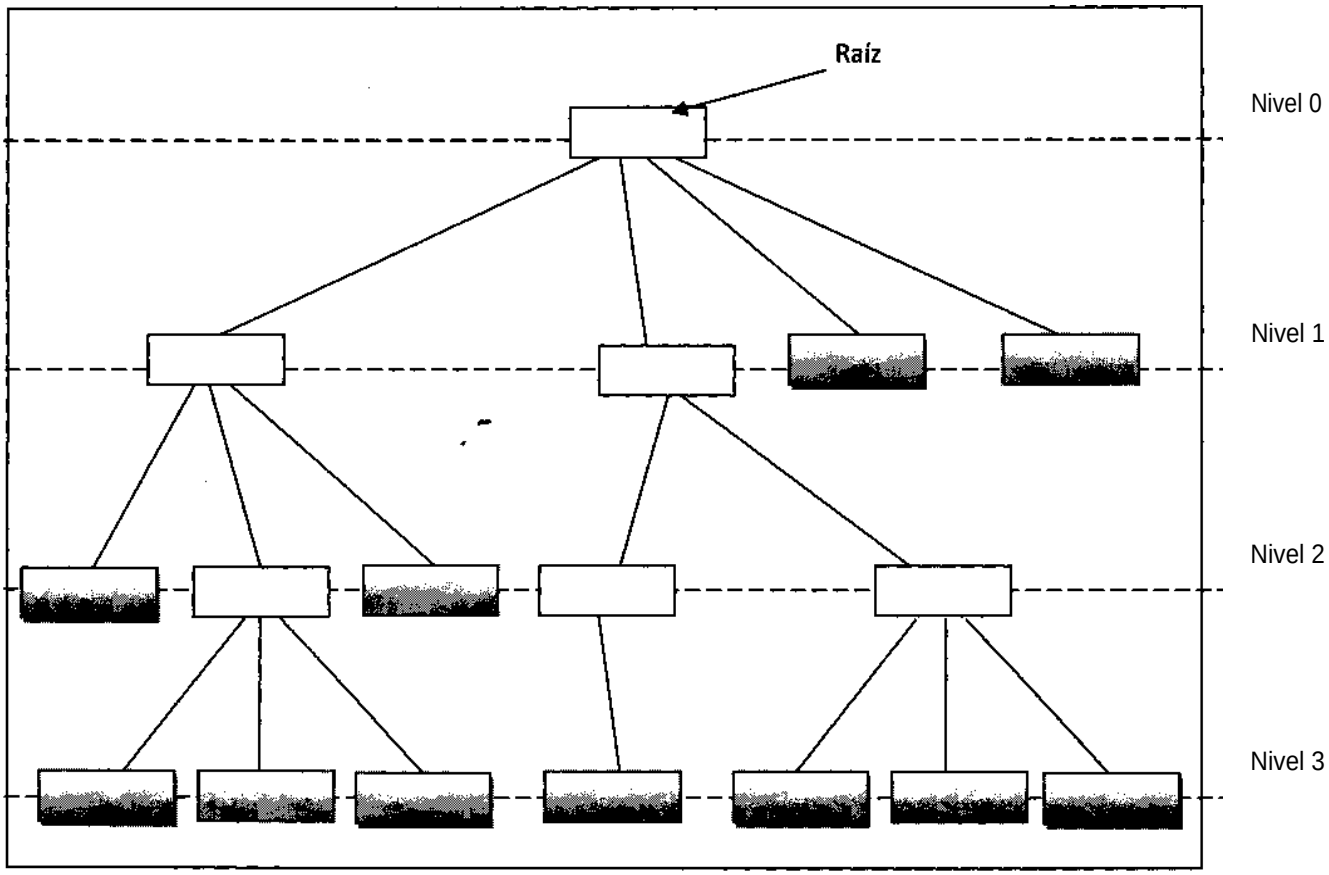
ARBOLES

1. DEFINICIÓN

Un árbol es una estructura jerárquica compuesta por una colección de nodos. Si el árbol no está vacio este contiene un nodo especial denominado raíz.

Asimismo, se lo define como una estructura dinámica no lineal de datos en el sentido de que la estructura árbol puede cambiar durante la ejecución del programa, no lineal en el sentido de que para cada elemento del árbol se puede asociar varios elementos.

2. REPRESENTACIÓN GRAFICA



ELEMENTOS DE UN ARBOL BINARIO

Nodo raíz, es el nodo principal a partir del cual el árbol se define
Nodo Hoja, Nodo hoja es todo aquel nodo que no tiene descendientes.
Nodo descendiente directo de otro. Es aquel nodo inmediato que le sucede al nodo referenciado.
Grado de un nodo, Es el número de descendientes directos del nodo
Grado de un árbol, Es el máximo grado encontrado considerando a todos los nodos del árbol
Nivel de un nodo, Es el número de arcos hasta llegar al nodo.
Altura de un nodo, es el nivel máximo considerándose a todos los nodos del árbol

TIPOS DE ARBOLES

Existen dos tipos de arboles según su grado:

- **Arboles Binarios:** Son árboles que pueden tener 0 o a lo mas dos descendientes

- **Arboles N-arios**: Son arboles de grado N; donde N >2.

ARBOLES BINARIOS

ESTRUCTURA DEL NODO

Un árbol es un conjunto de nodos organizados de manera jerárquica, por lo tanto es necesario saber cómo están conformados los nodos dentro del árbol:

- Nodo Arbol Binario: de grado 2 constan de dos áreas de enlace

Referencia al subárbol izquierdo	Elemento(s)	Referencia al subárbol derecho

- Nodo Arbol N-ario: de grado N constan de N áreas de enlace

Dato(s) Objeto(s)	1	2	3	. . .	N

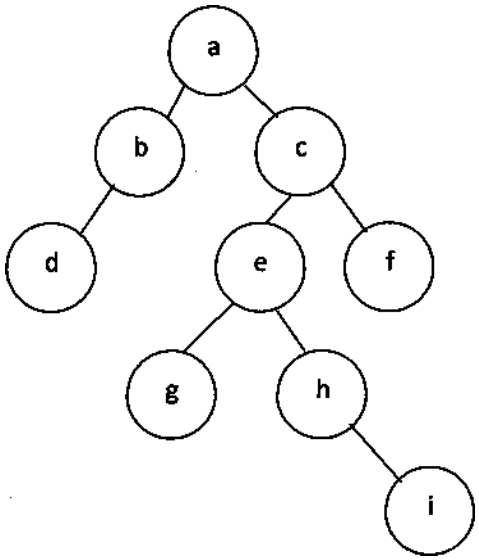
En los arboles binarios se identifican dos tipos particulares de arboles binarios: arboles binarios normales y los arboles binarios de búsqueda

Arboles binarios normales: Son arboles cuyo contenido no están ordenados bajo ningún criterio.

Arboles binarios de búsqueda: Se caracterizan porque son arboles ordenados bajo algún criterio de existencia (ascendente o descendente), esto es en función del elemento del nodo de la raíz se establece un orden en el árbol por ejemplo si el criterio de existencia es “ascendente” todos los valores menores a la raíz se ubican en el lado izquierdo y los mayores en el lado derecho.

RECORRIDOS EN UN ARBOL BINARIO

Sea el árbol Binario A:



Recorrido por Niveles: Su funcionamiento es simple ya que recorre al árbol nivel tras nivel, por ejemplo si mostramos el árbol recorriendo por niveles es resultado sería:

a
be
d e f
gh

Recorrido PreOrden (RID):

a) Visitar Raiz
b) Recorrer subárbol izquierdo
c) Recorrer subárbol derecho

Para el ejemplo el recorrido (RID)= a, b, d, c, e, g, h, i, f

Recorrido InOrden (IRD):

a) Recorrer subárbol izquierdo
b) Visitar Raiz
c) Recorrer subárbol derecho

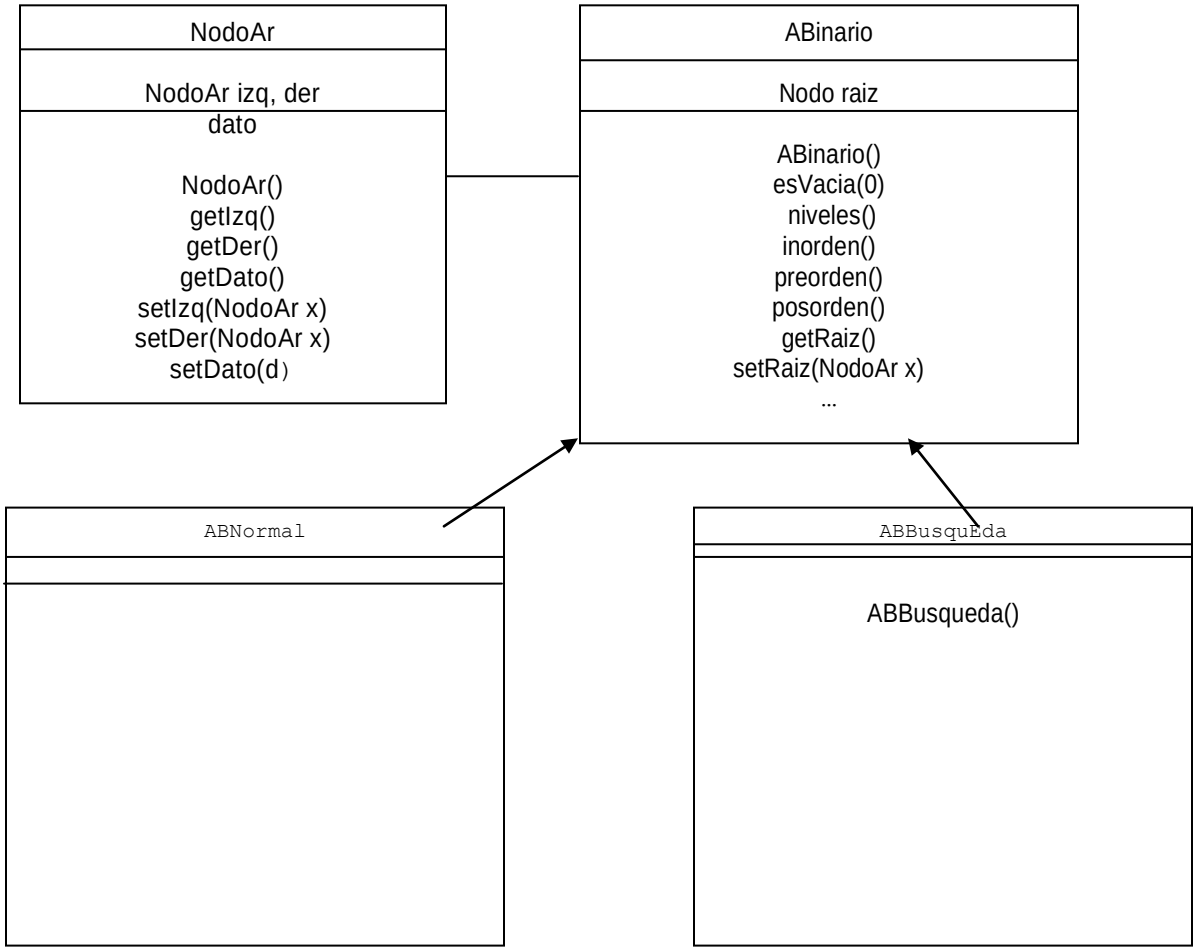
Para el ejemplo el recorrido (IRD)= d, b, a, g, e, h, i, c, f

Recorrido Post Orden (IDR):

a) Recorrer subárbol izquierdo
b) Recorrer subárbol derecho
c) Visitar Raiz

Para el ejemplo el recorrido (IRD)= d, b, g, i, h, e, f, c, a

DIAGRAMA DE CLASES



EJEMPLO: Sea un árbol binario de colores

```
class nodo
{
   nodo izq;
   String color;
   nodo der;
   nodo ()
   {
            izq = der = null;
   }
}
class arbol
{
   nodo raiz;
   arbol ()
   {
            raiz = null;
   }
}

class Pila
{
            int max=50;
   int top;
```

```
class arbolnormal extends arbol
{
   arbolnormal ()
   {
            super ();
   }


   boolean esvacia ()
   {
            if (raiz == null)
               return true;
            return false;
   }


   void crear ()
   {
            String resp;
            Pila nivel = new Pila ();
            Pila desc = new Pila ();

            nodo x = new nodo ();
```

```
   nodo v[]=new nodo[max];
   Pila ()
   {
            top = 0;
   }


   boolean esvacia ()
   {
            if (top == 0)
              return (true);
            return (false);
   }


   boolean esllena ()
   {
            if (top == max)
              return (true);
            return (false);
   }


   void adicion (nodo elem)
   {
            if (!esllena ())
            {
              top++;
              v [top] = elem;
            }
            else
              System.out.println ("Pila llena");
   }


   nodo eliminacion ()
   {
            nodo elem = null;
            if (!esvacia ())
            {
              elem = v [top];
              top--;

            }
            else
              System.out.println ("Pila vacia");
            return (elem);
   }
   void vaciar(Pila Z)
   {
            while(!Z.esvacia())
                     adicion(Z.eliminacion());
   }


}


public class Principal {

            /**
            System.out.print("Dato Raiz-->");
            x.color = Leer.dato ();
            raiz = x;
            nivel.adicion (raiz);
            while (!nivel.esvacia ())
            {
              while (!nivel.esvacia ())
              {
                       x = nivel.eliminacion ();
                       System.out.print (x.color + "Tendra
Izquierda ? S/N");
                       resp = Leer.dato ();
                       if (resp.compareTo ("s") == 0)
                       {
                         nodo y = new nodo ();
                         System.out.print("Dato-->");
                         y.color = Leer.dato ();
                         x.izq = y;
                         desc.adicion (y);
                       }
                       else
                         x.izq = null;
                       System.out.print (x.color + "Tendra
Derecha ? S/N");
                       resp = Leer.dato ();
                       if (resp.compareTo ("s") == 0)
                       {
                         nodo y = new nodo ();
                         System.out.print("Dato-->");
                         y.color = Leer.dato ();
                         x.der = y;
                         desc.adicion (y);
                       }
                       else
                         x.der = null;
              }
              nivel.vaciar(desc);
            }
   }


   void mostrar ()
   {

            nodo x;
            Pila nivel = new Pila ();
            Pila desc = new Pila ();

            nivel.adicion (raiz);
            while (!nivel.esvacia ())
            {
              while (!nivel.esvacia ())
              {
                       x = nivel.eliminacion ();
                       System.out.print (x.color + " ");

                       if (x.izq != null)
                         desc.adicion (x.izq);
                       if (x.der != null)
                         desc.adicion (x.der);
              }
```

```
	 * @param args
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method
stub
		arbolnormal A=new arbolnormal();
		//A.crear();
		//A.mostrar();
		A.raiz=new nodo();
		A.crear(A.raiz);
		A.preorden(A.raiz);

	}

}
```

```
		System.out.println ();
		while (!desc.esvacia ())

			nivel.adicion (desc.eliminacion ());
		}
	}



//PROCESOS RECURSIVOS

	void crear(nodo r)
	{
		if(r!=null)
		{
			System.out.print("Introducir Dato-->");
			r.color=Leer.dato();
			System.out.print(r.color+" Tendra Izq
? S/N");
			String resp=Leer.dato();
			if(resp.equals("s"))
			{
				nodo nue=new nodo();
				r.izq=nue;
				crear(r.izq);
			}
			System.out.print(r.color+" Tendra Der
? S/N");
			resp=Leer.dato();
			if(resp.equals("s"))
			{
				nodo nue=new nodo();
				r.der=nue;
				crear(r.der);
			}
		}

	}
	void preorden (nodo r)
	{
	  if(r!=null)
	  {
		System.out.println(r.color+" ");
		preorden(r.izq);
		preorden(r.der);
	  }
	}

	void inorden (nodo r)
	{
		if (r != null)
		{
		  inorden (r.izq);
		  System.out.print (r.color+" ");
		  inorden (r.der);
		}
	}

}
```

EJEMPLO: Sea un árbol binario de Películas

```
public class NodoP {
        String titulo,genero;
        NodoP izq,der;

        NodoP()
        {
                izq=der=null;
        }

}

public class ArbolBinarioP {
        NodoP raiz;

        ArbolBinarioP()
        {
                raiz=null;
        }

}



class Pila
{
        int max=50;
  int top;
  NodoP v[]=new NodoP[max];
  Pila ()
  {
        top = 0;
  }


  boolean esvacia ()
  {
        if (top == 0)
          return (true);
        return (false);
  }


  boolean esllena ()
  {
        if (top == max)
          return (true);
        return (false);
  }


  void adicion (NodoP elem)
  {
        if (!esllena ())
        {
          top++;
          v [top] = elem;
        }
        else
          System.out.println ("Pila llena");
  }


  NodoP eliminacion ()
  {
        NodoP elem = null;
        if (!esvacia ())
        {
          elem = v [top];
```

```
public class ABNormalP extends ArbolBinarioP{

        ABNormalP()
        {
                super();
        }
        void crear(NodoP r)
          {
            if(r!=null)
            {
                System.out.print("Titulo Pelicula-->");
                r.titulo=Leer.dato();
                System.out.print("Genero Pelicula-->");
                r.genero=Leer.dato();

                System.out.print(r.titulo+" Tendra Izq ? S/N");
                String resp=Leer.dato();
                if(resp.equals("s"))
                {
                    NodoP nue=new NodoP();
                    r.izq=nue;
                    crear(r.izq);
                }
                System.out.print(r.titulo+" Tendra Der ? S/N");
                resp=Leer.dato();
                if(resp.equals("s"))
                {
                    NodoP nue=new NodoP();
                    r.der=nue;
                    crear(r.der);
                }
            }

          }
          void preorden (NodoP r)
          {
           if(r!=null)
           {
              System.out.println(r.titulo+" "+r.genero);
              preorden(r.izq);
              preorden(r.der);
           }
          }
          int nronodos(NodoP r)
          {
                if(r!=null)
                {
                        return nronodos(r.izq)+nronodos(r.der)+1;
                }
                else
                        return 0;

          }
          void completar (NodoP r)
          {
           if(r!=null)
           {
              if(r.izq!=null && r.der==null)
              {
                    NodoP nue=new NodoP();
                    System.out.print("Titulo Pelicula-->");
                nue.titulo=Leer.dato();
                System.out.print("Genero Pelicula-->");
                nue.genero=Leer.dato();
```

```
                top--;

            }
            else
              System.out.println ("Pila vacia");
            return (elem);
  }

  void vaciar(Pila z)
  {
            while (!z.esvacia())

      adicion (z.eliminacion());
  }

}


public class Principal {

            /**
             * @param args
             */
            public static void main(String[] args) {
                        // TODO Auto-generated
method stub

                        //RECURSIVOS
                        ABNormalP z=new
ABNormalP();
                        z.raiz=new NodoP();
                        z.crear(z.raiz);
                        z.preorden(z.raiz);
                        //NroNodos

            System.out.print(z.nronodos(z.raiz));

                        //ITERATIVOS POR
NIVELES
                        ABNormalP zz=new
ABNormalP();
                        zz.crear();
                        zz.mostrar();
                        //1. NroNodos

            System.out.print(zz.nronodos());
                        //2. Completar
                        //3. Mostrar el ultimo nivel
de un arbol
                        //4. mostrar las peliculas de
z que existen en zz
                        //5. mostrar los
descendientes derechos del subarbol izquierdo

            }

}
```

```
                r.der=nue;

            }
            if(r.izq==null && r.der!=null)
            {
                        NodoP nue=new NodoP();
                        System.out.print("Titulo Pelicula-->");
              nue.titulo=Leer.dato();
              System.out.print("Genero Pelicula-->");
              nue.genero=Leer.dato();

              r.izq=nue;

            }

            completar(r.izq);
            completar(r.der);
          }
        }
        void crear ()
        {
          String resp;
          Pila nivel = new Pila ();
          Pila desc = new Pila ();

          NodoP x = new NodoP ();

          System.out.print("Datos Raiz-->");
          System.out.print("Titulo Pelicula-->");
          x.titulo = Leer.dato ();
          System.out.print("Genero Pelicula-->");
          x.genero = Leer.dato ();

          raiz = x;
          nivel.adicion (raiz);
          while (!nivel.esvacia ())
          {
            while (!nivel.esvacia ())
            {
              x = nivel.eliminacion ();
              System.out.print (x.titulo + "Tendra Izquierda ? S/N");
              resp = Leer.dato ();
              if (resp.compareTo ("s") == 0)
              {
                NodoP y = new NodoP ();
                System.out.print("Titulo Pelicula-->");
                        y.titulo = Leer.dato ();
                        System.out.print("Genero Pelicula-->");
                        y.genero = Leer.dato ();

                x.izq = y;
                desc.adicion (y);
              }

              System.out.print (x.titulo + "Tendra Derecha ? S/N");
              resp = Leer.dato ();
              if (resp.compareTo ("s") == 0)
              {
                  NodoP y = new NodoP ();
                System.out.print("Titulo Pelicula-->");
                        y.titulo = Leer.dato ();
                        System.out.print("Genero Pelicula-->");
                        y.genero = Leer.dato ();

                x.der = y;
                desc.adicion (y);
              }
            }
```

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


  void mostrar ()
  {

    NodoP x;
    Pila nivel = new Pila ();
    Pila desc = new Pila ();

    nivel.adicion (raiz);
    while (!nivel.esvacia ())
    {
      while (!nivel.esvacia ())
      {
        x = nivel.eliminacion ();
        System.out.print (x.titulo+""+x.genero + "   ");

        if (x.izq != null)
          desc.adicion (x.izq);
        if (x.der != null)
          desc.adicion (x.der);
      }
      System.out.println ();
      nivel.vaciar(desc);
    }
  }

  int nronodos ()
  {
          int c=0;
    NodoP x;
    Pila nivel = new Pila ();
    Pila desc = new Pila ();

    nivel.adicion (raiz);
    while (!nivel.esvacia ())
    {
      while (!nivel.esvacia ())
      {
        x = nivel.eliminacion ();
        c=c+1;

        if (x.izq != null)
          desc.adicion (x.izq);
        if (x.der != null)
          desc.adicion (x.der);
      }

      nivel.vaciar(desc);
    }
    return c;
  }
  void completar ()
  {

    NodoP x;
    Pila nivel = new Pila ();
    Pila desc = new Pila ();

    nivel.adicion (raiz);
    while (!nivel.esvacia ())
    {
      while (!nivel.esvacia ())
      {
        x = nivel.eliminacion ();
```

```
                    if(x.izq!=null && x.der==null)
                            {
                                    NodoP nue=new NodoP();
                                    System.out.print("Titulo Pelicula-->");
                              nue.titulo=Leer.dato();
                              System.out.print("Genero Pelicula-->");
                              nue.genero=Leer.dato();

                              x.der=nue;

                            }
                            if(x.izq==null && x.der!=null)
                            {
                                    NodoP nue=new NodoP();
                                    System.out.print("Titulo Pelicula-->");
                              nue.titulo=Leer.dato();
                              System.out.print("Genero Pelicula-->");
                              nue.genero=Leer.dato();

                              x.izq=nue;

                            }

                      if (x.izq != null)
                        desc.adicion (x.izq);
                      if (x.der != null)
                        desc.adicion (x.der);
                    }

                    nivel.vaciar(desc);
                  }
                }

}
```