Lógica de programación I

Juan Pablo Restrepo Uribe

Ing. Biomedico

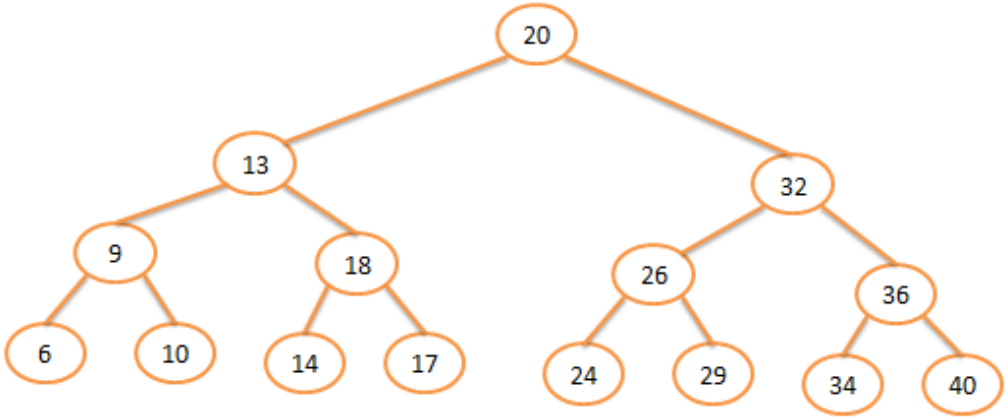
MSc. Automatización y Control Industrial

jprestrepo@correo.iue.edu.co

Institución Universitaria de Envigado

Árbol binario de búsqueda (ABB)

Los árboles binarios de búsqueda son un tipo especial de árbol binario cuya característica radica en la forma ordenada de insertar sus elementos, facilitando así la búsqueda de un nodo en particular. Para puntualizar aún más, se tratarán los árboles binarios de búsqueda, en los que se tiene preestablecido un cierto orden, que seguramente ayudará a encontrar un cierto dato dentro de un árbol con mucha rapidez.



Insertar dato

Todo comienza en la raíz, y después todos los valores se agregan de acuerdo con ésta, ya que si el valor es menor lo ponemos en la rama izquierda, y si es mayor en la derecha.

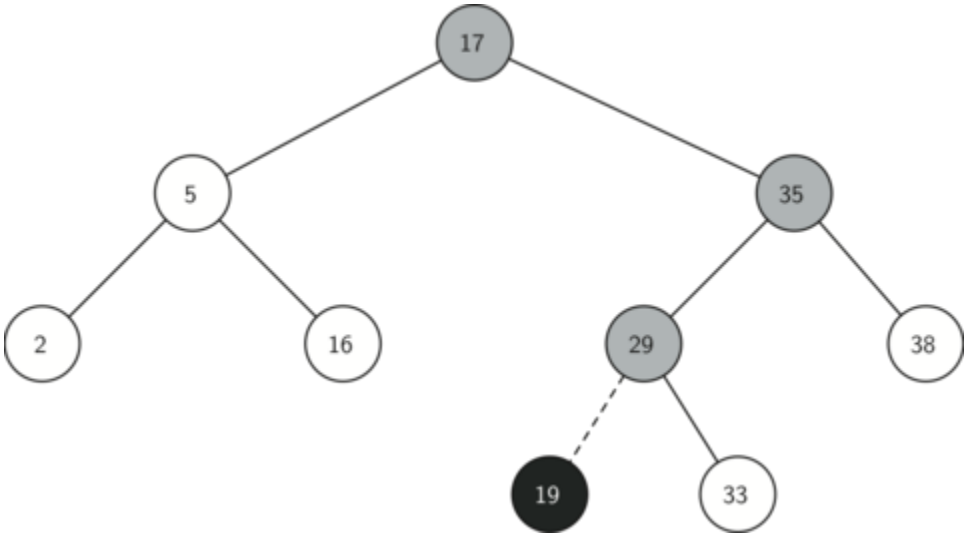
En caso de que la rama ya esté ocupada, buscamos su subrama para, de nuevo, acomodar el valor. Es por eso por lo que usamos recursión, ya que no se sabe cuándo se encontrará una rama disponible.

Árbol binario de búsqueda (ABB)

¿Cómo es este orden prefijado o preestablecido?
Solo se debe cumplir la condición que para cada nodo se tiene que:

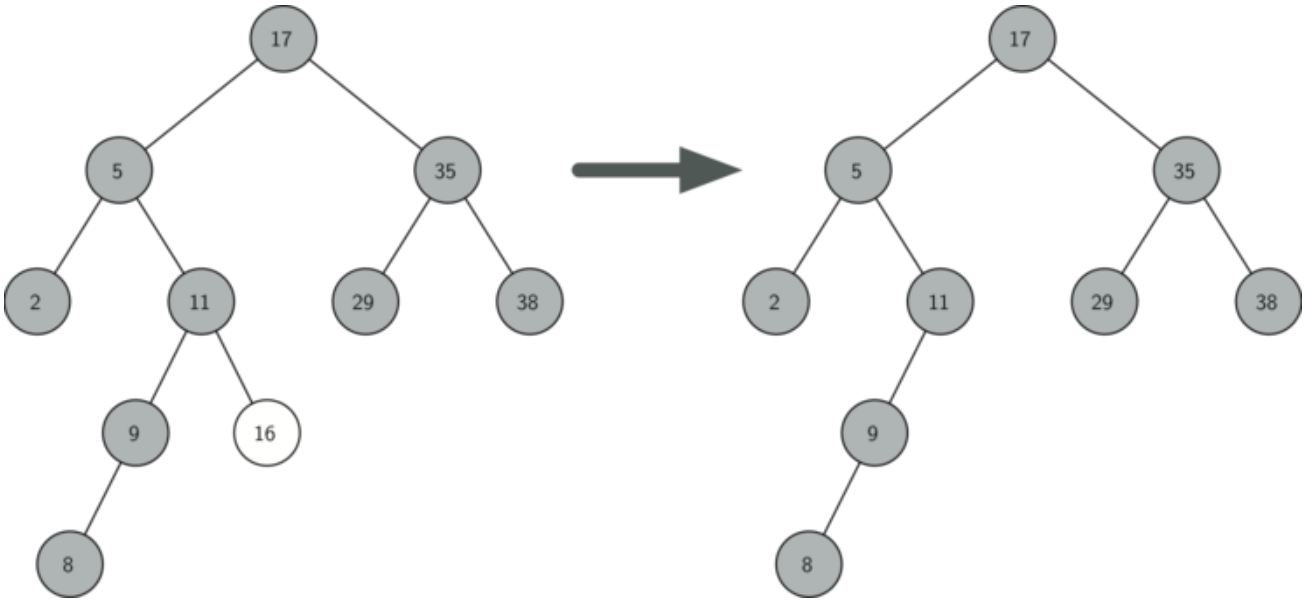
- La rama de la izquierda contendrá elementos menores.
- La rama de la derecha contendrá elementos mayores.

Insertar dato



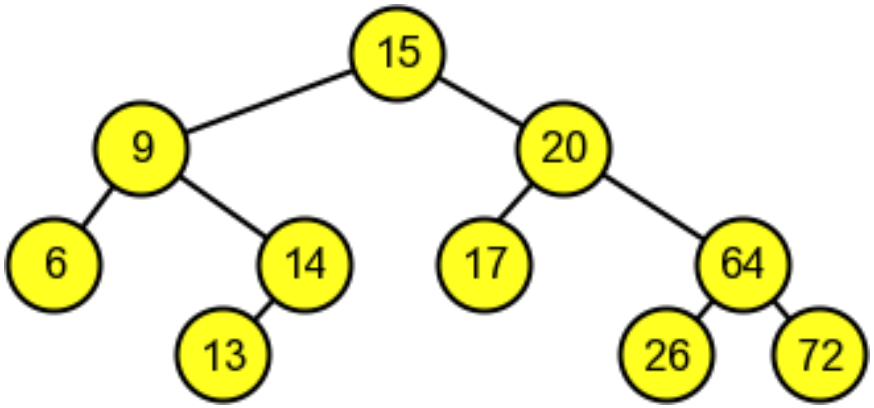
Formas de recorrer un árbol binario

Existen 3 métodos para recorrer un árbol y solo cambia el orden en el que se visita cada nodo del árbol.



Formas de recorrer un árbol binario

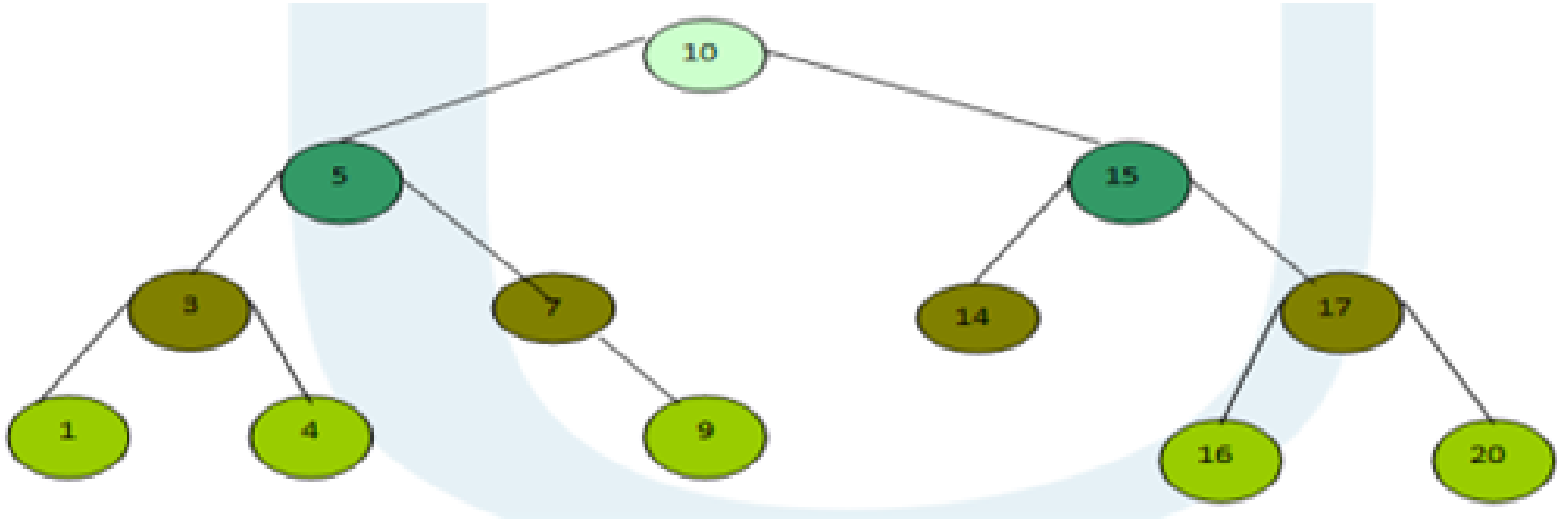
Los árboles binarios, son estructuras de datos no lineales, son considerados como estructuras jerárquicas y como tal su forma de recorrerlos difiere sustancialmente en comparación con las listas enlazadas que son estructuras de datos de tipo lineal. En ese orden de ideas, el recorrido de un árbol binario se lleva a cabo en tres sentidos: Preorden, Inorden y Postorden.



Recorrido en Preorden

Recorrer un árbol en preorden consiste en primer lugar, examinar el dato del nodo raíz, posteriormente se recorrer el subárbol izquierdo en preorden y finalmente se recorre el subárbol derecho en preorden. Esto significa que para cada subárbol se debe conservar el recorrido en preorden, primero la raíz, luego la parte izquierda y posteriormente la parte derecha.

Recorrido en Preorden



Recorrido en Preorden

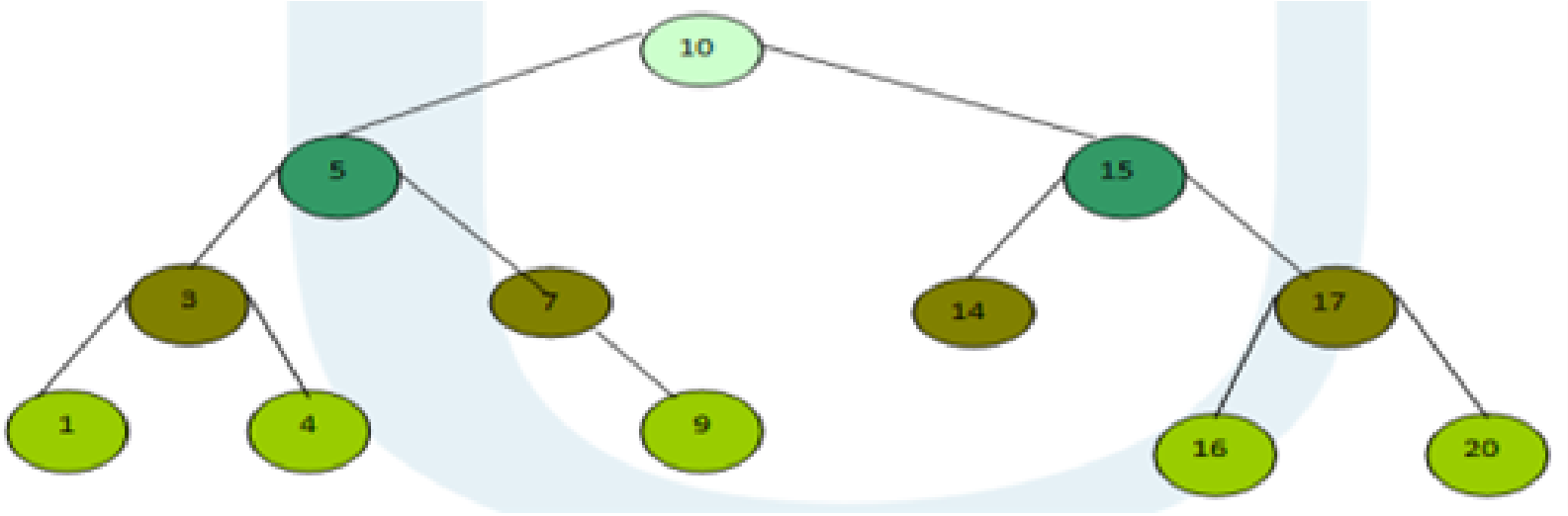
El recorrido inicia con el subárbol izquierdo

- El primer nodo para visitar es la raíz que es el nodo 10
- Luego se visita el subárbol izquierdo con el nodo 5
- Posteriormente el 3
- Luego el nodo 1
- Sigue con el nodo 4
- Pasamos al nodo 7
- Luego el 9.

Continuamos con el recorrido del subárbol derecho en preorden

- Visitamos el nodo 15
- Luego el 14
- Continuamos con el 17
- Se visita el 16
- Se finaliza con la visita del nodo 20.

Recorrido en Preorden



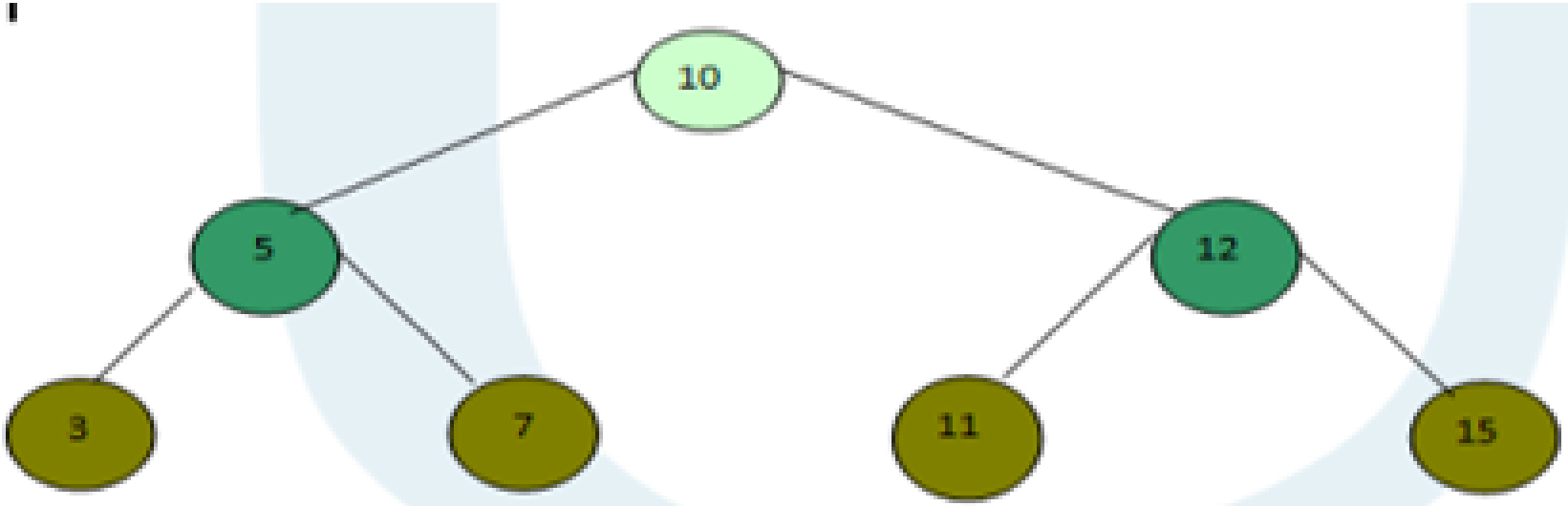
El resultado completo del recorrido en preorden para el árbol de la imagen es:

10 – 5 – 3 – 1 – 4 – 7 – 9 – 15 – 14 - 17 – 16 - 20

Recorrido en Inorden

Recorrer un árbol en Inorden consiste en primer lugar en recorrer el subárbol izquierdo en Inorden, luego se examina el dato del nodo raíz, y finalmente se recorre el subárbol derecho en Inorden. Esto significa que para cada subárbol se debe conservar el recorrido en Inorden, es decir, primero se visita la parte izquierda, luego la raíz y posteriormente la parte derecha.

Recorrido en Inorden



Recorrido en Inorden

El recorrido inicia con el subárbol izquierdo,

- el primer nodo para visitar es el 3
- Luego se visita el 5
- Posteriormente el 7

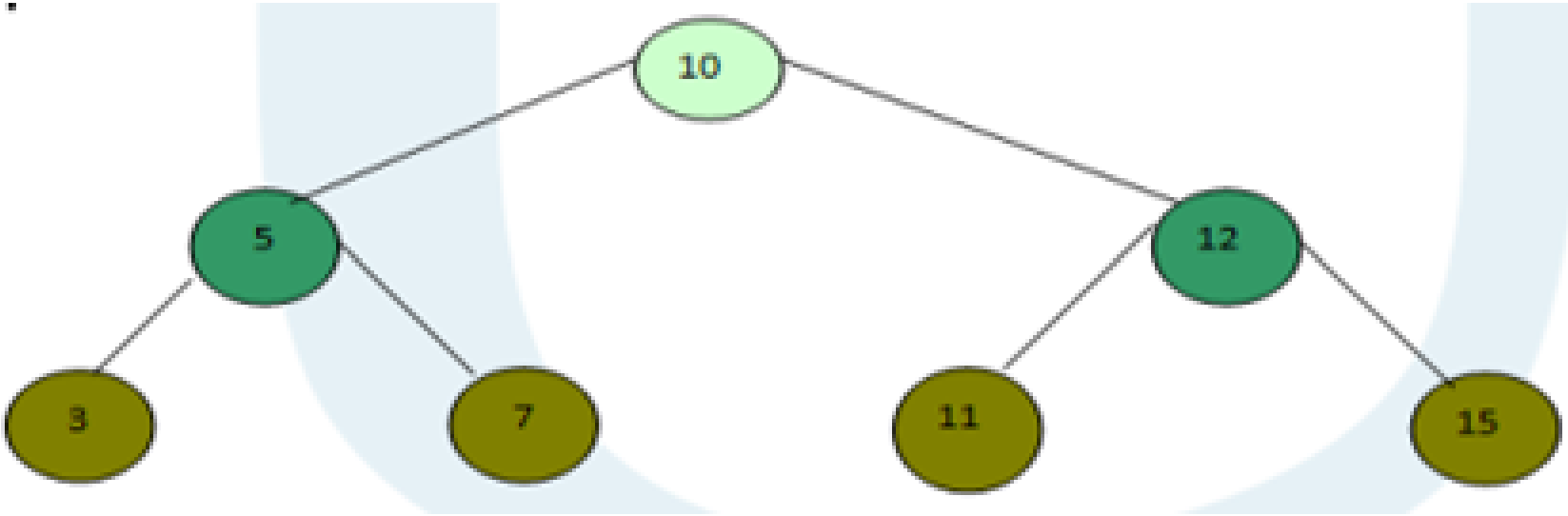
Con esto se garantiza que el recorrido del subárbol izquierdo se hizo en Inorden.
Finalizado el recorrido del subárbol izquierdo

- Se visita el nodo de la raíz, que es el número 10.

Solo queda recorrer el subárbol derecho en Inorden

- Se visita el 11
- Luego el 12
- Se finaliza con la visita del nodo 15

Recorrido en Inorden



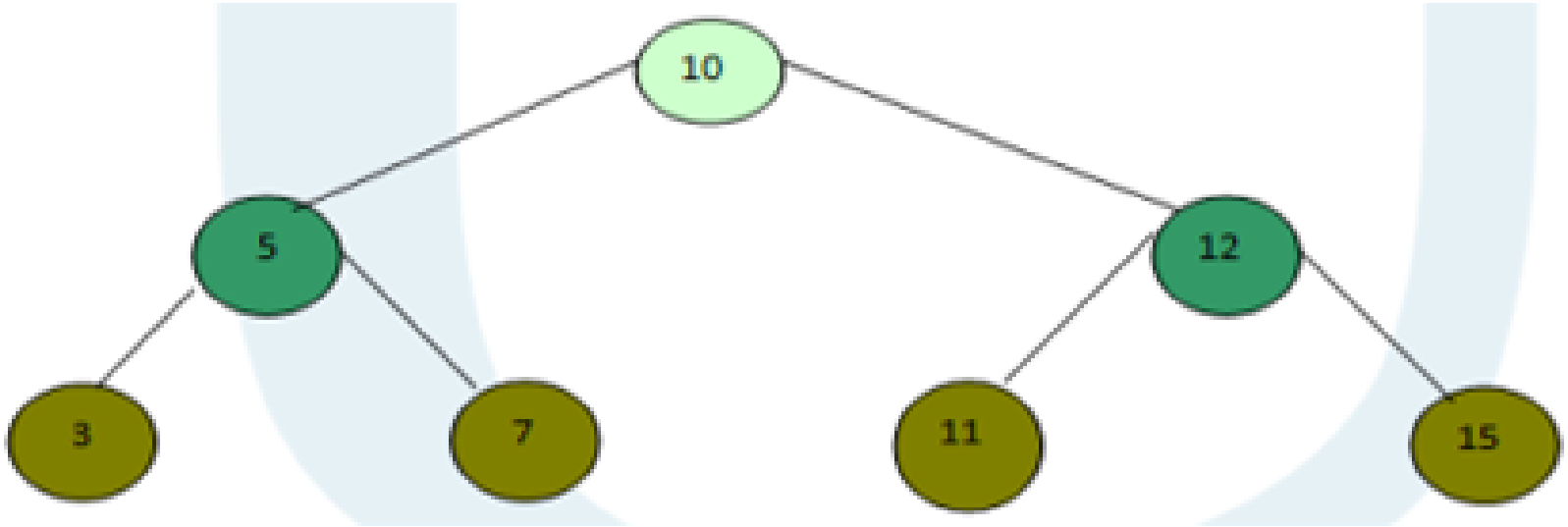
El resultado completo del recorrido en Inorden para el árbol de la imagen es:

3 – 5 – 7 – 10 – 11 – 12 - 15

Recorrido en Postorden

Recorrer un árbol en Postorden consiste en primer lugar en recorrer el subárbol izquierdo en Postorden, luego se recorre el subárbol derecho en Postorden y finalmente se visita el nodo raíz. Esto significa que para cada subárbol se debe conservar el recorrido en Postorden, es decir, primero se visita la parte izquierda, luego la parte derecha y por último la raíz.

Recorrido en Postorden



Recorrido en Postorden

El recorrido inicia con el subárbol izquierdo

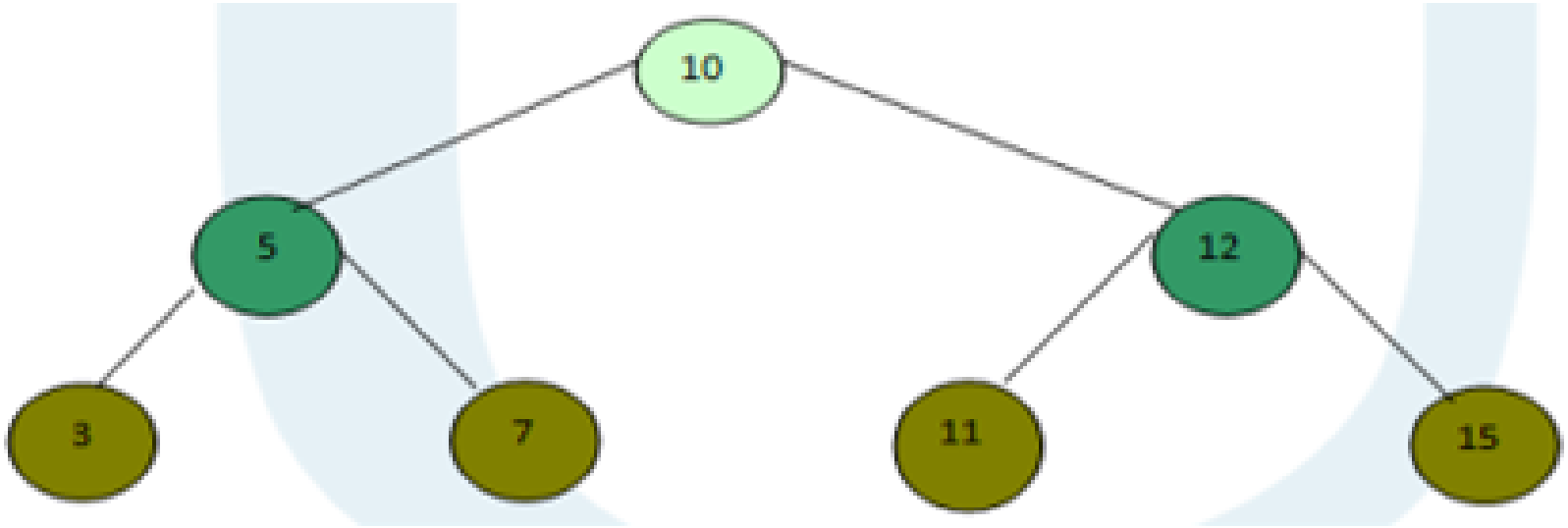
- El primer nodo a visitar es el 3
- Luego se visita el 7
- Posteriormente el 5

Finalizado el recorrido del subárbol izquierdo se inicia la visita al subárbol derecho en Postorden

- Se visita el 11
- Luego el 15
- Se finaliza con la visita del nodo 12

Solo queda recorrer la raíz del árbol que para este caso es el número 10.

Recorrido en Postorden



El resultado completo del recorrido en Postorden para el árbol de la imagen es:

3 – 7 – 5 – 11 – 15 – 12 - 10

Búsqueda dentro del árbol binario en Python

Para buscar, igualmente recorremos el árbol. Como estamos en un árbol binario que tiene los datos ya acomodados (pues lo hicimos al insertarlos) podemos usar esto a nuestro favor para hacer una búsqueda binaria y localizar la búsqueda en menos tiempo.

Eso sí, en caso de que el nodo que visitemos sea None, regresamos None, pues no encontramos lo que buscamos. Pero en caso de que sí lo encontremos, regresamos el nodo en donde está la búsqueda.

Públicos y privados

En Python, todos los elementos dentro de una clase serán "públicos", es decir, podrán ser consultados y/o modificados fuera de la clase. Esto aplica de igual forma tanto para elementos de clase como de la instancia. Basta con colocar ya sea, nuestra clase/instancia, punto y el elemento con el cual deseemos trabajar. Veamos un par de ejemplos.

```python
class User:

    def __init__(self, username):
        self.username = username

    def set_username(self, username):
        self.username = username

    def get_username(self):
        return self.username

cody = User('Cody')
print(cody.get_username())

cody.set_username('Cambio de nombre')
print(cody.get_username())
```

Públicos y privados

```
class User:

    def __init__(self, username, password):
        self.username = username
        self._password = password

    def set_username(self, username):
        self.username = username

    def get_username(self):
        return self.username

cody = User('Cody', 'password123')
print(cody._password)
```

¿Qué pasa si queremos definir un elemento como privado?
Por convención, se usa el guion bajo (_). El guion bajo debe colocarse como prefijo a nuestro elemento, de esta forma le indicamos a cualquier desarrollador que, ese elemento en particular debe tratarse como privado, y no debe ser expuesto ni modificado externamente.