

Estructura de datos y algoritmos

Rodrigo Alvarez

rodrigo.alvarez2@mail_udp.cl

Clase pasada

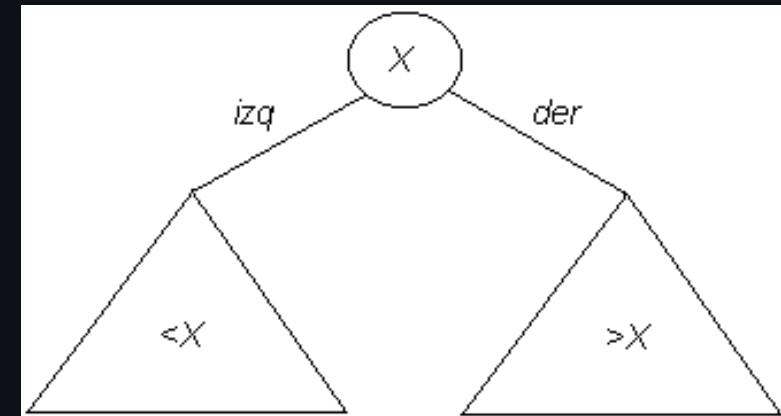
- Árboles
- Árboles binarios
 - Recorrido en árboles binarios
- Árboles binarios de búsqueda

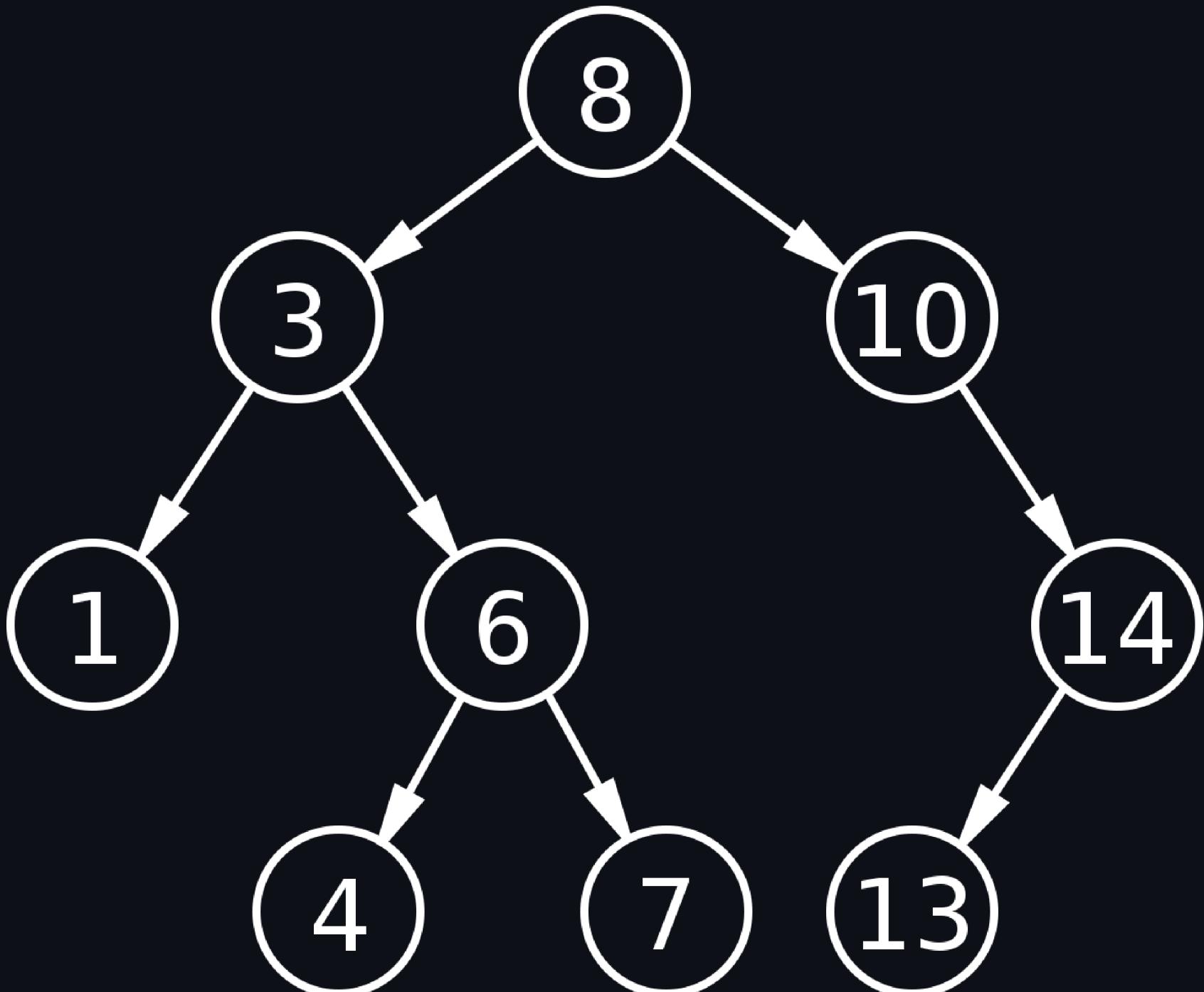
Árbol binario de búsqueda (BST)

Sea A un árbol binario de raíz R e hijos izquierdo y derecho H_I y H_D , respectivamente.

Decimos que A es un árbol binario de búsqueda (BST en inglés) si y solo si se satisfacen las dos condiciones al mismo tiempo:

- H_I es un BST con todos los elementos menores que $R \vee H_I$ es un árbol vacío
- H_D es un BST con todos los elementos mayores que $R \vee H_D$ es un árbol vacío



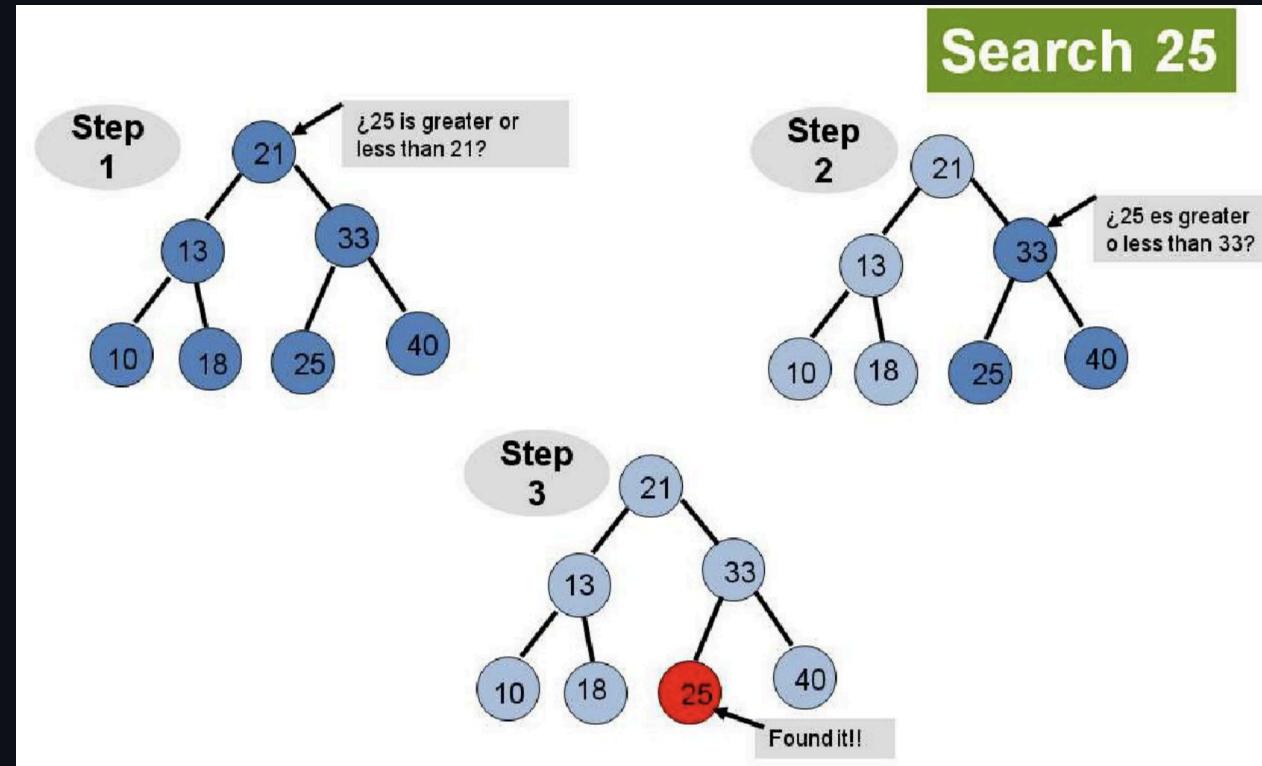


Árbol binario de búsqueda (BST): operaciones

- **Búsqueda:** Buscar un elemento en el árbol
- **Inserción:** Insertar un elemento en el árbol
- **Eliminación:** Eliminar un elemento del árbol

Árboles binarios de búsqueda: búsqueda

- Se recorre el árbol desde la raíz
- Si el nodo actual no es el nodo buscado, se decide si hay que buscar por la derecha o por la izquierda
- El algoritmo para, cuando se encuentra el nodo con el valor buscado o al llegar a un árbol vacío



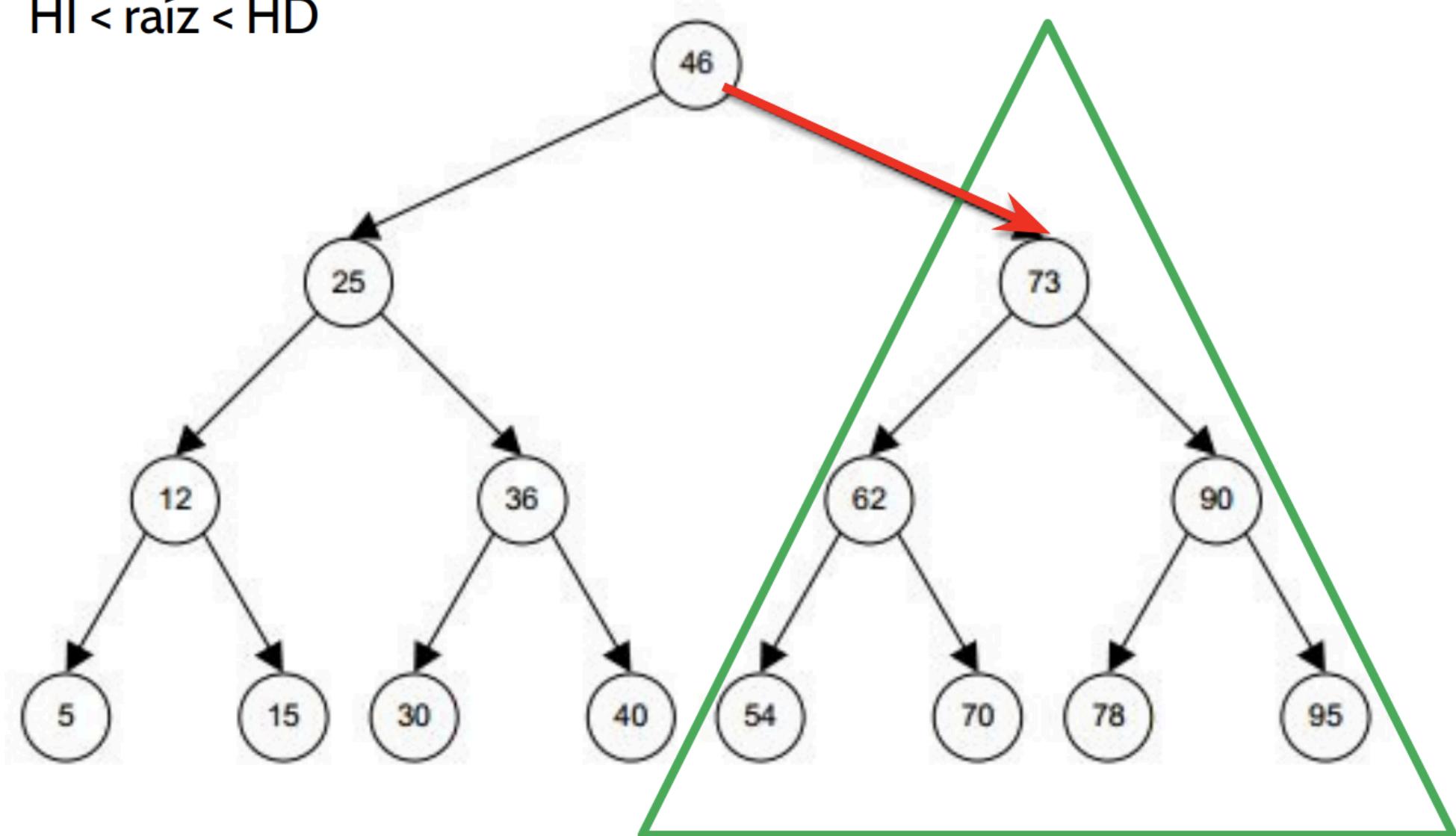
Árboles binarios de búsqueda: búsqueda

Recursivamente desde la raíz:

- Si el nodo que se busca es menor que el nodo actual, se busca en el subárbol izquierdo
- Si el nodo que se busca es mayor que el nodo actual, se busca en el subárbol derecho
- Si el nodo que se busca es igual al nodo actual, se ha encontrado el nodo
- Si el nodo que se busca no existe, se ha llegado a un nodo nulo

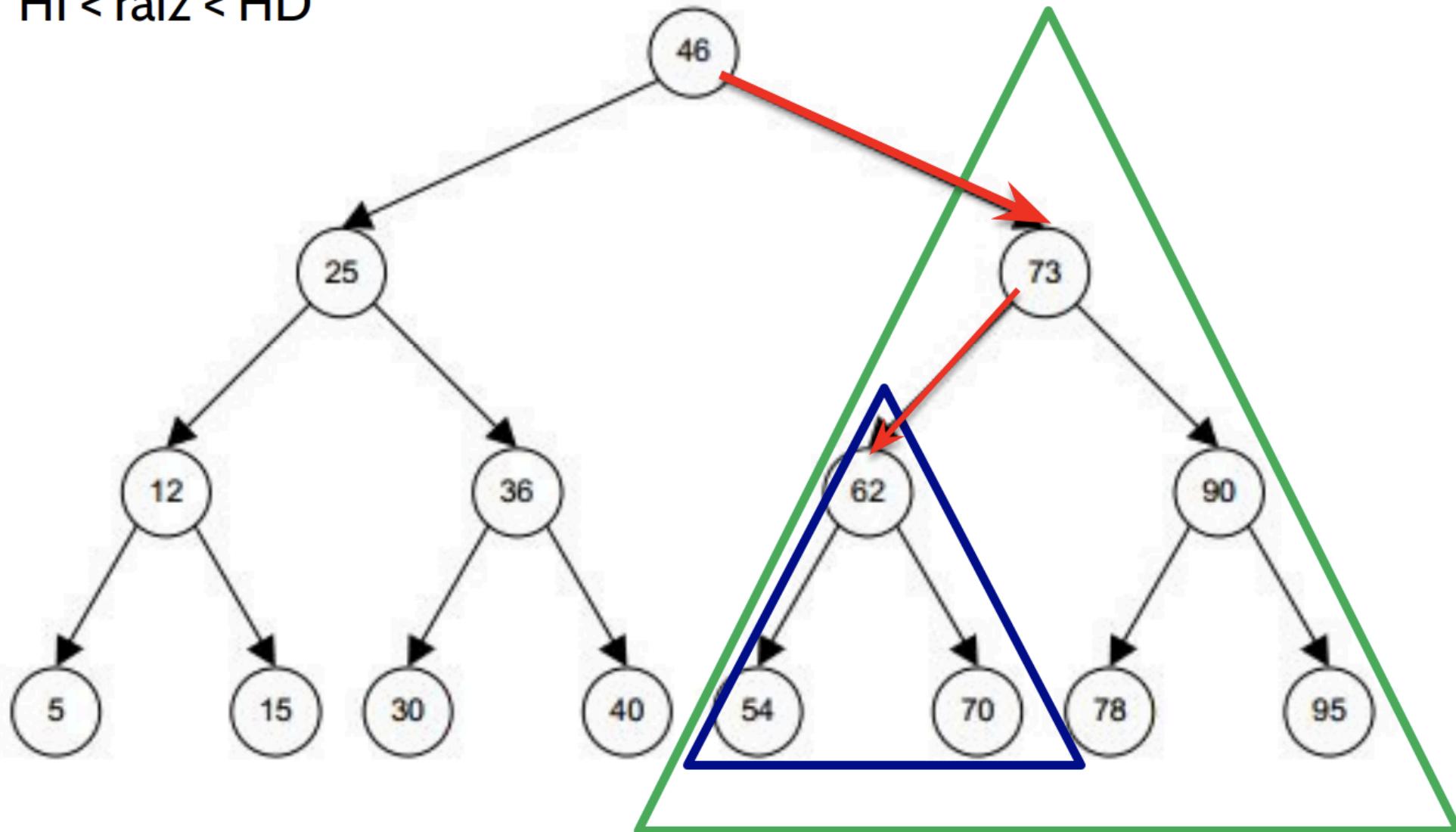
Solución: Buscar 70

HI < raíz < HD



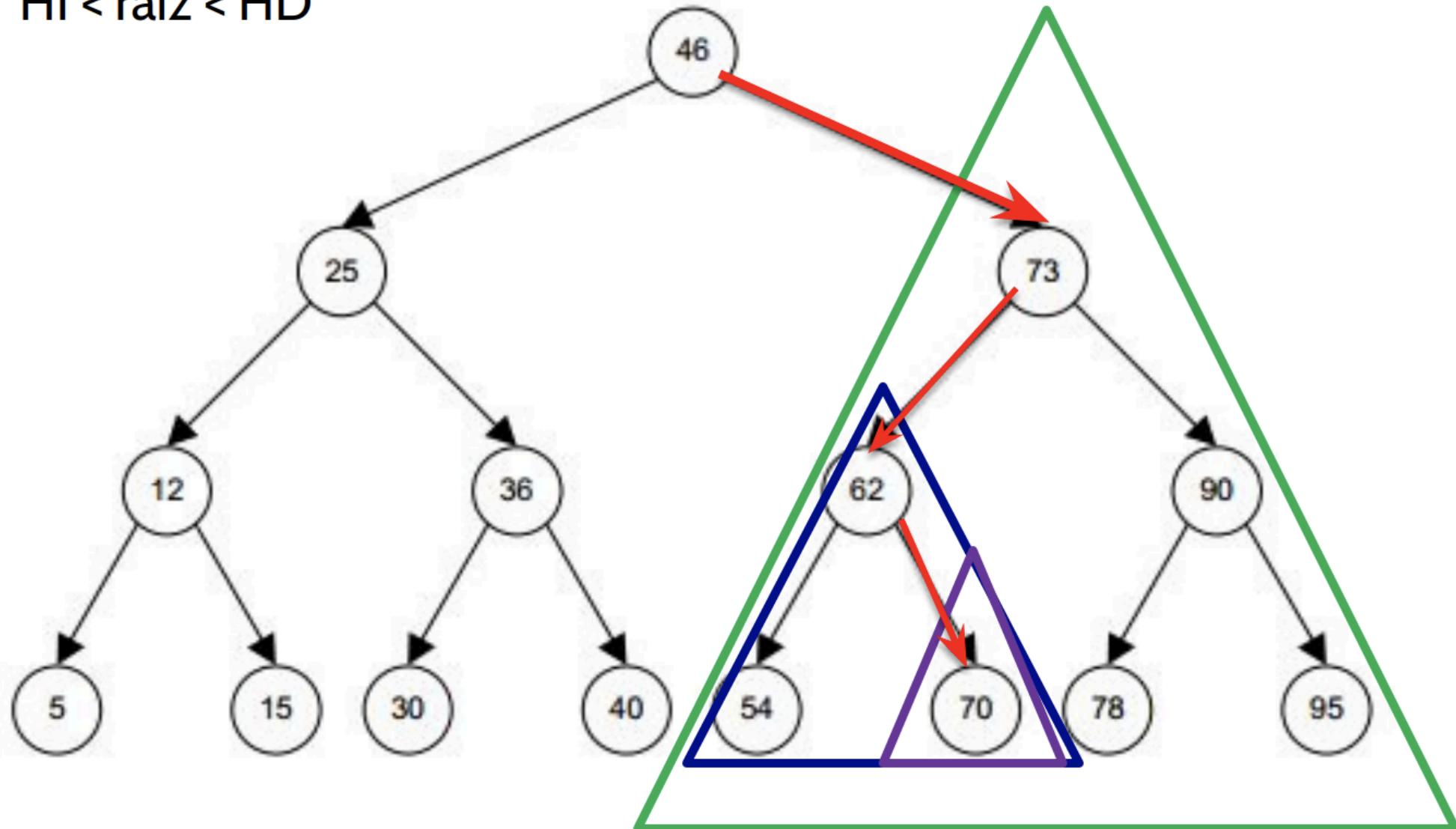
Solución: Buscar 70

HI < raíz < HD



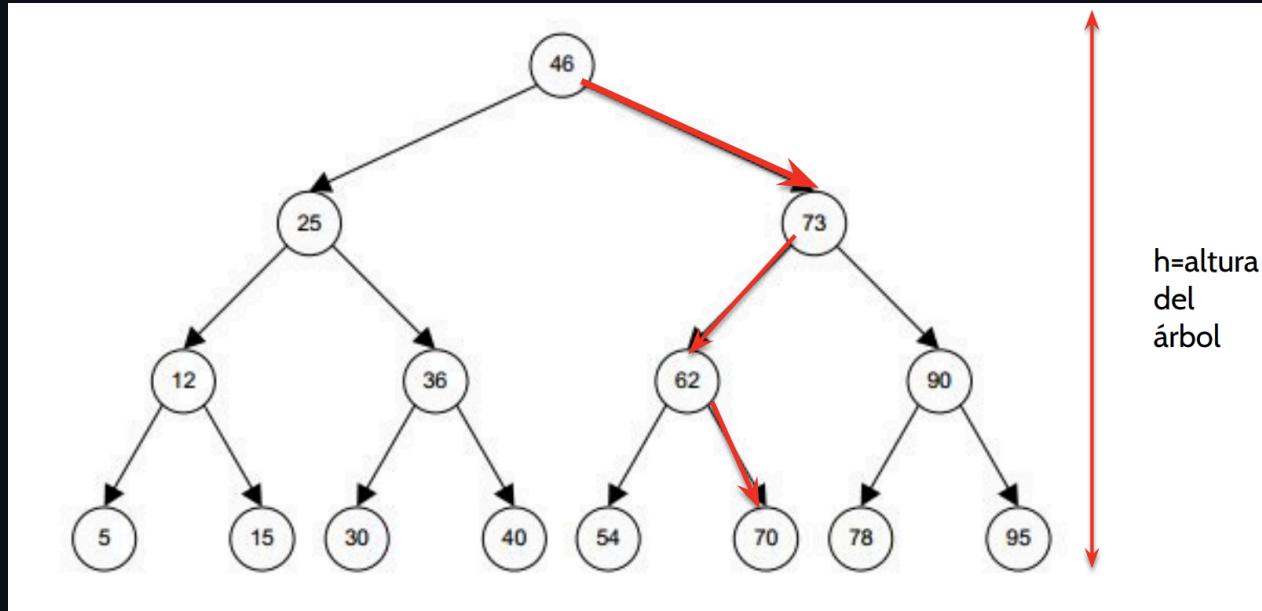
Solución: Buscar 70

HI < raíz < HD



Árboles binarios de búsqueda: búsqueda

- En el peor caso, en la búsqueda se realizan h comparaciones, donde h es la altura del árbol
- Por lo tanto, la complejidad será $O(h)$ donde h es la altura del árbol

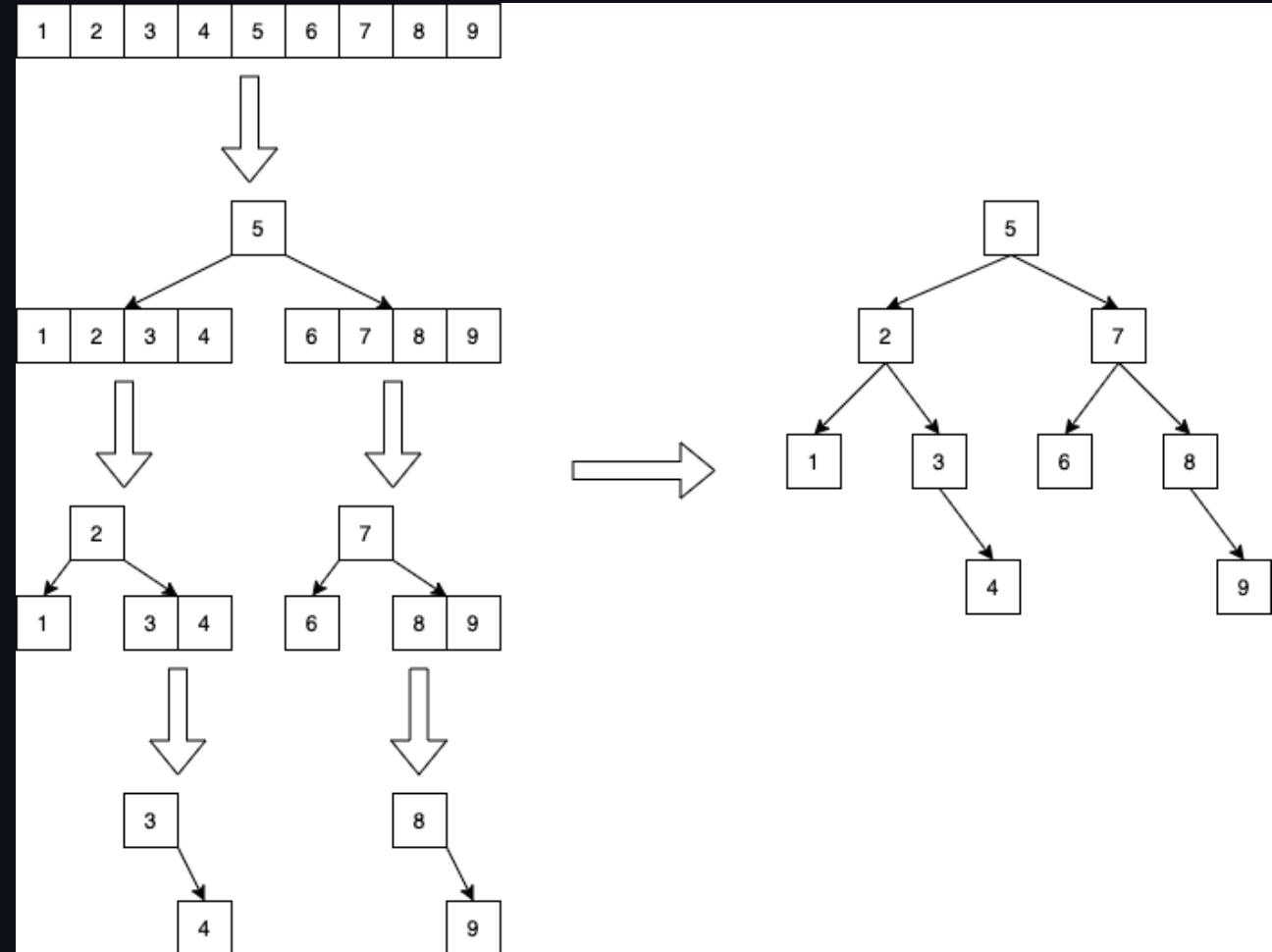


Árboles binarios de búsqueda: búsqueda

```
Node search(Node root, int value) {  
    if (root == null || root.data == value) return root;  
    if (root.data < value) return search(root.right, value);  
    else return search(root.left, value);  
}
```

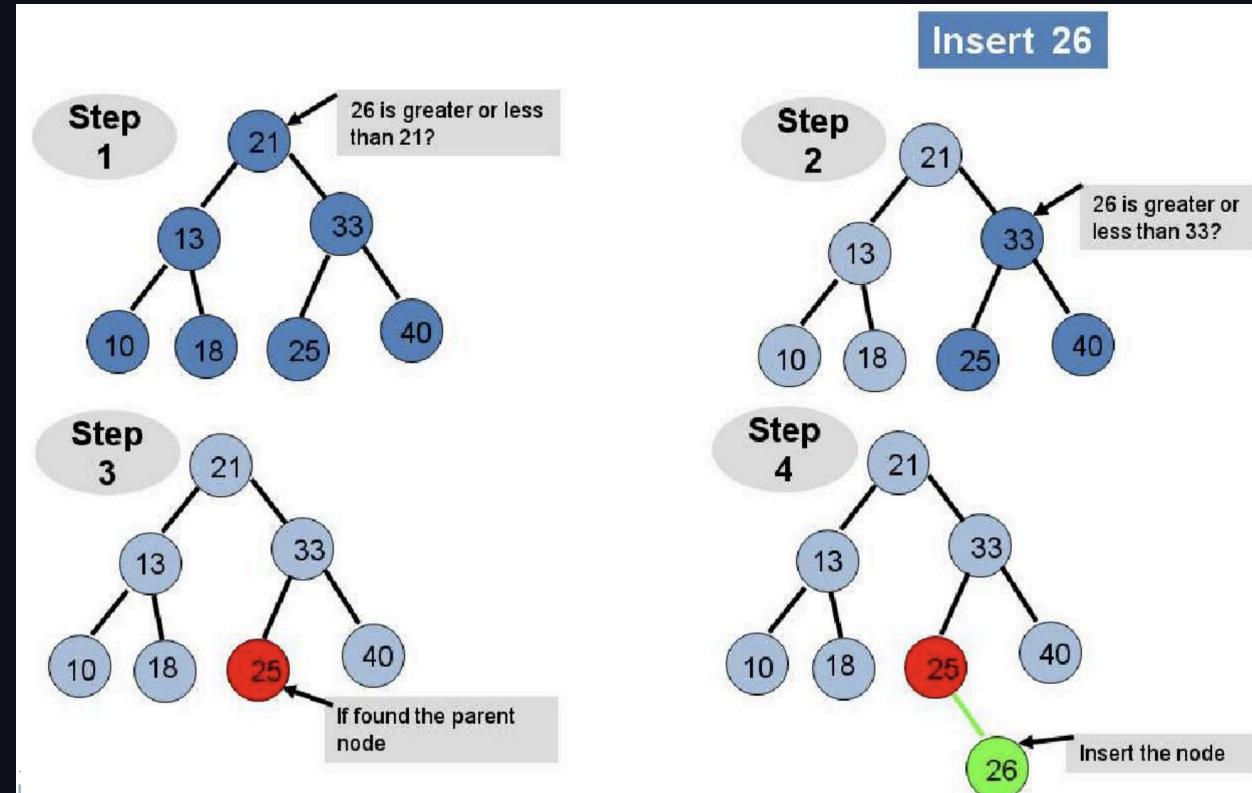
Búsqueda binaria a BST

- Existe una relación entre la búsqueda binaria y los árboles binarios de búsqueda
- La búsqueda en un BST sigue el mismo principio que la búsqueda binaria
- Un BST puede ser visto como una versión más flexible de la búsqueda binaria



Árboles binarios de búsqueda: inserción

- Los nodos se insertan siempre como nodos hoja
- La inserción en un árbol binario de búsqueda es similar a la búsqueda
- La diferencia radica en que si se llega a un nodo nulo, se inserta el nodo en ese lugar



Árboles binarios de búsqueda: inserción

Recursivamente desde la raíz:

- Si el nodo que se busca insertar es menor que el nodo actual, se busca en el subárbol izquierdo
- Si el nodo que se busca insertar es mayor que el nodo actual, se busca en el subárbol derecho
- Si el nodo que se busca insertar es igual al nodo actual, se ha encontrado el nodo y no se inserta (no se permiten duplicados)
- Si el nodo actual es nulo, se inserta el nodo en ese lugar

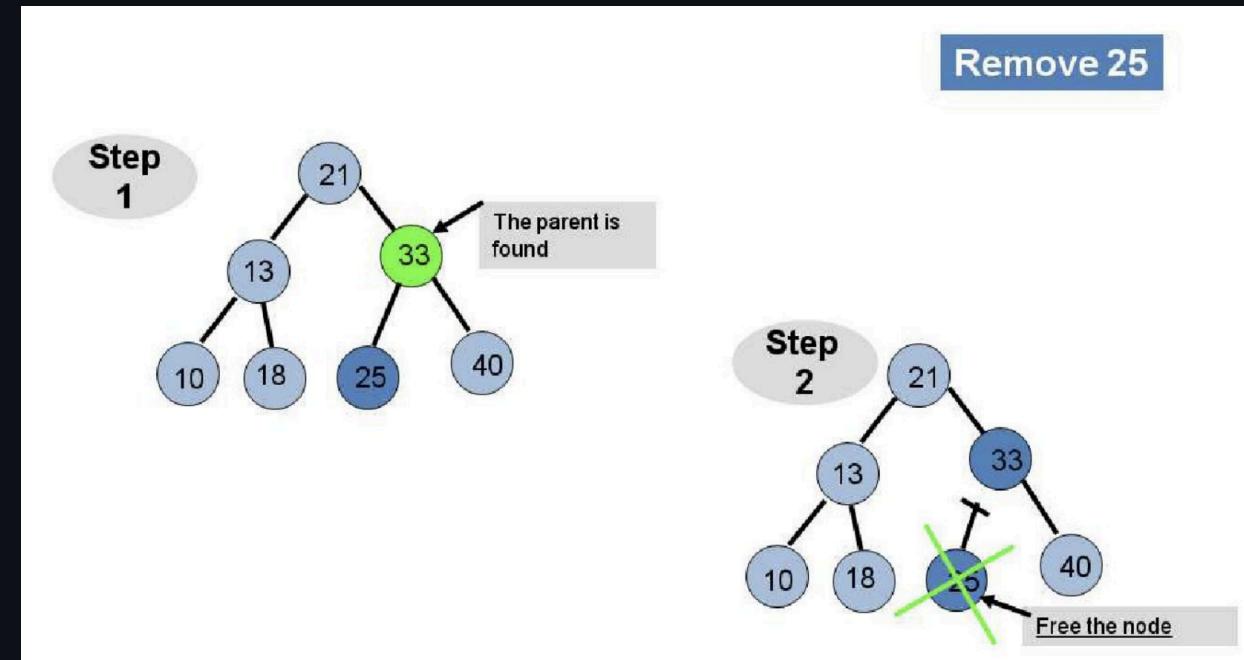
```
Node insert(Node root, int value) {  
    if (root == null) return new Node(value);  
    if (root.data < value) root.right = insert(root.right, value);  
    else if (root.data > value) root.left = insert(root.left, value);  
    return root;  
}
```

Árboles binarios de búsqueda: eliminación

- La eliminación en un árbol binario de búsqueda es más compleja que la búsqueda e inserción
- Se deben considerar tres casos:
 - Si el nodo a eliminar es una hoja:
 - Se elimina el nodo
 - Si el nodo a eliminar tiene un solo hijo:
 - Se elimina el nodo y se reemplaza por su hijo
 - Si el nodo a eliminar tiene dos hijos:
 - Se reemplaza el nodo por el nodo más pequeño del subárbol derecho o el nodo más grande del subárbol izquierdo (sucesor o predecesor)
 - Se elimina el nodo

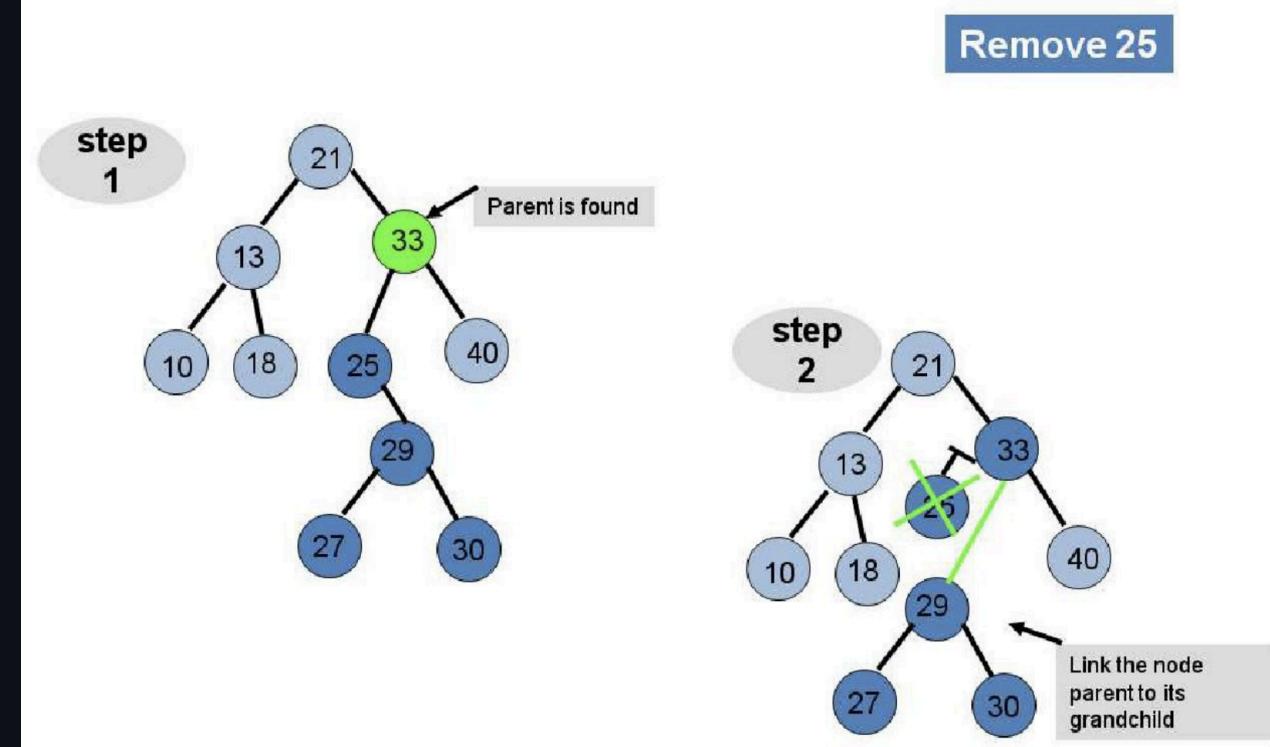
Árboles binarios de búsqueda: eliminación

Caso 1: Nodo hoja



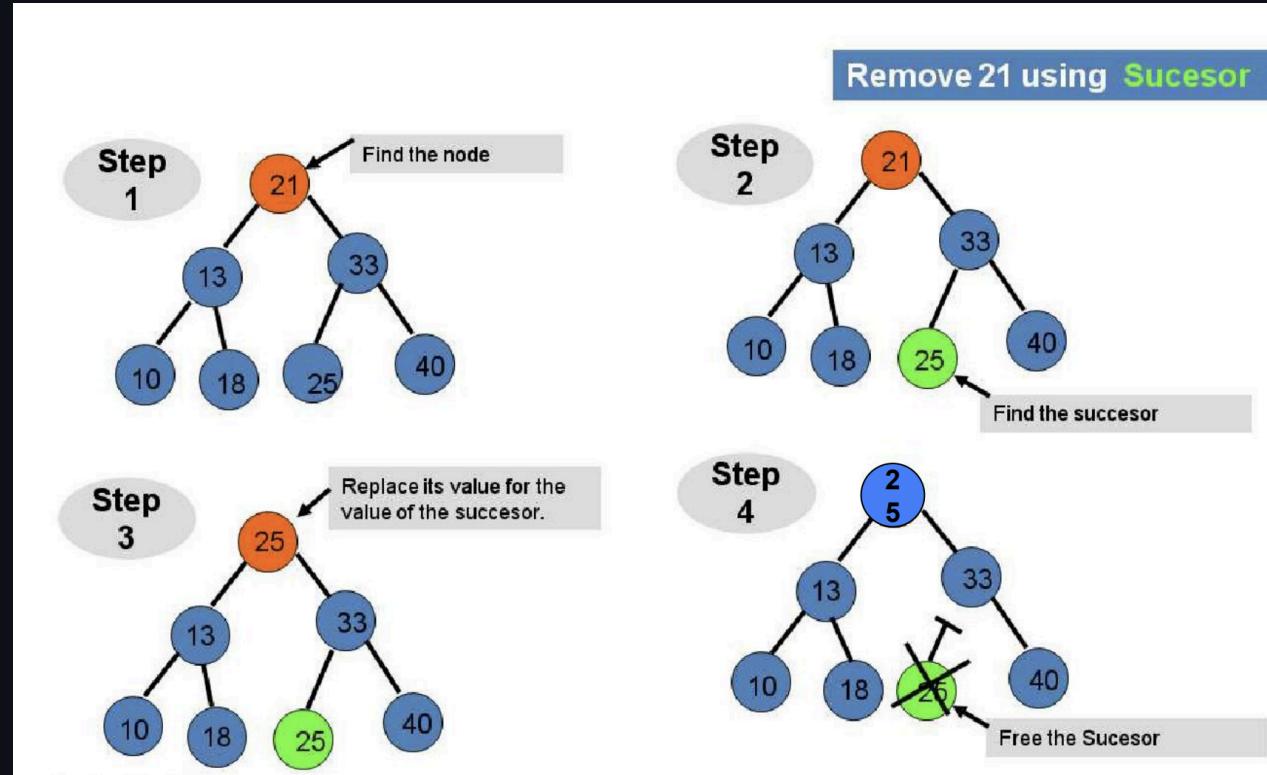
Árboles binarios de búsqueda: eliminación

Caso 2: Nodo con un hijo



Árboles binarios de búsqueda: eliminación

Caso 3: Nodo con dos hijos



Árboles binarios de búsqueda: eliminación

```
Node remove(Node root, int value) {
    if (root == null) return root;
    if (root.data < value) root.right = remove(root.right, value);
    else if (root.data > value) root.left = remove(root.left, value);
    else {
        if (root.left == null) return root.right;
        else if (root.right == null) return root.left;
        root.data = minValue(root.right);
        root.right = remove(root.right, root.data);
    }
    return root;
}
```

<https://visualgo.net/en/bst>

The screenshot shows the VisualGo.NET interface for Binary Search Trees (BST). The URL in the address bar is <https://visualgo.net/en/bst>. The page title is "BINARY SEARCH TREE AVL TREE". On the right, there are "e-Lecture Mode" and "LOGIN" buttons. The main area displays a tree with the statistics $N=18, h=4$. A context menu is open on the left, listing operations: Toggle BST Layout, Create, Search(v), Insert(v), Remove(v), Predec-/Succ-essor(v), Select(k), and Traverse(root). The bottom navigation bar includes a playback slider set to 1.0x, and links for About, Team, Terms of use, and Privacy Policy.

- Toggle BST Layout
- Create
- Search(v)
- Insert(v)
- Remove(v)
- Predec-/Succ-essor(v)
- Select(k)
- Traverse(root)

1.0x 1x | ▶◀◀ | ▶▶▶ | ■

About Team Terms of use Privacy Policy

Árboles binarios balanceados

- Un árbol binario de búsqueda puede degenerar en una lista enlazada si se insertan los elementos en orden
- Para evitar esto, se pueden utilizar árboles binarios balanceados
- Un árbol binario balanceado es un árbol binario de búsqueda en el que la altura de los subárboles izquierdo y derecho de cada nodo difiere en no más de 1
- Los árboles binarios balanceados permiten realizar operaciones en tiempo logarítmico
- Existen estructuras de datos que implementan árboles binarios autobalanceados, como los árboles AVL y los árboles red-black

AVL

- Un árbol AVL es un tipo especial de árbol binario ideado por los matemáticos soviéticos Adelson-Velskii y Landis.
- Fue el primer árbol de búsqueda binario auto-balanceable que se ideó.
- En un árbol AVL, la diferencia de alturas entre los subárboles izquierdo y derecho de cualquier nodo es de a lo más 1.