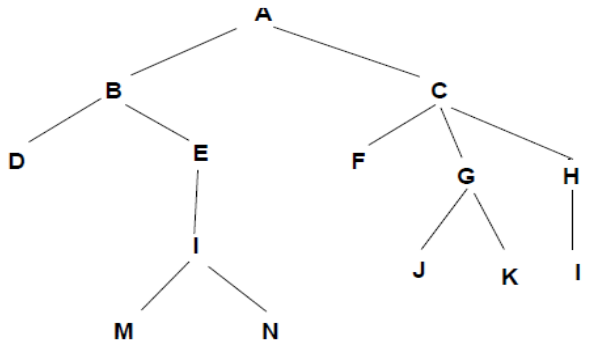
Universidad Católica de Salta
Facultad de Ingeniería
Estructuras de Datos y Algoritmos

Mag. Rosa Macaione

Bautista Genovese – 46.790.865
Trabajo Práctico 9: Contenedores No Lineales Arboles

Ejercicio 1:
Dado el siguiente árbol:



Responda las siguientes preguntas para el árbol de la figura:

a. ¿Qué nodos son hojas?
b. ¿Qué nodo es raíz?
c. ¿Cuál es el padre del nodo C?
d. ¿Qué nodos son los hijos de C?
e. ¿Qué nodos son los antecesores de C?
f. ¿Qué nodos son los descendientes de E?
g. ¿Cuáles son los hermanos derechos de D y E?
h. ¿Qué nodos están a izquierda y que nodos a derecha de G?
i. ¿Cuál es la profundidad del nodo C?
j. ¿Cuál es la altura del nodo C?
k. ¿Cuántos caminos de longitud 3 hay en el árbol representado en la figura anterior?

Desarrollo:

a. Los nodos que son hojas son: D, M, N, F, J, K, I
b. El nodo A es el nodo principal de una estructura de datos en forma de árbol, desde el cual parten todas las demás ramas.
c. El nodo padre de C es el A
d. Nodos hijos de C: F, G, H
e. Antecesor a C: A
f. Descendientes de E: I, M, N
g. Hermanos de D y E: No tienen, ellos son los hermanos
h. A la izquierda de G está F, y a la derecha está H.
i. Profundidad del nodo C: 3.
j. Altura del nodo C: 2
k. Cantidad de caminos: 15

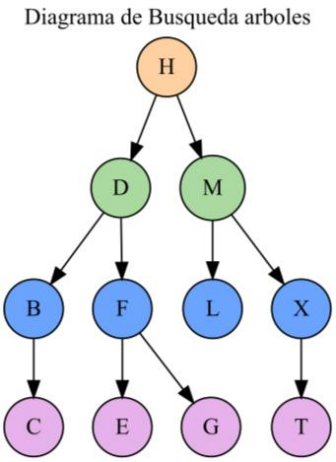
Ejercicio 2:
Para la siguiente lista de letras y Números

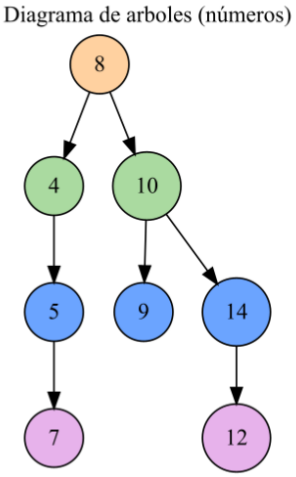
H D F M B X G T C L E
8-4-5-7-10-14-9-12

a) Dibujar los árboles binarios de búsqueda que se construye cuando las letras o números se insertan en el orden dado

b) Realizar los recorridos *EnOrden*, *PreOrden* y *PostOrden* de los árboles obtenidos en a) y mostrar la secuencia de los datos que resultan en cada caso.

Desarrollo:
a)





b) **EnOrden:** B-C-D-E-F-G-H-L-M-T-X / 4-5-7-8-9-10-12-14

PreOrden: H-D-B-C-F-E-G-M-L-X-T / 8-4-5-7-10-9-14-12

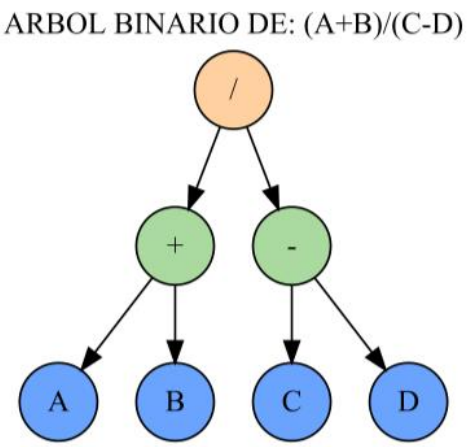
PostOrden: C-B-E-G-F-D-L-T-X-M-H / 7-5-4-9-12-14-10-8

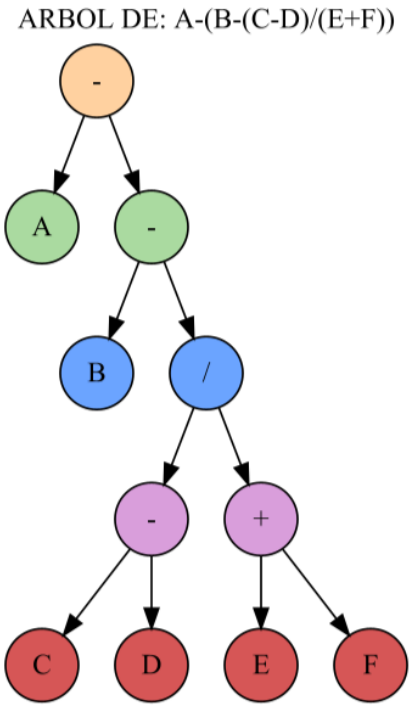
Ejercicio 3:

Dibujar los árboles binarios que representan las siguientes expresiones:

a) (A+B)/(C-D)

b) A-(B-(C-D)/(E+F))





Ejercicio 4:

Dadas las clases JAVA:

- *clsBinaryTreeNode*: clase para nodo de árbol binario.
- *clsBinaryTree*: clase abstracta correspondiente a una implementación para un TAD Árbol Binario.
- *clsABB*: clase abstracta correspondiente a una implementación para un TAD Árbol Binario de Búsqueda. Es una extensión de la clase anterior.
- *ClsIntABB*: clase adicional que permite operar ABB de números enteros.

Desarrollar las siguientes operaciones:

a) int *getLeaves(clsBinaryTreeNode node)*: devuelve la cantidad de nodos hoja de un árbol binario.
b) int *getChildren(clsBinaryTreeNode node)*: devuelve la cantidad de hijos de un árbol binario.
c) boolean i*sComplete (clsBinaryTreeNode node)*: devuelve verdadero si el árbol es completo o falso en caso contrario.
d) int *getHeight(clsBinaryTreeNode node)*: devuelve la altura del nodo/árbol.
e) int *getDifference(clsBinaryTreeNode node)*: devuelve la diferencia de altura entre subárbol izquierdo y derecho del nodo/árbol.
f) *void postOrder(clsBinaryTreeNode node)*: recorrido post-order para árbol binario.
g) *void meter(Object nodeInfo)*: operación meter() en forma recursiva para un árbol binario de búsqueda.
h) Pruebe con las dos secuencias de datos del ejercicio número 2, ¿Cuál sería la estructura de datos del ABB correspondiente?

Desarrollo:

h) Código en el main:

```
public static void main(String[] args) {
    System.out.println("--- Prueba con la secuencia de números ---");

    clsABB arbolNumeros = new clsABB();
    int[] numeros = {8, 4, 5, 7, 10, 14, 9, 12};

    for (int numero : numeros) {
        arbolNumeros.meter(numero);
    }

    arbolNumeros.postOrden();
    System.out.println("Altura del árbol: " + arbolNumeros.getAltura());
    System.out.println("Cantidad de hojas: " + arbolNumeros.getAltura());

    // ------------------------------------------------------------------

    System.out.println("\n--- Prueba con la secuencia de letras ---");

    clsABB arbolLetras = new clsABB();
    String[] letras = {"H", "D", "F", "M", "B", "X", "G", "T", "C", "L", "E"};

    for (String letra : letras) {
        arbolLetras.meter(letra);
    }

    arbolLetras.postOrden();
}
```

Ejercicio 5:

Dada la siguiente declaración de árbol binario:

```
public class clsNodoArbolBinario {
    private Object nodoInfo;
    private clsNodoArbolBinario hijoIzquierdo, hijoDerecho;

     clsNodoArbolBinario() {
        this.setNodoInfo(null);
        this.setHijoIzquierdo(null);
        this.setHijoDerecho(null);
    }
```

SE PIDE:

Codificar un algoritmo que, recibiendo un árbol perteneciente al tipo anteriormente descrito y dos números enteros, *n1* y *n2*, indique si la suma de las claves que se encuentra a nivel *n1* es igual a la suma de las claves que se encuentra a nivel *n2*.

OBSERVACIONES:

- Sólo se permitirá la realización de un único recorrido en el árbol.
- No se permitirá la utilización de ninguna estructura de datos auxiliares.
- Se supone que en el árbol siempre van a existir claves en los niveles *n1* y *n2*.

Reestructure el siguiente código, teniendo en cuenta la implementación dada en clase.

Esta clase se llamará MetodoArbolBinario

```
static int suma (NodoArbol arbol, int n1, int n2, int nivel) {
    int aux = 0;
    if (arbol != null) {
        if (nivel == n1)
            aux = arbol.clave;
        else
        if (nivel == n2)
            aux = -arbol.clave;
        else aux = 0;
        if ((nivel < n2) || (nivel < n1))
            aux = aux + suma (arbol.iz, n1, n2, nivel+1) + suma (arbol.de, n1,
n2, nivel+1);
    }
    return aux;
}
static boolean comprobarSumaClavesDosNiveles (Arbol arbol, int n1, int n2) {
    boolean aux = false;
    if (arbol.raiz != null)
        aux = (suma (arbol.raiz, n1, n2, 1) == 0);
    return aux;
}
```

```
public static void main(String[] args) {
    clsABB arbol = new clsABB();
    int[] valores = {8, 3, 10, 1, 6, 14, 4, 7, 13};
    for (int valor : valores) {
        arbol.meter(valor);
    }

    System.out.println("----------------------------------------");
    System.out.println("El árbol resultante tiene esta estructura:");
    System.out.println("      8        <-- Nivel 1 (Suma = 8)");
    System.out.println("     / \\");
    System.out.println("    3   10     <-- Nivel 2 (Suma = 3 + 10 = 13)");
    System.out.println("   / \\    \\");
    System.out.println("  1   6    14  <-- Nivel 3 (Suma = 1 + 6 + 14 = 21)");
    System.out.println("     / \\   /");
    System.out.println("    4   7 13   <-- Nivel 4 (Suma = 4 + 7 + 13 = 24)");
    System.out.println("----------------------------------------");

    int nivelA = 2;
    int nivelB = 3;
    System.out.println("Prueba 1: Comparamos Nivel " + nivelA + " (Suma=13) con
Nivel " + nivelB + " (Suma=21)");

    boolean resultado1 = MetodoArbolBinario.comprobarSumaClavesDosNiveles(arbol,
nivelA, nivelB);
    System.out.println(">>> El resultado es: " + resultado1 + " (Correcto, porque
13 != 21)");
    System.out.println("----------------------------------------");

    int nivelC = 4;
    System.out.println("Prueba 2: Comparamos Nivel " + nivelC + " (Suma=24) con-
sigo mismo");

    boolean resultado2 = MetodoArbolBinario.comprobarSumaClavesDosNiveles(arbol,
nivelC, nivelC);
    System.out.println(">>> El resultado es: " + resultado2 + " (Correcto, porque
24 == 24)");
}
```

Se recomienda probar el código