Universidad Nacional de San
Antonio Abad del Cusco

ARBOLES BINARIOS DE BUSQUEDA

Alumnos:

- Huaylla Huillca Rossbel
- Ninancuro Huarayo Diego Shaid
- Achahuanci Valenza Andree

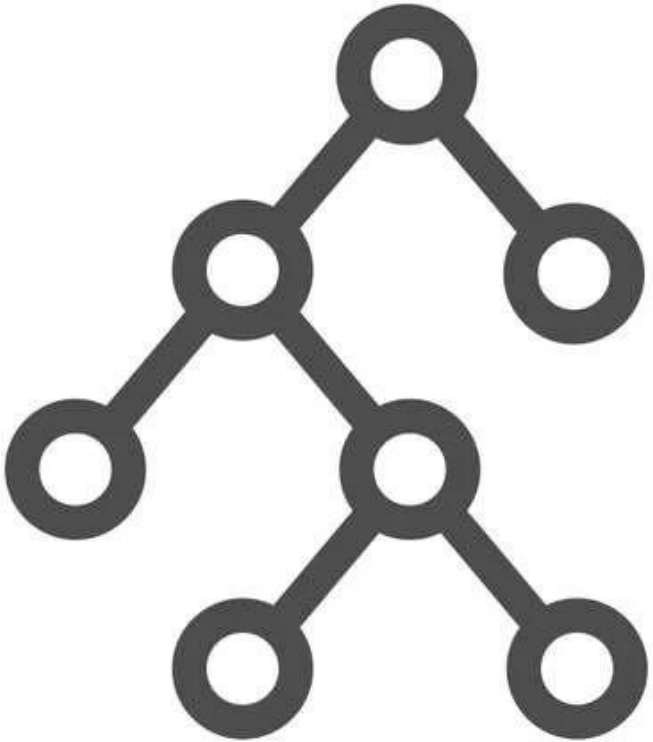
Introduccion

Los árboles binarios de búsqueda (ABB) son estructuras de datos que se utilizan para organizar y almacenar datos de manera jerárquica. Se caracterizan por tener una raíz, nodos internos y hojas, donde cada nodo puede tener como máximo dos hijos: uno izquierdo y otro derecho.

La propiedad clave que distingue a los árboles binarios de búsqueda es que para cada nodo:

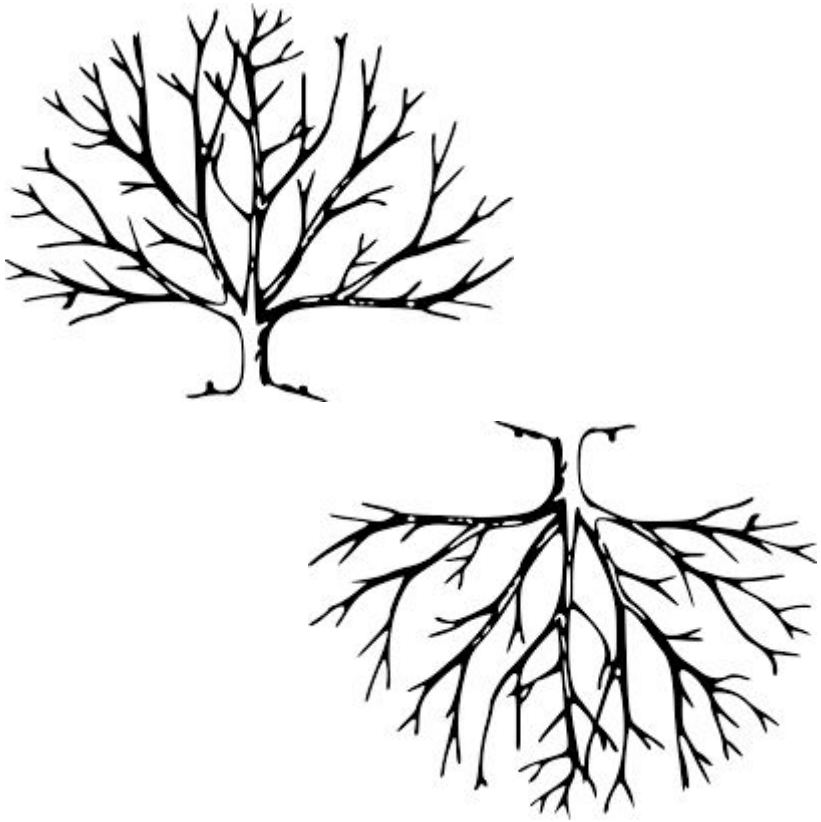
Todos los nodos en el subárbol izquierdo tienen valores menores que el valor del nodo actual.

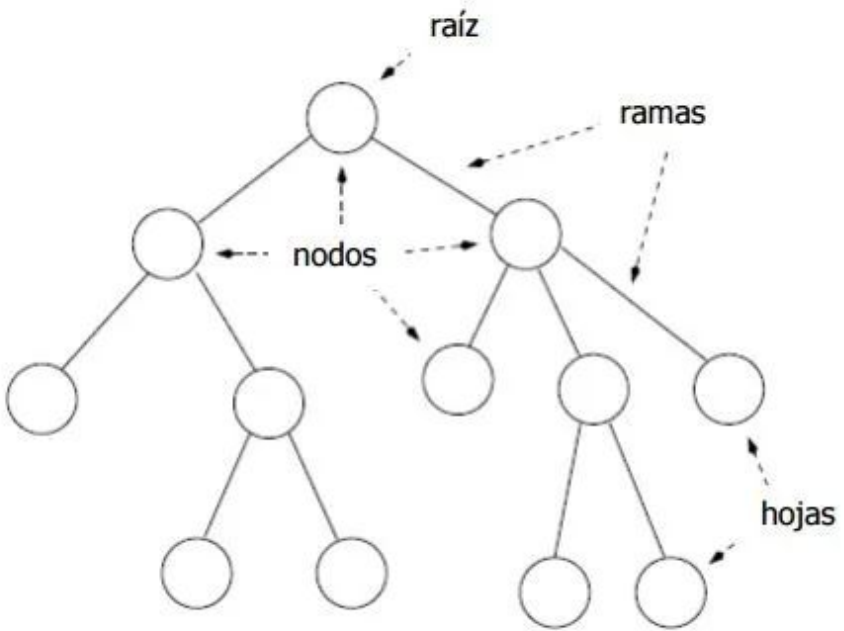
Todos los nodos en el subárbol derecho tienen valores mayores que el valor del nodo actual.



ÁRBOL

Un árbol es una estructura de datos no lineal que consta de un conjunto de nodos interconectados mediante enlaces llamados "aristas" o "ramas". Cada árbol tiene un nodo especial en la parte superior, conocido como "raíz", desde donde se ramifican todos los demás nodos. Los nodos que no tienen hijos se llaman "hojas".





Partes de un árbol

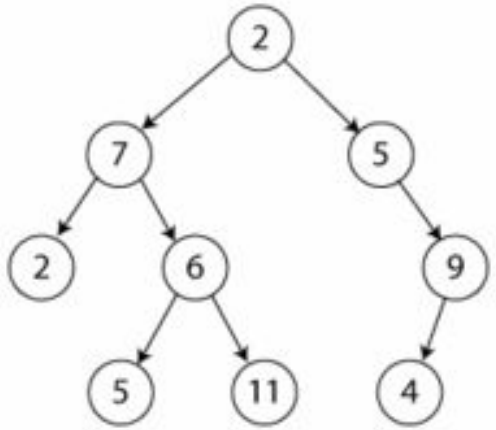
Además de las que se muestra en la imagen tenemos:

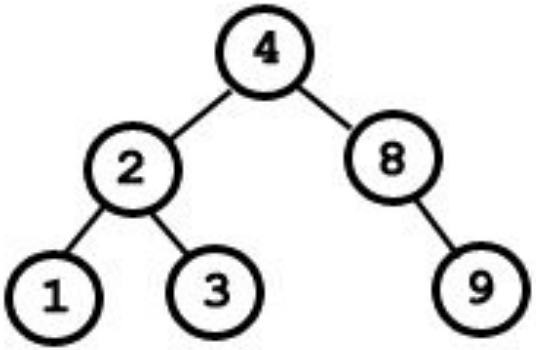
Nivel: La generación a la que pertenece un nodo desde la raíz. La raíz está en el nivel 0, sus hijos están en el nivel 1, los hijos de estos últimos están en el nivel 2 y así sucesivamente.

Altura: La altura de un árbol es la longitud máxima del camino desde la raíz hasta una hoja; es decir, el número máximo de niveles en el árbol.

Árbol binario

Es un árbol en el que de cada nodo cuelgan a los más dos nodos más.





Árbol binario de búsqueda

Es un árbol como el binario, solo que este cumple una condición importante:

Todos los elementos del lado izquierdo son menores que la raíz. Y los elementos del lado derecho son mayores

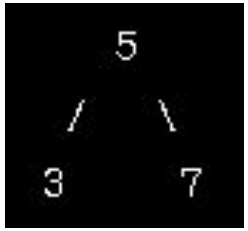
¿Cómo se forma un árbol binario de búsqueda?

Un árbol binario de búsqueda, de forma muy sencilla, se forma siguiendo una regla simple:

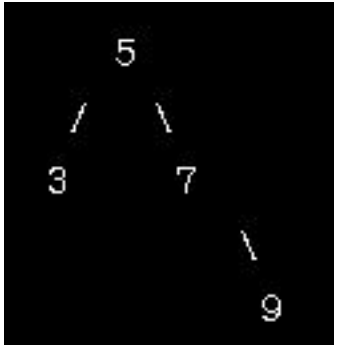
1) Si es menor que el nodo a comparar, va a la izquierda.
2) Si es mayor que el nodo a comparar, va a la derecha.

Agregamos una raíz

Agregamos un 7, al ser mayor que 5,va a la derecha



Agregamos un 3, al ser menor que 5, va a la derecha

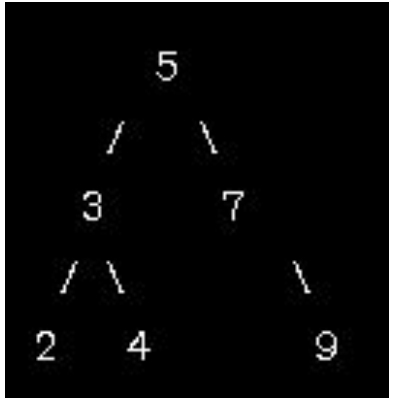


Agregamos un 9, al ser menor que 5, va a la derecha, luego se compara con 7 y repite el proceso

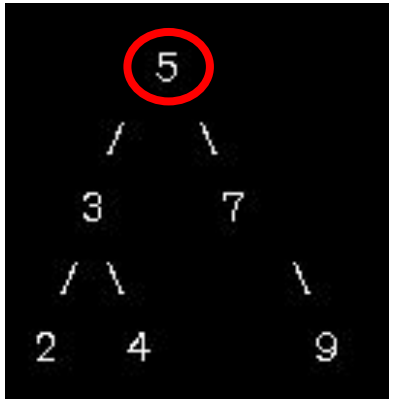
¿Cómo interpretamos un árbol binario de búsqueda?

Viendolo asi, no parece tener mucho sentido, pero para interpretarlo podemos ponerlo en una lista, siguiendo las siguientes condiciones:

I. Si hay nodo a la izquierda, pasar al nodo de la izquierda.
II. Si no hay o ya fue visitado, agregamos el nodo a la lista.
III. Si hay nodo a la derecha, pasamos al nodo de la derecha.
IV. Si no hay o ya fue visitado, regresamos al nodo anterior

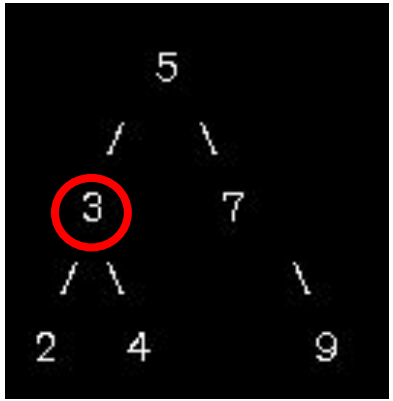


Haremos uso de este ABB



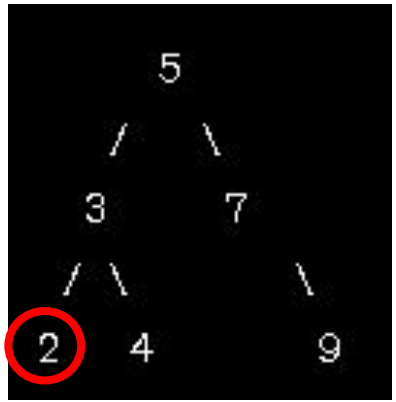
Revisamos la I.

Si tiene nodo a la izquierda.



Revisamos la I.

Si tiene nodo a la izquierda.

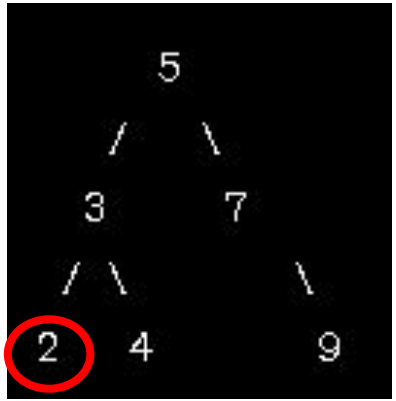


Revisamos la I.

No tiene nodo a la izquierda.

Entonces usamos la II. y agregamos a la lista.

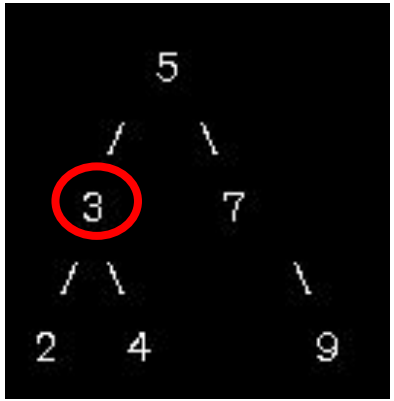
L = [2]



Comprobamos la III.

Comprobamos la IV.

Entonces regresamos al nodo anterior



Comprobamos la I.

Comprobamos la II.

Pero ya fue visitada, así que agregamos a la lista

L = [2,3]

IMPLEMENTACIÓN EN C#

```
using System;

8 referencias
class TreeNode
{
    public int Value;
    public TreeNode Left;
    public TreeNode Right;

    1 referencia
    public TreeNode(int value)
    {
        Value = value;
        Left = null;
        Right = null;
    }
}

3 referencias
class BinarySearchTree
{
    public TreeNode Root;

    1 referencia
    public BinarySearchTree()
    {
        Root = null;
    }

    5 referencias
    public void Insert(int value)
    {
        Root = InsertRecursive(Root, value);
    }
```

Este código implementa un árbol binario de búsqueda (BST) en C#. La clase TreeNode representa un nodo del árbol con su valor y referencias a los nodos izquierdo y derecho. La clase BinarySearchTree contiene los métodos para insertar y buscar elementos en el árbol. En el ejemplo de Main, se crea un árbol, se insertan algunos valores y se realiza una búsqueda para verificar si un valor dado está presente en el árbol.

```
    private TreeNode InsertRecursive(TreeNode root, int value)
    {
        if (root == null)
        {
            return new TreeNode(value);
        }

        if (value < root.Value)
        {
            root.Left = InsertRecursive(root.Left, value);
        }
        else if (value > root.Value)
        {
            root.Right = InsertRecursive(root.Right, value);
        }

        return root;
    }

    2 referencias
    public bool Search(int value)
    {
        return SearchRecursive(Root, value);
    }

    3 referencias
    private bool SearchRecursive(TreeNode root, int value)
    {
        if (root == null)
        {
            return false;
        }
        if (value == root.Value)
        {
            return true;
        }
        else if (value < root.Value)
        {
            return SearchRecursive(root.Left, value);
        }
        else
        {
            return SearchRecursive(root.Right, value);
        }
    }
}

0 referencias
class Program
{
    0 referencias
    static void Main(string[] args)
    {
        BinarySearchTree tree = new BinarySearchTree();

        tree.Insert(5);
        tree.Insert(3);
        tree.Insert(8);
        tree.Insert(1);
        tree.Insert(4);

        Console.WriteLine(tree.Search(4));  // Output: True
        Console.WriteLine(tree.Search(6));  // Output: False
    }
}
```

Conclusiones

Los árboles binarios de búsqueda son estructuras jerárquicas que ofrecen eficiencia en operaciones de búsqueda, inserción y eliminación cuando están equilibrados.

Son ampliamente utilizados en aplicaciones de búsqueda, como diccionarios y bases de datos, debido a su capacidad para realizar búsquedas rápidas.

Los recorridos inorden, preorden y postorden permiten acceder a los elementos del árbol en diferentes secuencias y son útiles para procesar los datos en orden ascendente o descendente.

Los árboles AVL y los árboles rojo-negro son variantes que garantizan un equilibrio y un rendimiento constante en sus operaciones, adecuados para aplicaciones que requieren alta eficiencia.

Para conjuntos de datos con distribuciones desequilibradas, otras estructuras de datos como tablas hash o árboles B pueden ser más adecuadas para obtener un mejor rendimiento.

Bibliografia

- Árbol binario de búsqueda - Wikipedia, la enciclopedia libre
- 6.14. Implementación de un árbol de búsqueda — Solución de problemas con algoritmos y estructuras de datos (runestone.academy)
- Arbol binario de búsqueda - EcuRed