Instituto Politécnico Nacional

Escuela Superior de Cómputo

Programa 5: Expresión Regular

Autor: Colín Ramiro Joel

Materia: Teoría de la Computación

Grupo: 4CM2

Profesor: Juarez Martínez Genaro

Fecha de entrega: 29 de Diciembre 2021

Introducción

En las ciencias de la computación, más específicamente en la teoría de lenguajes formales, una expresión regular también conocida como expresión racional, es una secuencia de caracteres que conforma un patrón de búsqueda. Su utilización recae principalmente en la búsqueda de patrones de cadenas de caracteres u operaciones de sustituciones. Una definición un poco más simplificada es que son patrones utilizados para encontrar una determinada combinación de caracteres dentro de una cadena de texto.Estas expresiones proporcionan una manera muy flexible de buscar o reconocer cadenas de texto.

La mayoría de las formalizaciones proporcionan los siguientes constructores:

1. Una expresión regular es una forma de representar los lenguajes regulares (finitos o infinitos)
2. Se construye utilizando caracteres del alfabeto sobre el cual se define el lenguaje.

Este patrón antes mencionado puede estar formado por un conjunto de caracteres ya sean letras, números o signos acompañado de metacaracteres que representan a otros caracteres o permiten una búsqueda contextual. Los metacaracteres no se representan a ellos mismos, sino que son interpretados de una manera especial. Por ejemplo, a través de metacaracteres podemos definir diferentes condiciones tales como agrupaciones, alternativas, comodines, multiplicadores, etcétera. Algunos ejemplos de metacaracteres pueden ser: * $ +, entre algunos otros.

Instrucciones

Realizar un programa que genere 10 cadenas a partir de la siguiente expresión regular: **(0+10)*(e+0+1)**

1. El programa deberá ser automático.
2. Generar 10 cadenas de manera aleatoria, accediendo a cada parte de la ecuación de manera aleatoria.
3. En el caso de la cerradura de Kleene, ésta podrá llegar como máximo hasta el valor 1,000.
4. Las cadenas generadas deben de pertenecer al lenguaje y se deberán almacenar en un archivo de texto.
5. En otro archivo imprimir la selección de las operaciones de la expresión.
6. Si se continua ejecutando el programa, las siguientes cadenas deberán anexarse al mismo archivo.
7. En el reporte debe de estar también el código de la implementación en latex, no en imágenes.

Desarrollo

En este 5to reporte se desarrolló un programa el cual genera 10 cadenas aleatoriamente de un tamaño máximo de 1000, a partir de la expresión regular previamente definida en la sección **Instruciones**. Todas ellas pertenecen al lenguaje de la expresión regular, el cual se trata de todas las cadenas de 0s y 1s sin que existan dos 1s consecutivos. Estas cadenas se imprimen en el archivo de texto **CadenasP5.txt**. Y a su vez la selección de las operaciones de dichas cadenas se imprimen en el archivo de textp **Operaciones**. Cabe aclarar genera las cadenas automáticamente sin embargo, el usuario puede decidir si el programa continua generando cadenas un **n** número de veces o si bien el programa termine ahi. Por lo tanto en los archivos de texto se encontraran todas las cadenas generadas globalmente hablando, asi como todas las selecciones de operaciones.

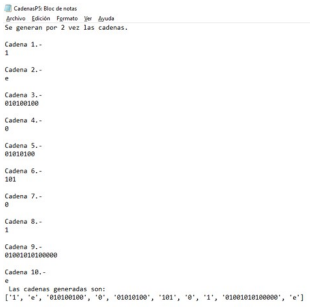
Capturas del Funcionamiento

En esta sección así como en los otros programas se encuentran las capturas del funcionamiento del programa, asi como de los archivos generados por el mismo.

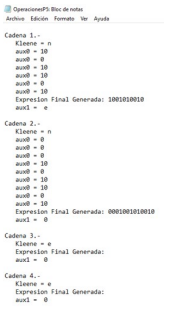
```
C:\Users\joelc\OneDrive\Documentos\Joel\4to\TC\Programas\Programa5-Expresión Regular>python Programa5_TC.py

Las cadenas generadas se encuentran en el archivo: CadenasP5.txt


Las Operaciones se encuentran en el archivo: OperacionesP5.txt

   ****Digite la opción****

        1.- Continuar con el programa
        2.- Salir
        1

Las cadenas generadas se encuentran en el archivo: CadenasP5.txt


Las Operaciones se encuentran en el archivo: OperacionesP5.txt

   ****Digite la opción****

        1.- Continuar con el programa
        2.- Salir
```

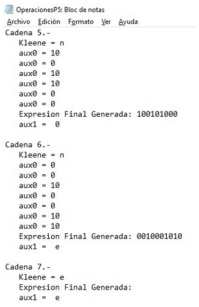
En la imagen superior, se puede observar que se generan las primeras 10 cadenas, posterior a eso, se le vuelve a preguntar al usuario si desea que continue el programa. Para las capturas que se encuentran debajo, se considero que se generen 20 cadenas es decir que se repita 2 veces.

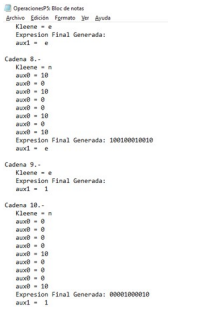
1. **Capturas de las Cadenas**

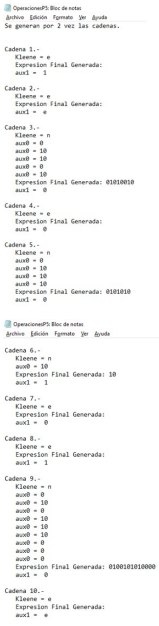


2. Capturas de las Operaciones









OperacionesP5: Bloc de notas
Archivo Edición Formato Ver Ayuda

```
Se generan por 2 vez las cadenas.


Cadena 1.-
   Kleene = e
   Expresion Final Generada:
   aux1 =  1

Cadena 2.-
   Kleene = e
   Expresion Final Generada:
   aux1 =  e

Cadena 3.-
   Kleene = n
   aux0 = 0
   aux0 = 10
   aux0 = 10
   aux0 = 0
   aux0 = 10
   Expresion Final Generada: 01010010
   aux1 =  0

Cadena 4.-
   Kleene = e
   Expresion Final Generada:
   aux1 =  0

Cadena 5.-
   Kleene = n
   aux0 = 0
   aux0 = 10
   aux0 = 10
   aux0 = 10
   Expresion Final Generada: 0101010
   aux1 =  0
```


OperacionesP5: Bloc de notas
Archivo Edición Formato Ver Ayuda

```
Cadena 6.-
   Kleene = n
   aux0 = 10
   Expresion Final Generada: 10
   aux1 =  1

Cadena 7.-
   Kleene = e
   Expresion Final Generada:
   aux1 =  0

Cadena 8.-
   Kleene = e
   Expresion Final Generada:
   aux1 =  1

Cadena 9.-
   Kleene = n
   aux0 = 0
   aux0 = 10
   aux0 = 0
   aux0 = 10
   aux0 = 10
   aux0 = 10
   aux0 = 0
   aux0 = 0
   aux0 = 0
   Expresion Final Generada: 0100101010000
   aux1 =  0

Cadena 10.-
   Kleene = e
   Expresion Final Generada:
   aux1 =  e
```

Código

```
# Programa 5.Expresi n Regular
# Nombre: Col n Ramiro Joel
# Profesor: Juarez Mart nez Genaro
# Grupo: 4CM2
# Materia: Teor a Computacional
import random
from tkinter import*
def generadorCad():
  numCad = 10
  expresionGen = []
  for i in range(0,numCad):
    archivo.write("\nCadena "+ str(i+1) + ".-")
    archivo2.write("\n\nCadena "+ str(i+1) + ".-")
    expresionGen.append ("")
    kleene = random.choice(["e","n"])
    archivo2.write("\n   Kleene = " + kleene)
    if (kleene == "n"):
      n = random.randint(1,1000)
      for j in range (0,n):
        aux0 = random.choice(["0","10"])
        archivo2.write("\n   aux0 = " + aux0)
        expresionGen[i] = expresionGen[i] + aux0
    archivo2.write("\n   Expresion Final Generada: " +
                                  expresionGen[i])
    aux1 = random.choice(["e","0","1"])
    expresionGen[i] = expresionGen [i] + aux1
    archivo2.write(" " + "\n   aux1 =  " + aux1 )
    archivo.write("\n"+ expresionGen[i]+ "\n ")
```

```
    archivo.write("Las cadenas generadas son:\n"+ str(expresionGen)
                                  )
    print("\nLas cadenas generadas se encuentran en el archivo:
                                  CadenasP5.txt\n")
    print("\nLas Operaciones se encuentran en el archivo:
                                  OperacionesP5.txt\n")

opc = 1
salir = 2
archivo = open("CadenasP5.txt","w")
archivo2 = open("OperacionesP5.txt","w")
cont = 1
while opc != salir:
  generadorCad()
  while (opc != salir):
    print("   ****Digite la opci n****")
    opc = int(input('''
    1.- Continuar con el programa
    2.- Salir
    '''))
    if (opc == 1):
      cont = cont + 1
      archivo.write("\n\nSe generan por " + str(cont) + " vez las
                                  cadenas.\n")
      archivo2.write("\n\nSe generan por " + str(cont) + " vez
                                  las cadenas.\n")
      generadorCad()
    elif opc == 2:
      print("Saliendo del Programa. Hasta Luego!!!!")
    else:
      print("Opcion inv lida , Vuelva a intentar")
```

Conclusiones

Al término de la realización y codificación de este programa pude reforzar y más que nada terminar de entender este tema tan amplio como lo son las expresiones regulares. Me apoye de los conocimientos adquiridos en clase, asi como de las láminas del curso de **Stanford**.

Considero que fue un poco complicado el entendimiento total del tema y puedo decir que este fue de los programas más complejos si consideramos el tiempo efectivo tanto de codificación como de entendimiento. No obstante ha sido un programa el cual me apoyó para aprender y reforzar conocimientos del lenguaje de programación **Python.**

Referencias

1. Software Guru. (2014). Expresiones Regulares.. Diciembre 21, 2021, de Software Guru Sitio web: https://sg.com.mx/content/view/545

2. Google. (2019). Acerca de las expresiones regulares (regex). Diciembre 21, 2021, de Google Sitio web: https://support.google.com/analytics/answer/1034324?hl=es

3. IBM Corporation. (2015). Expresiones regulares. Diciembre 21, 2021, de IBM Corporation Sitio web: https://www.ibm.com/docs/es/i/7.3?topic=expressions-regular