Universidad ICESI

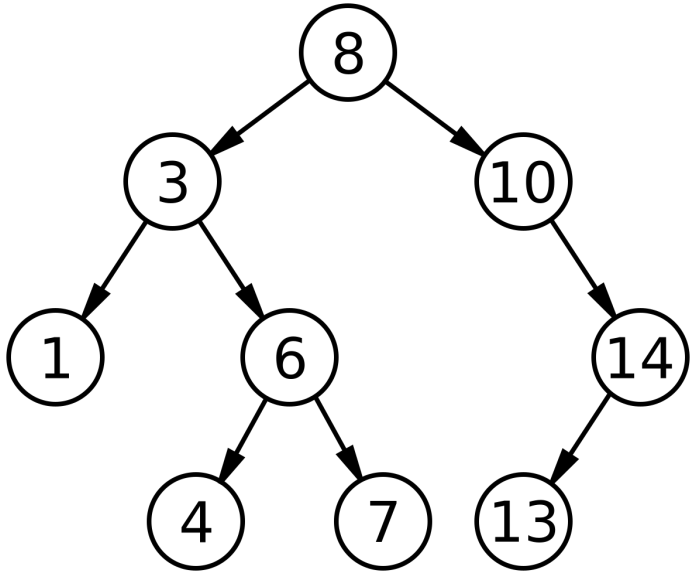
Árboles binarios de búsqueda - BST

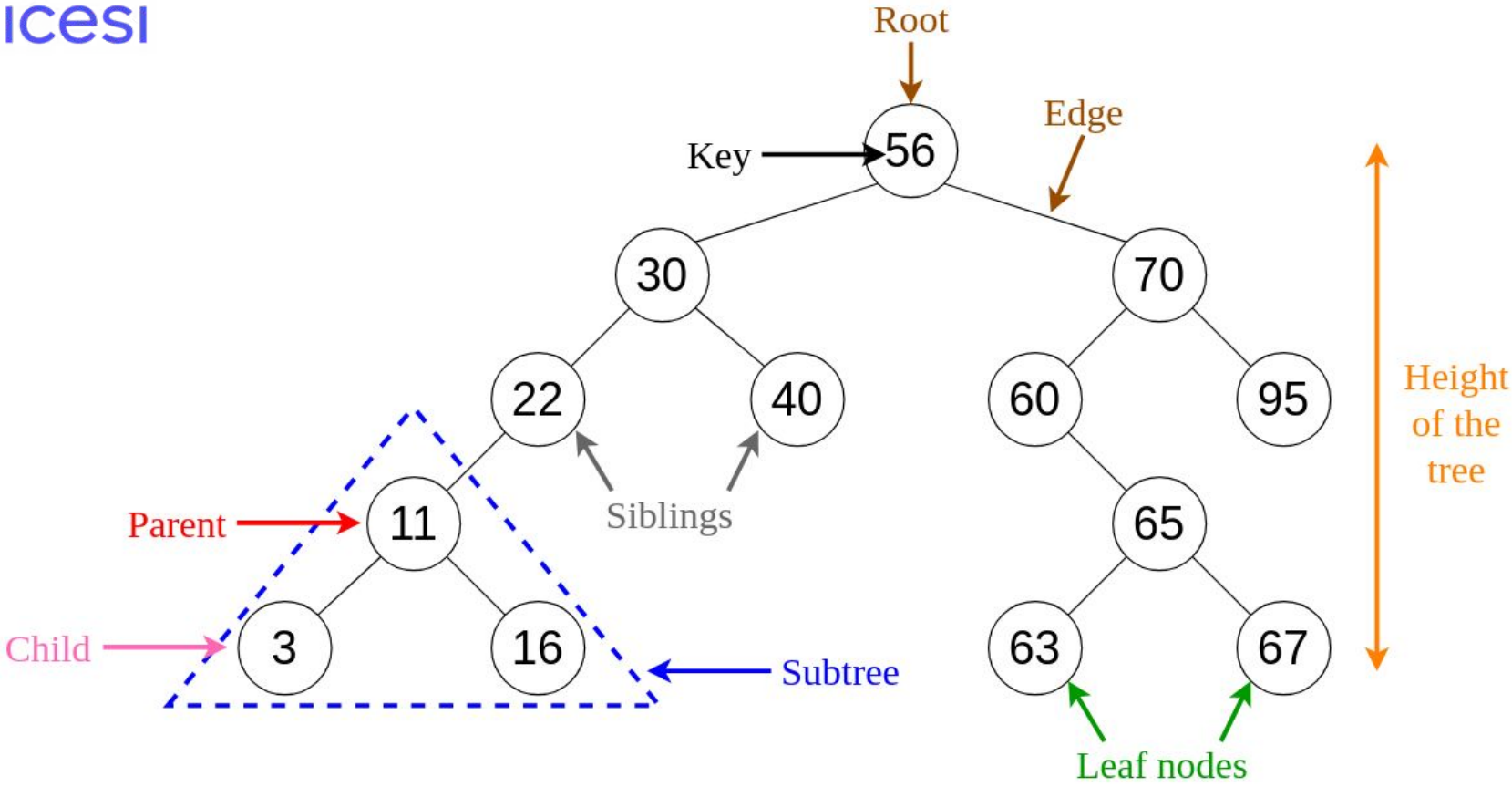
Esta presentación proporciona una exploración en el concepto de árboles binarios, su complemento con la recursividad. Cubre ejemplos y técnicas.

¿Qué son los BST?

Un árbol de búsqueda binaria es una **estructura de datos no lineal** que es utilizada para organizar y almacenar datos de forma ordenada.

Es importante distinguir la composición completa de un árbol y cada una de sus partes. Esta estructura jerárquica permite operaciones eficientes de búsqueda, inserción y eliminación de los datos almacenados en el árbol.



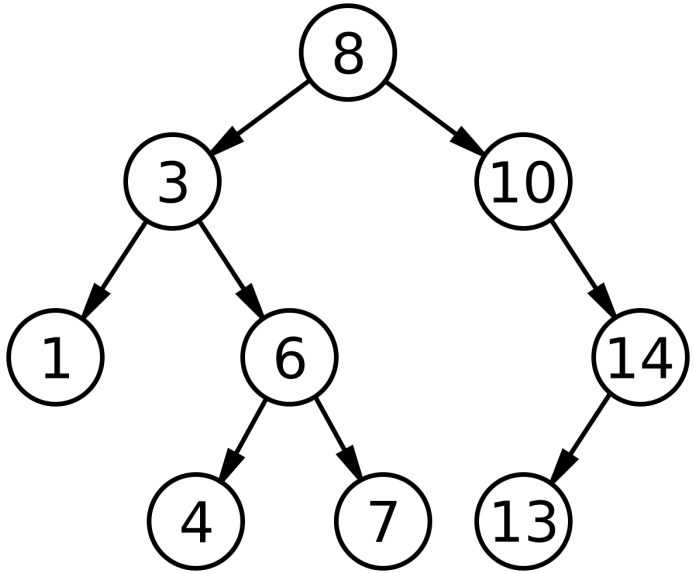


BST y sus partes. tomado de: https://miro.medium.com/v2/resize:fit:1194/1*ziYvZzrttFYMXkkV9u66jw.png

Propiedades BST

El árbol de búsqueda binaria es una estructura de datos de árbol binario basada en nodos que tiene las siguientes propiedades:

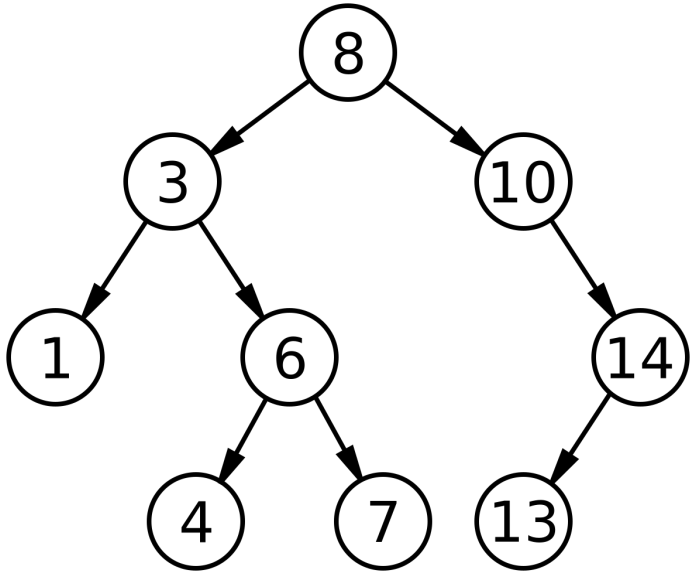
1. El subárbol izquierdo de un nodo contiene sólo nodos con claves menores que la clave del nodo.
2. El subárbol derecho de un nodo contiene sólo nodos con claves mayores que la clave del nodo.



Propiedades BST

Esto significa que todo lo que está a la izquierda de la raíz es menor que el valor de la raíz y todo lo que está a la derecha de la raíz es mayor que el valor de la raíz. Gracias a esta realización, la búsqueda binaria es muy sencilla.

1. Los subárboles izquierdo y derecho también deben ser un árbol de búsqueda binario.
2. No debe haber nodos duplicados (BST puede tener valores duplicados con diferentes enfoques de manejo)

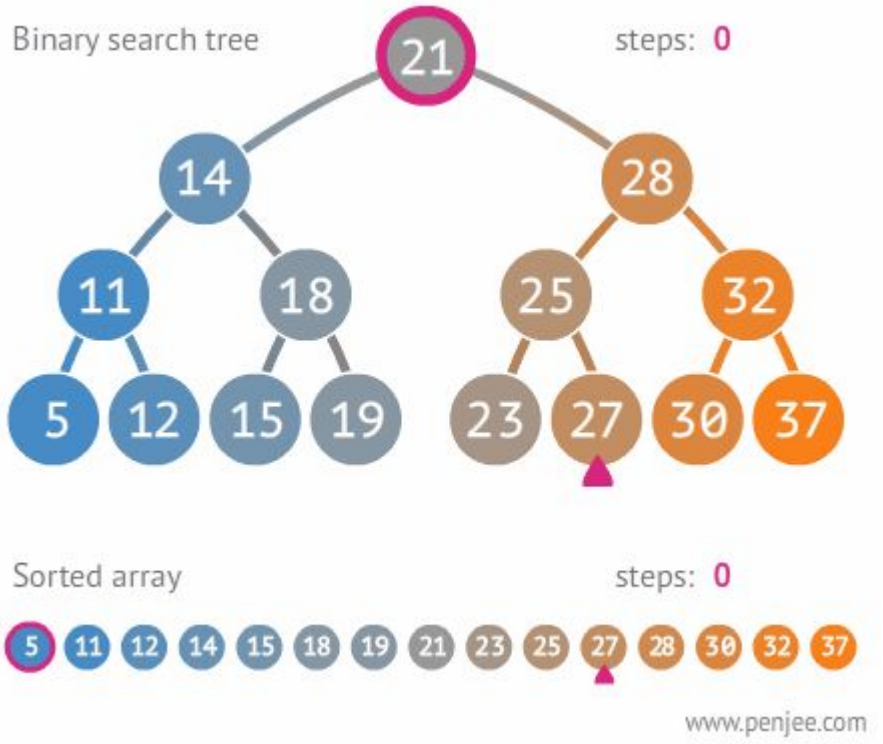


Complejidad Algorítmica

ALGORITHM	AVERAGE CASE	WORST CASE
Space	O(n)	O(n)
Insert	O(log n)	O(n)
Search	O(log n)	O(n)
Traverse	O(n)	O(n)

Complejidad algorítmica. tomado de:
https://static.studytonight.com/data-structures/images/binary-search-tree-4.png

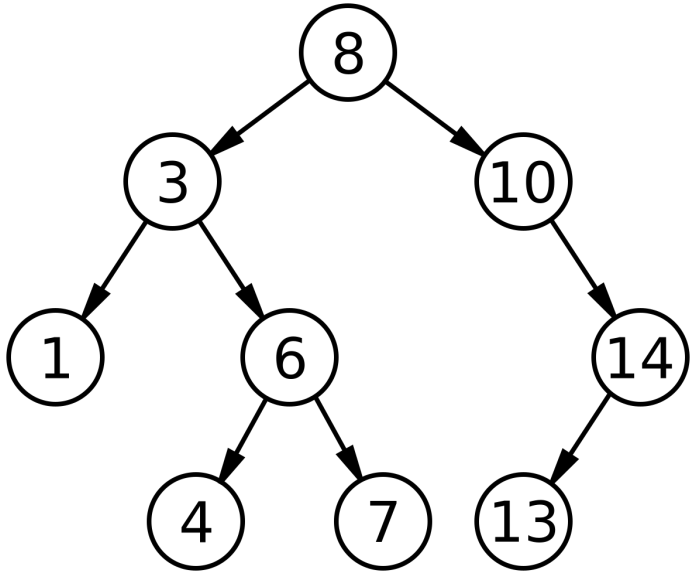
Complejidad Algorítmica



Búsqueda BST vs Búsqueda en Array https://upload.wikimedia.org/wikipedia/commons/9/9b/Binary_search_tree_example.gif

Ventajas de usar BST

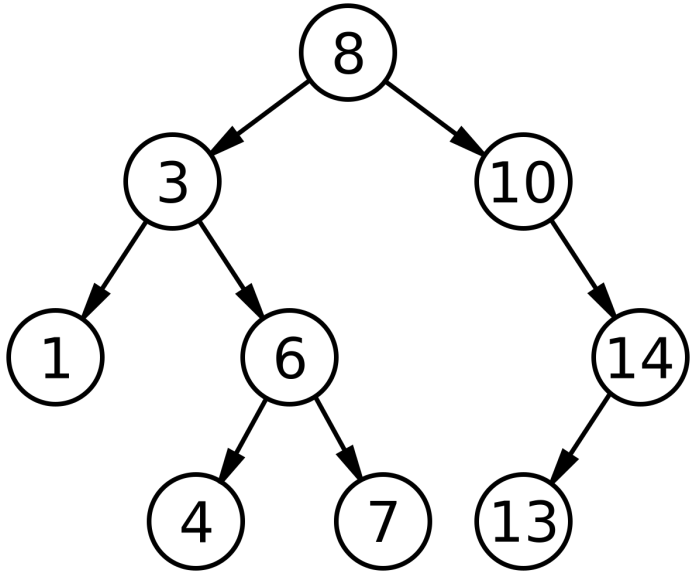
1. Búsqueda rápida: la búsqueda de un valor específico en un BST tiene una complejidad de tiempo promedio de O (log n), donde n es el número de nodos en el árbol. Esto es mucho más rápido que buscar un elemento en una matriz o lista enlazada, que tiene una complejidad temporal de O(n) en el peor de los casos.



Ventajas de usar BST

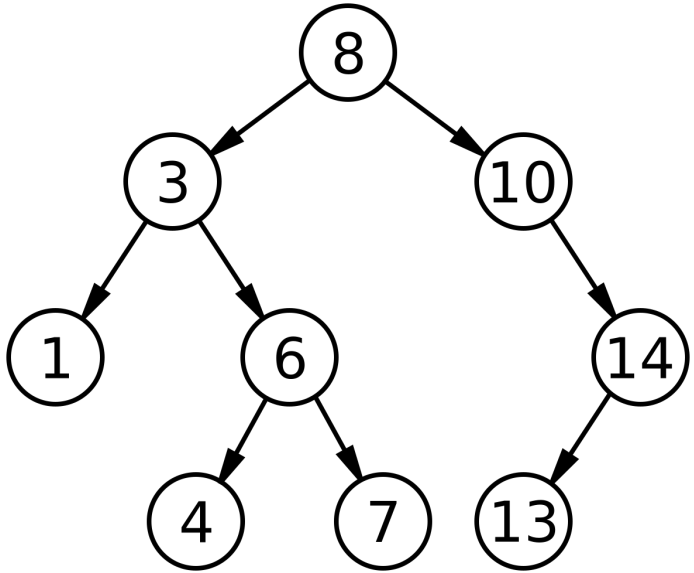
2. Recorrido en orden: los BST se pueden recorrer en orden, lo que visita el subárbol izquierdo, la raíz y el subárbol derecho. Esto se puede utilizar para ordenar un conjunto de datos.

3. Ahorro de espacio: los BST ahorran espacio ya que no almacenan información redundante, a diferencia de las matrices y las listas vinculadas.

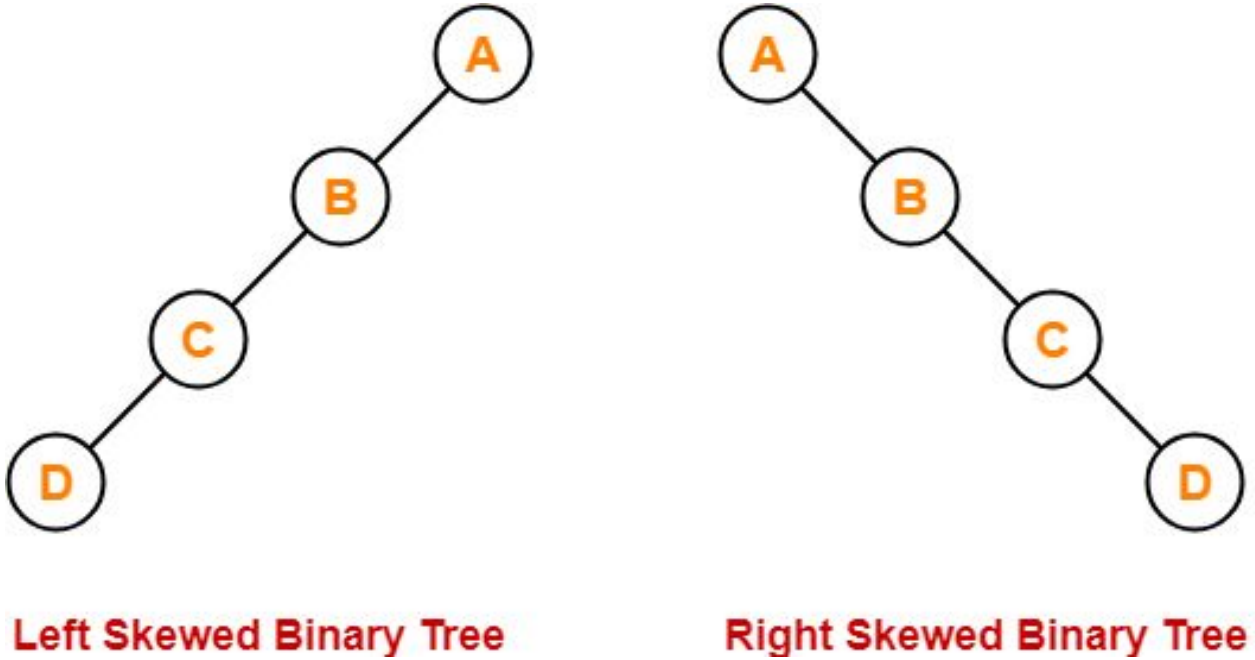


Desventajas de usar BST

1. Árboles torcidos/sesgados: si un árbol es torcido, la complejidad temporal de las operaciones de búsqueda, inserción y eliminación será O (n) en lugar de O (log n), lo que puede hacer que el árbol sea ineficiente.
2. Eficiencia: las BST no son eficientes para conjuntos de datos con muchos duplicados, ya que desperdician espacio.



Árboles Sesgados



Ejemplo Skewed BST. Tomado de:
https://qph.cf2.quoracdn.net/main-qimg-84b0bd76300dbd9fc35d9320dd4296d0

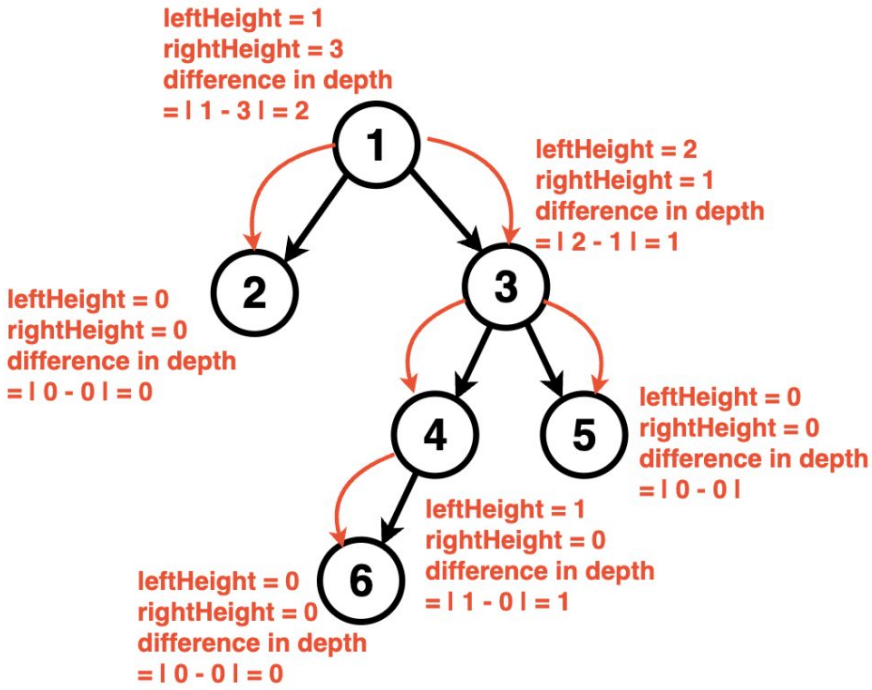
BST: Balanceo

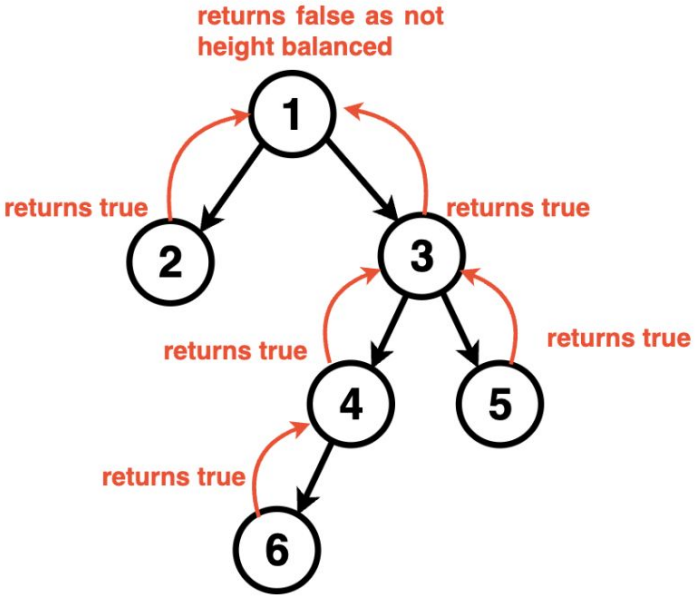
Un árbol binario balanceado, se define como un árbol binario en el que la altura de los subárboles izquierdo y derecho de cualquier nodo difiere en no más de 1.

Las siguientes son las condiciones para un árbol binario con altura equilibrada:

1. La diferencia entre el subárbol izquierdo y derecho para cualquier nodo no es más de uno.
2. el subárbol izquierdo está equilibrado
3. el subárbol derecho está equilibrado

Balanceo





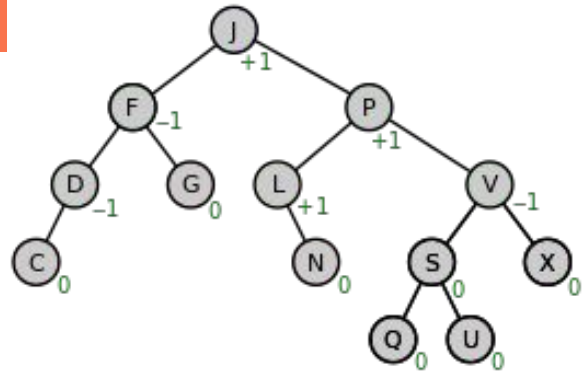
https://takeuforward.org/data-structure/check-if-the-binary-tree-is-balanced-binary-tree/

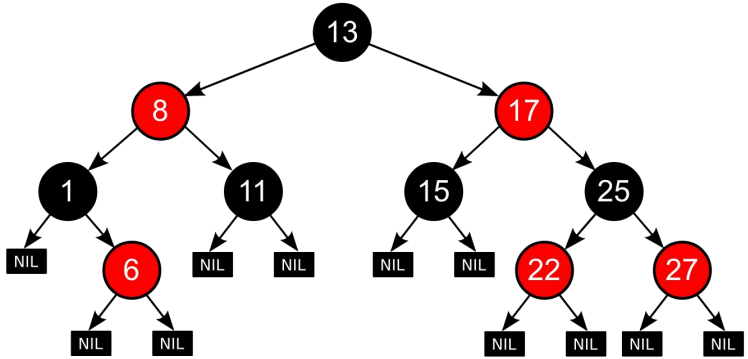
Caso Especial: Árboles Autobalanceados

Los BST auto balanceados son árboles de búsqueda binaria con altura equilibrada que mantienen automáticamente la altura lo más pequeña posible cuando se realizan operaciones de inserción y eliminación en el árbol.

Ejemplos: Los ejemplos más comunes de árboles de búsqueda binarios auto balanceados son:

1. Árbol AVL
2. Árbol negro rojo
3. Árbol de expansión

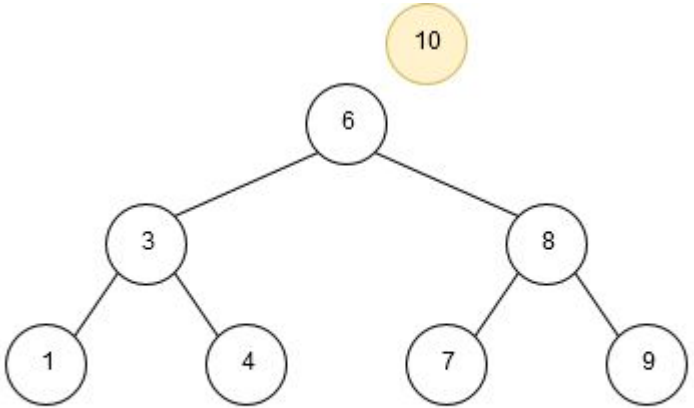




BST: Insertar

Siempre se inserta un nuevo valor en un nodo hoja manteniendo la propiedad del árbol de búsqueda binario. Se siguen los siguientes pasos mientras intentamos insertar un nodo en un árbol de búsqueda binario:

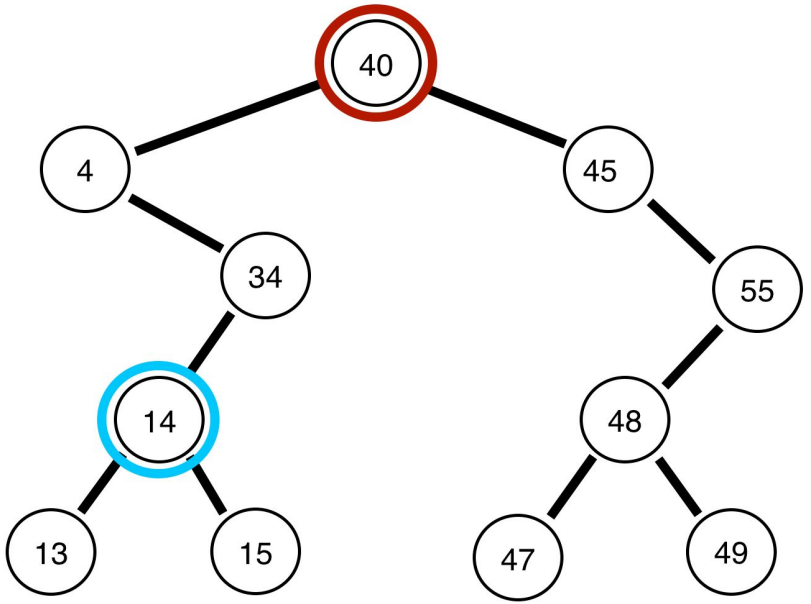
1. Verifique el valor a insertar (digamos X) con el valor del nodo actual (digamos val) en el que nos encontramos:
2. Si X es menor que val, muévase al subárbol izquierdo.
3. De lo contrario, vaya al subárbol derecho.
4. Una vez que se alcanza el nodo hoja, inserte X a su derecha o izquierda según la relación entre X y el valor del nodo hoja.



BST: Buscar

Para buscar un valor en BST, considéralo como un arreglo ordenada. Ahora podemos realizar fácilmente operaciones de búsqueda en BST usando el algoritmo de búsqueda binaria.

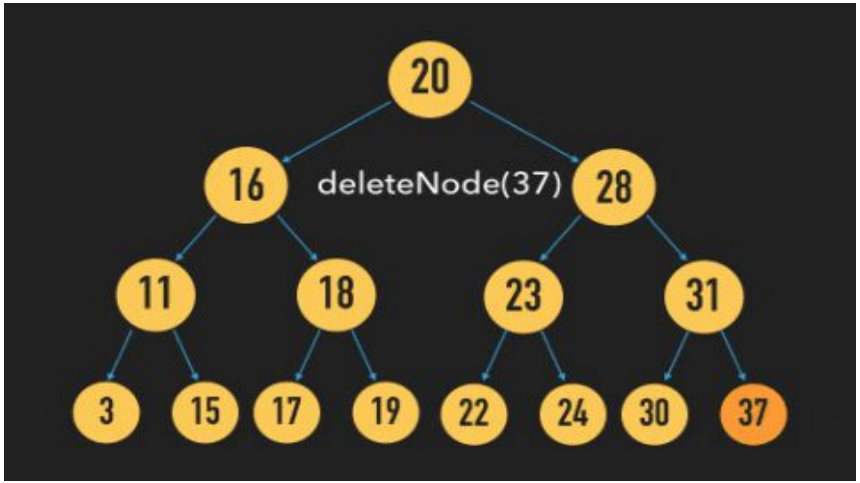
1. Comparamos el valor a buscar con el valor de la raíz.
2. Si es igual, hemos terminado con la búsqueda, si es más pequeño, sabemos que debemos ir al subárbol izquierdo porque en un árbol de búsqueda binario todos los elementos en el subárbol izquierdo son más pequeños y todos los elementos en el subárbol derecho son más grandes.
3. Repita el paso anterior hasta que no sea posible realizar más recorridos.
4. Si en cualquier iteración se encuentra el valor buscado, devuelve True. De lo contrario Falso.



BST: Eliminar Caso 1

Dado un BST, la tarea es eliminar un nodo, lo que se puede dividir en 3 escenarios:

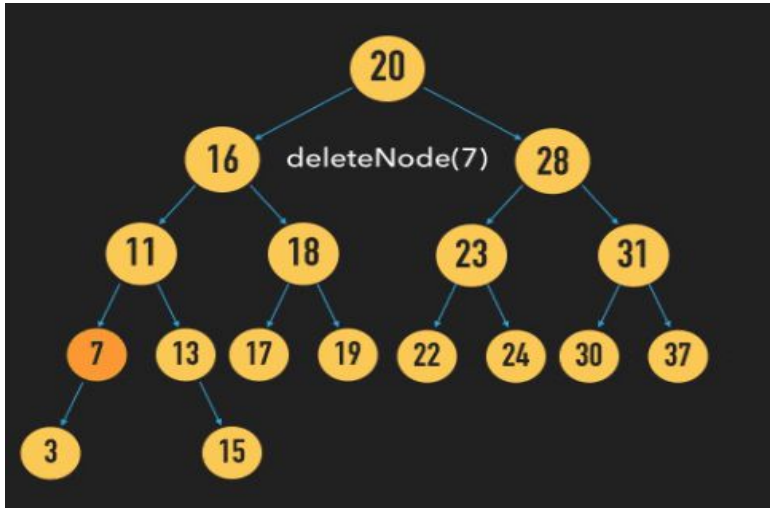
1. El nodo a eliminar es un nodo hoja
2. **El nodo a eliminar tiene un solo hijo**
    a. **El nodo tiene un solo hijo derecho**
    b. **El nodo tiene un solo hijo izquierdo**
3. El nodo a eliminar tiene dos hijos
4. El nodo a eliminar es la raíz



BST: Eliminar Caso 2

Dado un BST, la tarea es eliminar un nodo, lo que se puede dividir en 3 escenarios:

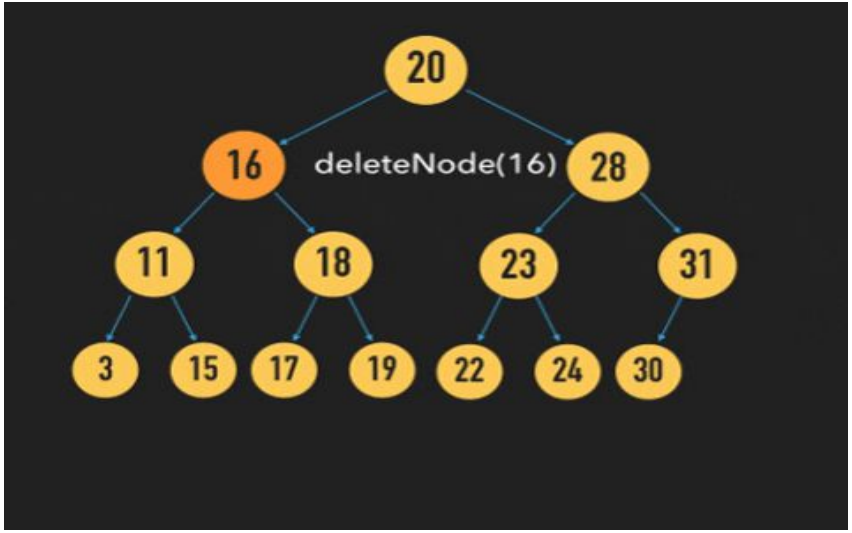
1. El nodo a eliminar es un nodo hoja
2. **El nodo a eliminar tiene un solo hijo**
    a. **El nodo tiene un solo hijo derecho**
    b. **El nodo tiene un solo hijo izquierdo**
3. El nodo a eliminar tiene dos hijos
4. El nodo a eliminar es la raíz



BST: Eliminar Caso 3

Dado un BST, la tarea es eliminar un nodo, lo que se puede dividir en 3 escenarios:

1. El nodo a eliminar es un nodo hoja
2. **El nodo a eliminar tiene un solo hijo**
    a. **El nodo tiene un solo hijo derecho**
    b. **El nodo tiene un solo hijo izquierdo**
3. El nodo a eliminar tiene dos hijos
4. El nodo a eliminar es la raíz

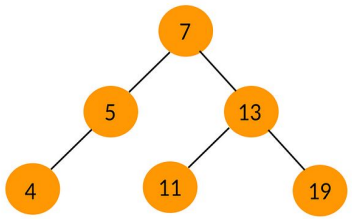


BST: Eliminar Caso 4

Dado un BST, la tarea es eliminar un nodo, lo que se puede dividir en 3 escenarios:

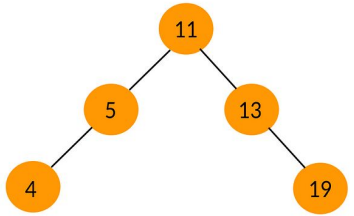
1. El nodo a eliminar es un nodo hoja
2. El nodo a eliminar tiene un solo hijo
    a. El nodo tiene un solo hijo derecho
    b. El nodo tiene un solo hijo izquierdo
3. El nodo a eliminar tiene dos hijos
4. **El nodo a eliminar es la raíz**

Before deleting 7



Inorder : 4 5 7 11 13 19

After deleting 7

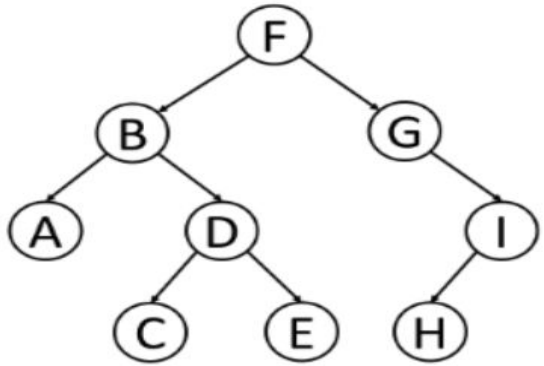


Inorder : 4 5 11 13 19

BST: Recorrido InOrder

Dado un BST, la tarea es recorrer el árbol, lo que se puede hacer de 3 formas:

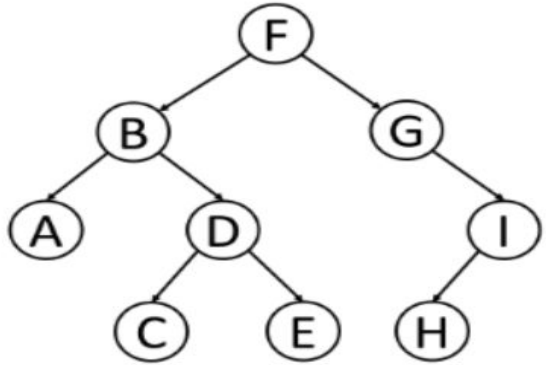
1. **Inorder - El recorrido en orden del BST proporciona los valores de los nodos en orden.**
2. Preorder - Este orden asegura que los nodos se visitan de manera "de arriba hacia abajo", comenzando desde la raíz y moviéndose hacia las hojas.
3. Postorder - Este orden asegura que los nodos sean visitados de manera "de abajo hacia arriba", comenzando desde las hojas y moviéndose hacia la raíz.



BST: Recorrido PreOrder

Dado un BST, la tarea es recorrer el árbol, lo que se puede hacer de 3 formas:

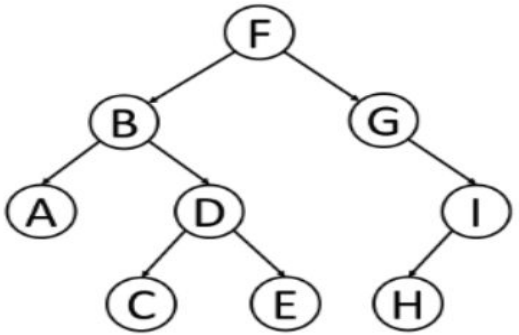
1. Inorder - El recorrido en orden del BST proporciona los valores de los nodos en orden.
2. **Preorder - Este orden asegura que los nodos se visitan de manera "de arriba hacia abajo", comenzando desde la raíz y moviéndose hacia las hojas.**
3. Postorder - Este orden asegura que los nodos sean visitados de manera "de abajo hacia arriba", comenzando desde las hojas y moviéndose hacia la raíz.



BST: Recorrido PostOrder

Dado un BST, la tarea es recorrer el árbol, lo que se puede hacer de 3 formas:

1. Inorder - El recorrido en orden del BST proporciona los valores de los nodos en orden.
2. Preorder - Este orden asegura que los nodos se visitan de manera "de arriba hacia abajo", comenzando desde la raíz y moviéndose hacia las hojas.
3. **Postorder - Este orden asegura que los nodos sean visitados de manera "de abajo hacia arriba", comenzando desde las hojas y moviéndose hacia la raíz.**



Caso de estudio

https://drive.google.com/file/d/1GxdMJ_kewtLfaI0j9kUPhGily5CAP1N2/view?usp=sharing

Universidad ICESI

Gracias. 😄