Taller N°2: Árbol binario sintáctico

Valentina Paz Campos Olguín
Departamento de Ingeniería Informática
Universidad de Santiago de Chile, Santiago, Chile
valentina.campos.o@usach.cl

I. Explicación breve

La idea principal del programa consiste en construir un árbol binario sintáctico a partir de un archivo de teto con una expresión con notación polaca y poder realizar distintas operaciones matemáticas a este árbol, tales como derivar, simplificar, reemplazar variables, evaluar y más.

II. Heurísticas

Para el almacenamiento de los datos primeramente se empleó una pila que guardó los nodos que fueran de tipo operación para poder conectarlo con los hijos, ya sean variables, números o más operaciones. Para las funciones de evaluar, clonar, derivar, simplificar y ordenar se empleó una técnica recursiva para poder recorrer todos los nodos y operarlos correctamente. En este caso se utilizó fuerza bruta, ya que sí o sí se deben explorar todos los nodos para saber si se pueden simplificar o no.

III. Funcionamiento

Parser Al ingresar el nombre del archivo, se lee carácter por carácter y por cada uno se crea un nodo dependiendo del tipo de dato leído. Si es nodo operación se agrega a la pila y se conecta con los demás nodos, asignando el hijo izquierdo y derecho. Al realizar esto con todos los nodos, se recorre el árbol para conectar los nodos hijos con los nodos padres.

Evaluación de números Ingresa un nodo y se evalúa recursivamente para los hijos izquierdo y derecho. Dependiendo de la operación, se realiza la función y devuelve el resultado de este en forma de nodo. Esto solo sirve mientras todos los nodos sean de tipo número.

Reemplazo de variables Ingresa un nodo, las variables a reemplazar y un arreglo de valores por las cuales se reemplazará. También hará un recorrido recursivo, y mientras no se encuentre con una variable seguirá reemplazando por la izquierda y derecha. Si se encuentra con una variable que fue ingresada, se crea un nuevo nodo de tipo número con el valor ingresado y se conecta con el resto del árbol.

Derivación con respecto a una variable Ingresa un nodo y una variable, por cada operación seguirá una regla determinada. La derivación de la suma es la suma entre la derivada del izquierdo y derecho. La derivación de la resta es la resta entre la derivada del izquierdo y derecho. La derivación de la multiplicación es la suma de la derivada del izquierdo por el derecho y el izquierdo por la derivada del derecho. La derivación de la elevación es la multiplicación entre el exponente y la derivada de la base.

Simplificación del árbol Se recorre el árbol de forma que se busca analizar los casos básicos, como por ejemplo multiplicación por 0, sumar un árbol con 0, restar un árbol con 0, elevar un árbol a 0 y más casos que logran reducir la expresión. Hay un par de casos particulares que emplea el método equal(), el cuál requiere de dos nodos para determinar si estos árboles contienen los mismos datos o no. En caso de que sí, se puede simplificar de mejor forma 2 árboles complejos que sean iguales al restarse, multiplicarse o sumarse. Por cada simplificación se debe realizar un enlace para asegurarse de que los nodos estén conectados correctamente y así imprimir el árbol. Este método debe ser repetido varias veces para alcanzar la máxima simplificación, y esto se logra según cuán simplificado el usuario quiera el árbol. Por lo tanto, si ingresa un número mayor, más simplificado quedará el árbol.

Todas estas funciones están disponibles para realizar en el menú, pudiendo además volver al árbol original.

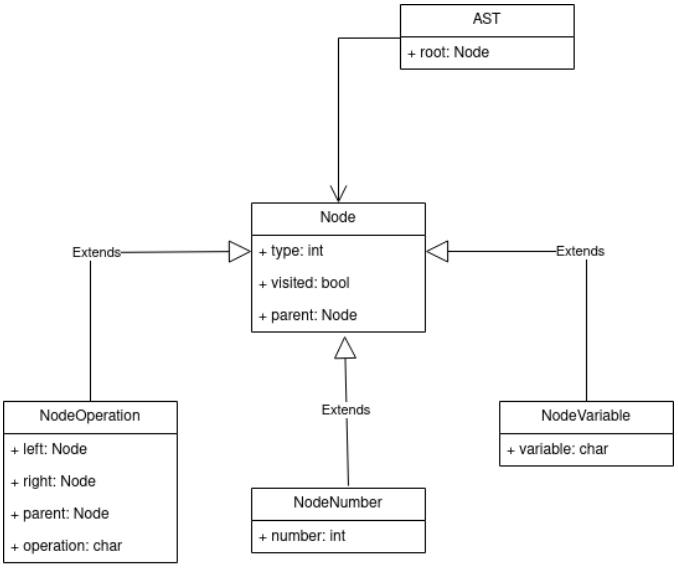

Figura 1: Diagrama de clases de la solución implementada

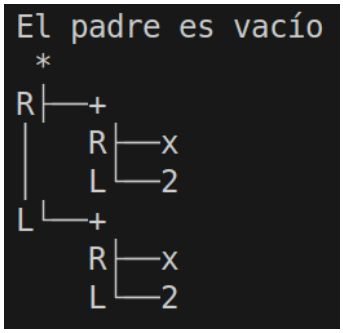

Figura 2: Árbol inicial

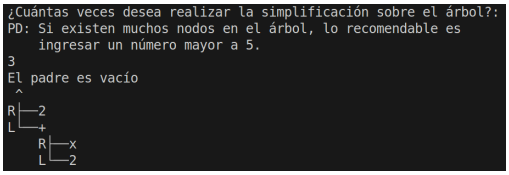

Figura 3: Árbol simplificado

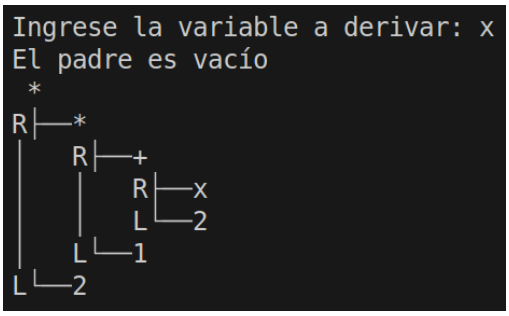

Figura 4: Árbol derivado con respecto a x

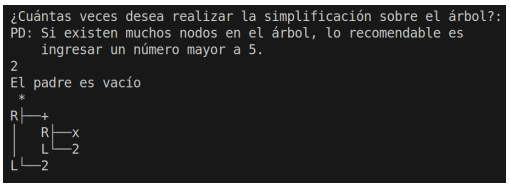

Figura 5: Árbol derivado y simplificado

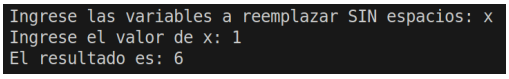

Figura 6: Evaluación de árbol derivado con x = 1

IV. Implementación y eficiencia

El código propuesto es eficiente para resolver problemas matemáticos, simplificando y resolviendo adecuadamente las expresiones, debido a que se emplea la librería STL, esta siendo óptima a la hora de ingresar, sacar o conectar los datos de una pila.

Cuadro I: Requerimientos físicos empleados

Materiales	Especificación
Notebook	Lenovo Yoga Slim 7
GPU	AMD Radeon
CPU	AMD Ryzen 5 4500U
Memoria	SSD 256GB
RAM	8GB
Sistema operativo	Linux
Distribución	Ubuntu
Versión	22.04.2LTS

A continuación se proponen expresiones de ejemplo y el resultado de aplicar cada una de las operaciones explicadas anteriormente.

Ejemplo 1:
* + x 2 + x 2

V. Ejecución

Se utilizaron las librerías string, fstream, iostream, sstream, stack y cmath.

El código implementado es eficiente, ya que minimiza la cantidad de nodos de árboles que pueden parecer complejos,

deriva correctamente, simplifica a árboles ya derivados o simplificados anteriormente, y es fácil de manejar para quién quiera operar sobre el árbol.

Para ejecutar el código se debe realizar lo siguiente:

1. Descomprimir la carpeta
2. Ingresar por terminal a la carpeta
3. Ingresar "make"por consola y pulsar enter.
4. Para correr el programa principal, se debe introducir "./main"
5. Para correr el test de NodeNumber, se debe ingresar "./testNodeNumber"
6. Para correr el test de NodeVariable, se debe ingresar "./testNodeVariable"
7. Para correr el test de NodeOperation, se debe ingresar "./testNodeOperation"
8. Para correr el test de AST, se debe ingresar "./testAST"