Máquina de Turing que duplica cadena de unos

Alumno: Monroy Martos Elioth
Profesor: Genaro Juárez Martínez
Materia: Computing Selected Topics
Grupo: 3CM8

21 de septiembre de 2018

Índice

1. Introducción

Las máquinas de Turing son un dispositivo que permite idealmente resolver cualquier problema, debido a que tiene una memoria infinita, sin embargo está implementación no es posible de realizar y deben ser acotadas a una memoria o cinta finita, de tal forma que se dice que si una máquina de Turing es capaz de resolver un problema, este problema es computable (quiere decir que una computadora lo puede resolver). En caso contrario, se dice que el problema no es computable.[1]

El ejercicio a realizar en está actividad, es diseñar e implementar una máquina de Turing que duplique la cantidad de unos de una cadena ingresada, y que además, se muestre el procedimiento en una animación y se escriba en un archivo el historial de los pasos realizados por la máquina de Turing.

2. Desarrollo

Para la elaboración del programa, se realizó el siguiente diagrama que modela los estados y las transiciones que tiene la máquina.

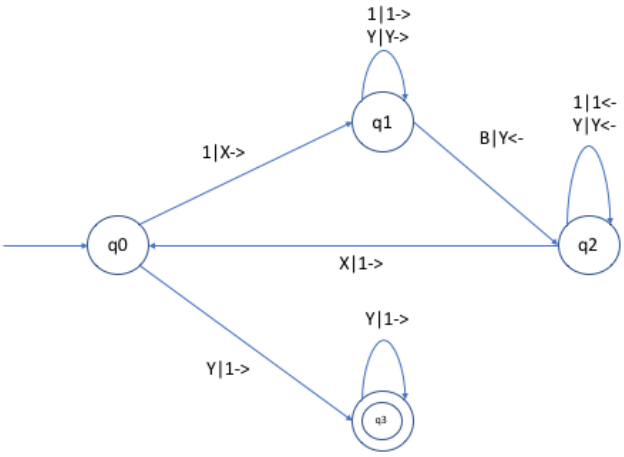

Figura 1: Estados y transiciones usados

Posteriormente, para el modelado de la máquina de Turing se realizó la siguiente implementación en Python 3.7, la cual es una clase que sirve para trabajar con la misma:

maquina.py

```
class MaquinaTuring(object):
  def __init__(self, estados, alfabeto, transiciones, inicial,
   final,cadena):
    self.estados=estados
    self.alfabeto=alfabeto
    self.transiciones=transiciones
    self.inicial=inicial
    self.estado_actual=self.inicial
    self.final=final
    self.apuntador=0
    self.blanco="B"
    self.cadena=list(cadena)
```

```
    self.direccion=""
  def evaluar(self):
    if len(self.cadena)-1<self.apuntador:
      caracter=self.blanco
    else:
      caracter=self.cadena[self.apuntador]

    if caracter in self.alfabeto:
      transicion=(self.estado_actual,caracter)
      if transicion in self.transiciones:
        siguiente=self.transiciones[transicion]
        if len(self.cadena)-1<self.apuntador:
          self.cadena.append(self.blanco)
        if self.apuntador<0:
          self.cadena.insert(0,self.blanco)
        self.cadena[self.apuntador]=siguiente[1]
        if siguiente[2]=="R":
          self.apuntador+=1
        else:
          self.apuntador=self.apuntador-1

        self.estado_actual=siguiente[0]
        self.direccion=siguiente[2]
        if self.estado_actual in self.estados:
          return True
        else:
          return False
      else:
        return False
    else:
      return False
  def esFinal(self):
    if self.estado_actual in self.final:
      if len(self.cadena)-1<self.apuntador:
        return True
    return False
```

Por si sola está máquina no tiene funcionalidad, por lo cual fue necesario el uso de la misma en otro archivo, el cual se encarga de manejar todo el funcionamiento del programa, como recibir la entrada (cadena de unos), y utilizar tkinter para realizar la animación de la máquina de Turing.

```
from tkinter import *
from tkinter import font as tkfont
import time
from maquina import MaquinaTuring
```

```
5
6  entrada = input("Ingrese la cadena a duplicar: ")
7  estados=["q0","q1","q2","q3"]
8  alfabeto=["1","Y","X","B"]
9  transiciones={
10   ("q0", "1"): ("q1", "X", "R"),
11   ("q0", "Y"): ("q3", "1", "R"),
12   ("q1", "1"): ("q1", "1", "R"),
13   ("q1", "Y"): ("q1", "Y", "R"),
14   ("q1", "B"): ("q2", "Y", "L"),
15   ("q2", "1"): ("q2", "1", "L"),
16   ("q2", "X"): ("q0", "1", "R"),
17   ("q2", "Y"): ("q2", "Y", "L"),
18   ("q3", "Y"): ("q3", "1", "R"),
19 }
20
21 maquina = MaquinaTuring(estados, alfabeto, transiciones, estados
       [0], estados[3], entrada)
22
23 master = Tk()
24 master.title("Turing")
25 c = Canvas(master, width=400, height=300)
26 c.pack()
27 tipo_letra = tkfont.Font(family="Helvetica", size=20)
28
29 flecha = c.create_line(175, 150, 175, 175, arrow=LAST, width=2)
30 texto = c.create_text(165, 200, text="".join(maquina.cadena),
       font=tipo_letra, anchor=W)
31 estado = c.create_text(120, 150, text=maquina.estado_actual,
       font=tipo_letra, anchor=W)
32
33 archivo = open("salida.txt", "w")
34 while not maquina.esFinal():
35   print("Cadena: "+"".join(maquina.cadena))
36   print("Estado actual: "+str(maquina.estado_actual)+",
       apuntador: "+str(maquina.apuntador+1)+"\n")
37
38   archivo.write("Cadena: "+"".join(maquina.cadena)+"\n")
39   archivo.write("Estado actual: "+str(maquina.estado_actual)+",
       apuntador: "+str(maquina.apuntador+1)+"\n")
40   if not maquina.evaluar():
41     print("----------------")
42     archivo.write("----------------------")
43     archivo.write("\n")
44   print("Siguiente estado: "+str(maquina.estado_actual))
```

```
45    print("--------------------")
46
47    archivo.write("Siguiente estado: "+str(maquina.estado_actual))
48    archivo.write("\n")
49    archivo.write("--------------------")
50    archivo.write("\n")
51
52    master.update()
53    time.sleep(.8)
54    c.itemconfigure(texto, text="".join(maquina.cadena))
55    c.itemconfigure(estado, text=str(maquina.estado_actual))
56    if maquina.direccion == 'R':
57      c.move(flecha, 9, 0)
58    else:
59      c.move(flecha, -9, 0)
60
61 print("Resultado: "+"".join(maquina.cadena))
62 archivo.write("Resultado: "+"".join(maquina.cadena)+"\n")
63 archivo.flush()
64 master.mainloop()
```

3. Resultados

Para comprobar la funcionalidad del programa, se realizó la siguiente prueba, se ingresó la cadena “11”.
Los resultados obtenidos fueron los siguientes:

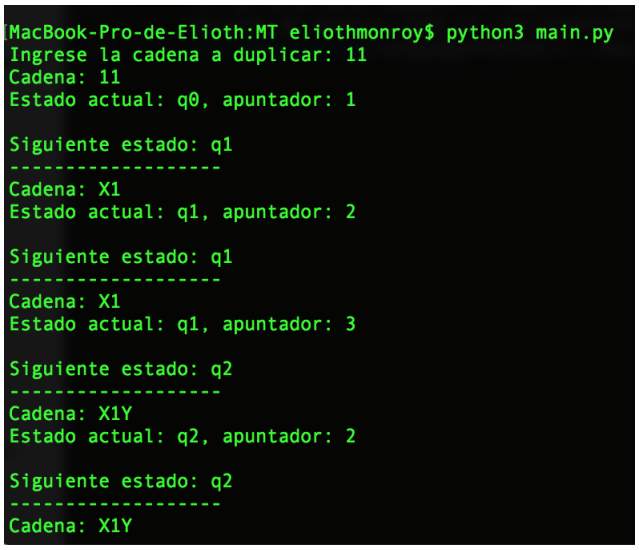

Figura 2: Resultados prueba (1)

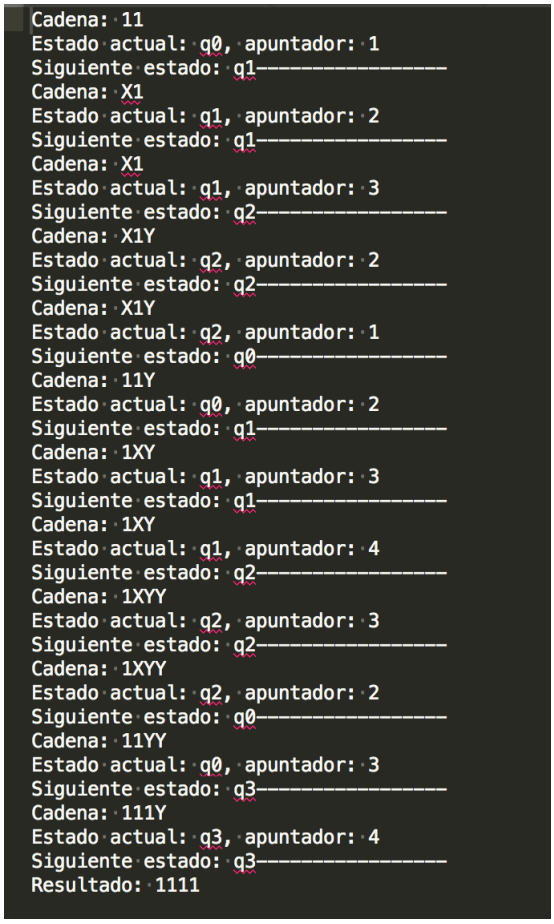

Figura 3: Resultados prueba (2)

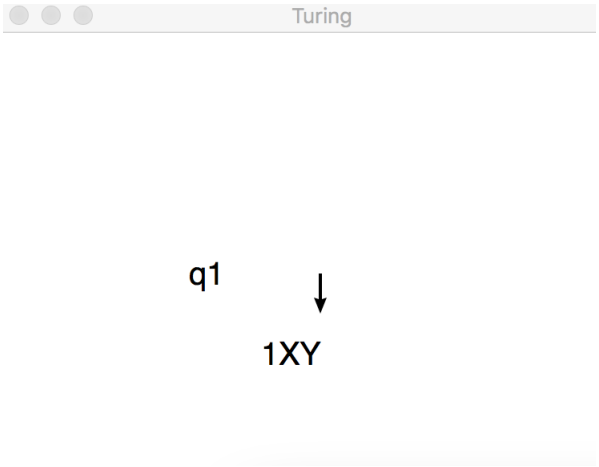

Figura 4: Resultados prueba (3)

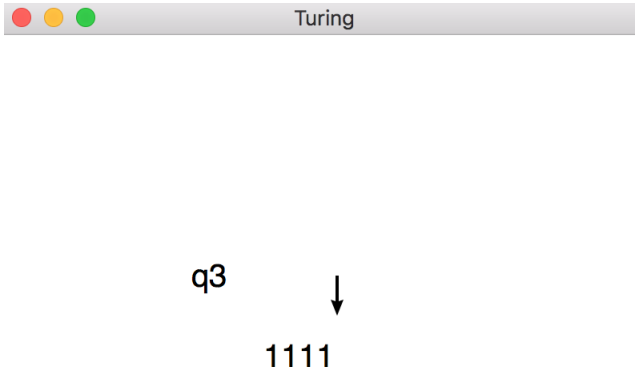

Figura 5: Resultados prueba (4)

Referencias

[1] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introducción a La Teoría De Autómatas, Lenguajes Y Computación.* Addison-Wesley, 2007.