INSTITUTO TECNOLÓGICO DE AERONÁUTICA - ITA

4º PROJETO DE CTC - 34

Projeto 4: Projeto de Máquina de Turing

Geradora da Sequência de Fibonacci

Aluno

Felipe Tuyama de Faria Barbosa - ftuyama@gmail.com

PROFESSOR

Carlos Henrique Q. Forster forster@ita.br

São José dos Campos, 01 de Dezembro de 2015

1 INTRODUÇÃO

O projeto consiste no desenvolvimento de uma Máquina de Turing geradora da sequência de Fibonacci, indefinidamente. Logo em seguida, é implementado também um simulador da Máquina de Turing em Python, a fim de verificar o funcionamento do projeto passo a passo, para um dado número de iterações.

2 METODOLOGIA

O desenvolvimento do projeto consiste nas seguintes etapas:

1. Criação do algoritmo baixo nível em pseudocódigo.
2. Projeto do Grafo de Transição de Estados da Máquina de Turing.
3. Simulação do GTE projetado no software JFLAP 7.0
4. Implementação da da Máquina de Turing na linguagem Python 2.7
5. Verificação dos resultados do projeto.

3 DESENVOLVIMENTO

3.1 Algoritmo de Fibonacci

O algoritmo em pseudocódigo para gerar a sequência de Fibonacci a partir de uma Máquina de Turing pode ser descrito da seguinte forma:

```
:Loop para determinar próximo termo: f(n) = 0
> Ir para o Fim da cadeia e inserir '1' delimitador
        :Loop para somar primeiro termo: f(n) += f(n-1)
        > Para cada '0' do elemento (n-1):
                - Substitui-lo por 'X'
                - Adicionar '0' ao elemento (n)
        :Loop para somar segundo termo: f(n) += f(n-2)
        > Para cada '0' do elemento (n-1):
                - Substitui-lo por 'X'
                - Adicionar '0' ao elemento (n)

> Restaurar, revertendo todo 'X' em '0'
> Temos assim o enésimo elemento: f(n) = f(n-1) + f(n-2)
```

3.2 Grafo de Transição de Estados

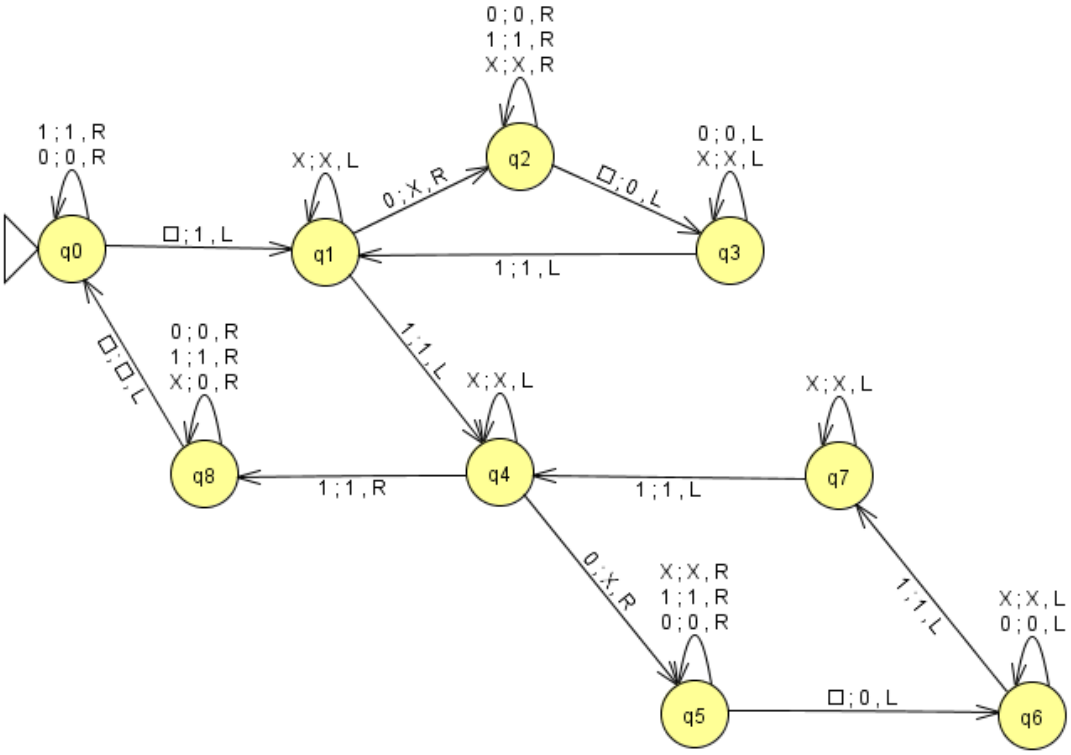


Figura 1 – Grafo de Transição de Estados da Máquina de Turing Geradora da Sequência de Fibonacci (feito em JFLAP 7.0).

A função de cada estado da máquina para gerar enésimo elemento da sequência pode ser explicada brevemente da seguinte forma:

Grande Ciclo: Determinando enésimo elemento de Fibonacci.

0. Inicia-se um novo ciclo. Vai para o fim da cadeia. INC.

Ciclo #1: Somando elemento (n-1).

1. Procura ‘0’ mais à direita do elemento (n-1).
   - Se encontrá-lo, substitui por ‘X’. INC.
   - Se não encontrar, vai para q4.
2. Vai para o fim da cadeia. Substitui ‘B’ por ‘0’. INC.
3. Volta até o elemento (n-1), repetindo ciclo 1-2-3.

Ciclo #2: Somando elemento (n-2).

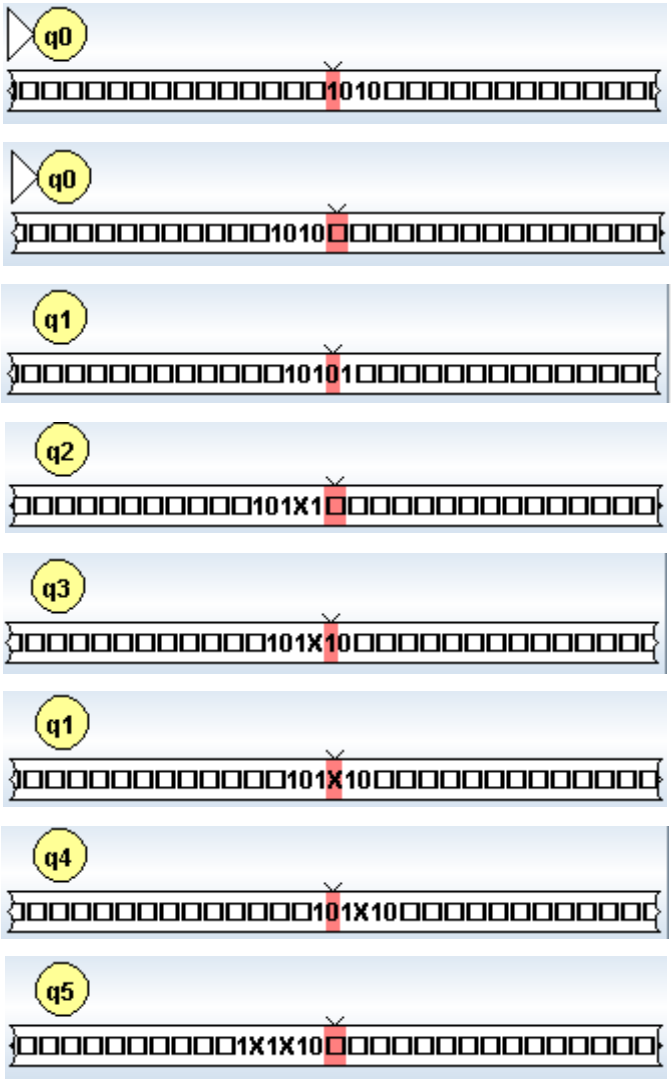
4. Procura ‘0’ mais à direita do elemento (n-2).
   - Se encontrá-lo, substitui por ‘X’. INC.
   - Se não encontrar, vai para q8.
5. Vai para o fim da cadeia. Substitui ‘B’ por ‘0’. INC.

6. Volta até o elemento (n-1).
7. Volta até o elemento (n-2), repetindo ciclo 4-5-6-7.
8. Substitui todos os ‘X’ dos elementos (n-2) e (n-1) por ‘0’.
   Vai para o último elemento da cadeia. Volta para q0.

3.3 Simulação no FLAP 7.0

Como o GTE foi construído graficamente no software FLAP 7.0, que também possui recursos de simulação de Autômatos, foi empregada essa ferramenta para verificar previamente o funcionamento da Máquina de Turing desenvolvida.

Porém, por questões de espaço, é inviável demonstrar neste relatório a simulação (gerando uma sequência infinita) passo a passo do JFLAP, que emprega recursos visuais em sua simulação. Mas podemos conferir na sequência de imagens abaixo o funcionamento da Máquina de Turing projetada:



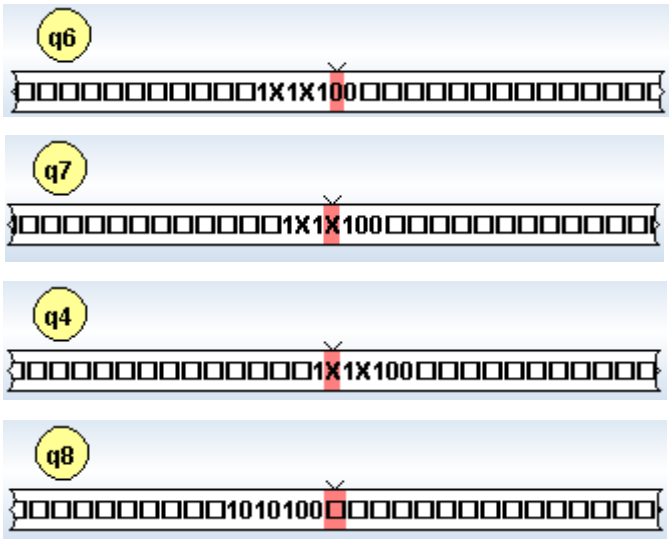


Figura 2 – Simulação (usando JFLAP 7.0) de um ciclo desempenhado pela Máquina de Turing, para a geração do elemento 2.

Na simulação do JFLAP abaixo (Figura 3), temos as iterações da Máquina de Turing até o elemento 7. Pode-se notar a sequência de Fibonacci corretamente produzida: 1, 1, 2, 3, 5, 7.

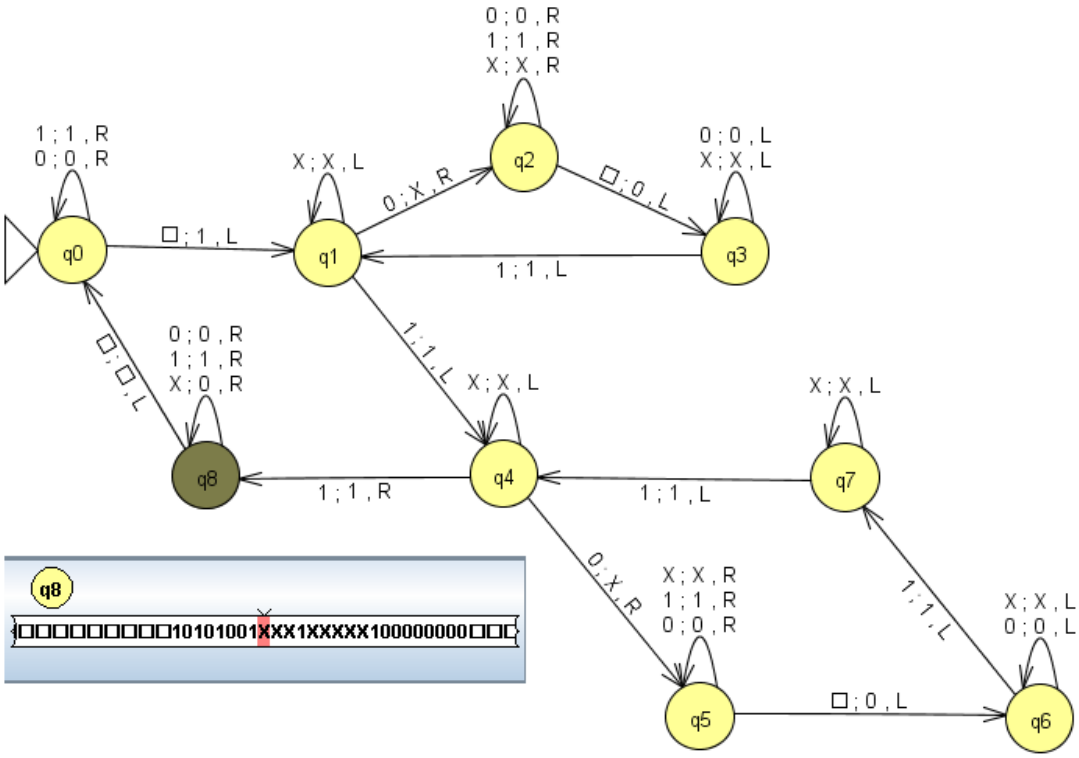


Figura 3 – Simulação da Máquina de Turing, gerando o elemento 7.

3.4 Simulador de Máquina de Turing

O algoritmo, assim como a simulação de Autômatos Finitos Determinísticos Bidimensionais (2AFD), possui um conjunto de estados Q, uma cadeia formada por símbolos de um alfabeto Σ, um conjunto de estados de aceitação F e um estado inicial q_0. Temos ainda uma função de transição δ que especifica o próximo estado e o sentido que o cabeçote de leitura deve ser mover.

A principal mudança ocorre na Função de Transição da Máquina de Turing, que permite a escrita de símbolos do alfabeto Γ sobre a cadeia (fita de leitura):

$$\delta: Q\ x\ \Gamma \rightarrow Q\ x\ \Gamma\ x\ \{L, R\}$$

Sua implementação na linguagem Python ocorreu com simplicidade. Primeiramente é lido um arquivo .txt com a especificação da Máquina de Turing (cujo formato é explicado no tópico 3.4.1). Logo em seguida ocorre a simulação da MT para um dado número finito de iterações, definida no próprio código.

A iteração consiste basicamente em imprimir a configuração instantânea da MT no console e depois aplicar a função de transição. A tabela de Fluxo de Estados é representada uma matriz (na verdade uma lista de estados Q contendo cada qual sua lista de transições possíveis, minimizando consumo de memória). Essa 'matriz' é percorrida.

Se nenhuma transição é encontrada para o símbolo lido na fita, temos término da computação com rejeição da cadeia. Se alguma transição é possível, atualiza-se a configuração instantânea da MT (atualiza o estado atual, escreve algum símbolo sobre o símbolo atual na fita e então move o cabeçote de leitura no sentido especificado). Se o cabeçote for além do tamanho da cadeia, um novo caractere 'B' é acrescentado a ela, garantindo a condição de que a fita da MT é semi-infinita.

3.4.1 Entrada

A entrada do programa é um arquivo .txt contendo as especificações da MT e a cadeia a ser simulada. As informações são divididas em blocos entre "#", um elemento da lista por linha, na seguinte ordem:

$$Alfabeto\ (\Sigma \cup \Gamma) - Número\ Estados\ Q - Transições\ \delta - Cadeia\ simulada\ u$$

Para mais detalhes da especificação, ver a seção de testes.

3.4.2 Saída

A saída do arquivo é escrita no próprio terminal de execução, exibindo a configuração instantânea da MT a cada iteração realizada. O número de iterações a serem simuladas é determinada no próprio código do programa.

Exemplo de saída:

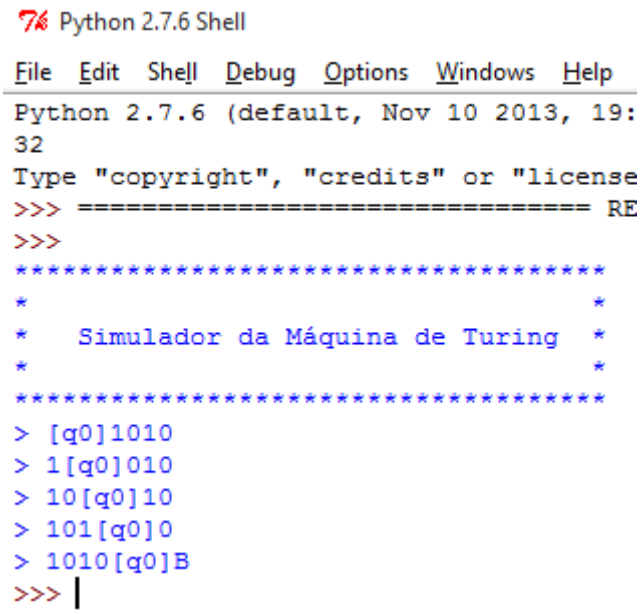


Figura 4 – Exemplo de saída do programa, para 5 iterações.

4 REFLEXÃO

Como essa MT pode ser usada para verificar se um dado número pertence à sequência de Fibonacci? Uma solução é projetar uma outra MT H que empregue a nossa MT geradora da sequência de Fibonacci, sendo que H será reconhecedora da linguagem dos elementos de Fibonacci.

Seja o número N a entrada da MT H, e f(n) o enésimo elemento gerado pela MT da sequência de Fibonacci. Assim o funcionamento de H pode ser descrito da seguinte forma:

> Enquanto N > f(n)
- n++
- MT gera f(n)
$> Se\ N = f(n) \rightarrow N \in Elementos\ de\ Fibonacci$
$> Se\ N < f(n) \rightarrow N \notin Elementos\ de\ Fibonacci$

5 CONCLUSÃO

O desenvolvimento desse projeto permitiu o aprendizado de como projetar uma Máquina de Turing para executar um dado algoritmo. Sua programação, embora em baixo nível, obedece a uma certa lógica que permitiu a escrita de um algoritmo em pseudocódigo para representa-la.

A implementação do simulador da Máquina de Turing foi bem simplificada, devido ao prévio desenvolvimento de um Autômato Finito Determinístico Bidirecional, cujo funcionamento é similar. A diferença é a MT contempla o reconhecimento das Linguagens Irrestritas, geradas por Gramáticas Irrestritas, a mais generalizada e poderosa conhecida até o momento[1].

6 REFERÊNCIAS BIBLIOGRÁFICAS

[1] RIBEIRO, C. H; FORSTER, C. H. Q. Slide 8: CTC-34 Automata e Linguagens Formais, Novembro de 2015. Notas de Aula.

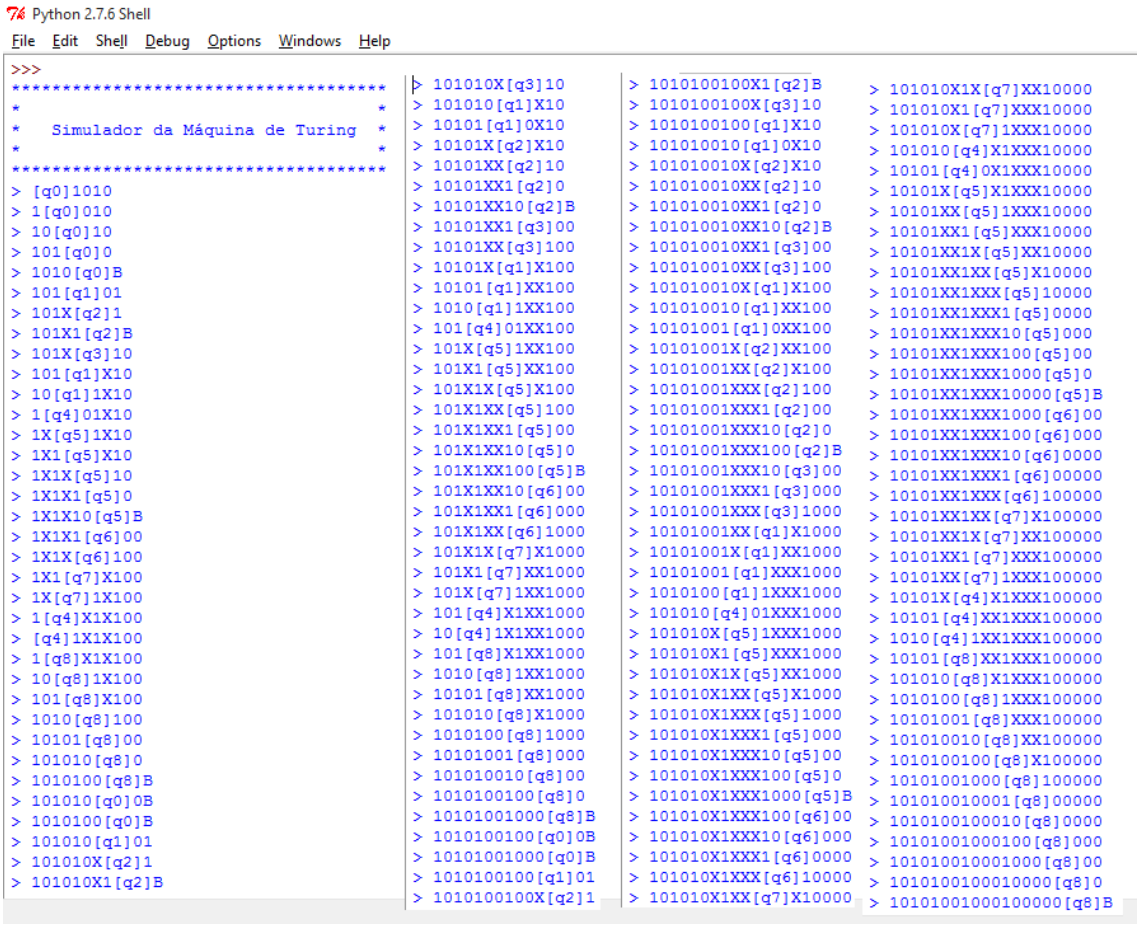
7 TESTES

MT.txt

```
**********************************                                *
*       Máquina de Turing       *
*                               *
**********************************
      Alfabeto
************************
#
0
1
X
B
#
   Número de Estados
************************
#
9
#
 Função de Transição
************************
#
q0 q0 0,0,R
q0 q0 1,1,R
q0 q1 B,1,L
q1 q1 X,X,L
q1 q2 0,X,R
q2 q2 0,0,R
q2 q2 1,1,R
q2 q2 X,X,R
q2 q3 B,0,L
q3 q3 0,0,L
q3 q3 X,X,L
q3 q1 1,1,L
q1 q4 1,1,L
q4 q4 X,X,L
q4 q5 0,X,R
q5 q5 0,0,R
q5 q5 X,X,R
q5 q5 1,1,R
q5 q6 B,0,L
q6 q6 0,0,L
q6 q6 X,X,L
q6 q7 1,1,L
q7 q7 X,X,L
q7 q4 1,1,L
q4 q8 1,1,R
q8 q8 0,0,R
q8 q8 1,1,R
q8 q8 X,0,R
q8 q0 B,B,L
#
   Cadeia simulada
************************
#
1010
#
```

Saída obtida:

Resultado da simulação passo a passo da Máquina de Turing geradora da Sequência de Fibonacci para 158 iterações, mostrando no console de Python o sucesso da implementação do Simulador e também do projeto da nossa MT:



8 Código desenvolvido em Python

```
# -*- coding: cp1252 -*-
#
#       Simulador da Máquina de Turing
#
#  Autor: Felipe Tuyama
import copy
import sys

# Nó de Transição
class Transition(object):
    def __init__(self, st2, RD, WR, DIR):
        self.st = st2
        self.RD = RD
        self.WR = WR
        if DIR == 'L':
            self.DIR = -1
        else: self.DIR = 1

# Leitura da Transição da MT
def nextData(char):
    global reader
    begin = 0
    while begin < len(reader) and reader[begin] != char:
        begin += 1
    if begin == 0: info = ""
    else: info = reader[0:begin]
    reader = reader[begin+1:len(reader)]
    return info

# Leitura de uma linha não vazia sem \n
def read():
    lido = arquivo.readline().rstrip('\n')
    while lido == "":
        lido = arquivo.readline().rstrip('\n')
    return lido

# Leitura do Arquivo de Entrada especificando MT
def readMT():
    global reader
    for i in range(0, 4):
        reader = " "
        while reader[0] != '#':
            reader = read()
        reader = read()
        while reader[0] != '#':
            if i == 0:   alphabet.append(reader)
            elif i == 1:
                for j in range(0, int(reader)):
                    TFE.append([])
            elif i == 2:
                st1 = int(nextData(' ')[1:])
                st2 = int(nextData(' ')[1:])
                RD = nextData(',')
                WR = nextData(',')
                DIR = nextData('')
                TFE[st1].append(Transition(st2, RD, WR, DIR))
            elif i == 3: alphabet.append(reader)
            reader = read()
```

```
# Simulação da Máquina de Turing
def simulate(iter):
    global string
    stt = q = 0
    while iter > 0:
        print "> "+string[:stt]+"[q"+str(q)+"]"+string[stt:]
        for j in range(0, len(TFE[q])+1):
            if j == len(TFE[q])+1:
                quit()
            if TFE[q][j].RD == string[stt]:
                break
        Trans = TFE[q][j]
        string = string[0:stt]+Trans.WR+string[stt+1:]
        q = Trans.st
        stt = stt + Trans.DIR
        if stt == len(string):
            string = string + "B"
        iter = iter - 1

# Dados da Máquina de Turing
alphabet = []    # Alfabeto de entradas.
TFE = []         # Tabela  de Fluxo de Estados.
string = ""      # Cadeia a ser simulada.

# Rotina main()
print "*************************************"
print "*                                   *"
print "*   Simulador da Máquina de Turing  *"
print "*                                   *"
print "*************************************"
arquivo = open('MT.txt', 'r')
readMT()
string = alphabet.pop()
result = simulate(760)
```