Árboles AVL
Árboles Rojo-Negros (RN)

Estructuras de Datos

Andrea Rueda

Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas

¿Cómo garantizar árboles “bonitos”?

- Balanceando:
  - Evitar “listas”.
  - Evitar ramas cortas.
- Garantizar búsqueda / inserción / eliminación en O(log n).
- Árboles AVL y RN.

Árboles AVL
(Adelson-Velskii and Landis)

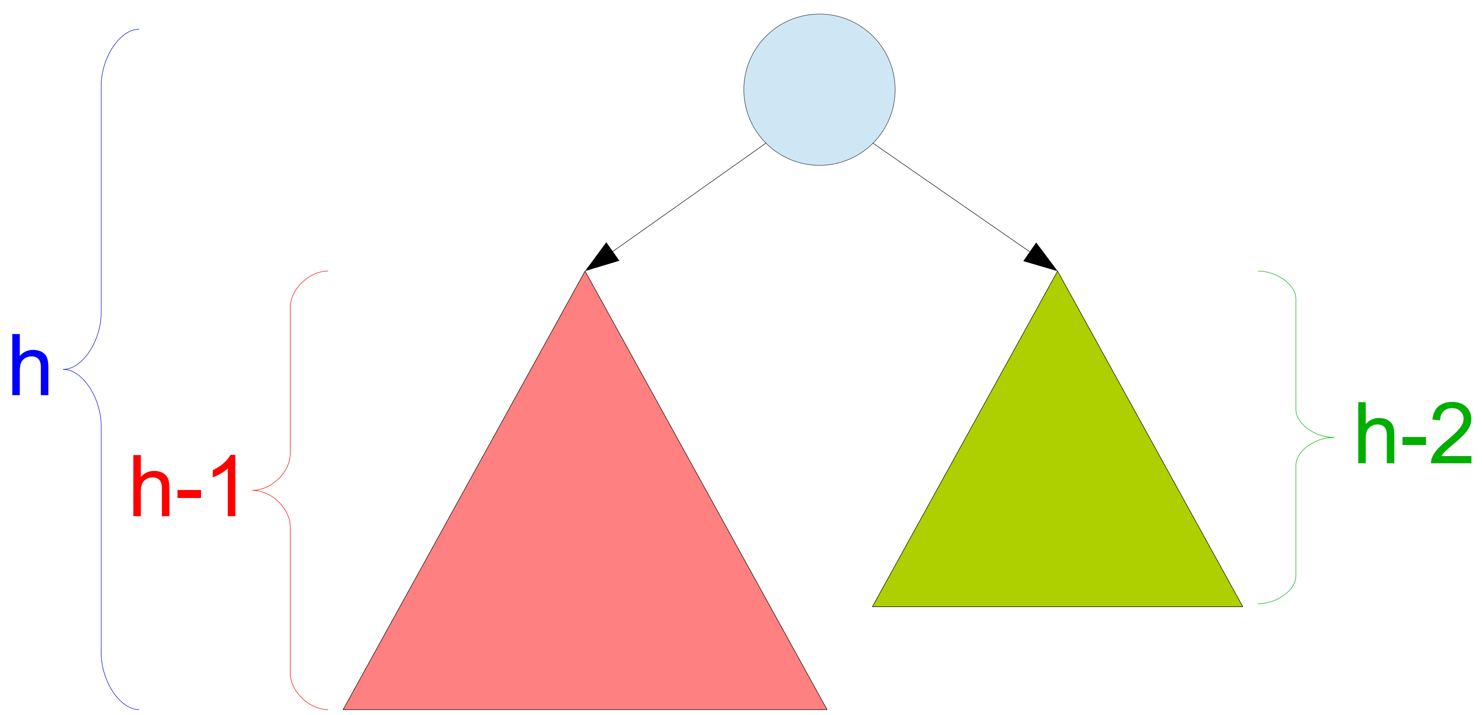
Árboles AVL

- Nombrados por las iniciales de sus inventores: G. M. Adelson-Velskii y E. M. Landis.
- Adelson-Velskii, G., Landis, E. M. (1962). "An algorithm for the organization of information". Proceedings of the USSR Academy of Sciences 146: 263–266.
- Garantiza que las operaciones de búsqueda, inserción y eliminación en un árbol binario ordenado toman en el peor caso O(log n).

Árboles AVL

- Propiedad a garantizar:

  Para cada nodo del árbol, las alturas de sus dos hijos (subárboles) difieren por mucho en 1.



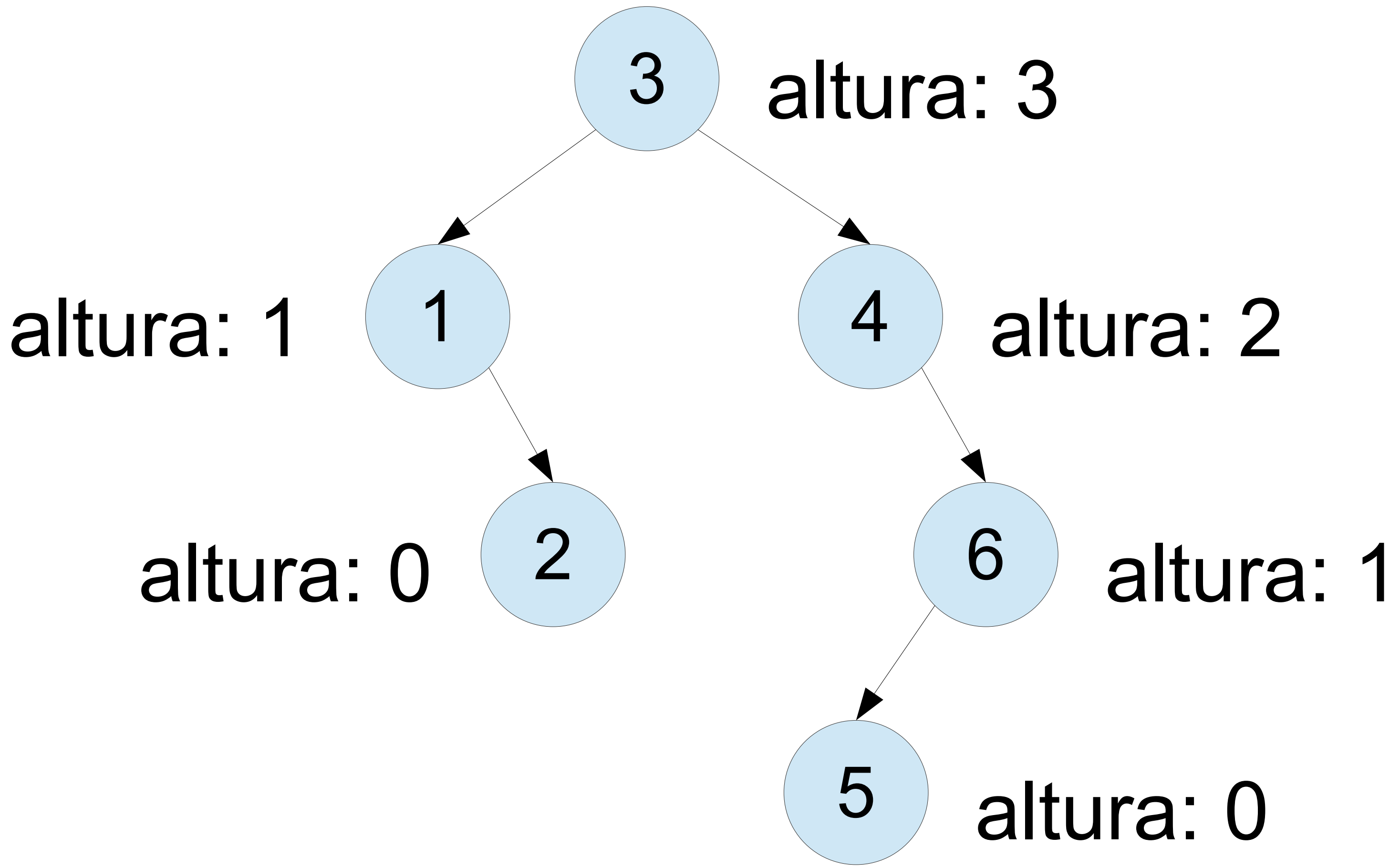
Árboles AVL

- Propiedad a garantizar:

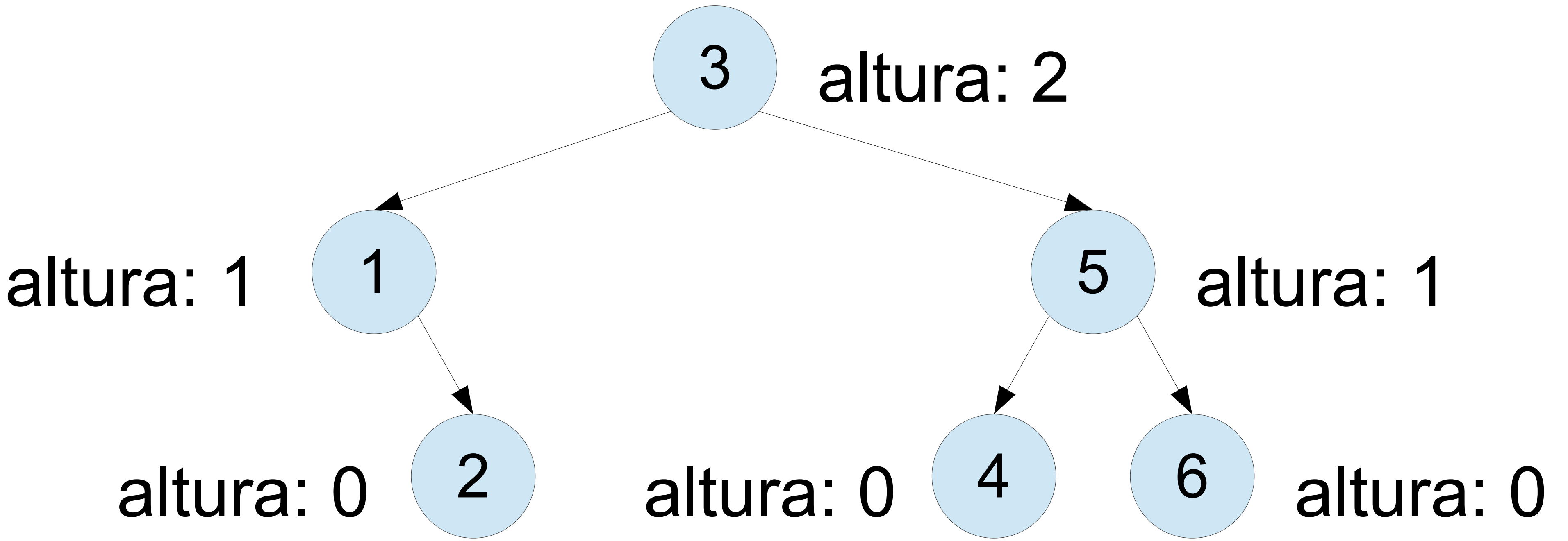
  Para cada nodo del árbol, las alturas de sus dos hijos (subárboles) difieren por mucho en 1.

  En las operaciones de inserción y eliminación, esta propiedad puede no cumplirse, por lo que se requiere aplicar operaciones para re-balancear o re-equilibrar el árbol.

Árboles AVL



Árboles AVL



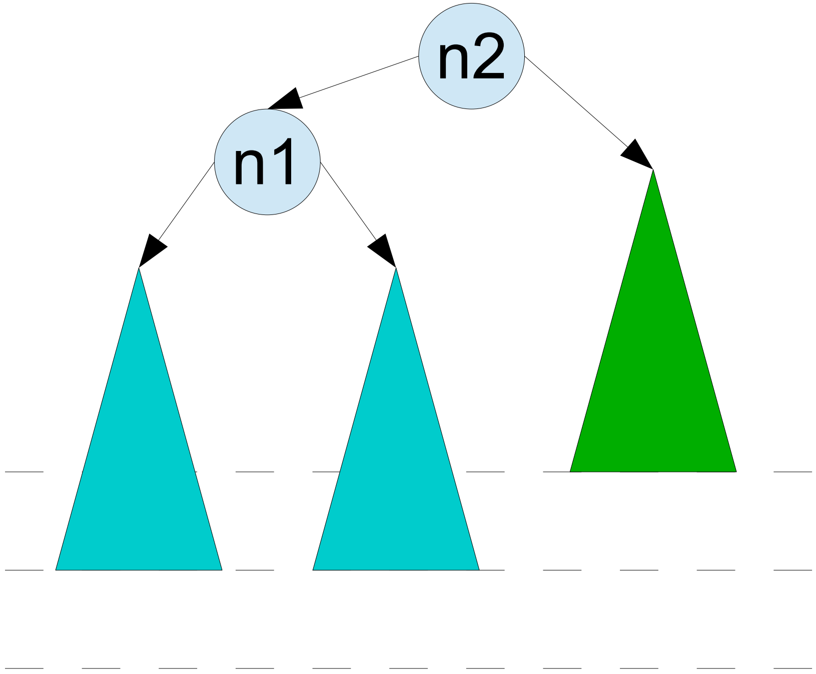
Árboles AVL

- Operaciones de re-balanceo → rotaciones.
  - Rotación a derecha.
  - Rotación a izquierda.
  - Doble rotación 1: rotación a izquierda seguida de rotación a derecha.
  - Doble rotación 2: rotación a derecha seguida de rotación a izquierda.

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

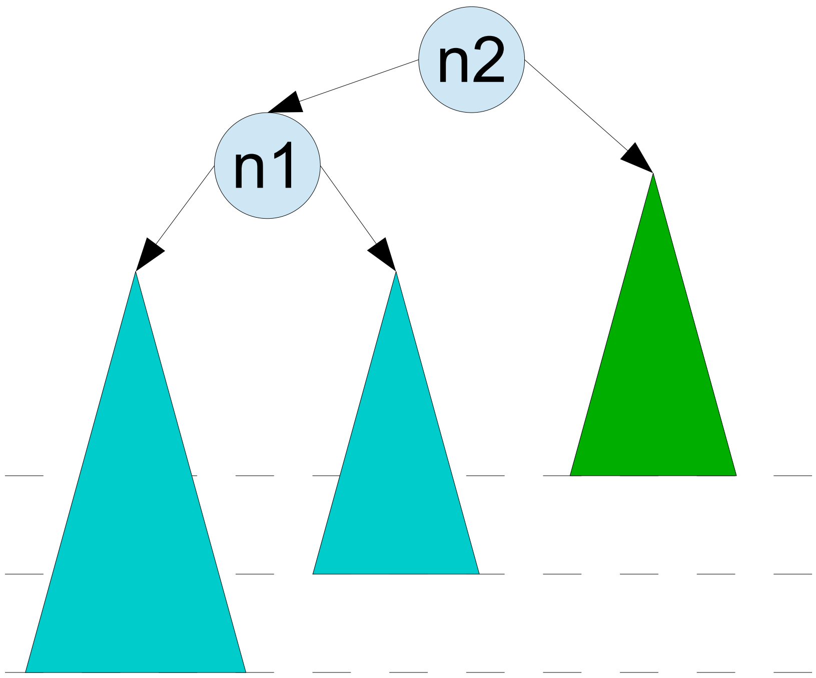
  Rotación a derecha.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

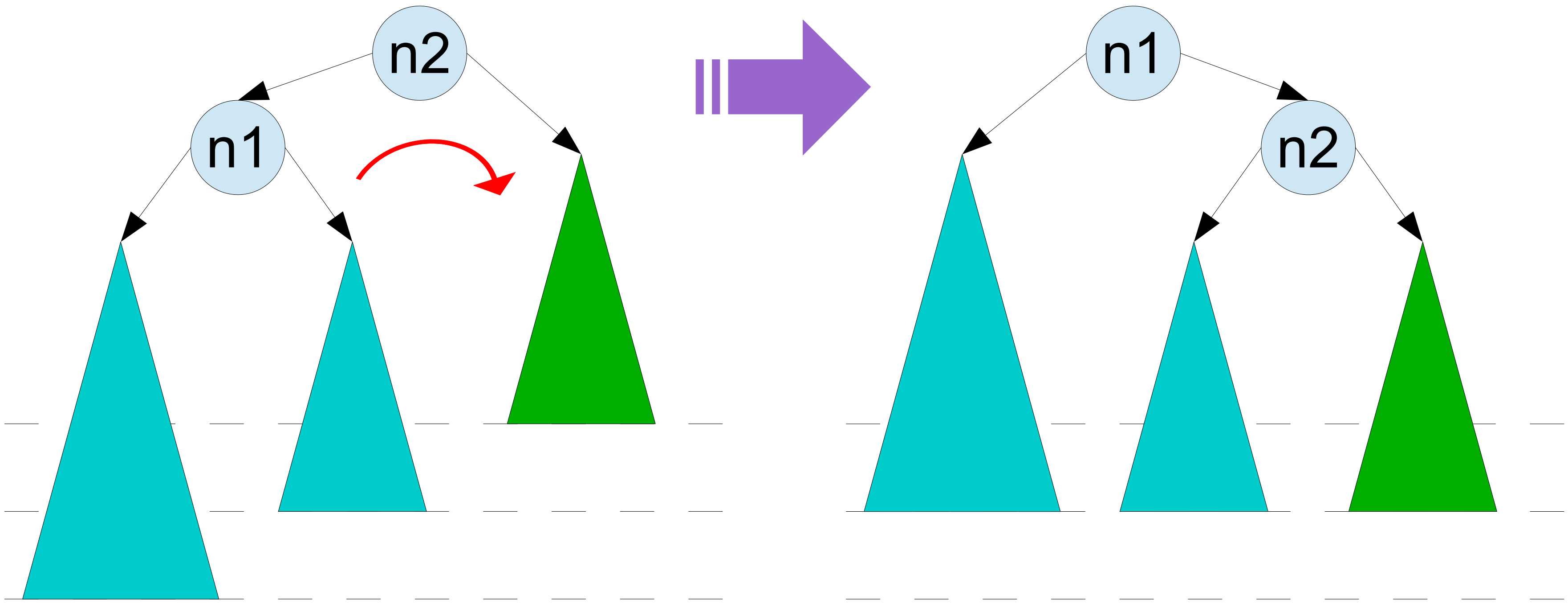
  Rotación a derecha.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

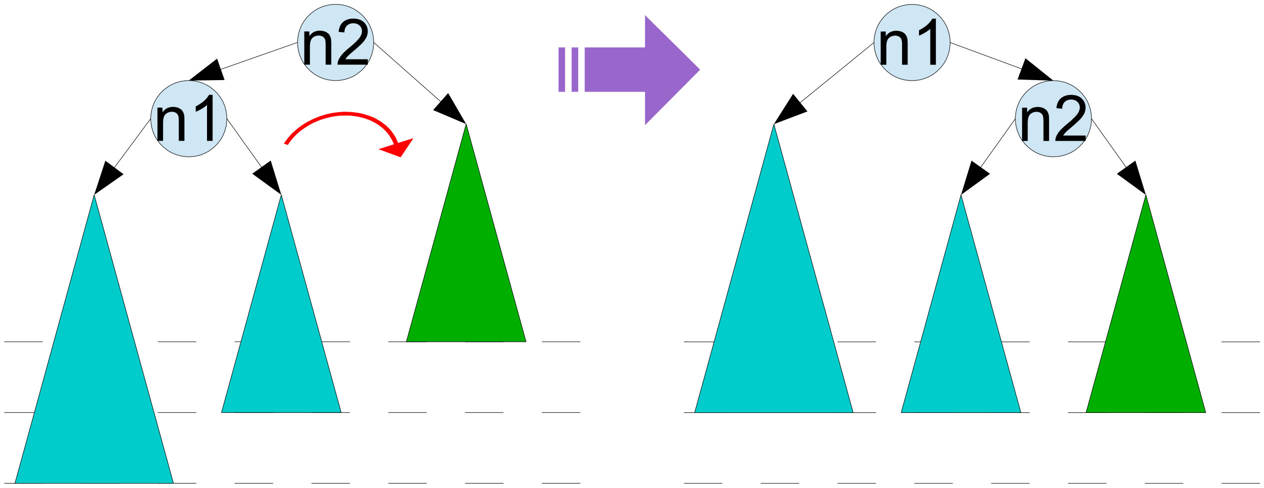
  Rotación a derecha.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Rotación a derecha.



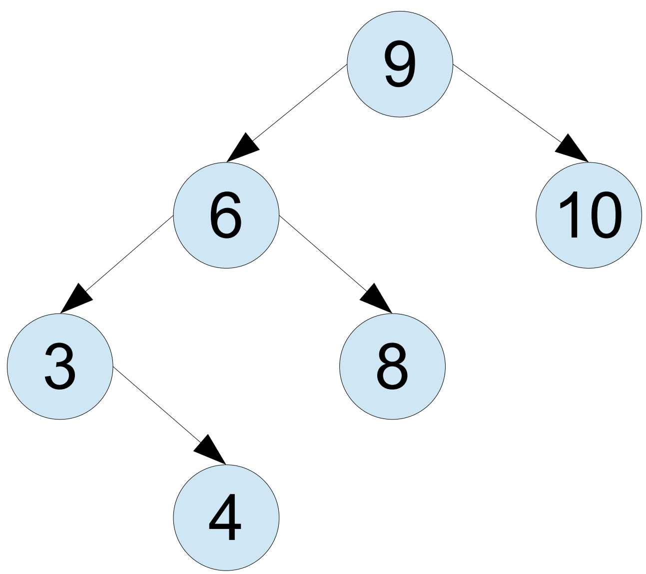
Rotación derecha sobre `n2`:

```
n_padre = n2->hijoIzq
n2->hijoIzq = n1->hijoDer
n1->hijoDer = n2
```

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Rotación a derecha.

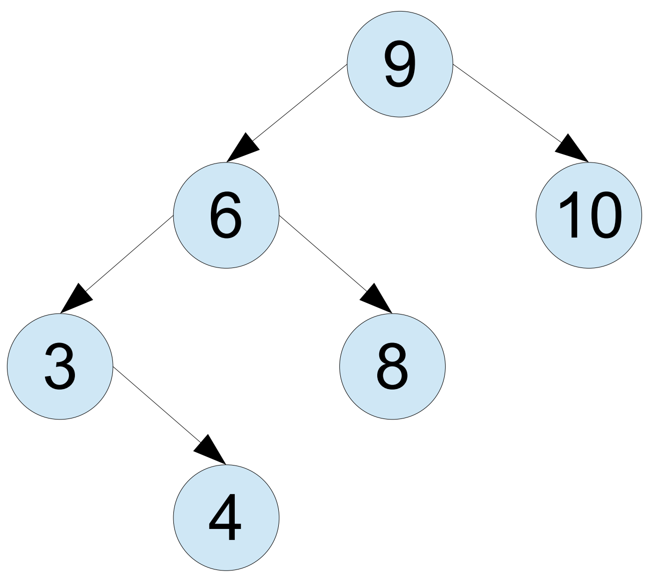


¿Es árbol AVL, está balanceado?

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Rotación a derecha.



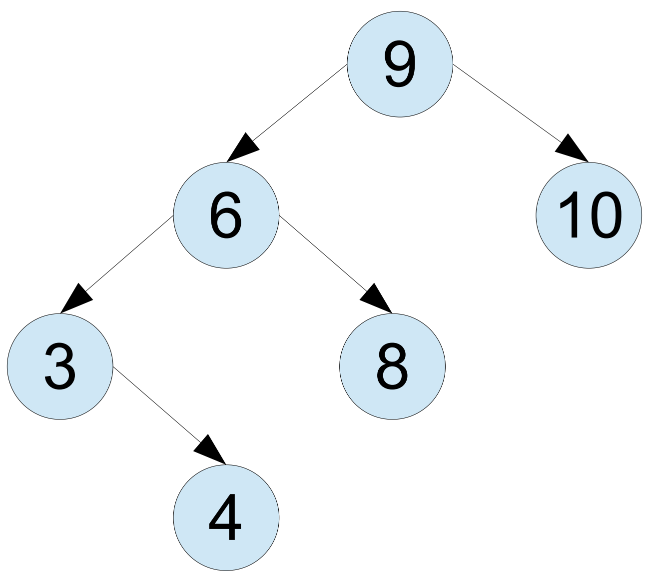
¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

¿En qué punto se incumple la propiedad?

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Rotación a derecha.



¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

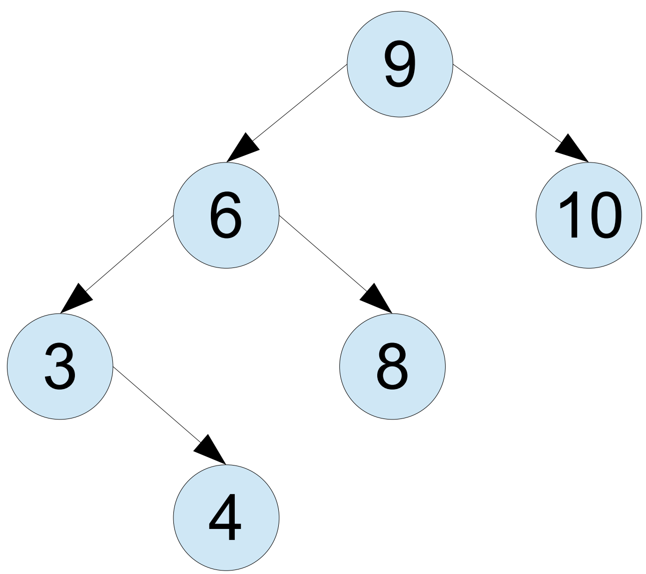
¿En qué punto se incumple la propiedad?
En 9, hijo izquierdo del izquierdo

¿Cómo se corrige?

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Rotación a derecha.



¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

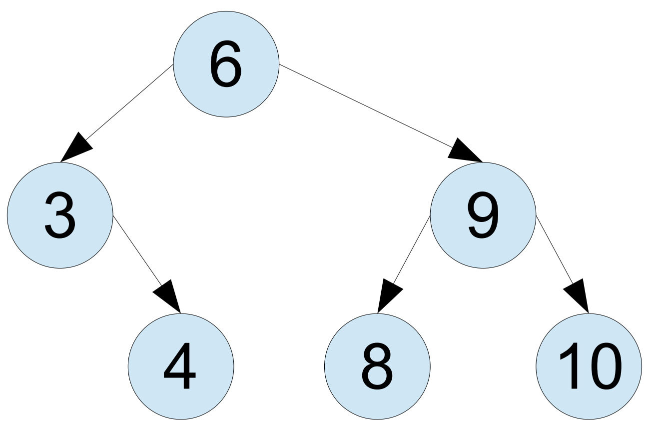
¿En qué punto se incumple la propiedad?
En 9, hijo izquierdo del izquierdo

¿Cómo se corrige?
Con rotación a derecha sobre 9

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Rotación a derecha.



¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

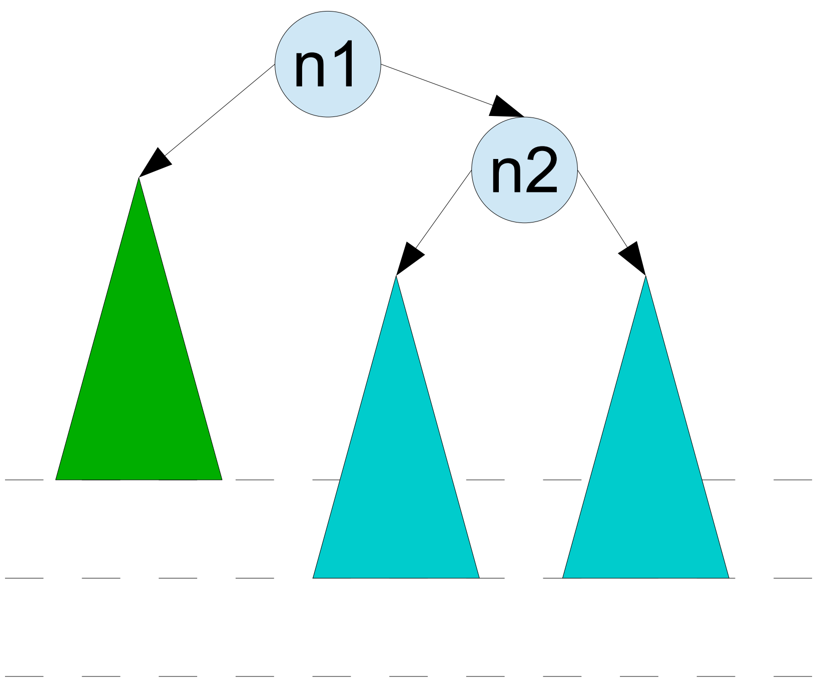
¿En qué punto se incumple la propiedad?
En 9, hijo izquierdo del izquierdo

¿Cómo se corrige?
Con rotación a derecha sobre 9

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

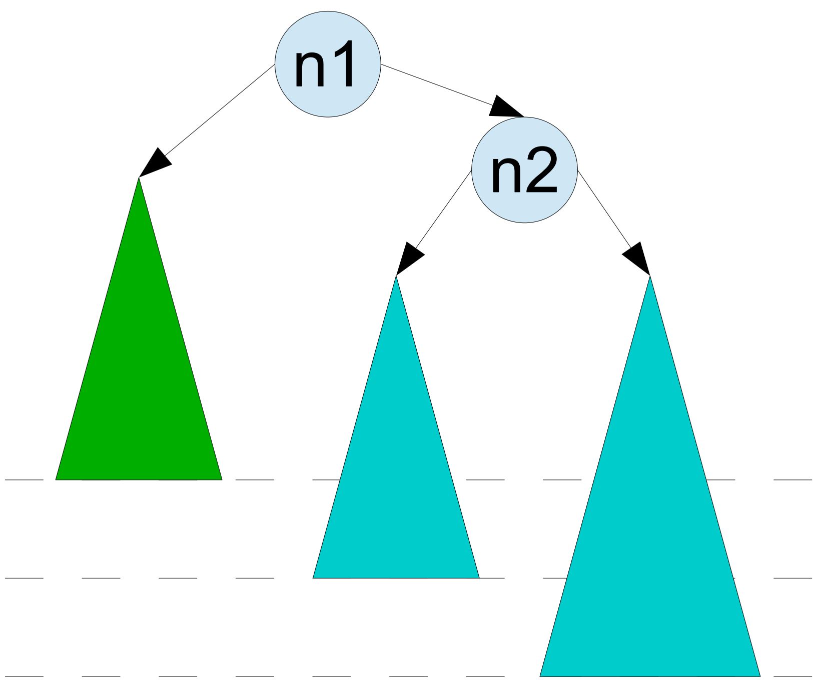
  Rotación a izquierda.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

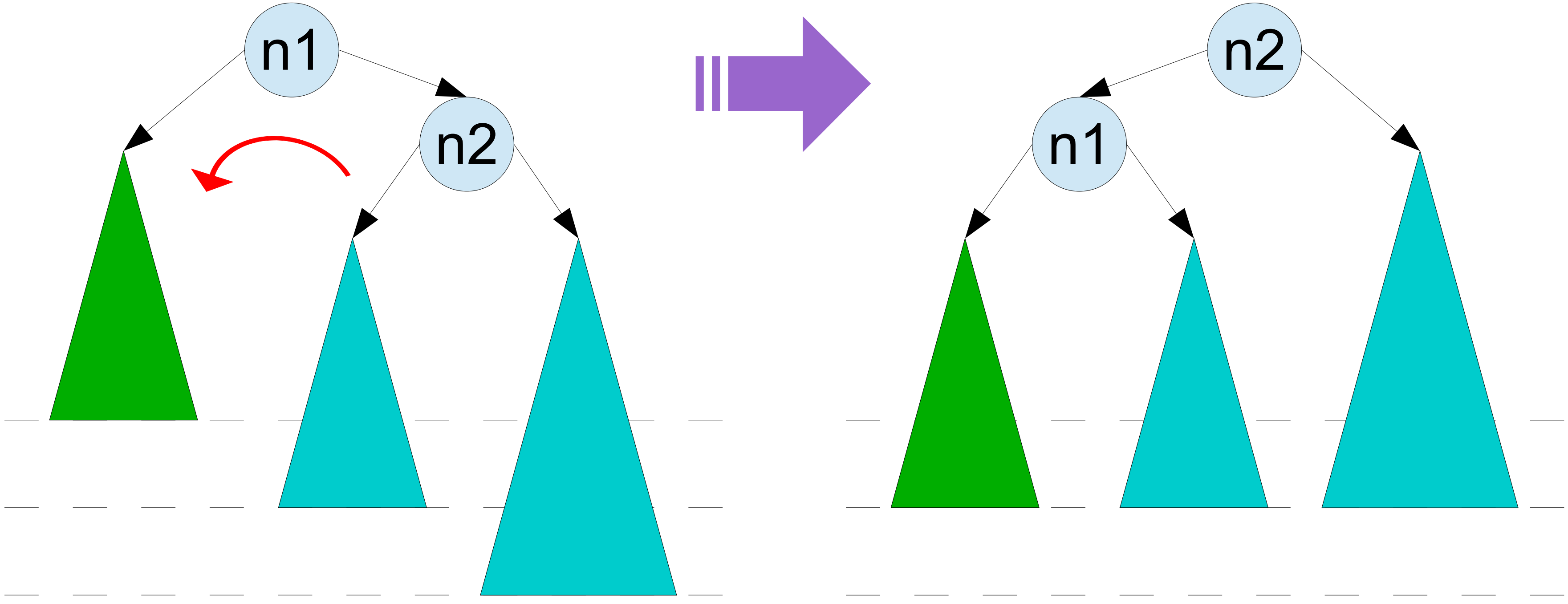
  Rotación a izquierda.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

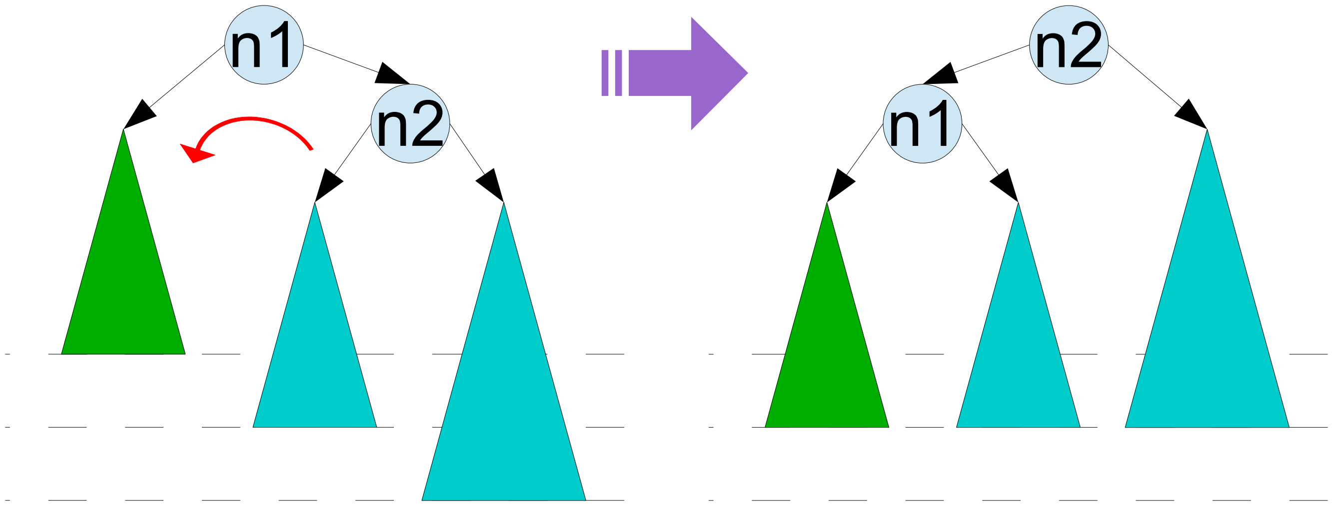
  Rotación a izquierda.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Rotación a izquierda.



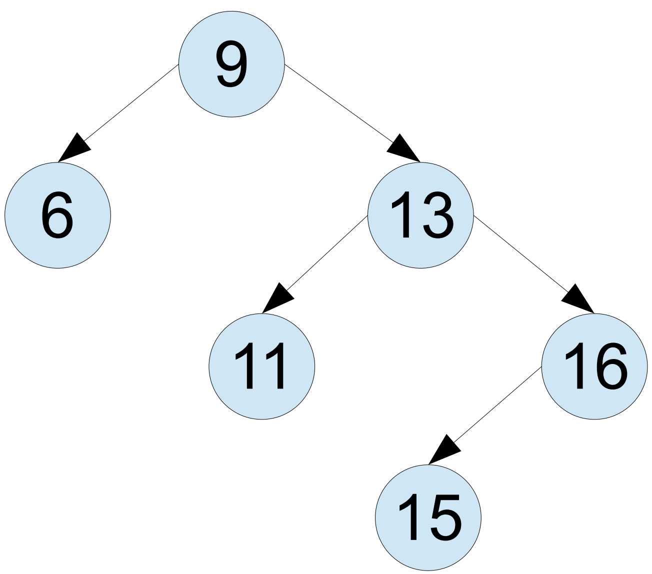
Rotación izquierda sobre `n1`:

```
n_padre = n1->hijoDer
n1->hijoDer = n2->hijoIzq
n2->hijoIzq = n1
```

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Rotación a izquierda.

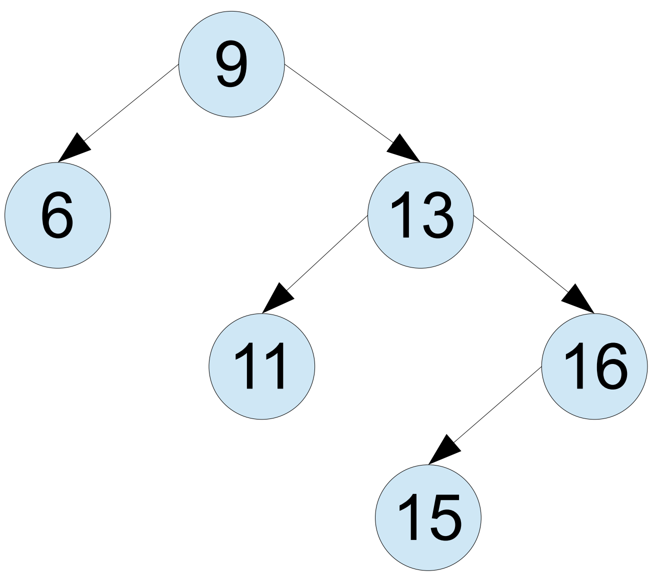


¿Es árbol AVL, está balanceado?

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Rotación a izquierda.



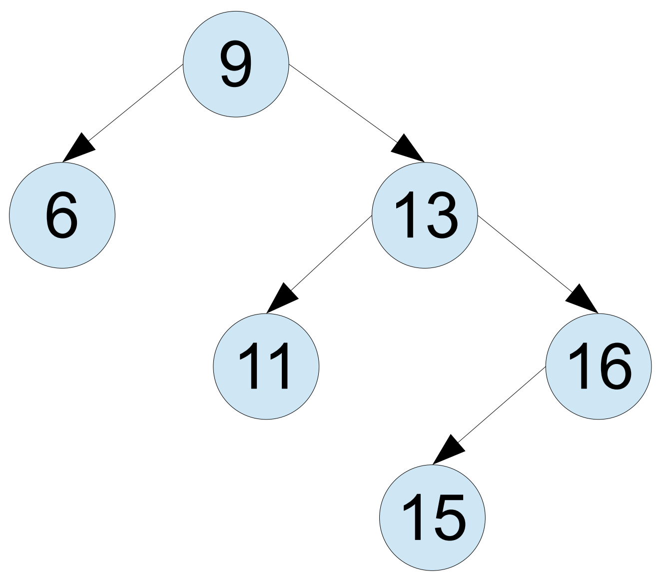
¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

¿En qué punto se incumple la propiedad?

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Rotación a izquierda.



¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

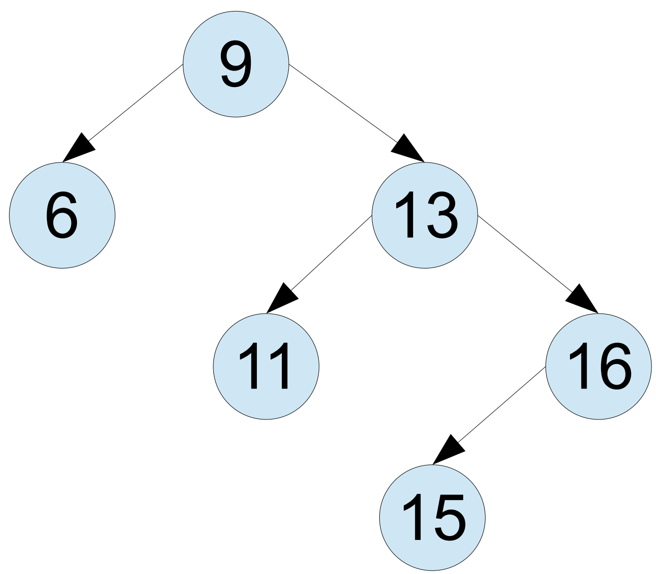
¿En qué punto se incumple la propiedad?
En 9, hijo derecho del derecho

¿Cómo se corrige?

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Rotación a izquierda.



¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

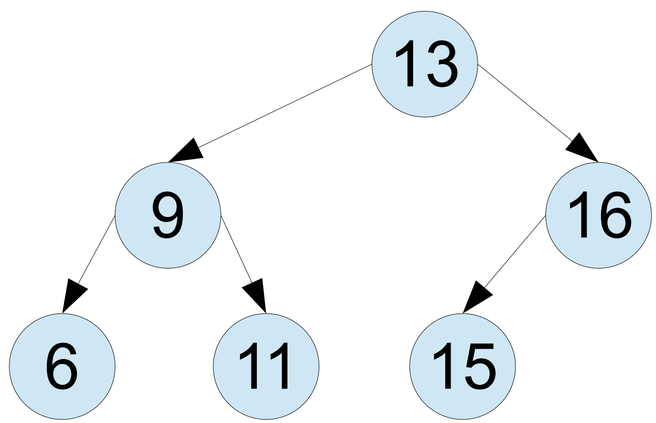
¿En qué punto se incumple la propiedad?
En 9, hijo derecho del derecho

¿Cómo se corrige?
Con rotación a izquierda sobre 9

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Rotación a izquierda.



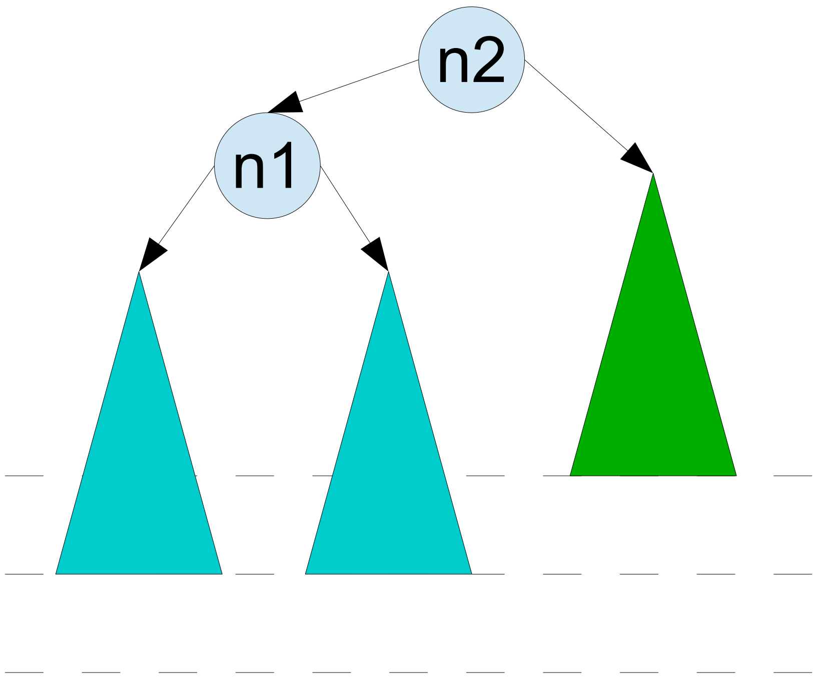
¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

¿En qué punto se incumple la propiedad?
En 9, hijo derecho del derecho

¿Cómo se corrige?
Con rotación a izquierda sobre 9

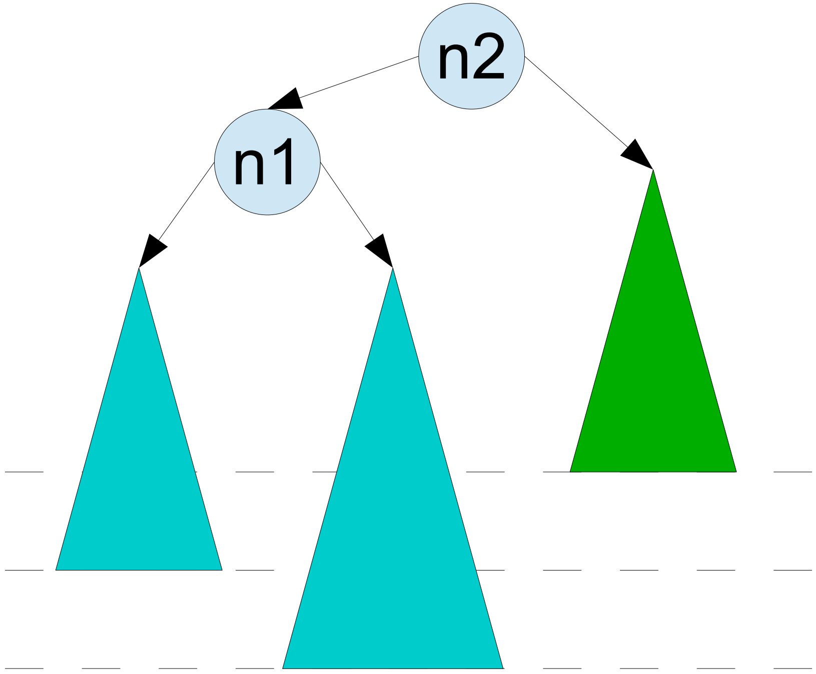
Árboles AVL

- Operaciones de re-balanceo → rotaciones.



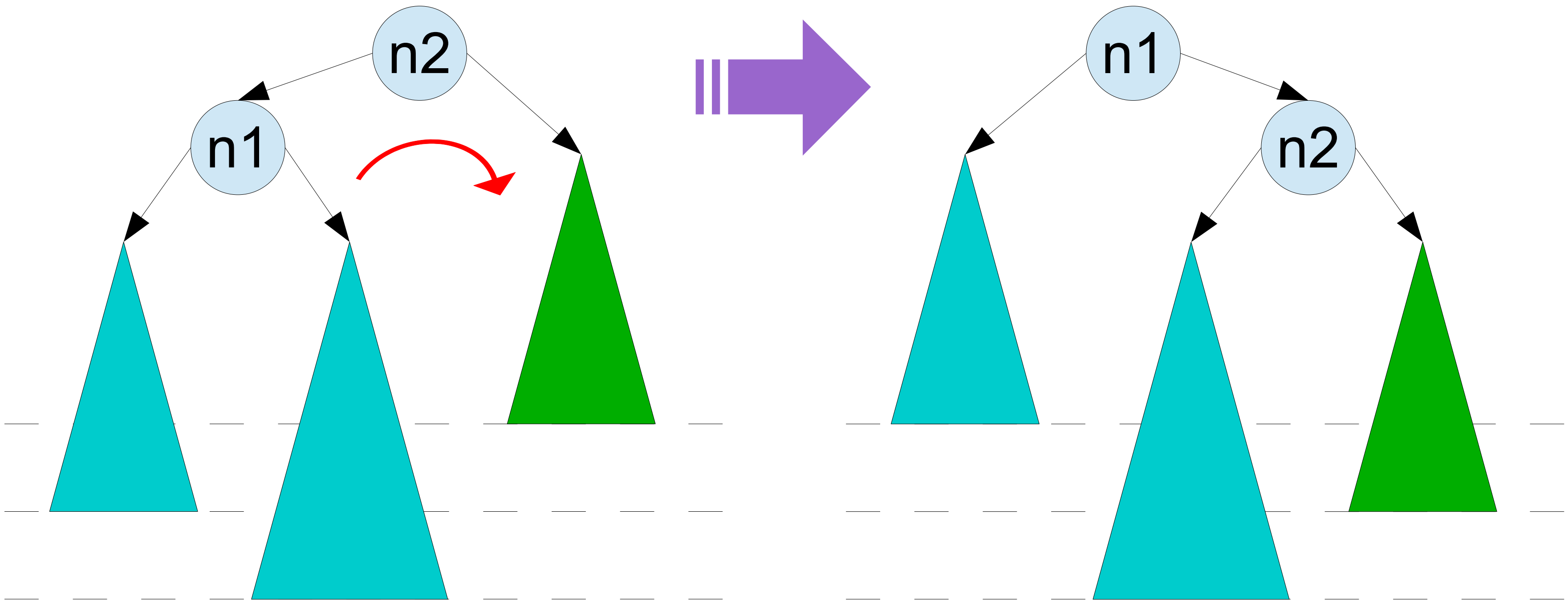
Árboles AVL

- Operaciones de re-balanceo → rotaciones.



Árboles AVL

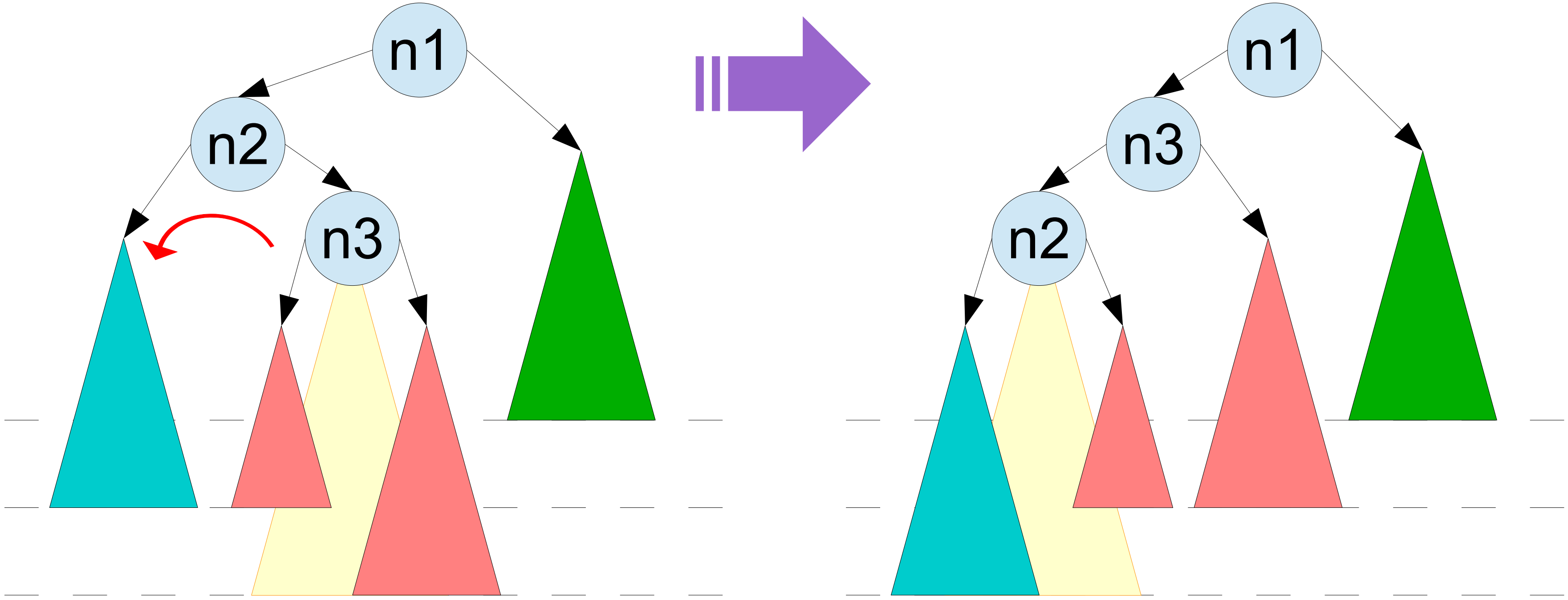
- Operaciones de re-balanceo → rotaciones.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

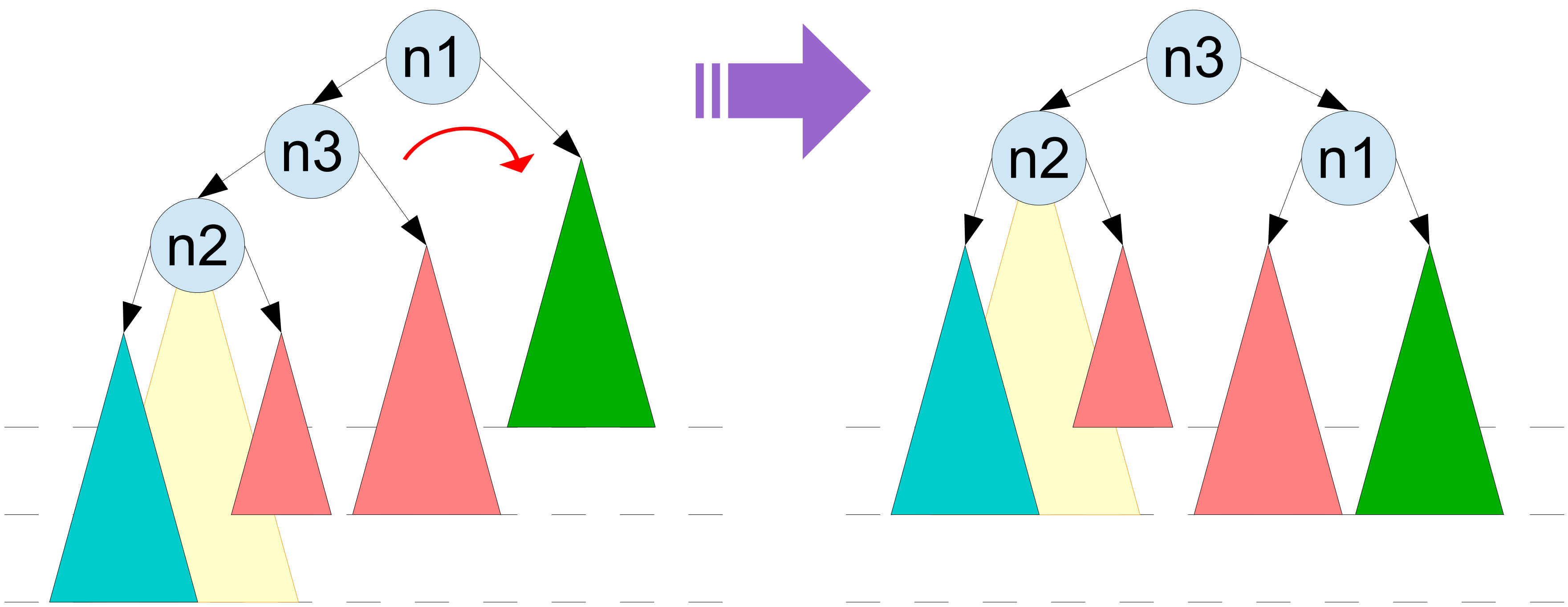
  Doble rotación 1: rotación a izquierda seguida de rotación a derecha.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

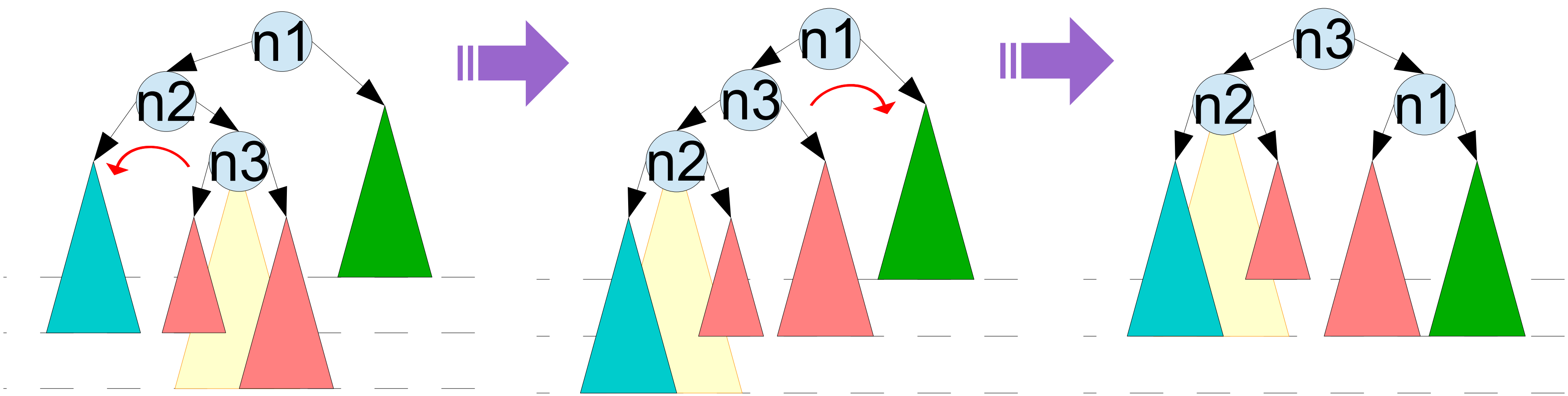
  Doble rotación 1: rotación a izquierda seguida de rotación a derecha.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Doble rotación 1: rotación a izquierda seguida de rotación a derecha.



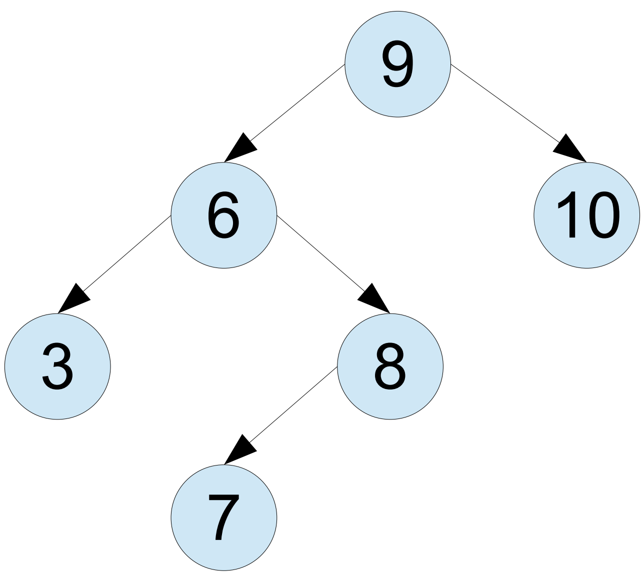
Rotación izquierda-derecha sobre `n1`:

```
aux = rotIzquierda(n1->hijoIzq)
n1->hijoIzq = aux
n_padre = rotDerecha(n1);
```

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Doble rotación 1: rotación a izquierda seguida de rotación a derecha.

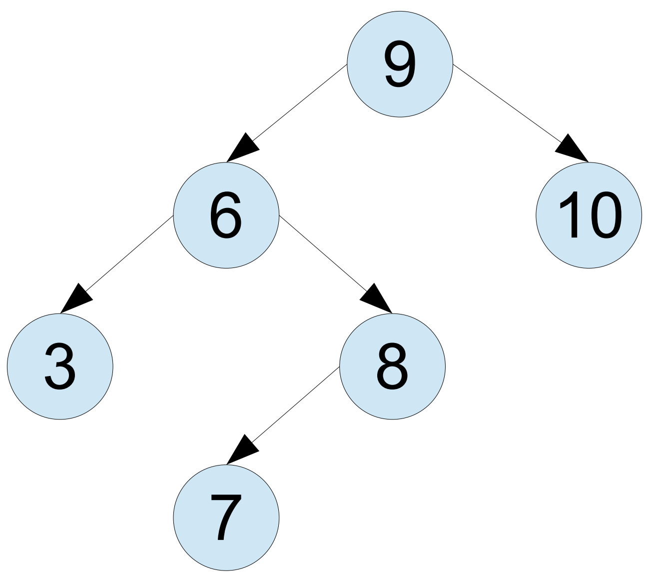


¿Es árbol AVL, está balanceado?

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Doble rotación 1: rotación a izquierda seguida de rotación a derecha.



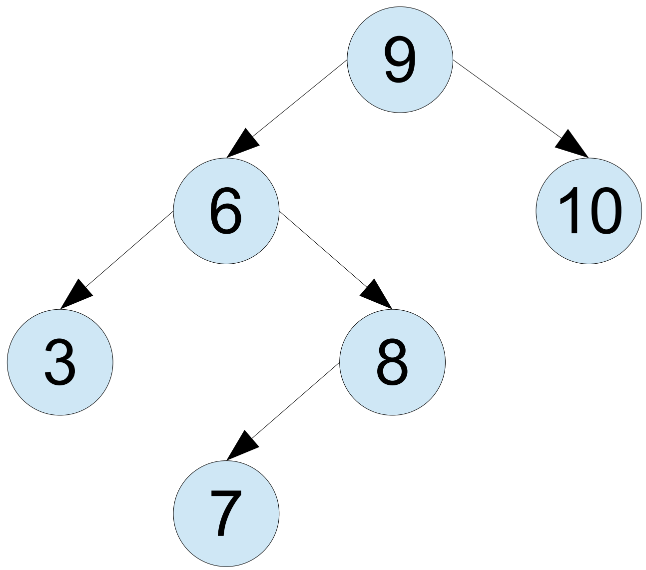
¿Es árbol AVL, está balanceado? No, incumple la propiedad AVL, no está balanceado

¿En qué punto se incumple la propiedad?

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Doble rotación 1: rotación a izquierda seguida de rotación a derecha.



¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

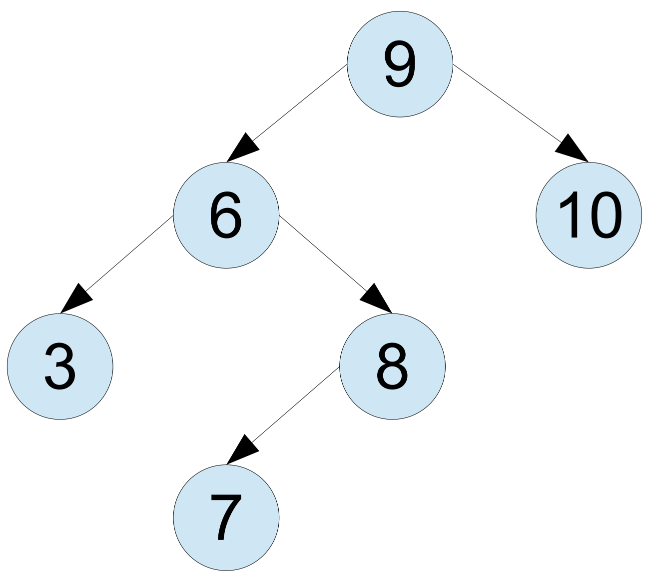
¿En qué punto se incumple la propiedad?
En 9, hijo derecho del izquierdo

¿Cómo se corrige?

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Doble rotación 1: rotación a izquierda seguida de rotación a derecha.



¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

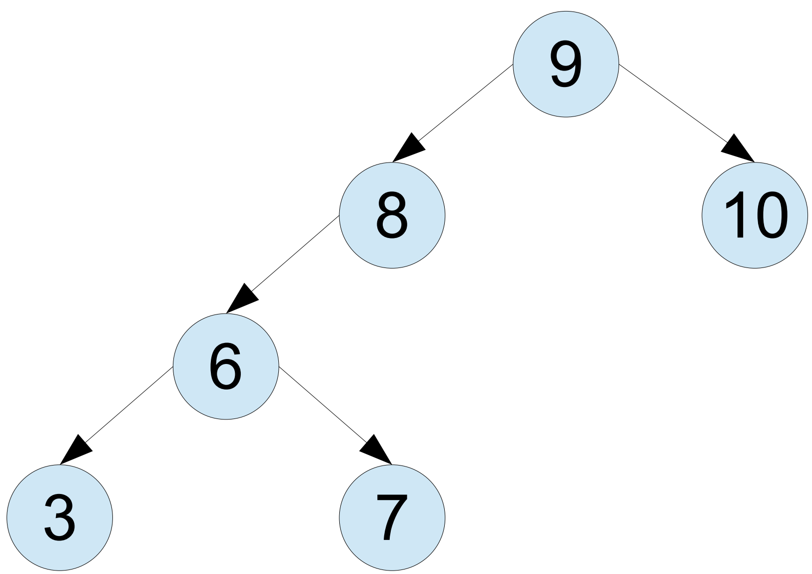
¿En qué punto se incumple la propiedad?
En 9, hijo derecho del izquierdo

¿Cómo se corrige?
Con doble rotación 1: izq - der

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Doble rotación 1: rotación a izquierda seguida de rotación a derecha.



¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

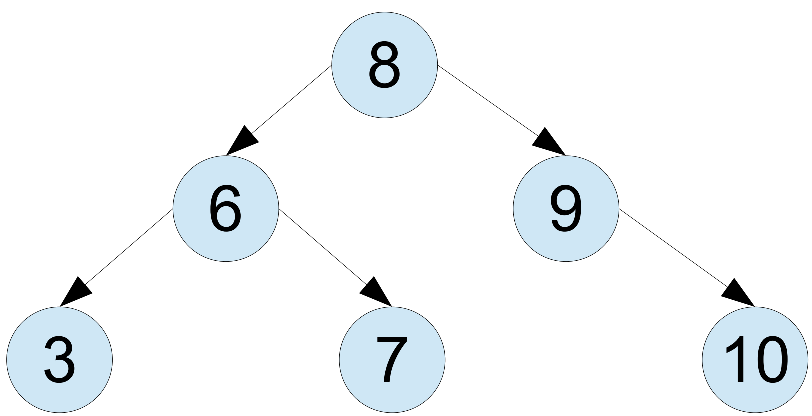
¿En qué punto se incumple la propiedad?
En 9, hijo derecho del izquierdo

¿Cómo se corrige?
Con doble rotación 1: izq - der

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Doble rotación 1: rotación a izquierda seguida de rotación a derecha.



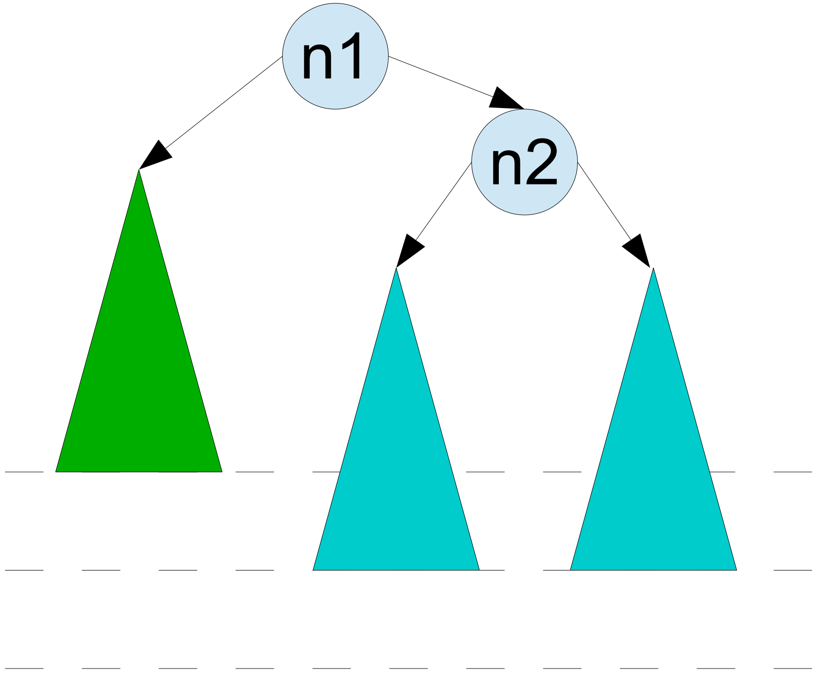
¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

¿En qué punto se incumple la propiedad?
En 9, hijo derecho del izquierdo

¿Cómo se corrige?
Con doble rotación 1: izq - der

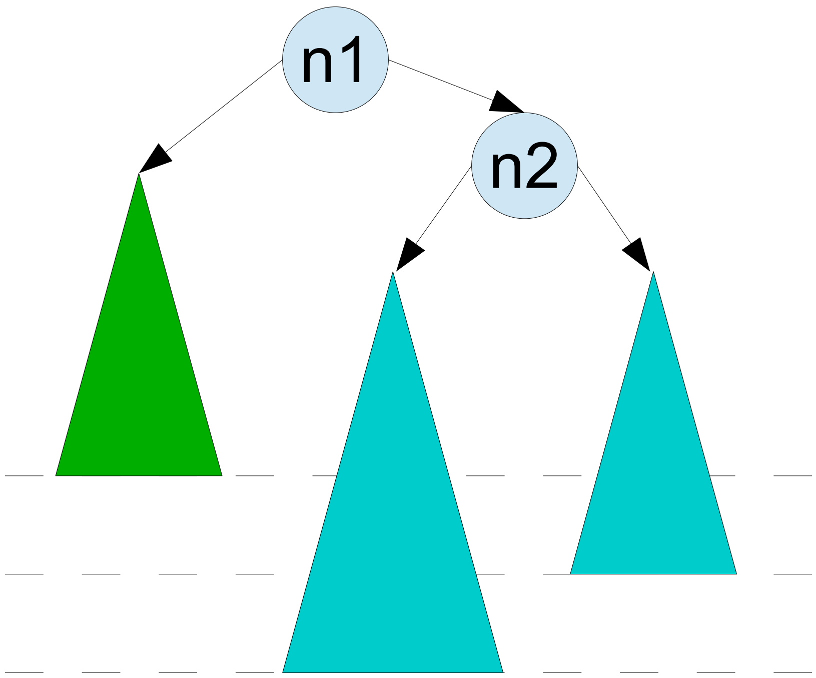
Árboles AVL

- Operaciones de re-balanceo → rotaciones.



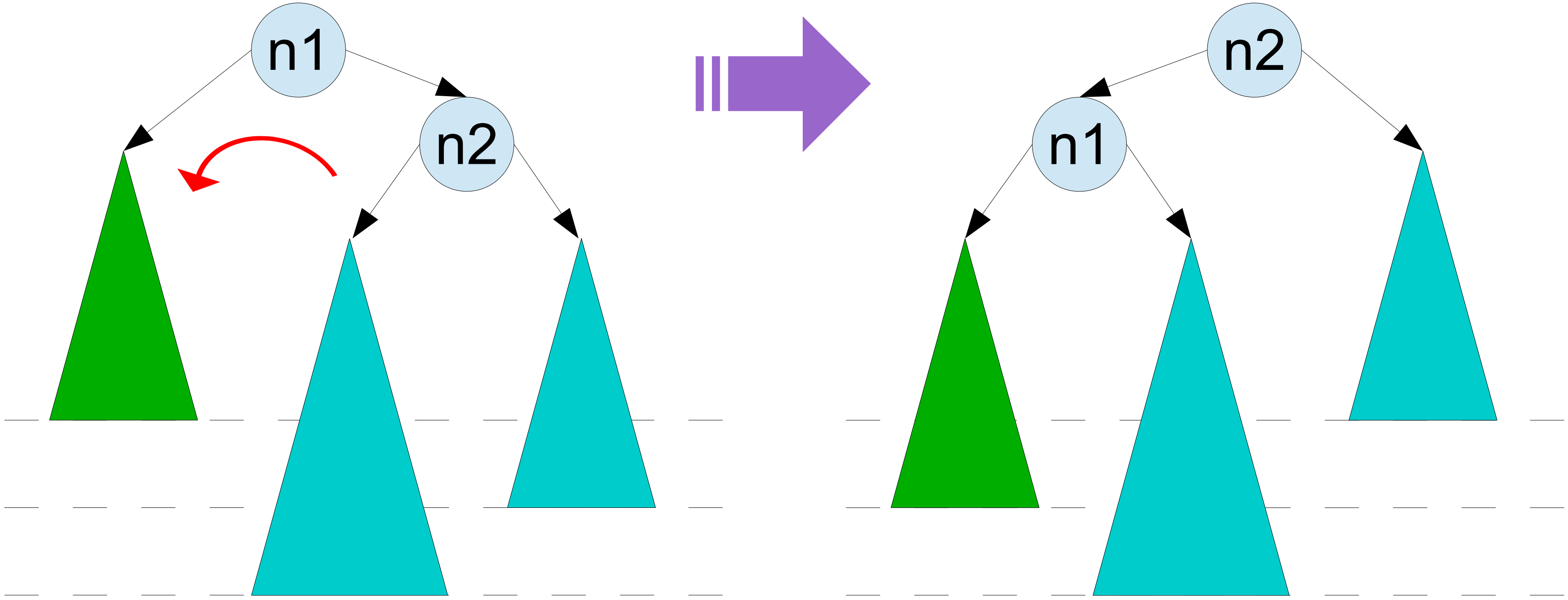
Árboles AVL

- Operaciones de re-balanceo → rotaciones.



Árboles AVL

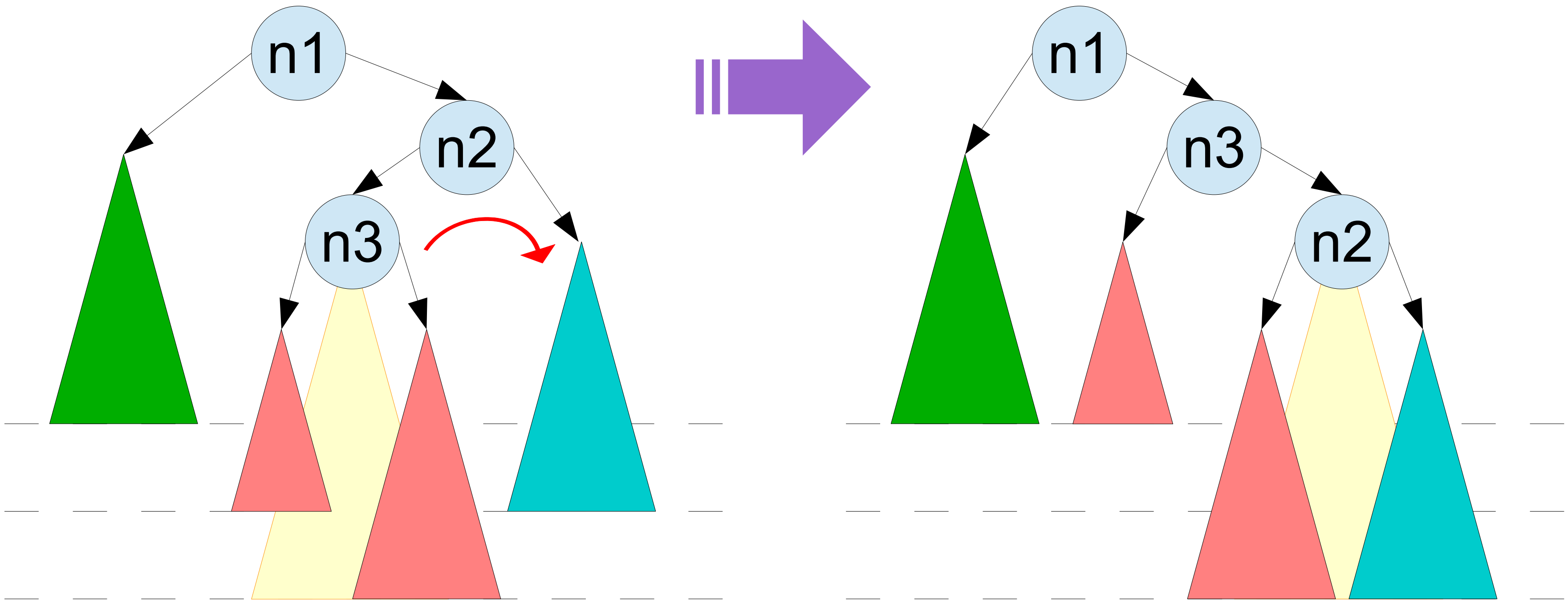
- Operaciones de re-balanceo → rotaciones.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

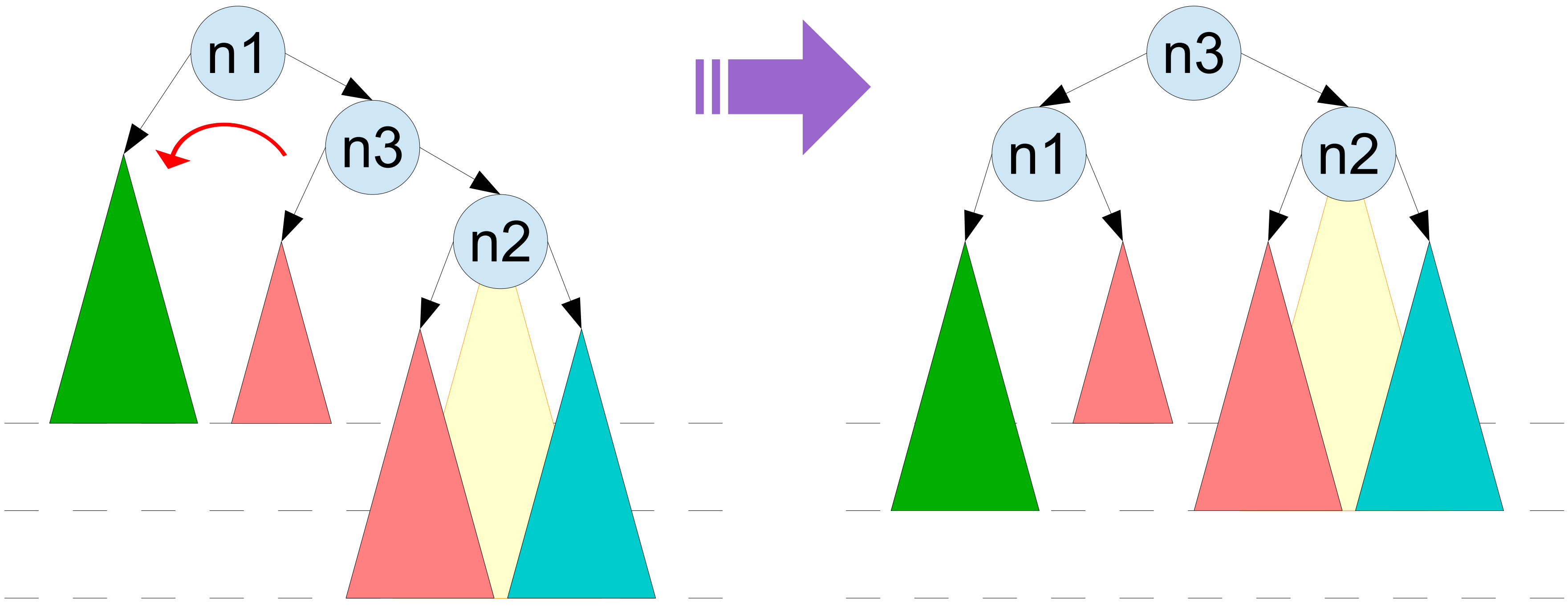
  Doble rotación 2: rotación a derecha seguida de rotación a izquierda.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

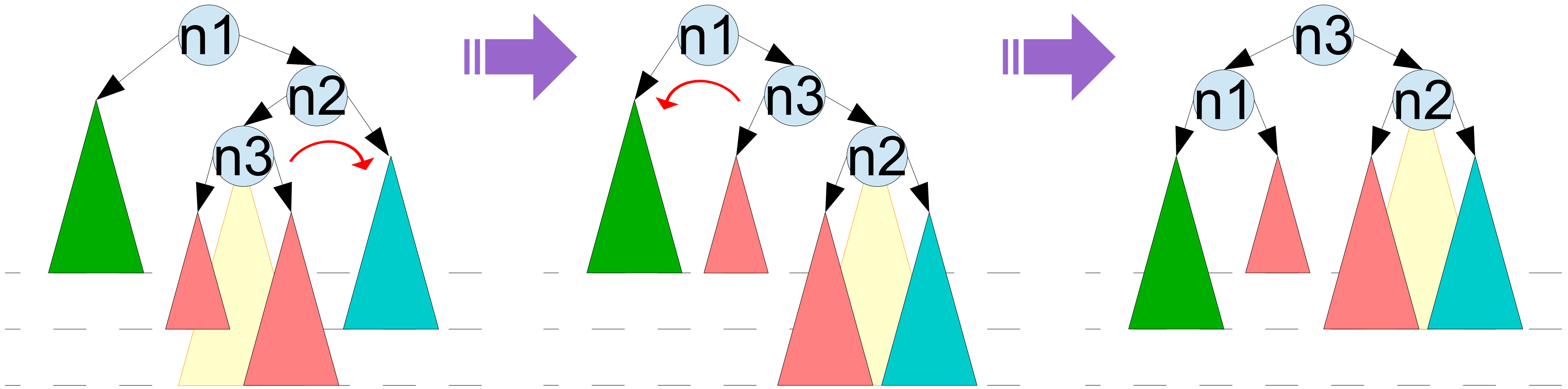
  Doble rotación 2: rotación a derecha seguida de rotación a izquierda.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Doble rotación 2: rotación a derecha seguida de rotación a izquierda.



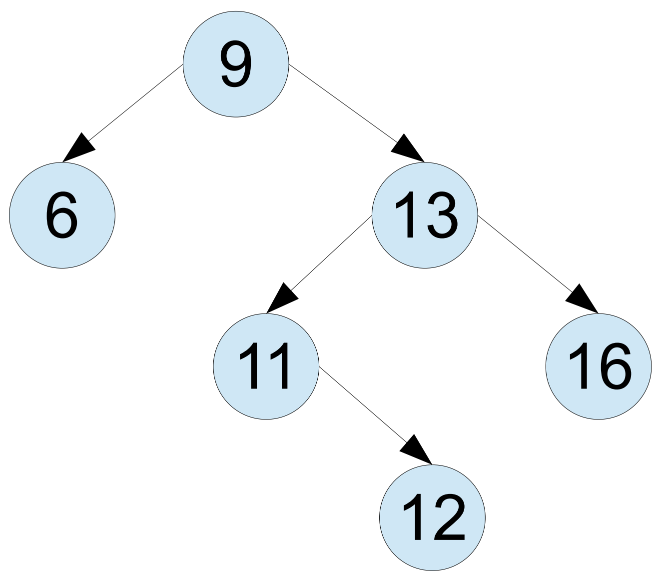
Rotación derecha-izquierda sobre `n1`:

```
aux = rotDerecha(n1->hijoDer)
n1->hijoDer = aux
n_padre = rotIzquierda(n1)
```

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Doble rotación 2: rotación a derecha seguida de rotación a izquierda.

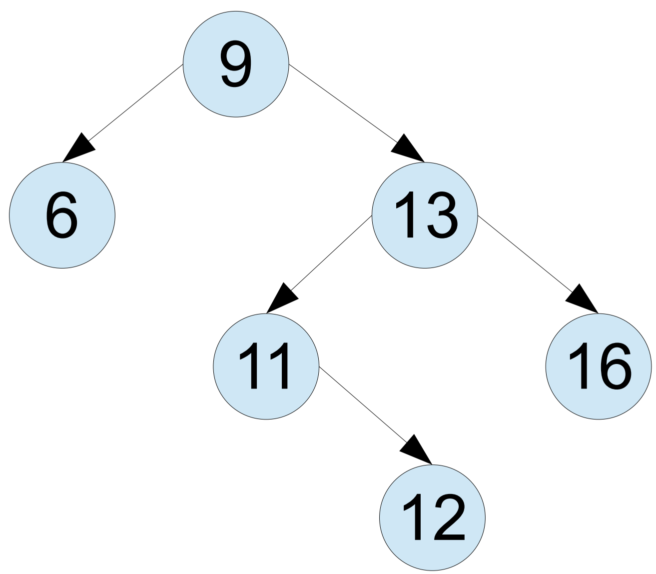


¿Es árbol AVL, está balanceado?

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Doble rotación 2: rotación a derecha seguida de rotación a izquierda.



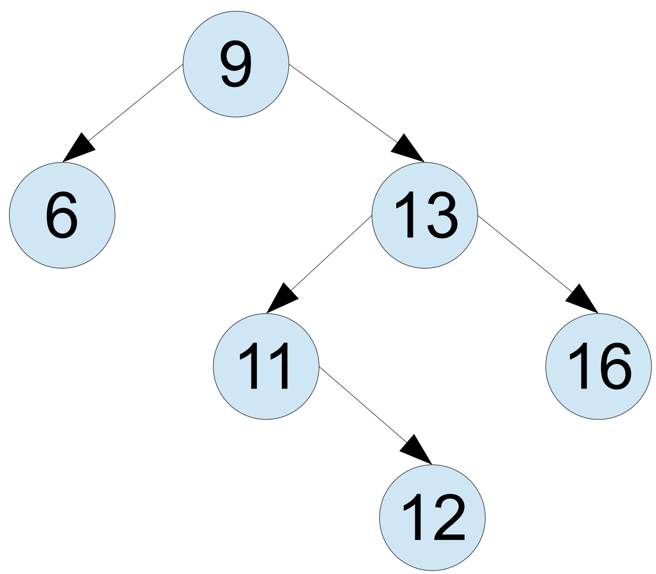
¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

¿En qué punto se incumple la propiedad?

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Doble rotación 2: rotación a derecha seguida de rotación a izquierda.



¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

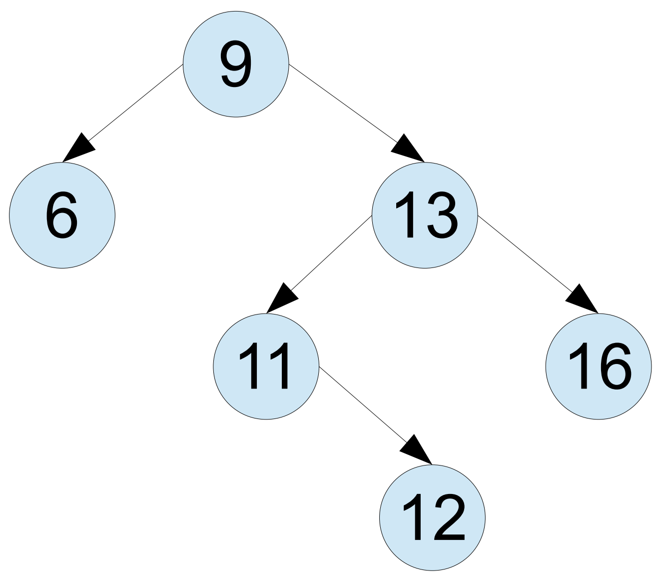
¿En qué punto se incumple la propiedad?
En 9, hijo izquierdo del derecho

¿Cómo se corrige?

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Doble rotación 2: rotación a derecha seguida de rotación a izquierda.



¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

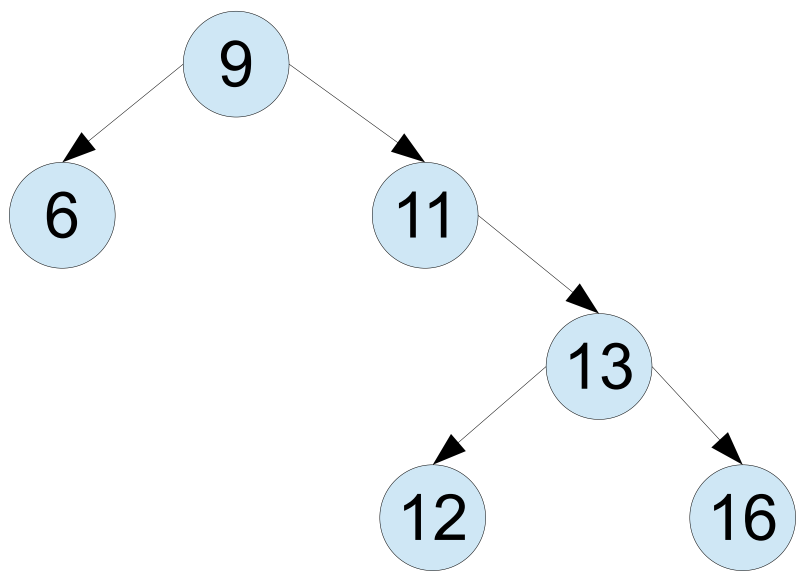
¿En qué punto se incumple la propiedad?
En 9, hijo izquierdo del derecho

¿Cómo se corrige?
Con doble rotación 2: der - izq

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Doble rotación 2: rotación a derecha seguida de rotación a izquierda.



¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

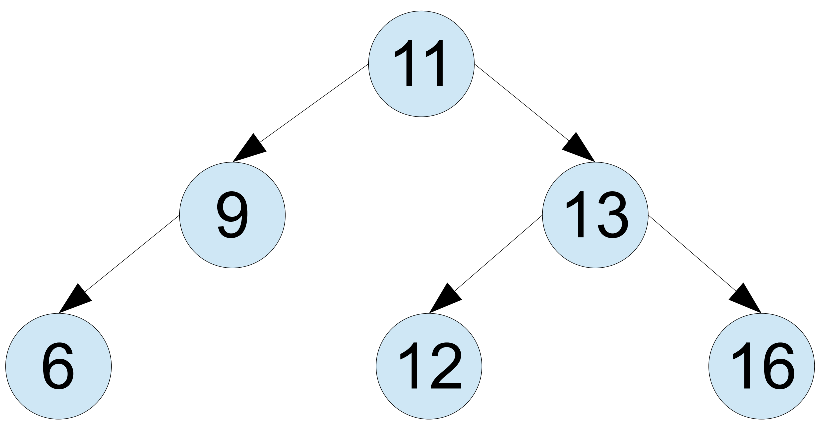
¿En qué punto se incumple la propiedad?
En 9, hijo izquierdo del derecho

¿Cómo se corrige?
Con doble rotación 2: der - izq

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Doble rotación 2: rotación a derecha seguida de rotación a izquierda.



¿Es árbol AVL, está balanceado?
No, incumple la propiedad AVL, no está balanceado

¿En qué punto se incumple la propiedad?
En 9, hijo izquierdo del derecho

¿Cómo se corrige?
Con doble rotación 2: der - izq

Árboles AVL

- Inserción y eliminación:

  Una vez realizada la operación, es necesario verificar desde el punto de cambio hacia arriba la condición:

  Para cada nodo del árbol, las alturas de sus dos hijos (subárboles) difieren por mucho en 1.

  De forma que cuando se encuentre que esta condición no se cumple, se debe realizar la rotación necesaria <u>en ese punto</u>.

Árboles AVL

- Balanceo → cuatro posibilidades que requieren rotación

  Diferencia entre alturas de hijos (izq - der)

  - Si diferencia == 2 (izquierdo más alto que derecho)

    Diferencia entre alturas de hijos (izq - der) del hijo izquierdo

    - Si diferencia > 0 → rotación a derecha

    - Si diferencia < 0 → rotación izquierda-derecha

Árboles AVL

- Balanceo → cuatro posibilidades que requieren rotación

  Diferencia entre alturas de hijos (izq - der)

  - Si diferencia == -2 (derecho más alto que izquierdo)

    Diferencia entre alturas de hijos (izq - der) del hijo derecho

    - Si diferencia < 0 → rotación a izquierda

    - Si diferencia > 0 → rotación derecha-izquierda

Árboles AVL

Implementación TAD ArbolAVL

- Todas las operaciones son iguales al TAD Arbol Binario Ordenado, salvo la inserción y eliminación.
- Se agrega una operación de balanceo, que verifica si en un nodo se cumple o no la propiedad, y si no se cumple hace el llamado a las rotaciones.
- Se agregan las rotaciones.
- En la inserción y eliminación se hace el llamado a balanceo en toda la ruta de modificación.

Árboles AVL

- Applet de demostración

  https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

¿Cómo garantizar árboles "bonitos"?

- Balanceando:
  - Evitar "listas".
  - Evitar ramas cortas.
- Garantizar búsqueda / inserción / eliminación en O(log n).
- Árboles AVL y RN.

Árboles RN
(Rojo-Negro)
(*Red-Black Trees*)

Árboles RN

- Árboles Rojo-Negro (*Red-Black trees*):

  Inicialmente conocido como Árbol-B binario simétrico.

  Guibas, Leo J., and Robert Sedgewick. "A dichromatic framework for balanced trees." 19th Annual Symposium on Foundations of Computer Science. IEEE, 1978.

Árboles RN

- Árboles Rojo-Negro (*Red-Black trees*):

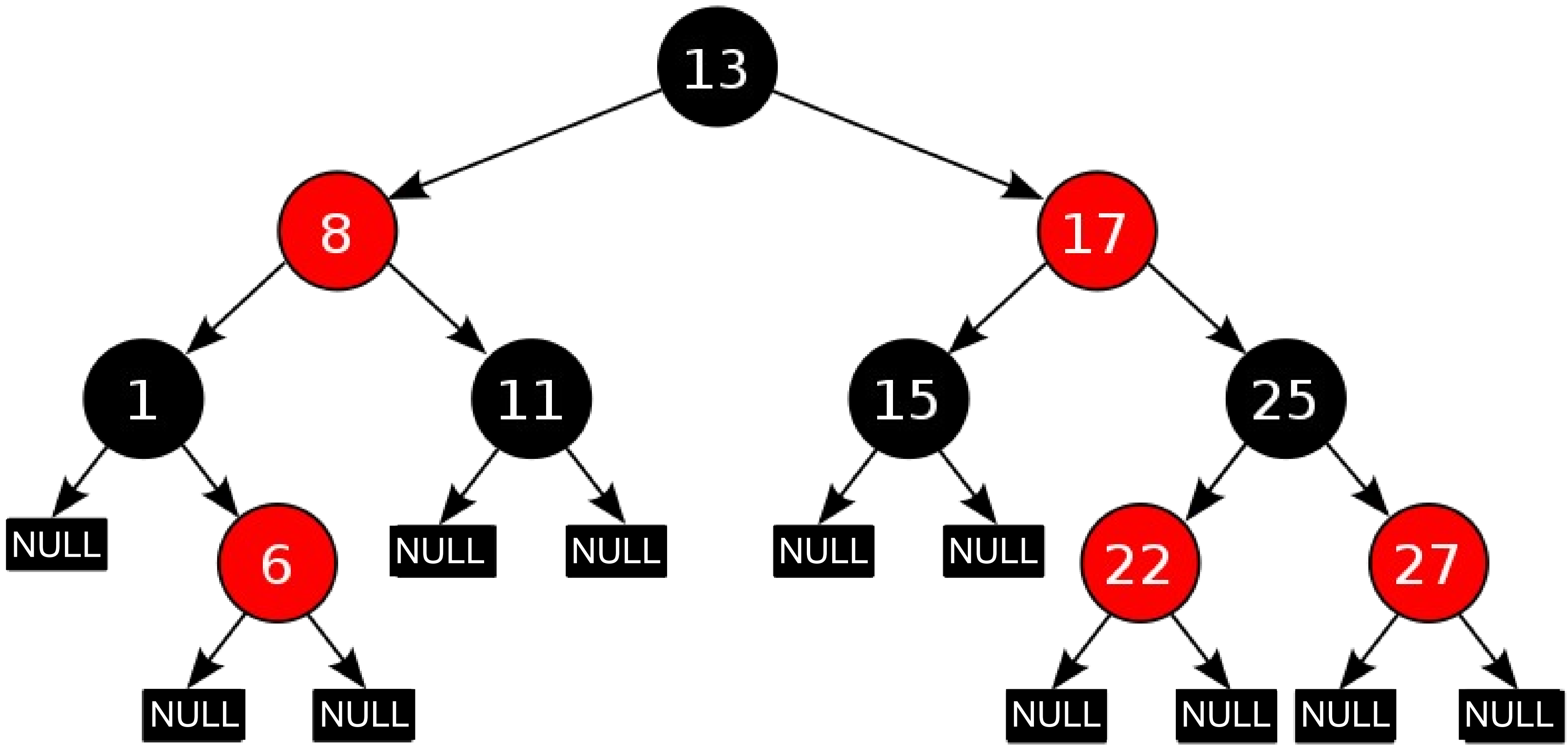
  Como los árboles AVL, garantiza que las operaciones de búsqueda, inserción y eliminación en un árbol binario ordenado toman en el peor caso O(log n).

  Requiere que el nodo del árbol incluya una propiedad adicional de <u>color</u> (**rojo** o **negro**).

Árboles RN

- Propiedades a garantizar:
  1. Cada nodo es **rojo** o **negro**.
  2. La raíz y las hojas (nulos) son siempre **negras**.
  3. Si un nodo es **rojo**, entonces su padre es **negro**.
  4. Cada nodo **rojo** debe tener dos hijos **negros**.
  5. Todas las rutas desde un nodo x hacia un descendiente hoja tienen el mismo número de nodos **negros** (altura-negra(x)).

Árboles RN


13
8
17
1
11
15
25
NULL
6
NULL
NULL
NULL
NULL
22
27
NULL
NULL
NULL
NULL
NULL
NULL
NULL
NULL

Árboles RN

- Inserción:

  Se realiza como en un árbol binario ordenado.

  ¿Cuál debe ser el color del nuevo nodo?

  - **Negro**

Árboles RN

- Inserción:

  Se realiza como en un árbol binario ordenado.

  ¿Cuál debe ser el color del nuevo nodo?

  - **Negro**, causa una diferencia en el número de nodos negros en una ruta, incumpliendo la propiedad 5, y es una situación más difícil de corregir.

Árboles RN

- Inserción:

  Se realiza como en un árbol binario ordenado.

  ¿Cuál debe ser el color del nuevo nodo?

  - Entonces siempre lo pintaremos **rojo**.

Árboles RN

- Inserción:

  Se realiza como en un árbol binario ordenado.

  ¿Cuál debe ser el color del nuevo nodo?

  – Entonces siempre lo pintaremos **rojo**.

  ¿Y si el padre del nodo es **rojo**?

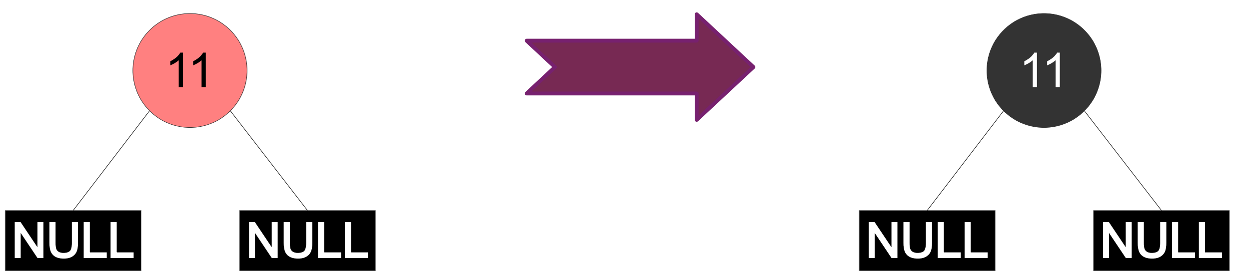
  – Se incumple la propiedad 3, pero se puede arreglar con rotaciones o cambiando el color de los ancestros.

Árboles RN

- Inserción:

  <u>Caso 1:</u> el nodo insertado es la raíz del árbol (primera inserción en un árbol vacío).

  - en este caso, el nodo se repinta a **negro**, para cumplir la propiedad 2.

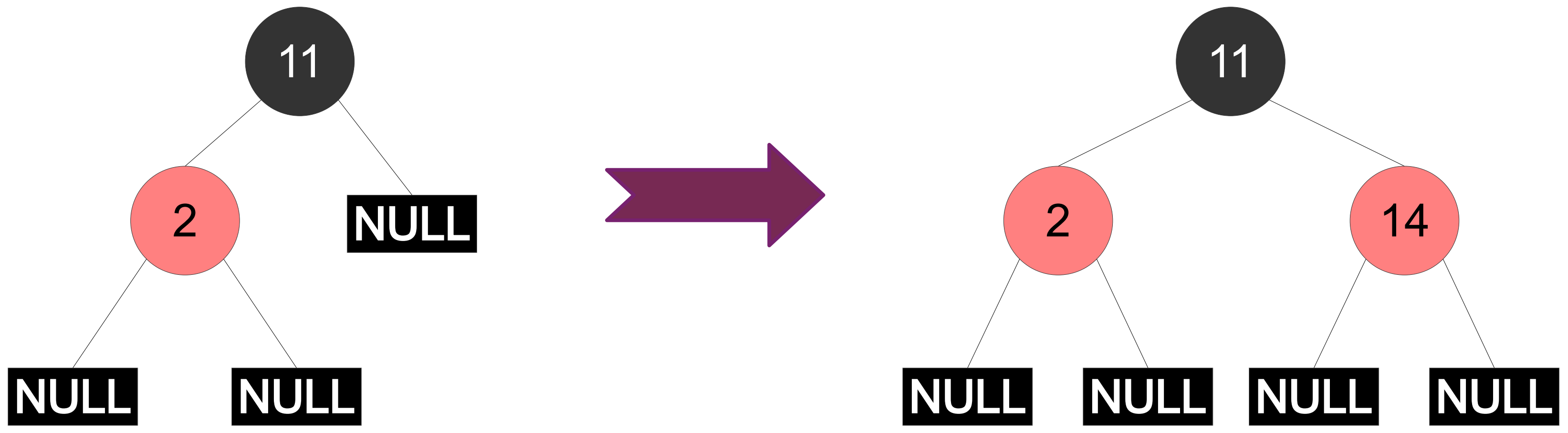


Árboles RN

- Inserción:

  <u>Caso 2:</u> el padre del nodo insertado es **negro**.

  - en este caso, el árbol es válido sin ninguna modificación.



Árboles RN

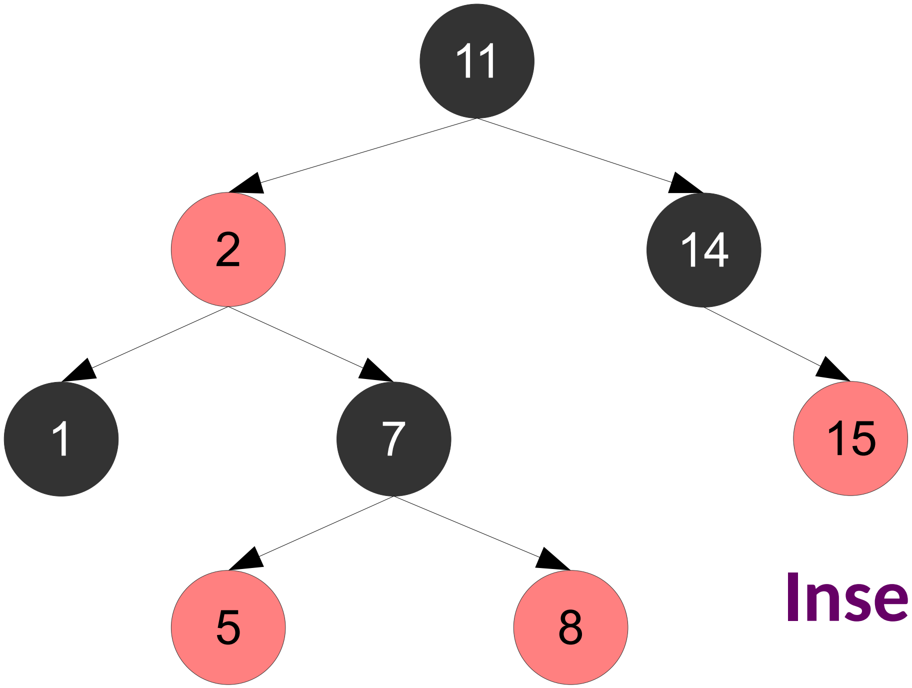
- Inserción:

  Caso 3: el padre del nodo insertado es **rojo**, se analizan varias situaciones:

  - **Situación 1:** Que el tío del nodo (hermano del padre) sea **rojo**.

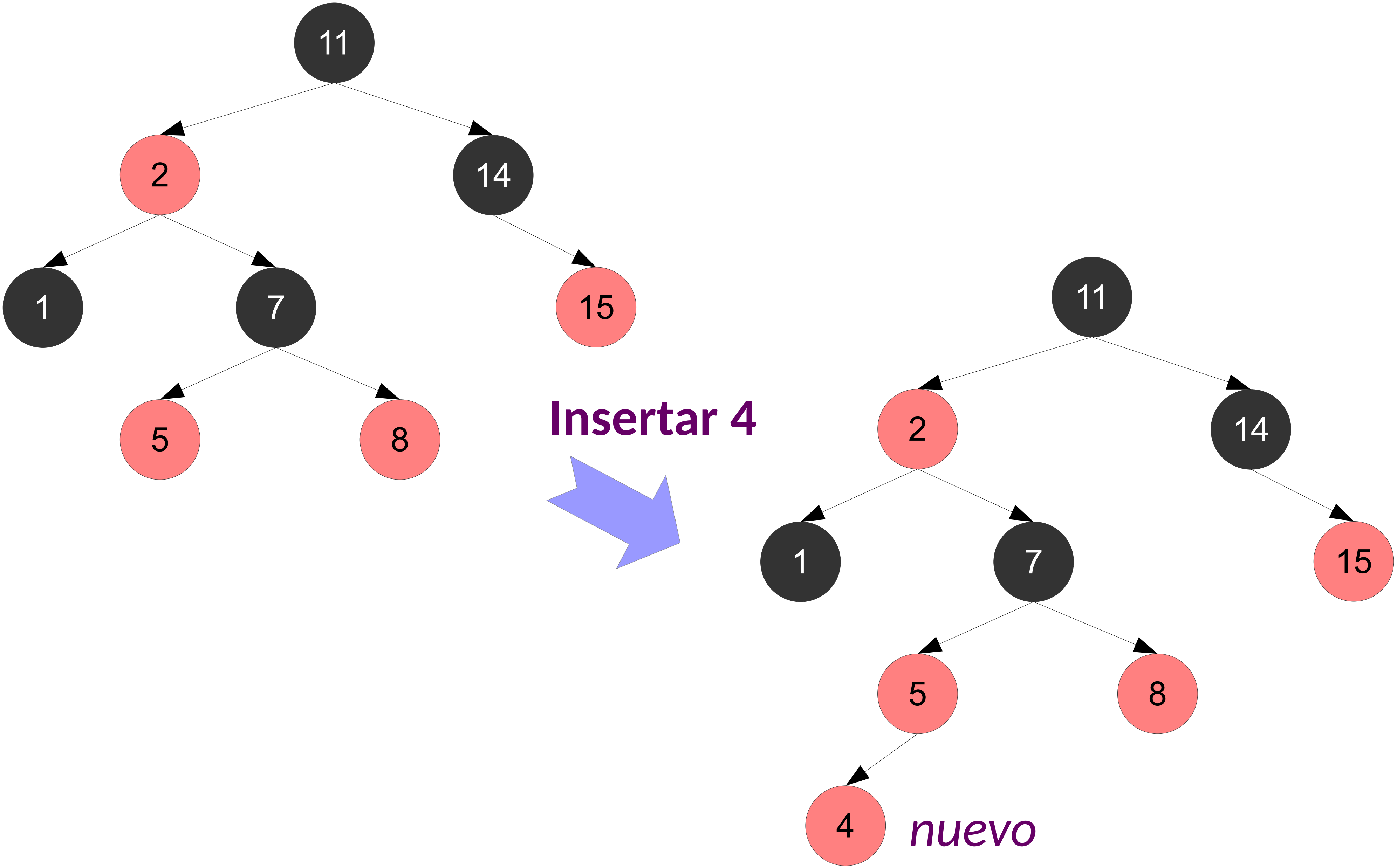
    En ese caso, se cambia el color del padre, del tío y del abuelo.

Árboles RN

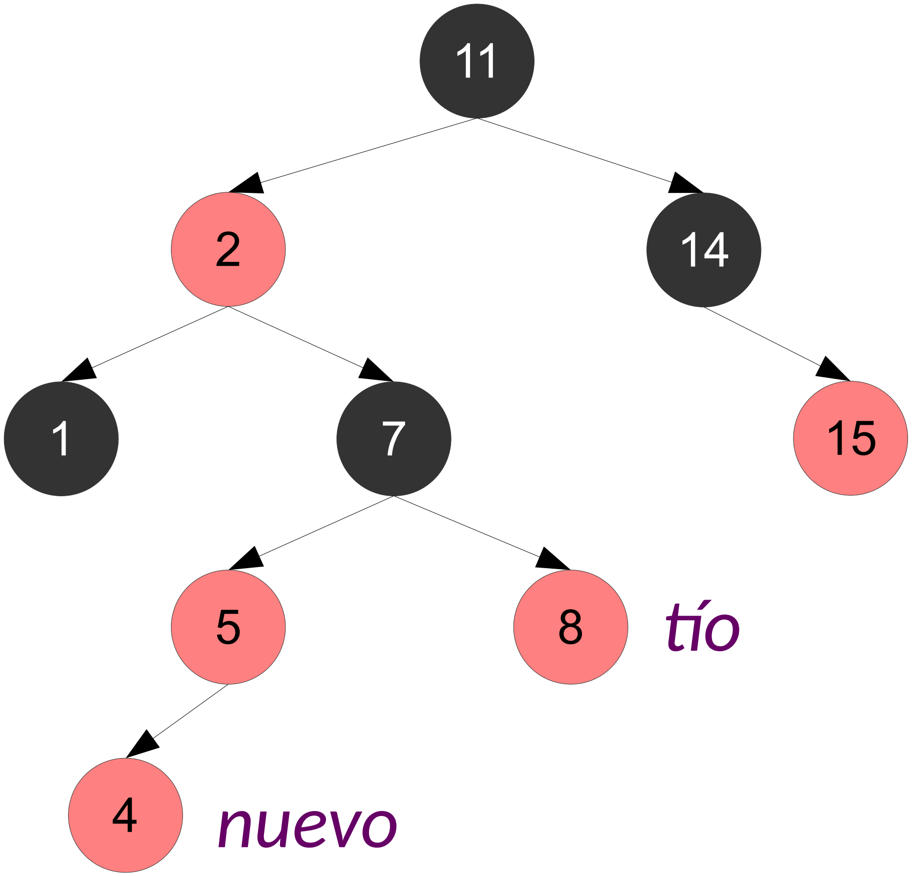


Insertar 4

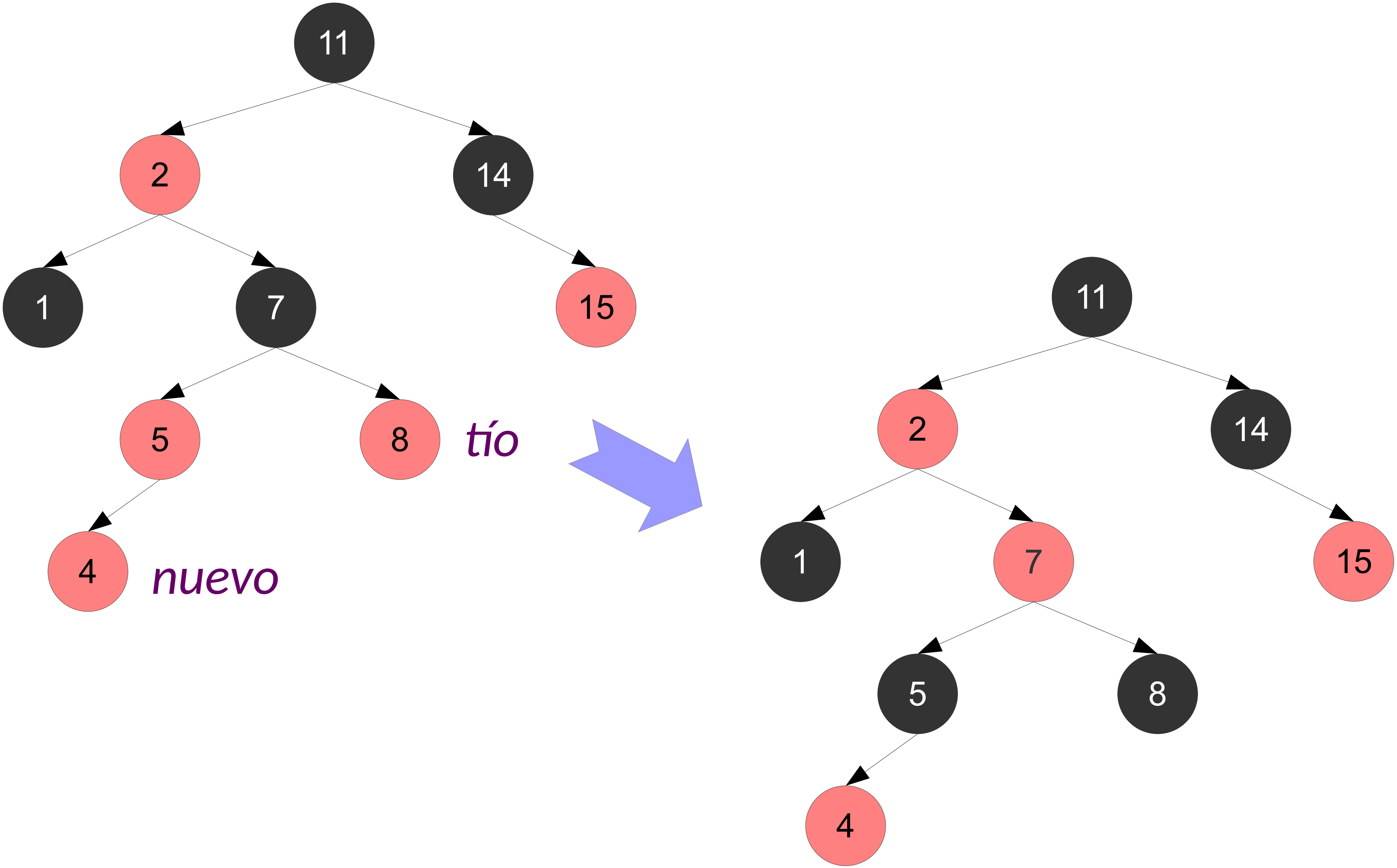
Árboles RN



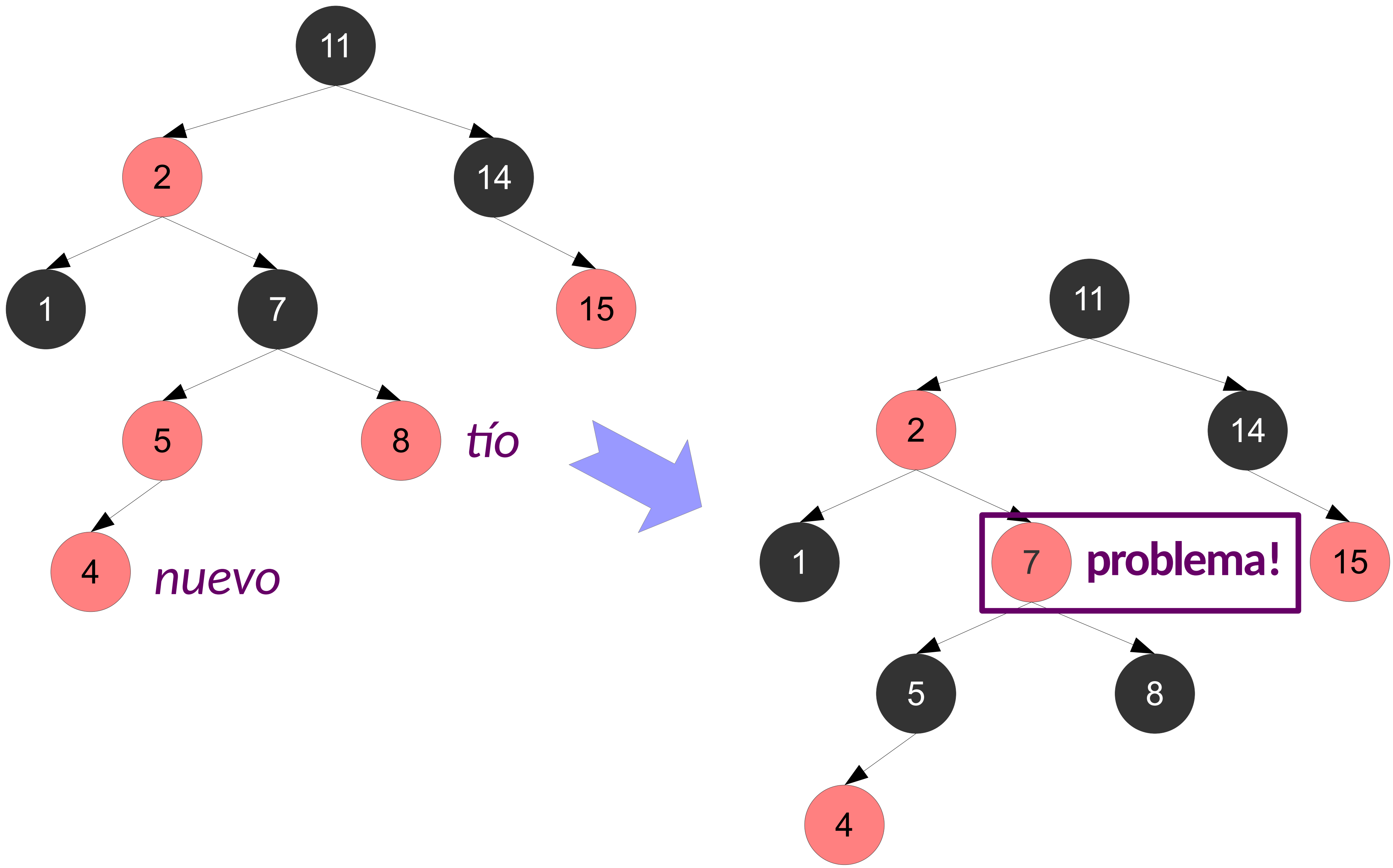
Árboles RN



Árboles RN



Árboles RN



Árboles RN

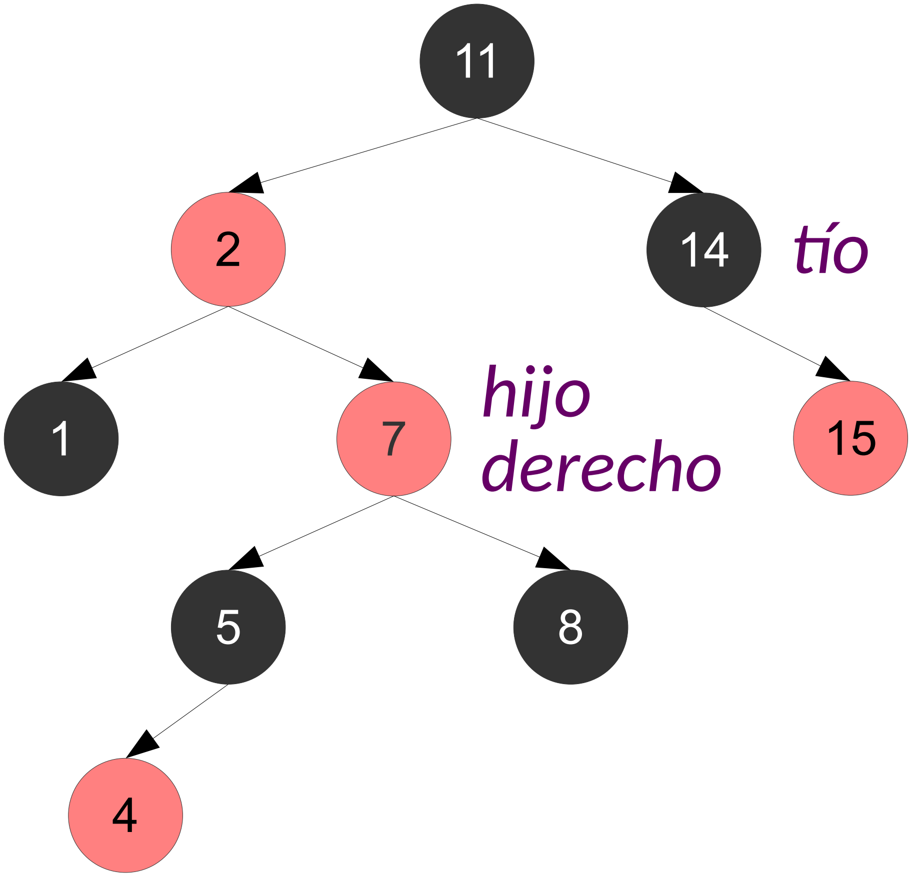
- Inserción

  Caso 3: el padre del nodo insertado es **rojo**, se analizan varias situaciones:

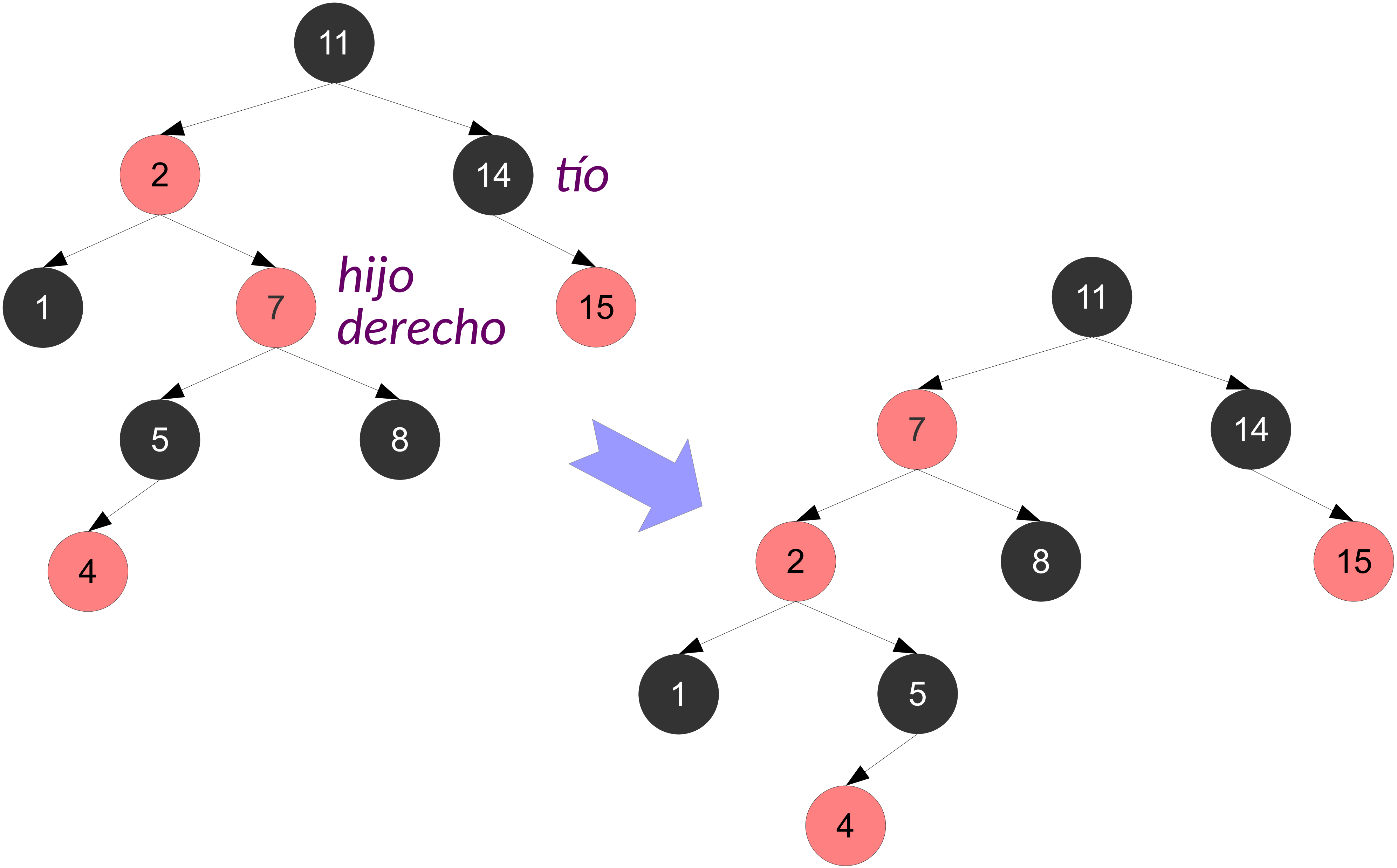
  - **Situación 2:** Que el tío del nodo (hermano del padre) sea **negro** y el nodo sea el hijo derecho del padre.

    En ese caso, se aplica una rotación a izquierda sobre el nodo.

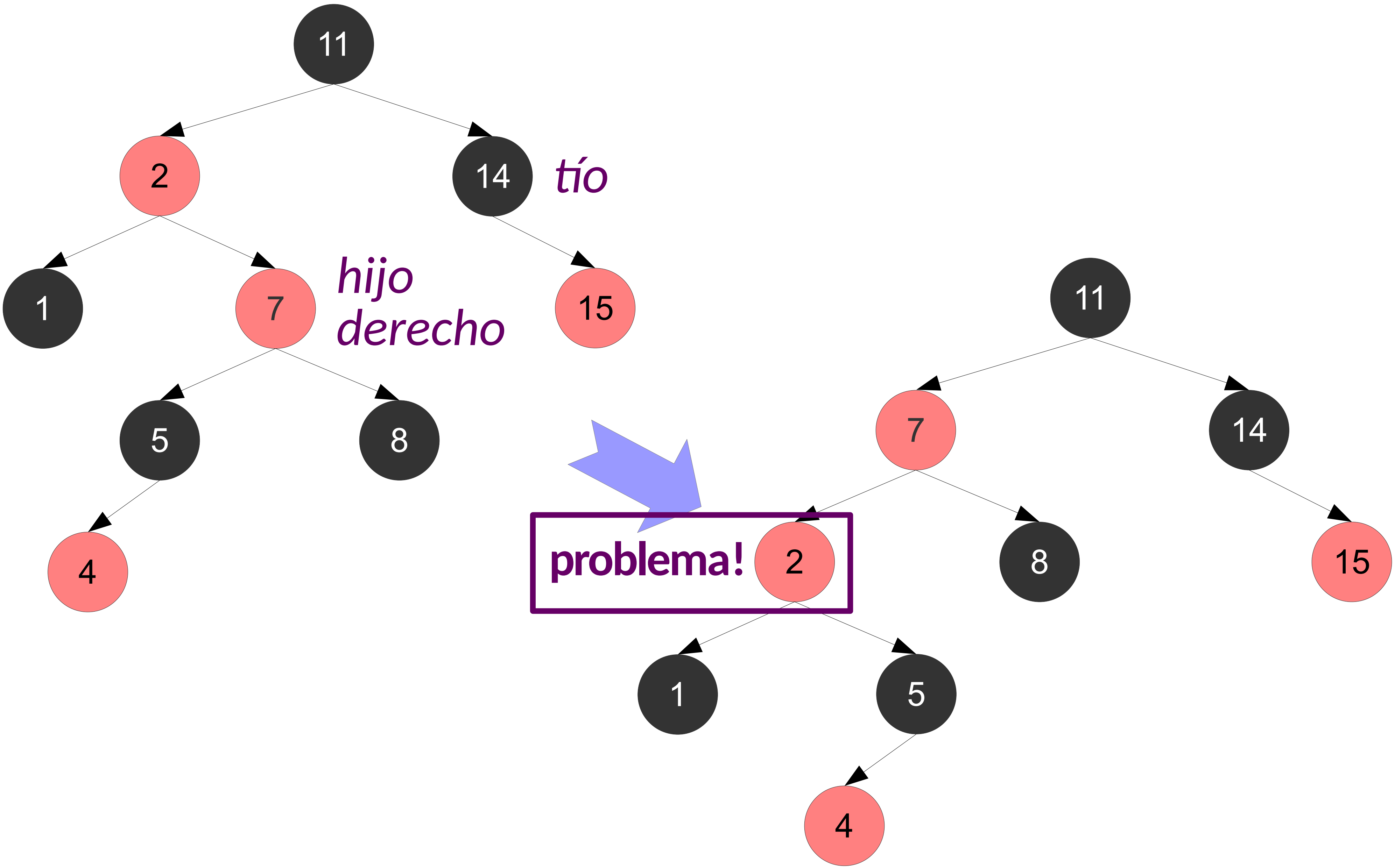
Árboles RN



Árboles RN



Árboles RN



Árboles RN

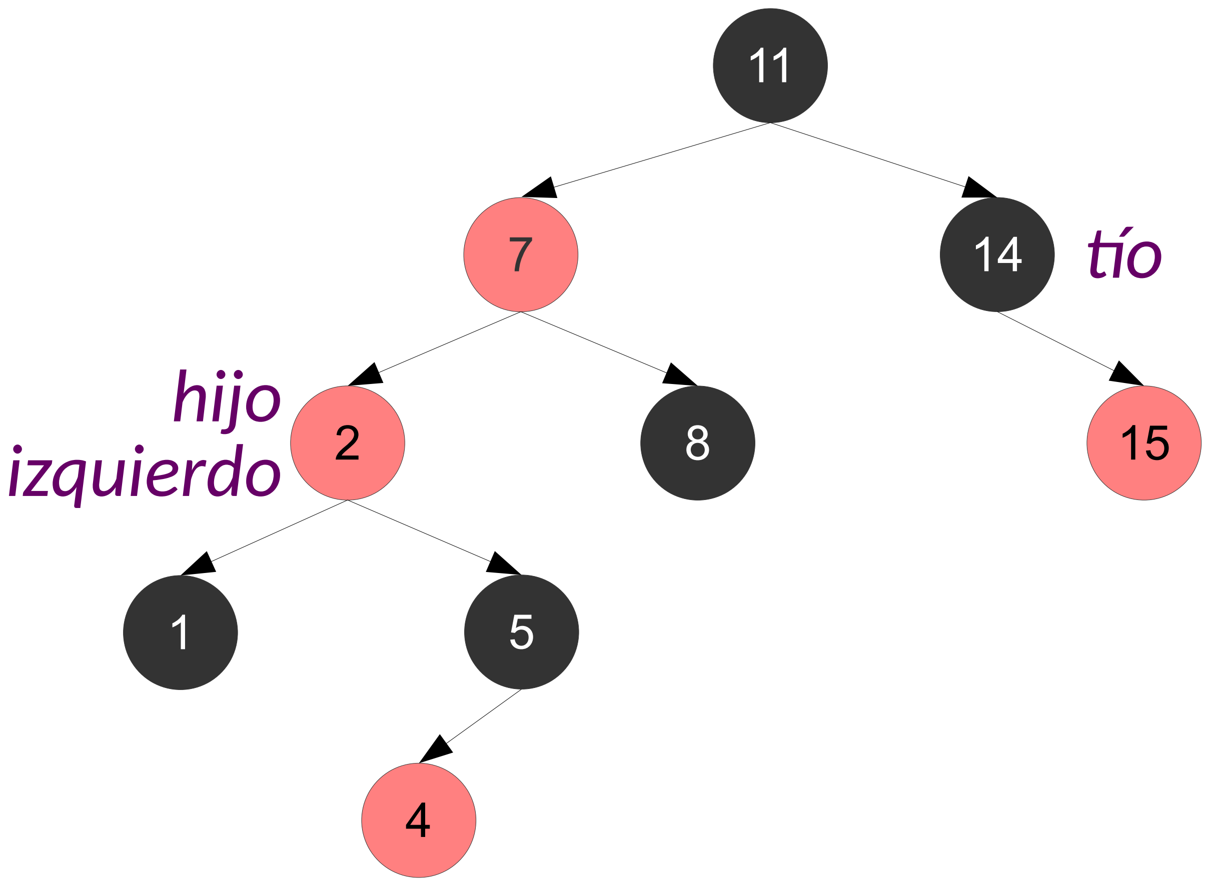
- Inserción

  Caso 3: el padre del nodo insertado es **rojo**, se analizan varias situaciones:

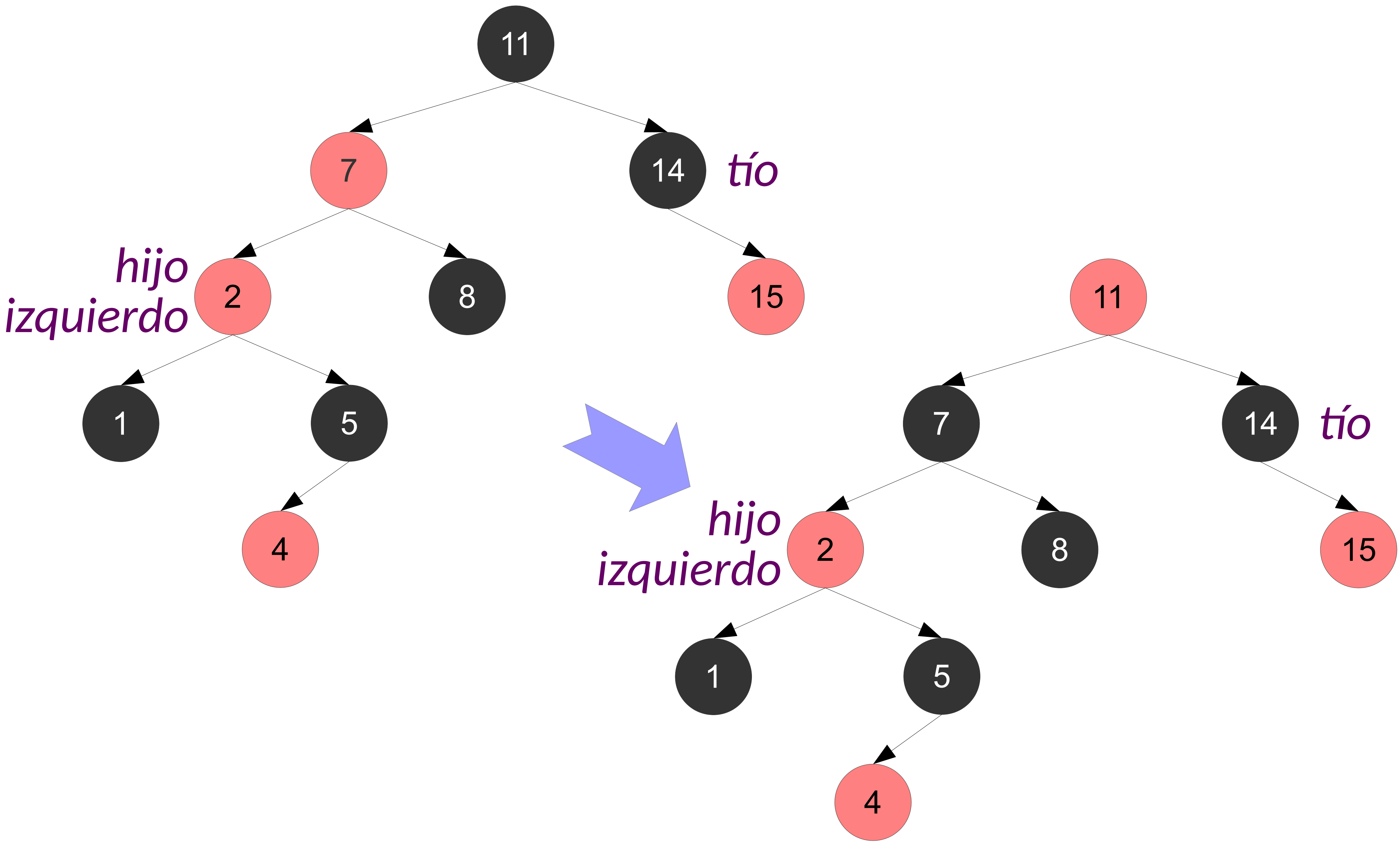
  - **Situación 3:** Que el tío del nodo (hermano del padre) sea **negro** y el nodo sea el hijo izquierdo del padre.

    En ese caso, se cambia el color del padre y del abuelo, y luego se aplica una rotación a derecha sobre el abuelo.

Árboles RN

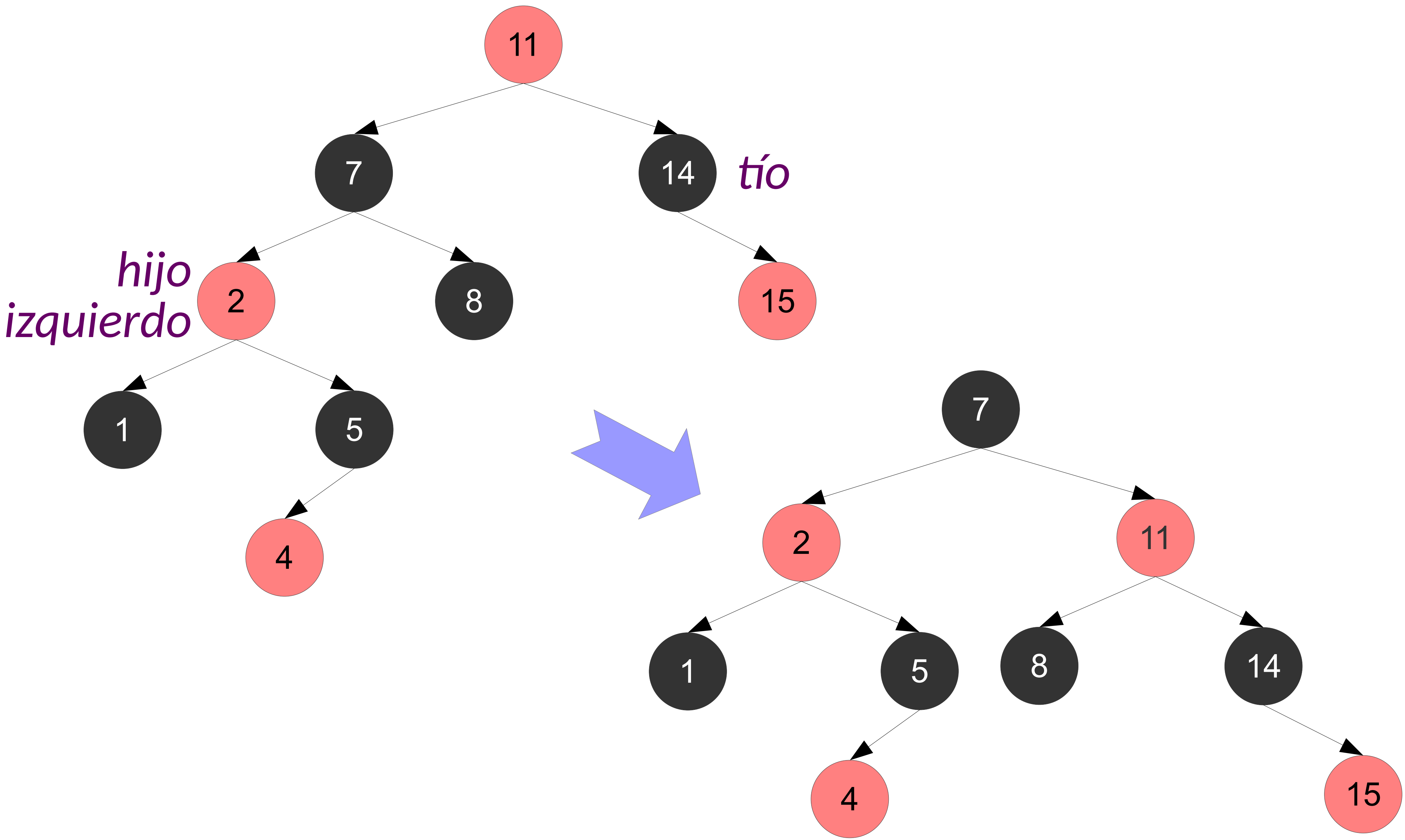


Árboles RN



11
7
14
tío
hijo izquierdo
2
8
15
1
5
4
11
7
14
tío
hijo izquierdo
2
8
15
1
5
4

Árboles RN



Árboles RN

- Eliminación:

  Se consideran los mismos casos de eliminación que en un árbol binario ordenado:

  - Eliminar un nodo hoja.
  - Eliminar un nodo con un solo hijo (derecho o izquierdo).
  - Eliminar un nodo con dos hijos.

Árboles RN

- Eliminación:

  ... pero además hay que tener en cuenta el color del nodo a eliminar ...

  Si se elimina un nodo **rojo**, usualmente las propiedades se siguen manteniendo.

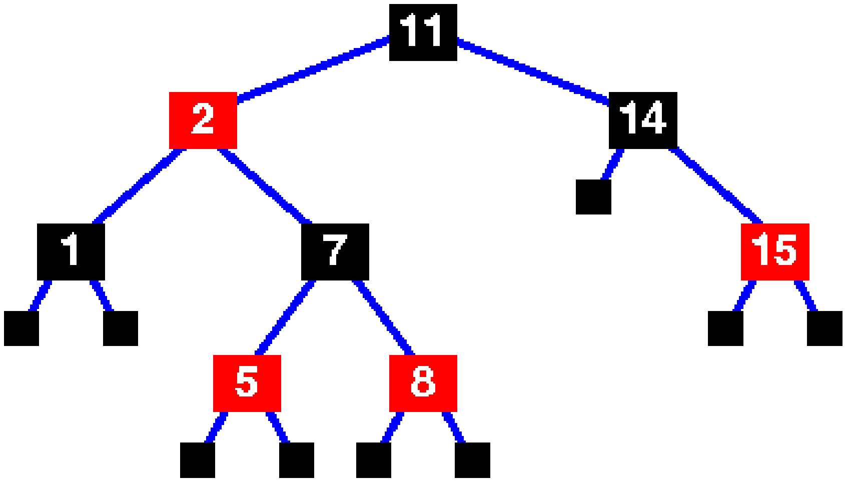
  Si se elimina un nodo **negro**, es necesario identificar las opciones de cambio de color y rotación para garantizar las propiedades del árbol.

Árboles RN

- Applet de demostración

  http://www.cs.usfca.edu/~galles/visualization/RedBlack.html

Árbol RN en la STL



- `#include <set>`
  - `std::set< T >`
  - `std::multiset< T >`
- `#include <map>`
  - `std::map< K, T >`
  - `std::multimap< K, T >`

STL: `std::set< T >`

- Se "ve" como una lista ordenada de elementos únicos.
- `T` debe ser un tipo de "ordenamiento estrictamente débil" (*strict weak ordering*).
  - i.e. Debe existir el operador "<" para `T`.

```
std::set< T >::insert( const T& v );
std::set< T >::erase( iterator pos );
std::set< T >::[r]begin( );
std::set< T >::[r]end( );
```

STL: `std::map< K, T >`

- Se “ve” como un vector dinámico de índices de elementos únicos.
- `K` debe ser un tipo de “ordenamiento estrictamente débil” (*strict weak ordering*).
  - i.e. Debe existir el operador “<” para `K`.

```
std::map< int, float > v;
        ↔ float v[ MAX ];
```

```
std::map< K, T >::iterator it;
```

```
it->first; // K
```

```
it->second; // T
```

```
struct ltstr
{
  bool operator()(const char* s1, const char* s2) const
  {
    return strcmp(s1, s2) < 0;
  }
};

int main()
{
  map<const char*, int, ltstr> months;

  months["january"] = 31;
  months["february"] = 28;
  months["march"] = 31;
  months["april"] = 30;
  months["may"] = 31;
  months["june"] = 30;
  months["july"] = 31;
  months["august"] = 31;
  months["september"] = 30;
  months["october"] = 31;
  months["november"] = 30;
  months["december"] = 31;

  cout << "june -> " << months["june"] << endl;
  map<const char*, int, ltstr>::iterator cur  = months.find("june");
  map<const char*, int, ltstr>::iterator prev = cur;
  map<const char*, int, ltstr>::iterator next = cur;
  ++next;
  --prev;
  cout << "Previous (in alphabetical order) is " << (*prev).first << endl;
  cout << "Next (in alphabetical order) is " << (*next).first << endl;
}
```

https://replit.com/@Andrea_del_Pila/EDD-UsandoMap

Árboles

- Implementaciones de:

  - Árbol general (`ArbolGeneral`), nodo general (`NodoGeneral`).

  - Árbol binario ordenado (`ArbolBinarioOrd`), nodo binario (`NodoBinario`).

  - Árbol AVL (`ArbolAVL`), nodo AVL (`NodoAVL`).

  - Árbol RN (implementación STL).

Referencias

- L. Joyanes Aguilar, I. Zahonero. Algoritmos y estructuras de datos: una perspectiva en C. McGraw-Hill, 2004.
- www.cs.duke.edu/~reif/courses/alglectures/skiena.lectures/lecture10.pdf
- www.cse.ohio-state.edu/~gurari/course/cis680/cis680Ch11.html
- www.stolerman.net/studies/cs521/red_black_trees.pdf
- http://lcm.csa.iisc.ernet.in/dsa/node114.html
- en.wikipedia.org/wiki/Red-black_tree

Referencias

- www.cs.umd.edu/~mount/420/Lects/420lects.pdf
- www.cs.nmsu.edu/~epontell/courses/cs272/disp/trees/2004/tree2004.pdf
- www.csd.uwo.ca/~vmazalov/CS1027a/notes/CS1027-Trees_6up.pdf
- people.cis.ksu.edu/~schmidt/300s05/Lectures/Week7b.html
- pages.cs.wisc.edu/~ealexand/cs367/NOTES/AVL-Trees/index.html
- www.dcs.gla.ac.uk/~pat/52233/slides/AVLTrees1x1.pdf