5. Árboles Binarios de Búsqueda

Carlos Zerón Martínez

Universidad Nacional Autónoma de México

zeron@ciencias.unam.mx

Martes 1 de Diciembre de 2020

Introducción

- Almacena objetos en general que poseen al menos un atributo que permite establecer la comparación con otros objetos dentro del árbol que sean del mismo tipo (comparable).

- El atributo que establece el orden entre objetos se le denomina **clave de búsqueda** y por medio de él se realiza la búsqueda. Una vez que ha sido encontrado dentro del árbol un nodo con el valor requerido, tenemos acceso a todos los atributos del objeto.

Definición

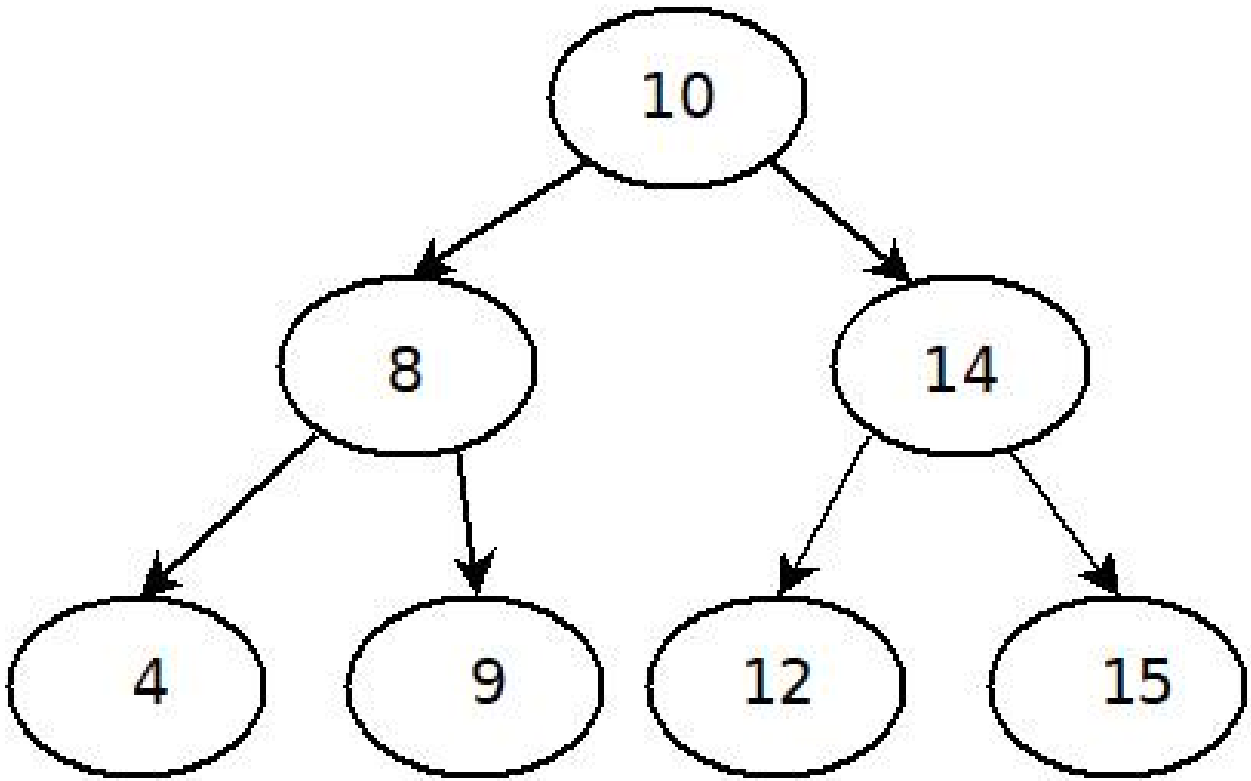
Un árbol binario de búsqueda es aquél que no tiene nodos (es vacío), o bien, tiene una raíz tal que:

- La clave de búsqueda almacenada en la raíz es mayor o igual que los valores almacenados en el subárbol izquierdo y menor que los valores almacenados en el subárbol derecho.

- Ambos subárboles son árboles binarios de búsqueda.

Cuando estas claves son únicas dentro del árbol, ambas propiedades cumplen las desigualdades estrictas.

Ejemplo de árbol binario de búsqueda



Comparación entre llaves

Para comparar cualesquiera dos llaves de búsqueda es necesario definir una relación de comparación $\leq$ que defina un orden total: es decir, una regla de comparación que sea válida para cualesquiera dos claves y satisfaga las propiedades siguientes:

- Reflexividad: $k \leq k$
- Antisimetria: si $k \leq k'$ y $k' \leq k$ entonces $k = k'$
- Transitividad: si $k \leq k'$ y $k' \leq k''$ entonces $k \leq k''$

Comparación entre llaves

En Java, la clase que representa un tipo de datos para la clave de búsqueda debe implementar la interfaz *java.lang.Comparable*, que consiste del método:

public int compareTo(Object o).

El resultado que devuelve este método es negativo si el objeto que invoca es *menor* que el objeto o, cero si son iguales y positivo si el objeto que invoca es *mayor* que o.
Ejemplos de clases que implementan la interfaz:

- java.math.BigDecimal
- java.math.BigInteger
- java.lang.Boolean
- java.lang.String
- java.util.Date
- java.io.File

TDA Arbol Binario de Búsqueda

Datos

Un conjunto de objetos, cada uno de los cuales tiene una lleva de tipo comparable.

Operaciones

- retrieve(k):Objeto. Devuelve un objeto con clave de búsqueda igual al objeto comparable k, en caso de existir. Si no existe se regresa un objeto nulo.

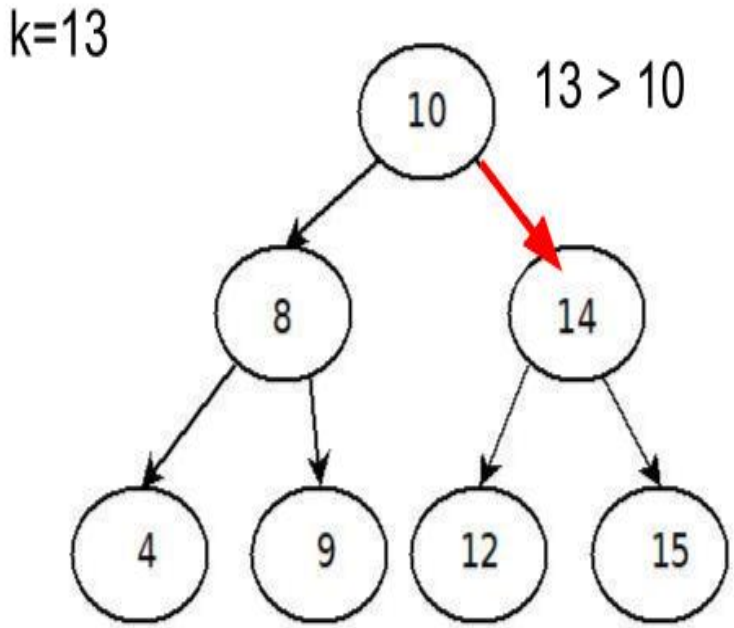
- insert(obj) : $\varnothing$. Inserta un objeto obj dentro del árbol. El objeto debe ser del mismo tipo que los almacenados en el árbol.

TDA Arbol Binario de Búsqueda

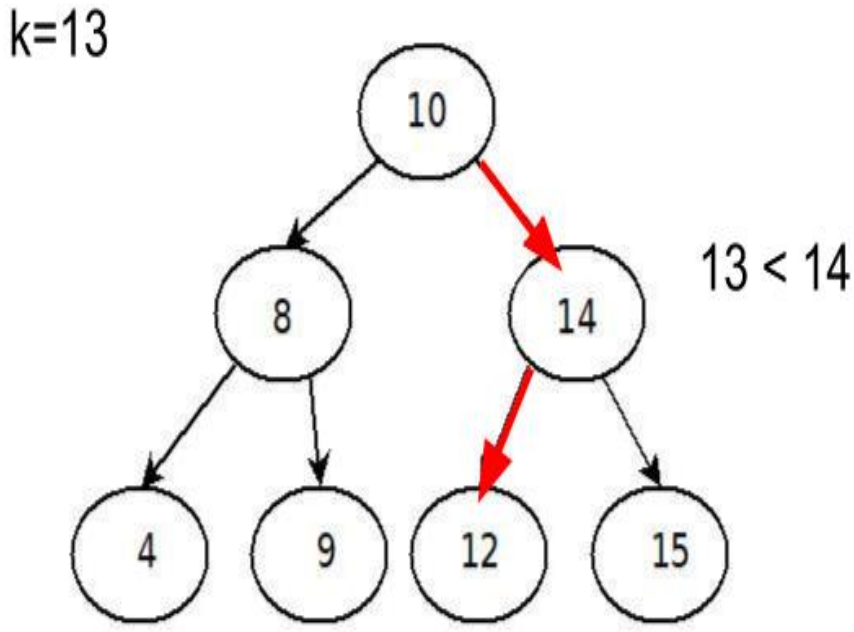
Operaciones

- delete(k) : *Objeto*. Elimina y devuelve un objeto dentro del árbol con clave de búsqueda igual al objeto comparable k y un objeto nulo si no existe la clave.
- findMin():Objeto. Devuelve un objeto dentro del árbol que tenga la mínima clave de búsqueda y un objeto nulo si no existe la clave.
- findMax():Objeto. Devuelve un objeto dentro del árbol que tenga la máxima clave de búsqueda y un objeto nulo si no existe la clave.

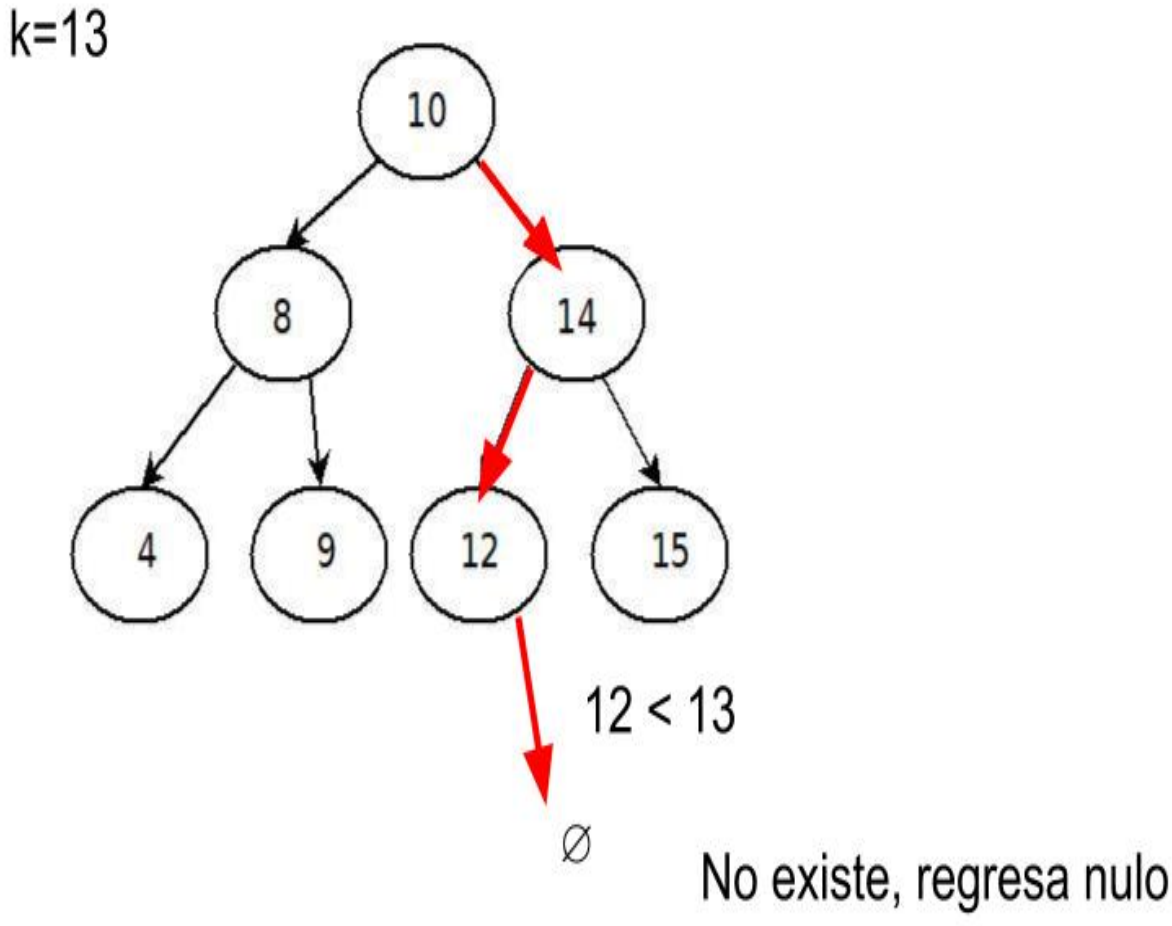
Esquema de búsqueda



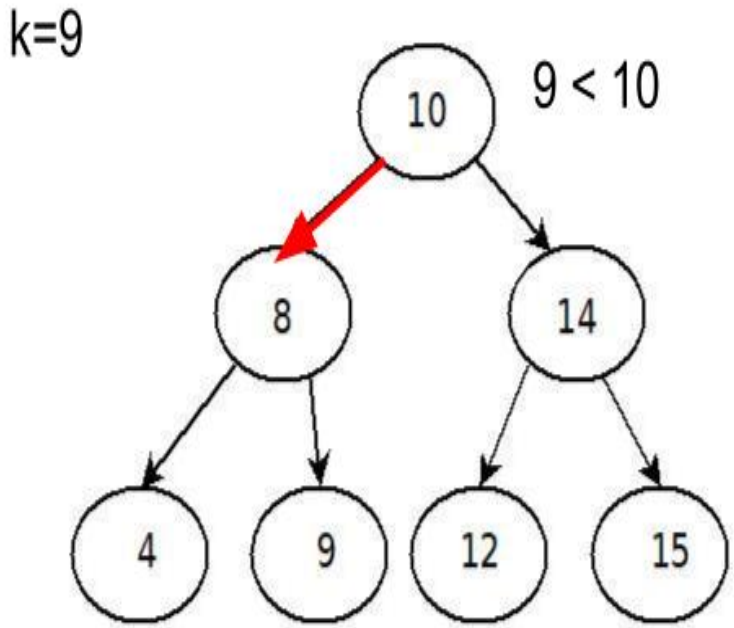
Esquema de búsqueda



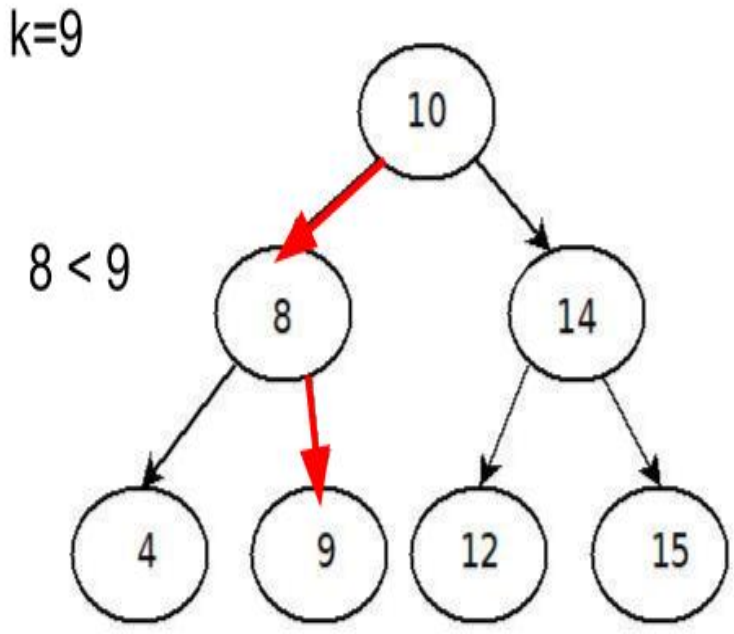
Esquema de búsqueda



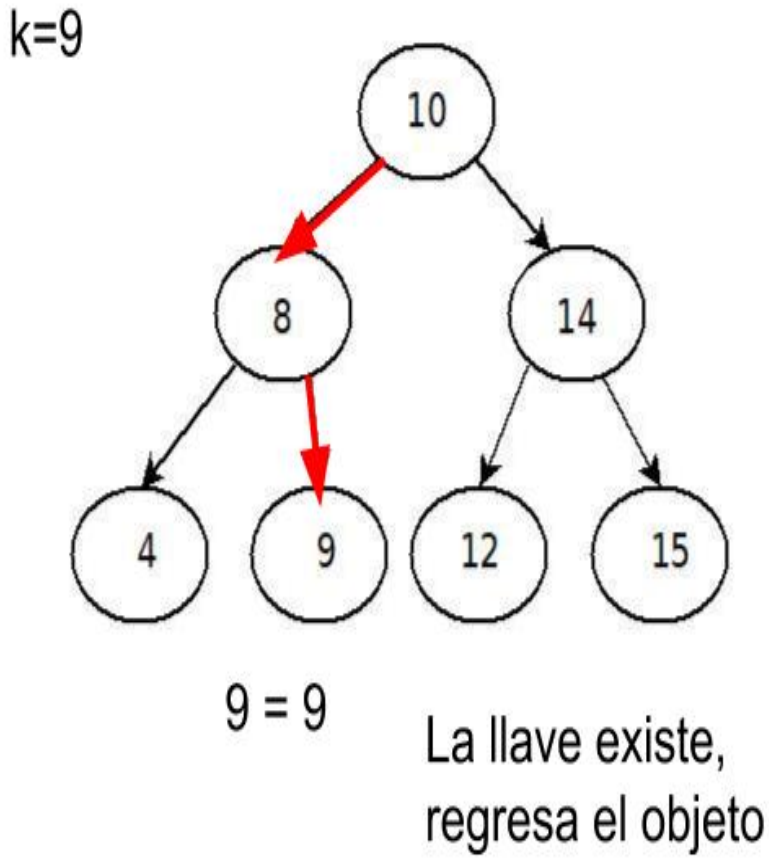
Esquema de búsqueda



Esquema de búsqueda

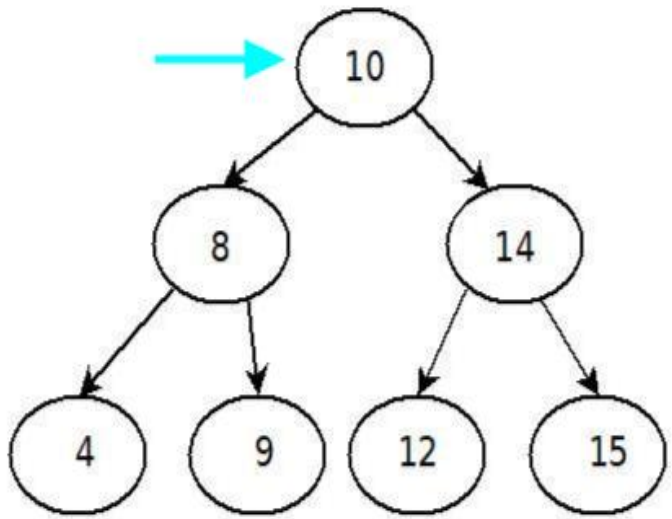


Esquema de búsqueda

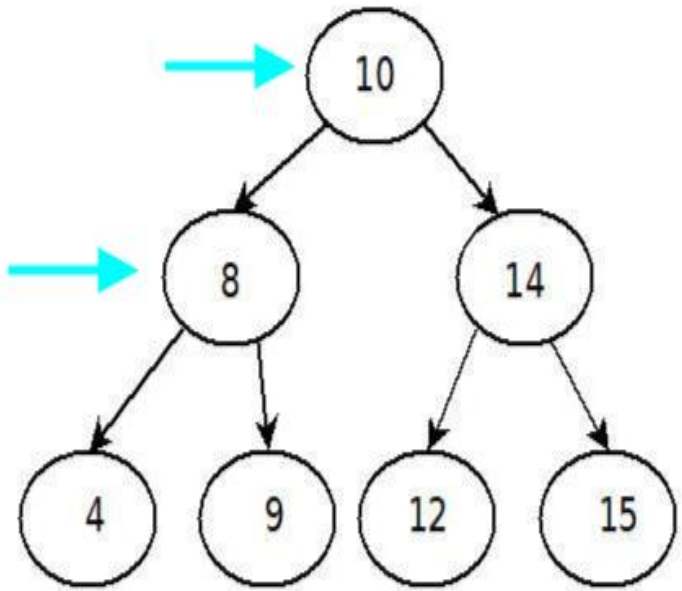


Ordenamiento de datos

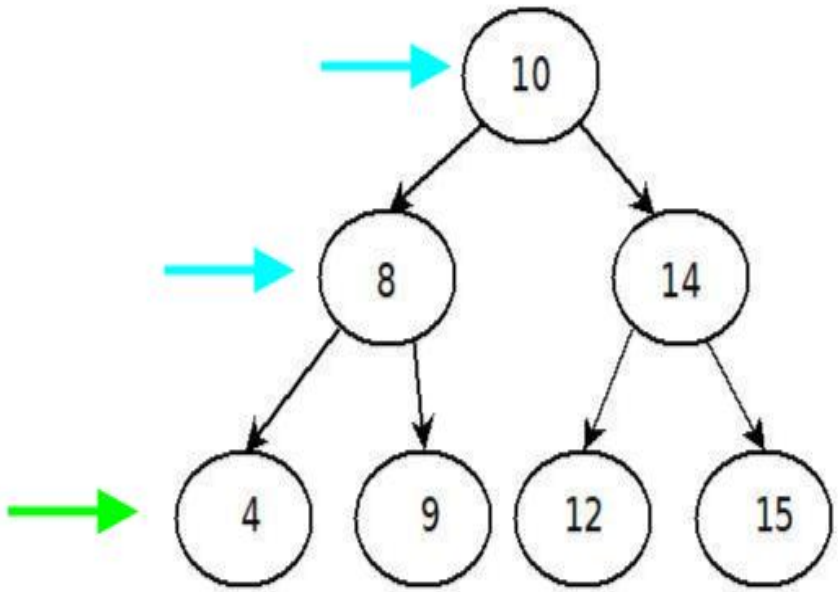
Simplemente aplicamos recorrido en inorden aprovechando la estructura del árbol binario de búsqueda, colocando los elementos en una lista en el orden en que se visitan.



Ordenamiento de datos

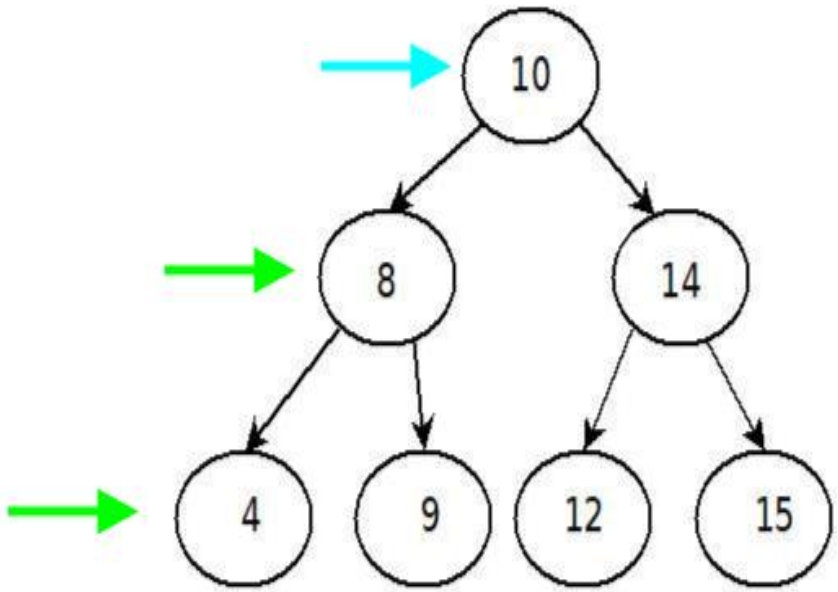


Ordenamiento de datos



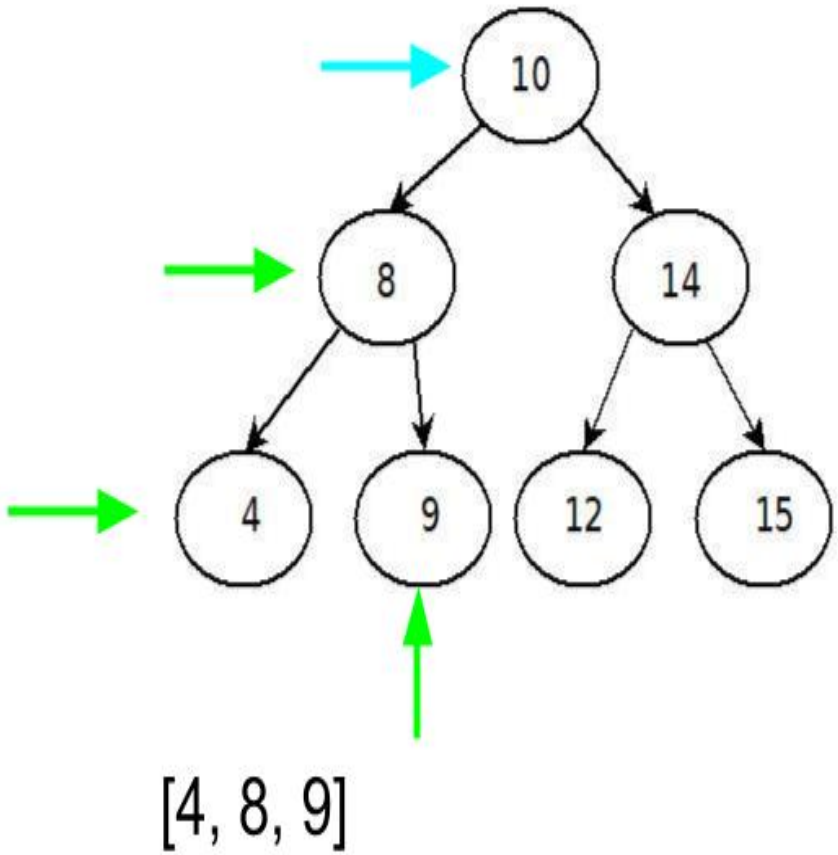
[4]

Ordenamiento de datos

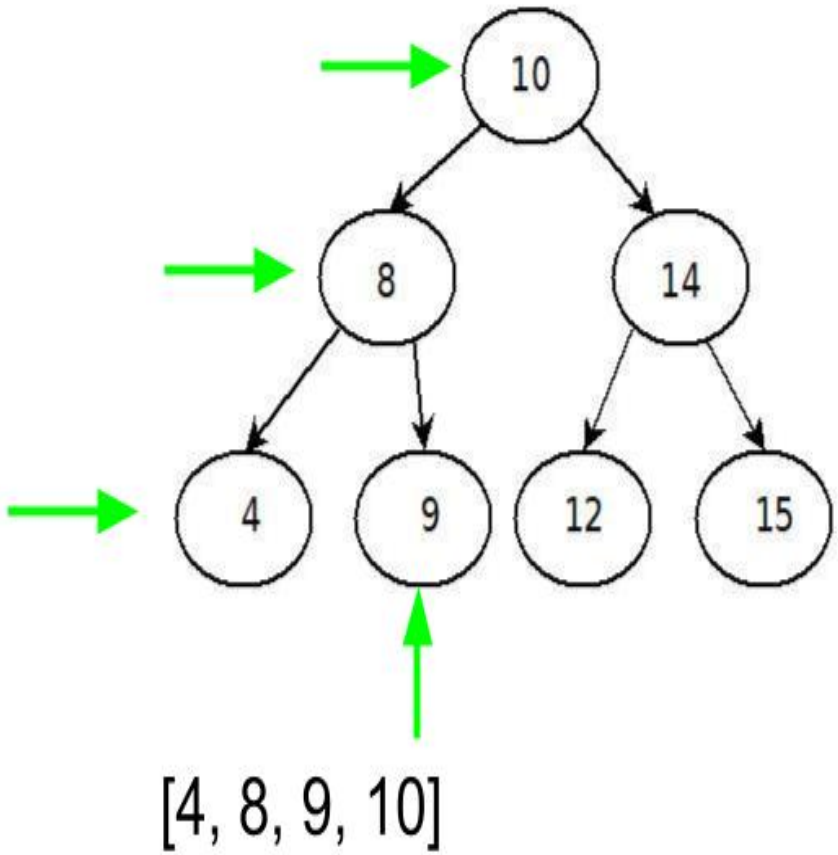


[4, 8]

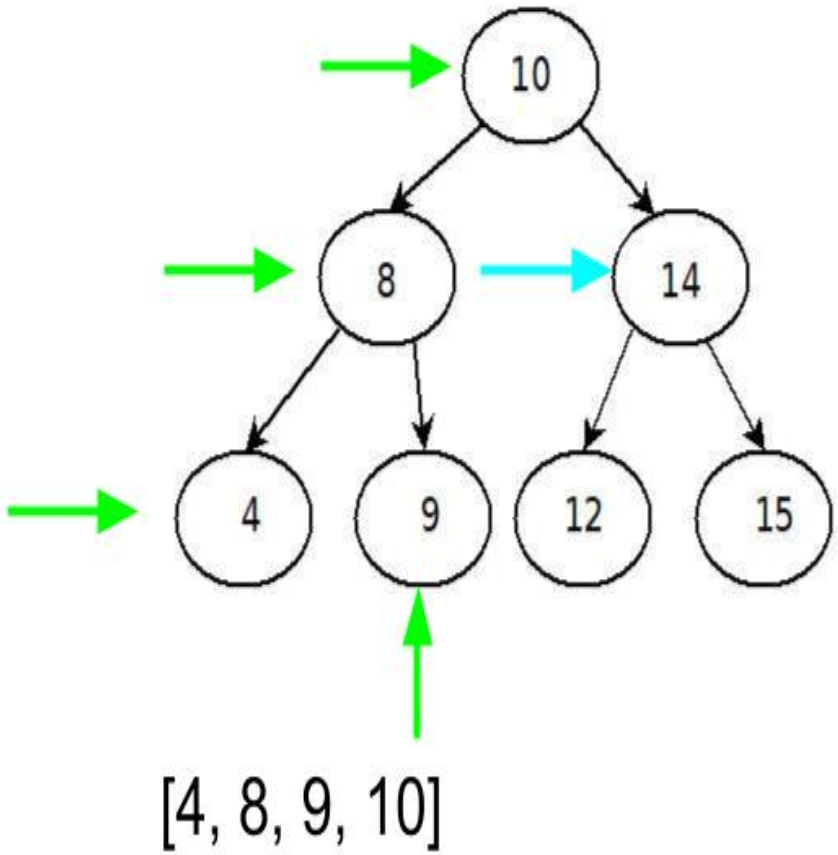
Ordenamiento de datos



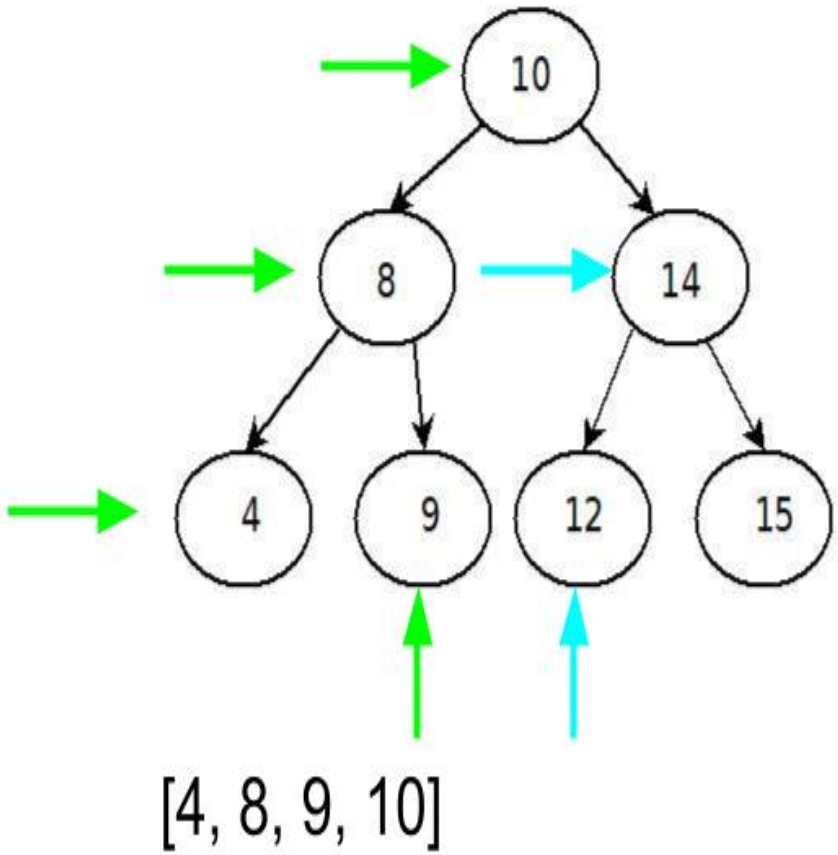
Ordenamiento de datos



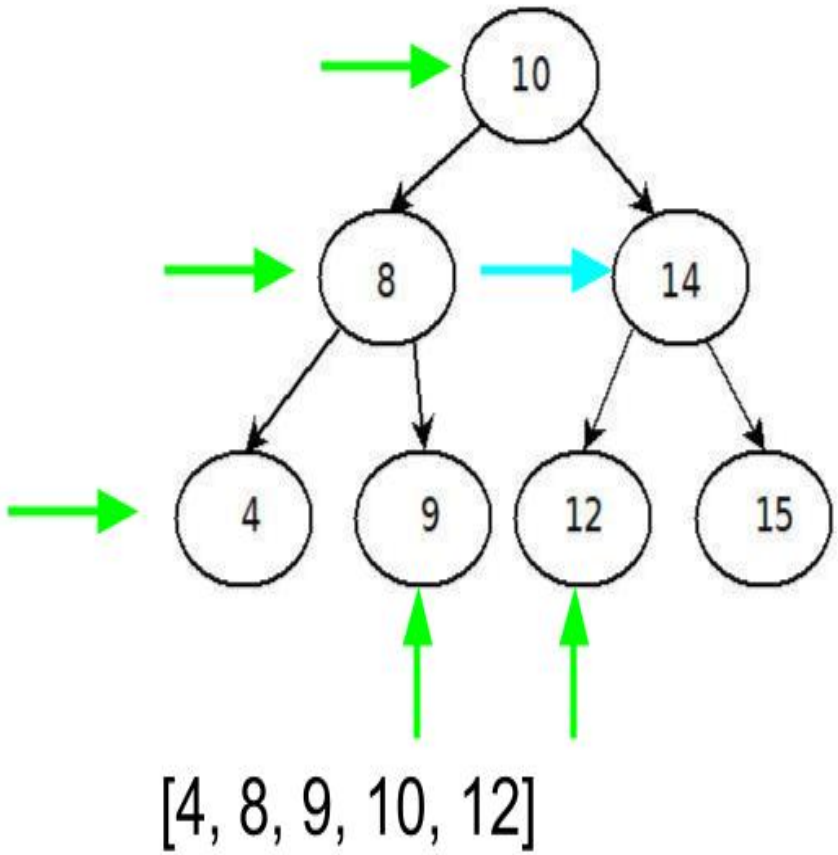
Ordenamiento de datos



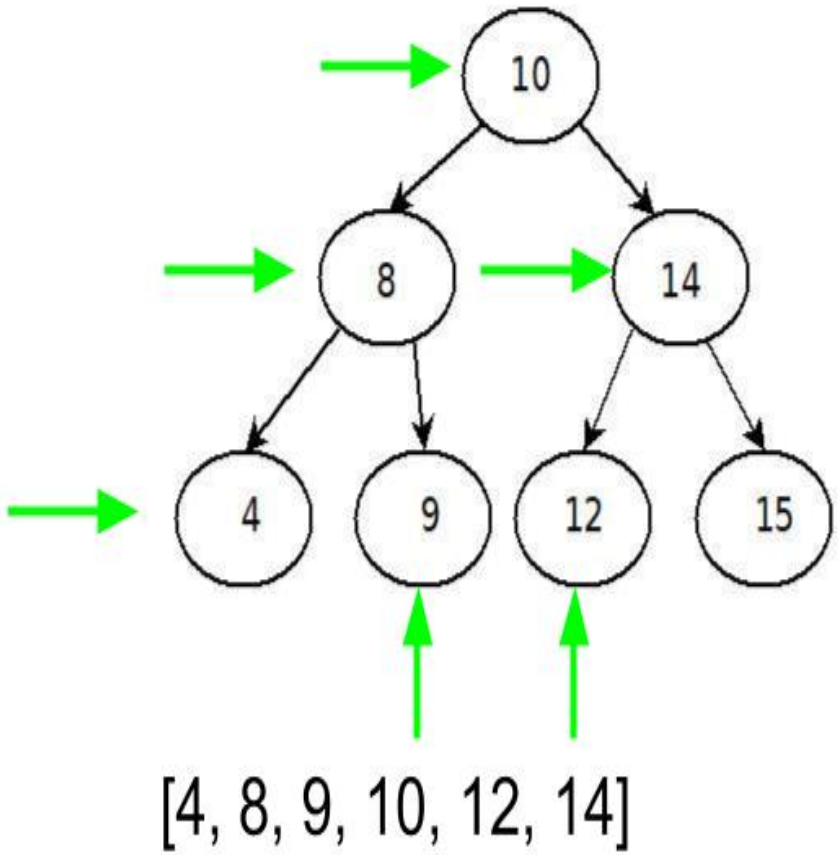
Ordenamiento de datos



Ordenamiento de datos



Ordenamiento de datos



Ordenamiento de datos