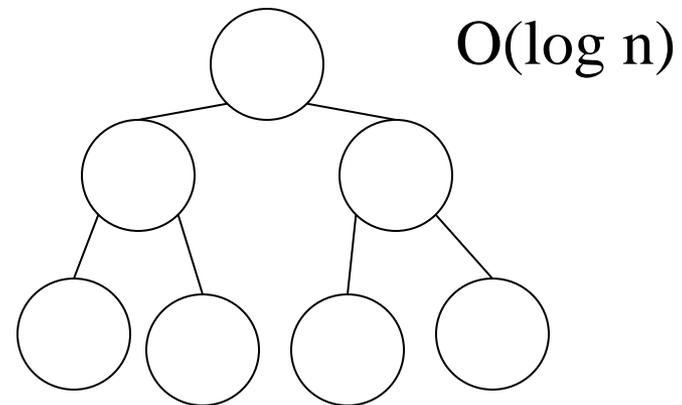
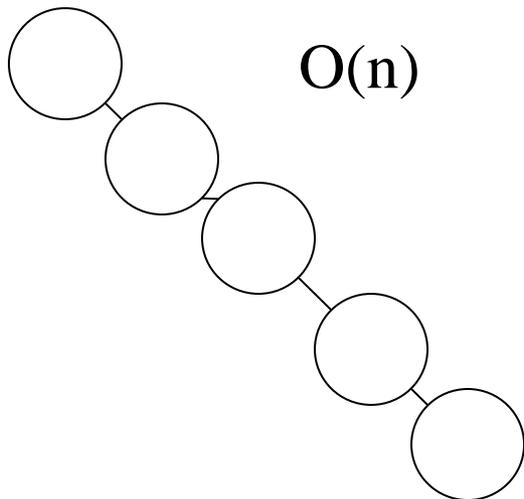


# Árboles equilibrados

---

- Búsqueda de elementos en un árbol binario de búsqueda tiene coste entre  $O(\log n)$  y  $O(n)$
- Depende de cómo fue el orden de inserción de elementos

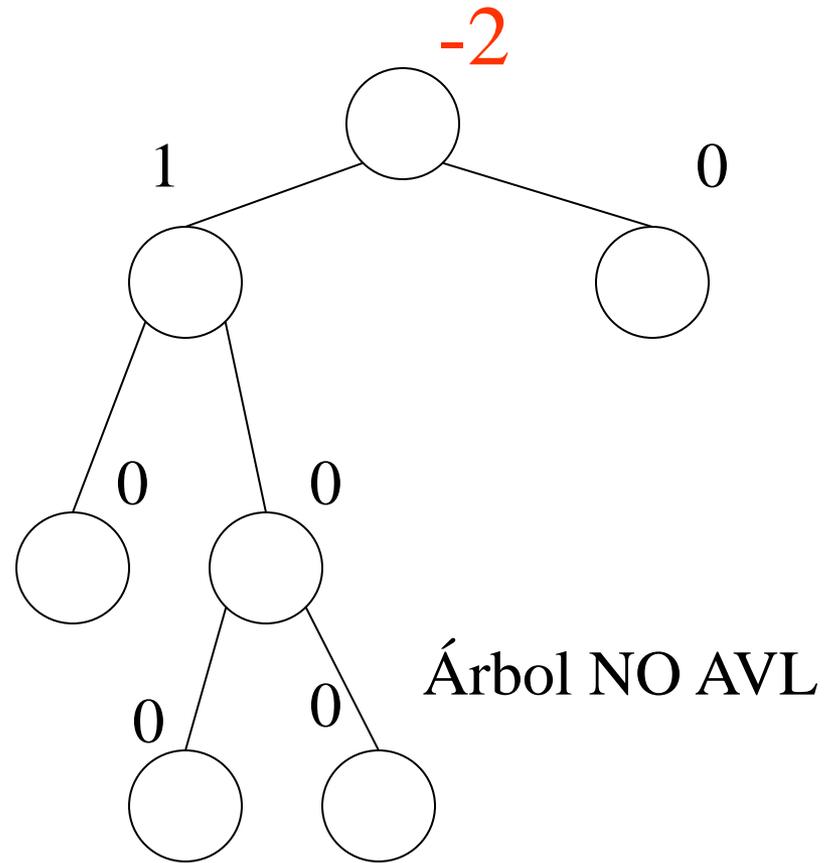
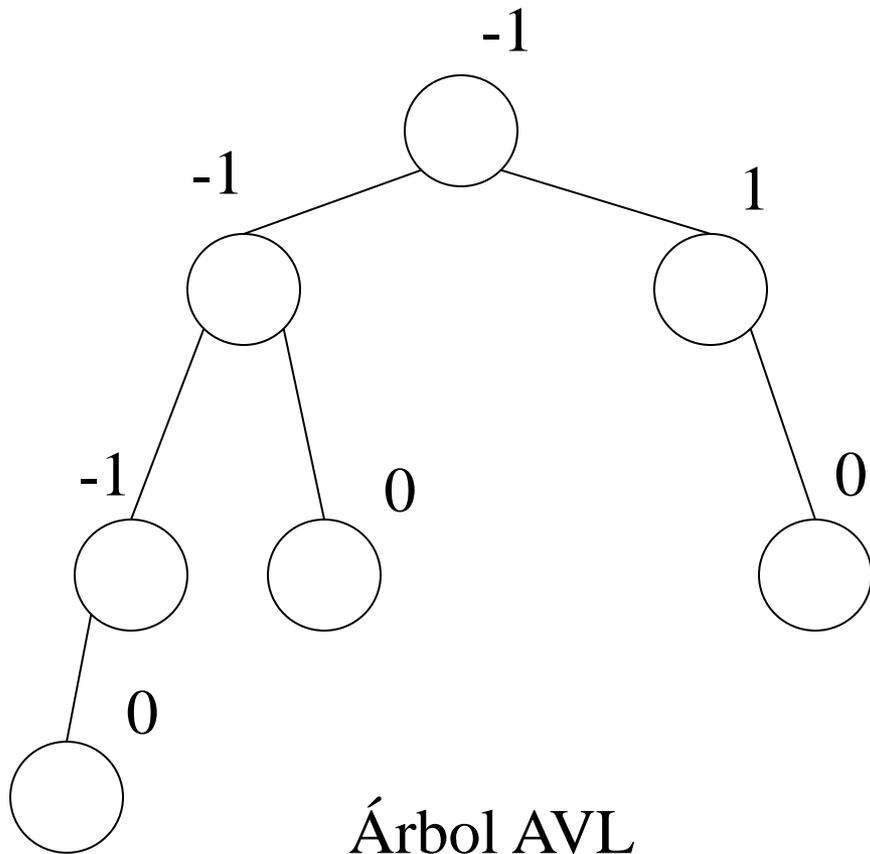


# Árboles equilibrados: AVL

---

- Los árboles equilibrados o balanceados surgen para mejorar el rendimiento de operaciones que involucren una búsqueda
- Un ejemplo de árbol equilibrado es el Árbol de Adelson-Velskii y Landis: Árbol AVL
- El árbol AVL es un árbol binario de búsqueda con una condición de equilibrio:
  - Las alturas de los 2 subárboles para cada nodo no difieren en más de una unidad.
  - Factor de equilibrio o balance de un nodo se define como altura del subárbol derecho menos altura del subárbol izquierdo para ese nodo
  - Cada nodo del árbol AVL puede tener un balance de  $-1, 0, 1$

# Árboles AVL



# Árboles AVL

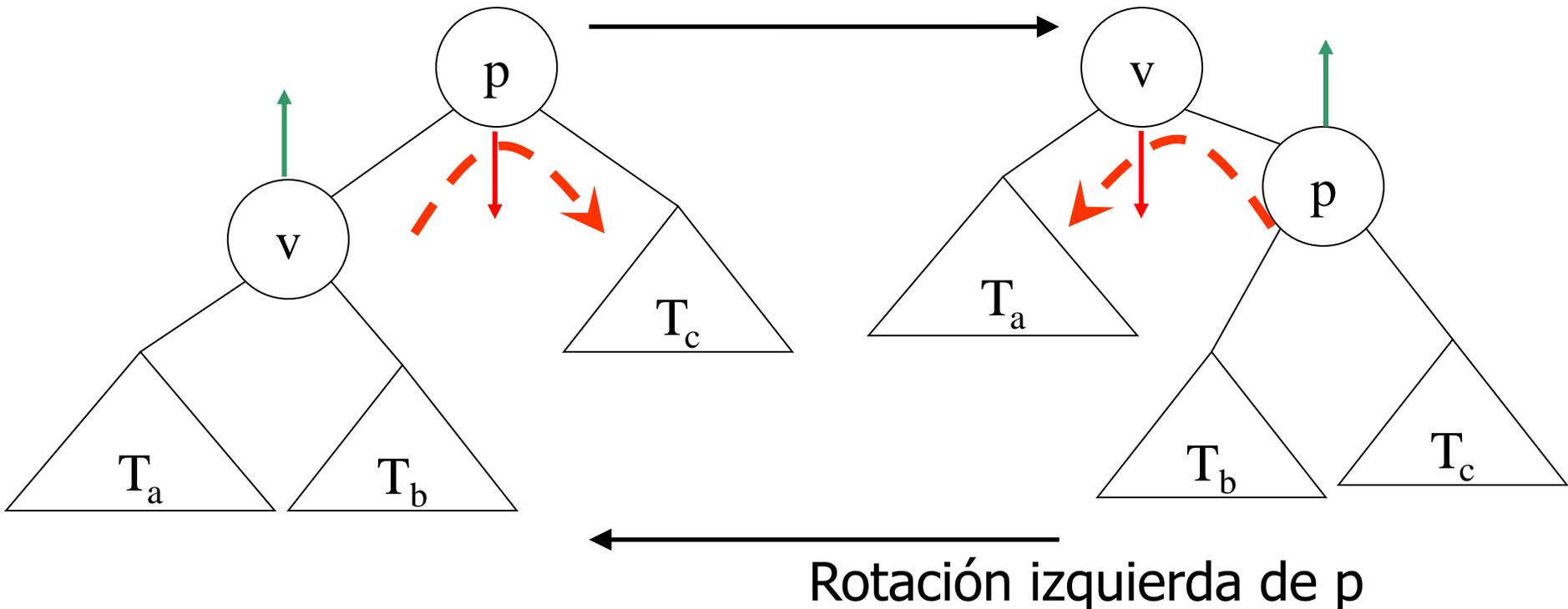
---

- Hay que mantener información de la altura de cada nodo
- Número mínimo de nodos en árbol AVL en función de la altura  $h$ :
  - $N(h) = N(h-1) + N(h-2) + 1$
- Una inserción de un nodo puede alterar la condición de equilibrio del árbol
- Para garantizarla es posible realizar una transformación: rotación
  - Rotación simple: izq-izq ó dcha-dcha
  - Rotación doble: izq-dcha ó dcha-izq

# Árboles AVL

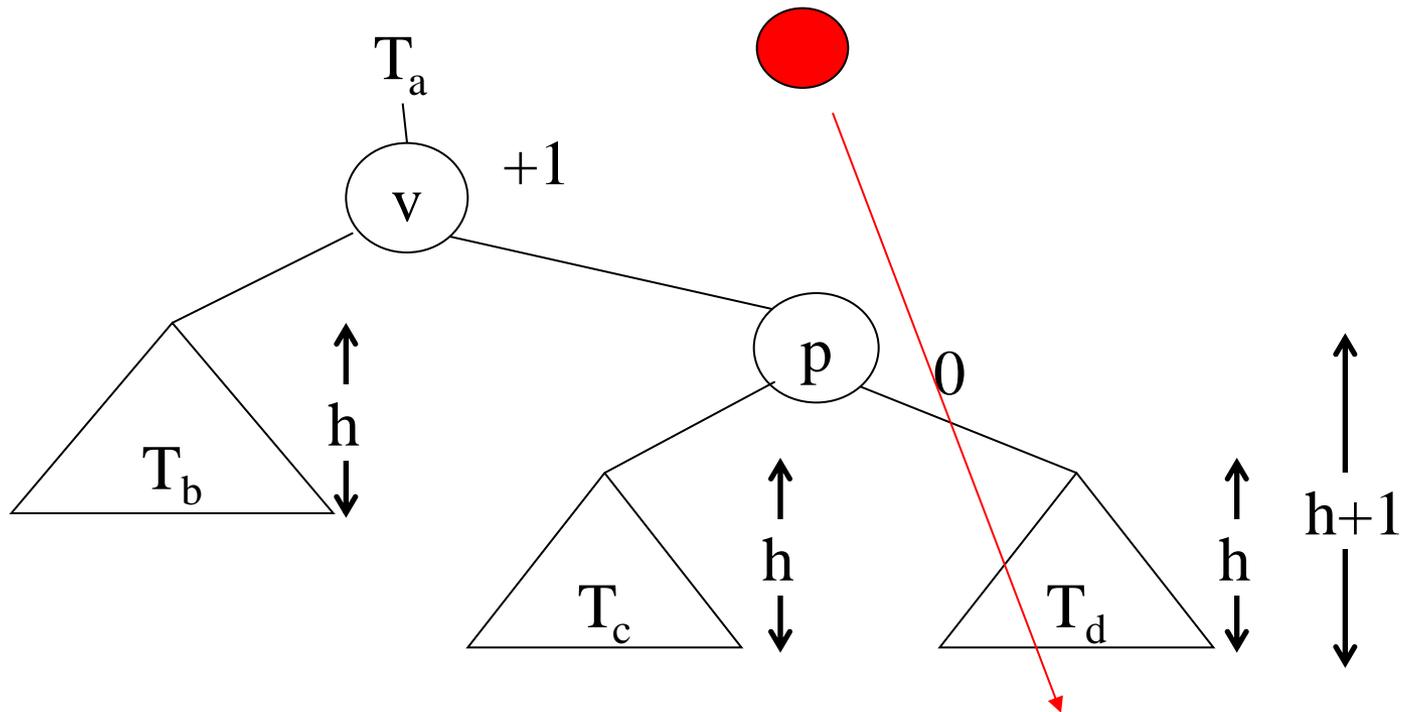
- Rotaciones simples

Rotación derecha de v



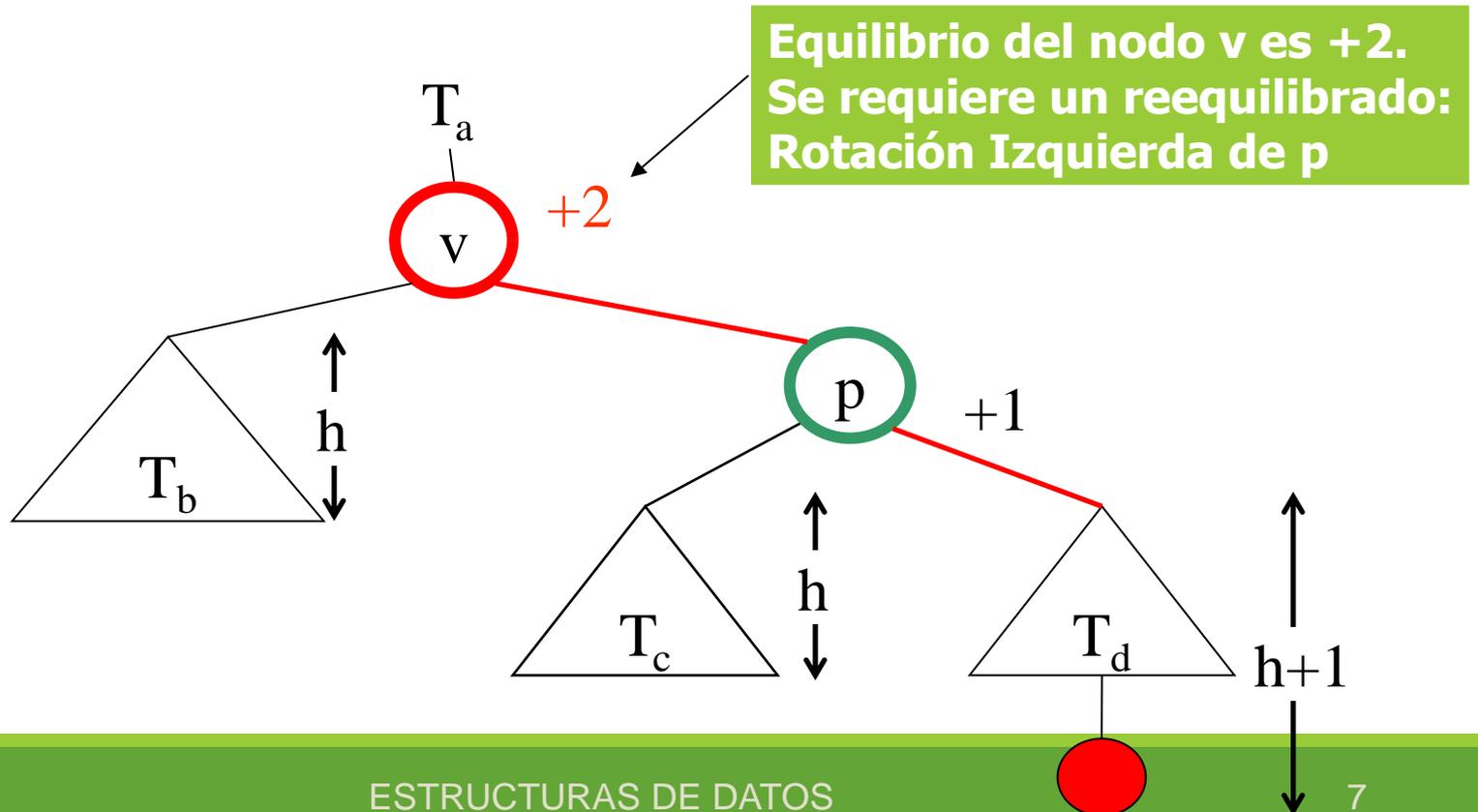
# Árboles AVL

- Ejemplo: rotación izquierda(-izquierda)  
1- Situación inicial:



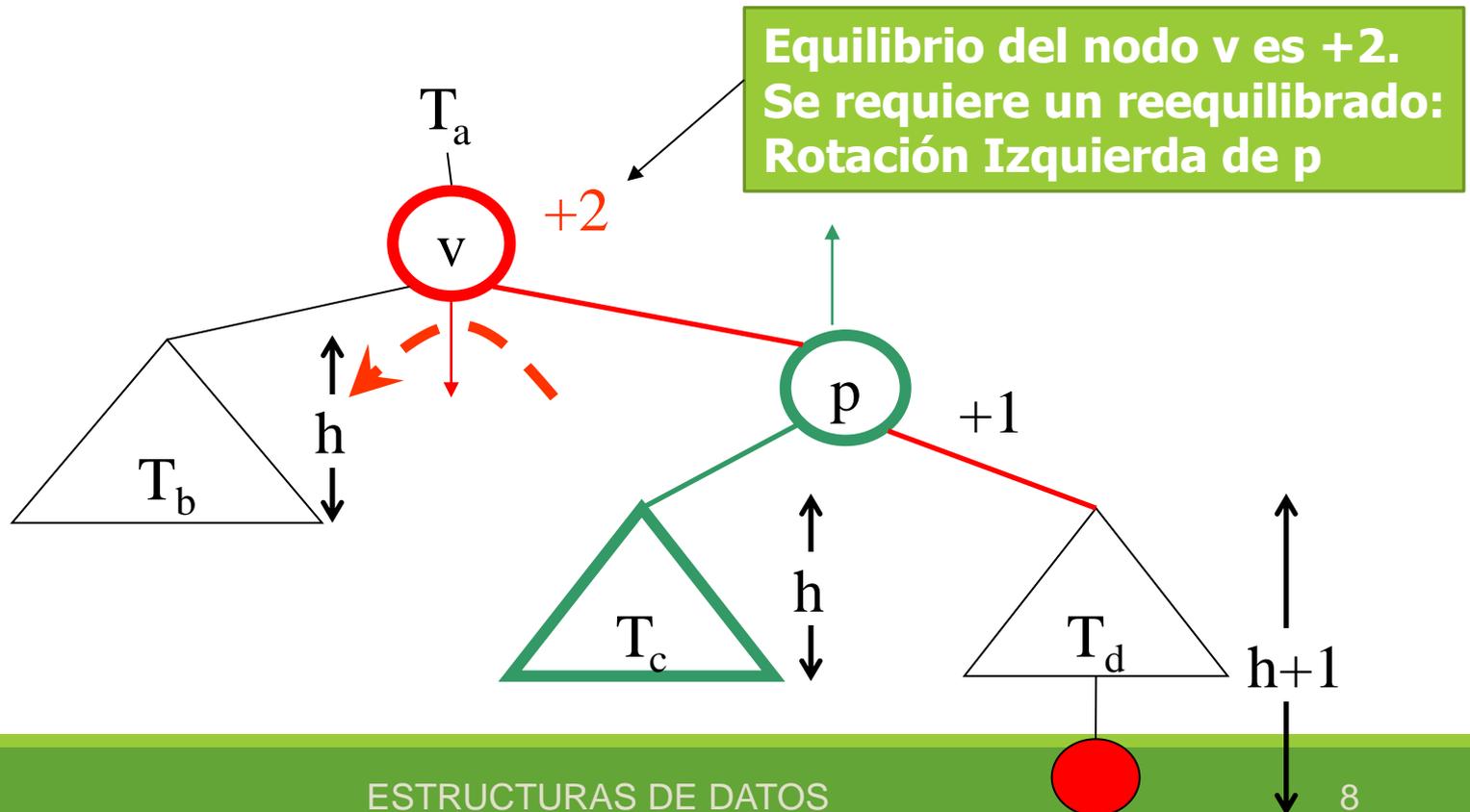
# Árboles AVL

- Ejemplo: rotación izquierda(-izquierda)  
2- Inserción de un nuevo nodo y calculo de las nuevas condiciones de equilibrio:



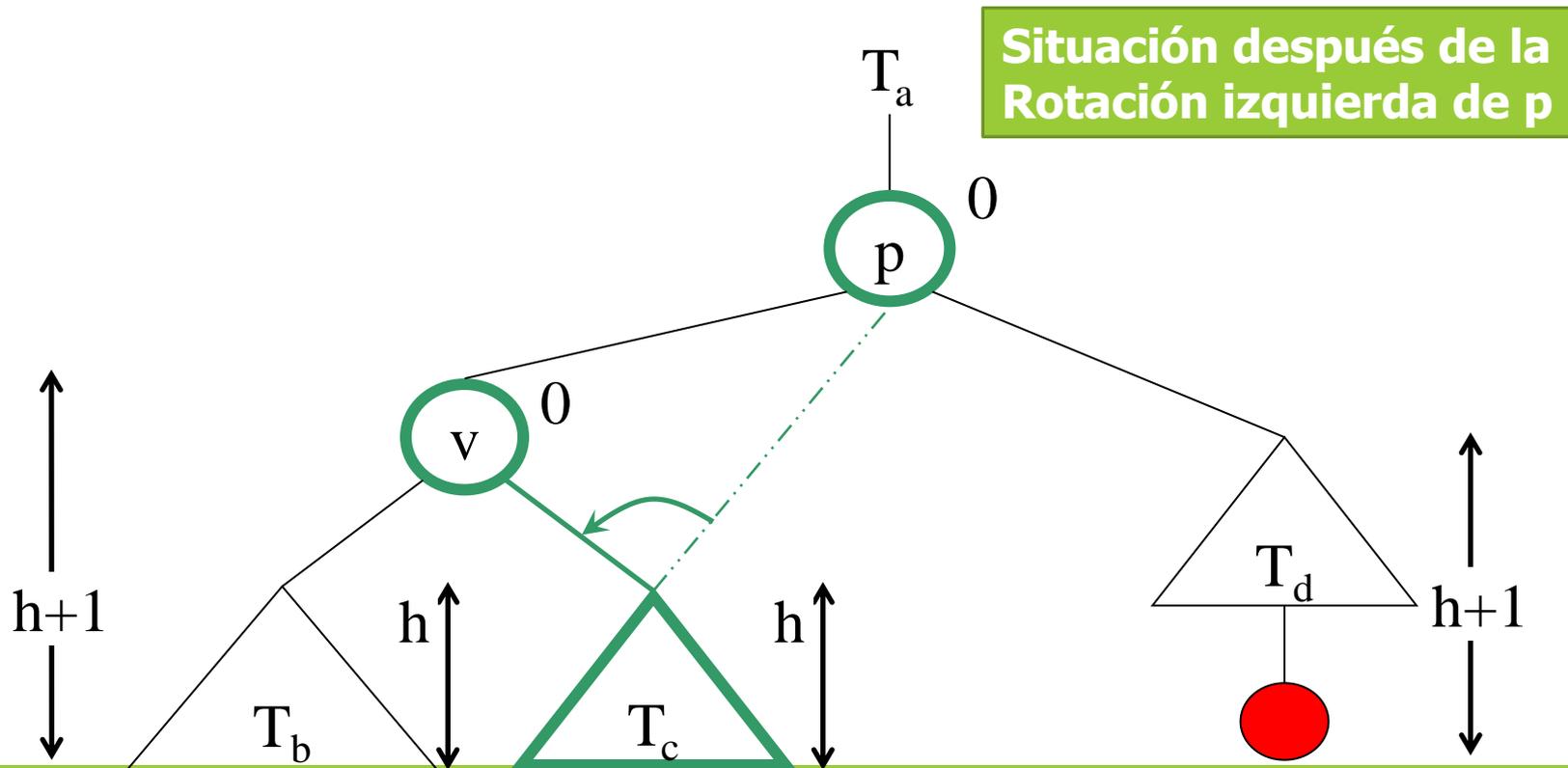
# Árboles AVL

- Ejemplo: rotación izquierda(-izquierda)  
2- Inserción de un nuevo nodo y calculo de las nuevas condiciones de equilibrio:



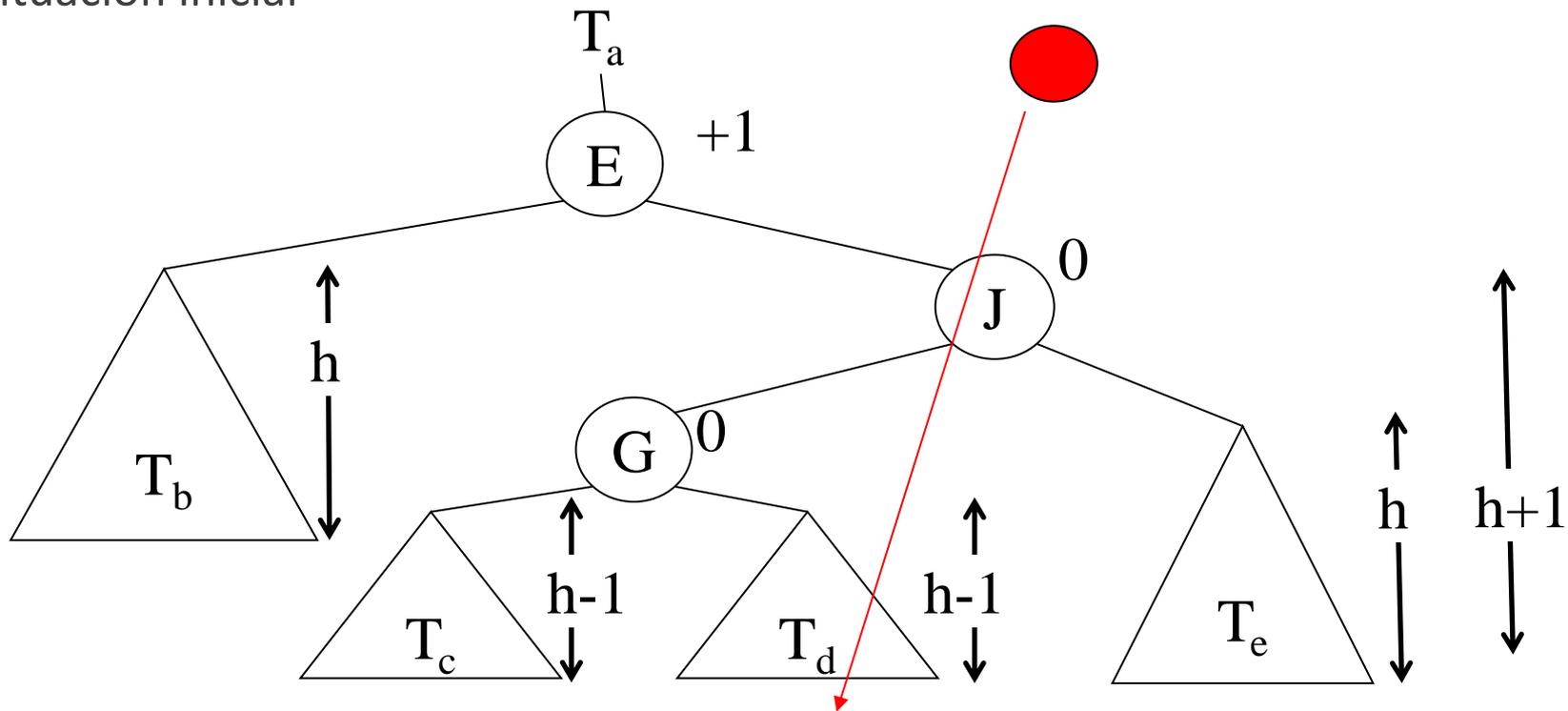
# Árboles AVL

- Ejemplo: rotación izquierda(-izquierda)  
3- Rotación Izquierda de p para recuperar condición de árbol AVL:



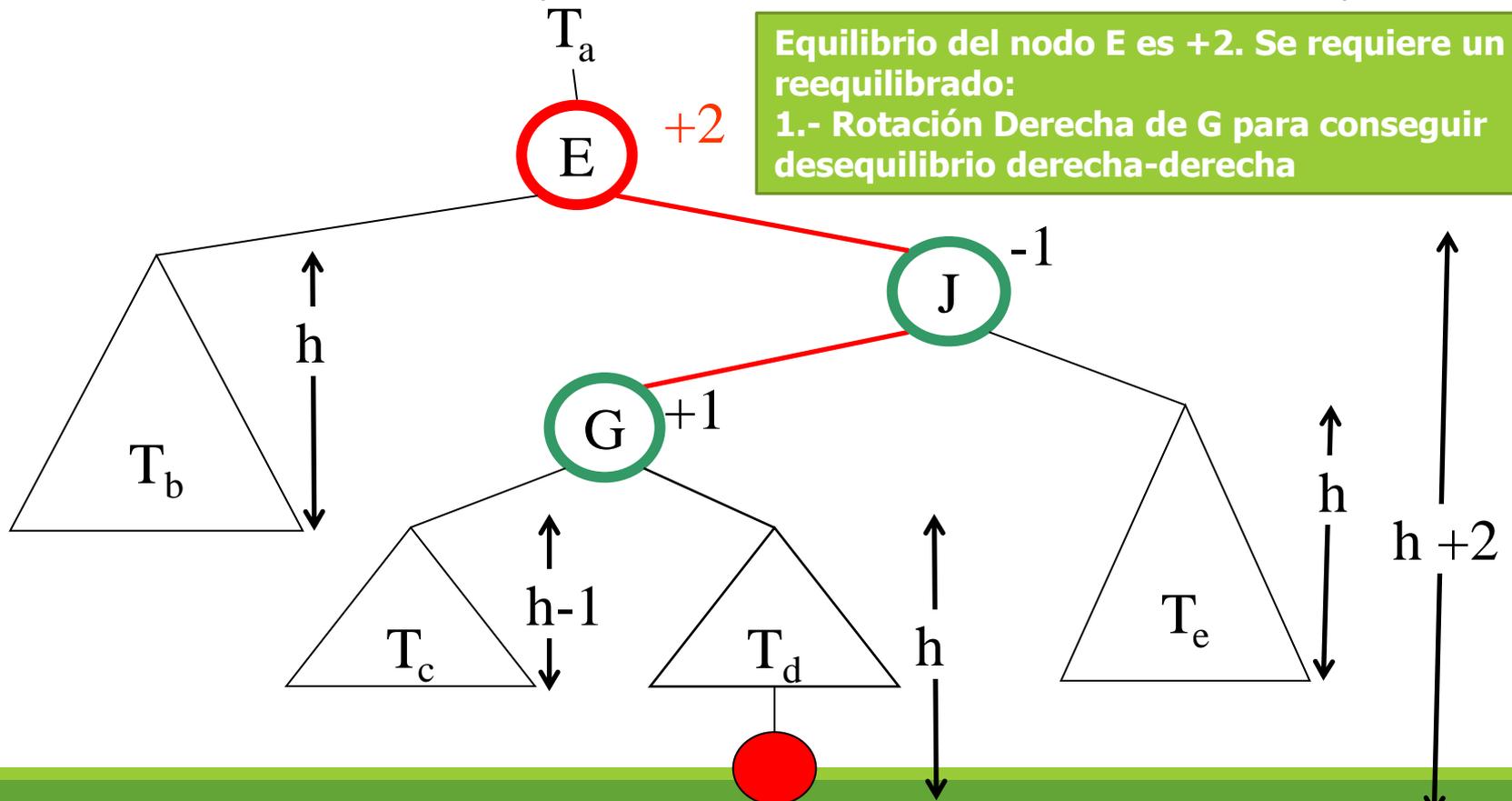
# Árboles AVL

- Rotación doble: derecha-izquierda  
1.- Situación inicial



# Árboles AVL

- Rotación doble: derecha-izquierda
- 2.- Inserción de un nuevo nodo y calculo de las nuevas condiciones de equilibrio

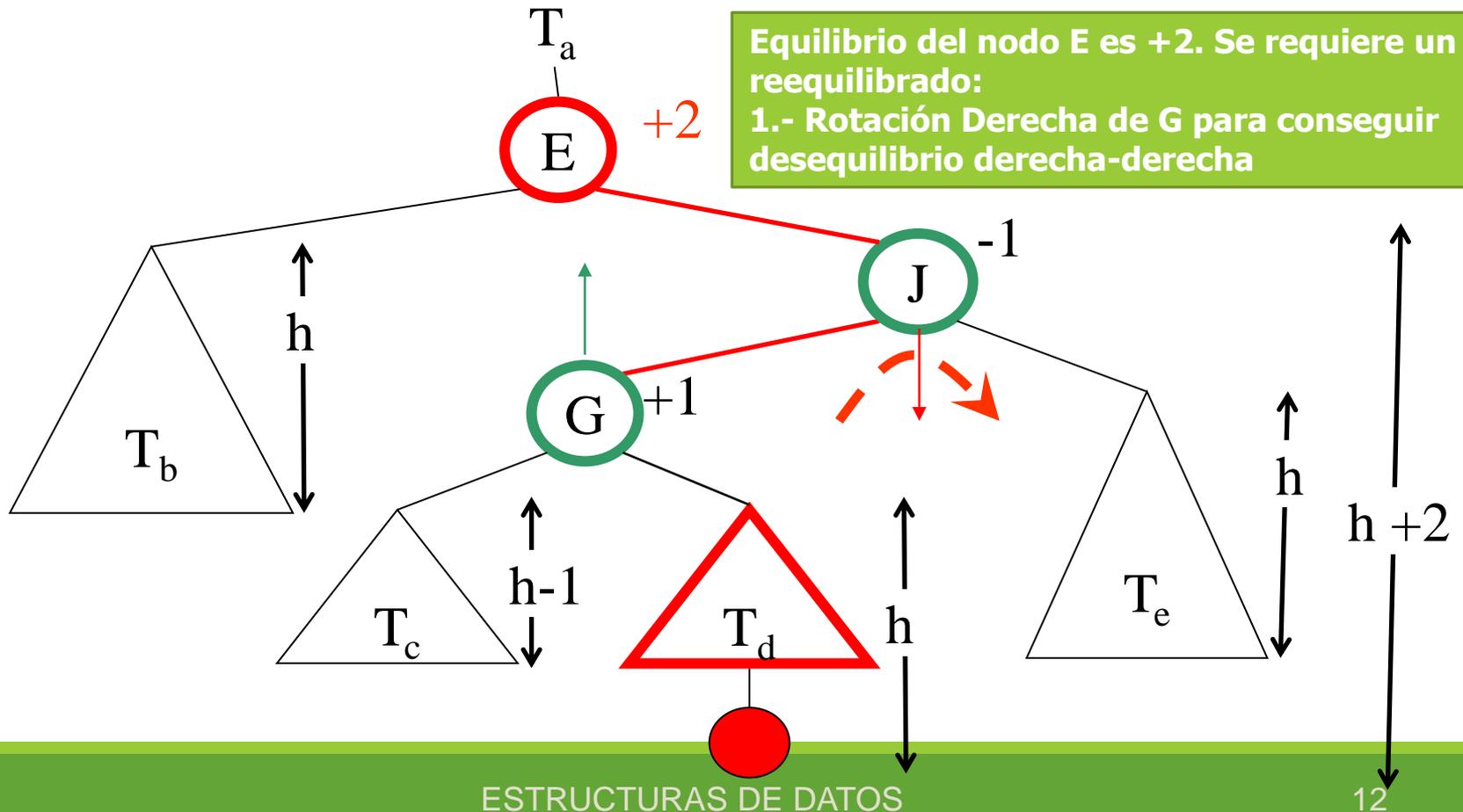


Equilibrio del nodo E es +2. Se requiere un reequilibrado:  
1.- Rotación Derecha de G para conseguir desequilibrio derecha-derecha

# Árboles AVL

## Rotación doble: derecha-izquierda

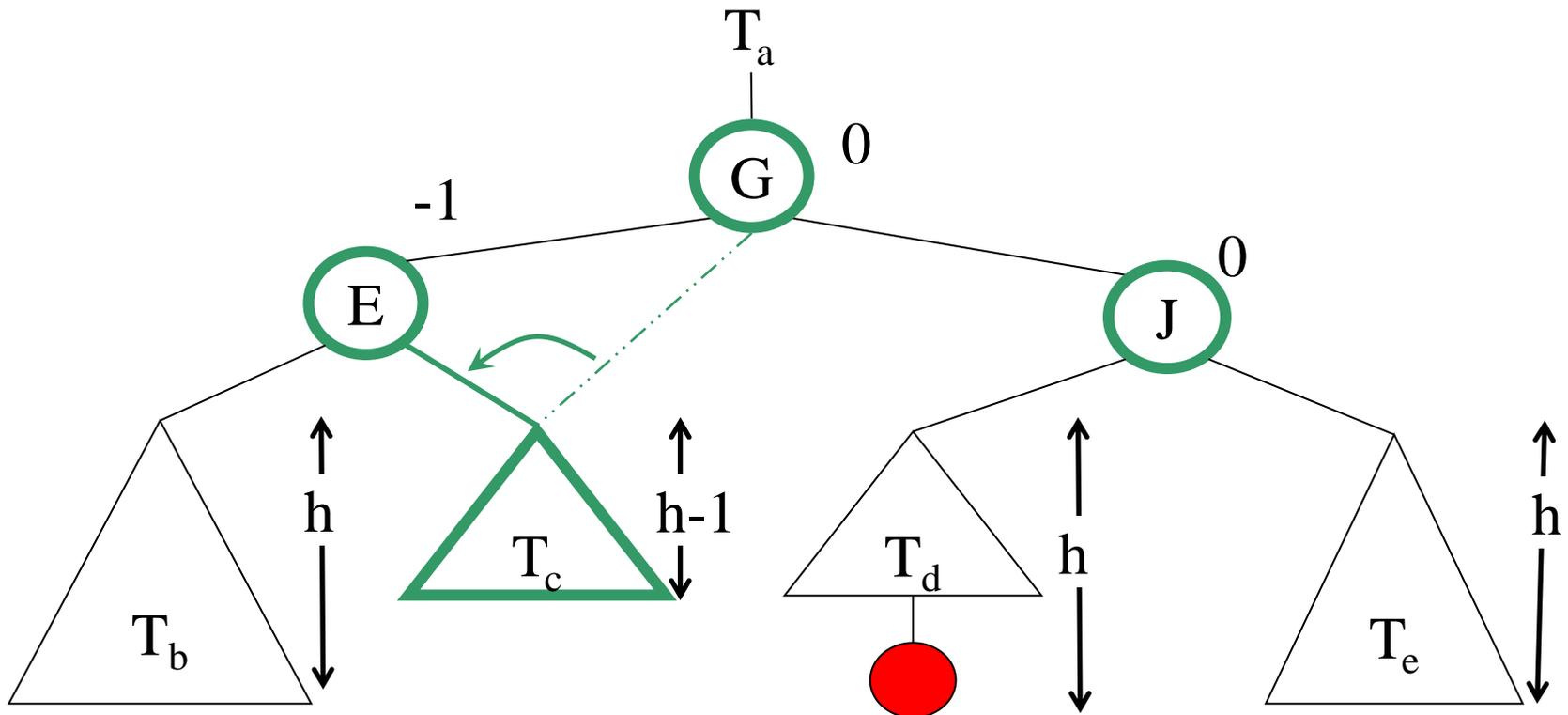
2.- Inserción de un nuevo nodo y calculo de las nuevas condiciones de equilibrio





# Árboles AVL

- Rotación doble: derecha-izquierda  
4.-Situación final. Árbol AVL



# Implementación

---

```
PROCEDURE Insertar(e: TipoElemento; VAR a: TipoArbolBin); {Complejidad: O(log(n))}
BEGIN
  IF EsArbolBinVacio(a) THEN
    {Crear ArbolBin de un nodo con elemento e: CAB(a,NIL,e,NIL)}
  ELSE BEGIN
    IF Menor(e,a^.elemento) THEN BEGIN
      Insertar(e, a^.izq);
      IF (Altura(a^.izq) - Altura(a^.der) = 2) THEN {está balanceado??}
        IF Menor(e,a^.izq^.elemento) THEN
          RotarSDer(a) {rotación simple derecha}
        ELSE
          RotarDIZqDer(a) {rotación doble izquierda-derecha}
      ELSE {si está balanceado actualizamos la altura}
        a^.altura := max(Altura(a^.izq), Altura(a^.der)) + 1;
    END
  ELSE
    {Caso simétrico para la derecha}
  END;{árbol no vacío}
END;
```

# Implementación

---

```
PROCEDURE RotarSDer (VAR a: TipoArbolBin);
  {Rotación a derechas del nodo insertado, y actualiz. alturas}
  {Complejidad: O(1)}
VAR
  aux: TipoArbolBin;
BEGIN
  IF EsArbolBinVacio(a^.izq) THEN {Existe hijo izquierdo}
    {Error, debe existir hijo izquierdo}
  ELSE BEGIN
    aux := a^.izq;
    a^.izq := aux^.der;
    aux^.der := a;
    a^.altura := max(Altura(a^.izq), Altura(a^.der)) + 1;
    aux^.altura := max(Altura(aux^.izq), Altura(a)) + 1;
    a := aux;
  END;
END;
```

# Implementación

---

```
PROCEDURE RotarDIzqDer(VAR a:TipoArbolBin);
  {Rotación doble (izquierda del nodo insertado y derecha de la rotación resultante anterior)}
  {Complejidad: O(1)}
VAR
  aux: TipoArbolBin;
BEGIN
  IF EsArbolBinVacio(a^.izq) THEN {Existe hijo izquierdo}
    {Error, debe existir hijo izquierdo}
  ELSE BEGIN
    RotarSIzq(a^.izq);
    RotarSDer(a);
  END;
END;
```