

RECORRIDO EN ARBOLES BINARIOS

Una de las operaciones mas importantes a realizar en un árbol binario es el recorrido de los mismos, recorrer significa visitar los nodos del árbol en forma sistemática, de tal manera que todos los nodos del mismo sean visitados una sola vez.

Existen 3 formas diferentes de efectuar el recorrido y todas ellas de naturaleza recursiva, estas son:

RECORRIDO PREORDEN: En el que se procesa el nodo y después se procesan recursivamente sus hijos.

RECORRIDO POSTORDEN: Donde el nodo dado se procesa después de haber procesado recursivamente a sus hijos.

RECORRIDO INORDEN: En este se procesa recursivamente el hijo izquierdo, luego se procesa el nodo actual y finalmente se procesa recursivamente el hijo derecho.

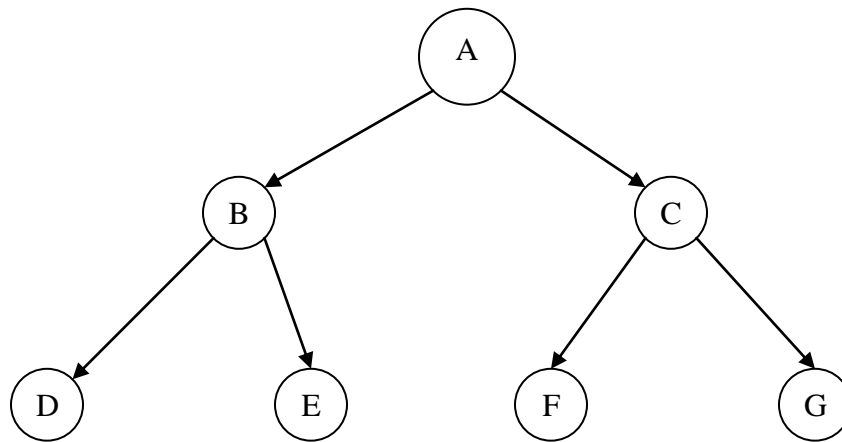
Hay un último recorrido que implementa a estos 3.

RECORRIDO POR NIVELES: Este recorrido procesa los nodos comenzando en la raíz y avanzando de forma descendente y de izquierda a derecha.

RECORRIDO PREORDEN

- VISITAR LA RAIZ
- RECORRER EL SUBARBOL IZQUIERDO
- RECORRER EL SUBARBOL DERECHO

Recorrido en **preorden**: consiste en visitar el nodo actual (visitar puede ser simplemente mostrar la clave del nodo por pantalla), y después visitar el subárbol izquierdo y una vez visitado, visitar el subárbol derecho. Es un proceso recursivo por naturaleza.



PREORDEN: A-B-D-E-C-F-G

ALGORITMO

El algoritmo realiza el recorrido preorden en un árbol binario.

NODO es un dato de tipo puntero.

INFO, IZQ y DER son campos del registro nodo.

INFO es una variable de tipo carácter, IZQ y DER son variables de tipo puntero.

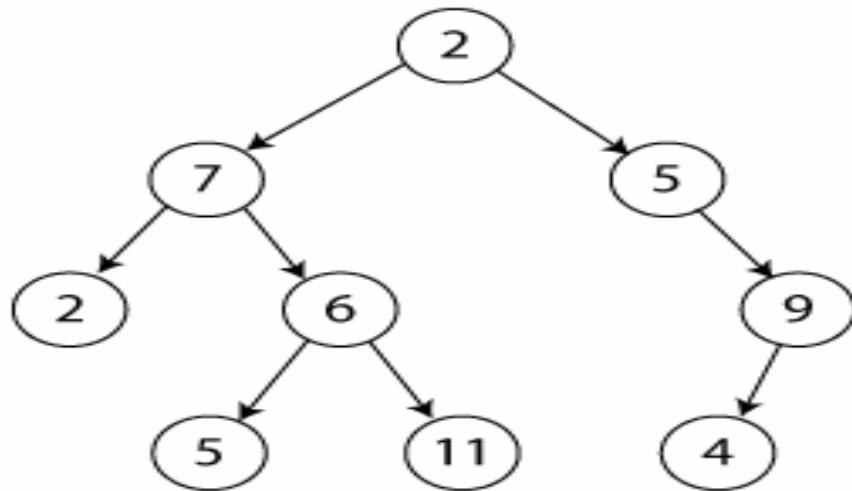
1. si $NODO \neq NULL$
entonces
Visitar el NODO { Escribir $NODO^{INFO}$ }

Regresar a PREORDEN con $NODO^{IZQ}$
{ Llamada recursiva a preorden con la rama izquierda del nodo en cuestión }
Regresa a PREORDEN con $NODO^{DER}$
{ Llamada recursiva a preorden con la rama derecha del nodo en cuestión }
2. Fin del condicional del paso I

CODIGO

```
void preorden(tArbol *a)
{
    if (a != NULL) {
        Visitar(a);           //Realiza una operación en nodo
        preorden(a->hIzquierdo);
        preorden(a->hDerecho);
    }
}
```

Ejemplo de Arbol Binario



RECORRIDO POSTORDEN

En este caso se trata primero el subárbol izquierdo, después el derecho y por último el nodo actual. En el árbol de la figura el recorrido en postorden sería: 2, 5, 11, 6, 7, 4, 9, 5 y 2.

```

void postorden(tArbol *a)
{
    if (a != NULL) {
        postorden(a->hIzquierdo);
        postorden(a->hDerecho);
        tratar(a);           //Realiza una operación en nodo
    }
}

```

RECORRIDO INORDEN

En este caso se trata primero el subárbol izquierdo, después el nodo actual y por último el subárbol derecho. En un AB este recorrido daría los valores de clave ordenados de menor a mayor. En el árbol de la figura el recorrido en inorden sería: 2, 7, 5, 6, 11, 2, 5, 4 y 9.

Pseudocódigo:

```

funcion inorden(nodo)
inicio
    si(existe(nodo))
        inicio
            inorden(hijo_izquierdo(nodo));
            visitar(nodo);           //Realiza una operación en nodo
            inorden(hijo_derecho(nodo));
        fin;
fin;

```

RECORRIDO POR NIVELES

Concluimos implementando el recorrido por niveles. este recorrido procesa los nodos comenzando en la raíz y avanzando en forma descendente y de izquierda a derecha.

El nombre se deriva del hecho de que primero visitamos:

- los nodos del nivel 0 (la raíz),
- después los del nivel 1 (los hijos de la raíz),
- los del nivel 2 (los nietos de la raíz),
- y así sucesivamente.

Un recorrido por niveles se implementa usando una cola en lugar de una pila. La cola almacena los nodos que van a ser visitados. Cuando se visita un nodo ,se colocan sus hijos al final de la cola, donde serán visitados después de los nodos que ya están en la cola. Es fácil ver que esto garantiza que los nodos se visitan por niveles.

El Algoritmo PorNiveles se
Muestra a Continuación.....

ALGORITMO

```
encolar(raiz);
mientras(cola_no_vacia( ))
    inicio
    nodo=desencolar(); //Saca un nodo de la cola      visitar(nodo); //Realiza una
operación en nodo      encolar_nodos_hijos(nodo); //Mete en la cola los hijos del
nodo actual
    fin;
```

CODIGO

```
void amplitud(tArbol *a)
{
    tCola cola;
    tArbol *aux;
    if (a != NULL)
    {   crearCola(cola);
        encolar(cola, a);
        while (!colavacia(cola)) {
            desencolar(cola, aux);
            visitar(aux);
            if (aux->hIzquierdo != NULL)   encolar(cola, aux->hIzquierdo );
            if (aux->hDerecho!= NULL)   encolar(cola, aux->hDerecho);   }
        }
    }
```