En la clase pasada definimos el **balance AVL** para un ABB

Las alturas de sus (árboles) hijos difieren a lo más en 1 entre ellas

Cada hijo a su vez está **AVL-balanceado**

(Recordemos que este balance implica mantener un dato adicional en cada nodo: un valor –1, 0 o +1)

¿Será posible tener otra noción de balance?

Árboles balanceados de otra manera

Queremos un árbol de búsqueda en que el balance esté dado porque todas las hojas están a la misma profundidad

… y que esa profundidad sea $O(\log n)$, si el árbol almacena n claves

¿Es esto posible con árboles binarios? ¿Y ternarios?

¿Será posible combinarlos?

En un **árbol** (de búsqueda) **2-3** hay dos tipos de nodos

Nodo 2, con una clave y, si no es una hoja, exactamente 2 hijos

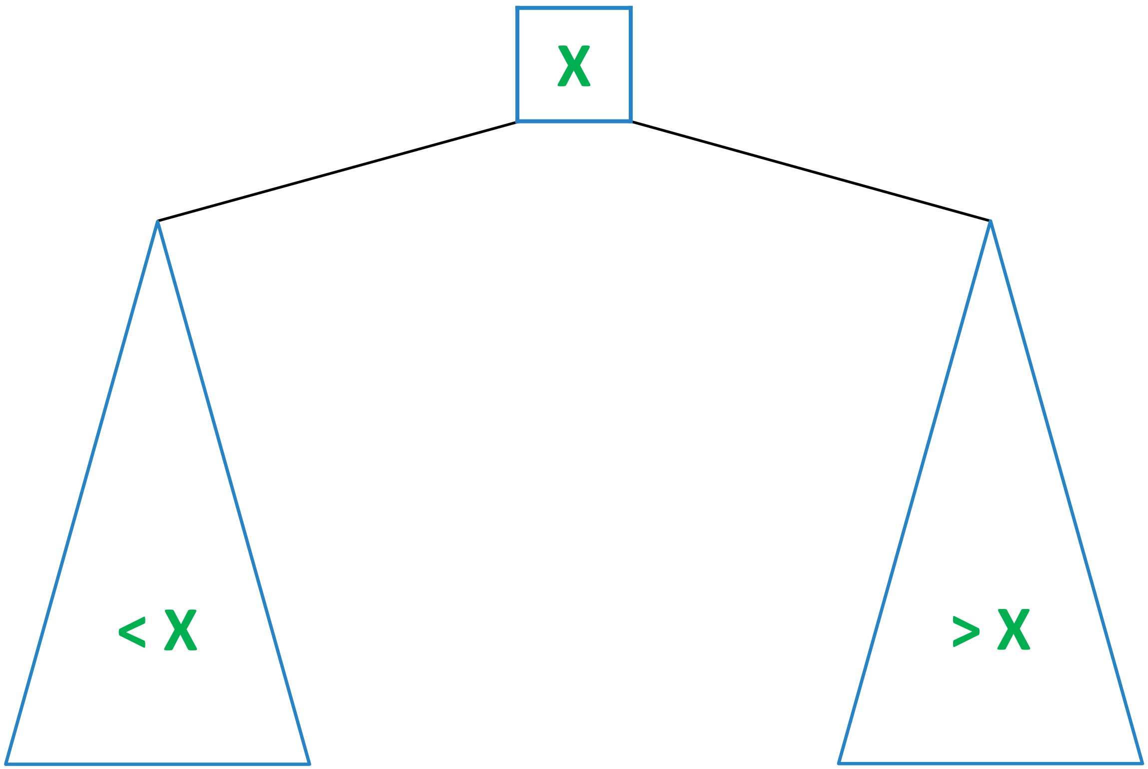
Nodo 3, con dos claves distintas y ordenadas y, si no es una hoja, exactamente 3 hijos

Como veremos, esto permite que todas las hojas estén a la misma profundidad

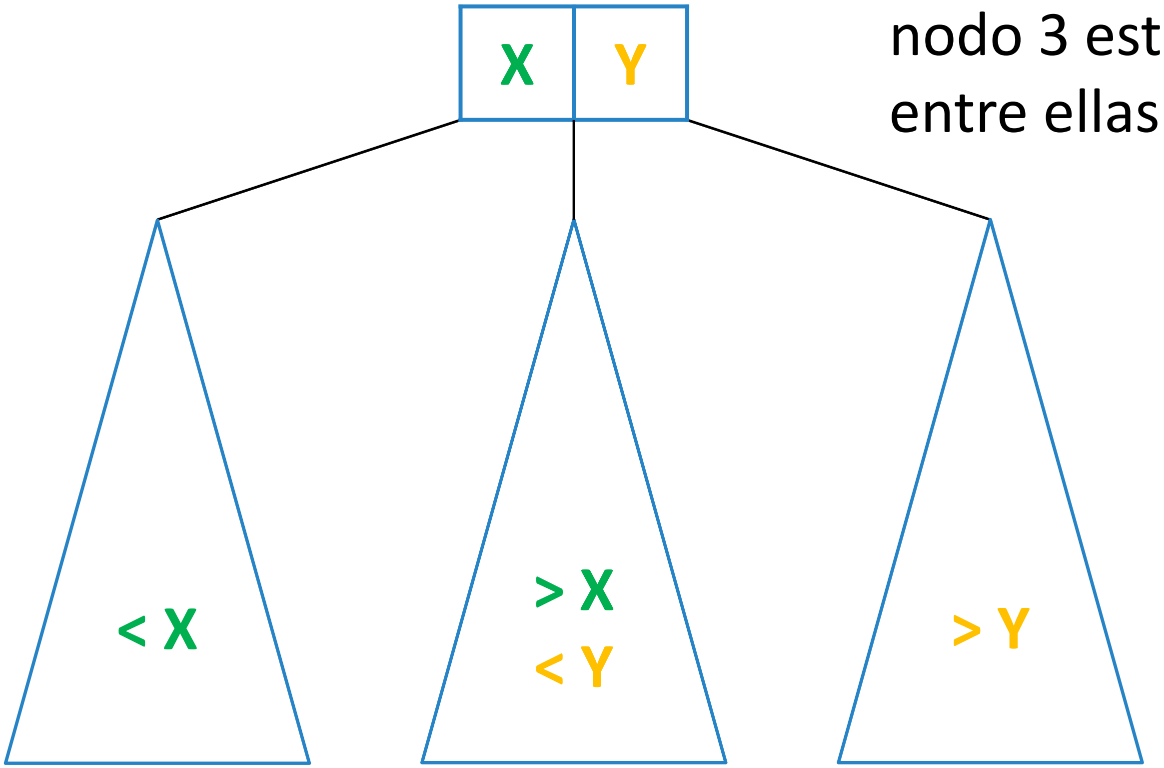
… y que esa profundidad sea $\mathrm{O}(\log n)$, si el árbol almacena n claves:

- en un árbol 2-3, número de nodos ≤ número de claves almacenadas

Nodo 2
(los árboles 2-3 son árboles de búsqueda)

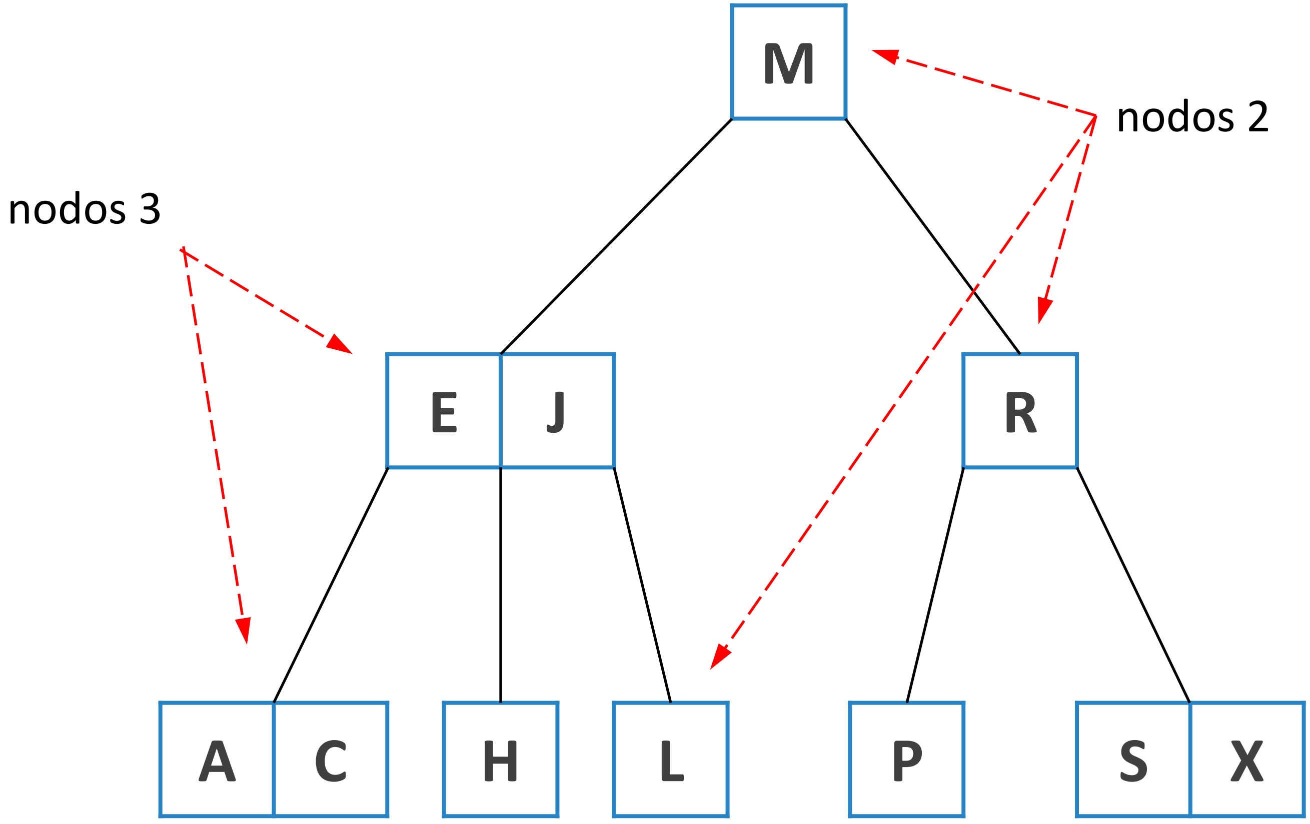


Nodo 3
(los árboles 2-3 son árboles de búsqueda)

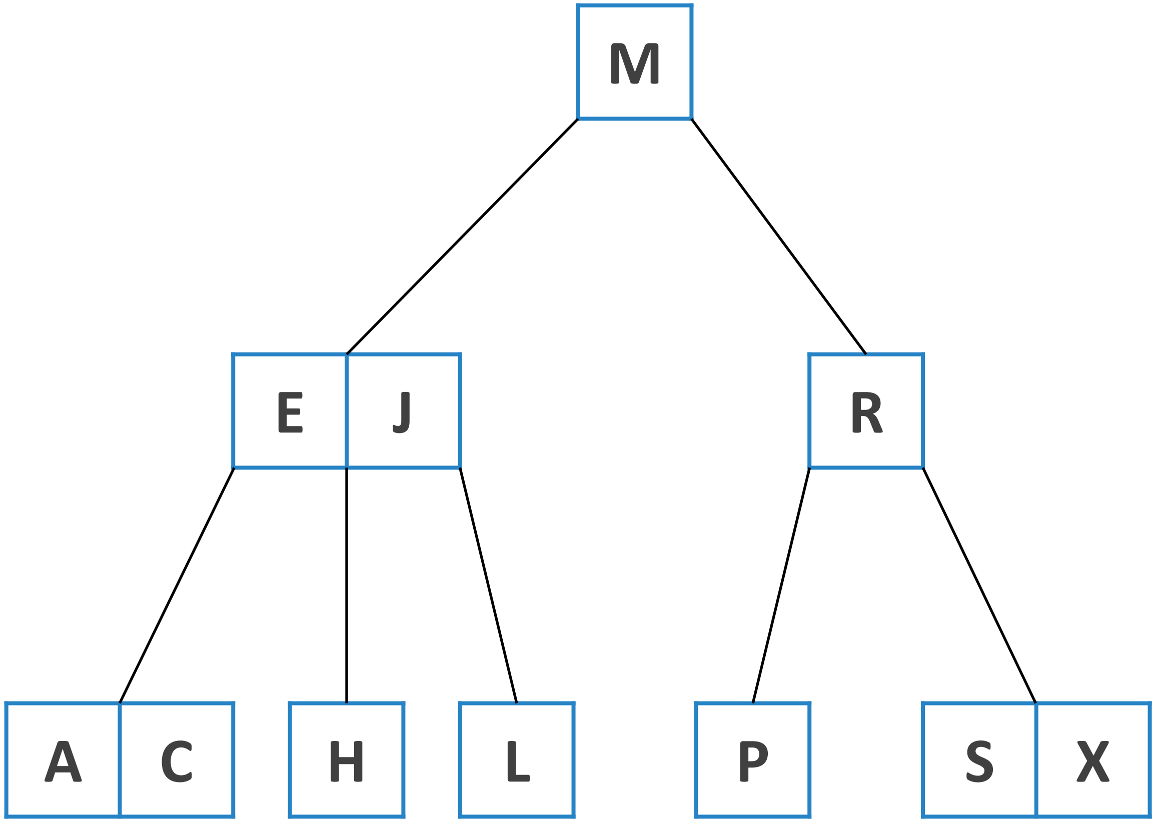


X < Y : las dos claves de un nodo 3 están ordenadas entre ellas

Ejemplo de árbol 2-3
(notar que las claves están ordenadas)

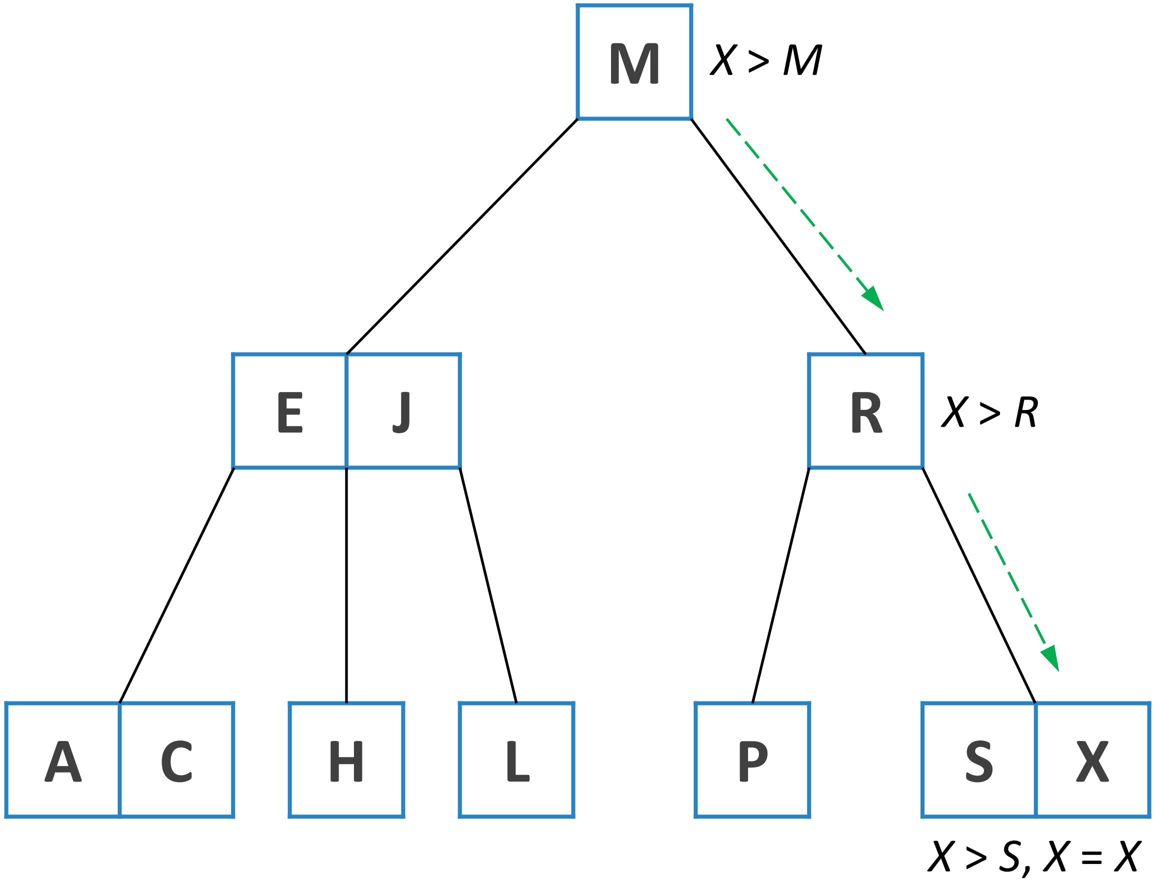


¿Cómo buscamos una clave en un árbol 2-3

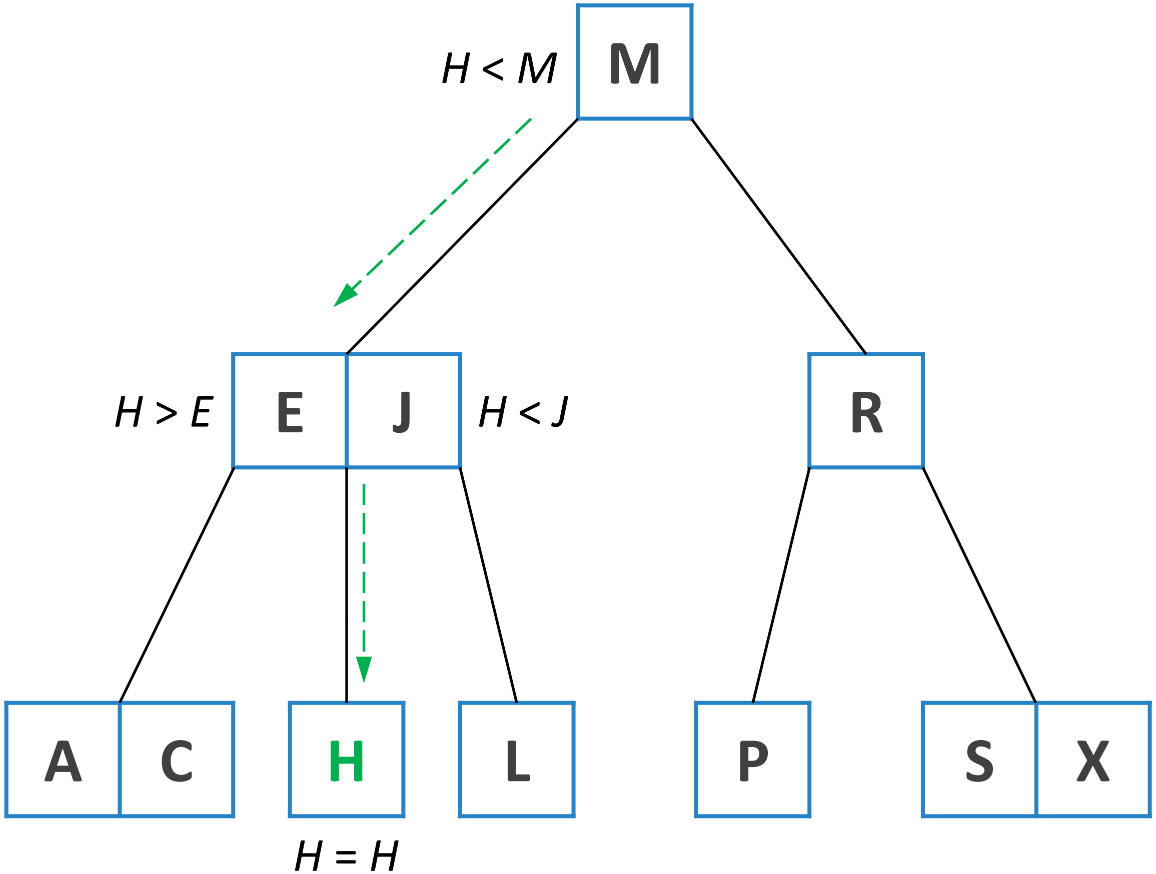


Aprovechamos el hecho de que el árbol está ordenado

P.ej., busquemos la clave *X*



P.ej., busquemos la clave *H*



Inserción en un árbol 2-3

Al insertar nuevas claves al árbol, podría cambiar su altura

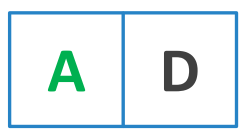
Queremos mantener todas las hojas a igual profundidad

¿Cómo podemos insertar las claves para que se cumpla esto?

P.ej., insertemos las claves *D*, *A*, *C*, *E*, *N*, *F*, *H* en un árbol 2-3 inicialmente vacío

Se inserta la clave *D* en un (nuevo) nodo 2, que, además, pasa a ser la raíz del árbol

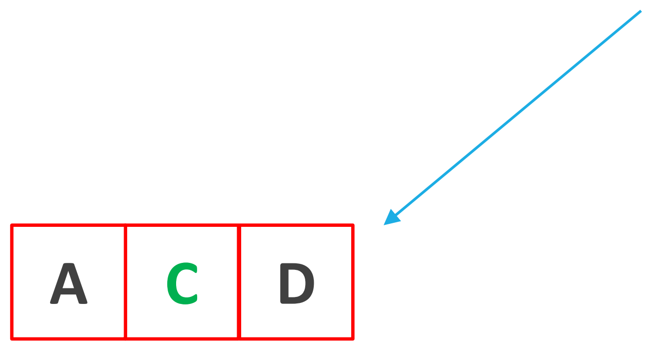
… insertemos las claves *A*, *C*, *E*, *N*, *F*, *H*



La clave *A* se inserta ordenadamente
(y el nodo cambia de tipo 2 a tipo 3)

... insertemos las claves *C*, *E*, *N*, *F*, *H*

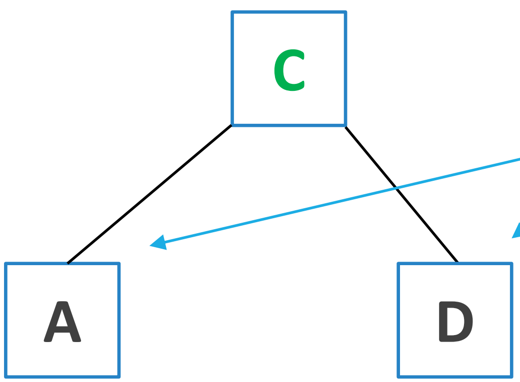
nodo temporal con
3 claves: no válido



Este nodo ya no es un nodo 2 ni un nodo 3,
por lo que debemos hacer algo al respecto

… insertemos las claves *C*, *E*, *N*, *F*, *H*

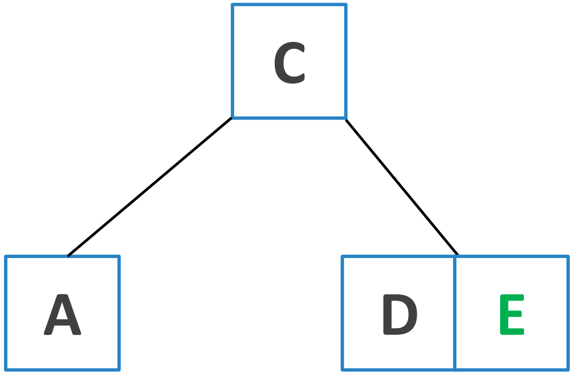
como en este caso la clave del medio no tiene a donde subir, creamos un nuevo nodo para almacenarla, que pasa a ser la **nueva raíz** del árbol



las otras dos claves se separan en sendos nodos 2

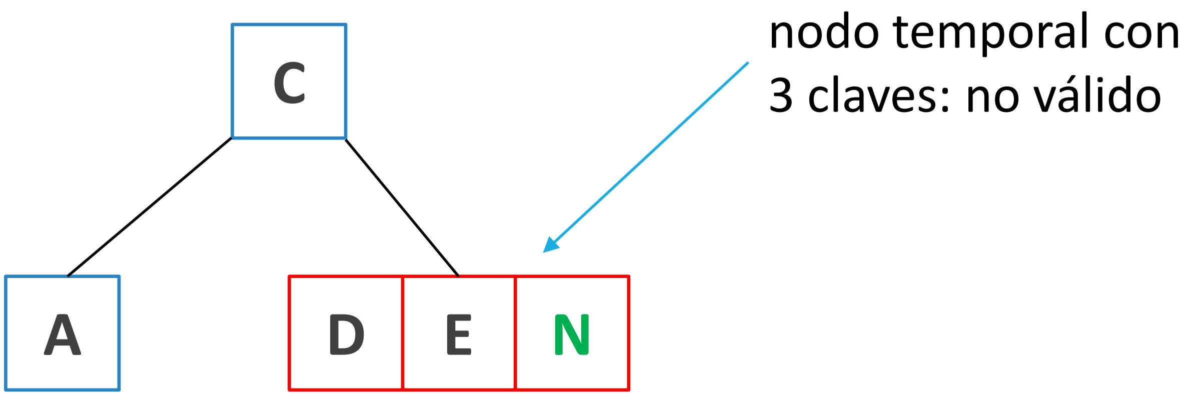
Llamamos ***split*** a esta operación, en que sube la clave del medio (y ahora sólo tenemos nodos 2)

… insertemos las claves *E*, *N*, *F*, *H*



La inserción siempre se hace en las hojas (que pueden cambiar válidamente de nodo 2 a nodo 3, como en este caso)

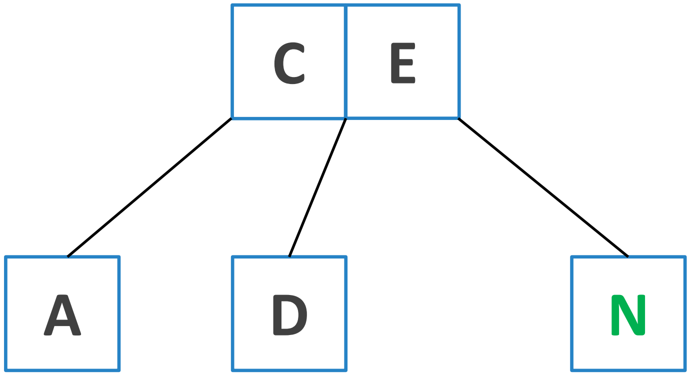
… insertemos las claves *N*, *F*, *H*



Nuevamente tenemos un nodo con 3 claves, …

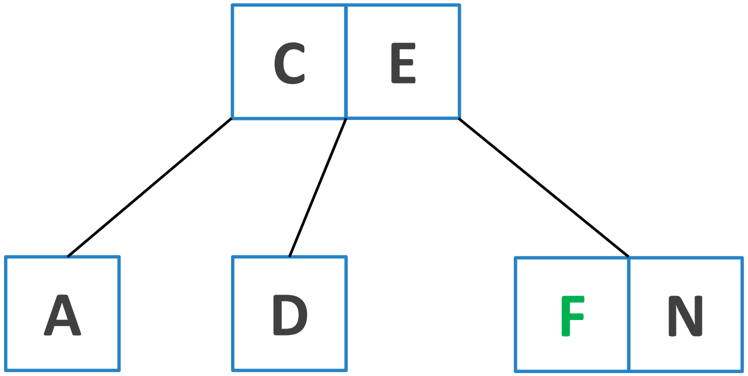
… insertemos las claves *N*, *F*, *H*

en este caso, la clave del medio, *E*, sube a un nodo existente, que cambia de nodo 2 a nodo 3

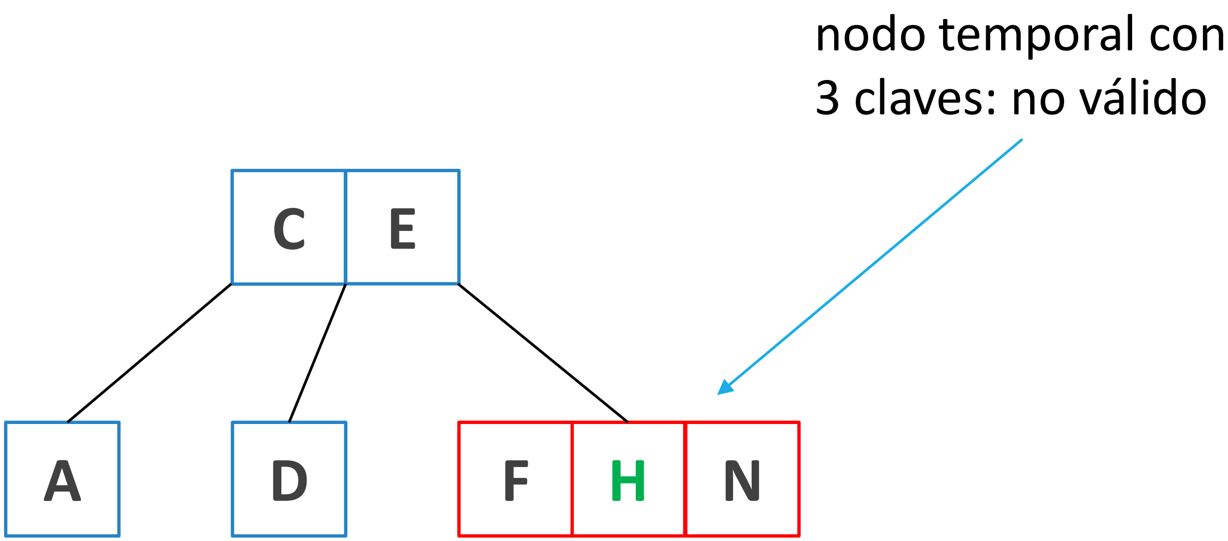


… hay que hacer ***split***: la clave del medio sube y se inserta ordenadamente en el nodo superior (que cambia de nodo 2 a nodo 3)

… insertemos las claves *F*, *H*



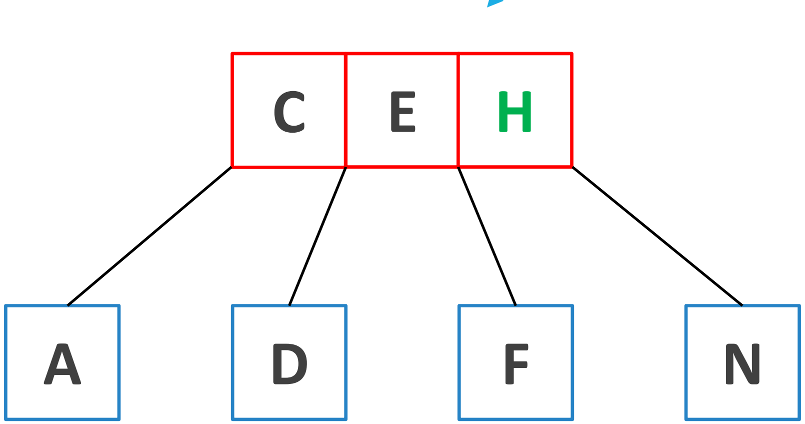
… finalmente, insertemos la clave *H*



De nuevo creamos un nodo con 3 claves: tenemos a hacer ***split***

… finalmente, insertemos la clave *H*

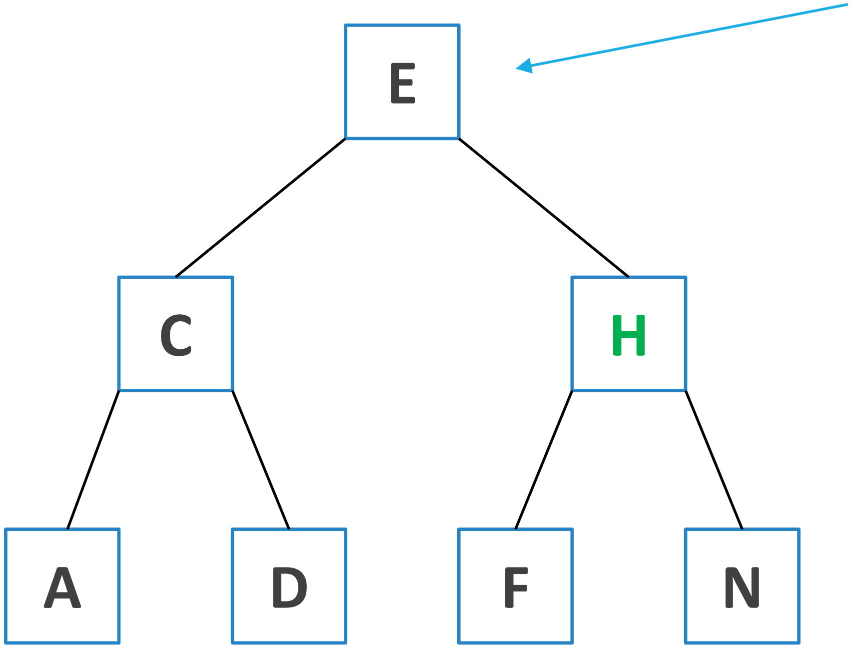
al subir la clave del medio, H, creamos otro nodo temporal con 3 claves, no válido



… y esto puede causar una reacción en cadena …

... finalmente, insertemos la clave *H*

de nuevo, la clave del medio no tiene a donde subir: creamos un nuevo nodo para almacenarla, que se constituye en la **nueva raíz** del árbol



Sólo cuando se hace ***split*** de la raíz (similar a la diap. #14), la altura del árbol aumenta en 1

Inserción en árboles 2-3: resumen

La inserción siempre se hace —inicialmente— en una hoja

Si un nodo está lleno (ya tiene dos claves) y debe recibir una tercera clave,

… entonces se hace subir la clave que habría quedado al medio —la clave mediana— al nodo padre

¡ El árbol sólo aumenta de altura cuando la raíz está llena y recibe una clave desde un hijo !

¿Cuál es la altura de un árbol 2-3 con *n* claves?

¿Cuál es el costo de una búsqueda en el árbol 2-3?

¿Cuál es el costo de una inserción en el árbol 2-3?

Altura de un árbol 2-3

El mejor caso es que todos los nodos sean nodos 3:

$$h = \log_3 n$$

El peor caso es que todos los nodos sean nodos 2:

$$h = \log_2 n$$

Por lo tanto,

$$h \in \Theta(\log n)$$

Costo de las operaciones

A diferencia de los árboles binarios, ahora podríamos tener que comparar más de una vez por nivel

Por lo tanto, el costo de buscar o insertar es $O(2h) = O(h)$