FC - UNI PRÁCTICA CALIFICADA 4

TEORÍA DE AUTÓMATAS LENGUAJES Y COMPUTACIÓN
CC321-A

Gramáticas

Las gramáticas utilizadas para este proyecto se denominan gramáticas libres de contexto, en las que cada regla tiene un solo no terminal en el lado izquierdo de la flecha $\rightarrow$.

Considere las siguientes producciones para la gramática G:

< inicio > $\rightarrow$ < historia > .
< historia > $\rightarrow$ < frase > | < frase > y < historia > | < frase > sino < historia >
< frase > $\rightarrow$ < articulo > < sustantivo > < verbo > < articulo > < sustantivo >
< articulo > $\rightarrow$ el | la | al
< sustantivo > $\rightarrow$ gato | niño | perro | niña
< verbo > $\rightarrow$ perseguia | besaba

Así, puede generar la siguiente historia empezando del símbolo no terminal < inicio >

el gato perseguia al niño y el niño besaba al gato.

Enteros PseudoAleatorios

El algoritmo de Park-Miller (llamado así por sus inventores) es una forma sencilla de generar una secuencia de términos enteros pseudoaleatorios. Funciona así. Sea N_0 un número entero llamado semilla. La semilla es el primer término de la sucesión y debe estar entre 1 y 2^{31} - 2, inclusive. A partir de la semilla, los términos posteriores N_1, N_2, ... se producen mediante la siguiente ecuación.

$$N_{k+1} = (7^5 N_k) \,\%\, (2^{31} - 1)$$

Aquí 7^5 es 16807 y 2^{31} es 2147483648. El operador % de Python devuelve el resto después de dividir un número entero por otro.

Siempre obtendrá la misma secuencia de términos de una semilla determinada. Por ejemplo, si comienza con la semilla 101, obtendrá una secuencia pseudoaleatoria cuyos primeros términos son 1697507, 612712738, 678900201, 695061696, 1738368639, 246698238, 1613847356 y 1214050682.

Podría usar esta secuencia para probar si su generador de números aleatorios funciona correctamente.

Los términos de la secuencia pueden ser grandes, pero puede hacerlos más pequeños usando el operador % nuevamente. Si N es un término de la sucesión y M es un número entero mayor que 0, entonces N % M da un número entero entre 0 y M - 1, inclusive. Por ejemplo, si necesita un número entero entre 0 y 9, entonces escribiría N %10.

Escriba una aplicación en Python que genere cadenas aleatorias usando una gramática. Debe implementar las siguientes clases:

La clase Aleatorio
La primera clase debe denominarse Aleatorio y debe implementar el algoritmo **Park-Miller**. El algoritmo de Park-Miller es una forma sencilla de generar una secuencia de términos enteros pseudoaleatorios.

Debe tener los siguientes métodos:

__ init __ (self, semilla)
Inicializa una instancia de la clase Aleatorio para que genere la secuencia de enteros pseudoaleatorios que comienzan con **semilla**. Puede suponer que la **semilla** es un número entero en el rango adecuado para que funcione el algoritmo de Park-Miller.

siguiente (self)
Genera el siguiente entero aleatorio en la secuencia y lo devuelve.

elegir(self, limite)
Llama al método **siguiente** para obtener un número entero aleatorio.

Luego, calcula un nuevo entero entre 0 y límite-1. Devuelve el nuevo entero.

Para mayor eficiencia, su clase Aleatorio no debe calcular números grandes. Calcúlese solo una vez y almacenarlos en variables, o escríbalos como constantes, para que no tenga que calcularlos.

Todos los métodos en la clase Aleatorio deben ser muy cortos, de una a tres líneas cada uno.

La clase Regla
Su instancia debe representar una regla de una gramática. Debe tener el siguiente constructor:

__ init__(self, izquierda, derecha)
Cada instancia de Regla debe tener tres variables. La variable **self.left** debe inicializarse a la cadena izquierda. La variable **self.right** debe inicializarse a la tupla de cadenas derecha. La variable **self.cont** debe inicializarse en el entero 1. la clase Gramática la usa para ayudar a elegir las reglas.

__ repr__ (self)
Devuelve una cadena de la forma 'C L → R1 R2 ... Rn', donde C está dado por **self.cont**, L está dado por **self.left** y R1, R2 ... Rn, son los elementos de **self.right**. Por ejemplo, si se crea una regla llamando al constructor:

Regla ('Lenguaje' , ('Frase', 'y', 'Lenguaje'))
entonces su método __ **repr**__ debe devolver la cadena:

'1 Historia → Frase y Lenguaje'

Es decir, la cadena debe verse como una regla. Cuando se imprime una instancia de la clase **Regla**, Python llama automáticamente al método __ repr__. La variable **self.left** solo la usa __ repr__

La clase Gramatica
Esta clase debe implementar una gramática usando reglas como las descritas anteriormente

__ init__(self, semilla)
Inicializa una instancia de Gramática. Debe crear una instancia del generador de números aleatorios **Aleatorio** que usa **semilla**. También debe definir un diccionario que almacene reglas. El diccionario debe estar inicialmente vacío.

regla(self, izquierda, derecha)
Agrega una nueva regla a la gramática. Aquí, **izquierda** es una cadena. Representa el símbolo no terminal en el lado izquierdo de la regla. Además, **derecha** es una tupla cuyos elementos son cadenas. Representan los terminales y los no terminales que están en el lado derecho de la regla.

Encuentre el valor de **izquierda** en el diccionario. Si no existe dicho valor; deje que el valor de **izquierda** en el diccionario sea una tupla cuyo único elemento es **Regla**(izquierda, derecha). Sin embargo, si existe tal valor, entonces será una tupla. Añadir **Regla** (izquierda, derecha) al final de esa tupla.

generar(self)
Genera una cadena. Si hay una regla con el lado izquierdo **'Inicio'** en el diccionario, llame a generar con la tupla **('Inicio',)** como argumento, y retorne el resultado. Si no existe dicha regla, entonces genere una excepción, porque no puede generar cadenas sin una regla para **'Inicio'**.

generando(self, strings)
Este método, junto con **seleccionar**, realiza la mayor parte del trabajo para esta aplicación. El parámetro strings es una tupla cuyos elementos son cadenas. Las cadenas representan símbolos terminales y no terminales.

Inicialice una variable llamada **resultado** a: " ", la cadena vacía. Esta almacena el resultado que será devuelto por este método. Luego use un bucle para visitar cada cadena en **strings**.

Si la cadena visitada no es una clave en el diccionario, entonces es un símbolo terminal. Agréguelo al final del resultado, seguido de un espacio en blanco ' '.

Si la cadena visitada es una clave en el diccionario, entonces es un símbolo no terminal. Llame a **seleccionar** en la cadena para obtener una tupla de cadenas. Luego llame a **generar** recursivamente en esa tupla de cadenas, para obtener una nueva cadena. Agregue la nueva cadena al final del resultado, sin un espacio en blanco al final.

Continúe de esta manera hasta que se hayan visitado todas las cadenas en **strings**. En ese punto, el resultado será una cadena generada por la gramática. Retornar **resultado**.

seleccionar(self, left)
Este método, junto con **generando**, hace la mayor parte del trabajo para esta aplicación. Elige una regla al azar cuyo lado izquierdo es la cadena **left**.

Esto sucede en varios pasos:

1. Establezca la variable **reglas** para que sea la tupla de todas las reglas con **left** en sus lados izquierdos (del diccionario). Establezca la variable **total** como la suma de todas las variables **cont** en las reglas.

2. Establezca la variable **índice** en un número entero elegido al azar. Debe ser mayor o igual que 0, pero menor que el **total**.

3. Visite cada regla en **reglas**, una a la vez. Reste la variable **cont** de la regla del índice. Continúe de esta manera hasta que la variable índice sea menor o igual a 0. Establezca la variable **elegido** para la regla que se estaba visitando cuando esto ocurrió. (Como resultado, es más probable que se elijan reglas con grandes variables de conteo que reglas con pequeñas variables de conteo).

4. Agregue 1 a las variables **cont** de todas las reglas en la variable **reglas**, excepto las elegidas. (Esto hace que sea más probable que se seleccione una regla distinta a la elegida más adelante, lo que proporciona un rango más amplio de cadenas aleatorias).

Finalmente, devuelve la variable **right** de **elegido**, una tupla de cadenas.