Árboles de búsqueda avanzados

Alberto Verdejo

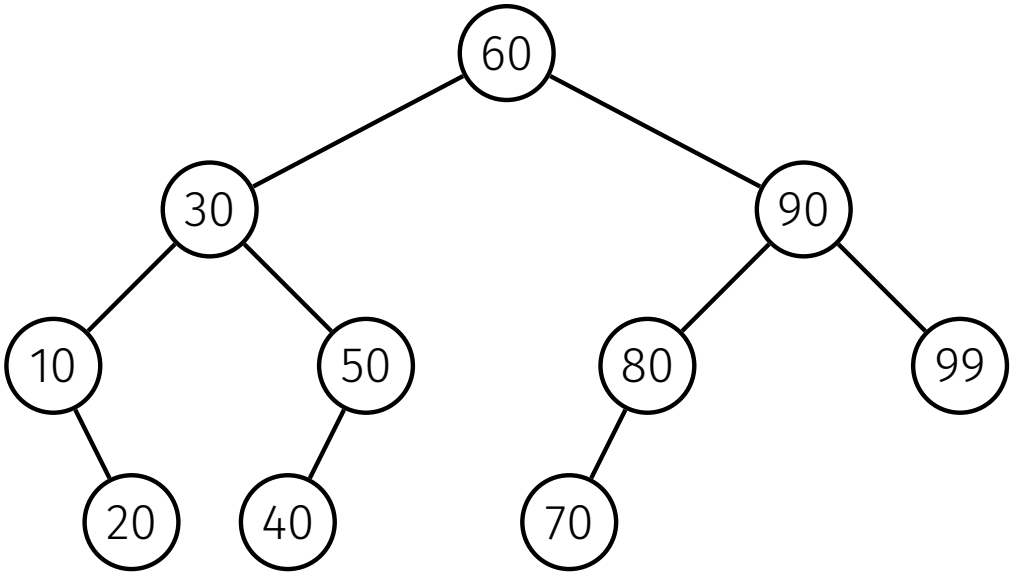
Dpto. de Sistemas Informáticos y Computación
Universidad Complutense de Madrid

Bibliografía

- M. A. Weiss. *Data Structures and Algorithm Analysis in C++*. Fourth edition. Pearson, 2014.
  Capítulo 4
- F. M. Carrano y T. Henry. *Data Abstraction & Problem Solving with C++: Walls and Mirrors*. Sixth edition. Pearson, 2013.
  Capítulo 19
- R. Sedgewick y K. Wayne. *Algorithms*. Fourth Edition. Addison-Wesley, 2014.
  Sección 3.3

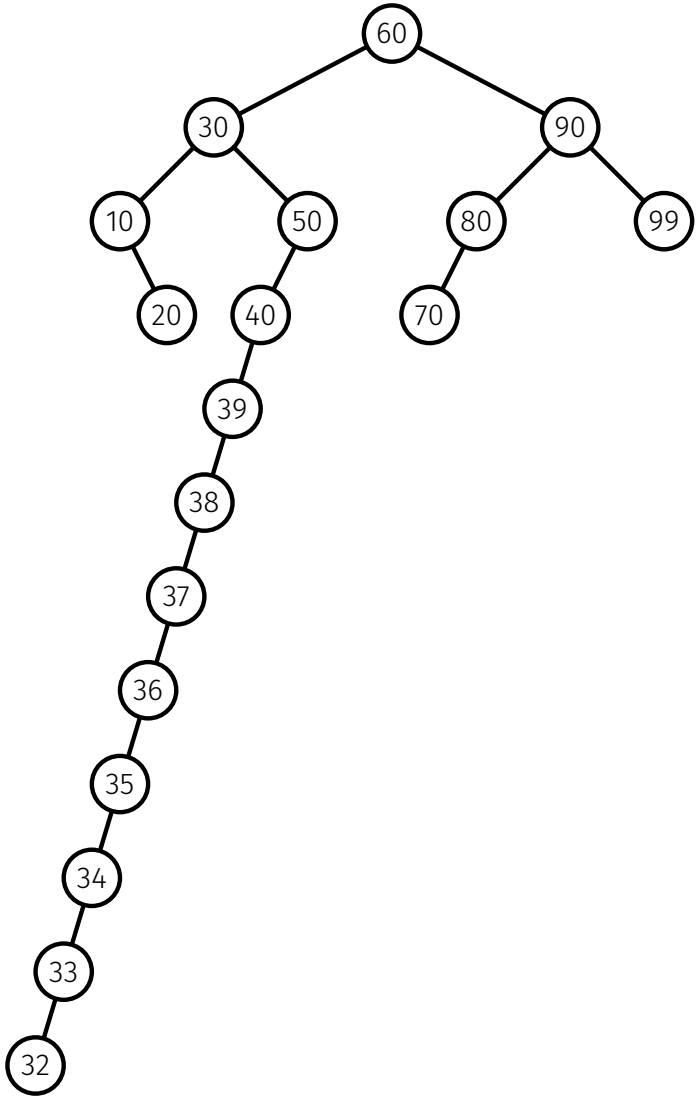
Árboles binarios de búsqueda

Ejemplo:



Árboles binarios de búsqueda

La profundidad del árbol puede degenerar.



Análisis del caso medio

- El coste de las operaciones de búsqueda, inserción y borrado está en $O(h)$, siendo h la altura del árbol. En el caso peor, $O(N)$.
- Si todas las posibles ordenaciones de la entrada son posibles, la profundidad media sobre todos los nodos está en $O(\log N)$.

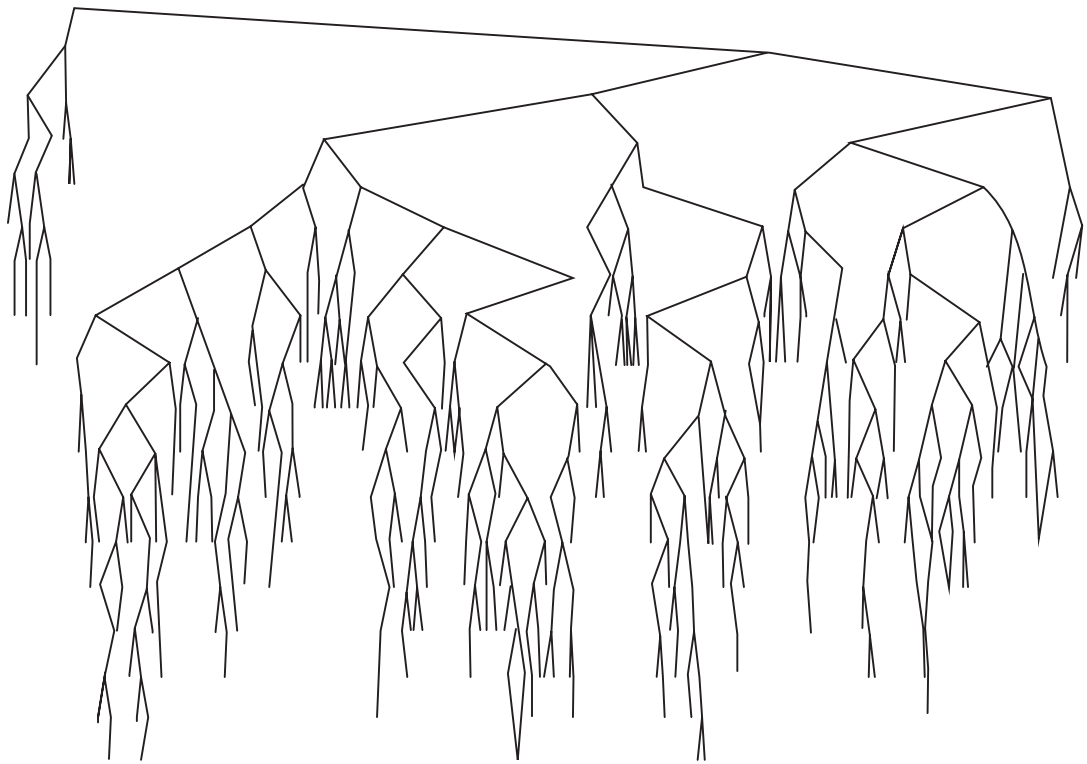


ABB generado aleatoriamente (500 hojas, profundidad media 9,98)

Análisis del caso medio

- Si también hay borrados, no está tan claro que todos los ABBs sean igual de probables.
- De hecho, la estrategia típica de borrado sustituye el nodo borrado por el menor elemento en su hijo derecho.
- El efecto exacto de esta estrategia aún se desconoce.

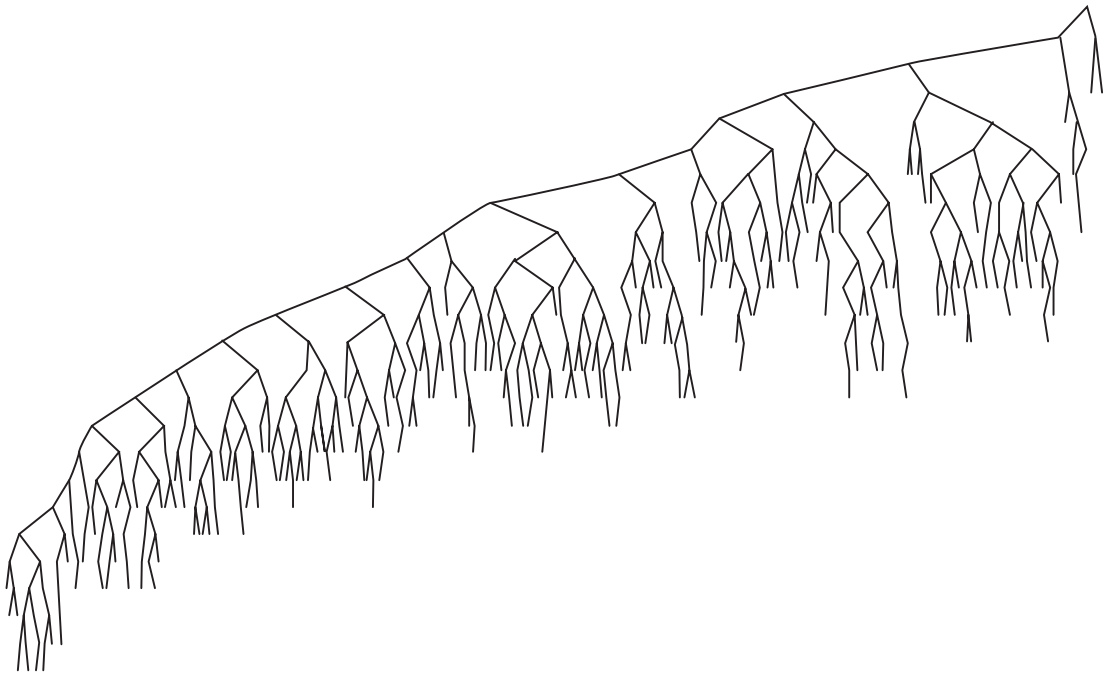
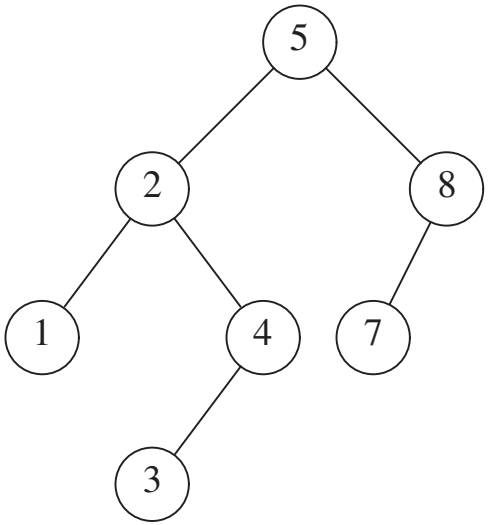


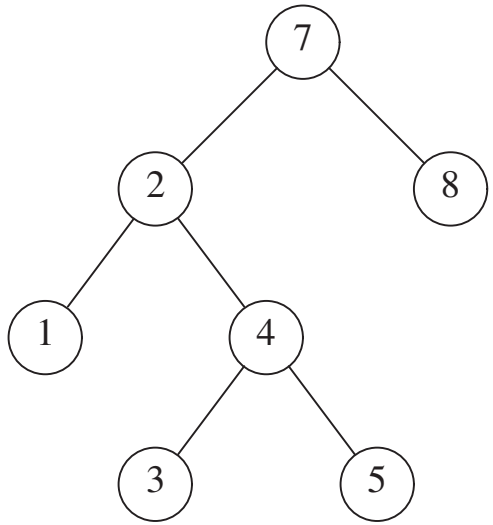
ABB después de $\Theta(N^2)$ pares inserción/borrado

Árboles AVL

- Los árboles AVL (Adelson-Velskii y Landis, 1962) son ABBs con una condición de equilibrio: todos los nodos del árbol cumplen que la diferencia de alturas de sus dos hijos es como mucho 1.

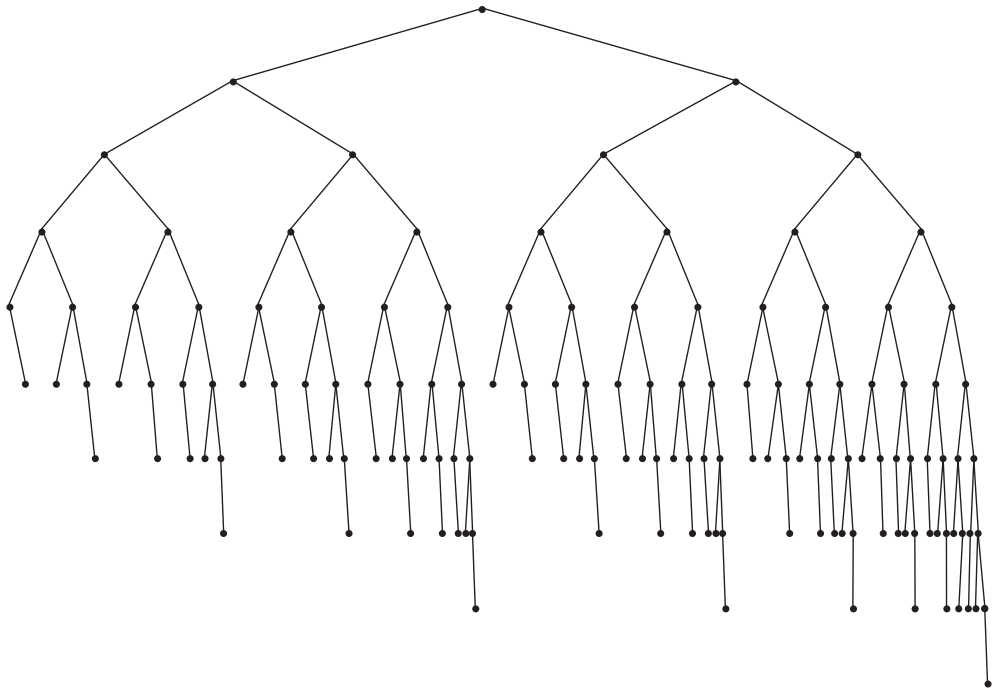
  ¿Son estos árboles AVL?





Árboles AVL

- La altura de un árbol AVL con N nodos es como mucho $1{,}44 \log(N+2) - 1{,}328$.
- Árbol AVL más pequeño de altura 10



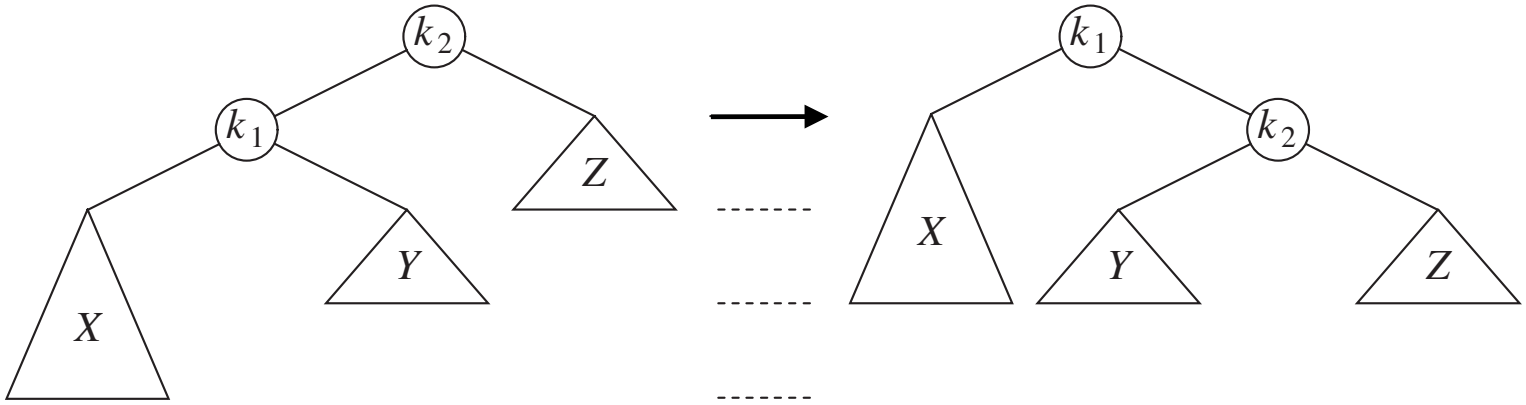
- El mínimo número de nodos, $S(h)$, en un árbol AVL de altura h viene dado por $S(h) = S(h-1) + S(h-2) + 1$, con $S(0) = 0$ y $S(1) = 1$. La función $S(h)$ está relacionada con los números de Fibonacci, de donde se saca la cota anterior.

Árboles AVL

- En un árbol AVL con N nodos todas las operaciones de los ABB pueden hacerse con coste en $O(\log N)$.
- La inserción de un nodo puede hacer que deje de cumplirse la condición de equilibrio.
- Si ese es el caso, el árbol debe reestructurarse mediante rotaciones.
- Después de una inserción, solo los nodos en el camino desde el nodo insertado a la raíz pueden haberse desequilibrado. Sea α el primer nodo desequilibrado en ese camino. La diferencia de alturas entre sus hijos es 2, con cuatro casos posibles:
  1. Inserción en el subárbol izquierdo del hijo izquierdo de α.
  2. Inserción en el subárbol derecho del hijo izquierdo de α.
  3. Inserción en el subárbol izquierdo del hijo derecho de α.
  4. Inserción en el subárbol derecho del hijo derecho de α.

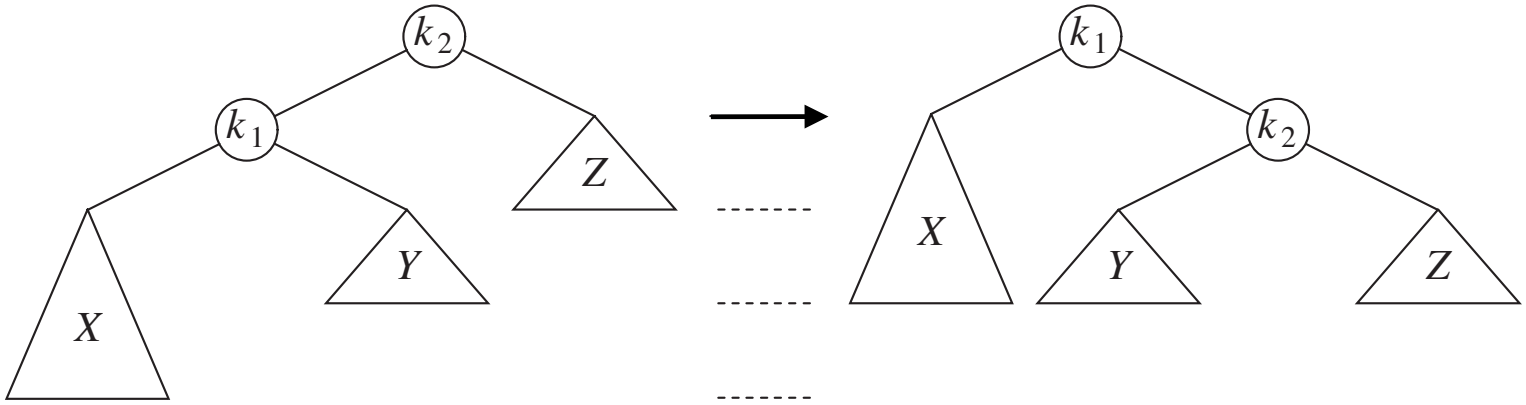
AVL: Rotación simple a la derecha

- El caso 1 puede equilibrarse con la siguiente rotación:

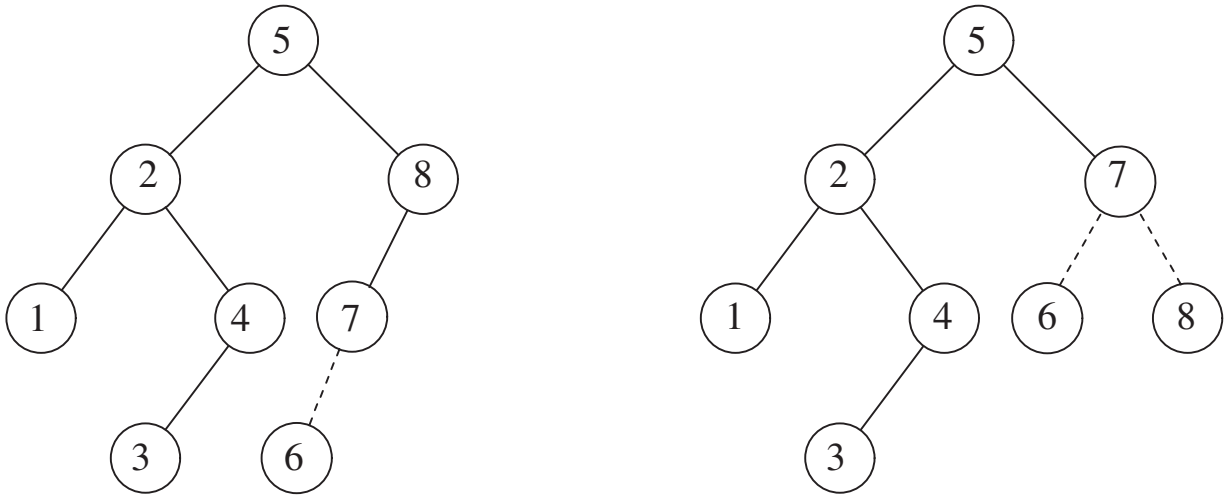


AVL: Rotación simple a la derecha

- El caso 1 puede equilibrarse con la siguiente rotación:

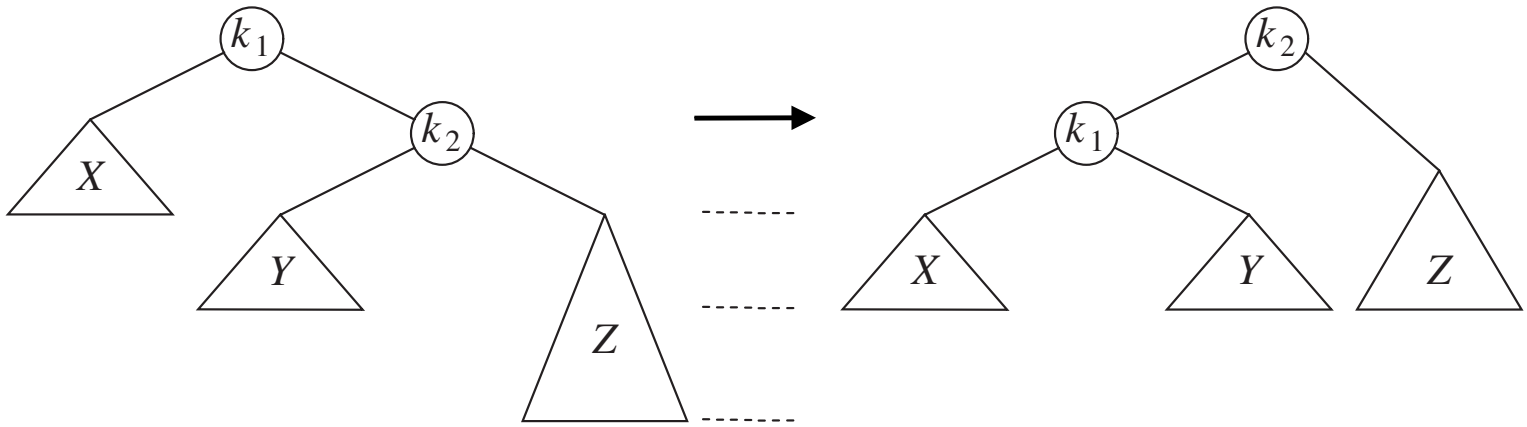


- Por ejemplo,



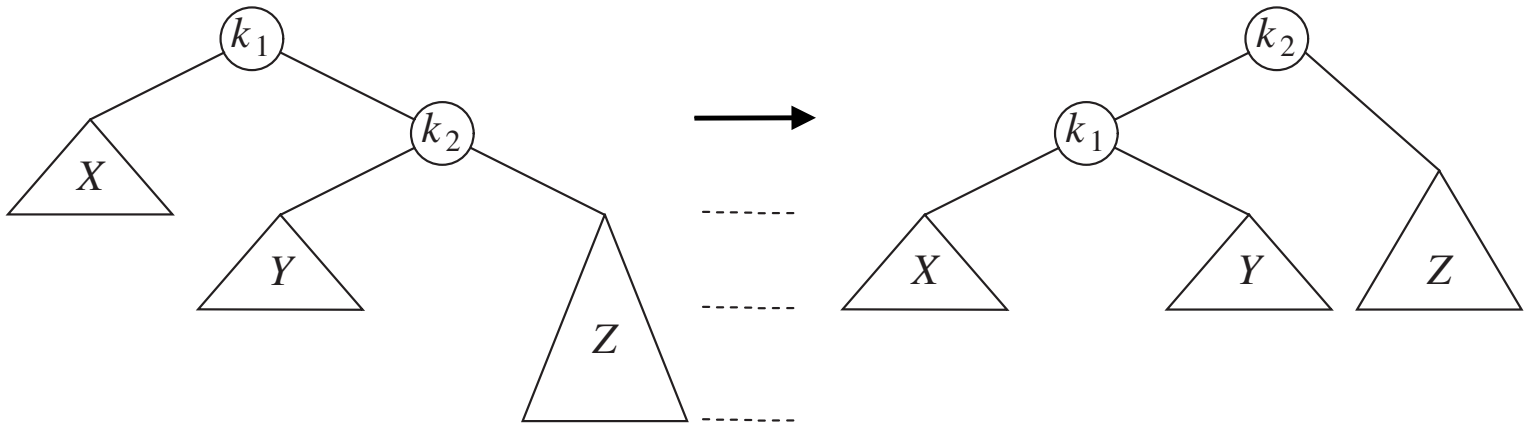
AVL: Rotación simple a la izquierda

- Necesaria en el caso 4:



AVL: Rotación simple a la izquierda

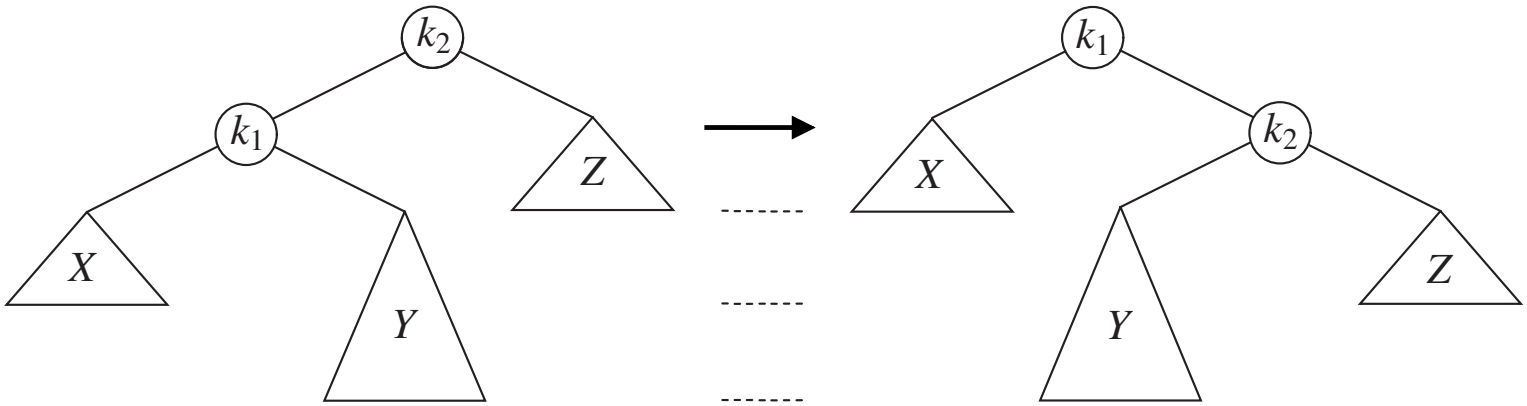
- Necesaria en el caso 4:



- **Ejercicio**: Insertar en un árbol AVL vacío los elementos 3, 2, 1 y del 4 al 7.

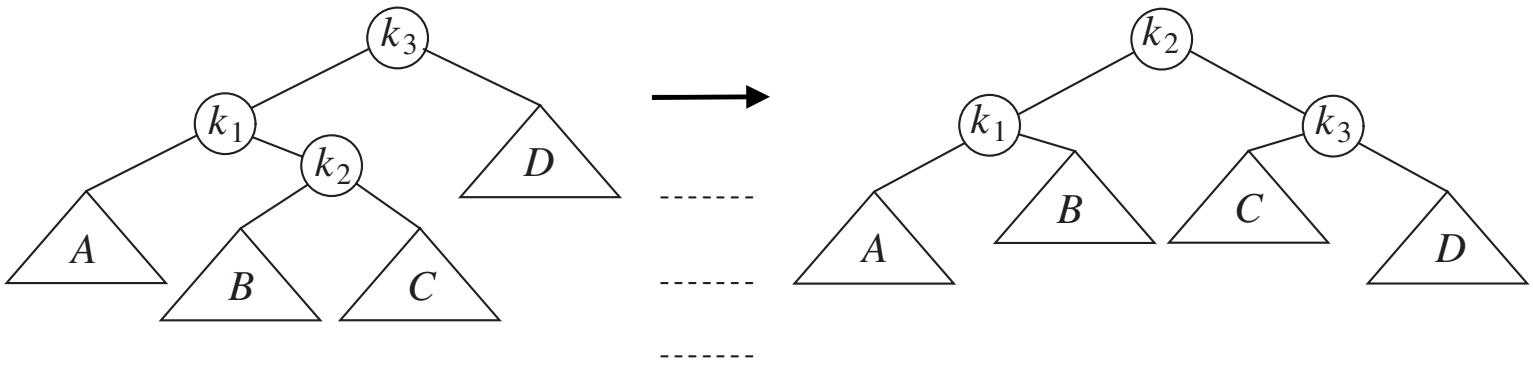
AVL: Rotación doble

- La rotación simple no funciona en los casos 2 y 3:



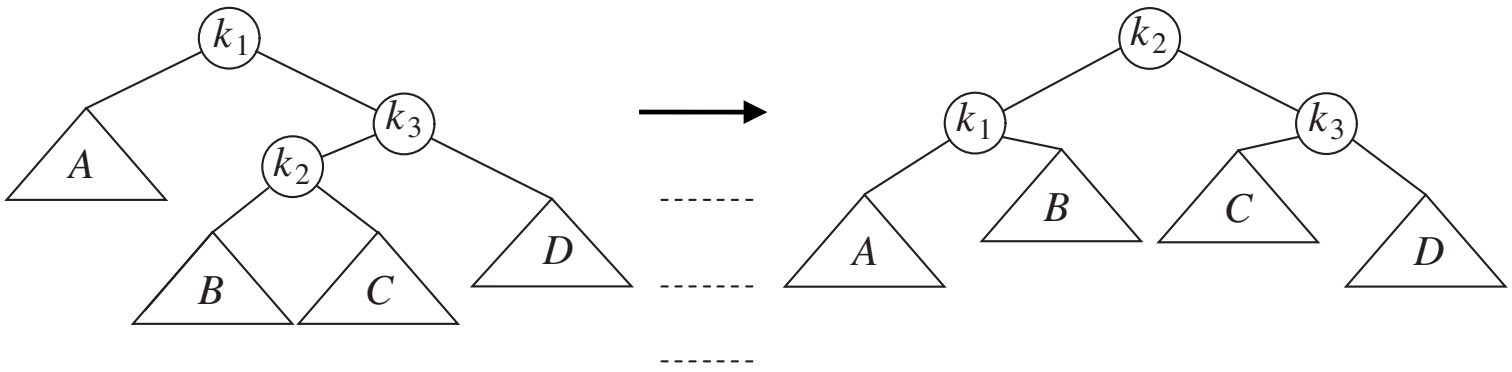
AVL: Rotación doble izquierda-derecha

- Hace falta una rotación doble. El hecho de que se haya insertado un nodo en el subárbol *Y* garantiza que no es vacío.
- Para resolver el caso 2:



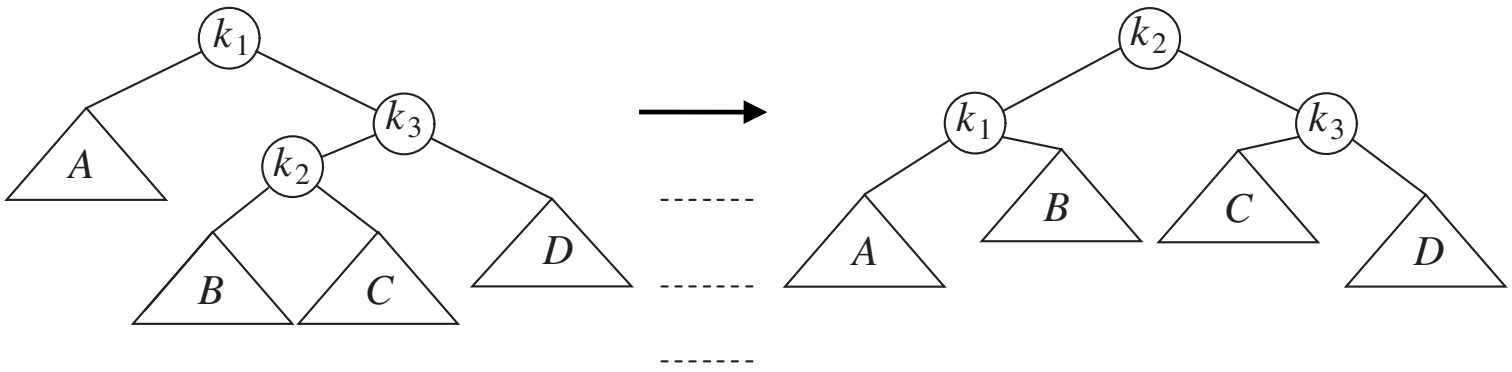
AVL: Rotación doble derecha-izquierda

- Para resolver el caso 3:



AVL: Rotación doble derecha-izquierda

- Para resolver el caso 3:



- Ejercicio: Insertar en el árbol AVL del ejercicio anterior del 10 al 16 en orden inverso y luego el 8 y el 9.

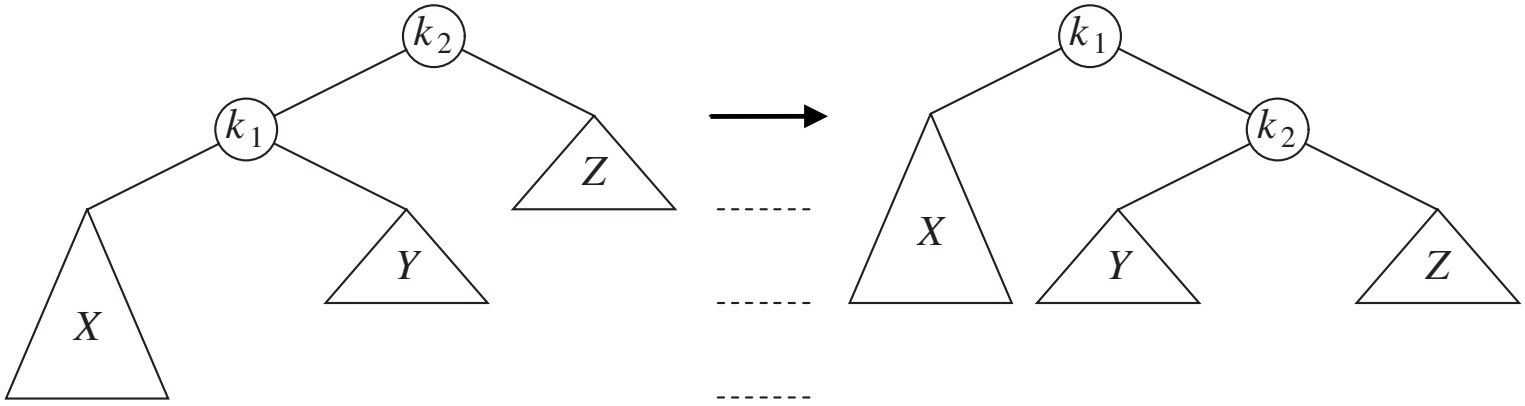
Implementación de los árboles AVL

TreeMap_AVL.h

Se añade a cada nodo un campo más con la altura de ese subárbol.

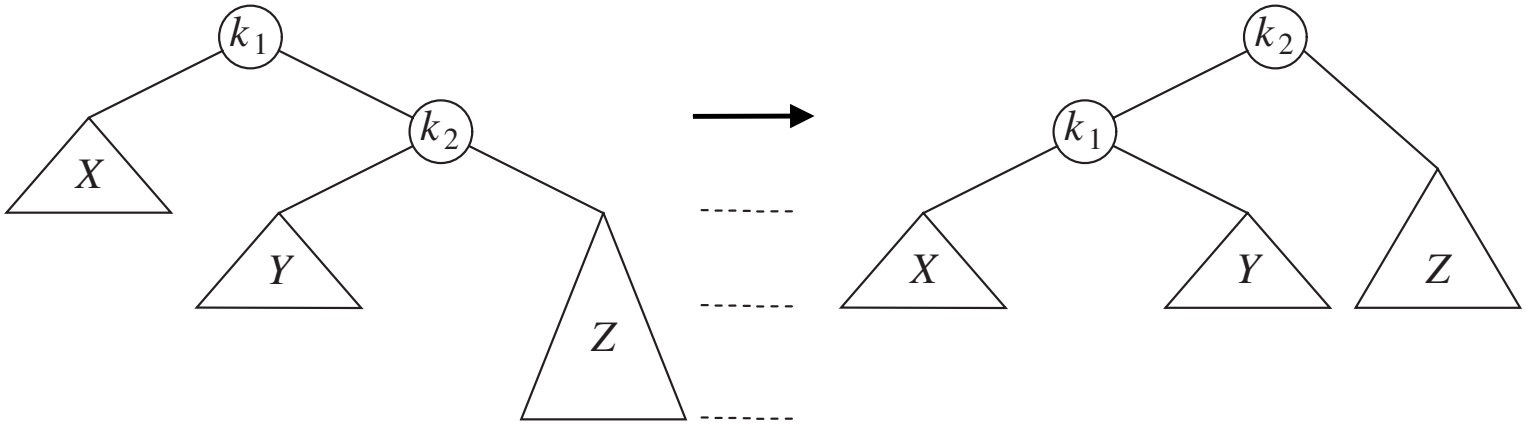
```
protected:
   bool inserta(clave_valor const& cv, Link & a) {
      bool crece;
      if (a == nullptr) { // se inserta el nuevo par <clave, valor>
         a = new TreeNode(cv);
         ++nelems;
         crece = true;
      } else if (menor(cv.clave, a->cv.clave)) {
         crece = inserta(cv, a->iz);
         if (crece) reequilibraDer(a);
      } else if (menor(a->cv.clave, cv.clave)) {
         crece = inserta(cv, a->dr);
         if (crece) reequilibraIzq(a);
      } else // la clave ya estaba
         crece = false;
      return crece;
   }
```

Implementación de los árboles AVL



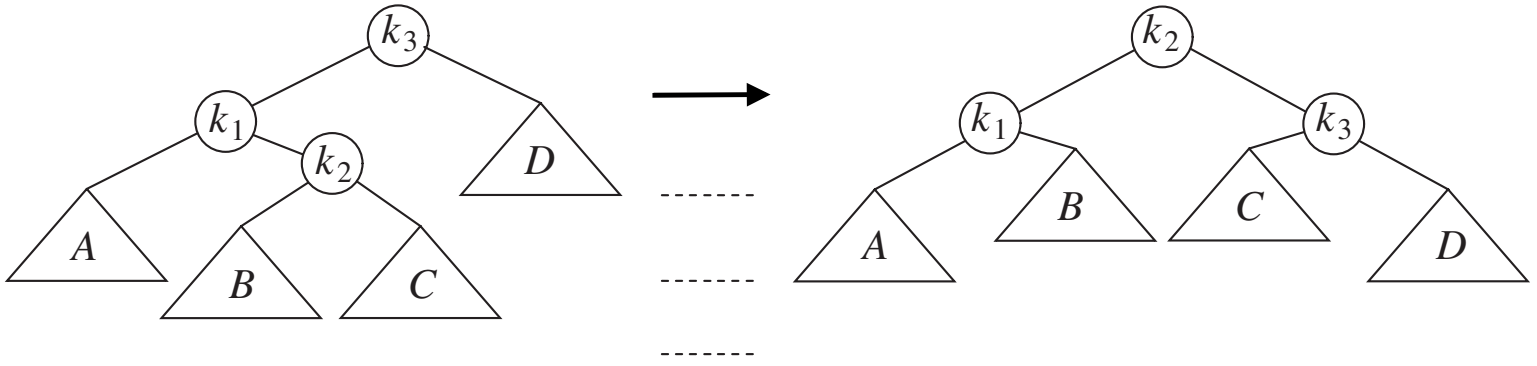

```
static void rotaDer(Link & k2) {
   Link k1 = k2->iz;
   k2->iz = k1->dr;
   k1->dr = k2;
   k2->altura = std::max(altura(k2->iz), altura(k2->dr)) + 1;
   k1->altura = std::max(altura(k1->iz), altura(k1->dr)) + 1;
   k2 = k1;
}

static int altura(Link a) {
   if (a == nullptr) return 0;
   else return a->altura;
}
```

Implementación de los árboles AVL



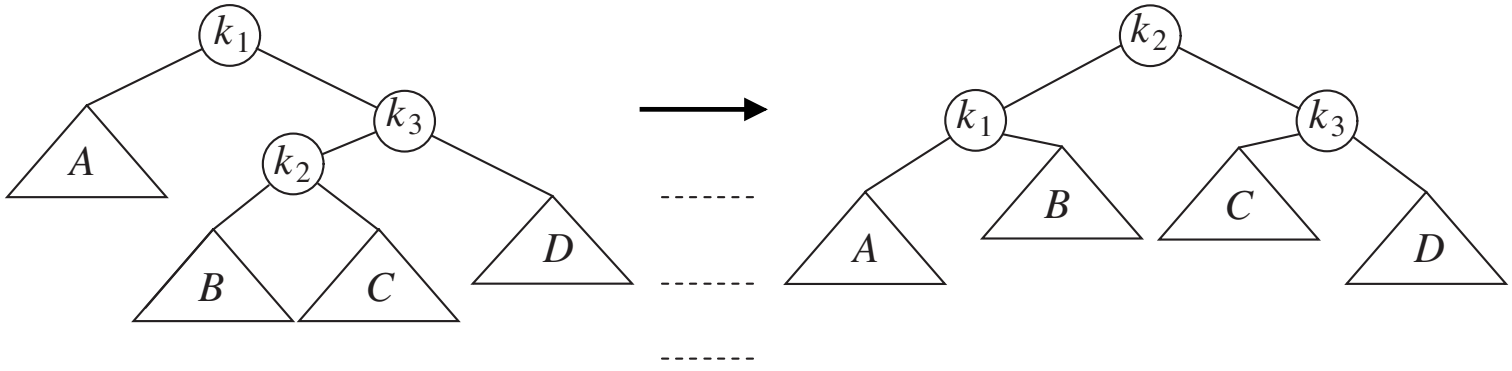

```
static void rotaIzq(Link & k1) {
   Link k2 = k1->dr;
   k1->dr = k2->iz;
   k2->iz = k1;
   k1->altura = std::max(altura(k1->iz), altura(k1->dr)) + 1;
   k2->altura = std::max(altura(k2->iz), altura(k2->dr)) + 1;
   k1 = k2;
}
```

Implementación de los árboles AVL




```
static void rotaIzqDer(Link & k3) {
   rotaIzq(k3->iz);
   rotaDer(k3);
}
```

Implementación de los árboles AVL



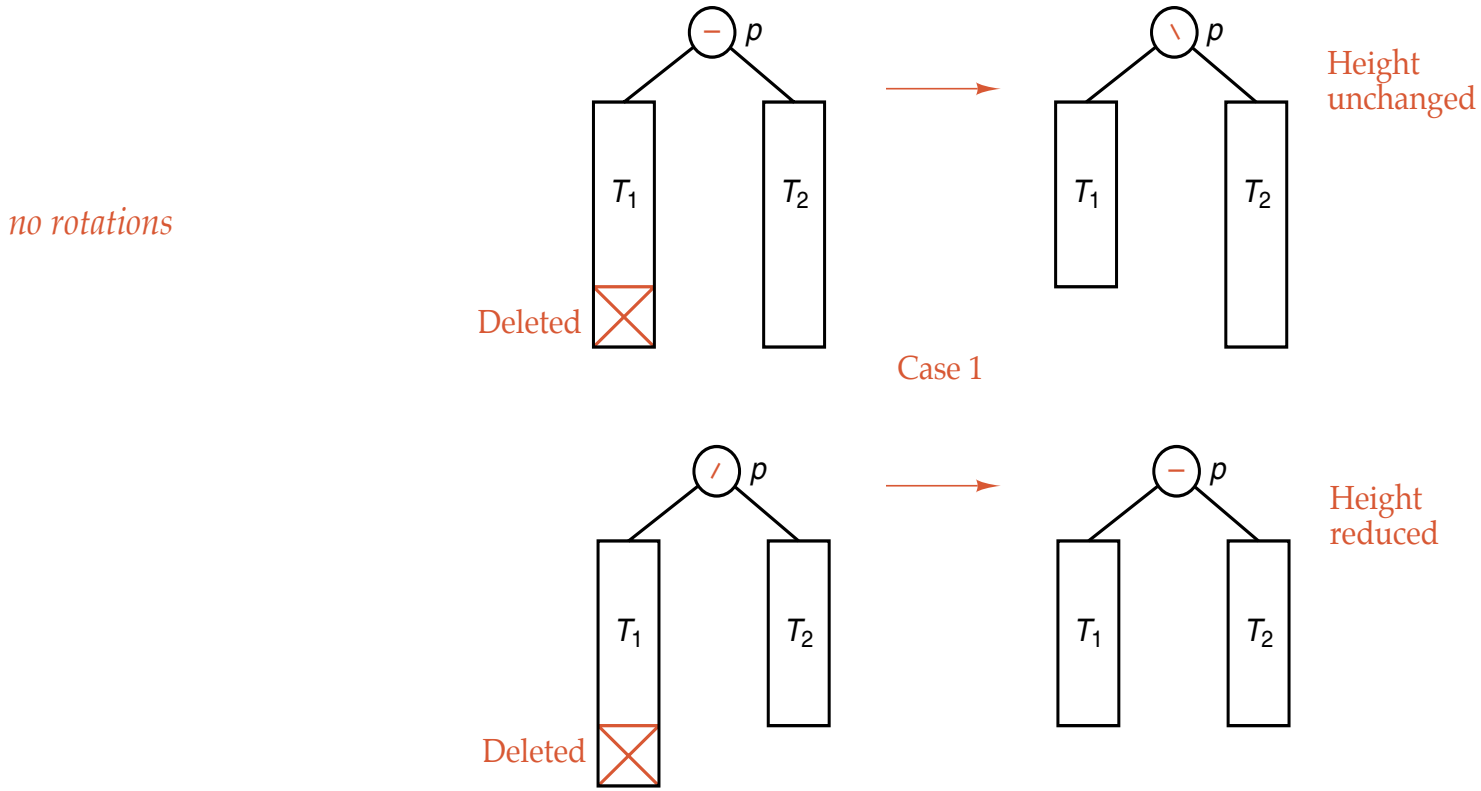

```
static void rotaDerIzq(Link & k1) {
   rotaDer(k1->dr);
   rotaIzq(k1);
}
```

Implementación de los árboles AVL

```
static void reequilibraDer(Link & a) {
   if (altura(a->iz) - altura(a->dr) > 1) {
      if (altura(a->iz->dr) > altura(a->iz->iz))
         rotaIzqDer(a);
      else rotaDer(a);
   }
   else a->altura = std::max(altura(a->iz), altura(a->dr)) + 1;
}

static void reequilibraIzq(Link & a) {
   if (altura(a->dr) - altura(a->iz) > 1) {
      if (altura(a->dr->iz) > altura(a->dr->dr))
         rotaDerIzq(a);
      else rotaIzq(a);
   }
   else a->altura = std::max(altura(a->iz), altura(a->dr)) + 1;
}
```

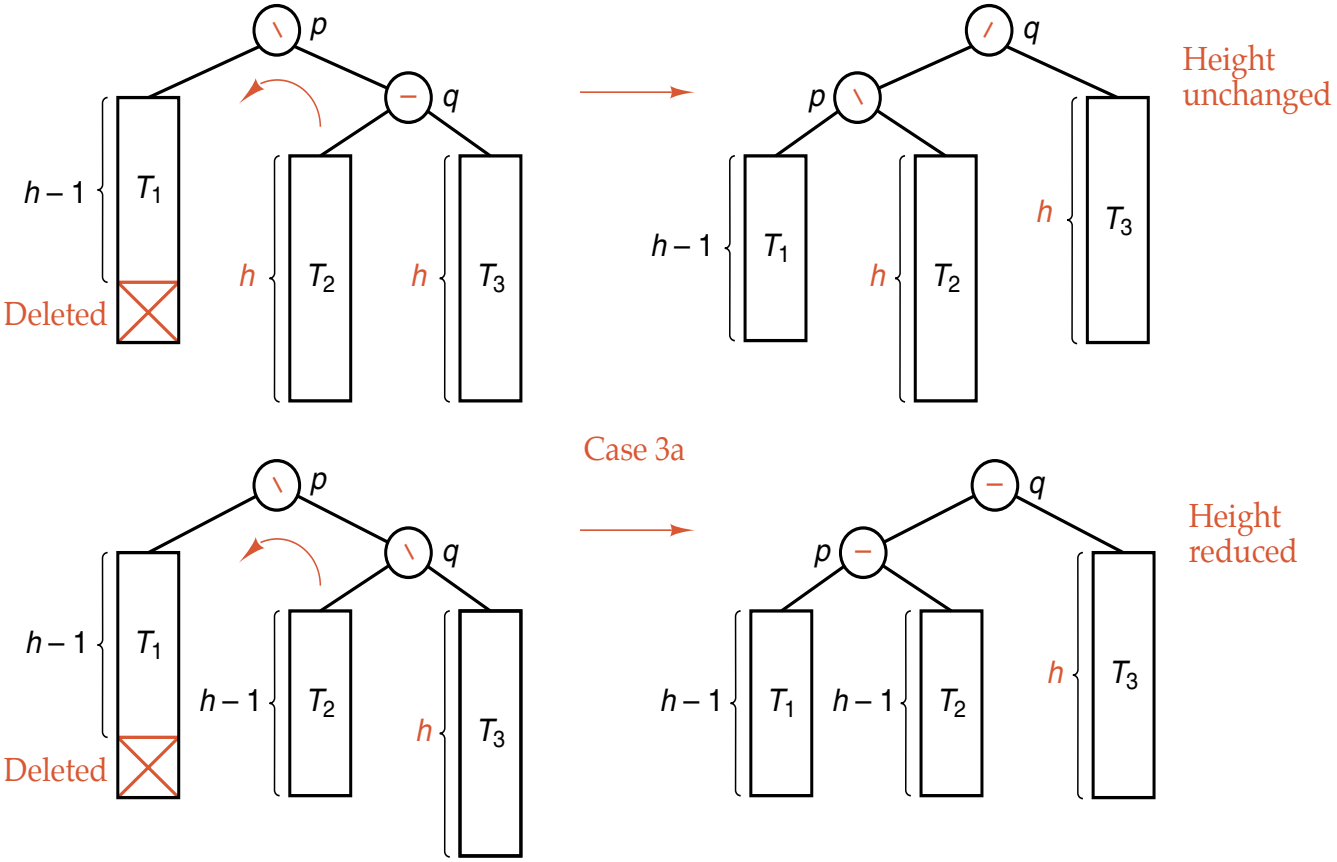
Borrado en árboles AVL

- Sigue los mismos pasos que el borrado en un ABB. Se reduce al caso de que el nodo a eliminar tiene solamente un hijo, por lo que se puede sustituir por él. La altura decrece por lo que pueden hacer falta rotaciones si algún subárbol se desequilibra.

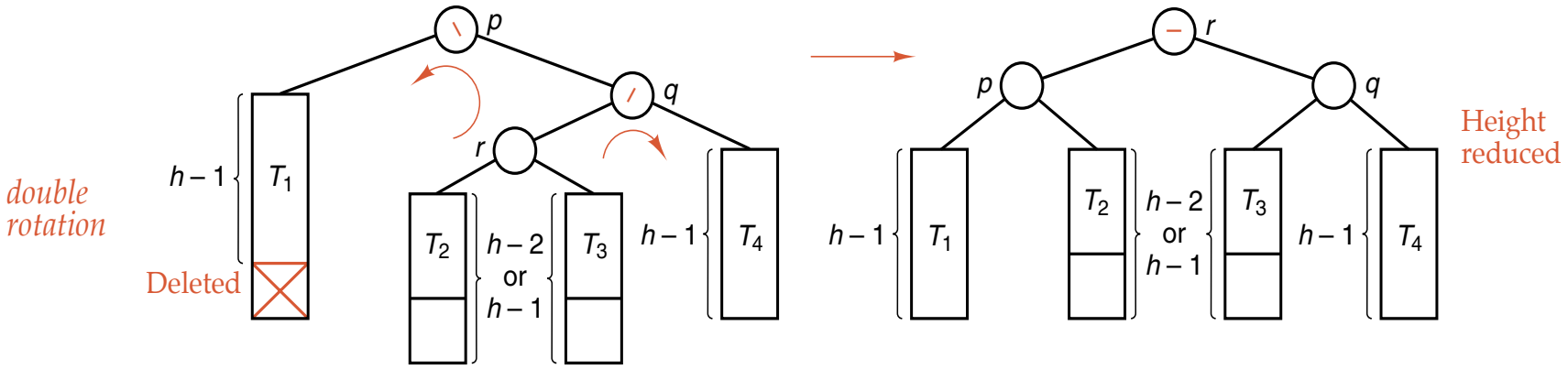


Borrado en árboles AVL

single left rotations



Borrado en árboles AVL



Problemas

- 10 - ¿Está el árbol equilibrado?
- 11 - ¿Es un árbol AVL?
- 12 - Rango de claves en un árbol binario de búsqueda
- 13 - Encontrar el k-ésimo elemento en un árbol AVL