Árboles AVL

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)

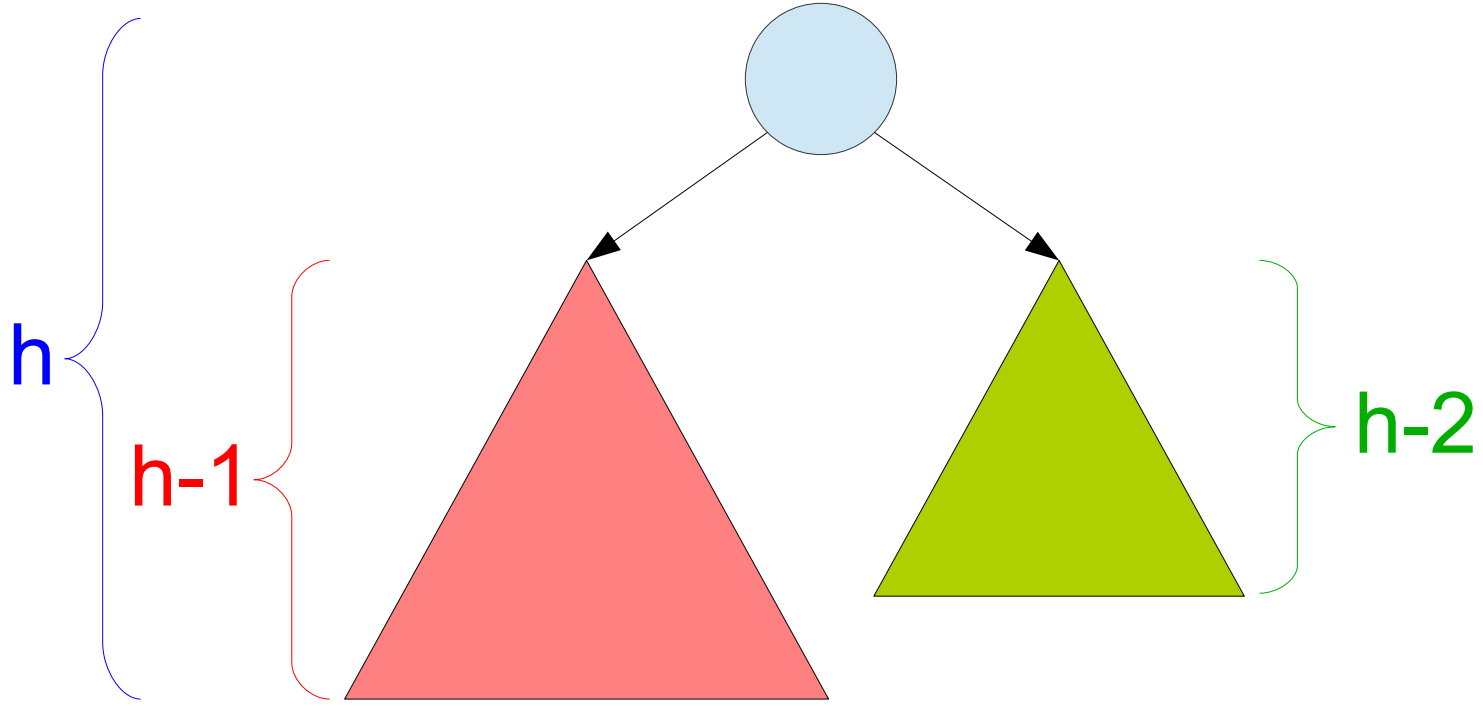
Árboles AVL

- 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.



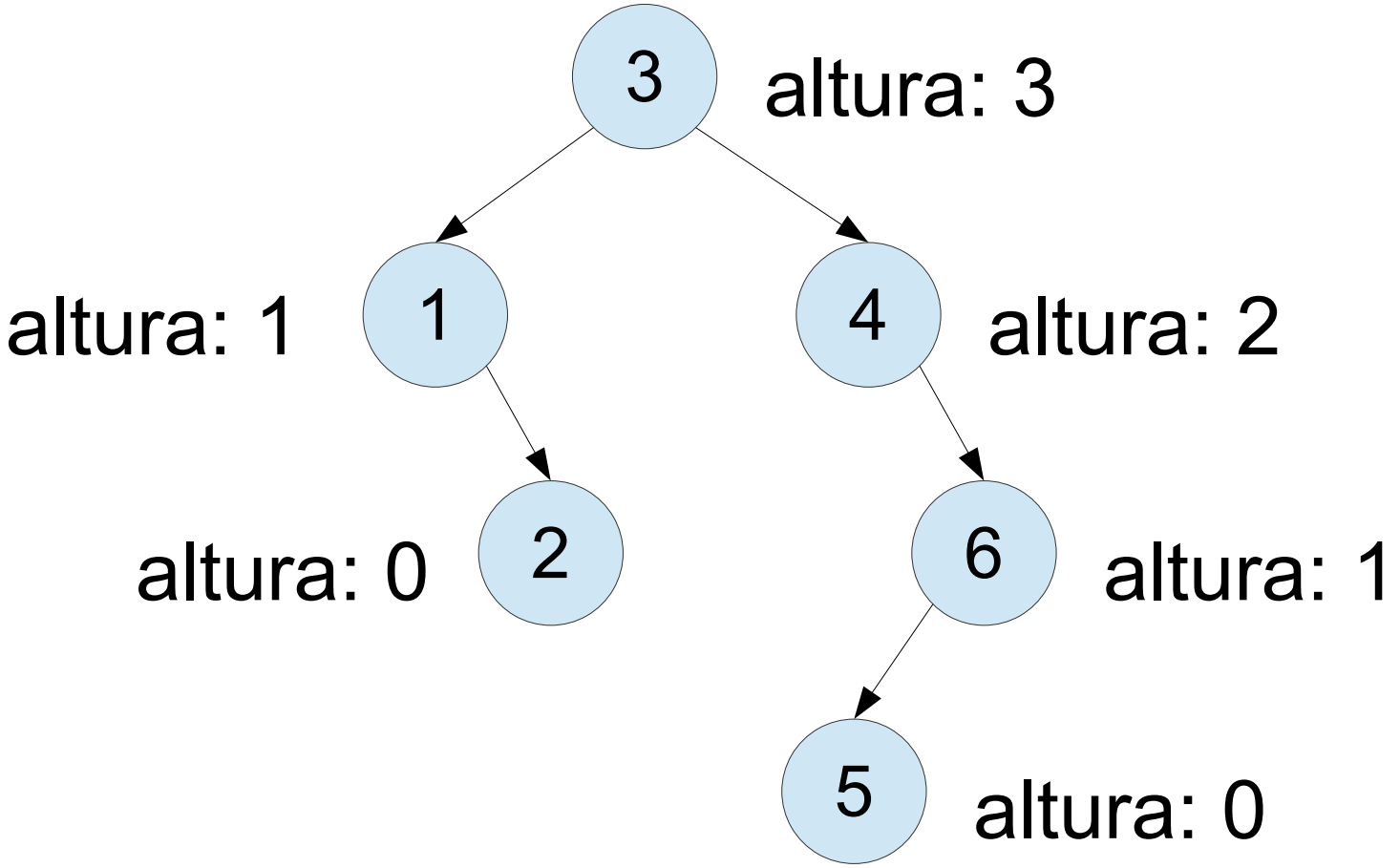
Árboles AVL

- Propiedad a garantizar:

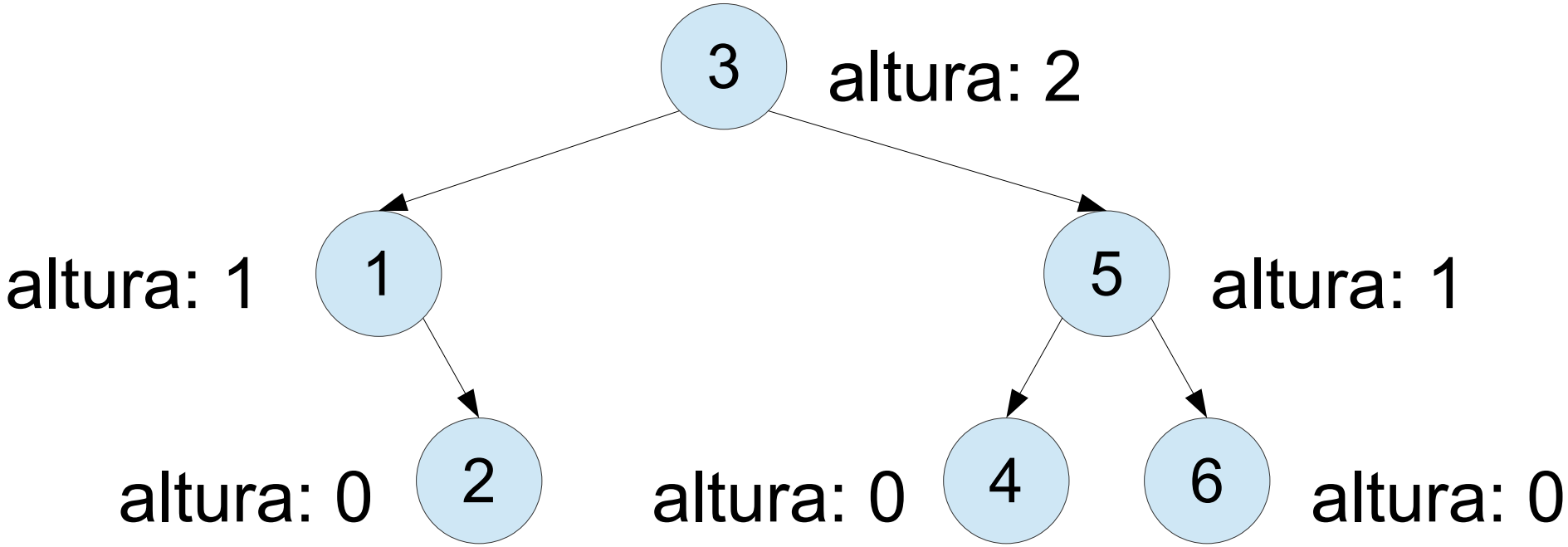
  Para cada nodo del árbol, las alturas de sus dos hijos (subárboles) difieren por mucho en 1.

  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



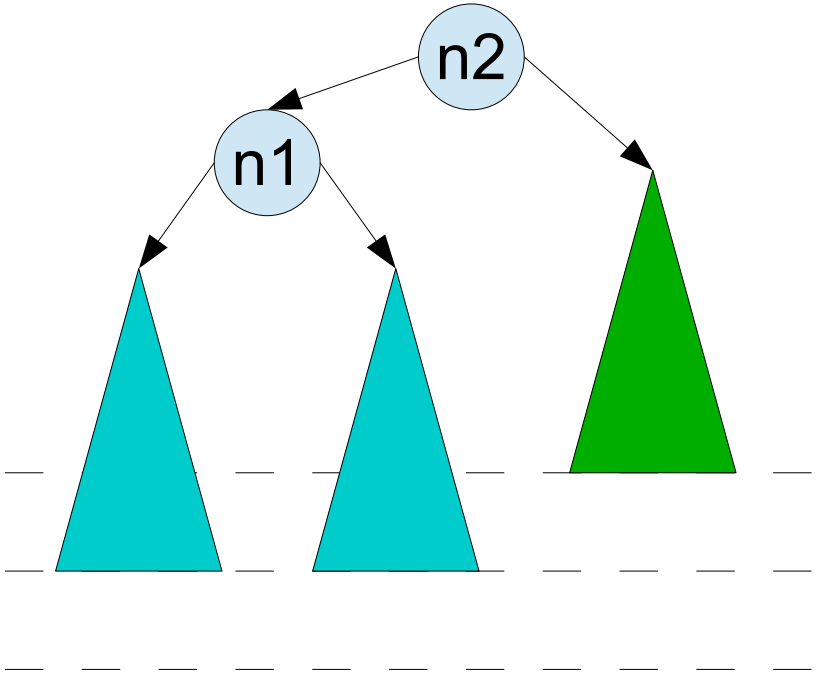
Árboles AVL

- 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.

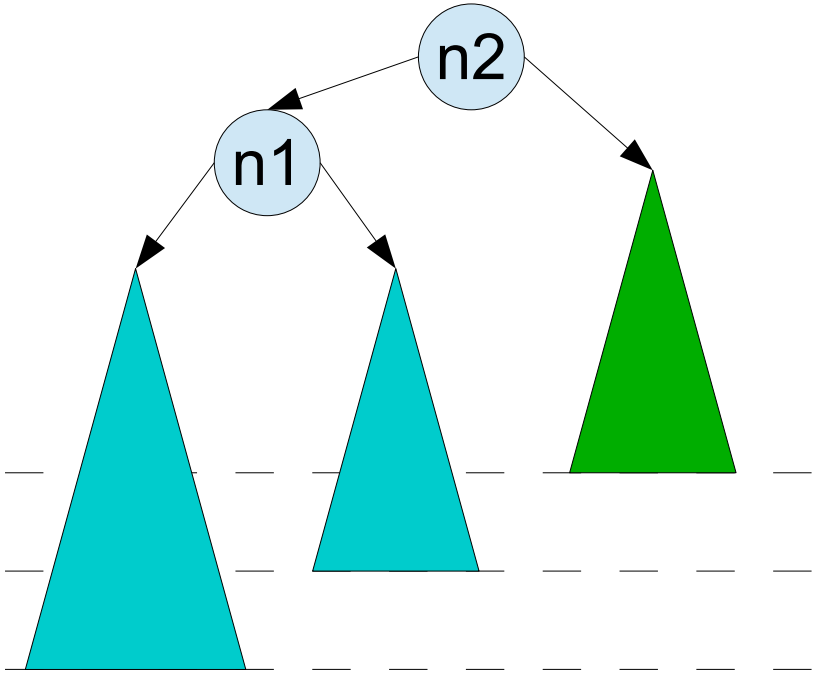
  Rotación a derecha.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

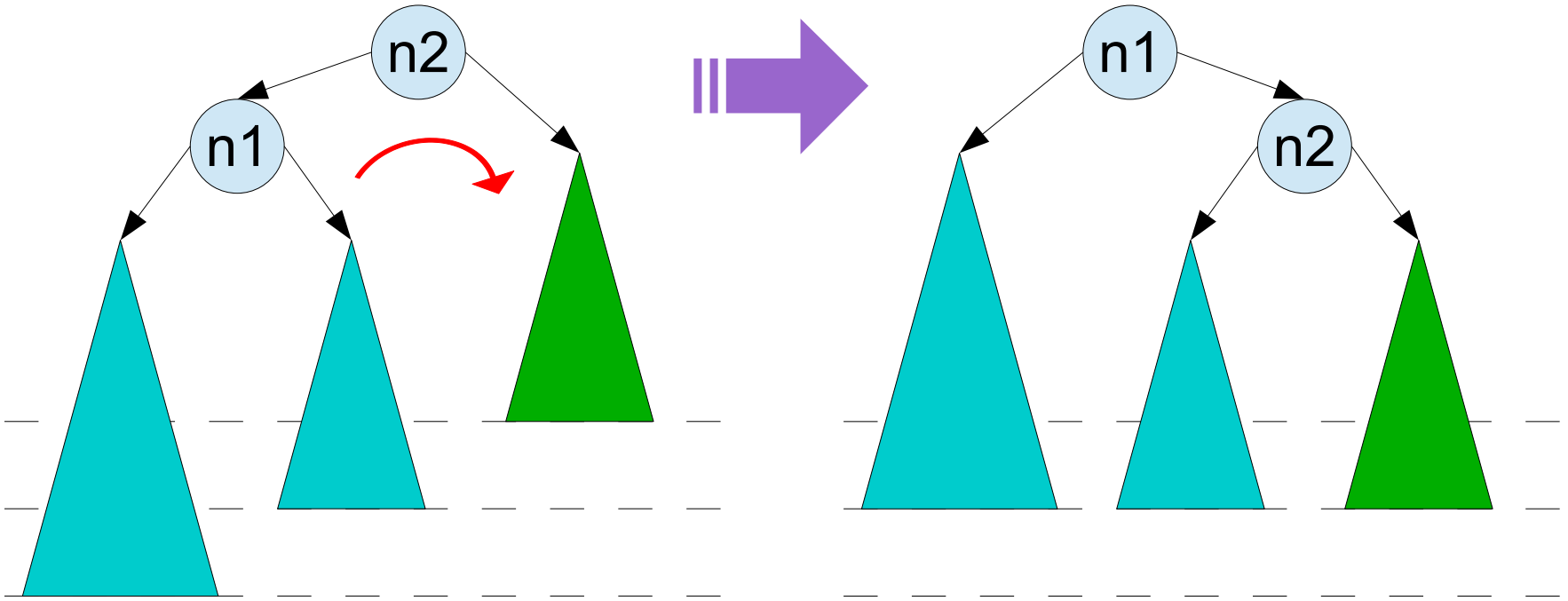
  Rotación a derecha.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

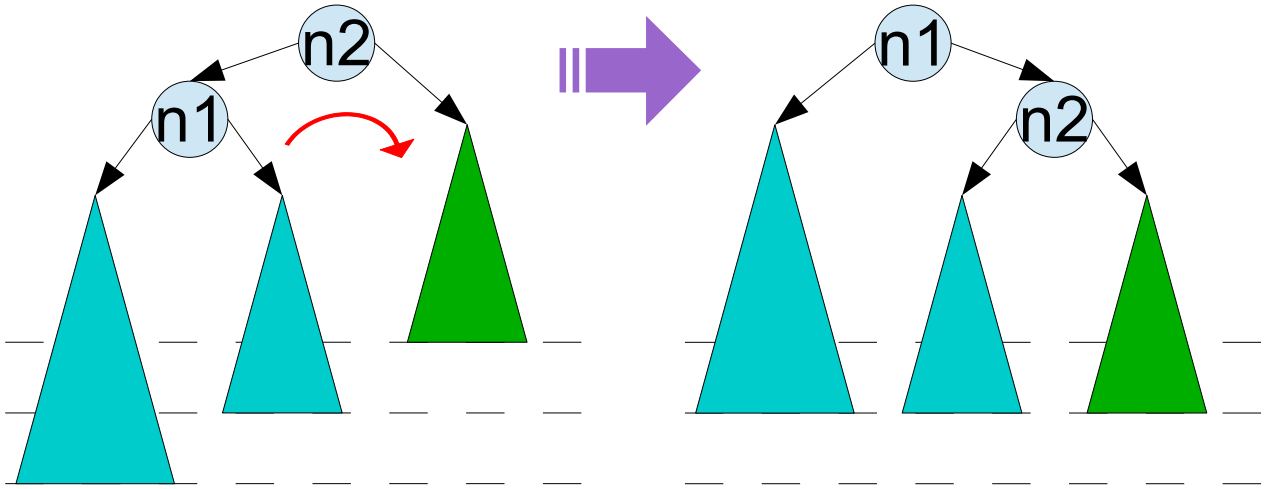
  Rotación a derecha.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Rotación a derecha.



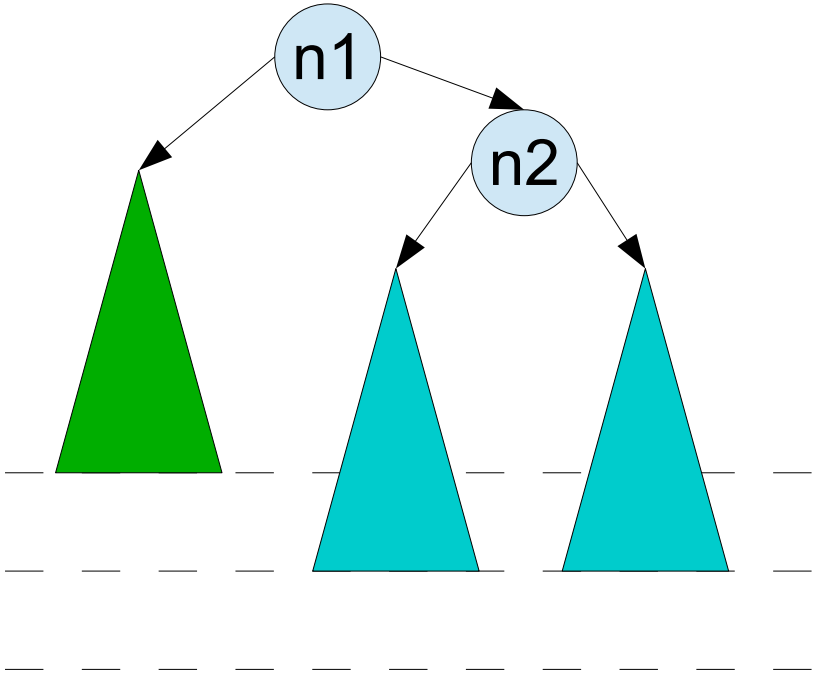
Rotación derecha sobre `n2`:

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

Árboles AVL

- Operaciones de re-balanceo → rotaciones.

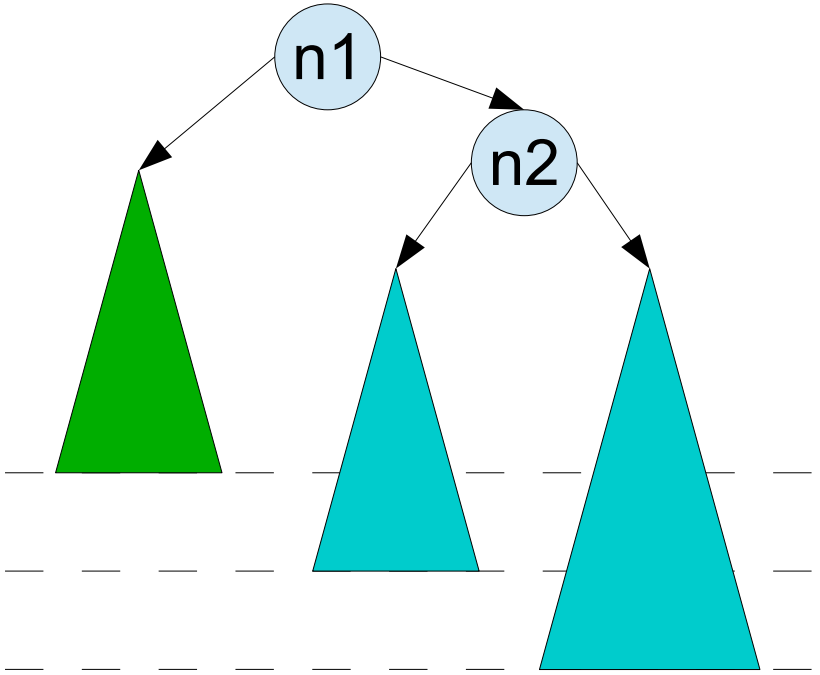
  Rotación a izquierda.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

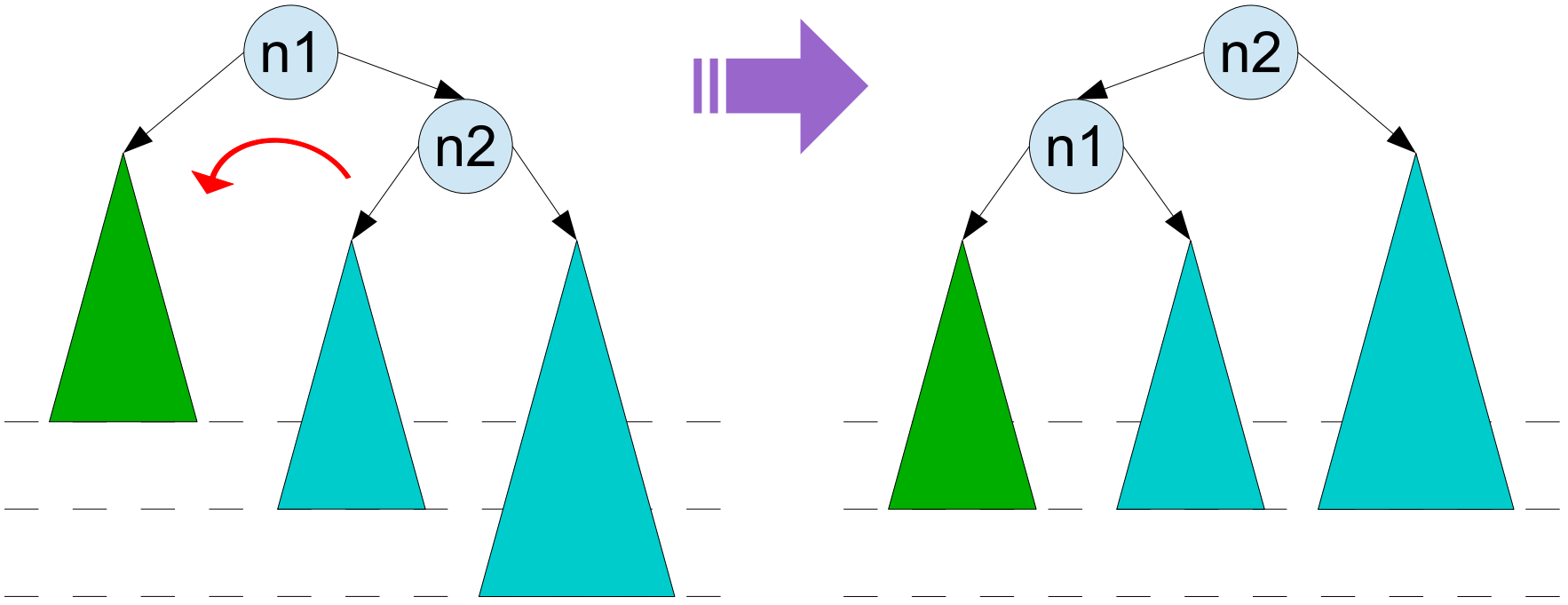
  Rotación a izquierda.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

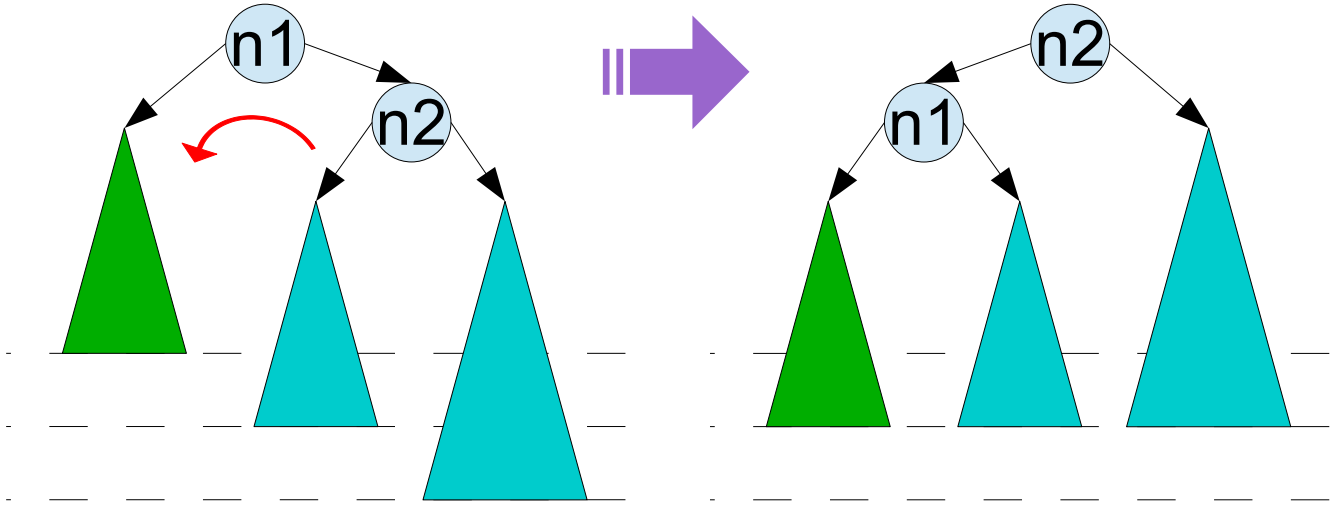
  Rotación a izquierda.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

  Rotación a izquierda.

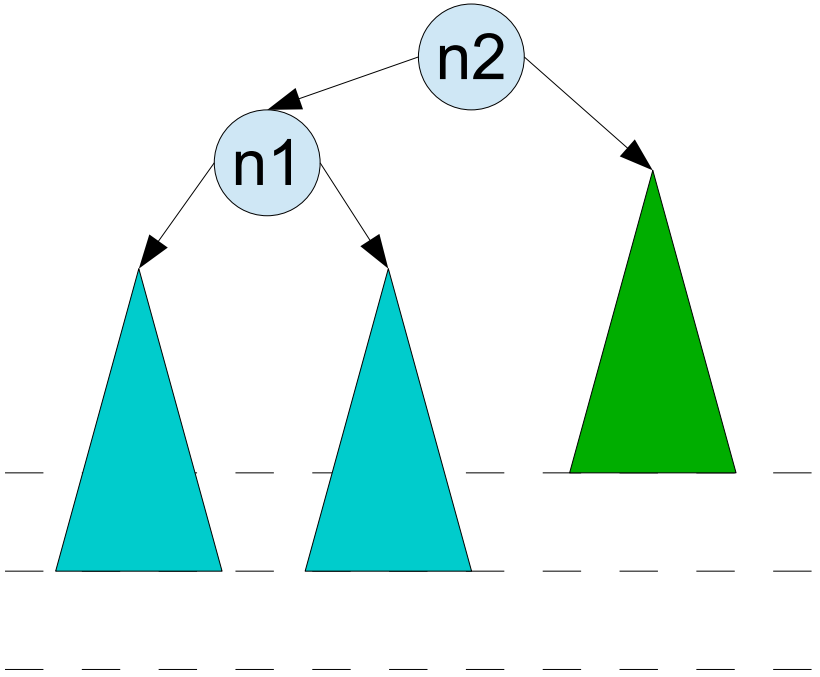


Rotación izquierda sobre `n1`:

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

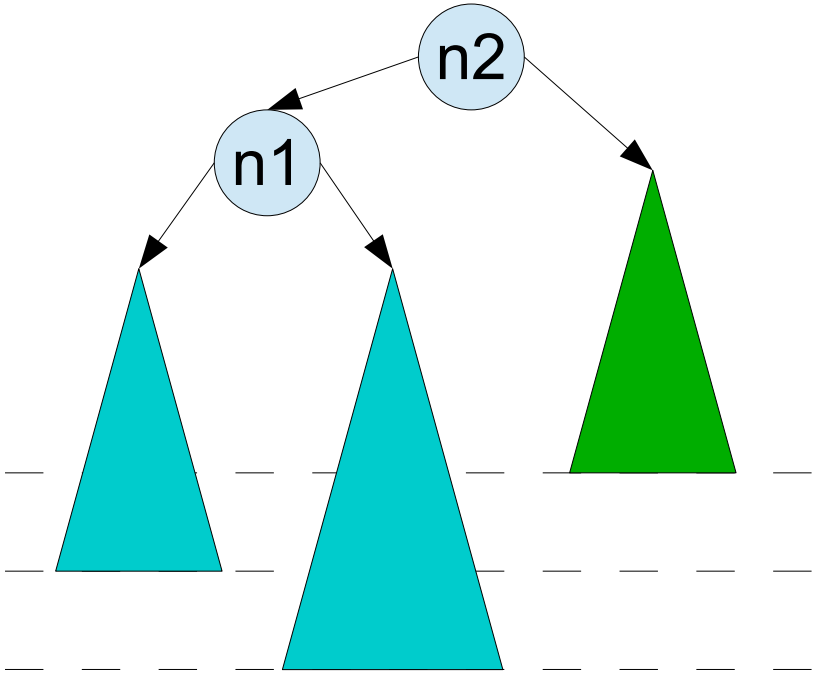
Árboles AVL

- Operaciones de re-balanceo → rotaciones.



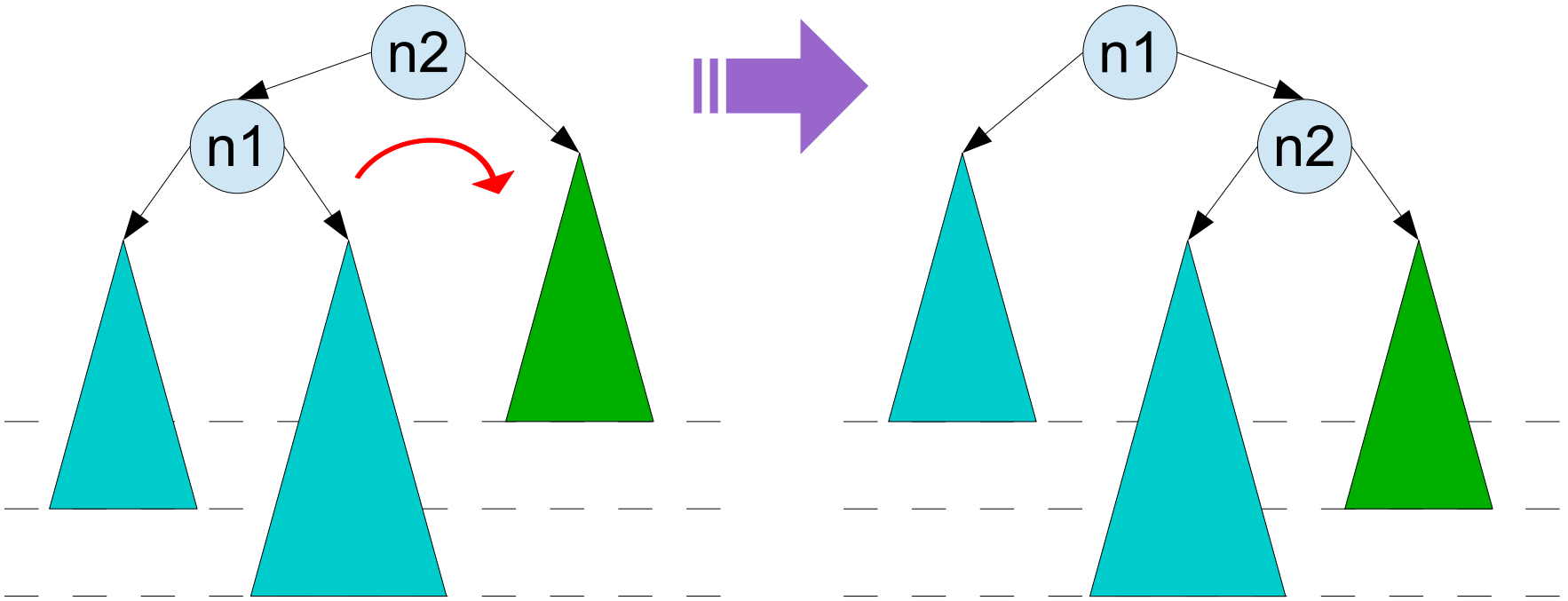
Árboles AVL

- Operaciones de re-balanceo → rotaciones.



Árboles AVL

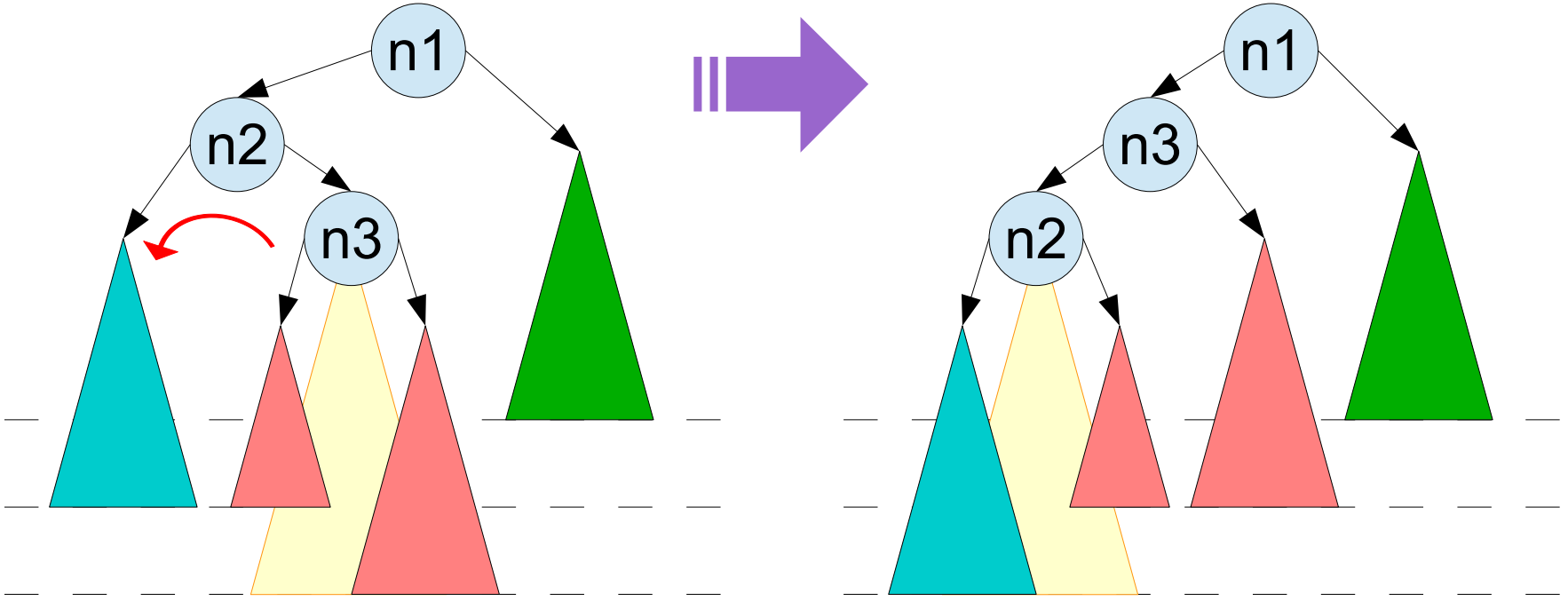
- Operaciones de re-balanceo → rotaciones.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

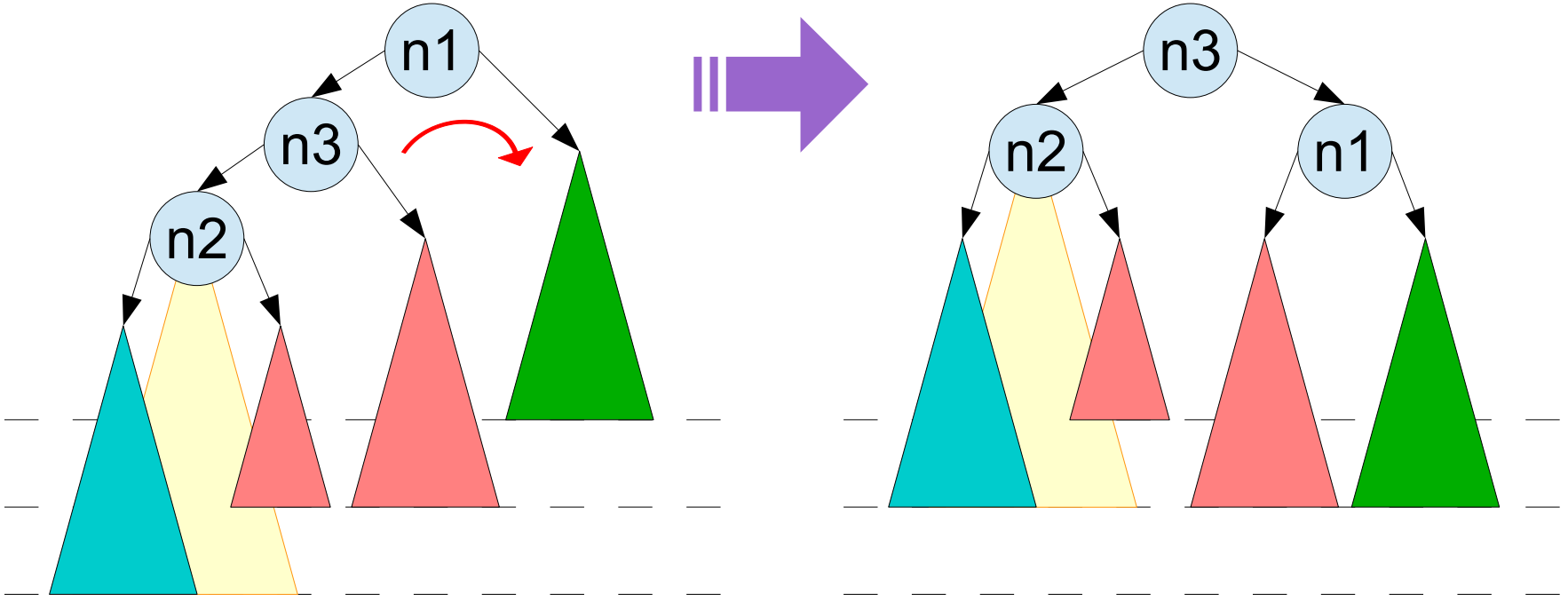
  Doble rotación 1: rotación a izquierda seguida de rotación a derecha.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

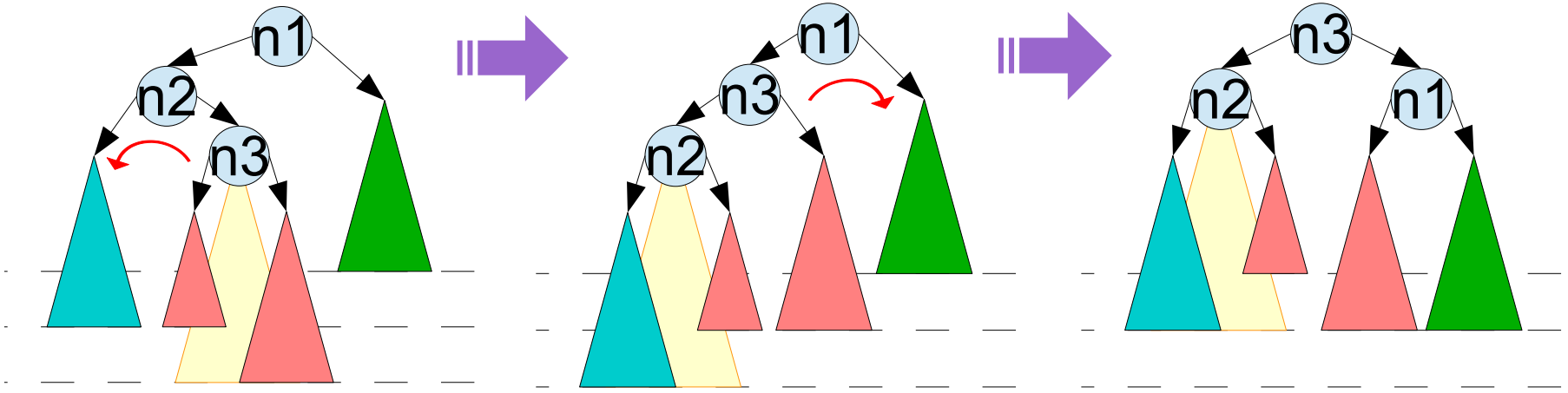
  Doble rotación 1: rotación a izquierda seguida de rotación a derecha.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

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

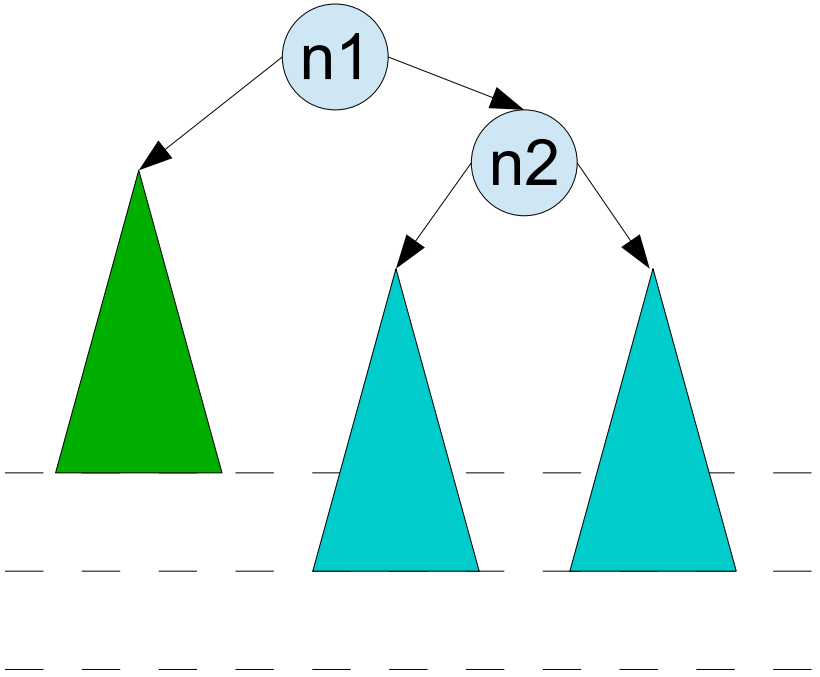


Rotación izquierda-derecha sobre `n1`:

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

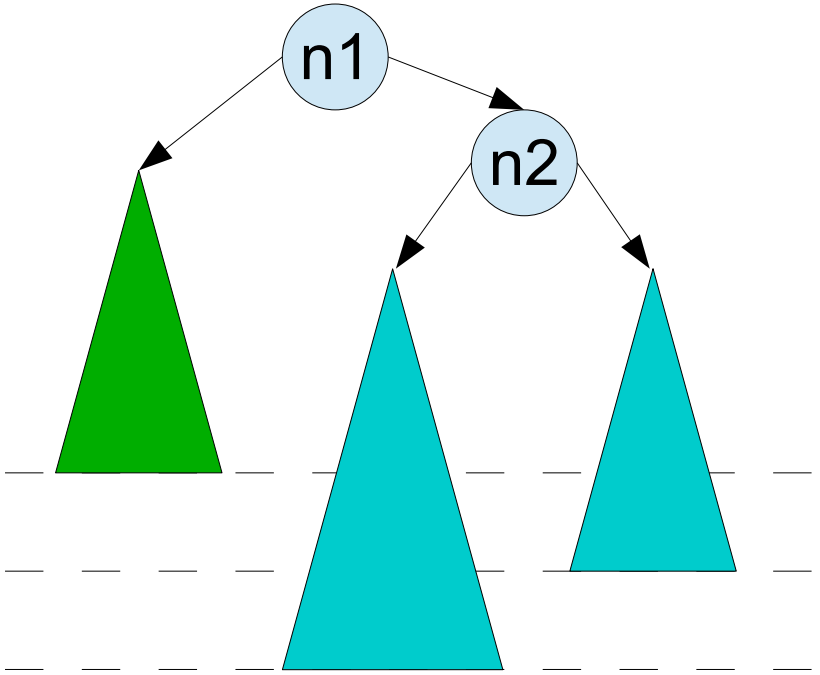
Árboles AVL

- Operaciones de re-balanceo → rotaciones.



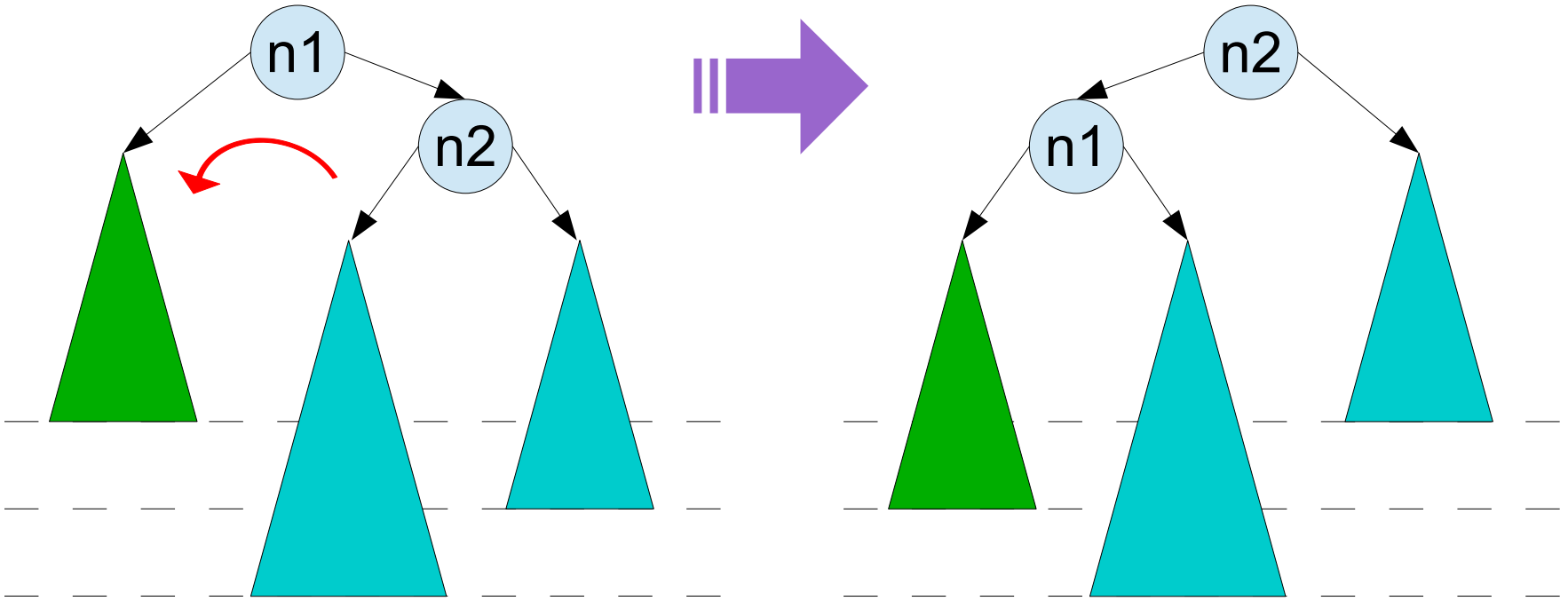
Árboles AVL

- Operaciones de re-balanceo → rotaciones.



Árboles AVL

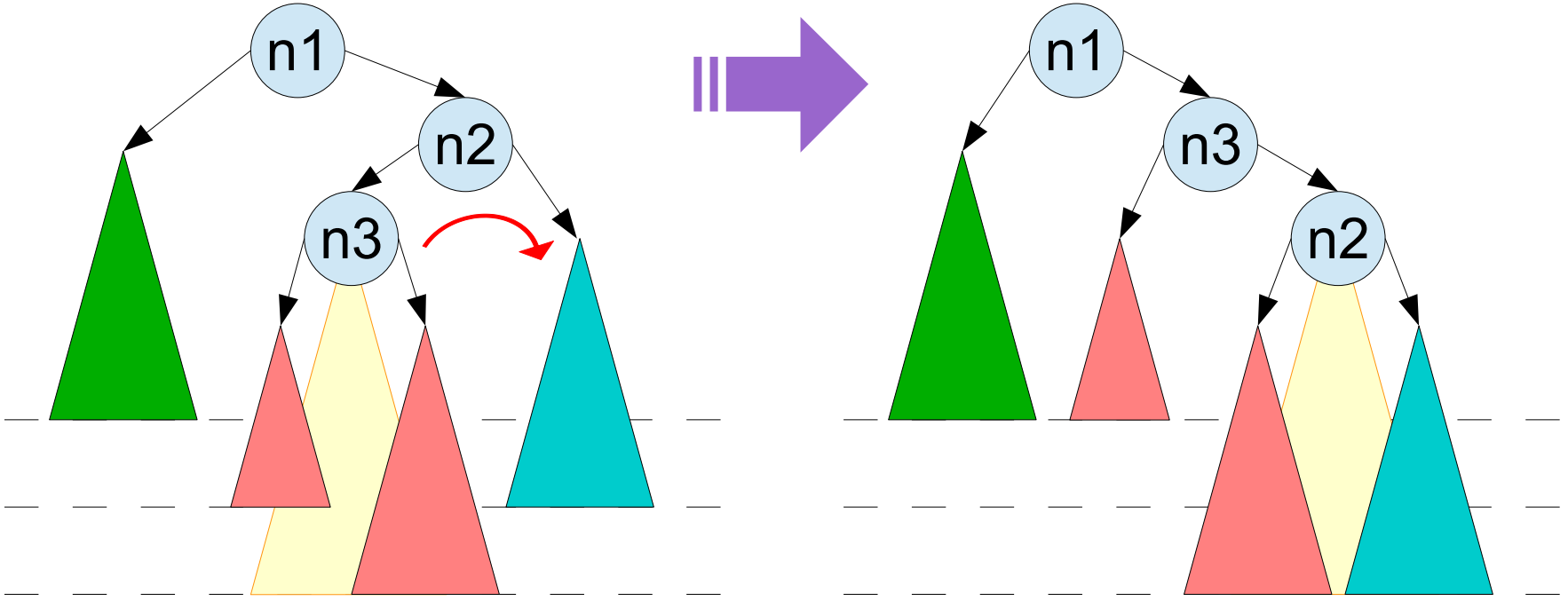
- Operaciones de re-balanceo → rotaciones.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

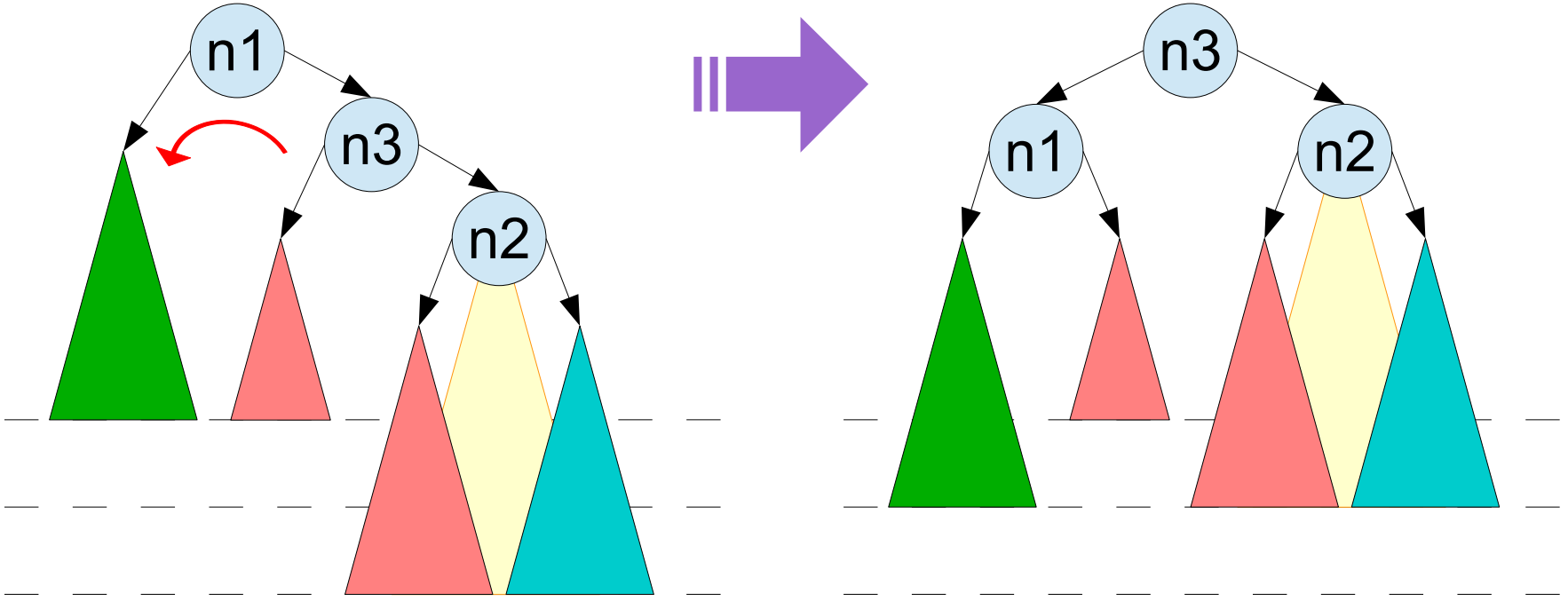
  Doble rotación 2: rotación a derecha seguida de rotación a izquierda.



Árboles AVL

- Operaciones de re-balanceo → rotaciones.

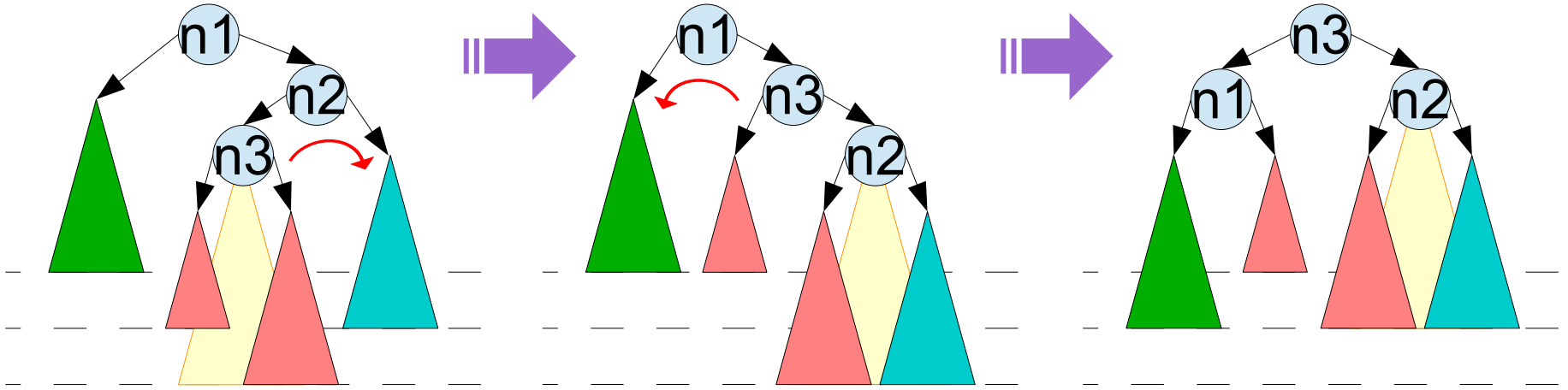
  Doble rotación 2: rotación a derecha seguida de rotación a izquierda.



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

- 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 en ese punto.

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.

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