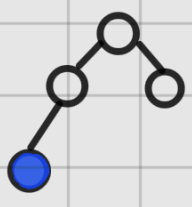
Eliminar un elemento de un Árbol Binario Ordenado

El método elimina de ABO tiene 3 casos importantes

1: El vértice a eliminar no tiene hijos (es hoja)

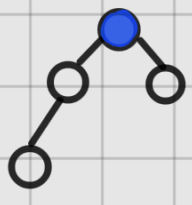
2: El vértice a eliminar tiene a lo más un hijo

3: El vértice a eliminar tiene dos hijos distintos de null



Caso 1

Caso 2

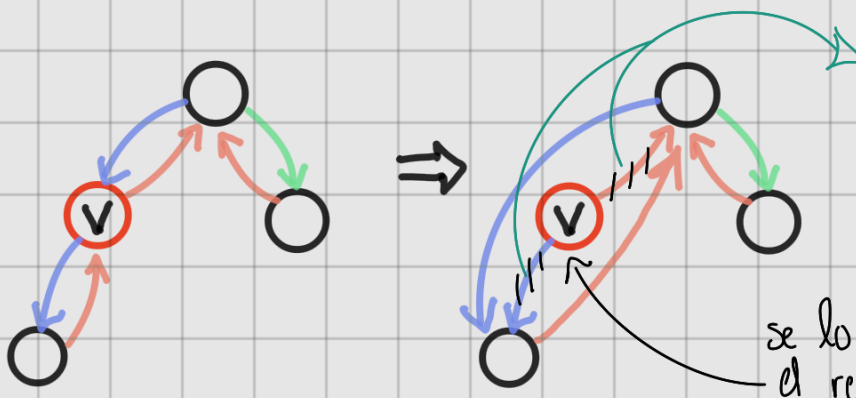


Caso 3

Siendo v el vértice a eliminar:

Caso 1: Simplemente vemos que hijo es v, es decir vemos si es hijo izq o der de su padre y lo volvemos null

Caso 2: Revisamos que hijo es el que tiene v (si el distinto de null es izq o derecho) y revisamos que hijo es v si es hijo izq o der. y con esto reasignamos para que el hijo de v ahora sea hijo del padre de v en la misma posición que v y hacemos que el padre del hijo de v ahora sea el padre de v

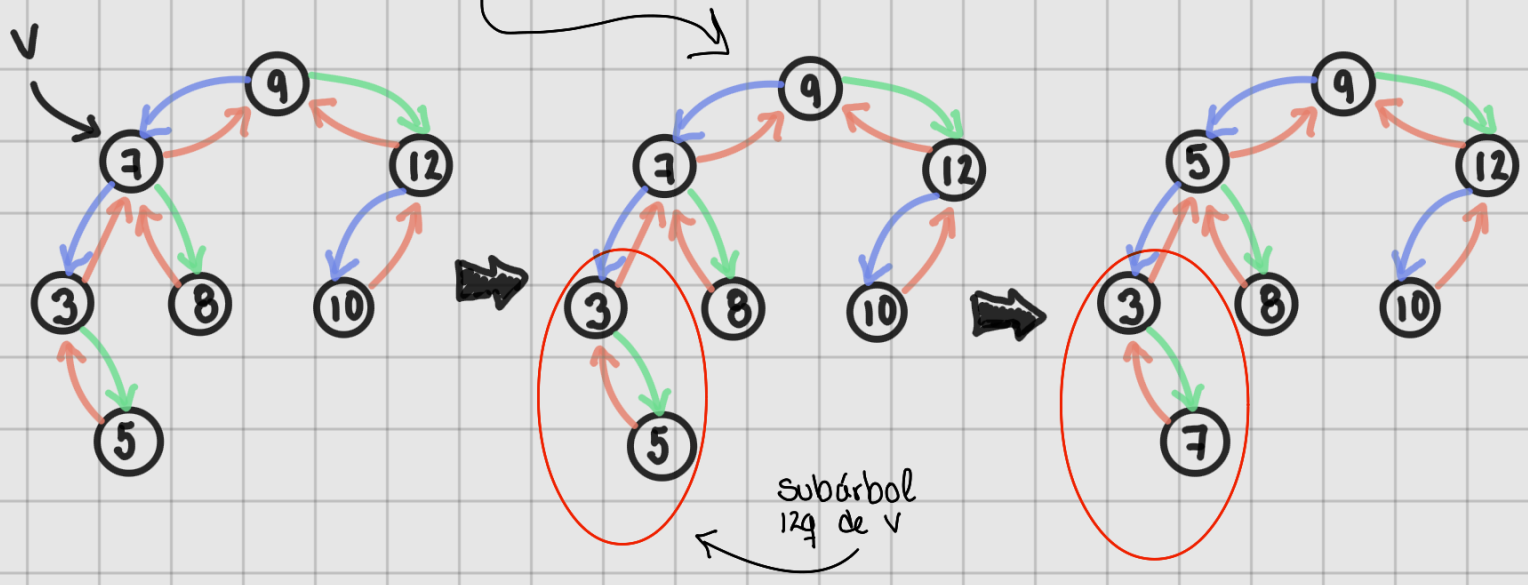


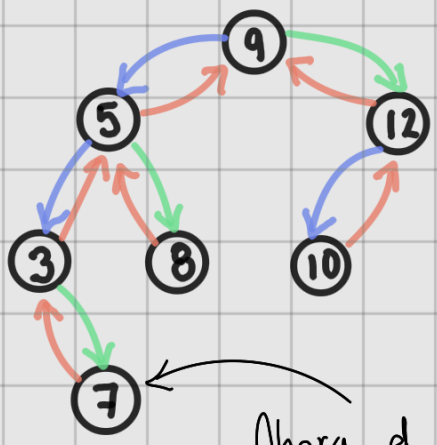
Aunque aún hay conexiones de v a otros vértices ya no hay conexiones del árbol a v

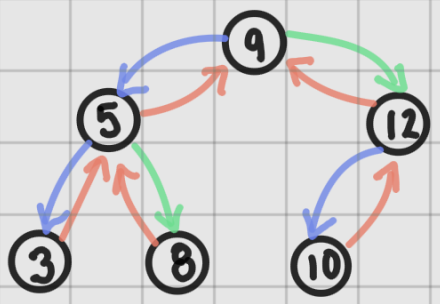
se lo lleva el recolector de basura en algún momento

Caso 3: v tiene dos hijos distintos de null ent. vamos a buscar un vértice que tenga cero o un hijo, este vértice será el maximo del subarbol izq de v, e intercambiaremos los elementos de ambos vértices, con ello podremos eliminar el elemento que ahora estará en la posición donde estuba el máximo, es decir, ya solo tendra cero o un hijo por lo que podemos aplicar lo del caso 1 o lo del caso 2 de estu forma garantizaremos que todos los vértices seguiran ordenados.

Buscamos en el subárbol izq. el primer vértice cuyo hijo derecho sea null.







Ahora el 7 ya está en una posición donde lo podemos eliminar ya que cumple con tener cero o un sólo hijo.

```java
@Override public void elimina(T elemento) {
    // Buscamos el elemento a eliminar
    Vertice eliminar = busca(raiz, elemento);
    // Revisamos que el vértice exista, si no es el caso terminamos
    if (eliminar == null) {
        return;
    }
    // Revisamos si el vertice tiene dos hijos != null
    if (eliminar.hayIzquierdo() && eliminar.hayDerecho()) {
        // Si es el caso intercambiamos el elemento
        // por el máximo de su subarbol izquierdo.
        // El vértice regresado tiene a lo más un hijo
        eliminar = intercambiaEliminable(eliminar);
    }
    // Revisamos si el vertice a eliminar es hoja
    if (!eliminar.hayIzquierdo() && !eliminar.hayDerecho()) {
        eliminaHoja(eliminar);
    // Si no es el caso entonces tiene un hijo != null
    } else {
        eliminaVertice(eliminar);
    }
}
```