



# Escuela Superior de Cómputo

TEORÍA DE LA COMPUTACIÓN

## **PRÁCTICA 9:** *TURING MACHINE*

Autor:

Rodrigo Gerardo Trejo Arriaga

Diciembre 2023

# Práctica 9:

## TURING MACHINE

### I. Introducción

La Máquina de Turing, nombrada así por su inventor Alan Turing en 1936, es un modelo matemático fundamental en la teoría de la computación. Turing introdujo este concepto en su trabajo seminal "On Computable Numbers, with an Application to the Entscheidungsproblem", que sentó las bases para la computación moderna [1].

### II. Definición de la Máquina de Turing

Una Máquina de Turing es un dispositivo teórico que manipula símbolos contenidos en una cinta de longitud infinita según un conjunto de reglas. Formalmente, una Máquina de Turing se define como una 7-tupla  $(Q, \Gamma, b, \Sigma, \delta, q_0, F)$  donde [1]:

- $Q$  es un conjunto finito de estados.
- $\Gamma$  es un conjunto finito de símbolos de la cinta, donde  $b \in \Gamma$  es el símbolo en blanco.
- $\Sigma \subseteq \Gamma \setminus \{b\}$  es el conjunto de símbolos de entrada.
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$  es la función de transición.
- $q_0 \in Q$  es el estado inicial.
- $F \subseteq Q$  es el conjunto de estados finales o de aceptación.

### III. Funcionamiento

La máquina comienza con la cinta conteniendo una cadena de símbolos del alfabeto de entrada y el resto de la cinta en blanco. La máquina opera de acuerdo con las reglas definidas por la función de transición  $\delta$ , que dicta el comportamiento de la máquina en función del estado actual y el símbolo leído de la cinta [2].

### IV. Relevancia en la Teoría de la Computación

La Máquina de Turing es un modelo importante por varias razones [2]:

1. **Universalidad:** La Máquina de Turing es capaz de simular cualquier algoritmo computacional.
2. **Modelo de Cómputo:** Sirve como un modelo estándar para definir lo que es computable.
3. **Problemas indecidibles:** Turing utilizó este modelo para demostrar la existencia de problemas indecidibles, es decir, problemas que no pueden ser resueltos por ninguna máquina de Turing.

## V. Definición de Identificadores (ID)

En el contexto de Máquinas de Turing (TM), un identificador (ID) es una cadena que representa el estado actual de la máquina. Se compone de tres partes: la parte izquierda de la cinta ( $\alpha$ ), el estado actual ( $q$ ), y la parte derecha de la cinta ( $\beta$ ). Esto se denota como  $\alpha q \beta$ , donde  $\alpha \beta$  es la porción de la cinta entre los símbolos no blancos más a la izquierda y más a la derecha, inclusive [3].

### V.1. Comportamiento de los ID [3]

- El estado  $q$  está inmediatamente a la izquierda del símbolo de la cinta que está siendo escaneado.
- Si  $q$  está al final derecho, está escaneando el símbolo blanco ( $B$ ).
- Si  $q$  escanea un  $B$  al principio de la izquierda, entonces cualquier  $B$  consecutivo a la derecha de  $q$  forma parte de  $\alpha$ .

### Notación de Transiciones

Para las transiciones de la TM, podemos usar los símbolos  $\vdash$  y  $\vdash^*$  para representar "se convierte en un movimiento" o "se convierte en cero o más movimientos", respectivamente [3].

### Ejemplo

Un ejemplo de las movidas de una TM puede ser representado como sigue [3]:

$$q_0 00 \vdash 0 q_0 0 \vdash 00 q_0 \vdash 00 q_1 \vdash^* 000 q_f$$

## VI. Instrucciones

El objetivo de esta práctica es implementar una Máquina de Turing que reconozca el lenguaje  $\{0^n 1^n | n \geq 1\}$ . Este lenguaje consiste en cadenas con un número igual de ceros seguidos por unos. La especificación de la máquina se basa en el ejercicio 8.2, segunda edición, del libro de John Hopcroft.

## Requerimientos

1. El programa debe aceptar una cadena de entrada definida por el usuario o generada automáticamente, con una longitud máxima de 1000 caracteres.
2. La salida del programa debe ser un archivo de texto con descripciones instantáneas de cada paso de la computación de la máquina.
3. Se debe proporcionar una animación de la Máquina de Turing para cadenas de entrada de longitud menor o igual a 16 caracteres.

## Tabla de Transiciones

La siguiente tabla representa la función de transición de la Máquina de Turing para el ejercicio mencionado.

State	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 transiciones de la Máquina de Turing para aceptar el lenguaje  $\{0^n 1^n | n \geq 1\}$ .

## VII. Resultados

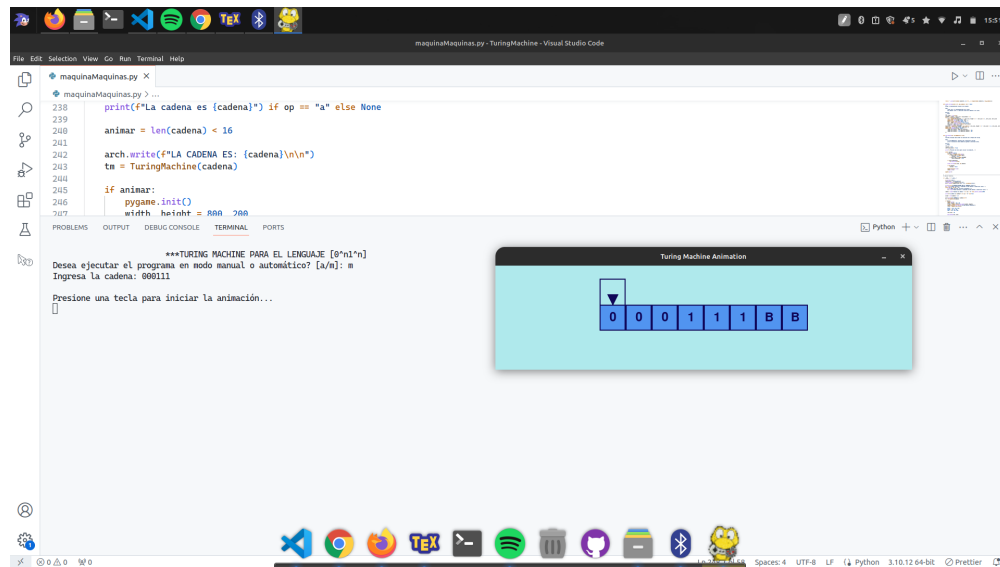


Figura 1: Cadena 000111 – Animación de la Máquina de Turing

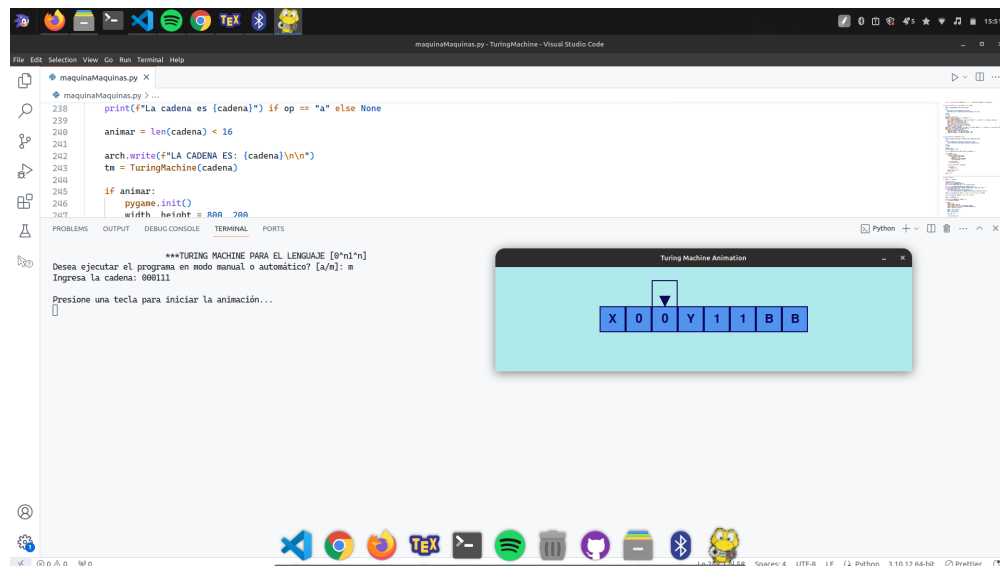


Figura 2: Cadena 000111 – Animación de la Máquina de Turing

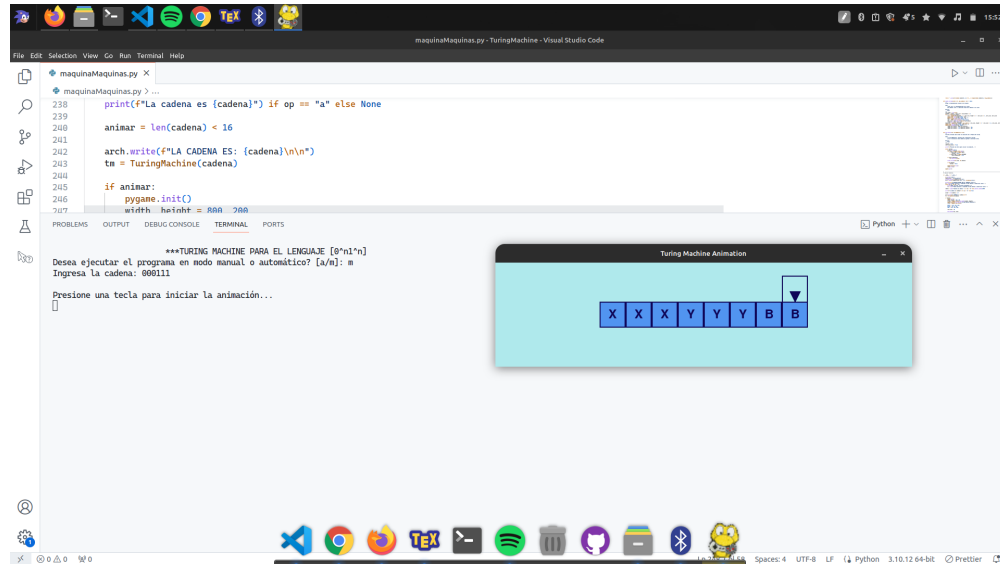


Figura 3: Cadena 000111 – Animación de la Máquina de Turing

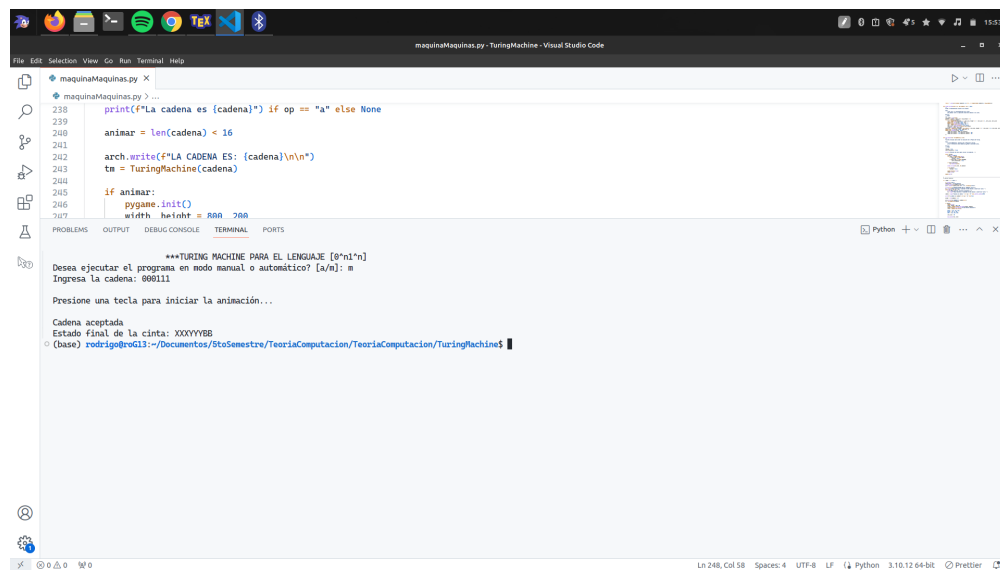


Figura 4: Cadena 000111 – Resultados de la Máquina de Turing

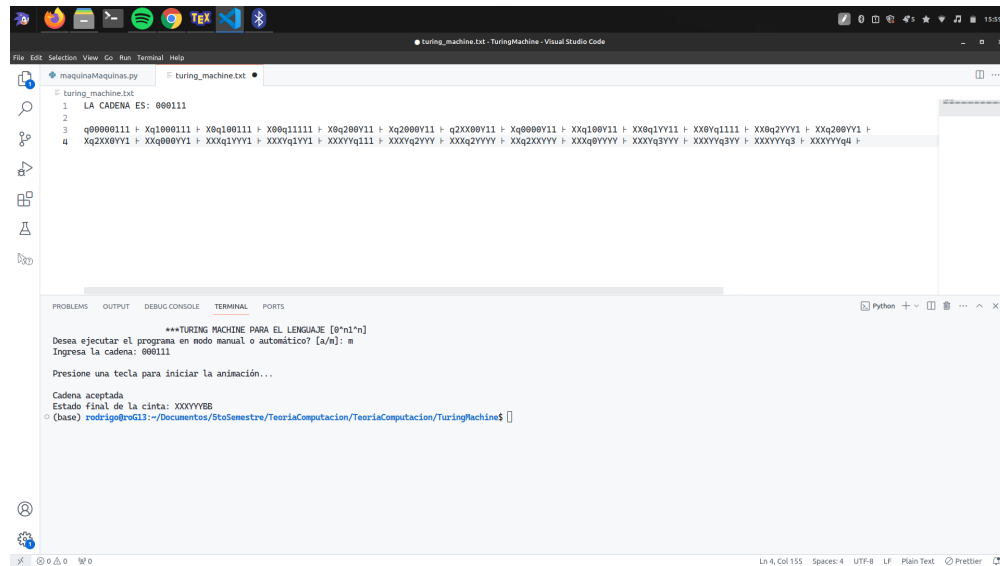


Figura 5: Cadena 000111 - Archivo de los ID's

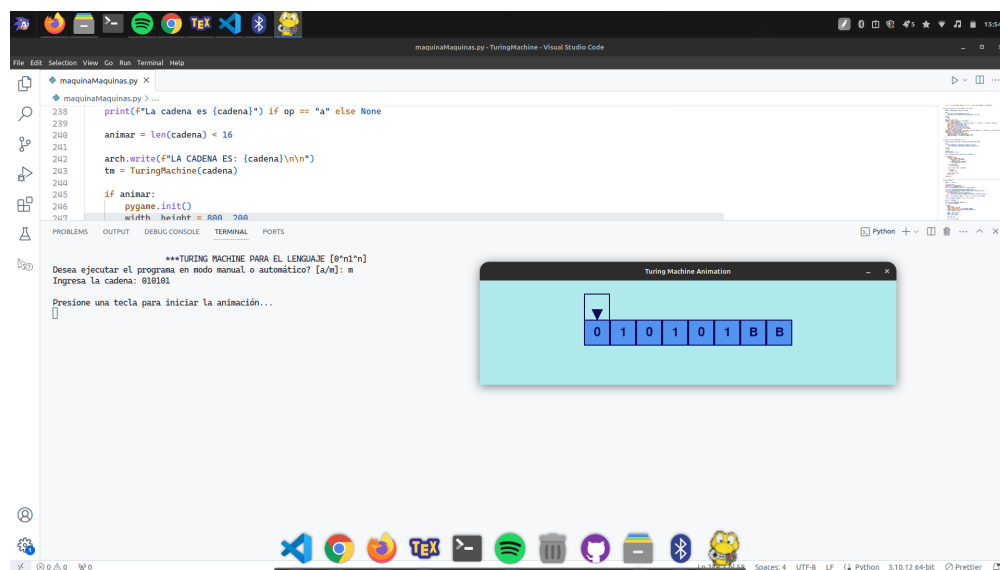


Figura 6: Cadena 010101 – Animación de la Máquina de Turing

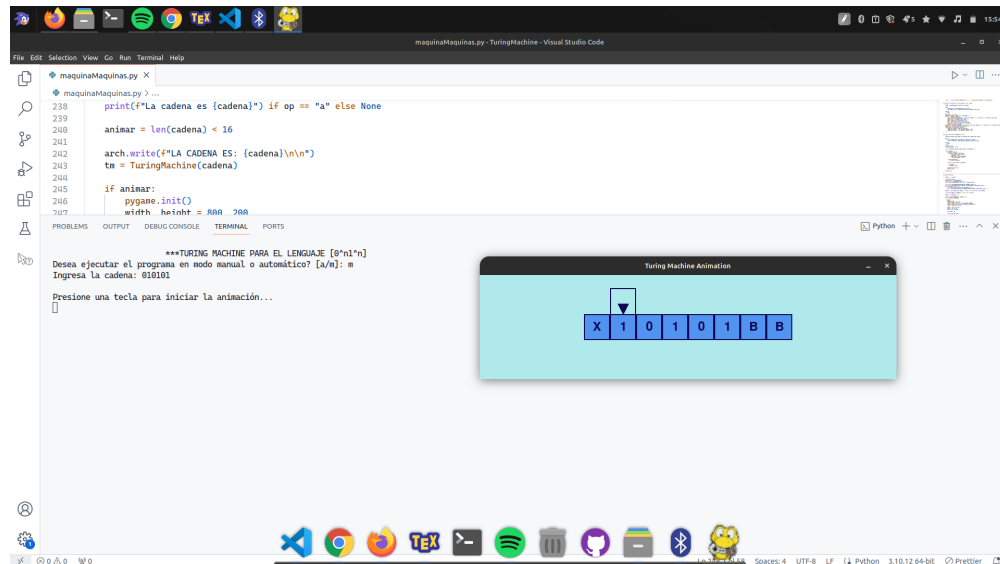


Figura 7: Cadena 010101 – Animación de la Máquina de Turing

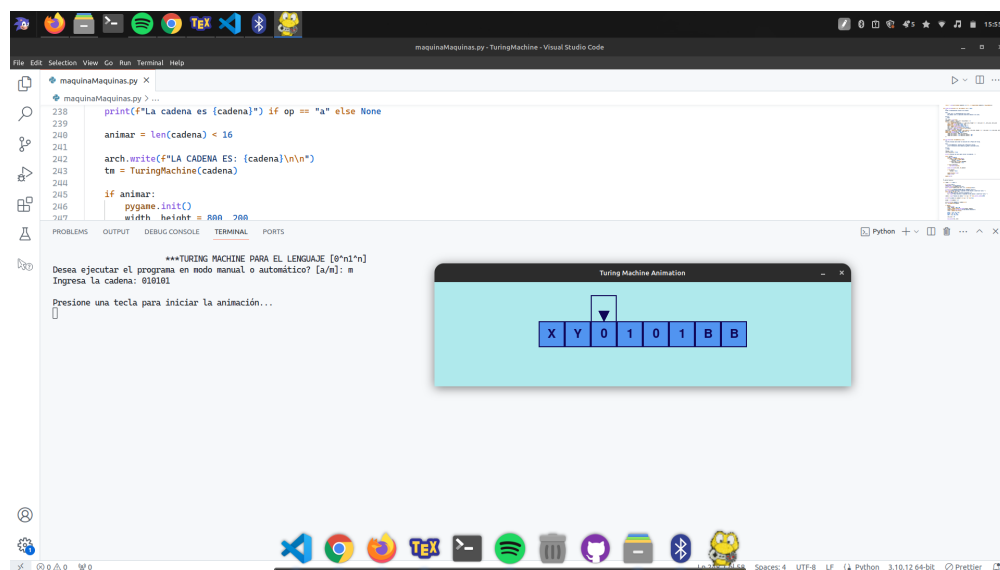
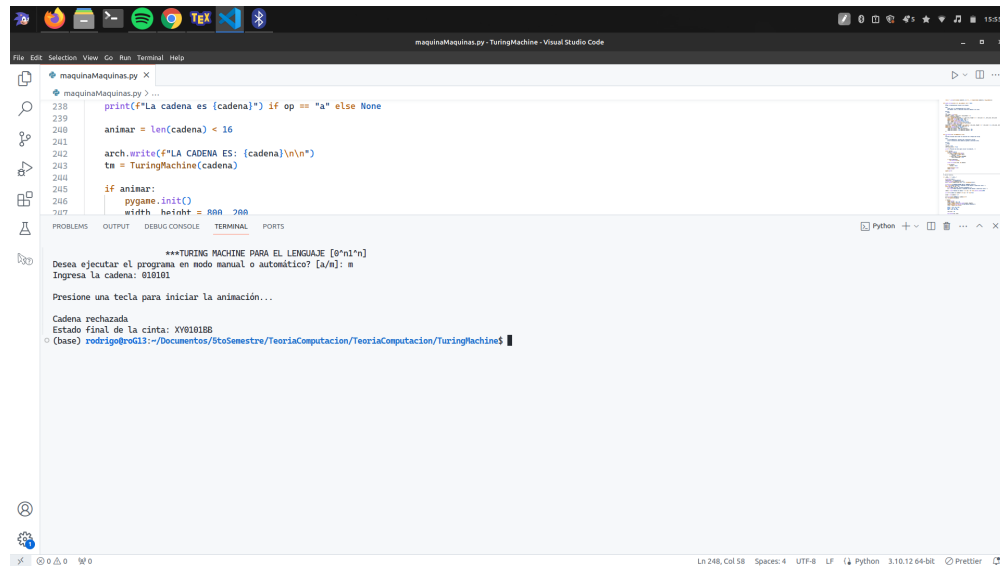


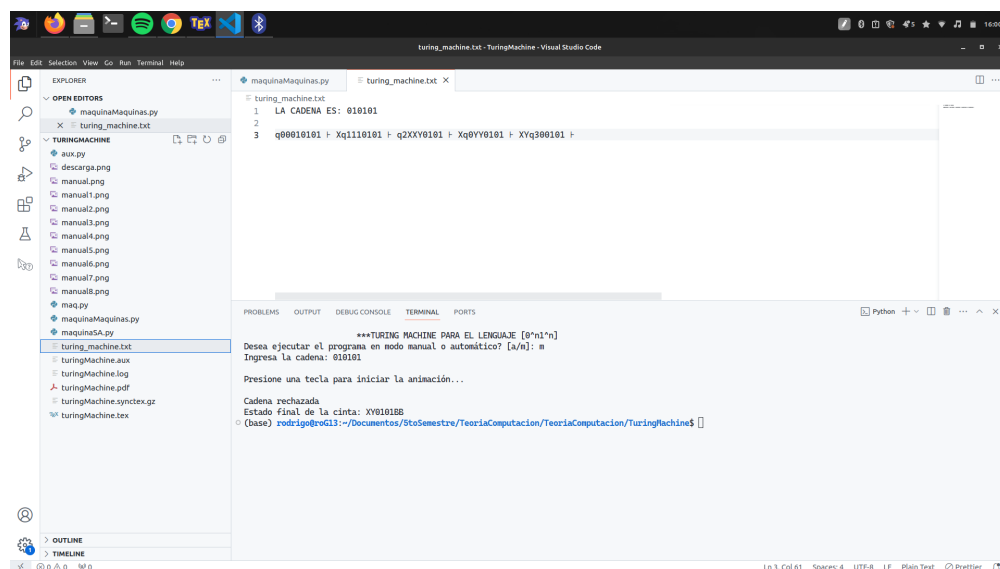
Figura 8: Cadena 010101 – Animación de la Máquina de Turing





```
maquinaMaquinas.py
238 print(f"La cadena es {cadena}") if op == "a" else None
239
240 animar = len(cadena) < 16
241
242 arch.write(f"LA CADENA ES: {cadena}\n\n")
243 tn = TuringMachine(cadena)
244
245 if animar:
246     pygame.init()
247     width, height = 800, 300
248
249 ***TURING MACHINE PARA EL LENGUAJE [0^n 1^n]
250
251 Desea ejecutar el programa en modo manual o automático? [a/n]: m
252 Ingrese la cadena: 010101
253
254 Presione una tecla para iniciar la animación...
255
256 Cadena rechazada
257 Estado final de la cinta: XY0101BB
258 (base) rodrigo@roG13:~/Documentos/5toSemestre/TeoriaComputacion/TeoriaComputacion/TuringMachine$
```

Figura 9: Cadena 010101 – Resultados



```
turing_machine.txt
1 LA CADENA ES: 010101
2
3 q00010101 -> Xq1110101 -> q2XXY0101 -> Xq0Y0101 -> XYq300101 ->
```

Figura 10: Cadena 010101 – Archivo de los ID's

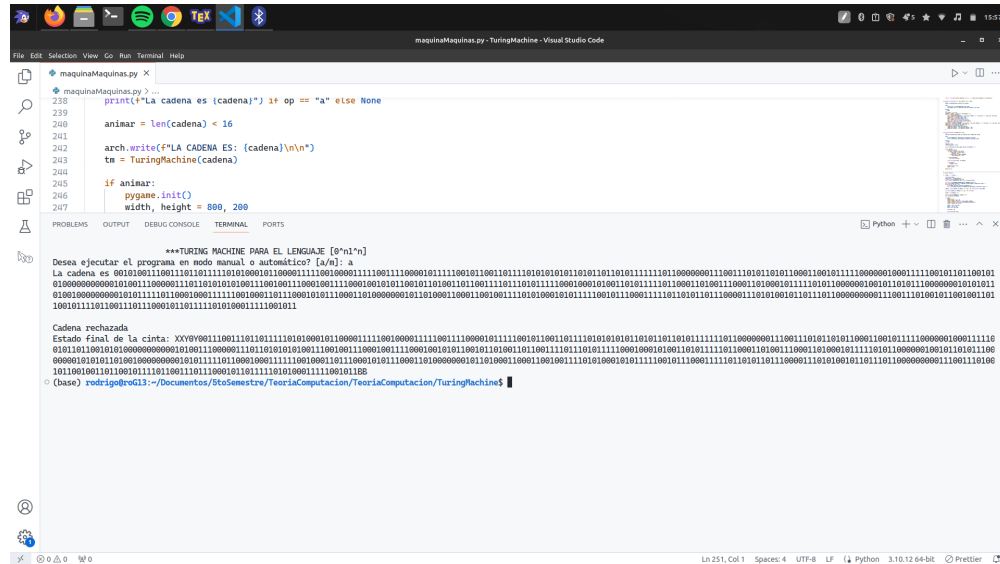


Figura 11: Modo automático

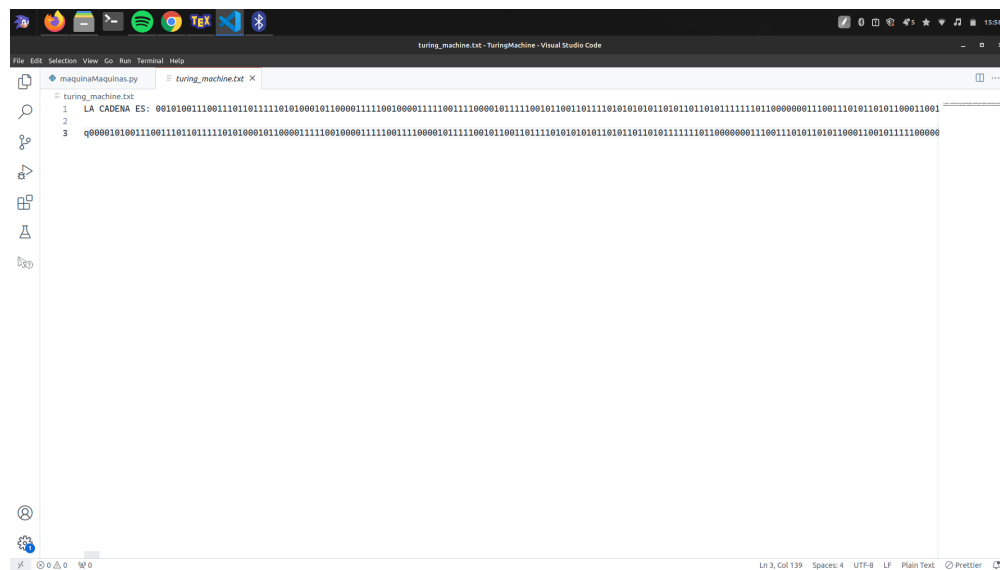


Figura 12: Modo automático - Archivo de ID's

## Conclusión

En esta práctica, he explorado y desarrollado una Máquina de Turing capaz de reconocer el lenguaje  $\{0^n 1^n | n \geq 1\}$ , un lenguaje clásico en la teoría de la computación que requiere un equilibrio preciso entre ceros y unos. A través de la implementación de la tabla de transiciones proporcionada y las pruebas realizadas con diversas cadenas de entrada, se ha demostrado la capacidad de la Máquina de Turing para determinar la aceptación de cadenas válidas dentro del lenguaje.

La práctica me permitió profundizar en el funcionamiento interno de las Máquinas de Turing, desde la definición de estados y transiciones hasta la visualización del procesamiento de la cinta en tiempo real. A pesar de su simplicidad conceptual, la Máquina de Turing que se ha construido ilustra la potencia del modelo de Turing para la computación y cómo incluso las máquinas más simples pueden realizar tareas de reconocimiento de patrones complejos.

Es así que, esta experiencia ha reforzado mi entendimiento de los conceptos teóricos subyacentes a las Máquinas de Turing y ha me proporcionado una base sólida para futuros estudios en el campo de la teoría de la computación y la ciencia de la computación en general.

## Bibliografía

- [1] "Turing Machine in TOC - GeeksforGeeks". GeeksforGeeks. Accedido el 25 de diciembre de 2023. [En línea]. Disponible: <https://www.geeksforgeeks.org/turing-machine-in-toc/>
- [2] "Definición de máquina de Turing y ejemplos – BorrowBits". BorrowBits. Accedido el 25 de diciembre de 2023. [En línea]. Disponible: <https://borrowbits.com/2013/03/maquinas-de-turing/>
- [3] J. Ullman. "CS154: Introduction to automata and complexity theory". The Stanford University Info-Lab. Accedido el 25 de diciembre de 2023. [En línea]. Disponible: [http://infolab.stanford.edu/~ullman/ialc/spr10/spr10.html#LECTURE %20NOTES](http://infolab.stanford.edu/~ullman/ialc/spr10/spr10.html#LECTURE%20NOTES)

## VIII. Anexo - Código de Implementación

```
1
2     '''
3     INSTITUTO POLITECNICO NACIONAL
4     ESCUELA SUPERIOR DE COMPUTO
5
6     INGENIERIA EN INTELIGENCIA ARTIFICIAL
7
8     TEORIA DE LA COMPUTACION
9     MAQUINA DE TURING
10
11     GRUPO: 5BM1
12     ALUMNO: TREJO ARRIAGA RODRIGO GERARDO
13
14     ESTE PROGRAMA GENERA UNA MAQUINA DE TURING QUE VALIDA SI UNA CADENA
15     PERTENECE AL LENGUAJE 0^n1^n Y:
16     i) SOLICITA AL USUARIO UNA CADENA A VALIDAR O LA GENERA DE MANERA
17     RANDOM
18     ii) ANIMA LA MAQUINA DE TURING SI LA LONGITUD DE LA PALABRA ES
19     MENOR A 16
20     iii) GENERA LA HISTORIA DE ID's EN UN ARCHIVO DE TEXTO
21
22     ULTIMA MODIFICACION: 25/12/2023
23     '''
24
25     # -----
26     # MODULOS Y LIBRERIAS IMPORTADAS
27
28     import pygame
29     import os
30     import random
31
32     # -----
33     # CLASES
34
35     class TuringMachine:
36
37         def __init__(self, cinta_string: str, simbolo_blanco: str = "B")
38             -> None:
39             """
40             Inicializa la Maquina de Turing.
41
42             Args:
43             cinta_string (str): La cadena de entrada para la maquina.
44             simbolo_blanco (str): El simbolo que representa un espacio
45             en blanco en la cinta.
46             """
47             self.cinta = list(cinta_string) + [simbolo_blanco] * 2
48             self.cabezal = 0
49             self.estado = 'q0'
50             self.simbolo_blanco = simbolo_blanco
```

```

47     self.detener = False
48     self.tabla_trans = {
49         ('q0', '0'): ('q1', 'X', 'R'),
50         ('q0', 'Y'): ('q3', 'Y', 'R'),
51         ('q1', '0'): ('q1', '0', 'R'),
52         ('q1', '1'): ('q2', 'Y', 'L'),
53         ('q1', 'Y'): ('q1', 'Y', 'R'),
54         ('q2', '0'): ('q2', '0', 'L'),
55         ('q2', 'X'): ('q0', 'X', 'R'),
56         ('q2', 'Y'): ('q2', 'Y', 'L'),
57         ('q3', 'Y'): ('q3', 'Y', 'R'),
58         ('q3', self.simbolo_blanco): ('q4', self.simbolo_blanco
59             , 'R'),
60     }
61
62     def es_aceptada(self) -> bool:
63         """
64         Verifica si la maquina de Turing ha llegado a un estado de
65         aceptacion.
66
67         Returns:
68         bool: True si esta en un estado de aceptacion, False de lo
69             contrario.
70         """
71         return self.estado == 'q4'
72
73     def es_rechazada(self) -> bool:
74         """
75         Verifica si la maquina de Turing ha llegado a un estado de
76         rechazo.
77
78         Returns:
79         bool: True si esta en un estado de rechazo, False de lo
80             contrario.
81         """
82         return self.estado != 'q4'
83
84     def transicion(self, arch) -> None:
85         """
86         Realiza una transicion de la maquina de Turing.
87
88         Args:
89         arch: El archivo de texto donde se guarda la historia de
90             IDs.
91         """
92         if self.detener:
93             return None
94
95         simbolo_act = self.cinta[self.cabezal]
96
97         alfa = ''.join(self.cinta[:self.cabezal]).replace("B", "")

```

```

95     q = self.estado
96     beta = ''.join(self.cinta[self.cabezal:]).replace("B", "")
97     arch.write(f"{alfa}{q}{simbolo_act if simbolo_act != 'B'
98                else ''}{beta} -> ")
99
100     accion = self.tabla_trans.get((self.estado, simbolo_act))
101
102     if accion is None:
103         self.detener = True
104     else:
105         nuevo_estado, nuevo_simbolo, direccion = accion
106         self.cinta[self.cabezal] = nuevo_simbolo
107         self.cabezal += 1 if direccion == 'R' else -1
108         self.estado = nuevo_estado
109
110     def run(self, arch) -> bool:
111         """
112         Ejecuta la Maquina de Turing hasta que se detiene.
113
114         Args:
115         arch: El archivo de texto donde se guarda la historia de
116               IDs.
117
118         Returns:
119         bool: True si la cadena fue aceptada, False de lo contrario
120               .
121         """
122         while not self.detener:
123             self.transicion(arch)
124
125         return self.estado == 'q4'
126
127     # -----
128     # FUNCIONES
129
130
131     def eliminar_archs(nombre_arch: str) -> None:
132         """Funcion que elimina un archivo si existe en el directorio
133
134         Args:
135         nombre_arch (str): Nombre del archivo que deseas eliminar
136         """
137         archivo1 = nombre_arch
138         if os.path.exists(archivo1):
139             os.remove(archivo1)
140
141
142     def generar_cadena(long_cadena:int) -> str:
143         """Funcion que genera la cadena binaria random
144
145         Args:

```

```
146     long_cadena (int): Longitud de la palabra binaria
147
148     Returns:
149     str: Palabra binarias
150     """
151     return ''.join([str(random.randint(0, 1)) for _ in range(random
152         .randint(2, long_cadena))])
153
154 def animar_turing(cinta: str, pos_cabecal: int) -> None:
155     """
156     Anima la representacion visual de la maquina.
157
158     Args:
159     cinta (str): La representacion de la cinta.
160     pos_cabecal (int): La posicion actual del cabecal en la cinta.
161
162     Returns:
163     None
164     """
165     num_celdas = len(cinta)
166     acomodo = (width - (cell_size * num_celdas)) // 2
167     for i in range(num_celdas):
168         rect = pygame.Rect(acomodo + i * cell_size, height // 2 -
169             cell_size // 2, cell_size, cell_size)
170         pygame.draw.rect(screen, GRAY, rect)
171         pygame.draw.rect(screen, BLACK, rect, 2)
172         font = pygame.font.SysFont(None, 36)
173         text = font.render(cinta[i], True, BLACK)
174         text_rect = text.get_rect(center=rect.center)
175         screen.blit(text, text_rect)
176         head_rect = pygame.Rect(acomodo + pos_cabecal * cell_size,
177             height // 2 - cell_size * 1.5, cell_size, cell_size * 2)
178         pygame.draw.rect(screen, BLACK, head_rect, 2)
179         pygame.draw.polygon(screen, BLACK, [
180             (head_rect.centerx, head_rect.centery),
181             (head_rect.centerx - 10, head_rect.centery - 20),
182             (head_rect.centerx + 10, head_rect.centery - 20)
183         ])
184
185 def main_animar(tm: TuringMachine, arch):
186     """
187     Funcion principal para animar la ejecucion de la Maquina de
188     Turing.
189
190     Args:
191     tm (TuringMachine): Instancia de la Maquina de Turing.
192     arch: El archivo de texto donde se guarda la historia de IDs.
193
194     Returns:
195     None
196     """
197     running = True
```





```
247     pygame.init()
248     width, height = 800, 200
249     screen = pygame.display.set_mode((width, height))
250     pygame.display.set_caption('Turing Machine Animation')
251     clock = pygame.time.Clock()
252
253     WHITE = (175, 233, 236)
254     BLACK = (15, 6, 88)
255     GRAY = (81, 148, 240)
256
257     cell_size = 50
258
259     main_animar(tm, arch)
260
261     else:
262         is_accepted = tm.run(arch)
263
264     if tm.es_aceptada():
265         print("Cadena aceptada")
266     elif tm.es_rechazada():
267         print("Cadena rechazada")
268     else:
269         print("Detenida!!!")
270
271     print(f"Estado final de la cinta: {''.join(tm.cinta)}")
272     arch.close()
```