

Recorridos de Árboles Binarios

Ing. Luis Humberto González

Recorrer el árbol es “pasar por” o “visitar” todos los nodos del mismo.

Recorridos típicos:

- ✓ Preorden
- ✓ Inorden
- ✓ Postorden

Otro tipo de recorridos

- ✓ Conversos
- ✓ Nivel por nivel

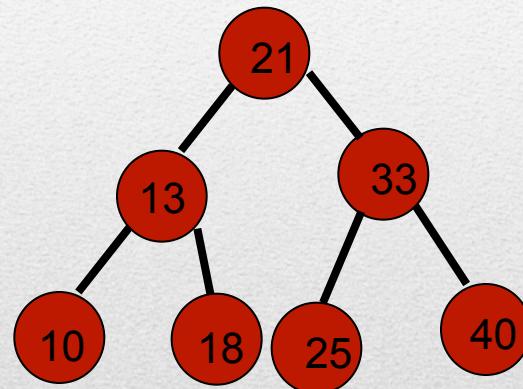
Los recorridos se aplican sobre un **Árbol Binario NO TIENE que ser BST**

Recorridos

Recorrido Preorden

Proceso:

- ✓ Visita el nodo raíz del árbol.
- ✓ Recorre el preorden el subárbol izquierdo del nodo raíz.
- ✓ Recorre el preorden el subárbol derecho del nodo raíz.



Recorrido en Preorden

21, 13, 10, 18, 33, 25, 40

Aplicación: Generar una réplica del árbol.

```
void BST::preorden(NodeT*r)
{
    if ( r != NULL)
    {
        cout << r->getData();
        preorden (r->getLeft());
        preorden (r->getRight());
    }
}
```

Implementación del Preorden para desplegar el contenido del árbol.

Llamada a la función:
preorden(root);

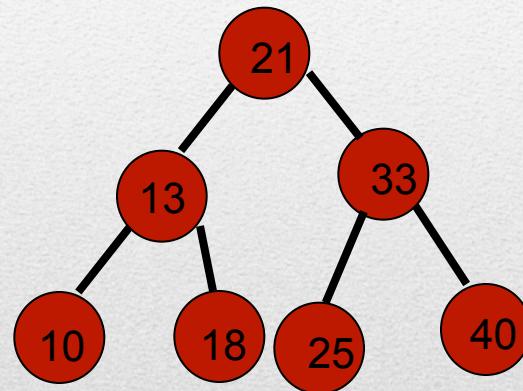
Implementación del Preorden

Recorrido Inorden

Proceso:

- ✓ Recorre en inorden el subárbol izquierdo.
- ✓ Visita la raíz del árbol.
- ✓ Recorre en inorden el subárbol derecho.

Aplicación: Desplegar en orden creciente los elementos del árbol si este es un ABB.



Recorrido en Inorden

10, 13, 18, 21, 25, 33, 40

```
void BST::inorden(NodeT*r)
{
    if ( r != NULL)
    {
        inorden (r->getLeft());
        cout << r->getData();
        inorden (r->getRight());
    }
}
```

Implementación del INorden para desplegar el contenido del árbol.

Llamada a la función:
INorden(root);

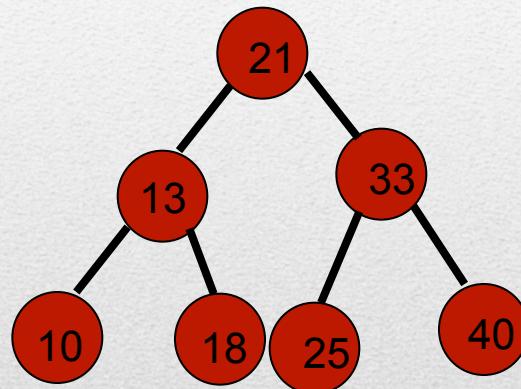
Implementación del Inorden

Recorrido Postorden

Proceso:

- ✓ Recorre en postorden el subárbol izquierdo.
- ✓ Recorre en postorden el subárbol derecho.
- ✓ Visita la raíz del árbol.

Aplicación: Liberar los nodos de un árbol.



Recorrido en Postorden

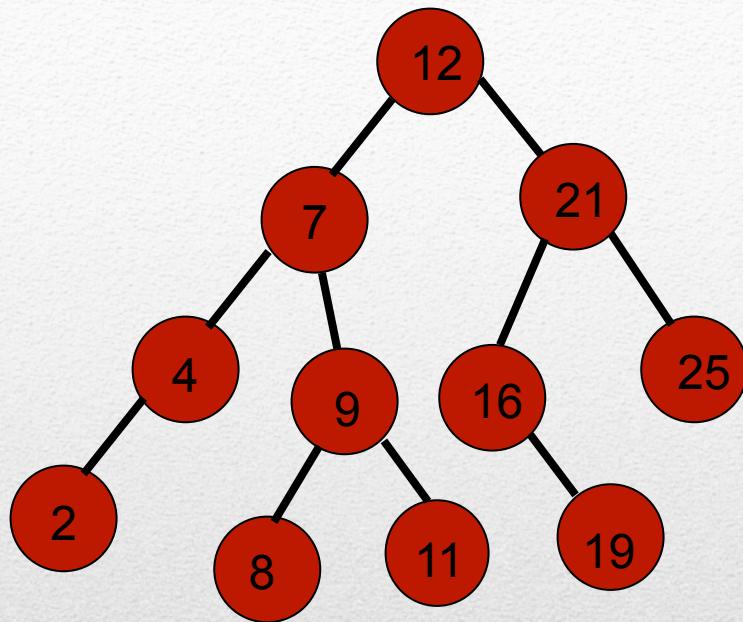
10, 18, 13, 25, 40, 33, 21

```
void BST::postorden(NodeT*r)
{
    if ( r != NULL)
    {
        postorden (r->getLeft());
        postorden (r->getRight());
        cout << r->getData();
    }
}
```

Implementación del Postorden para desplegar el contenido del árbol.

Llamada a la función:
Postorden(root);

Implementación del Postorden



Recorrido en Preorden

12, 7, 4, 2, 9, 8, 11, 21, 16, 19, 25

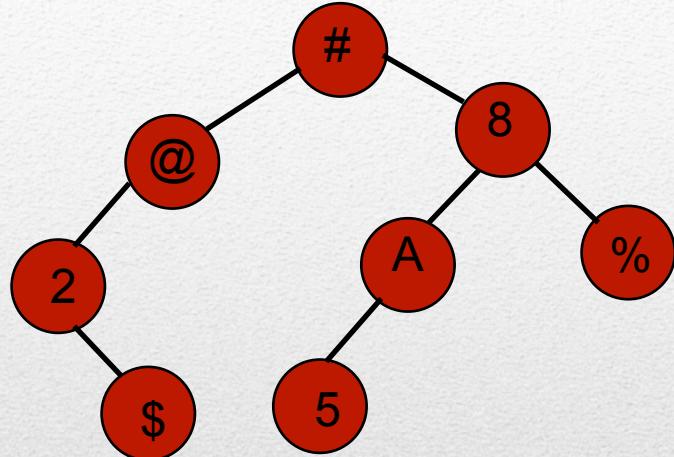
Recorrido en Inorden

2, 4, 7, 8, 9, 11, 12, 16, 19, 21, 25

Recorrido en Postorden

2, 4, 8, 11, 9, 7, 19, 16, 25, 21, 12

Ejemplo....



Recorrido en Preorden

#, @, 2, \$, 8, A, 5, %

Recorrido en Inorden

2, \$, @, #, 5, A, 8, %

Recorrido en Postorden

\$, 2, @, 5, A, %, 8, #

Ejemplo....

Dado los siguientes recorridos:

Recorrido en Preorden

\$, %, A, &, #

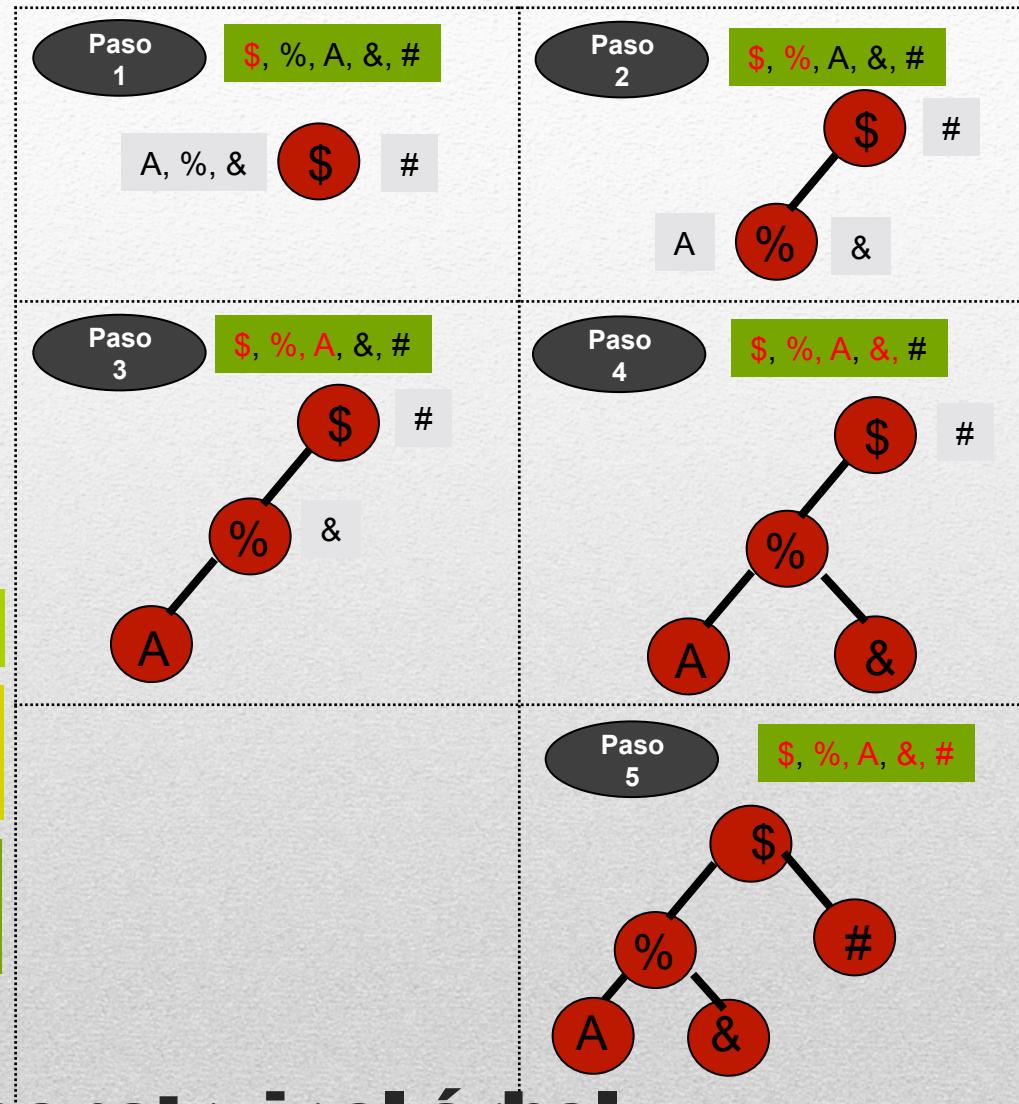
Recorrido en Inorden

A, %, &, \$, #

El **Preorden** indica que la raíz es: \$

El **Inorden** indica quién está a la izquierda y quién a la derecha

El **Preorden** también indica cuál es el siguiente valor a procesar



Dado 2 recorridos construir el árbol

Visitan los nodos en orden contrario es decir, primero a la derecha y después a la izquierda.

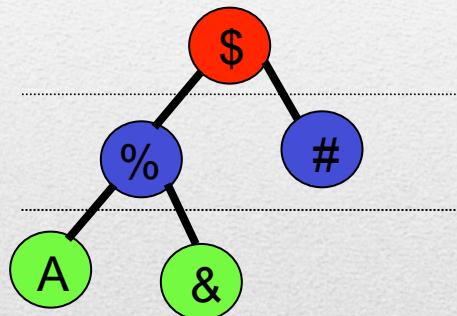
Recorridos conversos:

- ✓ INORDEN CONVERSO
- ✓ PREORDEN CONVERSO
- ✓ POSTORDEN CONVERSO

No es muy común encontrar una aplicación para ellos.

Recorridos conversos

Recorre el árbol en forma horizontal.



Recorrido Nivel por Nivel

\$, %, #, A, &

Recorridos Nivel por Nivel

Para implementar este recorrido se utiliza una Fila que almacena a los nodos del árbol.

Proceso:

Meter el nodo raíz a una Fila.

Mientras la Fila no se vacíe:

- ✓ Sacar un Nodo de la Fila y procesarlo.
- ✓ Meter a la Fila a los hijos del nodo procesado (si éstos existen).

Implementación del Recorrido Nivel por Nivel
