



Instituto Politécnico Nacional
Escuela Superior de Cómputo

Teoría Computacional

Practica 8: Máquina de Turing

Alumna: Ramírez Galindo Karina

Profesor: Rosas Trigueros Jorge Luis

Fecha de realización: 25-11-2019

Fecha de entrega: 02-12-2019



Marco Teórico

MAQUINA DE TURING

Una máquina de Turing es una 7-tupla $M=(Q,\Sigma,\Gamma,s,\mathbb{b},F,\delta)$ [1] [2]

Donde:

- Q es un conjunto finito de estados
- Σ es el alfabeto de entrada
- Γ es un alfabeto llamado “alfabeto de la cinta”
- $s \in Q$ es el estado inicial
- $\mathbb{b} \in \Gamma$ es el símbolo blanco (no está en Σ)
- $F \subseteq Q$ es el conjunto de estados finales o de aceptación
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$ es una función parcial que se llama función de transición

Podemos visualizar una máquina de Turing como se muestra en la Imagen 1. La máquina consta de una unidad de control, que puede encontrarse en cualquiera de un conjunto finito de estados. Hay una cinta dividida en cuadrados o casillas y cada casilla puede contener un símbolo de entre un número finito de símbolos. [2]

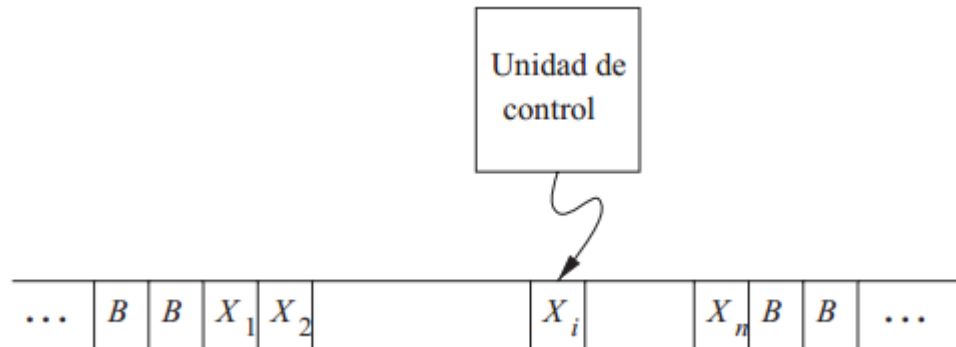


Imagen 1 Una máquina de Turing

Inicialmente, la entrada, que es una cadena de símbolos de longitud finita elegidos del alfabeto de entrada, se coloca en la cinta. Las restantes casillas de la cinta, que se extiende infinitamente hacia la izquierda y la derecha, inicialmente almacenan un símbolo especial denominado espacio en blanco. El espacio en blanco es un símbolo de cinta, pero no un símbolo de entrada, y pueden existir también otros símbolos de cinta además de los símbolos de entrada y del espacio en blanco. Existe una cabeza de la cinta que siempre está situada en una de las casillas de la cinta. Se dice que la máquina de Turing señala dicha casilla. Inicialmente, la cabeza de la cinta está en la casilla más a la izquierda que contiene la entrada. [3]

MOVIMIENTOS EN UNA MAQUINA DE TURING

Un movimiento de la máquina de Turing es una función del estado de la unidad de control y el símbolo de cinta al que señala la cabeza. En un movimiento, la máquina de Turing: [3] [2]

1. Cambiará de estado. El siguiente estado puede ser opcionalmente el mismo que el estado actual.
2. Escribirá un símbolo de cinta en la casilla que señala la cabeza. Este símbolo de cinta reemplaza a cualquier símbolo que estuviera anteriormente en dicha

casilla. Opcionalmente, el símbolo escrito puede ser el mismo que el que ya se encontraba allí.

3. Moverá la cabeza de la cinta hacia la izquierda o hacia la derecha. En nuestro formalismo, exigiremos que haya un movimiento y no permitiremos que la cabeza quede estacionaria. Esta restricción no limita lo que una máquina de Turing puede calcular, ya que cualquier secuencia de movimientos con una cabeza estacionaria podría condensarse, junto con el siguiente movimiento de la cabeza de la cinta, en un único cambio de estado, un nuevo símbolo de cinta y un movimiento hacia la izquierda o hacia la derecha.

EJEMPLO 1:

$$Q=\{q1,q2\}$$

$$\Sigma=\{a,b\}$$

$$\Gamma=\{a,b,\sqcup\}$$

$$F=\{q2\}$$

$$S=q1$$

$$\delta(q1,a) = (q1,a,R)$$

$$\delta(q1,b) = (q1,a,R)$$

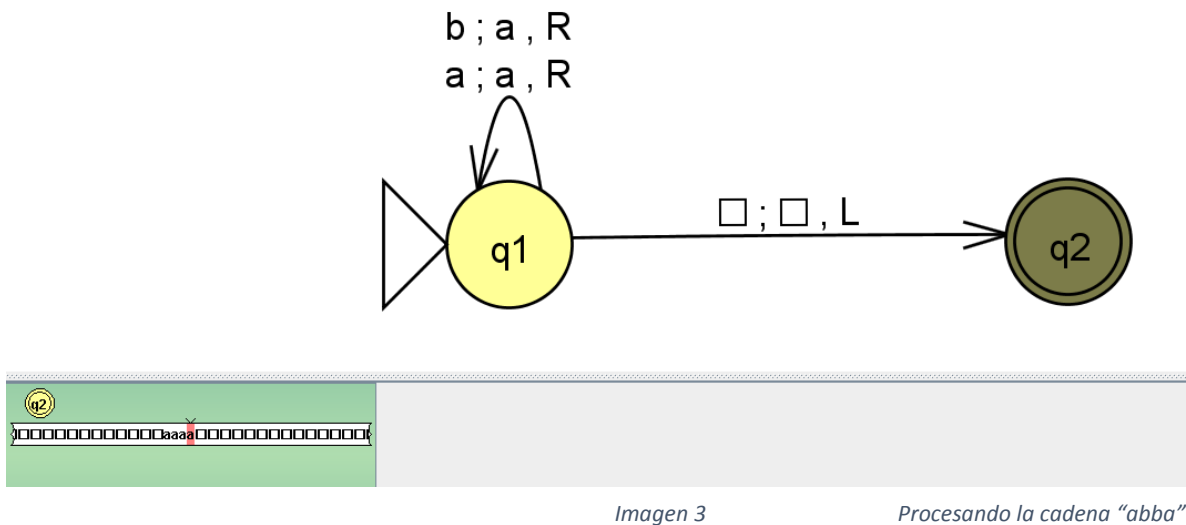
$$\delta(q1,\sqcup) = (q2,\sqcup,L)$$



Imagen 2

Máquina de Turing del ejemplo 1

Probando la cadena “abba” paso por paso obtenemos lo siguiente:



Como se puede observar en la Imagen 3, esta Máquina de Turing convierte las b's en a's.

IMPORTANCIA DE LA MAQUINA DE TURING

La máquina fue descrita por Turing como una *máquina automática* en 1936 en la revista Proceedings of the London Mathematical Society, en el estudio “Sobre los números computables, con una aplicación al Entscheidungsproblem”. La idea básica de todo el formalismo de Turing es que reduce el concepto de ‘método’ a operaciones simples que pueden, de manera incuestionable, ser efectuadas. Estas operaciones sencillas son la serie de instrucciones lógicas en las que se basan las acciones del artillero.. [1]



Turing definía su creación como:

“ ...una ilimitada capacidad de memoria obtenida en forma de una cinta infinita marcada con cuadrados, en cada uno de los cuales podría imprimirse un símbolo.

En todo momento hay un símbolo en la máquina, llamado el 'símbolo leído', que la máquina puede alterar. El comportamiento de la maquina está en parte determinado por el símbolo leído, pero no por los símbolos en otros lugares de la cinta.

Una de las operaciones elementales de la máquina es el movimiento hacia adelante y hacia atrás a través de la máquina de la cinta, por lo que cualquier símbolo en la cinta puede convertirse en el símbolo leído. (Turing 1948, p. 61)". [1]

Material y equipo

-  Sistema Operativo Ubuntu
-  Python 3

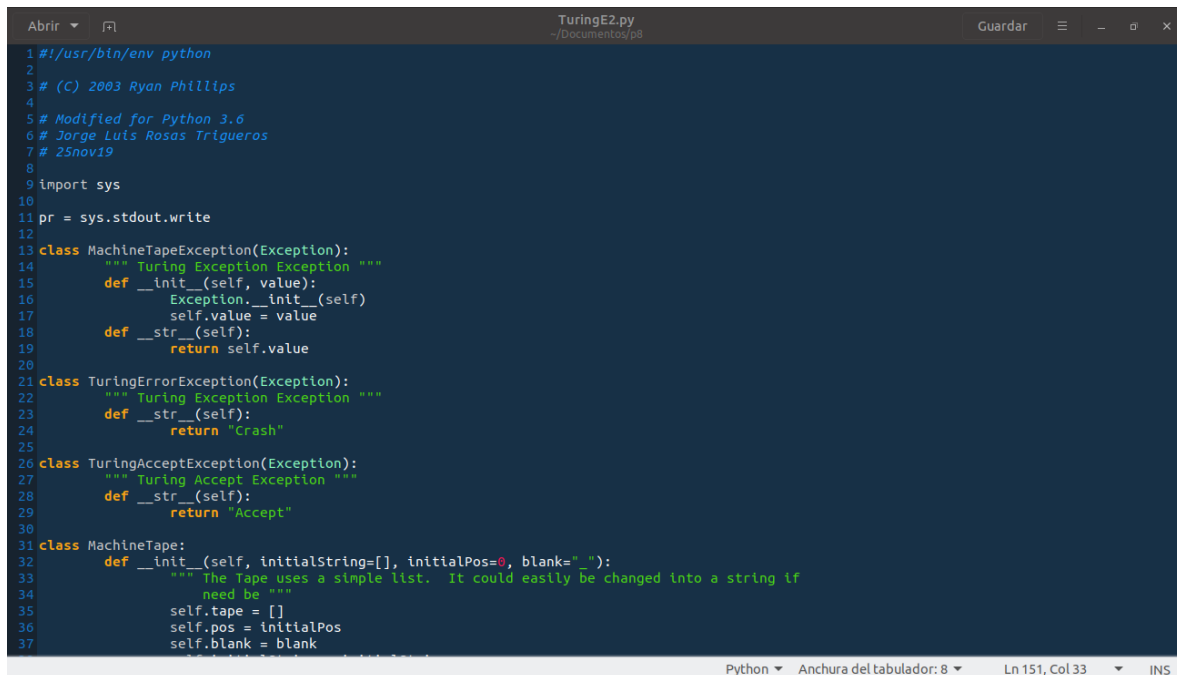
Desarrollo de la práctica

Objetivo: diseñar y programar en python máquinas de turing que acepten diferentes lenguajes.

Ejercicios: Diseñar las siguientes máquinas de Turing

- 1) Se le presenta una cadena de $\{0,1\}^*$ y cambia todos los 0's por 1's y todos los 1's por 0's

El código base para todos los ejercicios es el siguiente:



```
1 #!/usr/bin/env python
2
3 # (C) 2003 Ryan Phillips
4
5 # Modified for Python 3.6
6 # Jorge Luis Rosas Trigueros
7 # 25nov19
8
9 import sys
10
11 pr = sys.stdout.write
12
13 class MachineTapeException(Exception):
14     """ Turing Exception Exception """
15     def __init__(self, value):
16         Exception.__init__(self)
17         self.value = value
18     def __str__(self):
19         return self.value
20
21 class TuringErrorException(Exception):
22     """ Turing Exception Exception """
23     def __str__(self):
24         return "Crash"
25
26 class TuringAcceptException(Exception):
27     """ Turing Accept Exception """
28     def __str__(self):
29         return "Accept"
30
31 class MachineTape:
32     def __init__(self, initialString=[], initialPos=0, blank="_"):
33         """ The Tape uses a simple list. It could easily be changed into a string if
34             need be """
35         self.tape = []
36         self.pos = initialPos
37         self.blank = blank
```

Imagen 4

```

Abrir  TuringE2.py  Guardar  -  +  x
~/Documentos/p8

37     self.blank = blank
38     self.initialString = initialString
39     if len(initialString) > 0:
40         for ch in initialString:
41             self.tape.append(ch)
42     else:
43         self.tape.append(blank)
44
45     def reinit(self):
46         self.__init__(self.initialString)
47
48     def move(self, check_char, changeto_char, direction):
49         """ Only R, L directions are supported """
50         # check to see if the character under the head is what we need
51         if check_char != self.tape[self.pos]:
52             raise MachineTapeException ("Tape head doesn't match head character")
53
54         # at this point the head is over the same character we are looking for
55         # change the head character to the new character
56         self.tape[self.pos] = changeto_char
57
58         if direction == "L":
59             self.move_left()
60         elif direction == "R":
61             self.move_right()
62         else: raise MachineTapeException ("Direction is invalid")
63
64     def read(self):
65         """ return the character over the head """
66         return self.tape[self.pos]
67
68     def move_left(self):
69         if self.pos <= 0:
70             self.tape.insert(-1, self.blank)
71             self.pos = 0
72         else:
73             self.pos -= 1

```

Python Anchura del tabulador: 8 Ln 151, Col 33 INS

Imagen 5

Código General Parte 2

```

Abrir  TuringE2.py  Guardar  -  +  x
~/Documentos/p8

74
75     def move_right(self):
76         self.pos += 1
77         if self.pos >= len(self.tape): self.tape.append(self.blank)
78
79     def show(self):
80         """ print the tape """
81         for ch in self.tape:
82             pr(ch)
83         pr("\n"); pr(" *self.pos + "^"); pr("\n")
84
85 """
86 The program structure for the TM is created with a dictionary.
87 To step algorithm:
88     1. Check to see if the length of the string is zero and if we
89        are in a final state
90     2. If the currentstate is in the final states then raise an Accept
91     3. If the currentstate is not in the program then raise an Error
92     4. Check the head character
93     5. If the head character is not in the program and in the current state then
94        raise an Error
95     6. Retrieve from the dictionary the dest_state, char_out, and movement
96     7. set the current state to the new state
97     8. write the tape, and move the head
98
99 Program Layout:
100 [state][char_in] --> [(dest_state, char_out, movement)]
101 """
102
103 class TuringMachine:
104     def __init__(self, initialString, finalStates=[], blank="_"):
105         self.blank = blank
106         self.tape = MachineTape(initialString)
107         self.fstates = finalStates
108         self.program = {}
109         self.initState = 0
110         self.state = self.initState

```

Python Anchura del tabulador: 8 Ln 73, Col 39 INS

Imagen 6

Código General Parte 3

```

111         self.lenStr = len(initialString)
112
113     def reinit(self):
114         self.state = self.initState
115         self.tape.reinit()
116
117     def addTransition(self, state, char_in, dest_state, char_out, movement):
118         # if not self.program.has_key(state): #Python 2
119         if not (state in self.program.keys()): #Python 3
120             self.program[state] = {}
121
122         tup = (dest_state, char_out, movement)
123         self.program[state][char_in] = tup
124
125     def step(self):
126         """ Steps 1 - 3 """
127         if self.lenStr == 0 and self.state in self.fstates: raise TuringAcceptException
128         if self.state in self.fstates: raise TuringAcceptException
129         if self.state not in self.program.keys(): raise TuringErrorException
130
131         """ Steps 4 and 5 """
132         head = self.tape.read()
133         if head not in self.program[self.state].keys(): raise TuringErrorException
134
135         """ Steps 6 and 7 """
136         # execute transition
137         (dest_state, char_out, movement) = self.program[self.state][head]
138         self.state = dest_state
139         try:
140             """ Step 8 """
141             self.tape.move(head, char_out, movement)
142         except MachineTapeException, s: #Python 2
143         except MachineTapeException as s: #Python 3
144             print s #Python 2
145             print(s) #Python 3
146
147     def execute(self):

```

Imagen 7

Código General Parte 4

```

137         (dest_state, char_out, movement) = self.program[self.state][head]
138         self.state = dest_state
139         try:
140             """ Step 8 """
141             self.tape.move(head, char_out, movement)
142         except MachineTapeException, s: #Python 2
143         except MachineTapeException as s: #Python 3
144             print s #Python 2
145             print(s) #Python 3
146
147     def execute(self):
148         """ The TM will keep stepping forever until the TM accepts or rejects.
149             This does allow for looping TM's """
150         try:
151             while 1:
152                 m.tape.show()
153                 m.step()
154         except (TuringErrorException, TuringAcceptException), s: #Python 2
155         except (TuringErrorException, TuringAcceptException) as s: #Python 3
156             print s #Python 2
157             print(s) #Python 3
158
159 if __name__ == "__main__":
160     # machine to convert a string of A's and B's to
161     # all A's and accept
162     m = TuringMachine("aa", [3])
163
164     m.addTransition(0, 'a', 1, 'a', 'R')
165     m.addTransition(1, 'a', 0, 'a', 'R')
166     m.addTransition(0, 'b', 3, 'b', 'L')
167     m.addTransition(1, 'b', 2, 'b', 'R')
168     m.addTransition(2, 'a', 2, 'a', 'R')
169     m.addTransition(2, 'b', 2, 'b', 'R')
170     m.addTransition(2, 'b', 2, 'b', 'R')
171
172     # run the TM
173     m.execute()

```



Esta parte del código se modifica para los diferentes ejercicios

Imagen 8

código general parte 5

Entonces el ejercicio 1 queda de la siguiente manera:

```
162 m = TuringMachine("000000", [1]) #111111
163
164 m.addTransition(0, '0', 0, '1', 'R')
165 m.addTransition(0, '1', 0, '0', 'R')
166 m.addTransition(0, '_', 1, '_', 'L')
167
```

Imagen 9

Ejercicio 1 – Prueba 1

```
ubuntu@ubuntu:~/Documentos/p8/ejercicio1$ python3 TuringE1.py
000000
^
100000
^
110000
^
111000
^
111100
^
111110
^
111111_
^
111111_
^
Accept
ubuntu@ubuntu:~/Documentos/p8/ejercicio1$
```

Imagen 10

Compilación del Ejercicio 1 – Prueba 1

Como se observa en la Imagen 9, la máquina de turing nos cambiara la cadena de 0's por una cadena de 1's , esto se puede comprobar por la Imagen 10.

```
162 m = TuringMachine("1010101", [1]) #0101010
163
164 m.addTransition(0, '0', 0, '1', 'R')
165 m.addTransition(0, '1', 0, '0', 'R')
166 m.addTransition(0, '_', 1, '_', 'L')
167
```

Imagen 11

Ejercicio 1 – Prueba 2

```
ubuntu@ubuntu:~/Documentos/p8/ejercicio1$ python3 TuringE1.py
1010101
^
0010101
^
0110101
^
0100101
^
0101101
^
0101001
^
0101011
^
0101010_
^
0101010_
^
Accept
ubuntu@ubuntu:~/Documentos/p8/ejercicio1$
```

Imagen 12

Compilación del Ejercicio 1 – Prueba 2

Como se observa en la Imagen 11, la máquina de turing nos cambiara la cadena de “1010101” por “0101010”, esto se puede comprobar por la Imagen 12.

2) Para si se le presenta una cadena de $\{a^{2n}\}$

```
162 m = TuringMachine("aaaa", [3])
163
164 m.addTransition(0, 'a', 1, 'a', 'R')
165 m.addTransition(1, 'a', 0, 'a', 'R')
166 m.addTransition(0, '_', 3, '_', 'L')
167 m.addTransition(1, 'b', 2, 'b', 'R')
168 m.addTransition(2, 'a', 2, 'a', 'R')
169 m.addTransition(2, '_', 2, '_', 'R')
170 m.addTransition(2, 'b', 2, 'b', 'R')
171
```

Imagen 13

Ejercicio 2 – Prueba 1

```
ubuntu@ubuntu:~/Documentos/p8$ python3 TuringE2.py
aaaa
^
aaaa
^
aaaa
^
aaaa
^
aaaa_
^
aaaa_
^
Accept
ubuntu@ubuntu:~/Documentos/p8$
```

Imagen 14

compilación del ejercicio 2 – Prueba 1

La cadena “aaaa” que se muestra en la Imagen 13 se encuentra dentro del lenguaje que nos pide el ejercicio, por lo tanto la máquina de turing para y la acepta (Imagen 14).

```
162 m = TuringMachine("aaa", [3])
163
164 m.addTransition(0, 'a', 1, 'a', 'R')
165 m.addTransition(1, 'a', 0, 'a', 'R')
166 m.addTransition(0, '_', 3, '_', 'L')
167 m.addTransition(1, 'b', 2, 'b', 'R')
168 m.addTransition(2, 'a', 2, 'a', 'R')
169 m.addTransition(2, '_', 2, '_', 'R')
170 m.addTransition(2, 'b', 2, 'b', 'R')
171
172 # sup the TM
```

Imagen 15

Ejercicio 2 – Prueba 2

```
ubuntu@ubuntu:~/Documentos/p8$ python3 TuringE2.py
aaa
^
aaa
^
aaa
^
aaa_
^
Crash
ubuntu@ubuntu:~/Documentos/p8$
```

Imagen 16

compilación del ejercicio 2 – Prueba 2

La cadena “aaa” que se muestra en la Imagen 15 no se encuentra dentro del lenguaje que nos pide el ejercicio, por lo tanto la máquina de turing no la acepta (Imagen 16).

3) Lee una cadena de $\{w \in \{0,1\}^* \mid |w| = 2\}$ deja un espacio en blanco y escribe el resultado de sumar los dígitos leídos.

```
162 m = TuringMachine("10", [7,8,9,10])
163
164 #-----
165 #si la cadena inicia con "0"
166 #m.addTransition(0, '0', 1, '0', 'R')
167 #m.addTransition(1, '0', 3, '0', 'R')
168 #m.addTransition(3, '_', 7, '_00', 'R')
169
170 #-----
171 #Si la cadena empieza con "1"
172 m.addTransition(0, '1', 2, '1', 'R')
173 m.addTransition(2, '0', 5, '0', 'R')
174 m.addTransition(5, '_', 9, '_01', 'R')
175 m.addTransition(2, '1', 6, '1', 'R')
176 m.addTransition(6, '_', 10, '_10', 'R')
177
178 #-----
179 m.addTransition(0, '_', 3, '_', 'L')
180
```

Imagen 17

Ejercicio 3 – Prueba 1

```
ubuntu@ubuntu:~/Documentos/p8/ejercicio3$ python3 TuringE3.py
10
^
10
^
10_
^
10_01_
^
Accept
ubuntu@ubuntu:~/Documentos/p8/ejercicio3$
```

Imagen 18

compilación del ejercicio 3 – Prueba 1

```

162 m = TuringMachine("00", [7,8,9,10])
163
164 #-----
165 #Si la cadena inicia con "0"
166 m.addTransition(0,'0',1,'0','R')
167 m.addTransition(1,'0',3,'0','R')
168 m.addTransition(3,'_',7,'_00','R')
169
170 #-----
171 #Si la cadena empieza con "1"
172 m.addTransition(0,'1',2,'1','R')
173 m.addTransition(2,'0',5,'0','R')
174 m.addTransition(5,'_',9,'_01','R')
175 m.addTransition(2,'1',6,'1','R')
176 m.addTransition(6,'_',10,'_10','R')
177
178 #-----
179 m.addTransition(0,'_',3,'_', 'L')
180
181

```

Imagen 19

Ejercicio 3 – Prueba 2

```

ubuntu@ubuntu:~/Documentos/p8/ejercicio3$ python3 TuringE3.py
00
^
00
^
00_
^
00_00_
^
Accept
ubuntu@ubuntu:~/Documentos/p8/ejercicio3$

```

Imagen 20

compilación del ejercicio 3 – Prueba 2

4) Para cuando se le presenta una cadena de $\{aba^*b\}$

```

162 m = TuringMachine("abaaaab", [3])
163
164 m.addTransition(0,'a',1,'a','R')
165 m.addTransition(1,'b',2,'b','R')
166 m.addTransition(2,'a',2,'a','R')
167 m.addTransition(2,'b',3,'b','R')
168 m.addTransition(3,'a',4,'x','R')
169 m.addTransition(3,'b',4,'x','R')
170 m.addTransition(4,'b',4,'x','R')
171 m.addTransition(4,'a',4,'x','R')
172

```

Imagen 21

Ejercicio 4 – Prueba 1

```

ubuntu@ubuntu:~/Documentos/p8/ejercicio4$ python3 TuringE4.py
abaaaab
^
abaaaab
^
abaaaab
^
abaaaab
^
abaaaab
^
abaaaab
^
abaaaab
^
Accept
ubuntu@ubuntu:~/Documentos/p8/ejercicio4$

```

Imagen 22

compilación del ejercicio 4 – Prueba 1

```

162 m = TuringMachine("baab", [3])
163
164 m.addTransition(0,'a',1,'a','R')
165 m.addTransition(1,'b',2,'b','R')
166 m.addTransition(2,'a',2,'a','R')
167 m.addTransition(2,'b',3,'b','R')
168 m.addTransition(3,'a',4,'x','R')
169 m.addTransition(3,'b',4,'x','R')
170 m.addTransition(4,'b',4,'x','R')
171 m.addTransition(4,'a',4,'x','R')
172
173
174
175

```

Imagen 23

Ejercicio 4 – Prueba 2

```

ubuntu@ubuntu:~/Documentos/p8/ejercicio4$ python3 TuringE4.py
baab
^
Crash
ubuntu@ubuntu:~/Documentos/p8/ejercicio4$

```

Imagen 24

compilación del ejercicio 4 – Prueba 2

Conclusiones

Con esta práctica aprendí la importancia de la máquina de turing y que es una maquina muy poderosa, y hasta la fecha no se ha descubierto otra máquina más poderosa.

Los programas fueron algo complicados para mí porque me costó un poco de trabajo entender el funcionamiento de dicha máquina, pero al final salieron todos los ejercicios.

Referencias

- [1] D. G. P. Luis Migel Pardo Vasallo, «Teoria de Automatas Finitos,» [En línea]. Available: https://ocw.unican.es/pluginfile.php/1516/course/section/1946/2-1_Introduccion.pdf.
- [2] J. E. Hopcroft, R. Motwani y J. D. Ullman, Introducción a la teoría de autómatas, lenguajes y computación, Madrid: PEARSON EDUCACIÓN S.A, 2007.
- [3] D. Kelley, Teoría de autómatas y lenguajes formales, Madrid: PEARSON EDUCACIÓN, S. A, 1995.