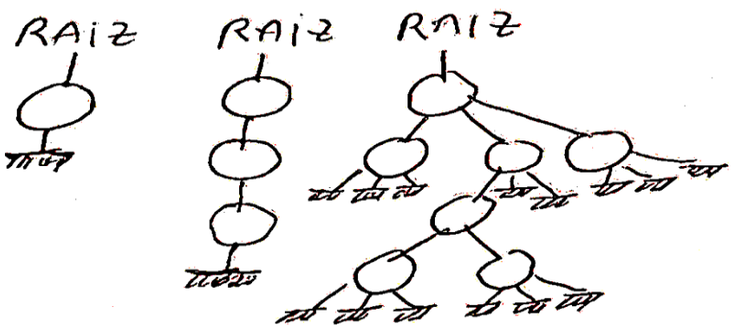
ARBOL

Es una estructura de datos **NO LINEAL** definida del siguiente modo "**Un árbol es una estructura que posee un enlace dominado RAIZ que apunta a UNO o NINGUN nodo. Todo nodo de este árbol apunta NIGUNO, UNO o MAS nodos**"

Entonces estos son árboles…



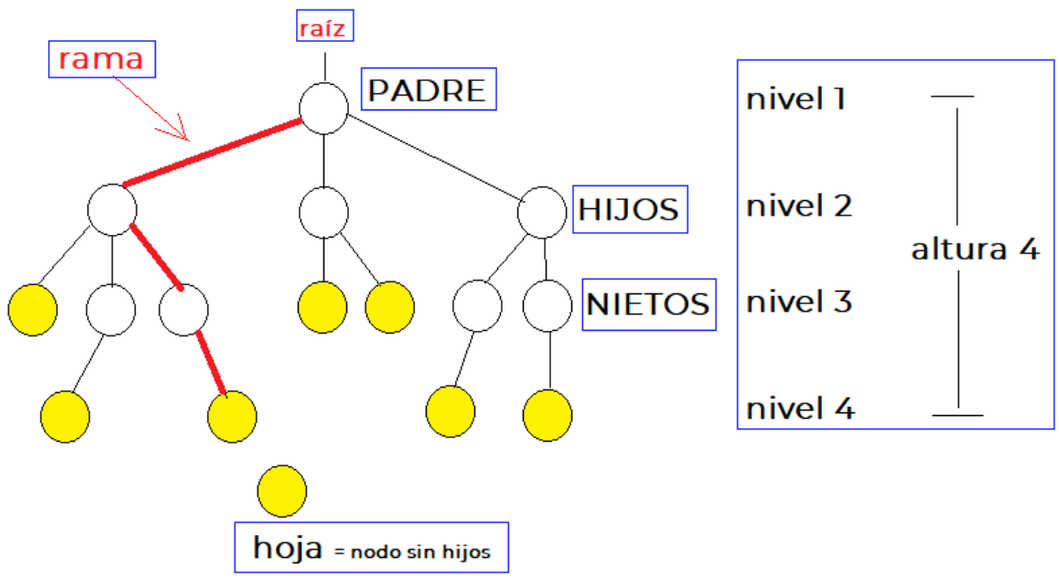
Pues todas ellos cumplen con la definición (se acostumbra a dibujar los nodos de un árbol usando círculos)

Entonces una Lista Enlazada también es un árbol, en el cual cada nodo tiene un solo enlace a otro.

Pero en ese caso la estructura es LINEAL (cada nodo tiene un anterior y un siguiente, excepto el primer nodo y el último nodo) y se maneja de modo diferente a como se maneja un árbol.

NOMENCLATURAS USADAS CON ARBOLES

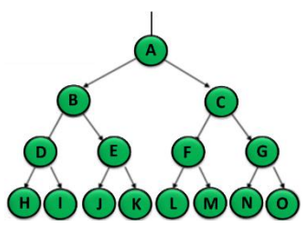
Para denominar algunas características de los árboles se usan términos tales como…



ARBOL BINARIO

La definición anterior permite imaginar árboles inmanejables, en que cada nodo tenga muchos enlaces a otros nodos.

Por eso los árboles usados para crear estructuras de datos son esencialmente **BINARIOS**, es decir árboles en que cada nodo tiene a lo más dos enlaces a otros nodos.



AGREGAR UN NODO A UN ARBOL BINARIO

Se tiene el enlace **RAIZ** en que comienza el árbol y el enlace **NUEVO** que apunta a un nuevo nodo ¿Cómo agregar ese nuevo nodo al árbol?

Pasándole esos dos enlaces a un método **AGREGAR** para que agregue ese nuevo nodo en algún libre en dicho árbol.

¿Qué debería devolver ese método?
¡**El mismo enlace RAIZ que se le pasó**!, **idéntico a como lo recibió** o **cambiado**

¿Qué? ¿Qué significa eso?
Cuando a ese método se le pasan los enlaces **RAIZ** y **NUEVO** podrían darse dos situaciones…

- Si el árbol está vacío, el **NUEVO** nodo será el primero y por lo tanto **RAIZ** debe quedar apuntando a él, entonces el método **AGREGAR** debe hacer **RAIZ = NUEVO**. Como lo que devuelve es **RAIZ** entonces ahora **RAIZ** apuntará a su primer y único nodo
- Si el árbol no está vacío entonces el método **AGREGAR** recorrerá nodo por nodo hasta encontrar un lugar libre y allí colocará el nuevo nodo, por lo cual el enlace **RAIZ no cambiará** (quien cambiará será un enlace del nodo al cual quedó conectado ese NUEVO nodo).

  Entonces, cuando el método **AGREGA** termine y devuelva el enlace **RAIZ** lo estará devolviendo intacto (lo cual es correcto pues el nuevo nodo se agregó más adentro del árbol, no en la raíz)

Entonces, escribiendo esto en Java tendremos

```
public void agregar(Nodo nuevo) {
   raiz = ragrega(raiz, nuevo);
}

private Nodo ragrega(Nodo r, Nodo n) {
   if(r==null) {
      r = n;
   }
   else {
      float x = Math.random();
      if(x<0.5) r.izq = ragrega(r.izq, n);
      else      r.der = ragrega(r.der, n);
   }
   return r;
}
```

Método agregar recibe enlace a nodo nuevo
raiz recibirá lo devuelto por el método ragrega recursivo que recibe raiz y nuevo

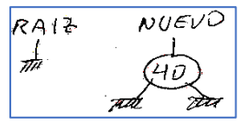
Método recibe esos enlaces como **r** y **n**
Si **r** es nulo lo hace igual a **n** (entonces ahora **r** apuntará al nuevo nodo)

pero si **r** no es nulo…
genera un valor al azar entre 0.0 y 0.9999999999
sí menor que 0.5 busca lugar libre hacia izquierda
de lo contrario busca hacia la derecha

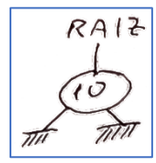
regresa con misma raíz recibida (o modificada)

(el uso de esa variable **x** es un "truco" para evitar que muchos nodos queden para el mismo lado)

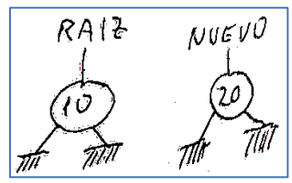
UN INTENTO DE EXPLICAR ESTO CON MAS DETALLE
Al comenzar se tiene un enlace **raíz nulo** (aún no hay nodo en el árbol) y un enlace **nuevo** que apunta a un nuevo nodo que se desea agregar al árbol. Para lograr eso se realiza un llamado al método **ragegra** pasándole los enlaces **raiz** y **nuevo**. La respuesta de ese método será asignada al enlace **raiz**



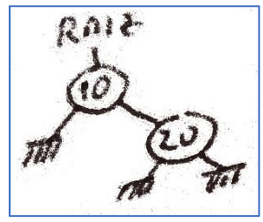
Entonces en método **ragrega** el enlace **r** recibirá un valor nulo y se ejecutará **r = n** de modo que **r** ya no será nulo, sino que apuntará a nodo apuntado por enlace **n**, es decir al nuevo nodo. Al regresar al método **agrega** ese enlace lo recibirá **raíz**, de modo que quedará apuntando al nuevo nodo… se ha logrado agregar ese nuevo nodo al árbol



Ahora se desea agregar un segundo nodo, apuntado por el enlace **nuevo**. Para lograr eso se realiza un llamado al método **ragegra** pasándole los enlaces **raiz** y **nuevo**. La respuesta de ese método **no cambiará al** enlace **raiz** pues devolverá ese mismo enlace ya que el nuevo nodo no puede ser colocado en ese lugar, en el cual ya existe un nodo (el de valor 10)



Como en **ragrega** ahora **r** no es nulo se ejecutará una de las otras dos instrucciones según valor de **x**. Supongamos que es **r.der = ragrega(r.der, n)**. Entonces enlace **der** de este nodo **r** recibirá la respuesta del método **ragrega**. A esta nueva instancia del método **ragrega** se le pasa ese enlace **der** (que es nulo) y el enlace **n** que apunta al nuevo nodo de modo que la respuesta será ese enlace **nuevo** nodo y el nuevo nodo quedará conectado a la derecha de este nodo



Si intento seguir explicando te confundirás, demasiadas palabras para algo que sería fácil entender si se tiene clara la forma como funciona la recursividad.

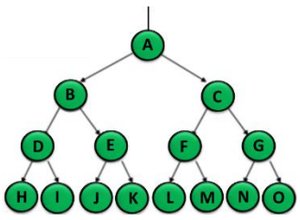
Pero igual lee hasta comprenderlo (ten en cuenta que es recursivo). También acude a alguno de los números libros que existen respecto del tema

FORMAS DE LEER LOS NODOS AL RECORRER EL ARBOL

A cualquier nodo del árbol se llega mediante un enlace entonces, como todo nodo es también el comienzo de un árbol (formado por él y todos los que siguen de él) se dice que ese nodo es RAIZ (obviamente no es la raíz inicial, en donde comienza el árbol, sino que es la raíz del árbol que comienza en él)

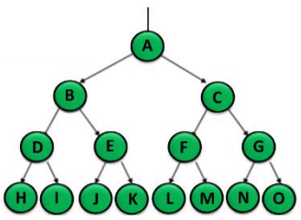
Para este árbol los recorridos posibles son…

RID: Se lee primero la información que hay en él nodo raíz (es decir en el nodo al que se ha llegado), luego se sigue a la izquierda de este nodo para recorrer y leer todos los nodos que hay para ese lado. Cuando se regrese desde ese recorrido seguirá a la derecha de este nodo, para recorrer y leer todos lo que hay para ese lado… entonces regresará al nodo anterior.



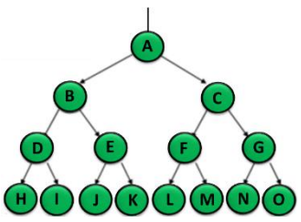
Para el árbol mostrado será: A – B – D – H – I – E – J – K – C – F – L – M – G – N – O

IRD No se lee la información del nodo al que se ha llegado, sino que se sigue hacia la izquierda para recorrer y leer todos los que existan para ese lado. Cuando regrese de ese recorrido leerá la información de este nodo y luego seguirá hacia la derecha para recorrer y leer todos los nodos que hay para ese lado.



Para el árbol mostrado será: H – D – I – B – J – E – K – A – L – F – M – C – N – G – O

IDR No se lee la información del nodo al que se ha llegado, sino que se sigue hacia la izquierda para recorrer y leer todos los que existan para ese lado. Cuando regrese de ese recorrido seguirá hacia la derecha para recorrer y leer todos los nodos que hay para ese lado. Recién cuando regrese de este segundo recorrido leerá este nodo



Para el árbol mostrado será: H – I – D – J – K – B – L – M – F – N – O – G – C – R

Ejercicio individual:

En la clase de este viernes 26, a la cual no debes faltar, me entregas una hoja con un dibujo a mano del árbol binario que tiene los recorridos indicados más abajo, incluyendo una explicación manuscrita del razonamiento que te permitió determinar que esa es la forma del árbol.

Para un cierto árbol binario los recorridos IRD y RID son…

IRD = 80 – 30 – 60 – 90 – 20 – 10 – 40 – 50 – 70
RDI = 20 – 60 – 80 – 30 – 90 – 40 – 10 – 70 – 50

¡no te confundas!, No son dos árboles, son dos recorridos de un único árbol.

Si te hubiese proporcionado un solo recorrido podrías encontrar muchos árboles que tendrían ese recorrido. Por eso he proporcione dos, entonces se espera que “a punta de razonamiento” logres determinar cual debería ser a forma del árbol que cumple con ambos recorridos

Recuerda que, según el plan de actividades para el resto del semestre que publiqué, este próximo lunes no debes faltar pues deberás desarrollar el primer ejercicio de la segunda prueba del semestre