# Algoritmos y Estructuras de Datos I

Universidad Católica del Uruguay

Resumen clase 7 de junio de 20121
Arboles Binarios AVL

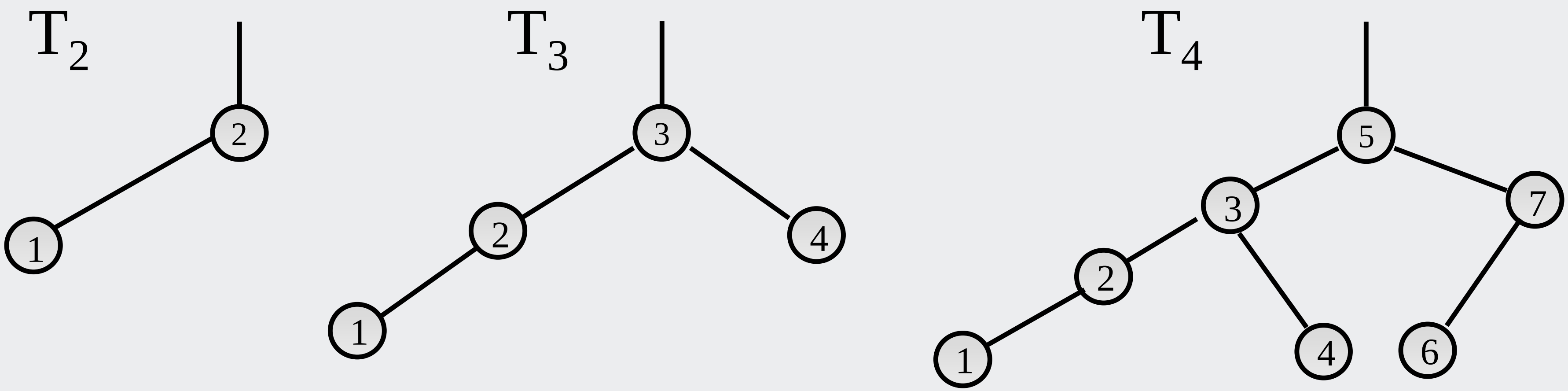
# Análisis de las operaciones en Arboles Binarios de Búsqueda

- Se puede demostrar que si la inserciones de los nodos se realiza en forma aleatoria, la longitud de trayectoria es aproximadamente 1.38 * N * logN, y sucesivas inserciones y eliminaciones (también aleatorias) no afectarán mayormente.
- Sin embargo, si las inserciones se realizan en forma ordenada, se produce el peor caso en el que la longitud de trayectoria resulta proporcional a N al cuadrado.
- Por ello se han ideado métodos para que el árbol se pueda ir balanceando a medida que se inserta o se elimina de él, a efectos de asegurar el orden logarítmico de las operaciones.

# Arboles Balanceados

- Criterio de Adelson, Velskii y Landis:
  - Un árbol está balanceado si y sólo si para cada nodo las alturas de sus dos subárboles difieren a lo sumo en 1.
  - Asegura un O log(n) en el peor caso para las tres operaciones: búsqueda, inserción y eliminación.
  - Se demuestra con la ayuda de los árboles de Fibonacci.

# Arboles de Fibonacci de alturas 1, 2 y 3




$$S_H = S_{H-1} + S_{H-2} + 1$$

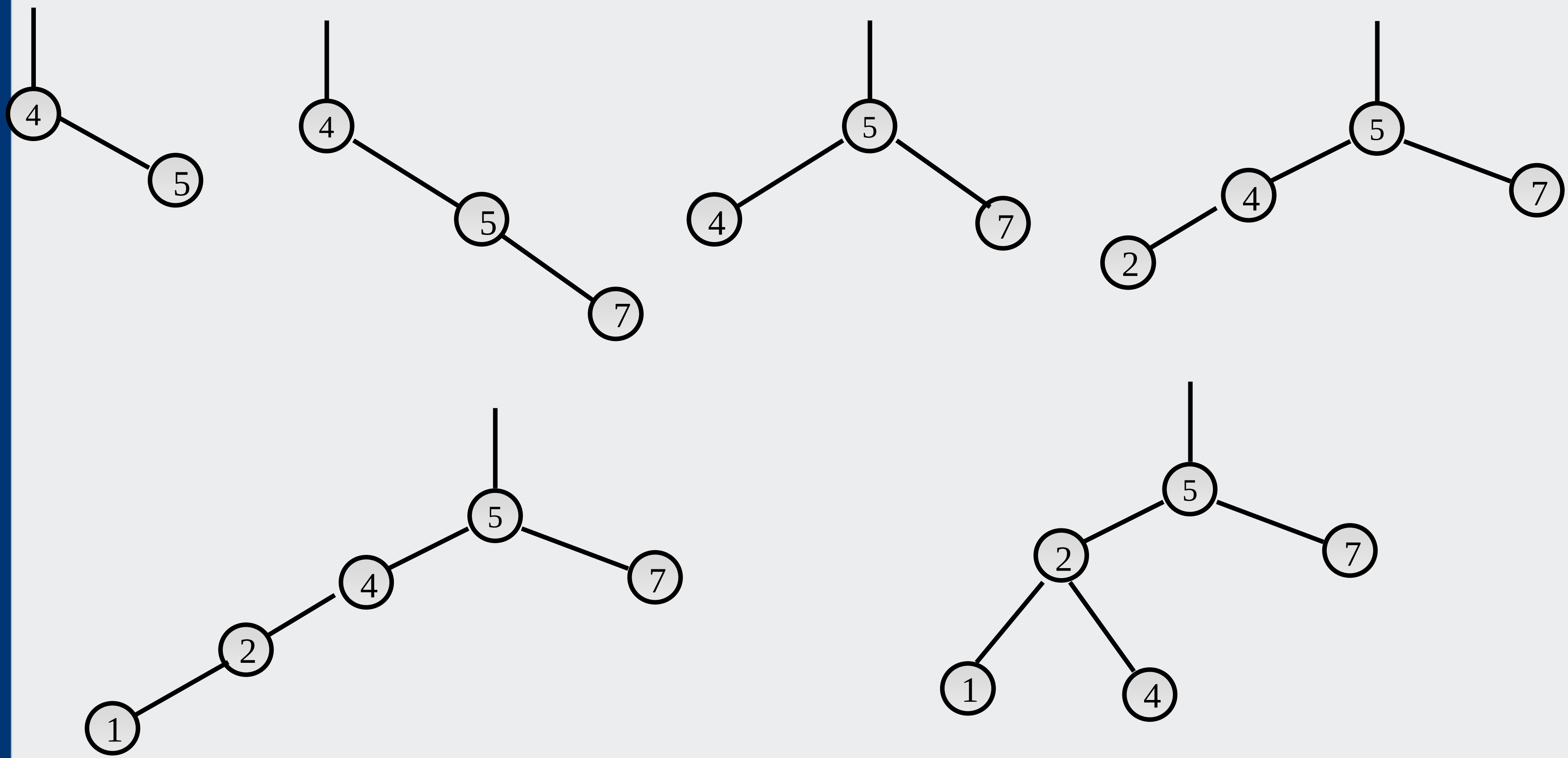
# Arboles AVL

- Criterio de Adelson, Velskii y Landis:
  - El teorema de Adelson-Velski y Landis garantiza que un árbol balanceado nunca tendrá una altura mayor que el **45 %** respecto a su equivalente perfectamente balanceado.
  - Si la altura de un árbol balanceado de n nodos es h (n), entonces

$$h(n) <= 1.44*\log(n+2) - 1.328$$

# ARBOLES BALANCEADOS

- Inserción balanceada.



# AVL

- Una inserción o eliminación puede destruir el balance
- Se debe revisar la condición de balance y corregir si es necesario
- Después de la Inserción, sólo los nodos que se encuentran en el camino desde el punto de inserción hasta la raíz pueden tener el balance alterado
- Si se repara correctamente el equilibrio del nodo desequilibrado más profundo, se recupera el equilibrio de todo el árbol

# Diferentes condiciones de desequilibrio

- Supongamos que X es el nodo cuyo equilibrio queremos ajustar
- Se pueden dar cuatro casos:
  1. Inserción en el subárbol izquierdo del hijo izquierdo de X
  2. Inserción en el subárbol derecho del hijo izquierdo de X
  3. Inserción en el subárbol izquierdo del hijo derecho de X
  4. Inserción en el subárbol derecho del hijo derecho de X

# Diferentes condiciones de desequilibrio

- Supongamos que X es el nodo cuyo equilibrio queremos ajustar
- Se pueden dar cuatro casos:
  1. **Inserción en el subárbol izquierdo del hijo izquierdo de X**
  2. Inserción en el subárbol derecho del hijo izquierdo de X
  3. Inserción en el subárbol izquierdo del hijo derecho de X
  4. **Inserción en el subárbol derecho del hijo derecho de X**

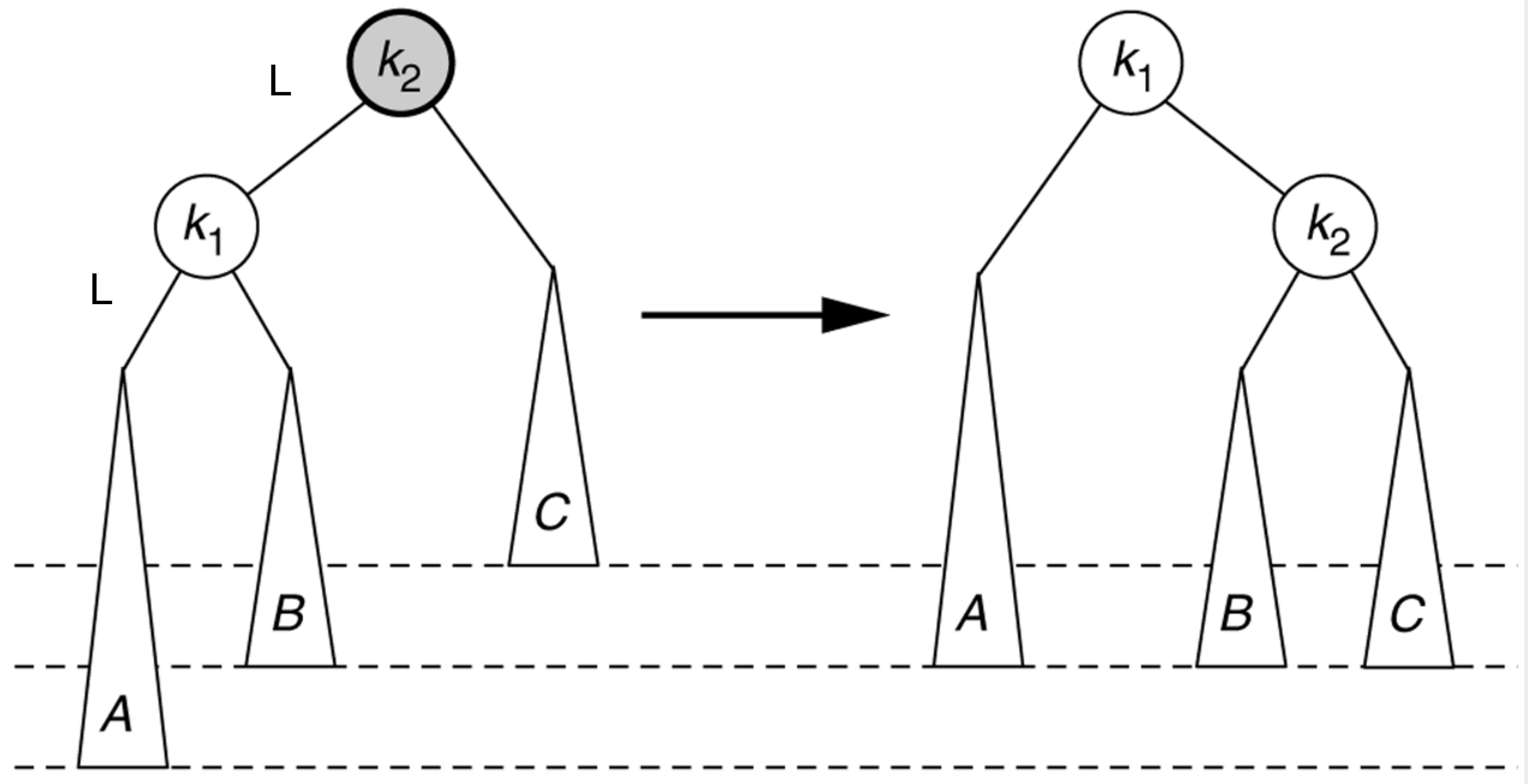
- 1 y 4 son simétricos respecto a X

# Diferentes condiciones de desequilibrio

- Supongamos que X es el nodo cuyo equilibrio queremos ajustar
- Se pueden dar cuatro casos:
  1. Inserción en el subárbol izquierdo del hijo izquierdo de X
  2. **Inserción en el subárbol derecho del hijo izquierdo de X**
  3. **Inserción en el subárbol izquierdo del hijo derecho de X**
  4. Inserción en el subárbol derecho del hijo derecho de X

- 1 y 4 son simétricos respecto a X
- 2 y 3 son simétricos respecto a X

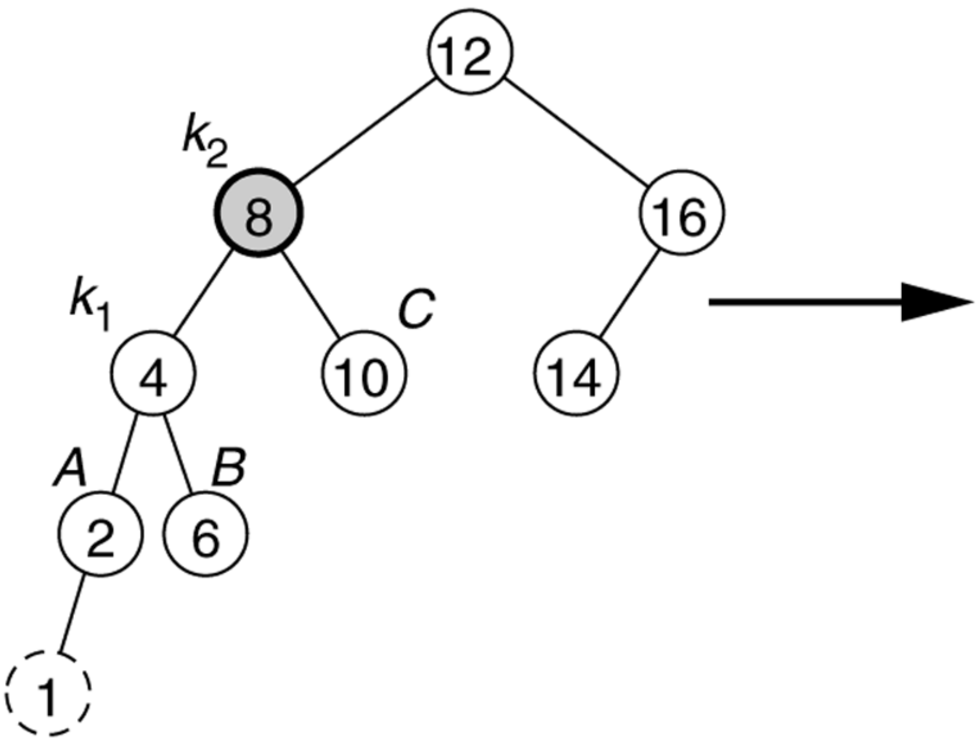
# Rotación Simple para resolver el caso 1




a) Antes de la rotación

b)Después de la rotación

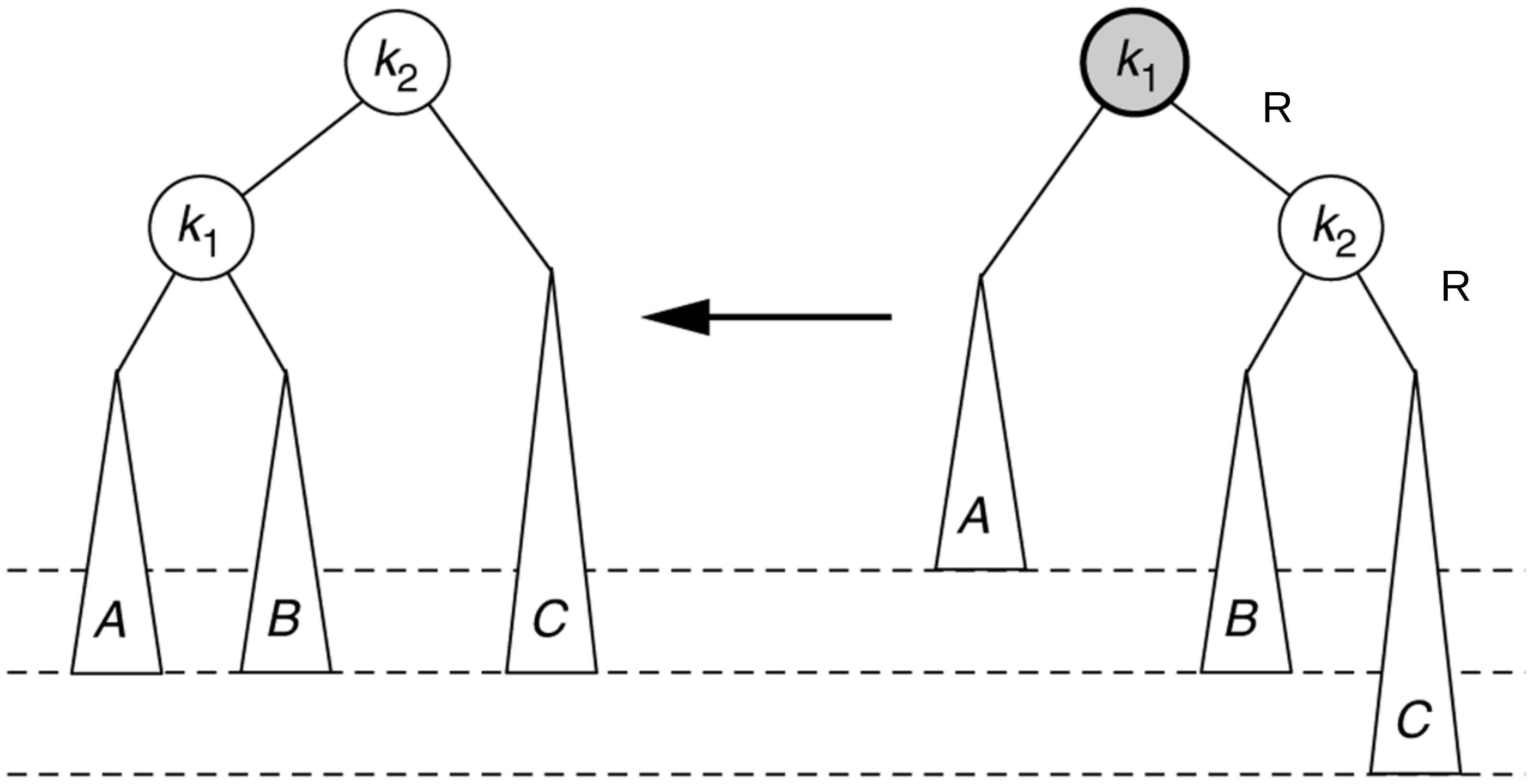
# La rotación simple balancea el árbol después de la inserción del 1.




a) Antes de la rotación

b)Después de la rotación

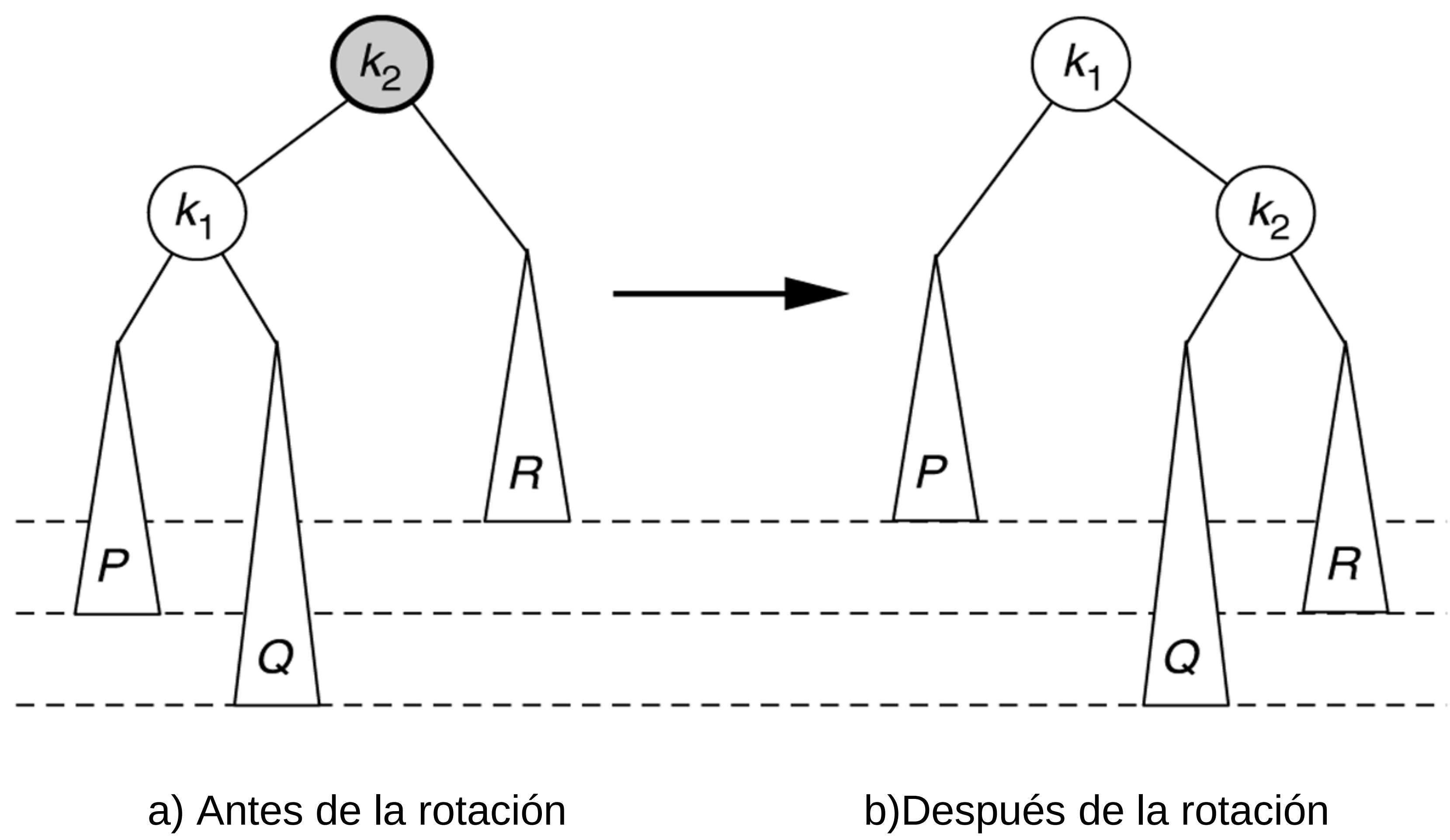
# Rotación simple simétrica para resolver el caso 4



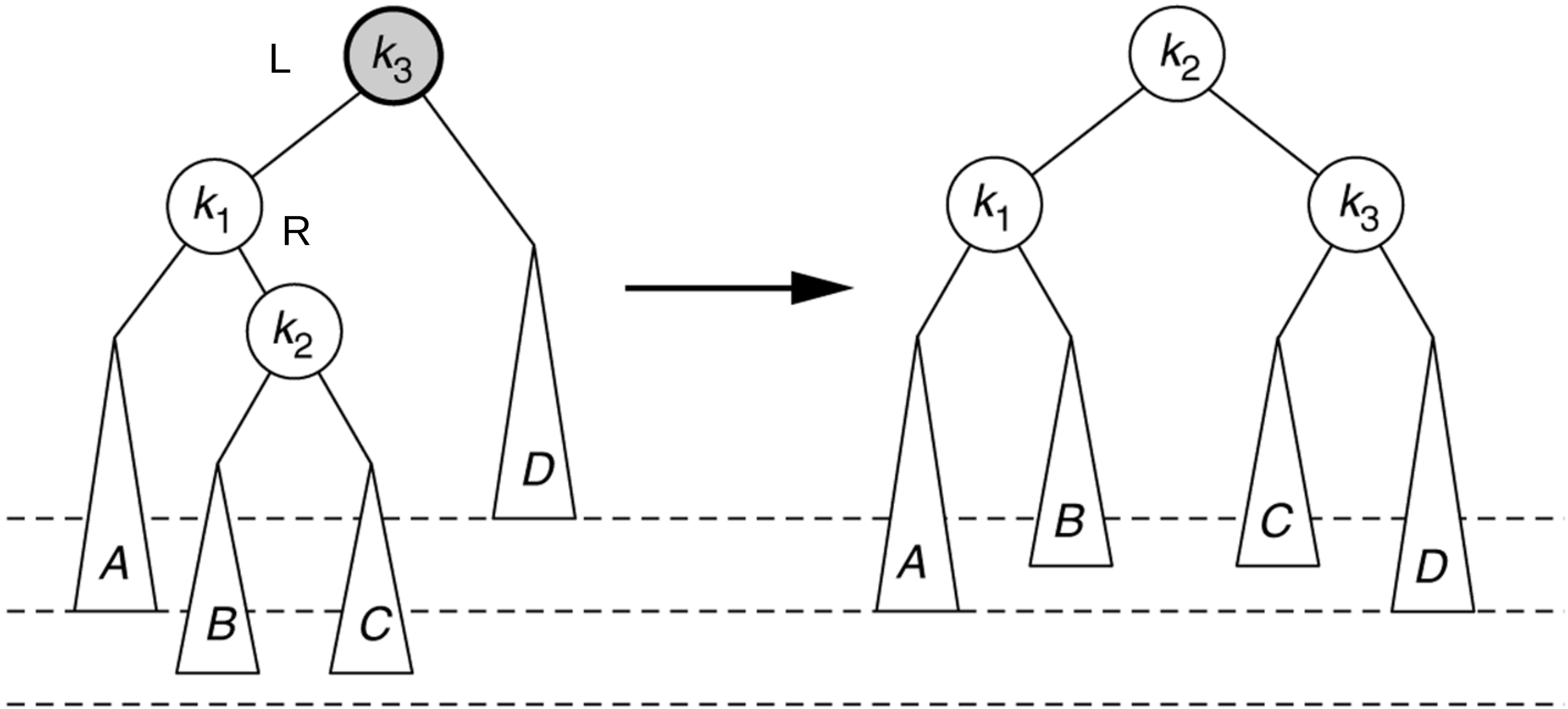

a) Antes de la rotación

b)Después de la rotación

# La rotación simple no resuelve el caso 2.



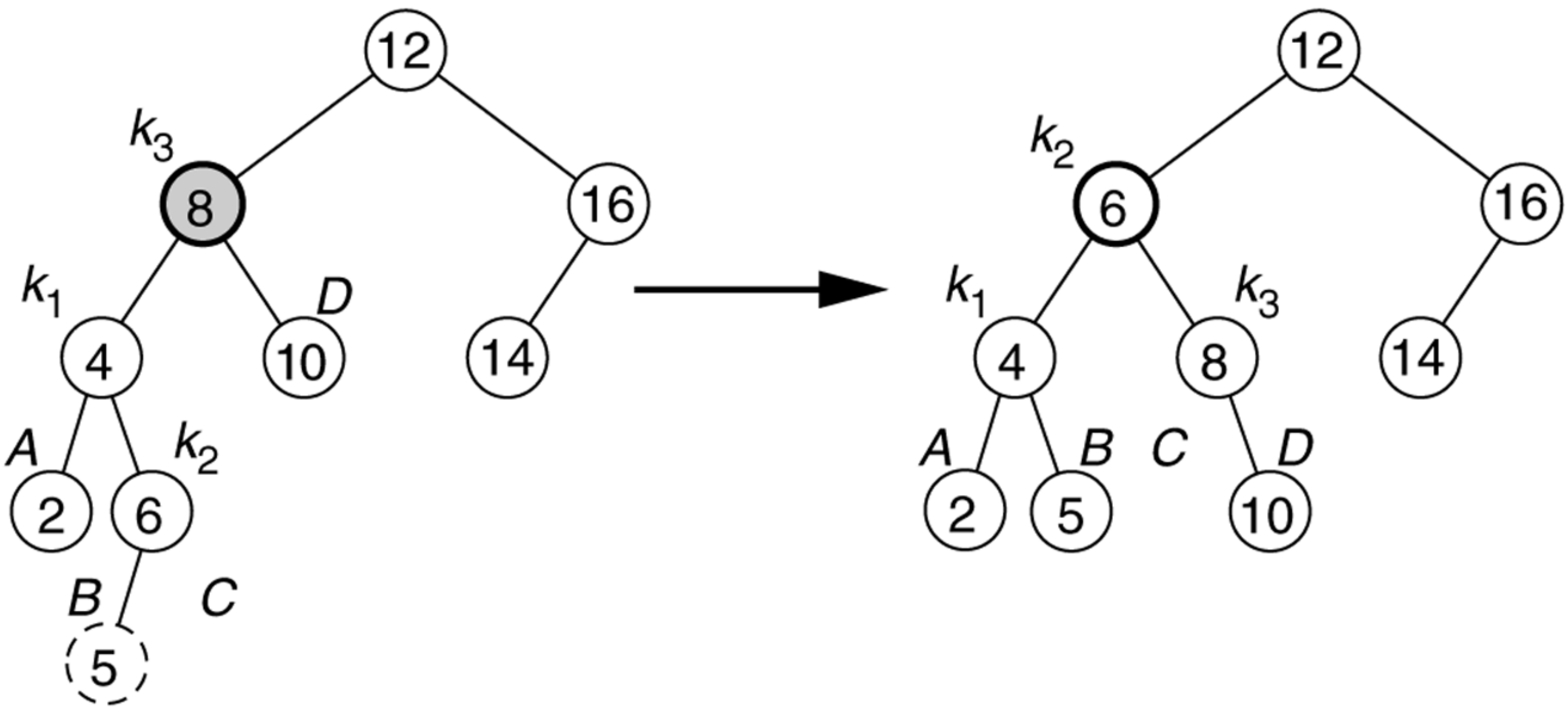
# Rotación doble LR para resolver el caso 2




a) Antes de la rotación

b)Después de la rotación

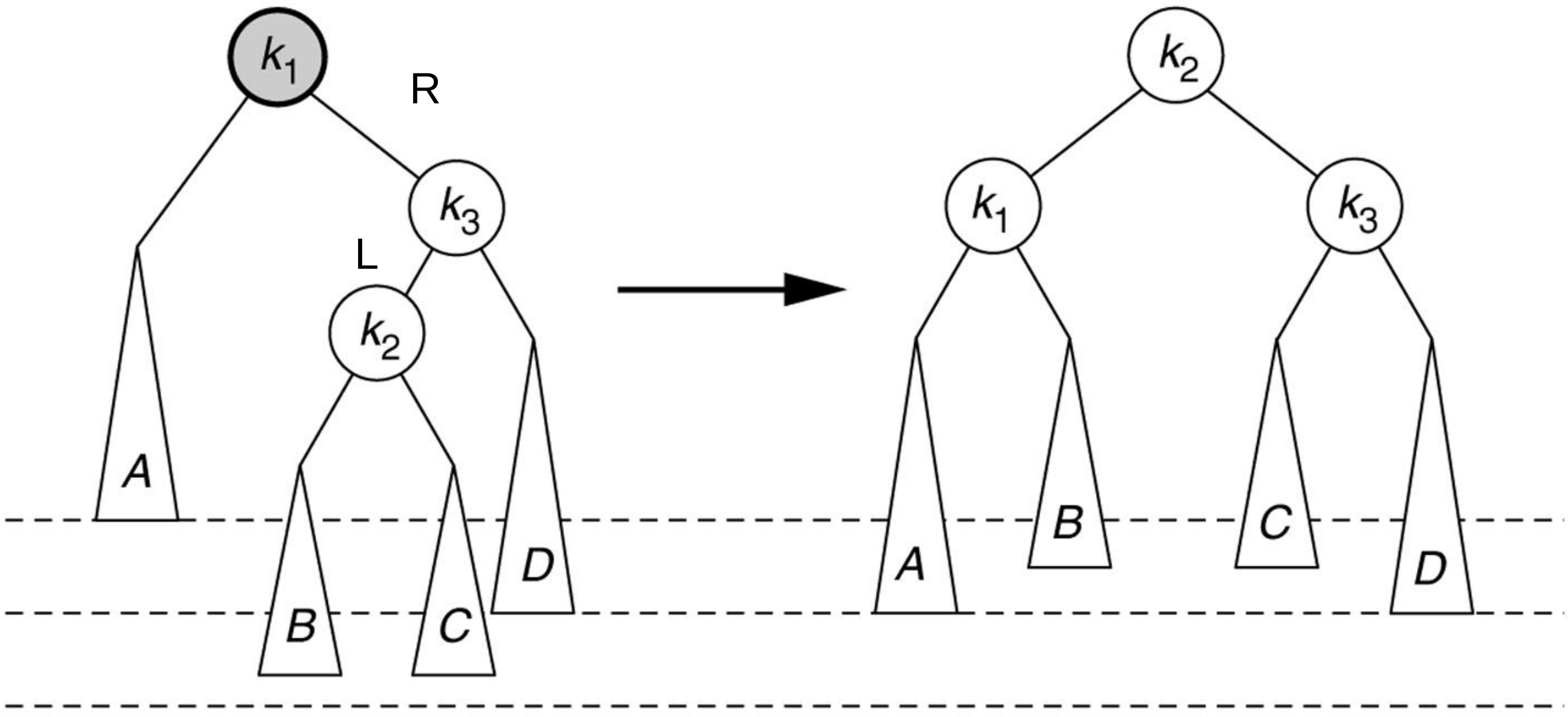
# Rotación doble restablece el balance después de la inserción del 5.




a) Antes de la rotación

b)Después de la rotación

# Rotación doble RL para resolver el caso 3.




a) Antes de la rotación

b)Después de la rotación

## CASO 1 LL

```
TElementoAB rotacionLL ( TElementoAB k2 )
   {
 TElementoAB k1 = k2.hijoIzq;
       k2. hijoIzq = k1.hijoDer;
       k1. hijoDer = k2;
       return k1;
   }
```

## CASO 4 RR

```
TElementoAB rotacionRR(TElementoAB k1 )
   {
 TElementoAB k2 = k1.hijoDer;
       k1. hijoDer = k2.hijoIzq;
       k2.hijoIzq = k1;
       return k2;
   }
```

## CASO 2 LR

```
TElementoAB rotacionLR ( TElementoAB k3 )
   {
      k3. hijoIzq = rotacionRR( k3. hijoIzq );
      return rotacionLL( k3 );
   }
```

## CASO 3 RL

```
TElementoAB rotacionRL ( TElementoAB k1 )
   {
      k1. hijoDer = rotacionLL( k1. hijoDer );
      return rotacionRR( k1 );
   }
```