Examen Extraordinario de Programación
Curso 2023

Gramáticas Evolutivas

Nota: *Recuerde descomprimir la prueba antes de comenzar a trabajar. Si programa dentro del comprimido es posible que pierda su solución sin posibilidad de reclamar.*

Se desea implementar un mecanismo para *derivar árboles a partir de una gramática*. Sin embargo, las gramáticas de interés son un tanto peculiares pues derivan en árboles siguiendo una serie de *pasos evolutivos*. Estas gramáticas reciben el nombre de *Gramáticas Evolutivas* y funcionan de la siguiente forma.

Gramática

- Una **gramática evolutiva** queda completamente definida a partir de una secuencia de producciones.
- Una **producción** se compone de dos elementos: la cabecera y el cuerpo (ej: $S \rightarrow aB$, donde S es la cabecera y aB el cuerpo).
- La **cabecera** de una producción puede estar compuesta por uno o más símbolos.
- El **cuerpo** de una producción puede estar compuesto por cero o más símbolos.
- Un **símbolo** es cualquier carácter (ej: *a*, *b*, *A*, *B*, *+*, *-*, *&*.)

Por ejemplo, a continuación, se define una gramática evolutiva con una secuencia de 9 producciones. En cada línea, los símbolos antes del "→" constituyen la cabecera de cada producción, y los símbolos luego del "→" constituyen el cuerpo de la producción.

S → AS
S → BS
A → aA
A → a
B → b B
B → b
SS → c
SSSS → d
AA → e

Paso Evolutivo

Se definen dos operaciones básicas para dar un **paso evolutivo** en un árbol según una gramática: *mutación* y *cruzamiento*. Entre un paso evolutivo y el siguiente, se cruzan el árbol del último paso y su mutación.

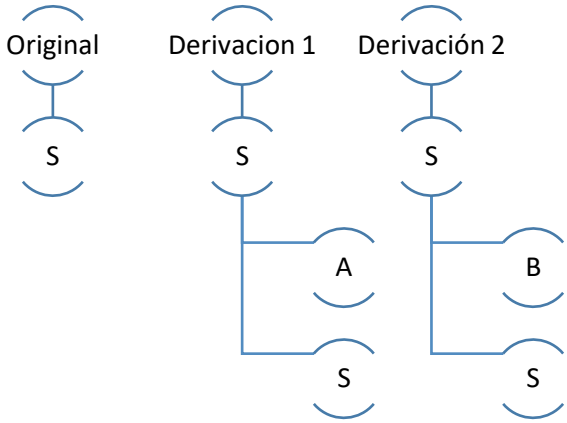
Mutación

La operación de **mutación** de *un árbol* ocurre según las reglas siguientes:

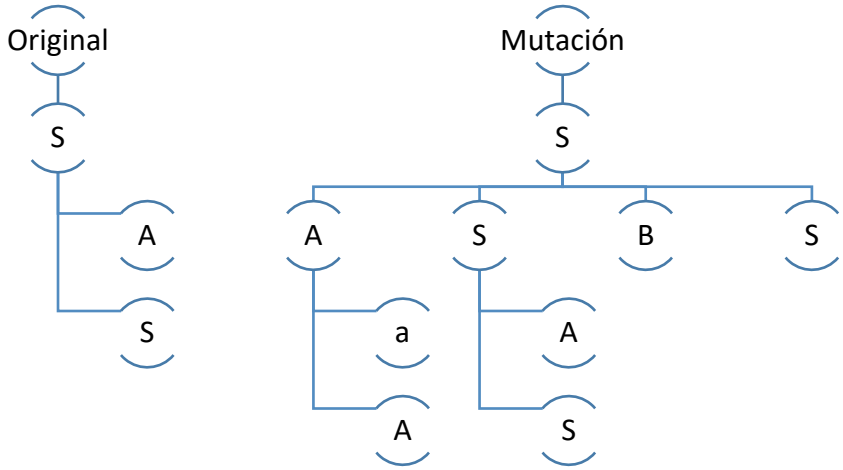
- El resultado de la operación es un nuevo árbol.

- Cada nodo presente en el árbol puede derivar a lo sumo una vez durante la operación.
- Todo nodo debe derivar antes que sus ancestros.
- Todos los nodos presentes en el árbol deben derivar de ser posible.
- Para que un nodo **derive**, debe seleccionar una producción de la gramática cuya cabecera coincida con la cadena de símbolos que contiene el nodo.
- Una vez seleccionada la producción, los símbolos en el cuerpo de la producción se añaden cada uno como un nodo hijo del nodo que derivó. Los nodos se añaden a la derecha de los hijos que ya tuviera el nodo.
- Los nodos creados en la mutación no pueden derivar durante la misma operación.

Por ejemplo, a continuación, se muestra las **dos posibles derivaciones** que puede tener un nodo con símbolo S según la gramática presentada anteriormente.



Además, a continuación, se muestra **una posible mutación** de un árbol según una selección de producciones de la gramática presentada anteriormente.

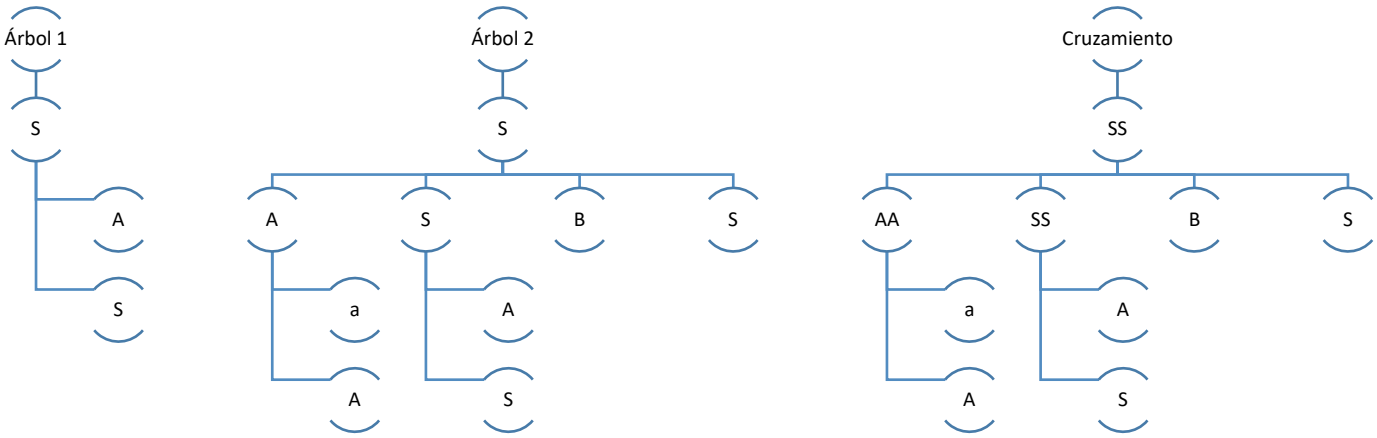


Cruzamiento

La operación de **cruzamiento** entre *dos árboles* sigue las siguientes reglas:

- El resultado de la operación es un nuevo árbol.
- Dado dos árboles, el cruzamiento se obtiene **concatenando** los símbolos que almacena el nodo raíz del primer árbol con los del segundo, y **cruzando** sus respectivos hijos en el orden en que aparecen.
- En caso de que algunos hijos se queden sin pareja, esos nodos hijos se incluyen tal cual sin modificaciones.

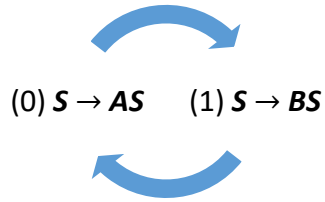
Por ejemplo, a continuación, se muestra el resultado de cruzar dos árboles.



Derivación de un árbol

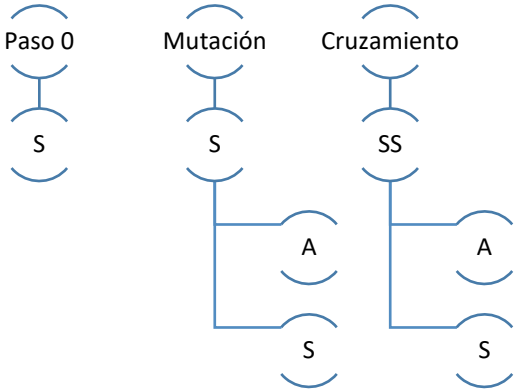
Finalmente, la forma de **derivar un árbol de una gramática evolutiva** es la siguiente:

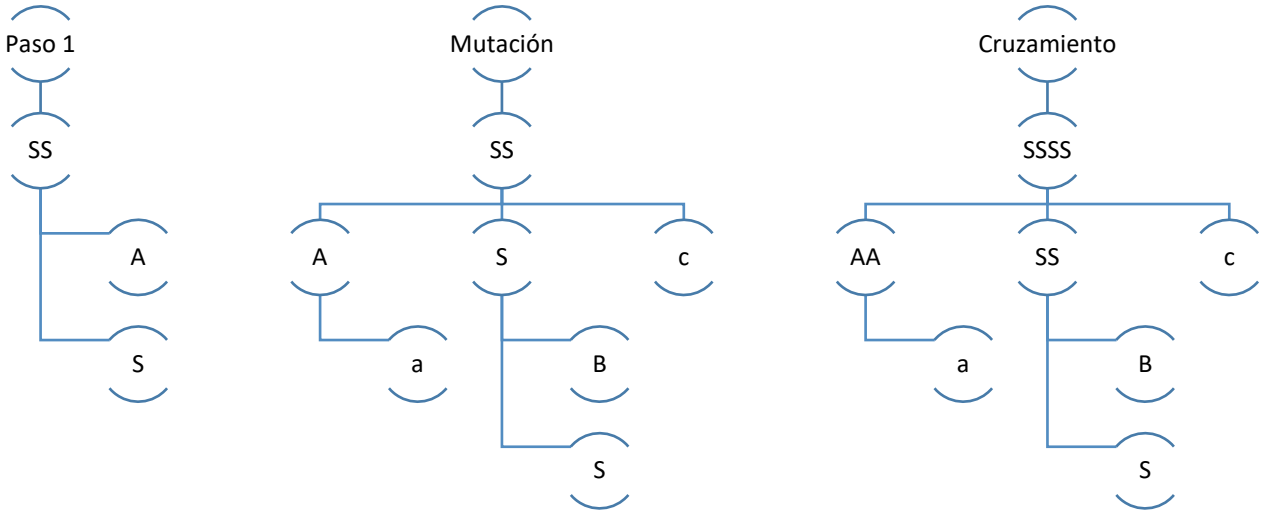
- Entrada:
  - Se **parte de cuatro elementos**:
    - Un árbol n-ario de símbolos.
    - Un número de pasos evolutivos.
    - Una lista de producciones de la gramática evolutiva.
    - Un generador de números aleatorios.
  - El **árbol** almacena en cada nodo una secuencia de símbolos.
  - El **número de pasos evolutivos** indica cuantas evoluciones se quieren realizar antes de concluir el árbol. Derivar un árbol en cero pasos produce el mismo árbol del que se parte.
  - Las **producciones** de la gramática tienen el formato descrito anteriormente.
  - El **generador de números aleatorios** se utiliza para determinar cual de las posibles producciones seleccionar al hacer una derivación.
- Procedimiento:
  - Partiendo del árbol inicial, se realizando tantos pasos evolutivos sobre él como se haya indicado en la entrada.
  - Las producciones a usar para la evolución se obtienen de la lista de producciones de entrada y se van seleccionando según el generador de números aleatorios.
  - Si se tienen *N* posibles producciones aplicar (*aquellas cuya cabecera coincide con la cadena de símbolos a derivar*) y el generador de números aleatorios devuelve mayor o igual que *N*, se selecciona la producción que cíclicamente corresponda (*según se ilustra a continuación para N=2*). El generador siempre devolverá números no negativos.
    Por ejemplo, si se tiene 2 posibles producciones a aplicar entonces el comportamiento debería ser el siguiente:

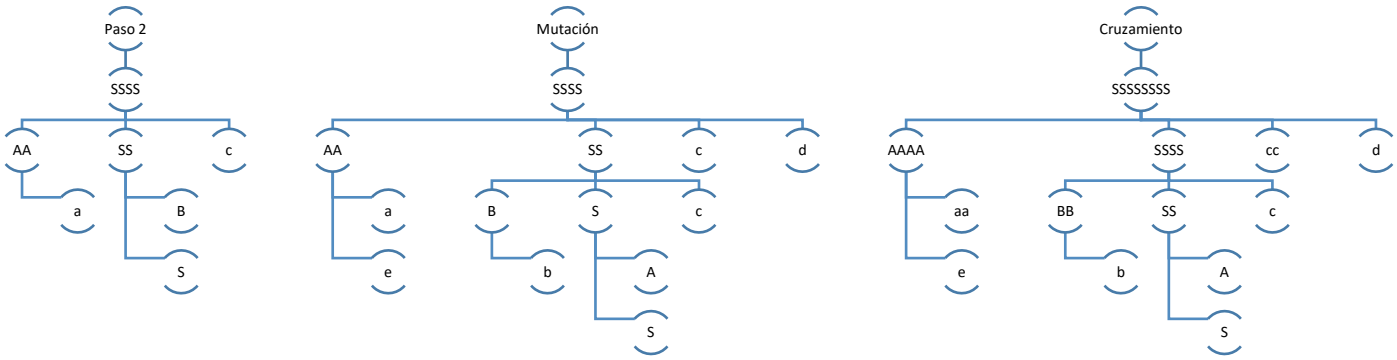


Ejemplo

A continuación, se muestra una posible secuencia de dos pasos de evolución por los que pasa un árbol inicial.







Programa

Usted debe escribir la respuesta al problema en el archivo *Solution.cs*. En *Solution.cs* se encuentra la función *DeriveFromGrammar* dentro de la cual debe escribir su lógica o cualquier llamado a esta.

```
public static Tree DeriveFromGrammar(
    Tree start, int iterations, Production[] productions, Func<int> sampler
)
```

Notas importantes

- *Su solución debe funcionar para cualquier **árbol** y **conjunto de producciones** que se dé como entrada, no solo para los casos ejemplificados.*
- *El **árbol** inicial no tiene por qué tener solo un nodo.*
- *Puede asumir que el **número de pasos evolutivos** nunca será negativo.*
- *No llamar al **generador de números aleatorios** si no se va a utilizar su valor.*
- *Durante la **mutación** de un árbol, todo nodo debe derivar **antes** que sus ancestros.*