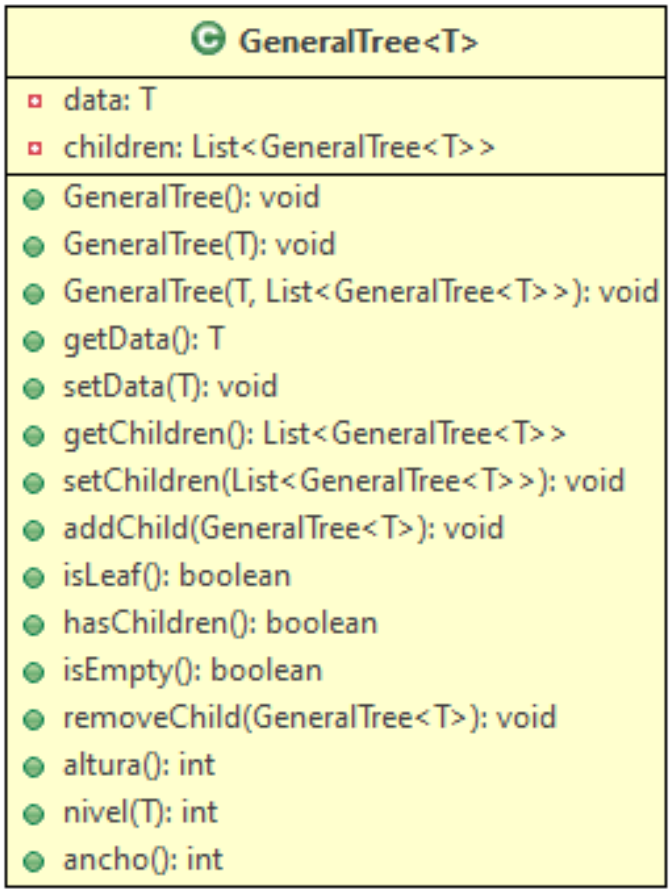
Juan Menduiña

Trabajo Práctico N° 3:
Árboles Generales.

Ejercicio 1.

Considerar la siguiente especificación de la clase GeneralTree (con la representación de Lista de Hijos).



- *El constructor GeneralTree(T data) inicializa un árbol que tiene como raíz un nodo y este nodo tiene el dato pasado como parámetro y una lista vacía.*
- *El constructor GeneralTree(T data, List<GeneralTree <T>> children) inicializa un árbol que tiene como raíz a un nodo y este nodo tiene el dato pasado como parámetro y como hijos children.*
- *El método getData(): T retorna el dato almacenado en la raíz del árbol.*
- *El método getChildren(): List<GeneralTree <T>> retorna la lista de hijos de la raíz del árbol.*
- *El método addChild(GeneralTree <T> child) agrega un hijo al final de la lista de hijos del árbol.*
- *El método hasChildren() devuelve verdadero si la lista de hijos del árbol no es null y tampoco es vacía.*
- *El método isEmpty() devuelve verdadero si el dato del árbol es null y, además, no tiene hijos.*

- *El método removeChild(GeneralTree <T> child) elimina del árbol el hijo pasado como parámetro.*
- *Los métodos altura(), nivel(T) y ancho() se resolverán en el Ejercicio 3.*

Analizar la implementación en JAVA de la clase GeneralTree brindada por la cátedra.

Ver paquete “tp3.ejercicio1” en Java.

Ejercicio 2.

(a) *Implementar, en la clase RecorridosAG, los siguientes métodos:*

- *public List<Integer> numerosImparesMayoresQuePreOrden(GeneralTree<Integer> a, Integer n) Método que retorna una lista con los elementos impares del árbol "a" que sean mayores al valor "n" pasados como parámetros, recorrido en preorden.*
- *public List<Integer> numerosImparesMayoresQueInOrden(GeneralTree<Integer> a, Integer n) Método que retorna una lista con los elementos impares del árbol "a" que sean mayores al valor "n" pasados como parámetros, recorrido en inorden.*
- *public List<Integer> numerosImparesMayoresQuePostOrden(GeneralTree<Integer> a, Integer n) Método que retorna una lista con los elementos impares del árbol "a" que sean mayores al valor "n" pasados como parámetros, recorrido en postorden.*
- *public List<Integer> numerosImparesMayoresQuePorNiveles(GeneralTree<Integer> a, Integer n) Método que retorna una lista con los elementos impares del árbol "a" que sean mayores al valor "n" pasados como parámetros, recorrido por niveles.*

(b) *Si, ahora, se tuviera que implementar estos métodos en la clase GeneralTree<T>, ¿qué modificaciones se harían tanto en la firma como en la implementación de los mismos?*

Ver paquete "tp3.ejercicio2" en Java.

Ejercicio 3.

Implementar, en la clase GeneralTree, los siguientes métodos:

(a) *public int altura(): Devuelve la altura del árbol, es decir, la longitud del camino más largo desde el nodo raíz hasta una hoja.*

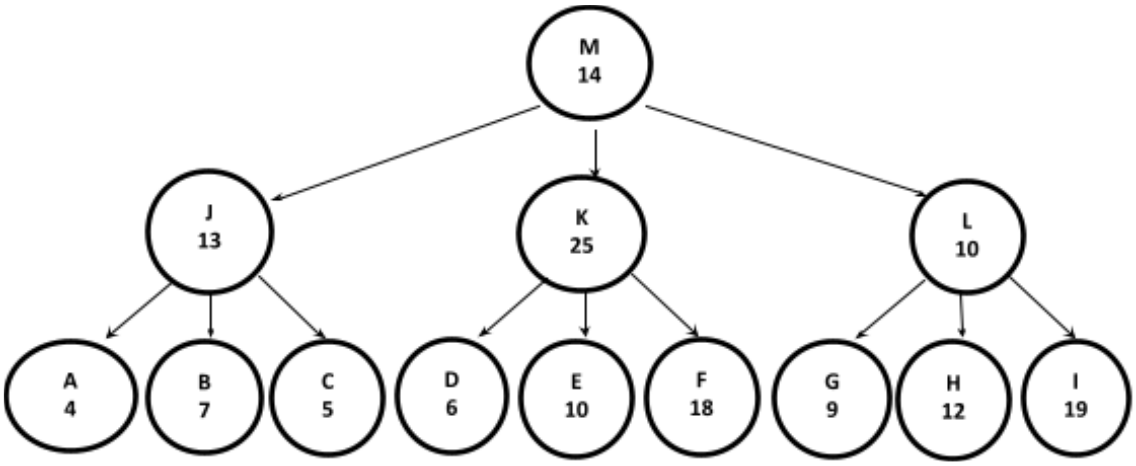
(b) *public int nivel(T dato): Devuelve la profundidad o nivel del dato en el árbol. El nivel de un nodo es la longitud del único camino de la raíz al nodo.*

(c) *public int ancho(): Devuelve la amplitud (ancho) de un árbol, que se define como la cantidad de nodos que se encuentran en el nivel que posee la mayor cantidad de nodos.*

Ver paquete “tp3.ejercicio3” en Java.

Ejercicio 4.

El esquema de comunicación de una empresa está organizado en una estructura jerárquica, en donde cada nodo envía el mensaje a sus descendientes. Cada nodo posee el tiempo que tarda en transmitir el mensaje.



Se debe devolver el mayor promedio entre todos los valores promedios de los niveles. Para el ejemplo presentado, el promedio del nivel 0 es 14, el del nivel 1 es 16 y el del nivel 2 es 10. Por lo tanto, debe devolver 16.

(a) *Indicar y justificar qué tipo de recorrido se utilizará para resolver el problema.*

El tipo de recorrido que se utilizará para resolver el problema será por niveles.

(b) *Implementar, en una clase AnalizadorArbol, el método con la siguiente firma:*

public double devolverMaximoPromedio(GeneralTree<AreaEmpresa> arbol),

donde AreaEmpresa es una clase que representa a un área de la empresa mencionada y que contiene la identificación de la misma representada con un String y una tardanza de transmisión de mensajes interna representada con int.

Ver paquete “tp3.ejercicio4” en Java.

Ejercicio 5.

Se dice que un nodo n es ancestro de un nodo m si existe un camino desde n a m. Implementar un método en la clase GeneralTree con la siguiente firma:

public boolean esAncestro(T a, T b): Devuelve true si el valor "a" es ancestro del valor "b".

Ver paquete "tp3.ejercicio5" en Java.

Ejercicio 6.

Sea una red de agua potable, la cual comienza en un caño maestro y la misma se va dividiendo, sucesivamente, hasta llegar a cada una de las casas. Por el caño maestro, ingresan "x" cantidad de litros y, en la medida que el caño se divide de acuerdo con las bifurcaciones que pueda tener, el caudal se divide en partes iguales en cada una de ellas. Es decir, si un caño maestro recibe 1.000 litros y tiene, por ejemplo, 4 bifurcaciones, se divide en 4 partes iguales, donde cada división tendrá un caudal de 250 litros. Luego, si una de esas divisiones se vuelve a dividir, por ejemplo, en 5 partes, cada una tendrá un caudal de 50 litros y así sucesivamente hasta llegar a un lugar sin bifurcaciones.

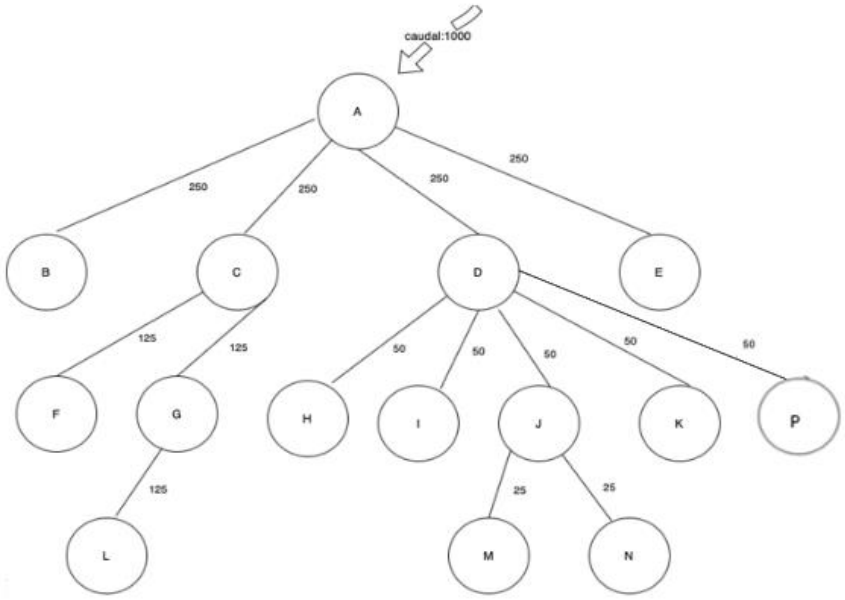
Se debe implementar una clase RedDeAguaPotable que contenga el método con la siguiente firma:

public double minimoCaudal(double caudal),

que calcule el caudal de cada nodo y determine cuál es el caudal mínimo que recibe una casa.

Asumir que la estructura de caños de la red está representada por una variable de instancia de la clase RedAguaPotable y que es un GeneralTree<Character>.

Extendiendo el ejemplo en el siguiente gráfico, al llamar al método minimoCaudal con un valor de 1.000,0, debería retornar 25,0.



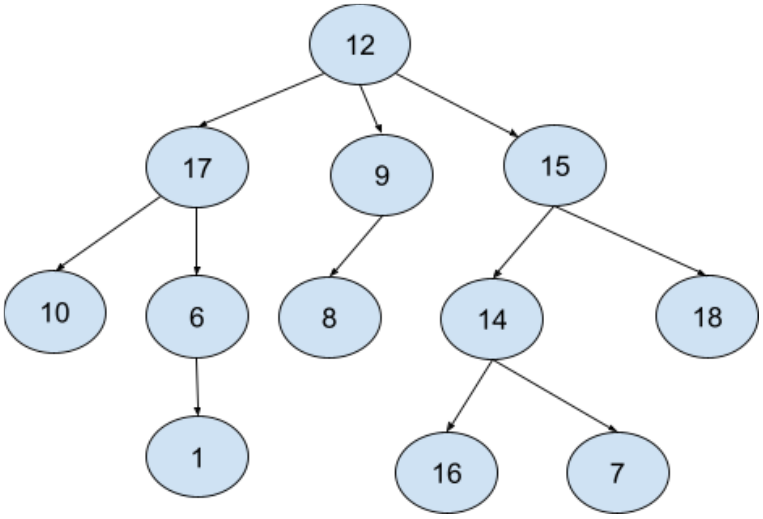
Ver paquete "tp3.ejercicio6" en Java.

Ejercicio 7.

Dada una clase Caminos que contiene una variable de instancia de tipo GeneralTree de números enteros, implementar un método que retorne el camino a la hoja más lejana. En el caso de haber más de un camino máximo, retornar el primero que se encuentre. El método debe tener la siguiente firma:

public List<Integer> caminoAHojaMasLejana().

Por ejemplo, para el siguiente árbol, la lista a retornar sería: 12, 17, 6, 1 de longitud 3 (los caminos 12, 15, 14, 16 y 12, 15, 14, 7 son también máximos, pero se pide el primero).



Ver paquete “tp3.ejercicio7” en Java.

Ejercicio 8.

Retomando el ejercicio abeto navideño visto en teoría, crear una clase Navidad que cuenta con una variable de instancia GeneralTree que representa al abeto (ya creado) e implementar el método con la firma: public String esAbetoNavidenio().

Ver paquete “tp3.ejercicio8” en Java.

Los siguientes ejercicios fueron tomados en parciales, en los últimos años. Tener en cuenta que:

1. No se pueden agregar más variables de instancia ni de clase a la clase ParcialArboles.
2. Se debe respetar la clase y la firma del método indicado.
3. Se puedem definir todos los métodos y variables locales que se consideren necesarios.
4. Todo método que no esté definido en la sinopsis de clases debe ser implementado.
5. Se debe recorrer la estructura sólo 1 vez para resolverlo.
6. Si corresponde, completar, en la firma del método, el tipo de datos indicado con signo de "?".

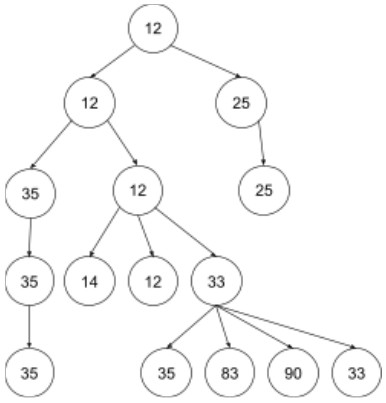
Ejercicio 9.

Implementar, en la clase ParcialArboles, el método:

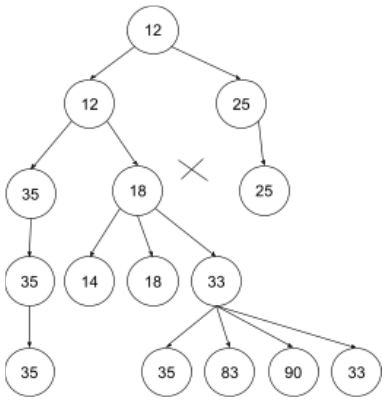
public static boolean esDeSeleccion(GeneralTree<Integer> arbol),

que devuelve true si el árbol recibido por parámetro es de selección, falso sino lo es.

Un árbol general es de selección si cada nodo tiene, en su raíz, el valor del menor de sus hijos. Por ejemplo, para el siguiente árbol, se debería retornar true.



Para este otro árbol, se debería retornar false (el árbol con raíz 18 tiene un hijo con valor mínimo 14).



Ver paquete "tp3.ejercicio9" en Java.

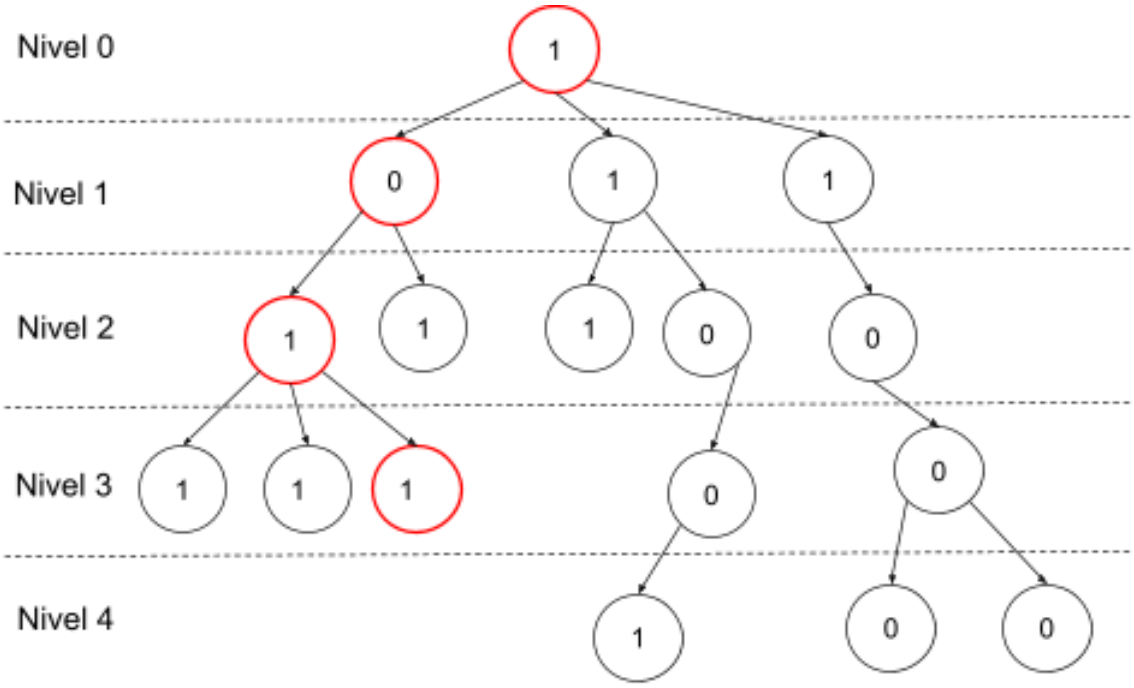
Ejercicio 10.

Implementar la clase ParcialArboles y el método:

public static List<Integer> resolver(GeneralTree<Integer> arbol),

que recibe un árbol general de valores enteros, que sólo pueden ser 0 o 1, y devuelve una lista con los valores que componen el "camino filtrado de valor máximo". Se llama "filtrado" porque sólo se agregan al camino los valores iguales a 1 (los 0 no se agregan), mientras que es "de valor máximo" porque se obtiene de realizar el siguiente cálculo: es la suma de los valores de los nodos multiplicados por su nivel. De haber más de uno, devolver el primero que se encuentre.

*Por ejemplo, para el árbol general que aparece en el gráfico, el resultado de la invocación al método resolver debería devolver una lista con los valores 1, 1, 1, y NO 1, 0, 1, 1, dado que se filtró el valor 0. Con esa configuración, se obtiene el mayor valor según el cálculo: 1*0 + 0*1 + 1*2 + 1*3 (el camino 1*0 + 1*1 + 0*2 + 0*3 + 1*4 también da 5, pero no es el primero).*



NOTA: No se puede generar la lista resultado con 0/1 y, en un segundo recorrido, eliminar los elementos con valor 0.

Ver paquete "tp3.ejercicio10" en Java.

Ejercicio 11.

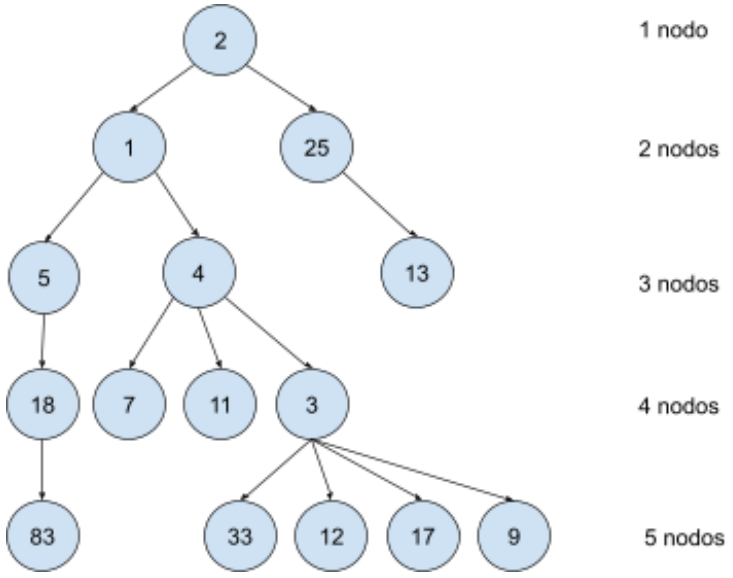
Implementar, en la clase ParcialArboles, el método:

public static boolean resolver(GeneralTree<Integer> arbol),

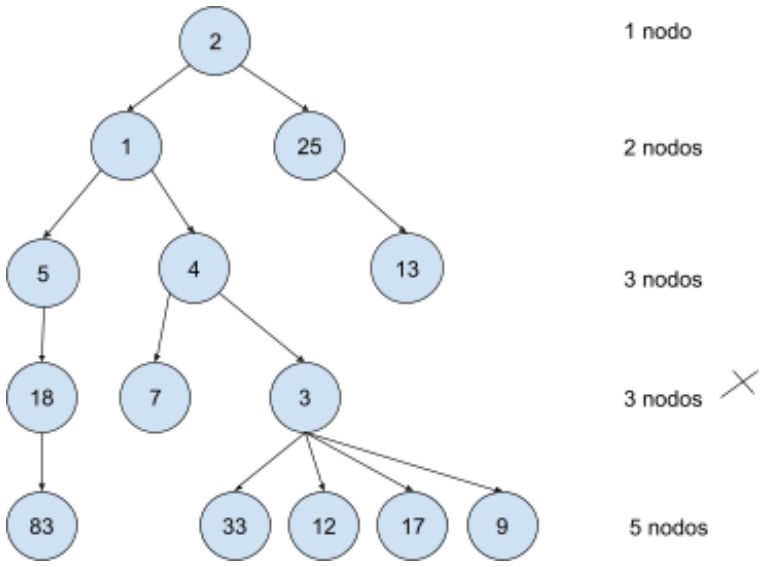
que devuelve true si el árbol es creciente, falso sino lo es.

Un árbol general es creciente si, para cada nivel del árbol, la cantidad de nodos que hay en ese nivel es, exactamente, igual a la cantidad de nodos del nivel anterior + 1.

Por ejemplo, para el siguiente árbol, se debería retornar true.



Para este otro árbol, se debería retornar false (ya que, en el nivel 3, debería haber 4 nodos y no 3).



Ver paquete “tp3.ejercicio11” en Java.