1", "X", 1), "Y": ("q3", "Y", 1)},
        "q1": {"0": ("q1", "0", 1), "1": ("q2", "Y", -1), "Y": ("q1",
            ↪ "Y", 1)},
        "q2": {"0": ("q2", "0", -1), "X": ("q0", "X", 1), "Y": ("q2",
            ↪ "Y", -1)},
        "q3": {"Y": ("q3", "Y", 1), "B": ("q4", "B", 1)},
    }

    try:
        while estado != "q4":
            if animar and ventana is not None:
                dibujar_cinta(canvas, cinta, puntero, ventana)

            actual = cinta[puntero]

            #Obtener transición
            if estado in transiciones and actual in
            ↪ transiciones[estado]:
                nuevo_estado, nuevo_simbolo, movimiento =
                ↪ transiciones[estado][actual]

                direccion = "R" if movimiento == 1 else "L"

```

```

        ↪ archivo_log.write(f"({estado},{actual})->({nuevo_estado},{nuevo_s

cinta[puntero] = nuevo_simbolo
estado = nuevo_estado
puntero += movimiento
else:
    mensaje = f"Cadena rechazada en el estado {estado}
        ↪ con símbolo '{actual}'"
    if animar:
        mostrar_mensaje(mensaje, ventana)
    else:
        print(mensaje)
    return

#Extender cinta si es necesario
if puntero >= len(cinta):
    cinta.append("B")
elif puntero < 0:
    cinta.insert(0, "B")
    puntero = 0

#Si se llega a q4
if animar:
    mostrar_mensaje("Cadena aceptada", ventana)
else:
    print("Cadena aceptada")

except Exception as e:
    if animar:
        mostrar_mensaje(f"Error en la ejecución: {str(e)}",
            ↪ ventana)
    else:
        print(f"Error en la ejecución: {str(e)}")

def main():
    while True:
        cinta = obtener_cadena()

```

```

longitud = len(cinta) - 1 #Para que el símbolo 'B' no afecte

try:
    with open("registro_transiciones.txt", "w") as
        ↪ archivo_log:
        if longitud > LIMITE_ANIMACION:
            print(f"La cadena tiene {longitud} símbolos.
                ↪ Animación omitida.")
            procesar_maquina(cinta, None, None, archivo_log,
                ↪ animar=False)
        else:
            ventana, canvas = configurar_interfaz(cinta)
            procesar_maquina(cinta, ventana, canvas,
                ↪ archivo_log, animar=True)
            ventana.mainloop()
except Exception as e:
    print(f"Error inesperado: {str(e)}")

repetir = input("¿Desea probar otra cadena? (s/n):
    ↪ ").strip().lower()
if repetir != "s":
    break

if __name__ == "__main__":
    main()

```