Árboles

MC Beatriz Beltrán Martínez

Árboles y equilibrio

- Objetivo:
  - Conseguir que la altura del árbol sea mínima
- Estrategias:
  - Árboles binarios equilibrados:
    - Ej: **Árboles AVL**
  - Estructuras autoajustables:
    - Después de cada operación se aplican reglas para reestructurar el árbol
    - Ej: **Splay tree** (árboles que se "separan")

Árboles AVL

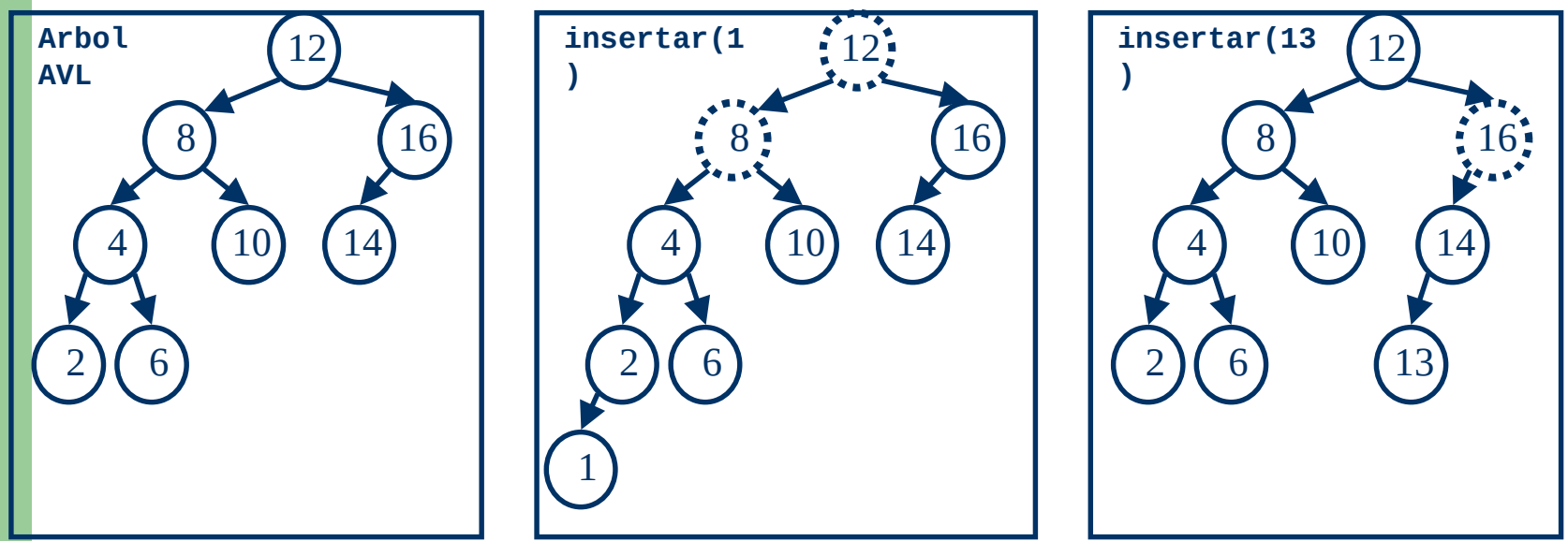
- Es un árbol binario de búsqueda equilibrado
- AVL: Adelson-Velskii & Landis (1962)
- ABB + condiciones adicionales de equilibrio
  - La condición debería de ser fácil de mantener.
- Una primera aproximación:
  - Exigir a los subárboles izquierdo y derecho la misma altura
  - Aplicar la condición a todos los nodos.

Árboles AVL

- Primera aproximación
  - Pero es una condición demasiado restrictiva:
    - Difícil insertar elementos y mantener la condición
- Otra aproximación menos "exigente":
  - **Árbol AVL**: Árbol binario de búsqueda con la condición adicional de que para cualquier nodo del árbol, la diferencia izq/der es como máximo 1.
  - Definimos la altura del subárbol vacío como –1.

Árboles AVL

- Inconveniente: Modificaciones (insertar/borrar)
  - Pueden destruir el equilibrio de algunos nodos
  - Dificultad para mantener la condición de equilibrio



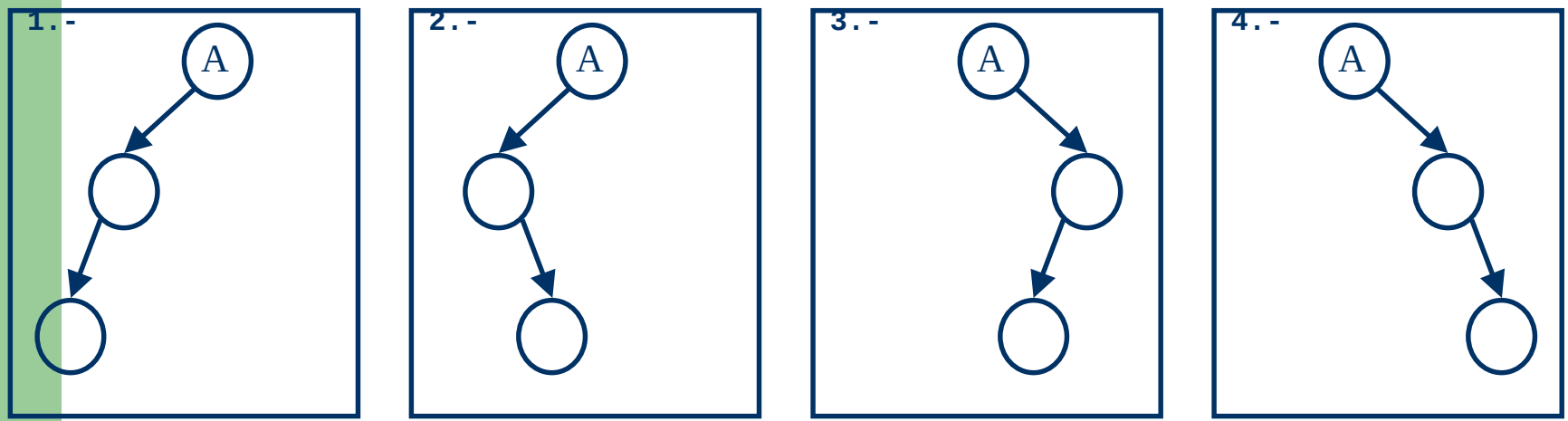
Árboles AVL. Inserción

- Nodos a los que afecta la inserción:
  - Nodos el camino entre la raíz y el punto de inserción
  - El resto no se ven afectados
- Recorrer ese camino y garantizar el equilibrio
  - Se reequilibra el más profundo de los afectados
- Esta operación reequilibra el árbol AVL

Árboles AVL. Inserción

- 4 posibles situaciones que causan desequilibrio
- Desequilibrio causado por inserción en:
  - ...subárbol izquierdo del hijo izquierdo de A
  - ...subárbol derecho del hijo izquierdo de A
  - ...subárbol izquierdo del hijo derecho de A
  - ...subárbol derecho del hijo derecho de A
- El equilibrio se restaura con una operación:
  - Rotación

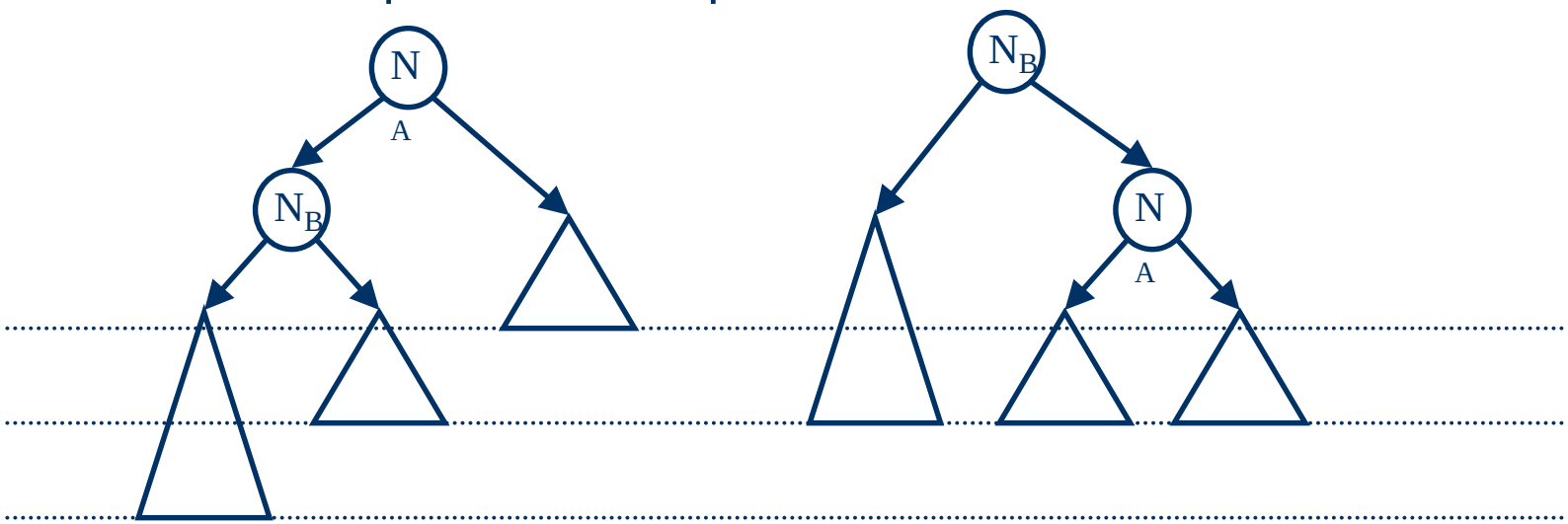
Árboles AVL. Inserción



- Casos 1 y 4: Desequilibrio "por el exterior"
  - Rotación simple
- Casos 2 y 3: Desequilibrio "por el interior"
  - Rotación doble

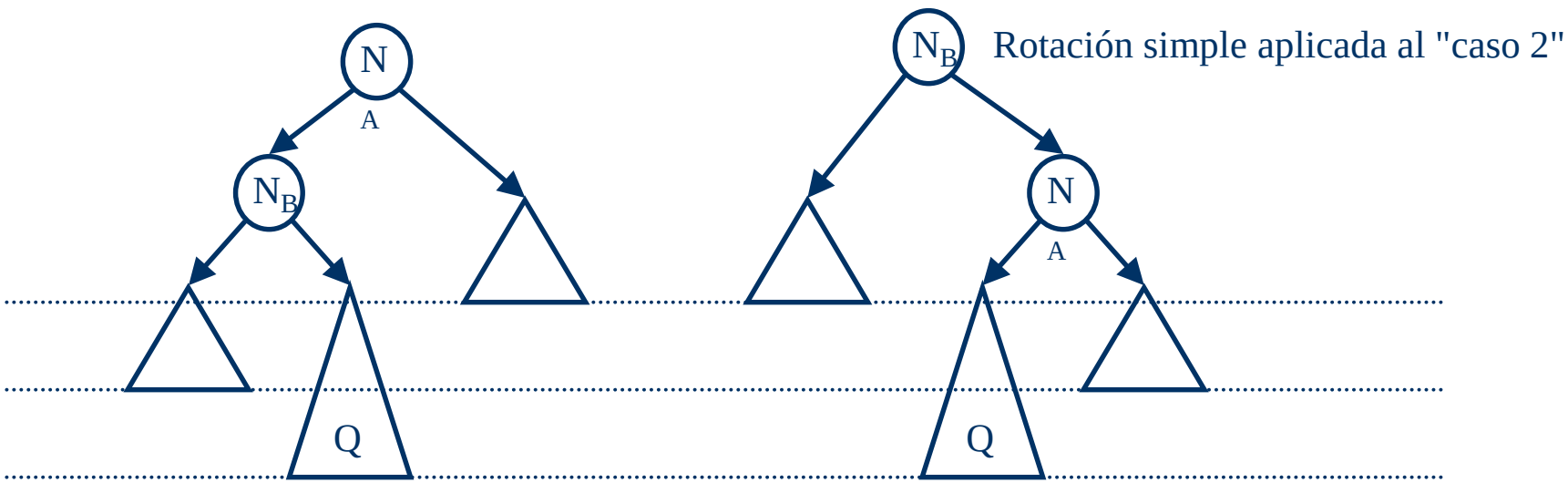
Árboles AVL. Rotación simple

- Se consiguen subárboles con la misma altura
  - ABB, AVL con diferencia de alturas = 0
- El nuevo árbol tiene la altura del original
  - Se ha reequilibrado completamente el árbol



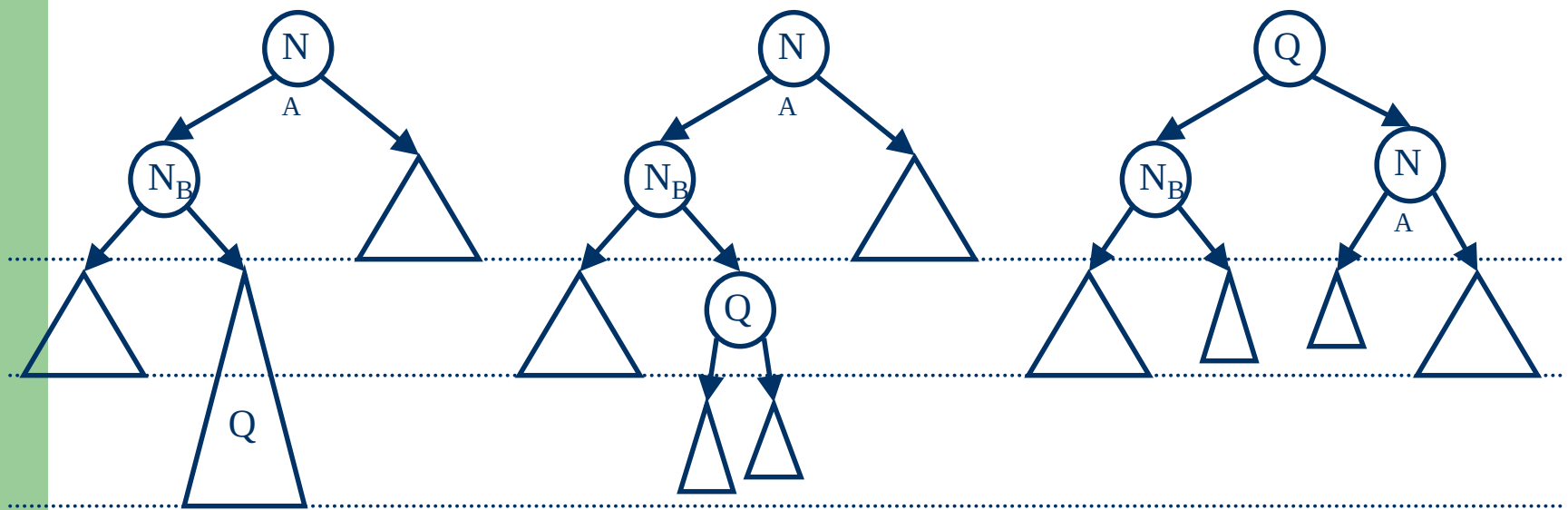
Árboles AVL. Rotación doble

- Rotación simple: No funciona en los casos 2 y 3
  - Q es demasiado profundo
  - Q al menos tiene una raíz
    - ...y dos subárboles, vacíos o con elementos



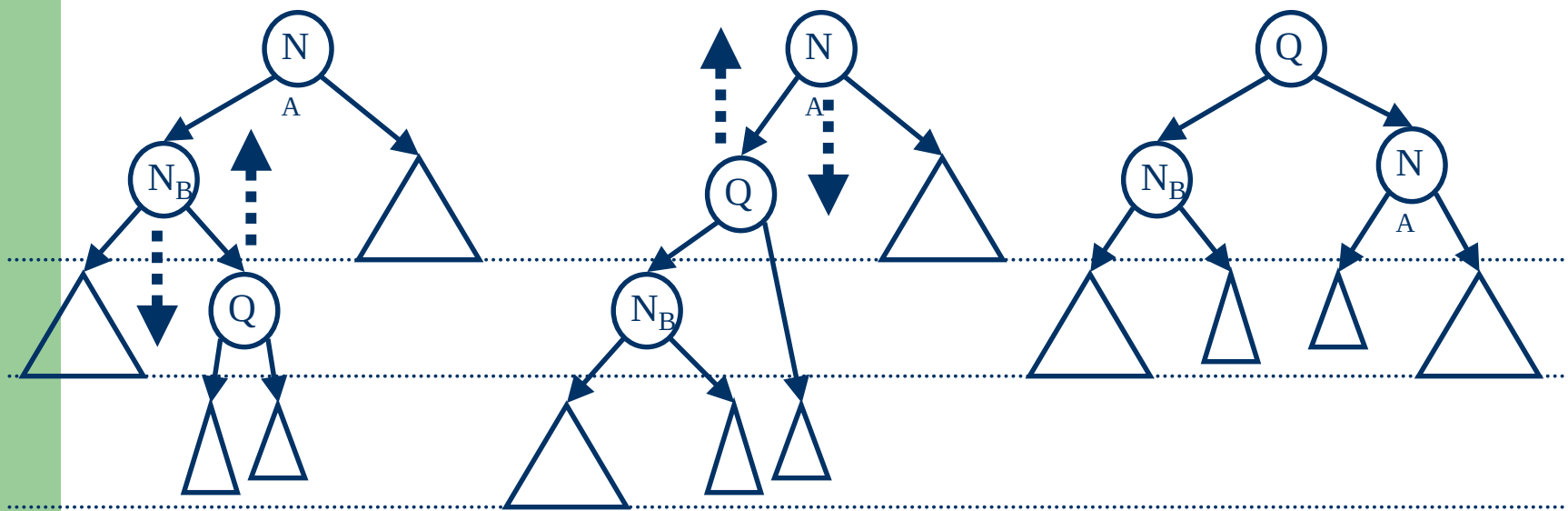
Árboles AVL. Rotación doble

- Se "eleva" el nodo Q como nueva raíz
- El árbol vuelve a tener la altura original
  - Como antes de insertar
  - Se reequilibra el árbol por completo



Árboles AVL. Rotación doble

- Rotación doble: Son dos rotaciones simples
  - Rotar Q y N_B
  - Rotar Q y N_A



Árboles AVL. Algoritmo

- Los nodos tienen un Factor de Balance (FB) que está entre –1 y 1.
  - FB = 0 alturas de subárboles iguales.
  - FB =1 subárbol derecho más grande que izquierdo.
  - FB = -1 subárbol izquierdo más grande que derecho
- Para realizar la inserción, se realiza igual que en un árbol binario y después se verifica el balanceo.

Árboles AVL. Algoritmo

- El mejor de los casos es que la inserción no realiza un desbalanceo, sólo hay que actualizar los FB de todos los ancestros.
- El otro caso es cuando hay que rebalancear, y se basa en un nodo pivote que es aquel que tiene un FB distinto de cero y es el más cercano de los ancestros del nodo recién insertado.

Árboles AVL. Algoritmo

1. Inserte el nodo.
2. Busque el nodo pivote. Coloque los apuntadores P1, P2, P3 y P4 donde:
   - P1 = apuntador al nodo padre del nodo pivote.
   - P2 = apuntador al nodo pivote.
   - P3 = apuntador al nodo hijo del nodo pivote, que es la raíz del subárbol más grande.
   - P4 = apuntador al nodo hijo del nodo apuntado por P3, que sigue en la ruta de búsqueda del nuevo nodo.
3. Si no existe el nodo pivote, entonces modificar FB de todos los ancestros.

Árboles AVL. Algoritmo

Si el nuevo nodo se insertó en el subárbol más pequeño, modificar los FB desde el pivote hasta el nuevo.

Si no, verificar el tipo de rotación.

Si es rotación simple: Modificar apuntadores y FB (rutina rotación simple)

Si no, modificar apuntadores y FB (rutina rotación doble)

4. Fin

Árboles AVL. Algoritmo

- Rotación Simple.

1. Si P1 no apunta a vacío

   Si la información del nuevo nodo es menor que la información apuntada por P1.

   Hijo izquierdo de P1 = P3

   si no

   Hijo derecho de P1 = P3

   si no

   P3 es la nueva raíz del árbol.

Árboles AVL. Algoritmo

2. Si la información del nuevo nodo es menor que la información apuntada por P2
   - Hijo izquierdo de P2 = hijo derecho de P3
   - Hijo derecho de P3 = P2
   - Modificar el FB desde el hijo izquierdo de P3 hasta el padre del nuevo nodo

   Si no
   - Hijo derecho de P2 = hijo izquierdo de P3
   - Hijo izquierdo de P3 = P2
   - Modificar el FB desde el hijo derecho de P3 hasta el padre del nuevo nodo
3. El FB del nodo señalado por P2 = 0

Árboles AVL. Algoritmo

- Rotación doble

1. Si P1 no apunta a vacío

   Si la información del nuevo nodo es menor que la información apuntada por P1.

   Hijo izquierdo de P1 = P4

   si no

   Hijo derecho de P1 = P4

   si no

   P4 es la nueva raíz del árbol.

Árboles AVL. Algoritmo

2. Si la información del nuevo nodo es menor que la información apuntada por P2

    Hijo derecho de P3 = hijo izquierdo de P4

    Hijo izquierdo de P2 = hijo derecho de P4

    Hijo izquierdo de P4 = P3

    Hijo derecho de P4 = P2. Seguir en 3.

  Si no

    Hijo izquierdo de P3 = hijo derecho de P4

    Hijo derecho de P2 = hijo izquierdo de P4

    Hijo derecho de P4 = P3

    Hijo izquierdo de P4 = P2. Seguir en 4.

Árboles AVL. Algoritmo

3. Si la información del nuevo nodo es menor que la información de P4:
   - Modificar FB desde el hijo derecho de P3 hasta le padre del nuevo nodo.
   - Modificar FB del nodo señalado por P2 (ahora vale +1)

   Si no
   - Si la información del nuevo nodo es mayor a la información de P4:
     - Modificar FB desde el hijo izquierdo de P2 hasta el padre del nuevo nodo.
     - Modificar FB del nodo señalado por P3 (ahora vale -1)
   - Modificar FB del nodo señalado por P2 (ahora vale 0)

Árboles AVL. Algoritmo

4. Si la información del nuevo nodo es mayor que la información de P4:
   - Modificar FB desde el hijo izquierdo de P3 hasta le padre del nuevo nodo.
   - Modificar FB del nodo señalado por P2 (ahora vale -1)

   Si no
   - Si la información del nuevo nodo es menor que la información de P4:
     - Modificar FB desde el hijo derecho de P2 hasta el padre del nuevo nodo.
     - Modificar FB del nodo señalado por P3 (ahora vale +1)

   Modificar FB del nodo señalado por P2 (ahora vale 0)

Árboles AVL. Inconvenientes

- Inserción de un elemento en árbol AVL
  - Se inserta en un subárbol
  - Si no cambia la altura: OK
  - Si aparece algún desequilibrio:
    - Solucionar con rotaciones
- Problema:
  - Cálculo de alturas. ¿Recalcular cuando se necesitan?
  - ¿Almacenar en los nodos y mantener actualizada?

Otros árboles equilibrados

- Otros esquemas de árboles equilibrados:
  - Splay Trees
  - Red-Black Trees
  - AA-Trees
  - B-Trees
    - Árboles-B (Bayer, 1970)
    - Interesantes para manejo de datos en disco
- Problema común:
  - Reorganización del árbol tras insertar/eliminar

Árboles-B

- Optimizados para manejo de datos en disco
- Objetivo:
  - Minimizar el número de accesos a disco
- Árbol-B de orden N: Árbol N-ario
  - Es un árbol equilibrado
  - Con N claves en cada nodo
  - Coste de acceso (profundidad): $\log_N N$

Árboles-B. Propiedades

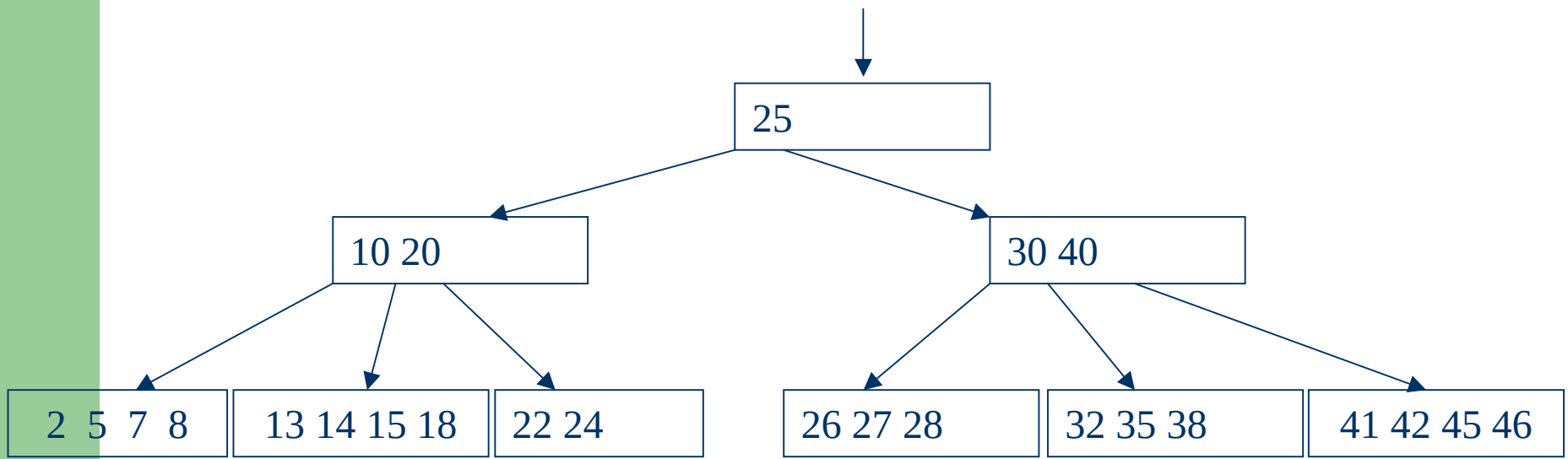
- Propiedades (aunque hay muchas variantes):
  - Cada página contiene a lo sumo 2N elementos (llaves).
  - Cada página, excepto la de la raíz, contiene N elementos por lo menos.
  - Cada página es una página de hoja, o sea que no tiene descendientes o tiene M+1 descendientes, donde M es el número de llaves en esa página.
  - Todas las páginas de hoja aparecen al mismo nivel.

Árboles-B

- Inserción:
  - Añadir el dato a su hoja. Reorganizar la hoja.
  - Si se llena la hoja:
    - Dos hojas con $\lceil L/2 \rceil$ elementos. Actualizar el padre
  - Si se llena el padre:
    - Partir en dos; actualizar nodo superior
    - Puede exigir una propagación hasta la raíz.
- Borrado
  - Fusión de hojas si no alcanza el mínimo de elementos.
  - El padre pierde hijos. Eliminación de nodos...

Árboles-B. Ejemplo

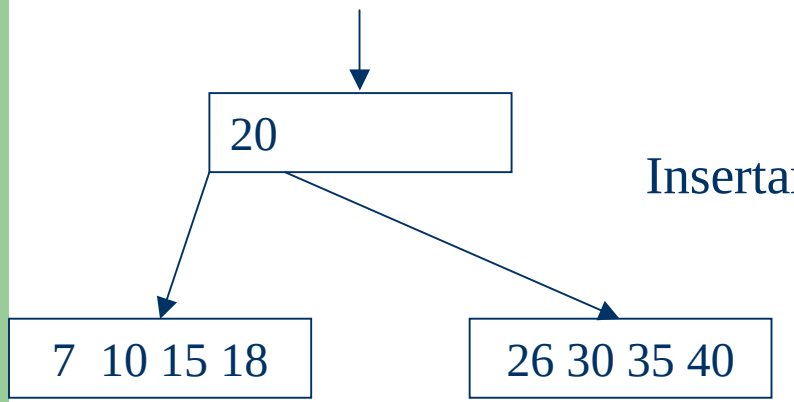
Árbol de orden 2 con 3 niveles.



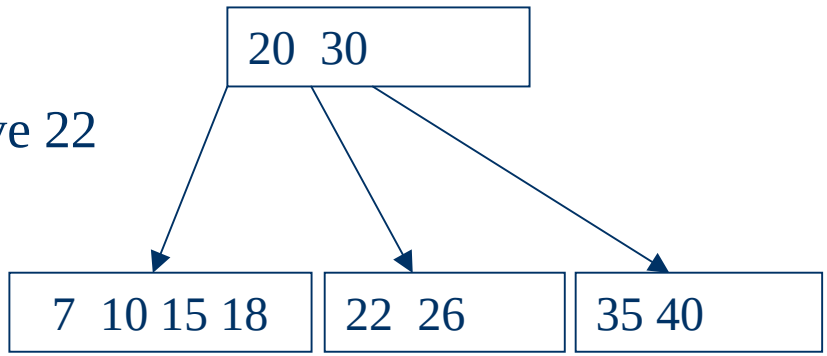
Árboles-B. Inserción

- La inserción en un árbol B es relativamente sencilla.
  - Si hay que insertar un elemento en una página con M<2N elementos, el proceso de inserción queda limitado a esa página.
  - En una página llena, se debe realizar la asignación de páginas nuevas.
  - En casos extremos, la propagación se lleva a la raíz, por lo tanto es cuando el árbol B puede crecer.

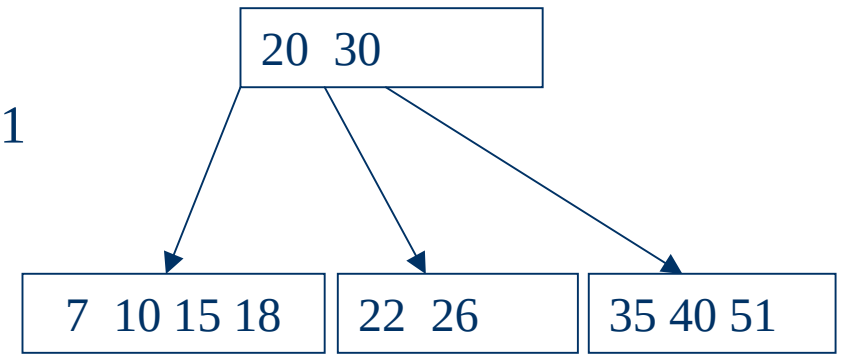
Árboles-B. Inserción



Insertar llave 22



Insertar llave 51



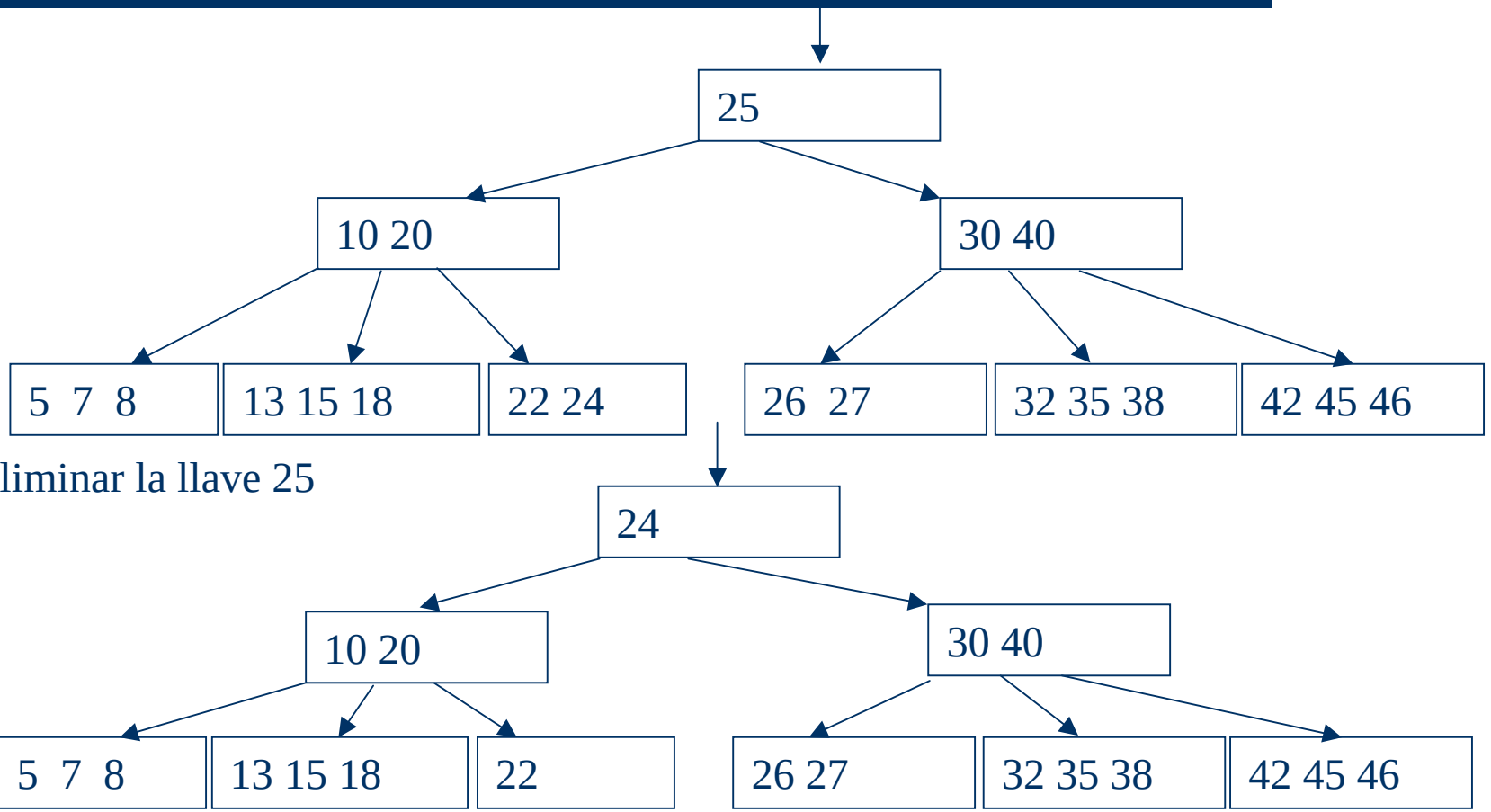
Árboles-B

- Los árboles B:
  - Crecen de las hojas hacia la raíz.
  - Son recursivos.
  - La búsqueda de elementos se realiza como en árboles binario, solo hay que modificar la búsqueda sobre un arreglo.
  - Son balanceados.
  - Cada página tiene entre N y 2N elementos.

Árboles-B. Eliminación

- La eliminación de elementos en un árbol B es en teoría sencilla, pero se complica en sus detalles. Se pueden distinguir dos circunstancias:
  1. El elemento que debe suprimirse se halla en una página de hoja, entonces el algoritmos de eliminación es sencillo.
  2. El elemento no se encuentra en una página de hoja; hay que sustituirlo por uno de los dos elementos lexicográficamente contiguos, que resultan estar en las páginas de hojas.

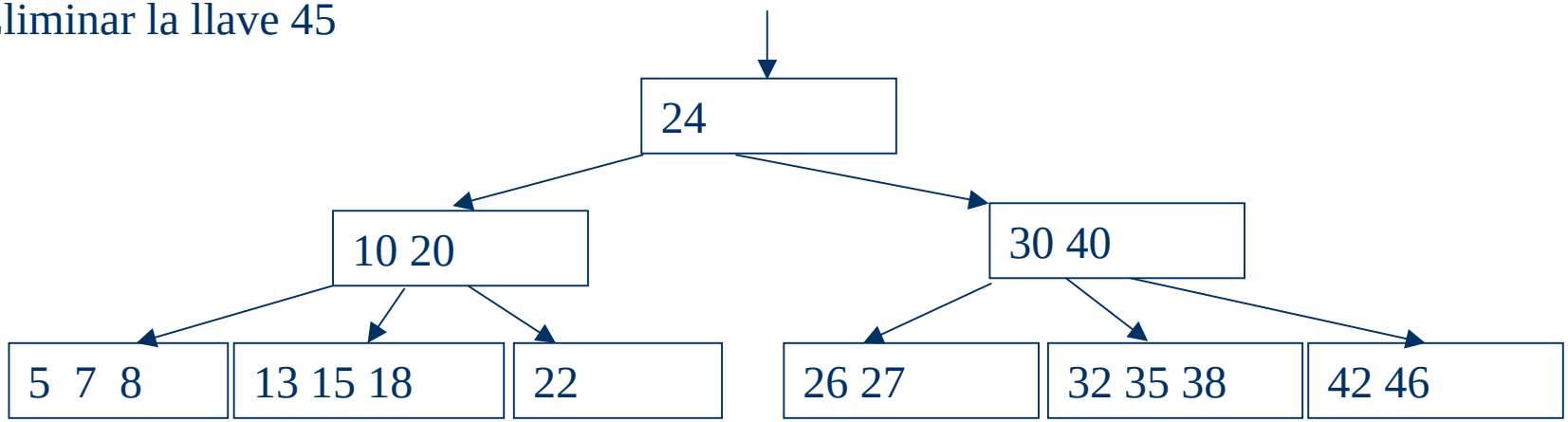
Árboles-B. Eliminación



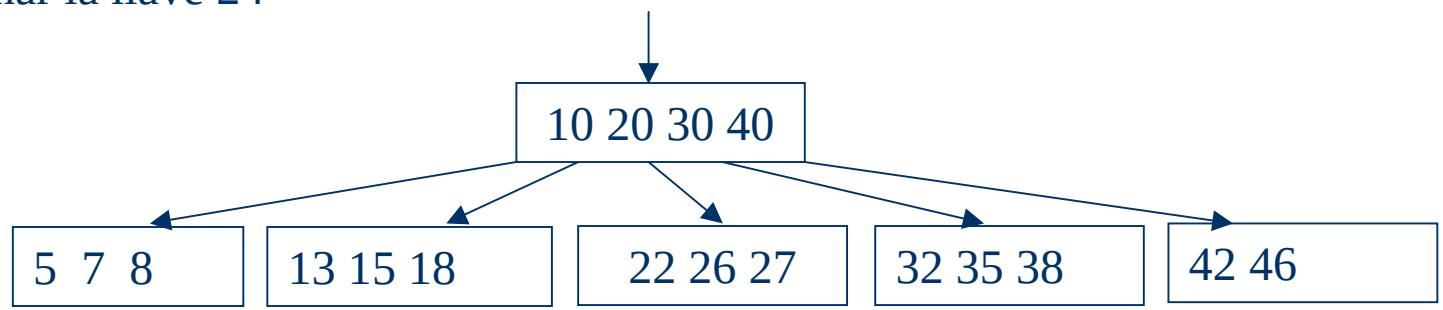
Eliminar la llave 25

Árboles-B. Eliminación

Eliminar la llave 45

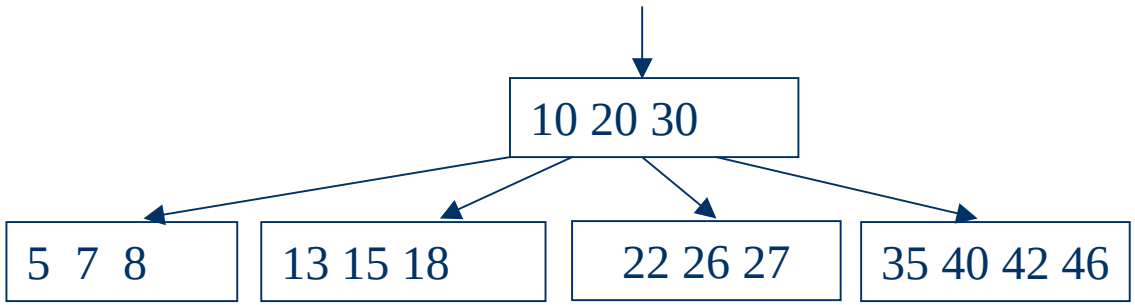


Eliminar la llave 24



Árboles-B. Eliminación

Eliminar las llaves 32 y 38



Eliminar las llaves 8 y 46