UNIVERSIDAD MAYOR DE SAN "ANDRÉS"
CARRERA DE INFORMÁTICA
PROGRAMACIÓN II
ÁRBOLES

Responsable: Lic. Victoria Hurtado Cerruto Gestion:2/2024

1 INTRODUCCIÓN

Antes de introducirnos al tratamiento de árboles, recordemos que se ha encarado el tratamiento de estructuras de datos lineales que se caracterizan porque a cada elemento le sucede otro elemento. Sin embargo, existe otro tipo de estructuras de datos denominados no lineales y se caracterizan porque a cada elemento no sólo le puede suceder un elemento sino más de un elemento entre los que se tienen a los árboles y grafos entre otros.

A continuación, se proporciona conceptos fundamentales sobre árboles posteriormente se describen su representación gráfica y terminología asociada. Finalmente, debido a su practicidad y simplicidad de tratamiento se pone más énfasis en el tratamiento de Árboles Binarios.

2 DEFINICIÓN

Existen varias definiciones de árboles, sin embargo, por su claridad y expresividad se rescatan las siguientes:

Según Sisa [9], un árbol es un *grafo dirigido, unidireccional, conexo, sin ciclos* tal que:

- Existe un nodo único llamado raíz.
- Entre los otros nodos cada uno tiene un arco único que entra en dicho nodo.
- Existe un camino único para ir a cualquier nodo del árbol.

Asimismo, en [9] se hace notar que la *recursividad* es una característica fundamental de los árboles razón por la que se proporciona una definición de árboles considerando este concepto que dice:

"Un árbol es un conjunto finito de uno o más nodos tal que existe un nodo especial llamado ***raíz*** y los demás nodos forman conjuntos disjuntos que a su vez son árboles. Esta definición expresa la aplicación de la *recursividad* en este tipo de estructura".

Según Cairo [11], un árbol es una estructura de datos *no lineal* y *dinámica*. No lineal porque a cada elemento le pueden suceder varios elementos y dinámica porque durante la ejecución del programa la estructura del árbol puede cambiar.

Adicionalmente en [11], se proporcionan otras definiciones adicionales:

- "Un árbol es una estructura *jerárquica* aplicada sobre una colección de elementos y objetos llamados nodos; uno de los cuales es conocido como raíz. Además, se crea una relación de parentesco entre los nodos dando lugar a términos como padre, hijo, hermano, antecesor, sucesor, etc..".

- "Se define un árbol de tipo T como una estructura homogénea que es la concatenación de un elemento de tipo T junto con un número finito de árboles disjuntos llamados subárboles. Una forma particular de árbol es la estructura vacía."

UNIVERSIDAD MAYOR DE SAN "ANDRÉS"
CARRERA DE INFORMÁTICA
PROGRAMACIÓN II
ÁRBOLES

Responsable: Lic. Victoria Hurtado Cerruto Gestion:2/2024

3 REPRESENTACIÓN GRÁFICA DE UN ÁRBOL

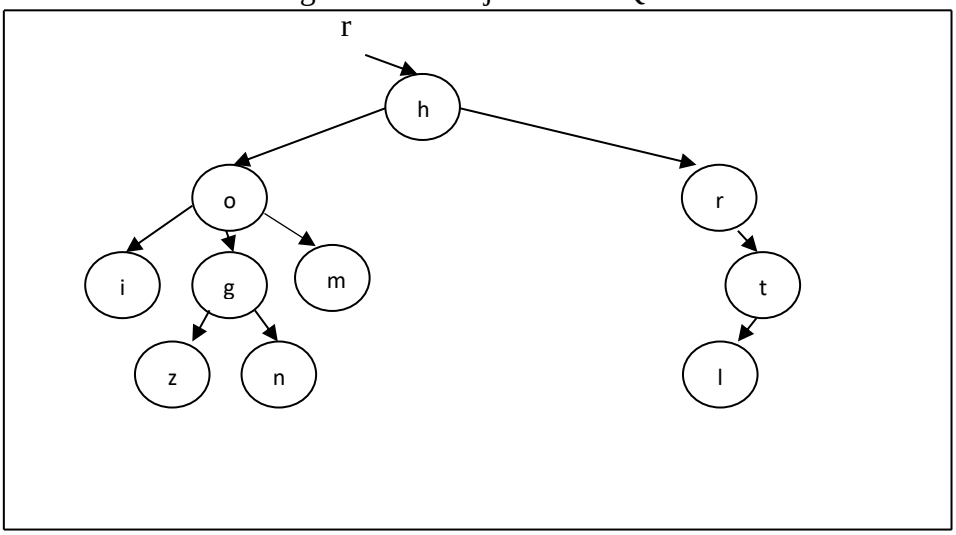
Representaremos los arboles como se cómo muestra en la Figura No. 1. Donde se puede observar que el nodo raíz ***r*** contiene al dato ***h***. Asimismo el dato ***b*** está almacenado en un nodo hoja o también denominado nodo terminal. Esta forma de representación es la más utilizada donde además se puede observar que la raíz se dibuja arriba en contraste con un árbol real cuya raíz se encuentra abajo.

4 CARACTERÍSTICAS DE LOS ÁRBOLES

A continuación, se proporcionan características y propiedades de los árboles en general:

- Existe un nodo único denominado nodo ***raíz***, a partir del cual se define al árbol.
- Cada nodo del árbol tiene un arco único que entra en dicho nodo.
- Existe un camino único para ir a cualquier nodo del árbol.

Figura No. 1 Objeto Árbol Q



TERMINOLOGÍA ASOCIADA AL TRATAMIENTO DEÁRBOLES

Nodo raíz. Nodo único que define al árbol.

***Nodo terminal*.** Es un nodo que no tiene descendientes. También es conocido como nodo hoja.

***Nodo descendiente directo de otro*.** Se dice que un nodo X es descendiente directo de un nodo Y si el nodo X es apuntado por el nodo Y. Por otro lado, también se suele utilizar la siguiente expresión X es hijo de Y.

***Antecesor directo*.** Se dice que un nodo X es antecesor directo de un nodo Y, si el nodo X apunta al nodo Y, en este caso se suele utilizar la expresión X es el padre de Y.

Nodos hermanos. Se dice que todos los nodos que son descendientes directos de un mismo nodo (padre) son nodos hermanos.

Nodo interior. Es todo nodo que no es raíz ni hoja.

Subárboles. También denominados ***ramas*** son árboles que se desprenden de un nodo.

Grado de un nodo. Es igual al número de descendientes directos de un nodo.

Grado de un árbol. Es el máximo valor encontrado entre los grados de cada uno de los nodos del árbol.

Nivel de un nodo. Es el número de arcos que se recorren hasta llegar al nodo. En el presente texto por convención asumiremos que el *nivel del nodo raíz* es 1.

Nivel de un árbol. También denominado **altura o profundidad de un árbol**, es el máximo nivel encontrado entre los niveles de todos los nodos del árbol.

Peso de un árbol. Es el número de nodos terminales.

Bosque. Conjunto de árboles.

Recorrido. Es la suma de las distancias, medidas en arcos de la raíz a cada uno de los nodos.

Ejemplo

Considerando el árbol de la Figura No. 1, identificar:

a) Nodo raíz : h
b) Nodos hojas: i,z,n,m,l
c) Descendiente directo: n es descendiente directo de g.
d) g,m son hermanos.
e) Rama: g,z,n
f) Grado de g es 2
g) Grado(Q)=max{Grado(h),Grado(o),Grado(r),Grado(i),Grado(g),Grado(m),Grado(t), Grado(z),Grado(n),Grado(l)}=max{2,3,1,0,2,0,1,0,0,0}=3

h) Altura del árbol (Q) es 4

6 TIPOS DE ÁRBOLES

Según [9] existen varios tipos de árboles entre los que se tienen:

UNIVERSIDAD MAYOR DE SAN "ANDRÉS"
CARRERA DE INFORMÁTICA
PROGRAMACIÓN II
ÁRBOLES

Responsable: Lic. Victoria Hurtado Cerruto Gestion:2/2024

Árboles n-arios

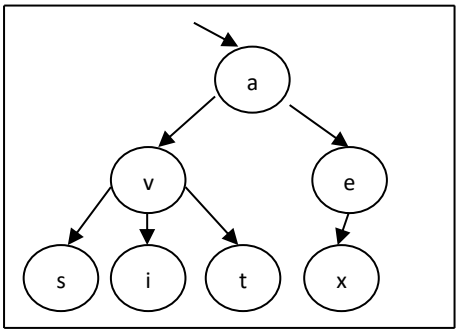
Son árboles donde uno o más nodos del árbol tienen como máximo *n-descendientes directos*. Por ejemplo: árboles binarios, árboles ternarios, árboles cuaternarios, etc.

Árbol ternario, árbol donde uno o más de sus nodos tienen como máximo tres descendientes directos (ver Figura No. 2).

Ejemplo

El árbol de la Figura No. 2 tiene el nodo que contiene al dato **v** cuyo máximo número de descendientes directos es tres por tanto el árbol es ternario.

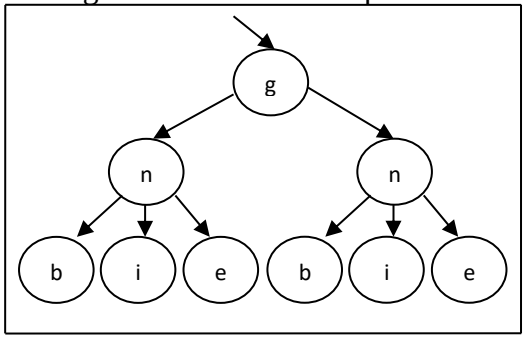
Figura No. 2 Árbol ternario



Árboles completos

Son árboles que se caracterizan porque todos sus nodos terminales tienen la misma altura (ver Figura No. 3).

Figura No. 3 Árbol completo



7ÁRBOLES BINARIOS

Se caracterizan porque cada nodo puede tener como máximo dos descendientes directos que son denominados *subárbol izquierdo* (**si**) y *subárbol derecho* (**sd**) tal como muestra la Figura No. 4.

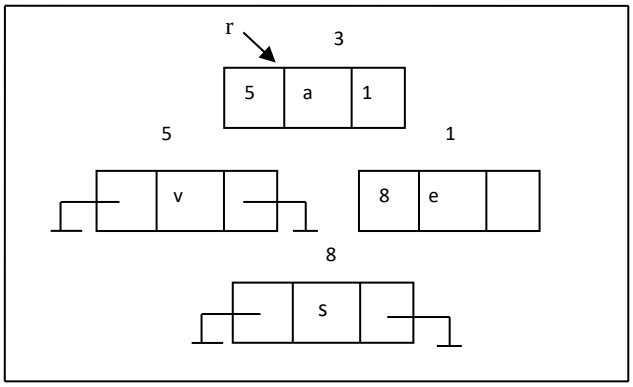
UNIVERSIDAD MAYOR DE SAN "ANDRÉS"
CARRERA DE INFORMÁTICA
PROGRAMACIÓN II
ÁRBOLES

Responsable: Lic. Victoria Hurtado Cerruto Gestion:2/2024

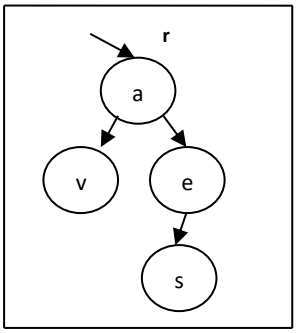
7.1 REPRESENTACIÓN GRÁFICA DE UN ÁRBOL BINARIO

Existen varias formas de representar a árboles binarios, entre las que se pueden destacar las que se muestran la Figura No. 4. En el formato a) se utilizan referencias de memoria que permiten enlazar los diferentes nodos que conforman el árbol. En tanto que, en el formato b) los nodos se relacionan entre sí a través de arcos unidireccionales.

Figura No.4Formatos de representación de un árbol binario



a)



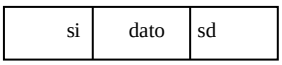
b)

Por tanto, considerando el formato de representación del inciso a) a continuación se describe la representación gráfica de los componentes de un árbol binario.

7.2REPRESENTACIÓN GRÁFICA DE UN NODO (NodoAr)

Considerando la representación de la Figura No. 4inciso a) de árboles binarios a continuación describiremos a sus elementos (ver Figura No. 5)

Figura No. 5 Nodo de un árbol binario

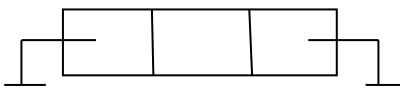


Donde:
si, referencia al subárbol izquierdo.
sd, referencia al subárbol derecho.
dato, dato o información que almacena el árbol.

7.3 ESTADO INICIAL DE UN NODO ÁRBOL BINARIO (NodoAr)

Por convención asumiremos el estado inicial de un nodo del árbol como muestra el nodo de la Figura No. 6.

Figura No. 6 Estado inicial

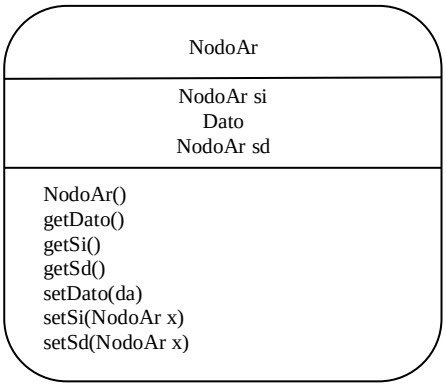


Es decir, donde la referencia al subárbol izquierdo ***si*** se inicializa a ***null*** (nil) y el subárbol derecho ***sd*** también se inicializa a ***null*** (nil)

7.5 DIAGRAMA DE CLASE DEL TDA NodoAr

Considerando la estructura de un nodo de un árbol binario a continuación la Figura No. 7 muestra el diagrama de clase respectivo.

Figura No. 7 Diagrama de clase del TDA ***NodoAr***



7.6 IMPLEMENTACIÓN DEL TDA NodoAr

```
public class NodoAr {
private NodoAr si;
private Object dato;
private NodoAr sd;
NodoAr()         //inicializaunnododelárbol
{
      si=null;sd=null;
}
public NodoAr getSi() {
      returnsi;
}
public void setSi(NodoAr si) {
      this.si = si;
}
public Object getDato() {
      returndato;
}
public void setDato(Object dato) {
      this.dato = dato;
}
public NodoAr getSd() {
      returnsd;
}
public void setSd(NodoAr sd) {
      this.sd = sd;
}
}
```

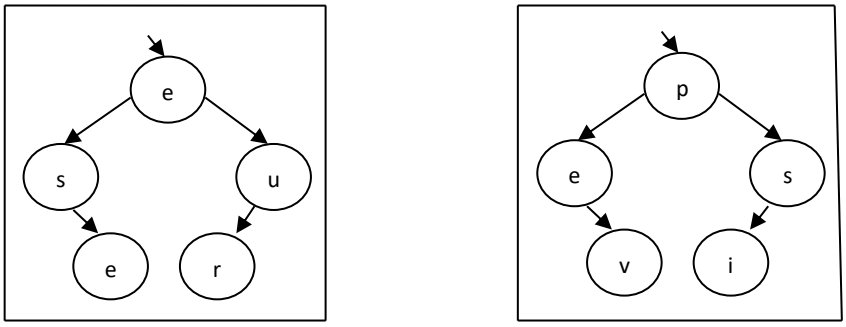
7.7 TIPOS DE ÁRBOLES BINARIOS

Según su estructura y contenido se identifican arboles binarios similares, equivalentes y distintos.

7.7.1ÁRBOLES BINARIOS SIMILARES

Dos árboles binarios son similares cuando tienen la misma estructura, pero diferente contenido en sus nodos (ver Figura No. 8).

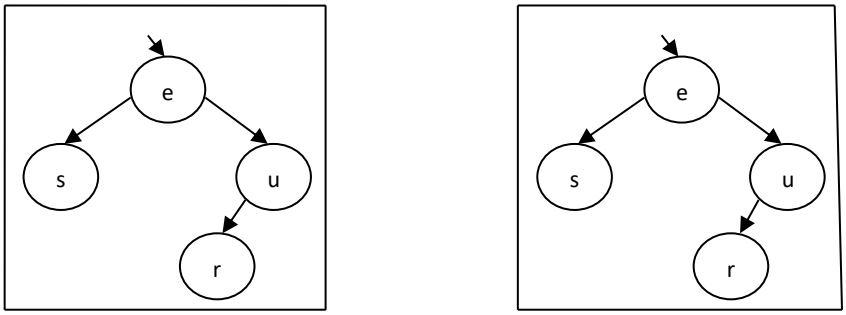
Figura No. 8Árbolesbinarios similares



7.7.2 ÁRBOLES BINARIOS EQUIVALENTES

Dos árboles binarios son equivalentes cuando tienen la misma estructura y el mismo contenido en sus nodos (ver Figura No. 9).

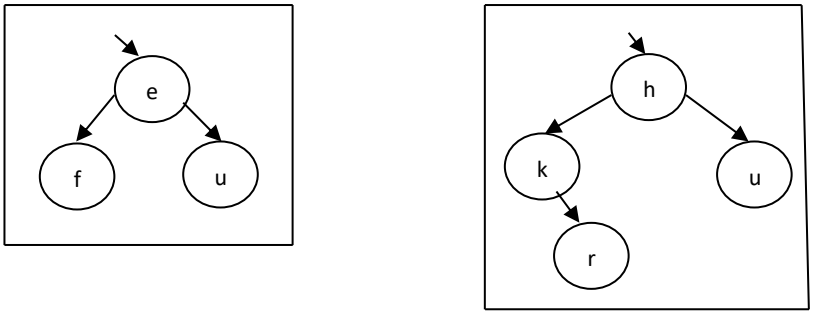
Figura No. 9Árboles binarios equivalentes



7.7.3 ÁRBOLES BINARIOS DISTINTOS

Dos árboles binarios son distintos si tienen diferentes estructuras (ver Figura No. 10).

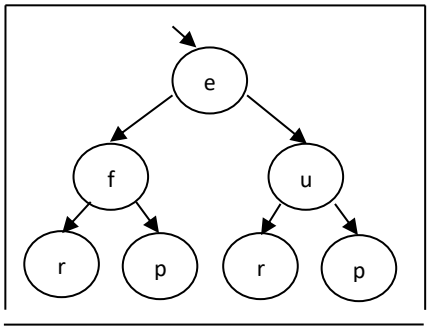
Figura No. 10Árboles binarios distintos



7.8ÁRBOL BINARIO COMPLETO

Es aquel árbol binario en el que todos sus nodos tienen dos descendientes directos excepto los nodos del ultimo nivel (ver Figura No. 11).

Figura No. 11Árbol binario completo Q



7.8.1PROPIEDADES DE ÁRBOLES BINARIOS COMPLETOS

Según [9] los árboles binarios completos tienen algunas propiedades importantes entre las que se tienen:

- Un árbol binario completo con N niveles tiene en total 2^N-1 nodos.
- Un árbol binario completo tiene 2^{N-1} nodos terminales.

Por ejemplo, considerando el árbol de la Figura No. 11 que tiene 3 niveles (N=3) setiene:

- Un total de $2^3-1=7$ nodos en el árbol.
- La cantidad de nodos terminales en el árbol: $2^{3-1}=2^2=4$ nodos terminales

Por su importancia es oportuno rescatar de [9]el siguiente fragmento de texto que dice:" *los árboles binarios son los más utilizados y debido a ello han sido la base de gran parte del tratamiento teórico y desarrollo algorítmico para esta estructura; de otra parte otras clases de árboles se pueden transformar en árboles binarios"*. En tal sentido a continuación, se describe un procedimiento que permite convertir un árbol general en un árbol binario.

7.9 CONVERSIÓN DE UN ÁRBOL GENERAL EN ÁRBOL BINARIO

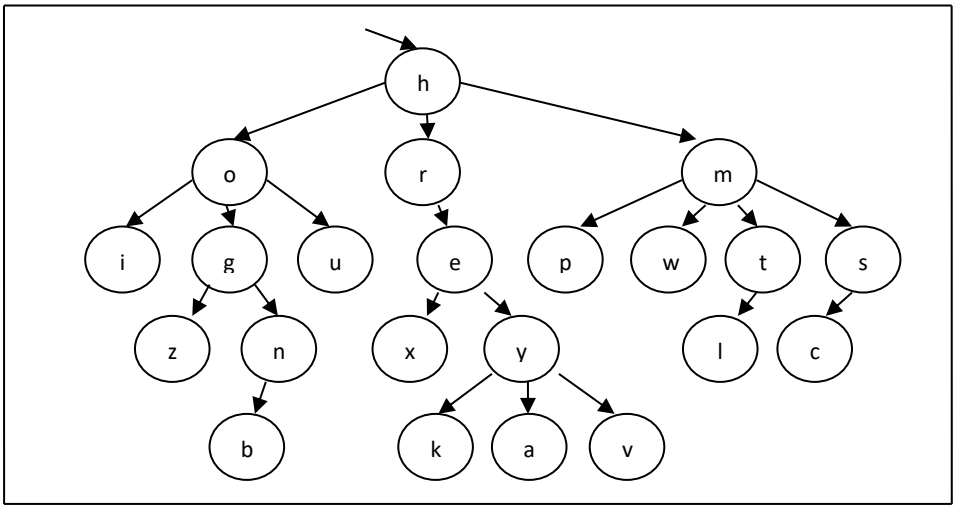
Según Cairo [10] para convertir un árbol general (*n-ario*) en un árbol binario se aplicarán los siguientes pasos:

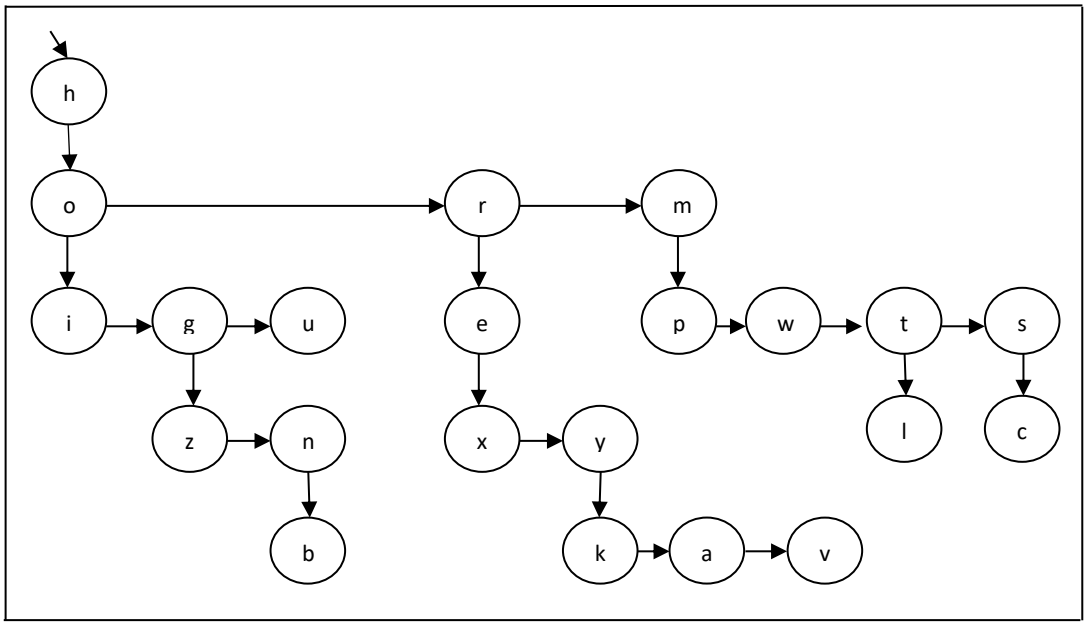
Paso 1. Enlazar en forma vertical el nodo padre con el hijo de más a la izquierda.
Paso 2. Enlazar en forma horizontal los nodos hermanos
Paso 3. Girar 45º el árbol generado, en el sentido de las agujas del reloj

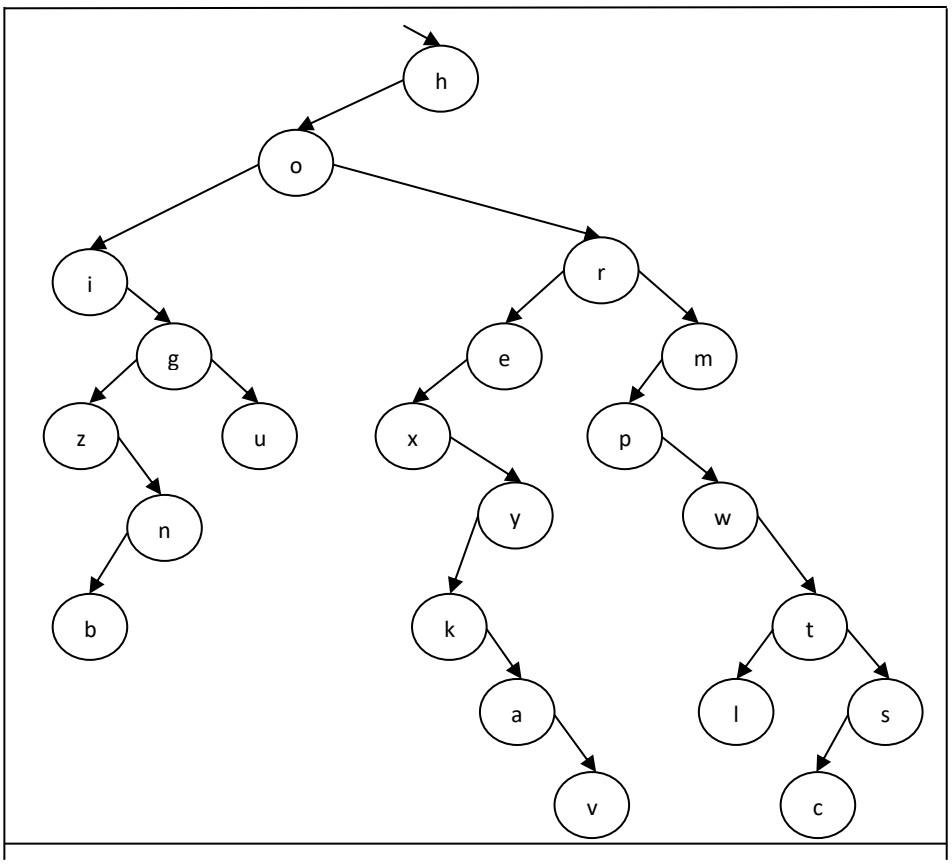
Ejemplo

Convertir el árbol cuaternario (ver Figura No. 12) en árbol binario.

Figura No. 12 Objeto Árbol







7.10 CONVERSIÓN DE UN BOSQUE EN UN ÁRBOL BINARIO

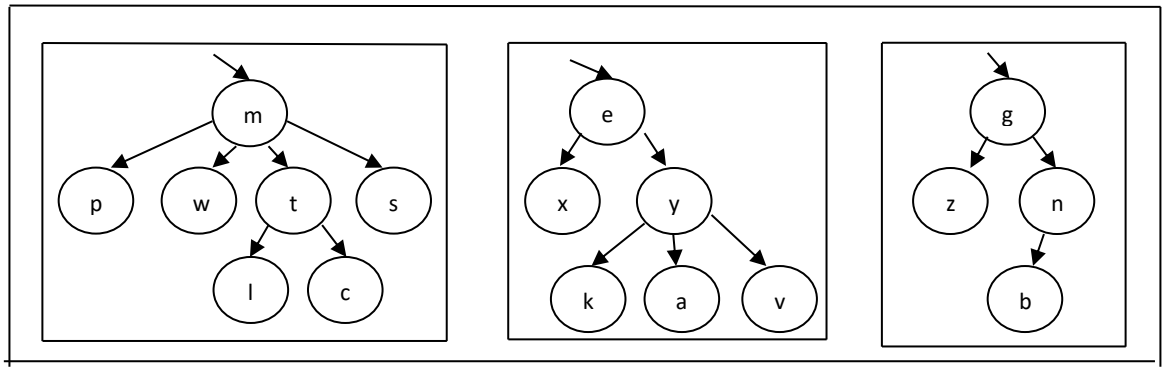
Para convertir un bosque en un árbol binario aplicar los siguientes pasos:

Paso 1. Enlazar horizontalmente las raíces de todos los árboles generales.
Paso 2. Enlazar en forma vertical el nodo padre con el hijo de más a la izquierda.
Paso 3. Enlazar en forma horizontal los nodos hermanos
Paso 4. Girar 45º el árbol generado, en el sentido de las agujas del reloj

Ejemplo

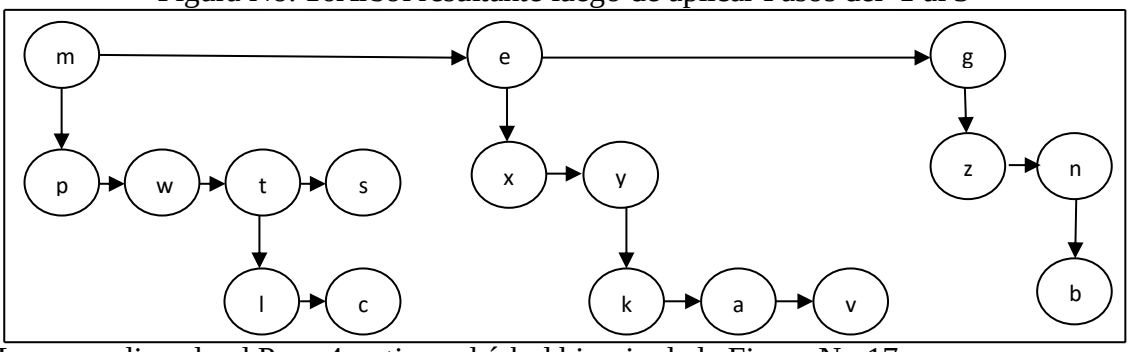
Dado el Bosque de la Figura No. 15 se pide generar un árbol binario.

Figura No. 15 Bosque



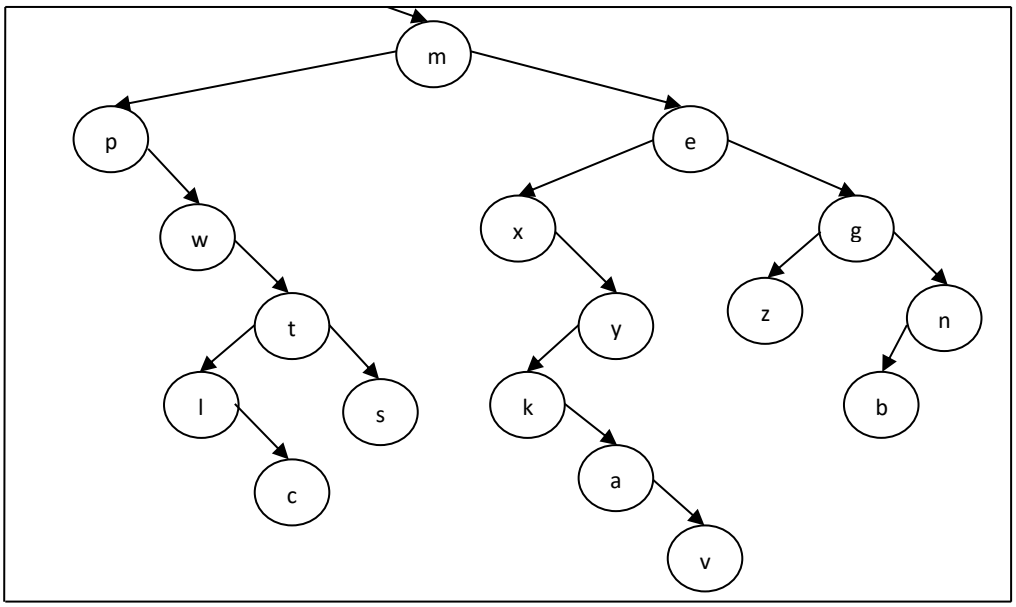
Aplicando tanto el Paso 1, Paso 2 y Paso 3 se genera el árbol de la Figura No. 16

Figura No. 16Árbol resultante luego de aplicar Pasos del 1 al 3



Luego, aplicando el Paso 4 se tiene el árbol binario de la Figura No.17

Figura No. 17Árbol binario resultante



7.11 RECORRIDOS NO RECURSIVOS EN UN ÁRBOL BINARIO

Una de las operaciones más utilizadas en un árbol binario son los recorridos que permiten básicamente visitar a los nodos del árbol una vez, que se describen a continuación.

7.11.1RECORRIDOS EN ANCHURA

Este tipo de recorrido también es conocido como *recorrido por niveles* y consiste en recorrer los nodos de manera descendente a partir de la raíz y de izquierda a derecha [11]. Como se puede constatar en el método *m_Nivel()* donde se utilizan dos pilas una que procesa a los niveles en la variable ***nivel*** y la otra pila en la variable ***su*** que almacena a los sucesores de los niveles según el nivel que se procesa.

Posteriormente, se debe garantizar que una vez que todos los nodos de un nivel son procesados (es decir la pila ***nivel*** está vacía) los nodos sucesores que están almacenados

en la pila ***su*** se convierten en nodos de nivel para ello se transfieren a la pila *nivel*las referencias de los nodos sucesores que están almacenados en la pila ***su***.

```
void m_Nivel()              //Mostrarelementospornivel
{
  Pila nivel=new Pila(100);
Pila su=new Pila(100);intn;
NodoAr x,w;intres;
nivel.adicionar(r);

while(!nivel.esVacia())
  {
while(!nivel.esVacia())
{
  x=(NodoAr)nivel.eliminar();
System.out.print(x.getDato()+"\t");
if (x.getSi()!=null) su.adicionar(x.getSi());
if (x.getSd()!=null) su.adicionar(x.getSd());
}
  nivel.vaciar(su);
  System.out.println();
   }
 }
```

Por otro lado, se afirma en varios textos entre los que podemos citar a [7] que el recorrido por niveles implementa una técnica de búsqueda denominada *búsqueda por anchura.*

7.11.2 RECORRIDOS EN PROFUNDIDAD Se basan en las relaciones padre-hijo de los nodos y entre los que se distinguen:

7.11.2.1RECORRIDO EN PREORDEN (RID)

Consiste en realizar las siguientes tareas:

- Visitar la raíz
- Recorrer el subárbol izquierdo
- Recorrer el subárbol derecho

El método *m_Preorden()* implementa este recorrido donde partiendo desde la raíz se utiliza una pila auxiliar ***p*** que rescata las referencias a los subárboles derechos siempre que existan, luego de que para cada nodo ***x*** se obtengan sus referencias tanto del subárbol izquierdo ***ni*** como del subárbol derecho ***nd***. En caso de que el nodo actual ***x*** tenga subárbol derecho su referencia se guarda en la pila ***p*** en tanto que si el nodo actual tiene un subárbol izquierdo dicho subárbol debe ser procesado (haciendo que ***x***=***ni***), caso contrario significa que subárbol izquierdo ya fue procesado y debe procesarse el lado derecho del subárbol para lo cual se extrae de la pila dicha referencia volviéndose a repetir el proceso hasta que la pila quede vacía. Con el objeto de evitar desbordamientos inicialmente la pila ***p*** debe almacenar el valor ***null*** como se observa en el siguiente código.

```
void m_Preorden()
{
  NodoAr x,nd,ni;Pila p=new Pila(100);
 p.adicionar(null);  x=r;
while(x!=null)
  {
   System.out.print(x.getDato()+ " ");
   nd=x.getSd();ni=x.getSi();
if (nd!=null) p.adicionar(nd);
if (ni!=null)x=ni;
else x=(NodoAr)p.eliminar();
}
 }
```

7.11.2.2 RECORRIDO ENORDEN (IRD)

Consiste en realizar las siguientes tareas:

- Recorrer el subárbol izquierdo
- Visitar la raíz
- Recorrer el subárbol derecho

En este caso el método *m_Enorden()*, parte de la raíz a la vez se almacena en la pila *p* el valor ***null*** de modo que se pueda evitar inconsistencia del método. Posteriormente mientras en la pila se tengan nodos por procesar se insertan en la pila los lados izquierdos de cada nodo y se avanza hasta llegar al nodo de más a la izquierda luego se extrae de la pila el último nodo del cual se despliega su contenido posteriormente se procesa el subárbol derecho volviéndose a recorrer hasta el nodo de más a la izquierda y así sucesivamente.

```
void m_Enorden()  //mostrardatosdelárbolenorden IRD
{
 Pila p=new Pila(100); NodoAr x=r;P.adicionar(null);
while(p.nElem()!=0)
{
while(x!=null)
{
     p.adicionar(x);
     x=x.getSi();
}
 x=(NodoAr)p.eliminar();
if(x!=null){
               System.out.print(x.getDato()+ " ");
               x=x.getSd();
              }
  }
}
```

7.11.2.3 RECORRIDO EN POSORDEN (IDR)

Consiste en realizar las siguientes tareas:

- Recorrer el subárbol izquierdo.

- Recorrer el subárbol derecho
- Visitar la raíz

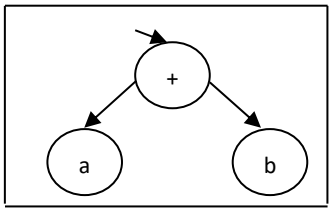
En este caso el método *m_Posorden()* implementa el procedimiento especificado para ello, se inicializa la pila *p* con el valor *null* de modo que el método no tenga inconsistencia al momento de su invocación. Asimismo partiendo desde la raíz ***r*** se insertan todos los nodos izquierdos en la pila ***p*** asimismo si dichos nodos tienen descendientes derechos***(nd!=null)*** estos también son introducidos en la pila seguidos del valor *null* que lo delimita. Una vez que el último nodo de más a la izquierda ***x*** es encontrado y agregado a la pila ***p*** se extrae de la misma y se procesa su árbol derecho si existe además del nodo ***x***, repitiéndose el proceso en el sentido de encontrar el nodo de más a la izquierda y nuevamente procesar el subárbol derecho si existe hasta encontrar el último nodo de más a la derecha posteriormente procesándose la subraiz (raíz) todo este proceso se repite hasta que la pila ***p*** quede vacía como muestra el siguiente código.

```
void m_Posorden()
{
  NodoAr nd,x;Pila p=new Pila(100);
p.adicionar(null);  x=r;
while(p.nElem()!=0)
{
 while(x!=null)
 {
       p.adicionar(x);nd=x.getSd();
      if(nd!=null){
       p.adicionar(nd);
       p.adicionar(null);
      }
       x=x.getSi();
}
  x=(NodoAr)p.eliminar();
while (x!=null)
    {
       System.out.print(x.getDato()+" ");
       x=(NodoAr)p.eliminar();
    }
if (p.nElem()!=0)x=(NodoAr)p.eliminar();
}
}
```

Ejemplo

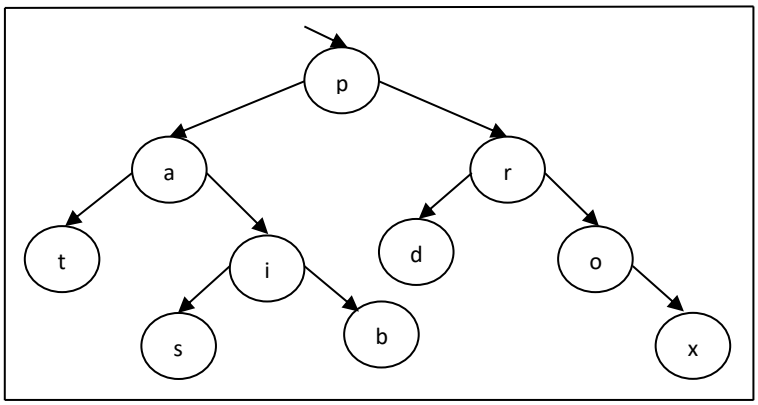
Dados los árboles binarios de las Figura No. 18, Figura No. 19 y Figura No. 20 recorrer dichos árboles en preorden, enorden y posorden respectivamente.

Figura No. 18Árbol binario



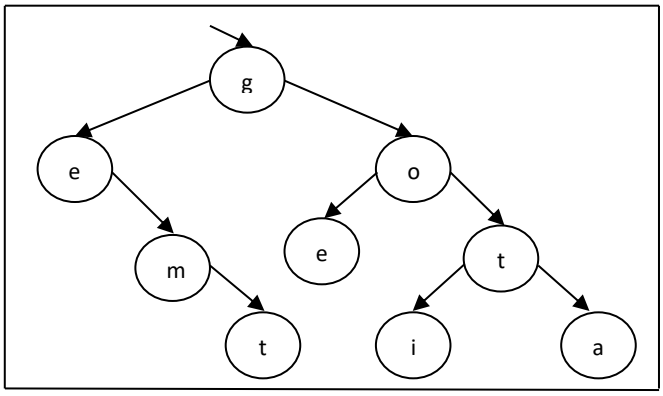
Recorrido preorden (RID): +ab
Recorrido enorden (IRD) : a+b
Recorrido posorden (IDR): ab+

Figura No. 19Árbol binario



Preorden(RID):patisbrdox
Enorden(IRD):tasibpdrox
Posorden(IDR):tsbiadxorp

Figura No. 20Árbol binario



Preorden(RID):gemtoetia
Enorden(IRD):emtgeoita
Posorden(IDR)=tmeeiatog

7.12 RECORRIDOS RECURSIVOS EN UN ÁRBOL BINARIO

Los recorridos de un árbol binario también pueden ser expresados recursivamente.

7.12.1RECORRIDO EN PREORDEN (RID), como se mencionó consiste en realizar las siguientes tareas:

- Visitar la raíz
- Recorrer el subárbol izquierdo
- Recorrer el subárbol derecho

```
void m_PreordenRecu(NodoAr y)
{
if(y!=null){
              System.out.print(y.getDato()+"\t");
              m_PreordenRecu(y.getSi());
              m_PreordenRecu(y.getSd());
              }
 }
```

7.12.2RECORRIDO EN ENORDEN (IRD), como se mencionó **c**onsiste en realizar las siguientes tareas:

- Recorrer el subárbol izquierdo
- Visitar la raíz
- Recorrer el subárbol derecho

```
voidm_EnordenRecu(NodoAr y)
{
if(y!=null){ m_EnordenRecu(y.getSi());
               System.out.print(y.getDato()+"\t");
               m_EnordenRecu(y.getSd());
            }
}
```

7.12.3 RECORRIDO EN POSORDEN (IDR), como se mencionó **c**onsiste en realizar las siguientes tareas:

- Recorrer el subárbol izquierdo
- Recorrer el subárbol derecho
- Visitar la raíz

```
void m_PosordenRecu(NodoAr y)
{
if(y!=null){
               m_PosordenRecu(y.getSi());
               m_PosordenRecu(y.getSd());
               System.out.print(y.getDato()+"\t");
            }
}
```

7.13 OPERACIONES BÁSICAS EN UN ÁRBOL BINARIO

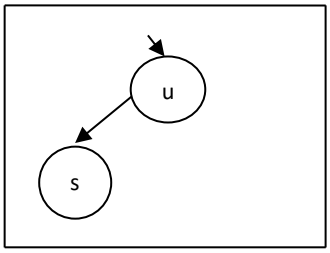
Entre las operaciones básicas se tienen:

7.13.1 INSERTAR UN ELEMENTO, se puede insertar un elemento al árbol considerando varios criterios como ser nodos sucesivos, por niveles, etc.

Ejemplo

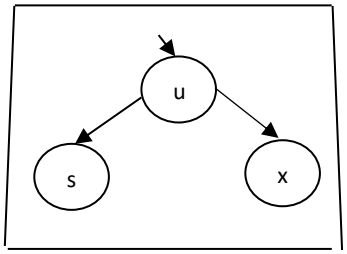
Dado el árbol binario de la Figura No. 21 insertar a la izquierda del dato ***u*** el dato ***x.*** Para ello, en principio ubicamos al nodo que contiene al dato ***u*** posteriormente creamos el nuevo nodo donde se insertamos el dato ***x*** y luego enlazamos este nuevo nodo con el nodo que contiene al dato ***u*** (ver Figura No. 22).

Figura No. 21Árbol binario



luego

Figura No.22Árbol binario con dato 'x'



7.13.2 ELIMINAR UN ELEMENTO. Para eliminar un elemento se puede considerar 3 casos:

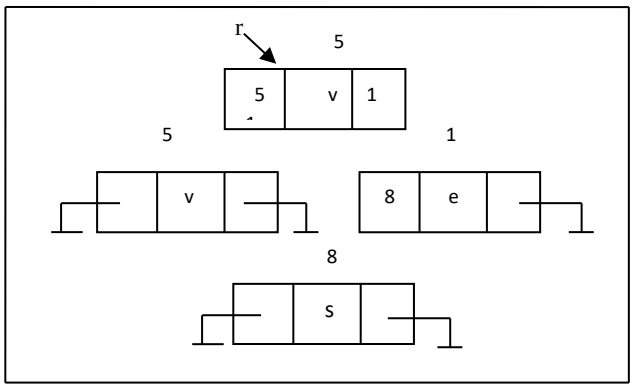
7.13.2.1 **EL DATO A ELIMINAR ESTÁ CONTENIDO EN UN NODO TERMINAL**

En tal caso simplemente se desconecta al nodo del árbol y se elimina el nodo.

Ejemplo

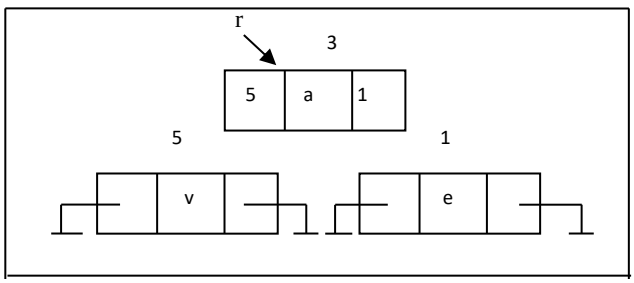
Considerando el árbol binario de la Figura No. 23 eliminar el dato '***s***'.

Figura No.23Árbol binario



En este caso como el dato ***'s'*** está contenido en un nodo terminal además se desprende del lado izquierdo del nodo que almacena al dato ***'e'*** luego entonces se procede a desconectar el nodo haciendo que su enlace tome el valor ***null*** (nil) y luego se elimina el nodo, como se muestra en la Figura No. 24.

Figura No.24Árbol binario

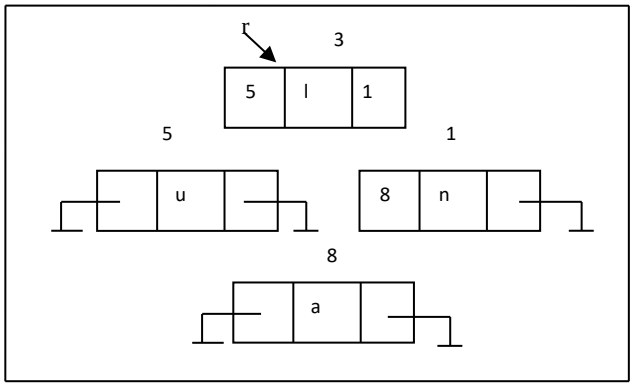


7.13.2.2 EL DATO A ELIMINAR ESTE CONTENIDO EN UN NODO CON UN SOLO DESCENDIENTE DIRECTO. En este caso se reemplaza por ese nodo descendiente directo.

Ejemplo

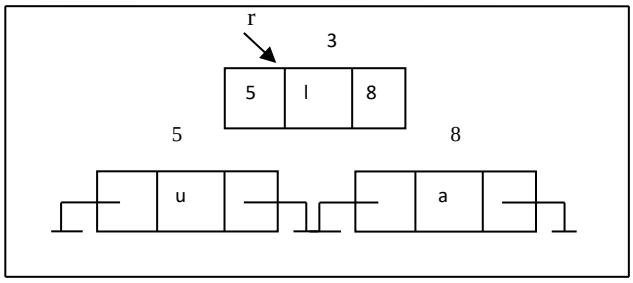
Considerando el árbol binario de la Figura No. 25 eliminar al dato ***'n'***.

Figura No.25Árbol binario



En este caso, el nodo que contiene al dato *'n'* tiene un descendiente directo por tanto al eliminarse este nodo, su descendiente directo toma su lugar, actualizándose las referencias de memoria involucradas como se muestra en la Figura No. 26.

Figura No.26 Árbol binario sin el dato ***'n'***

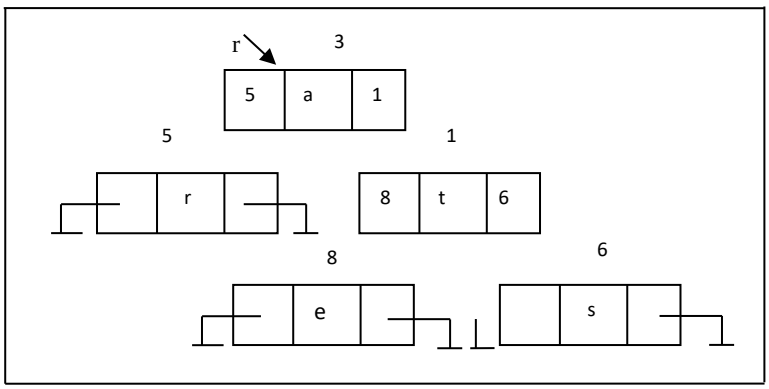


7.13.2.3 **EL DATO A ELIMINAR ESTE CONTENIDO EN UN NODO QUE TIENE DOS DESCENDIENTES DIRECTOS.** En este caso, se reemplaza por cualquiera de los dos siguientes nodos.

Ejemplo

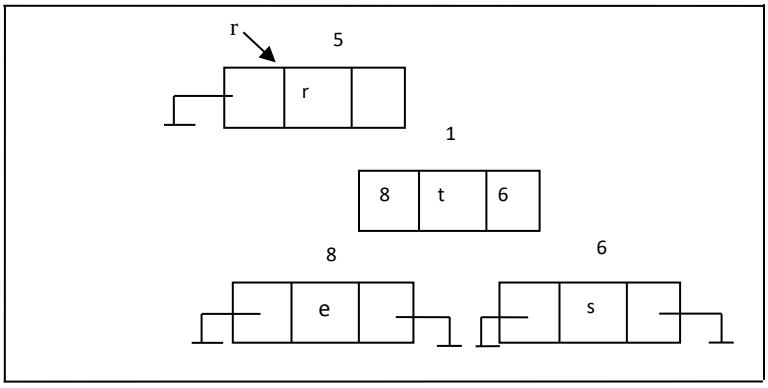
Dado el árbol binario de la Figura No. 27, eliminar el dato **'a'.**

Figura No.27Árbol binario



Por tanto, una solución se muestra en la Figura No. 28, donde se reemplaza al dato **'a'** con el nodo de su izquierda es decir con el nodo que contiene al dato ***'r'***

Figura No.28Árbol binario sin el dato **'a'**

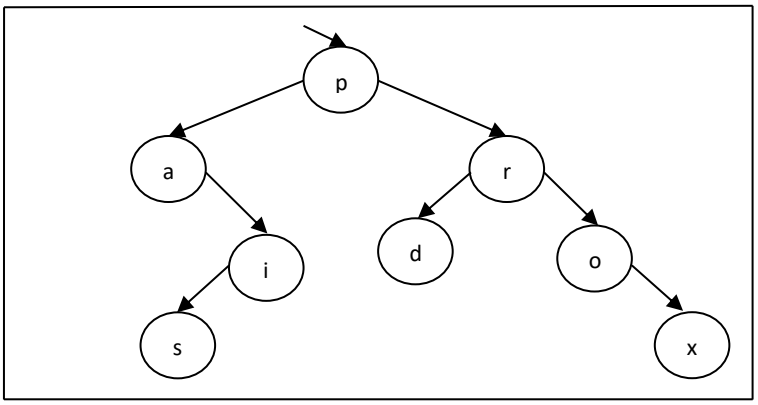


Como se mencionó, los árboles son estructuras de datos que permiten organizar la información jerárquicamente. Sin embargo, en ocasiones puede ser resultar ineficiente en ciertos procesos como los que se describe a continuación.

Ejemplo

Considerando el árbol de la Figura No. 29donde se busca al dato ***'x'*** aplicando los algoritmos por recorridos por anchura efectuaríamos comparaciones con todos los nodos del árbol haciendo que el tiempo de ejecución sea lineal lo cual es el peor caso.

Figura No. 29 Árbol binario



Para mejorar el tratamiento con árboles binarios se define como alternativa a los árboles binarios de búsqueda que se caracterizan porque su contenido refleja orden consistente de modo que van a permitir de alguna manera mejorar los tiempos de ejecución para distintos procesos.

7.14 ÁRBOLES BINARIOS DE BÚSQUEDA (ABB)

En [10] un árbol binario de búsqueda se define como una estructura sobre la cual se pueden realizar eficientemente las operaciones de inserción, búsqueda y eliminación por las características de la información que contiene pues este obedece a ciertos criterios de ordenación que básicamente se ajusta a su definición formal que dice: "Para todo nodo ***T*** del árbol debe cumplirse que todos los valores de los nodos del subárbol izquierdo de ***T*** serán menores o iguales al valor del nodo ***T***. De forma similar, todos los valores de los nodos del subárbol derecho de ***T*** deben ser mayores o iguales al valor del nodo ***T***". En sí, también se afirma que un ABB es un árbol binario que cumple con la propiedad de búsqueda ordenada.

Según Cairo [11] comparando un ABB con otras estructuras como arreglos y listas se afirma que:

- En un arreglo la operación de búsqueda resulta fácil si estos están ordenados, sin embargo, las operaciones de inserción y eliminación no son tan elementales.

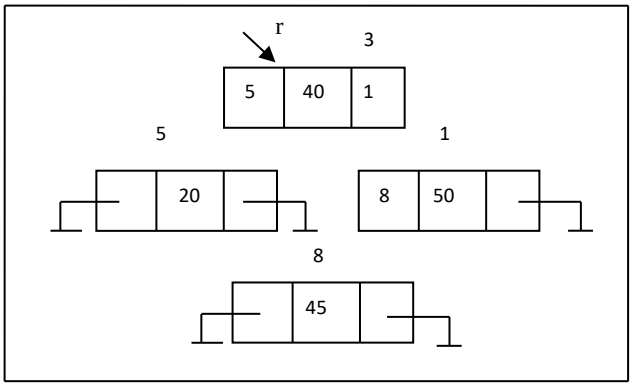
- En tanto que, en listas enlazadas las operaciones de inserción y eliminación se realizan con cierta facilidad sin embargo el proceso de búsqueda resulta costoso pues en el peor caso tendría que recorrerse toda la lista.

Finalmente, hay quienes afirman que un Árbol binario de búsqueda es una estructura de datos sencilla que puede considerarse como una extensión del algoritmo de la búsqueda binaria.

7.15 REPRESENTACIÓN GRÁFICA DE UN ABB

Representaremos los ABB como se muestra en la Figura No. 30 o la Figura No. 31.

Figura No. 30Árbol binario de búsqueda

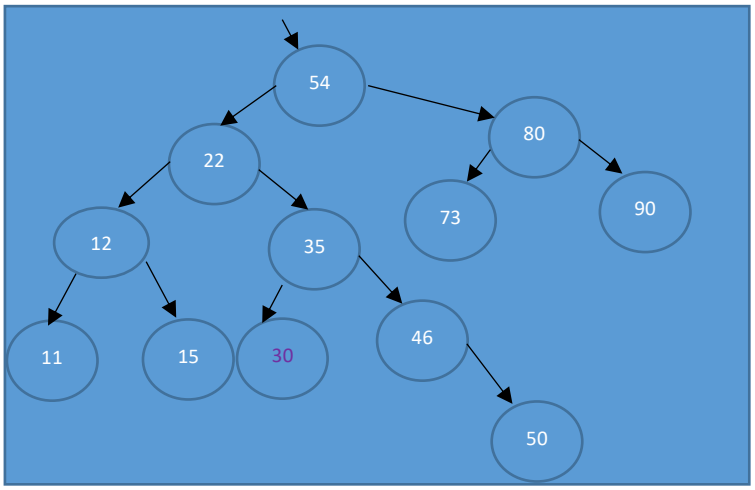


UNIVERSIDAD MAYOR DE SAN "ANDRÉS"
CARRERA DE INFORMÁTICA
PROGRAMACIÓN II
ÁRBOLES

Responsable: Lic. Victoria Hurtado Cerruto Gestion:2/2024

Ejemplo: 54,22,80,35,46,12,50,73,11,15,90,30 Criterio ascendente

Figura No. 31 Árbol Binario de Búsqueda (ABB)



7.16 OPERACIONES BÁSICAS EN UN ABB

Entre las operaciones básicas de un ABB se tienen:

7.16.1INSERTAR UN ELEMENTO. Para lo cual dependiendo de su valor se busca su inserción partiendo desde la raíz manteniendo el orden de los elementos del árbol. En sí, el procedimiento puede reflejarse en los siguientes pasos:

Paso 1. Comparar el dato a insertar con el dato de la raíz. Si es mayor debe avanzarse al subárbol derecho en tanto que si es menor se debe avanzar hacia el subárbol izquierdo.

Paso 2. Repetir el paso anterior hasta que el subárbol derecho es vacío o el subárbol izquierdo es vacío en tal caso se procede a insertar el dato en el lugar respectivo.

7.16.2ELIMINAR UN ELEMENTO. Considerando que el ABB refleja un orden en su contenido la eliminación de un elemento del árbol no debe distorsionar al árbol razón por la que la eliminación puede tratarse como alguno de los siguientes 3 casos:

7.16.2.1 SI EL DATO SE ENCUENTRA EN UN NODO TERMINAL. El proceso de eliminación es similar al descrito anteriormente para árboles binarios normales es decir se procede simplemente a desconectar el nodo que contiene al dato a eliminar.

Ejemplo

Eliminar el dato 46 del ABB de Figura No. 32 luego se tiene el ABB de la Figura No. 33.

Figura No. 32 Árbol Binario de Búsqueda (ABB)

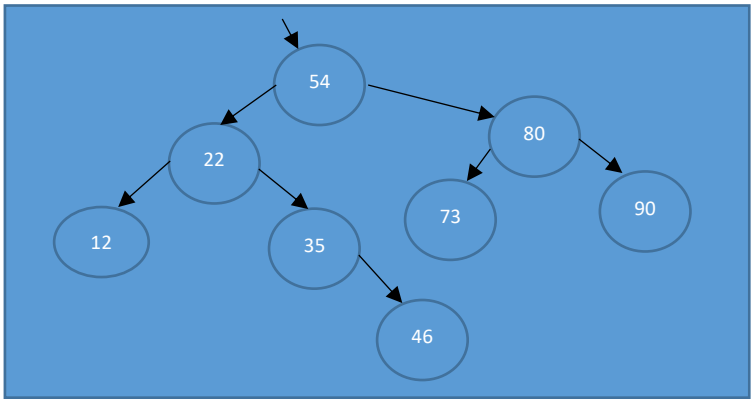
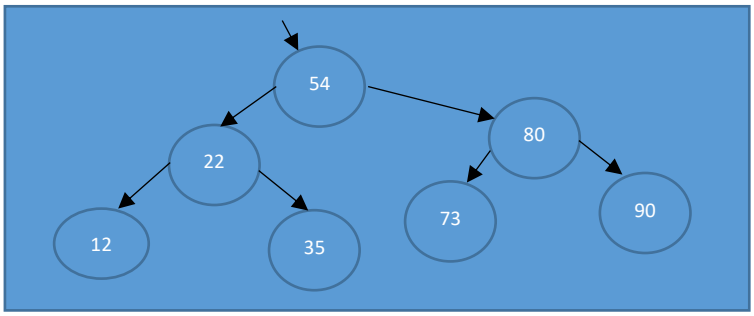


Figura No. 33 Árbol Binario de Búsqueda (ABB) sin 46

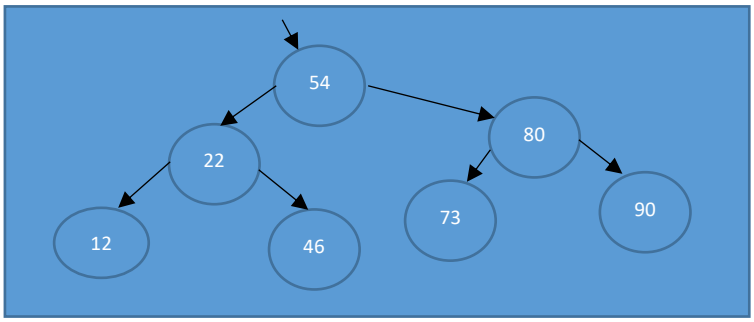


7.16.2.2SI EL DATO A ELIMINAR SE ENCUENTRA EN UN NODO EL CUAL TIENE UN NODO DESCENDIENTE DIRECTO. En tal caso reemplazarlo con ése nodo descendiente directo. Procedimiento similar al descrito para árboles binarios normales.

Ejemplo

Eliminar del ABB de la Fig. No. 32 el dato 35 luego se tiene el ABB de la Figura No.34

Figura No. 34 Árbol Binario de Búsqueda (ABB)



7.16.2.3SI EL DATO A ELIMINAR SE ENCUENTRA EN UN NODO CON DOS DESCENDIENTES. Para eliminar el nodo referido se identifican dos procedimientos:

i) Sustituirlo con el nodo de más a la derecha del subárbol izquierdo

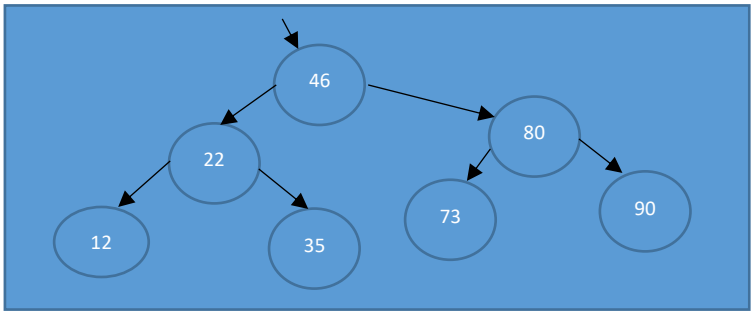
ó bien

ii) Sustituirlo con el nodo de más a la izquierda del subárbol derecho

Ejemplo

Eliminar del ABB de la Figura No. 32 el dato 54 luego se obtiene el ABB de la Figura No. 35 aplicando el paso i)

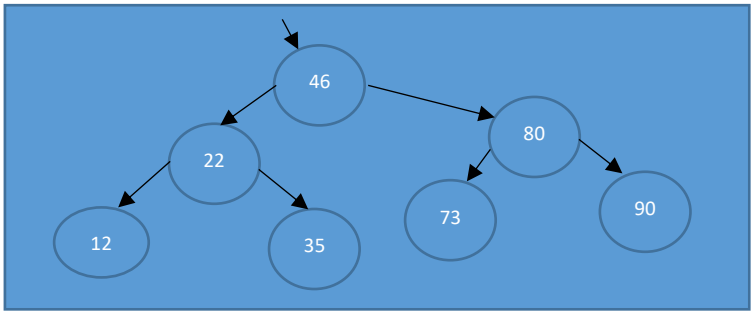
Figura No. 35 Árbol Binario de Búsqueda (ABB)



En el tratamiento de árboles binarios de búsqueda (ABB) existe adicionalmente una definición importante que es oportuno rescatar respecto al orden que manifiesta el contenido del ABB que dice:
"Si a un árbol binario de búsqueda se aplica el recorrido enorden se desplegarán los elementos del árbol ordenados"

Ejemplo

Figura No. 36 Recorrido enorden del (ABB)



Enorden:

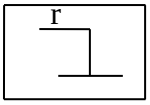
12,22,35,46,73,80,90

Por tanto, podemos distinguir principalmente dos tipos de árboles binarios según su contenido arboles binarios normales (ABNormal) y arboles binarios de búsqueda (ABB).

7.17ESTADO INICIAL DEL TDA ABNormal

Con el objeto de evitar desbordamientos de memoria a continuación se define el estado inicial de un ABNormal (ver Figura No. 37).

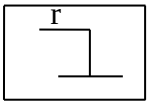
Figura No. 37 Estado inicial de un ABNormal



7.18ESTADO INICIAL DELTDA ABB

De manera similar el estado inicial de un ABB se representa en la Figura No. 38.

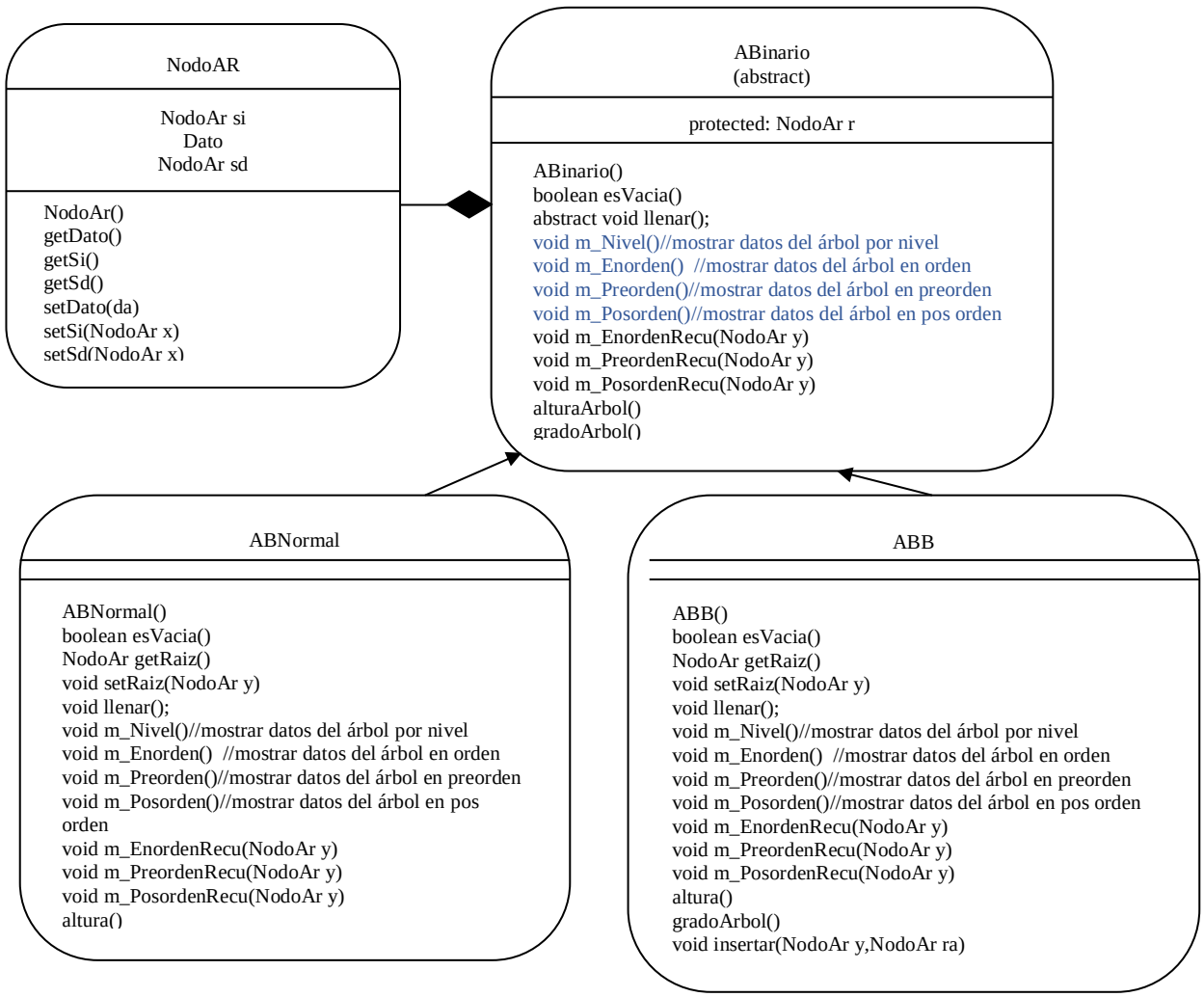
Figura No. 38 Estado inicial de un ABB



7.19 ESTRUCTURA DE CLASES DEL TDA ABinario

La Figura No 39 proporciona la estructura de clases correspondiente.

Figura No. 39 Estructura de clases del TDA ABinario, ABNormal y ABB



7.20IMPLEMENTACIÓN DEL TAD ABinario

```
import java.util.Scanner;
public abstract class ABinario {
protected NodoAr r;
ABinario()
{r=null;}
boolean esVacia()
{
if(r==null)return true;
return false;
}
NodoAr getRaiz()
{
returnr;
}

void setRaiz(NodoAr y)
{
r=y;
}
abstract void llenar();
void m_Nivel()              //Mostrarelementospornivel
{
  Pila nivel=new Pila(100);
  Pila su=new Pila(100);intn;
NodoAr x,w;
intres;
nivel.adicionar(r);
while(!nivel.esVacia())
  {
while(!nivel.esVacia())
{
  x=(NodoAr)nivel.eliminar();
System.out.print(x.getDato()+"\t");
if (x.getSi()!=null) su.adicionar(x.getSi());
if (x.getSd()!=null) su.adicionar(x.getSd());
}
  nivel.vaciar(su);
  System.out.println();
   }
 }

void m_Enorden()  //mostrardatosdelárbolenorden IRD
{
 Pila p=new Pila(100);
NodoAr x=r;
 p.adicionar(null);
while(p.nElem()!=0)
{
while(x!=null)
{
p.adicionar(x);
 x=x.getSi();
}
x=(NodoAr)p.eliminar();
if(x!=null){
```

```
System.out.print(x.getDato()+ " ");
x=x.getSd();
}
}
}

void m_Preorden()
{
  NodoAr x,nd,ni;
Pila p=new Pila(100);
 p.adicionar(null);
  x=r;

while(x!=null)
 {
 System.out.print(x.getDato()+ " ");
 nd=x.getSd();ni=x.getSi();
if (nd!=null) p.adicionar(nd);
if (ni!=null)x=ni;
else x=(NodoAr)p.eliminar();
 }
 }
void m_Posorden()
 {
NodoAr nd,x;
Pila p=new Pila(100);
 p.adicionar(null); x=r;
while(p.nElem()!=0)
 {
while(x!=null)
{
       p.adicionar(x);
       nd=x.getSd();
      if(nd!=null){p.adicionar(nd);
      p.adicionar(null);
                    }
       x=x.getSi();
}
 x=(NodoAr)p.eliminar();
while (x!=null)
 {
       System.out.print(x.getDato()+ " ");
       x=(NodoAr)p.eliminar();
}
if (p.nElem()!=0)x=(NodoAr)p.eliminar();
}
}

void m_EnordenRecu(NodoAr y)// IRD
 {
      if(y!=null){
                   m_EnordenRecu(y.getSi());
                   System.out.print(y.getDato()+"\t");
                   m_EnordenRecu(y.getSd());
                 }
 }
```

```
void m_PreordenRecu(NodoAr y)
{
      if(y!=null){
                    System.out.print(y.getDato()+"\t");
                    m_PreordenRecu (y.getSi());
                    m_PreordenRecu (y.getSd());
                  }
 }



void m_PosordenRecu(NodoAr y)//ID
 {
      if(y!=null){
                    m_PosordenRecu (y.getSi());
                    m_PosordenRecu (y.getSd());
                    System.out.print(y.getDato()+"\t");
      }
 }

int alturalArbol()              //Calculalaalturadelárbol
{
  Pila nivel=new Pila(100);
Pila su=new Pila(100);
intn,res,alt=0;
NodoAr x,w;
nivel.adicionar(r);
while(!nivel.esVacia())
  {
while(!nivel.esVacia())
  {
  x=(NodoAr)nivel.eliminar();
System.out.print(x.getDato()+"\t");
if (x.getSi()!=null) su.adicionar(x.getSi());
if (x.getSd()!=null) su.adicionar(x.getSd());
}
  nivel.vaciar(su); alt++;
  System.out.println();
 }
return alt;
}

int gradoArbol()
{
  Pila nivel=new Pila(100); Pila su=new Pila(100);intn;NodoAr x,w;
intres,maxgra=0,gra;
nivel.adicionar(r);
while(!nivel.esVacia())
{
while(!nivel.esVacia())
  {
  x=(NodoAr)nivel.eliminar();gra=1;
System.out.print(x.getDato()+"\t");
if (x.getSi()!=null) su.adicionar(x.getSi());
if (x.getSd()!=null) su.adicionar(x.getSd());
if ((x.getSi()!=null)&&(x.getSd()!=null))gra=2;
if ((x.getSi()==null)&&(x.getSd()==null))gra=0;
```

```
if (gra>maxgra) maxgra=gra;
  }
 nivel.vaciar(su);System.out.println();
}
return maxgra;
}
}
```

7.21 IMPLEMENTACIÓN DEL TAD ABNormal

```
import java.util.Scanner;
public class ABNormal extends ABinario {
ABNormal()
{ super();}

NodoAr getRaiz()
{return r;}

void setRaiz(NodoAr k)
{r=k;}

void llenar()              //llenaporniveldatosenteros
{
  Pila nivel=new Pila(100);
Pila su=new Pila(100);
intn;
NodoAr w,x; int res;
r=new NodoAr();
  Scanner lee=new Scanner(System.in);
System.out.println("Digite dato de la raiz");
Object dato=newInteger(lee.nextInt());
r.setDato(dato);

nivel.adicionar(r);
while(!nivel.esVacia())
  {
while(!nivel.esVacia())
  {
  x=(NodoAr)nivel.eliminar();
System.out.print("Desea descendiente izquierdo de "+x.getDato()+" Si=1
No<>1?");
res=lee.nextInt();
if (res==1){ w=new NodoAr();
System.out.println("Digite dato izquierdo");
dato=newInteger(lee.nextInt());
w.setDato(dato);x.setSi(w);
     su.adicionar(x.getSi());
   }
  System.out.print("Desea descendiente derecho de "+x.getDato()+" Si=1
No<>1?");
res=lee.nextInt();
if (res==1){ w=new NodoAr();
System.out.println("Digite dato derecho");
dato=newInteger(lee.nextInt());
w.setDato(dato);x.setSd(w);
su.adicionar(x.getSd());
```

```
    }
}
  nivel.vaciar(su);
  }
}

void m_Nivel()
{
super.m_Nivel();
}

void m_Enorden()
{
super.m_Enorden();
}

void m_Preorden()
{
super.m_Preorden();
}

void m_Posorden()
{
super.m_Posorden();}
void m_PreordenRecu(NodoAr y)
{
  super.m_PreordenRecu(y);
}

void m_EnordenRecu(NodoAr y)
{
  super.m_PreordenRecu(y);
}

void m_PosordenRecu(NodoAr y)
{
  super.m_PosordenRecu(y);
}

int alturaArbol()
{
return(super.alturalArbol());
}

int gradoaArbol()
{
return(super.gradoArbol());
}
}
```

7.22IMPLEMENTACIÓN TAD ABB

```
import java.util.Scanner;
public class ABB extends ABinario{

ABB()
```

```
{super();}

NodoAr getRaiz()
{returnr;}

voidsetRaiz(NodoAr k)
{ r=k;}

int alturaArbol()
{return(super.alturalArbol());}

int gradoaArbol()
{return(super.gradoArbol());}

void m_Nivel()
{super.m_Nivel();}

void m_Enorden()
{super.m_Enorden();}

void m_Preorden()
{ super.m_Preorden();}

void m_Posorden()
{ super.m_Posorden();}

void m_PreordenRecu(NodoAr y)
{  super.m_PreordenRecu(y);}

void m_EnordenRecu(NodoAr y)
{  super.m_PreordenRecu(y);}

void m_PosordenRecu(NodoAr y)
{  super.m_PosordenRecu(y);}

void llenar()
{
int i,sw,dx,dy; NodoAr y,x;Object da;
Scanner lee=new Scanner(System.in);
System.out.println("Número de elementos?");
int n=lee.nextInt(); x=new NodoAr();
System.out.println("Ingrese dato entero al árbol?");
 da=new Integer(lee.nextInt());
 x.setDato(da);setRaiz(x);
for(i=1;i<n;i++)
{
x=new NodoAr();
System.out.println("Ingrese dato entero al árbol?");
da=new Integer(lee.nextInt());
  x.setDato(da);
y=r;sw=0;
while (sw==0)
  {
   dx=x.getDato().hashCode();dy=y.getDato().hashCode();
if (dx<dy)
if (y.getSi()==null){
                y.setSi(x);
```

```
                        sw=1;
                       }
else y=y.getSi();
else if(y.getSd()==null){
y.setSd(x);
sw=1;
}
else y=y.getSd();
}
}
}

void insertar(NodoAr y,NodoAr ra)//recursivoqueinsertaundatoal ABB
{
int dar,day;
dar=(ra.getDato()).hashCode();
day=(y.getDato()).hashCode();
if(day<dar)
if(ra.getSi()==null)ra.setSi(y);
      else insertar(y,ra.getSi());
else if (ra.getSd()==null)ra.setSd(y);
      else insertar(y,ra.getSd());
}
}
```

8. ÁRBOLES EN JAVA

Según Ceballos [1], una de las clases relacionadas a árboles en Java es la clase **TreeSet** que básicamente permite manejar un conjunto ordenado de datos utilizando para su almacenamiento un árbol.

Asimismo se afirma que la clase TreeSet es una de las clases que implementa a la interfaz SortedSet.

En sí, entre las características más importantes de un TreeSet se tienen:

- No permite elementos duplicados
- Los elementos son ordenados en forma ascendente
- Un elemento a comparar debe contar con métodos: equals, hashCode y compareTo.

Por otro lado, entre algunos métodos asociados se tienen:

Tabla No. 1 Métodos de la clase TreeSet.

Método	Descripción
add(Object o)	Adiciona un elemento al conjunto si el aun no está en dicho conjunto.
addAll(Collection c)	Adiciona todos los elementos especificados en la colección.
clear()	Elimina todos los elementos del conjunto
Object first()	Devuelve el primer elemento más pequeño del conjunto.
Object last()	Devuelve el elemento más grande del conjunto ordenado
Boolean isEmpty()	Retorna true si el conjunto esta vacio
Object remove (Object o)	Elimina el objeto especificado del conjunto si existe.
int size()	Devuelve el número de elementos del conjunto

Ejemplo

```
package EjeTreeSet;
import java.util.*;
public class EjeTreeSet {
public static void main(String[] args) {
 TreeSet a=new TreeSet();
 a.add(5);
 a.add(3);
 a.add(8);
 a.add(2);
 a.add(6);
 System.out.println(a);
 a.remove(3);
 System.out.println(a);
}
}
```

TRAZA DE LA EJECUCIÓN

```
[2, 3, 5, 6, 8]
[2, 5, 6, 8]
```

9 APLICACIONES

Según Cairo [11] existen una variedad de aplicaciones de los árboles como ser:

- Construcción de árboles genealógicos
- Representación de fórmulas matemáticas.
- Organización de la información
- Análisis de circuitos electrónicos
- Numeración de capítulos y secciones de un libro.
- Tratamiento de conjuntos de datos.