
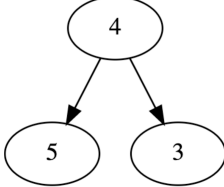
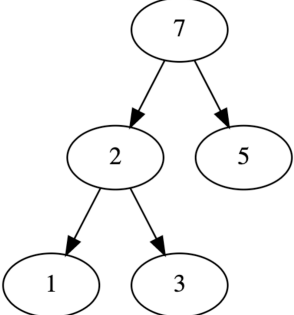
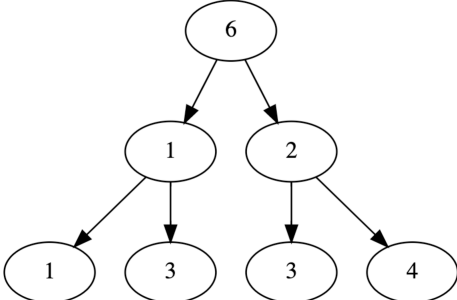


Para la representación de árboles binarios no vacíos de números enteros se puede usar la siguiente notación:

Cadena del lenguaje	Árbol
5	
{5,4,3}	
{{1,2,3},7,5}	
{{{1,1,3},6,{3,2,4}}}	

Se proporciona un analizador léxico implementado mediante *JFlex*, en el fichero *Arbol.flex*; la clase *Arbol.java* que contiene el método *main*, la salida, y las variables que deben usarse para la implementación. El fichero de entrada consiste en una sola línea que contiene una descripción de un árbol.

Para cada fichero que contiene un árbol se pide

- Construir una gramática que solo acepte como correctas cadenas de este lenguaje.
- Hallar el nodo raíz del árbol
- Hallar el valor del máximo nodo del árbol (que puede estar en las ramas o en las hojas)
- Hallar profundidad del árbol, entendiendo como tal la longitud del camino mas largo desde la raíz a algún nodo hoja.
- Hallar el número de nodos del árbol.
- Hallar la suma por niveles de los enteros del árbol, es decir, la suma de todos los nodos del mismo nivel de profundidad. (ver ejemplos).
- Hallar el árbol inverso, es decir, simétrico de izquierda a derecha, representado mediante una cadena de caracteres. (ver ejemplos).
- Hallar la representación del grafo del árbol, que consiste en el despliegue de todas las ramas del árbol, siguiendo el recorrido de izquierda a derecha, mediante una secuencia de expresiones de la forma:  $n \rightarrow \{n1, n2\}$ ; que indican que hay una flecha entre los nodos  $n$  y los nodos  $n1$  y  $n2$ . (ver ejemplos).

A continuación se muestra el resultado de ejecución de diversos ejemplos

Entrada	Salida
5	Raiz: 5 Maximo: 5 Profundidad: 1 Elementos: 1 Suma nivel 0: 5 Arbol inverso: 5 Grafo:
{5, 4, 3}	Raiz: 4 Maximo: 5 Profundidad: 2 Elementos: 3 Suma nivel 0: 4 Suma nivel 1: 8 Arbol inverso: {3,4,5} Grafo: 4 -> {5,3};
{{1, 2, 3}, 7, 5}	Raiz: 7 Maximo: 7 Profundidad: 3 Elementos: 5 Suma nivel 0: 7 Suma nivel 1: 7 Suma nivel 2: 4 Arbol inverso: {5,7,{3,2,1}} Grafo: 2 -> {1,3}; 7 -> {2,5};
{{1, 1, 3}, 6, {3, 2, 4}}	Raiz: 6 Maximo: 6 Profundidad: 3 Elementos: 7 Suma nivel 0: 6 Suma nivel 1: 3 Suma nivel 2: 11 Arbol inverso: {{4,2,3},6,{3,1,1}} Grafo: 1 -> {1,3}; 6 -> {1,2}; 2 -> {3,4};
{ {{1, 2, 3}, 4, {5, 6, 7}}, 8, { 9, 10, {{11, 12, 13}, 14, 15} } }	Raiz: 8 Maximo: 15 Profundidad: 5 Elementos: 15 Suma nivel 0: 8 Suma nivel 1: 14 Suma nivel 2: 31 Suma nivel 3: 43 Suma nivel 4: 24 Arbol inverso: {{{15,14,{13,12,11}},10,9},8,{{7,6,5},4,{3,2,1}}} Grafo: 2 -> {1,3}; 4 -> {2,6}; 6 -> {5,7}; 8 -> {4,10}; 10 -> {9,14}; 12 -> {11,13}; 14 -> {12,15};
{5, 4, 3, {2, 1}}	Syntax error ...

Para compilar y ejecutar este interprete se usarán la secuencia:

```
cup Arbol.cup
jflex Arbol.flex
javac Arbol.java
java Arbol <entrada>
```

### **NOTAS IMPORTANTES:**

- \* Se entrega solo el fichero *Arbol.cup* por lo que NO SE DEBEN MODIFICAR los demás ficheros, ya que la compilación se hará a partir de los ficheros originales.
- \* Para la resolución de este ejercicio NO DEBEN USARSE VARIABLES GLOBALES, ni métodos auxiliares. Para ello, se comprobará que no se ha incluido en el fichero *Arbol.cup* ninguna sección “action code” o “parser code”
- \* La implementación debe funcionar para TODAS las cadenas del lenguaje, no solamente para los ejemplos que aparecen en este enunciado.