Universidad de Panamá

Facultad de Informática, Electrónica y Comunicación

Escuela de Ingeniería de informática

Computabilidad y Complejidad de Algoritmo

Laboratorio #2

Simulación de una Máquina de Turing

(en Python)

Integrantes:

Jesús de Gracia / 8-1086-1646

Gisela Ojo / 8-904-2058

Profesor

Ayax Mendoza

Fecha de Entrega

9 de octubre de 2020

Laboratorio #2

Paso 1: Se muestra el código y el diseño de la interfaz para la simulación de la máquina de Turing en Python.

- Fragmento de Código:

```
    def stopTM(self):
        """Detiene la ejecución continua de la MT
    y Cancela todas las actualizaciones
        """
        if self.tm != None:
            for job in self._jobs:
                self.main.after_cancel(job)
            self.tm.go_back_to_step(self.lastRunStep)
        self._jobs = []

    # Callbacks
    def setTape(self, *args):
        """Devolución de llamada para cuando se cambia la entrada de cinta.
        Informa al MT y reiniciar la ejecución.
        """
        if self.tm != None:
            self.tm.set_input_string(self.textTapeInput.get())
            self.resetTM()

    def setBidirectional(self, *args):
        """Devolución de llamada para cuando se cambia la opción bidireccional.
        Informar a la MT, restablecer la ejecución y deshabilitar la otra casilla de verificación
        """
        if self.tm != None:
            self.tm.set_bidirectional(self.bidirectional.get())
            self.resetTM()
        else:
            if self.bidirectional.get():
                self.drawFirstTape()
            else:
                self.canvasSimOut.delete('left')

        if (self.bidirectional.get()):
            self.checkbox2Tape.configure(state='normal')
        else:
            self.checkbox2Tape.configure(state='disabled')

    def setTwoTape(self, *args):
        """Devolución de llamada para cuando se cambia la opción de dos cintas.
```

```
        else:
            tape1 = config[0][0]
            tape2 = config[0][1]
            position1 = config[3][0] - 8
            position2 = config[3][1] - 8
            for j in range(17):
                if (position1 + j) < 20000:
                    text1 = tape1[position1 + j] if tape1[position1 + j] != " " else ""
                    self.canvasSimOut.create_text(50 * j + 27, starty + 25, text=text1, font="Times 20", tag='text')
                if (position2 + j) < 20000:
                    text2 = tape2[position2 + j] if tape2[position2 + j] != " " else ""
                    self.canvasSimOut.create_text(50 * j + 27, starty + 175, text=text2, font="Times 20", tag='text')

    def writeOutText(self, config, step=None):
        """Write out the given configuration of the machine in the text output."""
        if step == None:
            step = self.tm.step
        self.lastRunStep = step
        self.textSimOut.config(state='normal')
        if (type(config) != str):
            self.textSimOut.insert('end', "Pasos: " + str(step) + '\n')
            self.textSimOut.insert('end', self.tm.format_config(config))
            if config[4] < 0:
                if config[4] == -1:
                    result = 'Aceptado'
                elif config[4] == -2:
                    result = 'Aceptado'
                else:
                    result = 'Detenido'
                self.textSimOut.insert('fin', result + '\n')
                self.textSimOut.insert('fin', str(step) + ' pasos' + '\n')
                tape = ''
                if not self.two_tape.get():
                    for j in range(config[1], config[2] + 1):
                        tape += config[0][j]
                    self.textSimOut.insert('fin', tape + '\n')
                else:
                    for j in range(config[1][0], config[2][0] + 1):
                        tape += config[0][0][j]
                    self.textSimOut.insert('fin', tape + '\n')
```

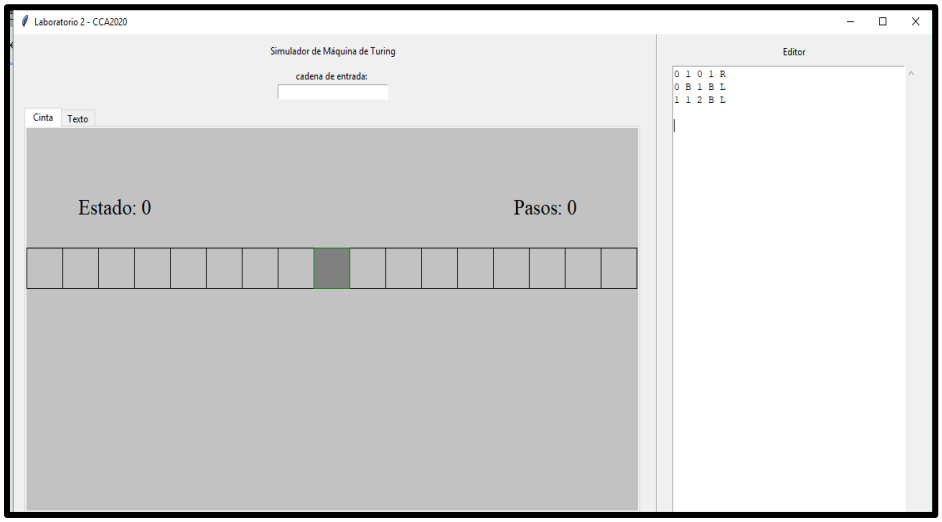
```
                    tape = ''
                    for j in range(config[1][1], config[2][1] + 1):
                        tape += config[0][1][j]
                    self.textSimOut.insert('fin', tape + '\n')

        else:
            self.textSimOut.insert('fin', config)
        self.textSimOut.config(state='disabled')
        self.textSimOut.yview(tk.END)


def default_resize(frame):
    """Cuando se le da un marco, establece todos los pesos de filas y columnas de la cuadrícula en 1.
    Esto significa que el marco cambia de tamaño uniformemente con la ventana..
    """
    (rows, columns) = frame.grid_size()
    for i in range(rows):
        frame.grid_rowconfigure(i, weight=1)
    for i in range(columns):
        frame.grid_columnconfigure(i, weight=1)


root = tk.Tk()
try:
    img = tk.Image("photo", file="PRUEBA.gif")
    root.call('wm', 'iconphoto', root._w, img)
except Exception:
    pass
tm_gui = TMCCA(root)
root.minsize(width=WIDTH, height=HEIGHT)
default_resize(root)
root.grid_columnconfigure(2, weight=2)
root.mainloop()
```

- Interfaz:



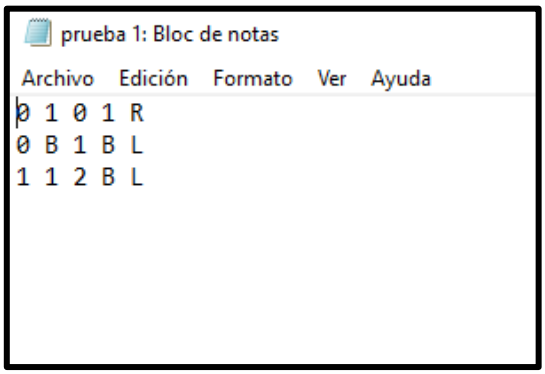
Paso 2: Se ingresa en un editor de texto con la extensión .tm las instrucciones para que se ejecute la máquina de Turing.

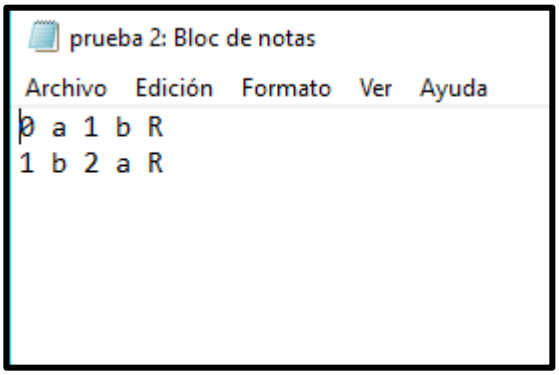
Definición de Lógica

Para iniciar el proceso de lectura, el numero 0 será predeterminado para el estado inicial y el numero -1 para el estado final.

Ingresando valores al programa

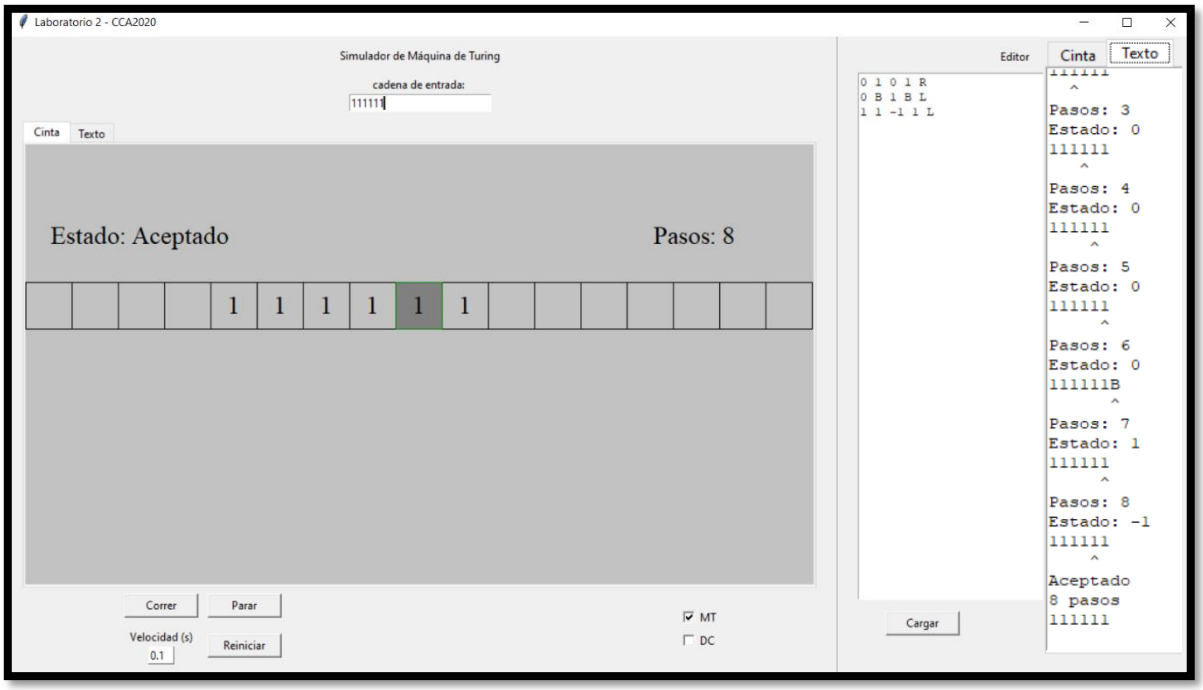
- Primera columna: estado inicial.
- Segunda columna: lectura
- Tercera columna: estado siguiente
- Cuarta columna: escritura
- Quinta columna: movimiento de la máquina.





Paso 3: Se realiza la ejecución de varias cadenas de entrada para

- **Cadena aceptada**



❖ Cadena no aceptada