

Práctica AFD - Justin Martinez Navarro

Descripción

Un autómata finito determinista (AFD) es una 5-tupla $(Q, \Sigma, \delta, q_0, F)$, donde:

1. Q es un conjunto finito llamado **estados**,
2. Σ es un conjunto finito llamado **alfabeto**,
3. $\delta : Q \times \Sigma \rightarrow Q$ es la **función de transición**,
4. $q_0 \in Q$ es el **estado inicial**, y
5. $F \subseteq Q$ es el **conjunto de estados de aceptación**.

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un AFD y sea $w = w_1 w_2 \dots w_n$ una cadena de símbolos donde $w_i \in \Sigma$. Entonces, M **acepta** w si existe una secuencia de estados r_0, r_1, \dots, r_n en Q con tres condiciones:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, para $i = 0, \dots, n-1$, y
3. $r_n \in F$.

La condición 1 establece que la máquina comienza en el estado inicial. La condición 2 establece que la máquina cambia desde un estado hacia otro estado de acuerdo a la función de transición. La condición 3 establece que la máquina acepta la cadena de entrada si termina en un estado de aceptación. Entonces, M **reconoce el lenguaje** A si $A = \{w | M \text{ acepta } w\}$.

Escribir un programa para determinar si un conjunto de cadenas W pertenecen o no pertenecen al lenguaje A reconocido por un AFD M .

Entrada

La primer línea de entrada contiene seis enteros N, S, D, q_0, T y C , ($1 \leq N, S, C \leq 100, 1 \leq D \leq 10^4, 1 \leq q_0 \leq N, 0 \leq T \leq N$), donde $N = |Q|$, $S = |\Sigma|$, $D = N \times S$ es el número de transiciones en el autómata, q_0 es el estado inicial, $T = |F|$, y C es la cantidad de cadenas a verificar si son aceptadas o no por el autómata M . Cada estado $q \in Q$ se identifica de manera implícita por un número entero con valor entre 1 y N .

La segunda línea contiene el alfabeto Σ , representado por una secuencia de S símbolos s_i separados por espacios, tal que cada símbolo s_i puede ser una letra, un dígito o cualquier carácter del código ASCII excepto por el espacio.

La tercer línea contiene el conjunto de estados de aceptación F , representado por una secuencia de T enteros t_i ($1 \leq t_i \leq N$) separados por espacios.

Las siguientes D líneas especifican las transiciones del autómata. Cada línea define una transición $\delta(I, X) = J$ por medio de un entero I , un carácter X y un entero J ($I, J \in Q$ y $X \in \Sigma$) separados por espacios, representando la transición desde el estado I hacia el estado J cuando el símbolo de la entrada sea X .

Finalmente, cada una de las siguientes C líneas contienen una cadena W , compuesta por símbolos que pertenecen al alfabeto Σ . La longitud de la cadena W está entre 0 y 100 caracteres.

Salida

Para cada una de las C cadenas W se deberá imprimir el mensaje **ACEPTADA** si el autómata M acepta la cadena W , ó **RECHAZADA** en caso contrario.

Código:

class AFD:

```
def __init__(self, num_estados, alfabeto, transiciones, estado_inicial, estados_aceptacion):
    self.num_estados = num_estados
    self.alfabeto = alfabeto
    self.transiciones = transiciones
    self.estado_inicial = estado_inicial
    self.estados_aceptacion = estados_aceptacion
```

```
def procesar_cadena(self, cadena):
    estado_actual = self.estado_inicial
    for simbolo in cadena:
        if (estado_actual, simbolo) in self.transiciones:
            estado_actual = self.transiciones[(estado_actual, simbolo)]
        else:
            return "RECHAZADA"
    return "ACEPTADA" if estado_actual in self.estados_aceptacion else "RECHAZADA"
```

N = 3

S = 2

D = 6

q0 = 1

T = 1

C = 3

alfabeto = ['0', '1']

estados_aceptacion = {2}

transiciones = {

(1, '0'): 1,

(1, '1'): 2,

(2, '0'): 3,

(2, '1'): 2,

(3, '0'): 3,

(3, '1'): 2,

}

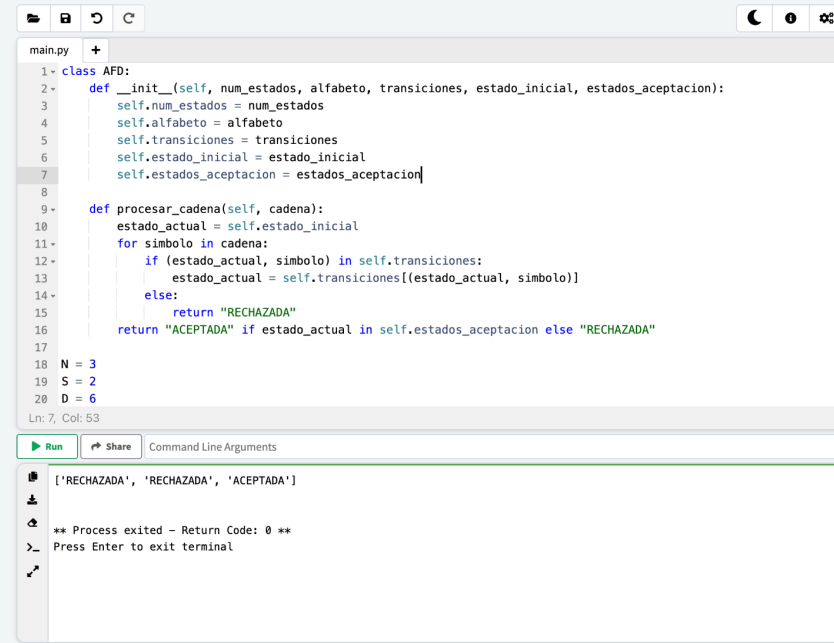
cadenas = ['100', '0', '11']

afd = AFD(N, alfabeto, transiciones, q0, estados_aceptacion)

resultados = [afd.procesar_cadena(cadena) for cadena in cadenas]

print(resultados)

Capturas de pantalla:



The screenshot shows a code editor with a file named `main.py`. The code defines a class `AFD` with an `__init__` method and a `procesar_cadena` method. The `__init__` method initializes attributes: `num_estados`, `alfabeto`, `transiciones`, `estado_inicial`, and `estados_aceptacion`. The `procesar_cadena` method processes a string `cadena` by iterating over its characters and checking the transition table. It returns `"RECHAZADA"` if the current state is not in the transition table or if the final state is not in the set of accepting states, and `"ACEPTADA"` otherwise. Below the code, the execution output is shown, displaying the results of processing the strings `"N"`, `"S"`, and `"D"`.

```
1- class AFD:
2-     def __init__(self, num_estados, alfabeto, transiciones, estado_inicial, estados_aceptacion):
3-         self.num_estados = num_estados
4-         self.alfabeto = alfabeto
5-         self.transiciones = transiciones
6-         self.estado_inicial = estado_inicial
7-         self.estados_aceptacion = estados_aceptacion
8-
9-     def procesar_cadena(self, cadena):
10-         estado_actual = self.estado_inicial
11-         for simbolo in cadena:
12-             if (estado_actual, simbolo) in self.transiciones:
13-                 estado_actual = self.transiciones[(estado_actual, simbolo)]
14-             else:
15-                 return "RECHAZADA"
16-         return "ACEPTADA" if estado_actual in self.estados_aceptacion else "RECHAZADA"
17-
18- N = 3
19- S = 2
20- D = 6
```

Ln: 7, Col: 53

Run Share Command Line Arguments

```
[ 'RECHAZADA', 'RECHAZADA', 'ACEPTADA' ]
```

** Process exited - Return Code: 0 **

>_ Press Enter to exit terminal