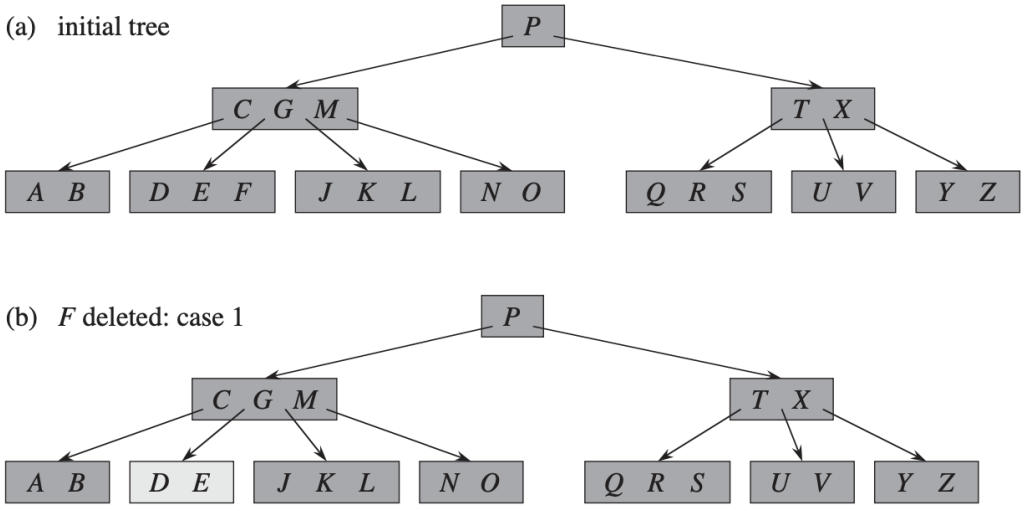
B-Trees

> ❗ Se mantiene la invariante de que cada vez que se descienda a un nodo en el recorrido para encontrar el key a borrar, el nodo tendrá más keys que el número mínimo.

▼ Case 1 (encontrado en hoja)

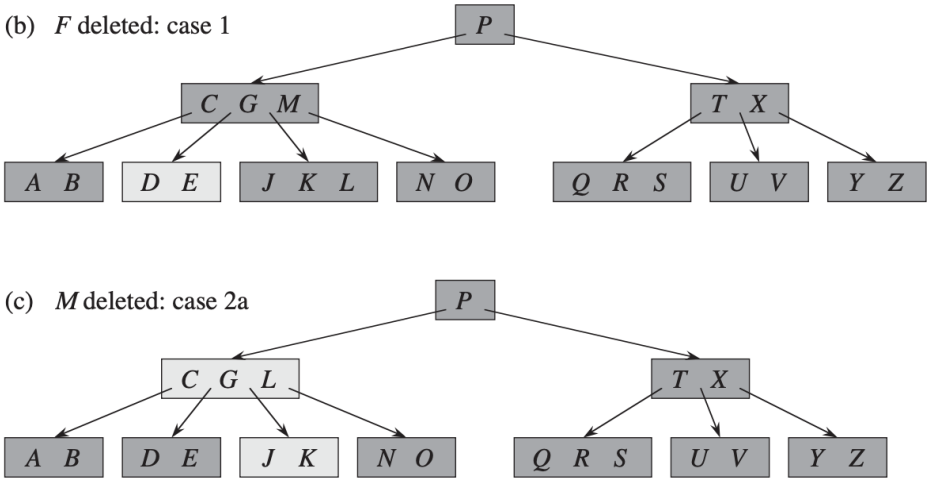
Si la key está en un nodo hoja, borrarlo del nodo hoja y reordenar los keys en el nodo como sea necesario.



▼ Case 2 (encontrado en interno)

Case 2A

Si el nodo X que tiene el predecesor del key tiene más keys que el mínimo número, entonces que X done un key mediante el padre al nodo actual.



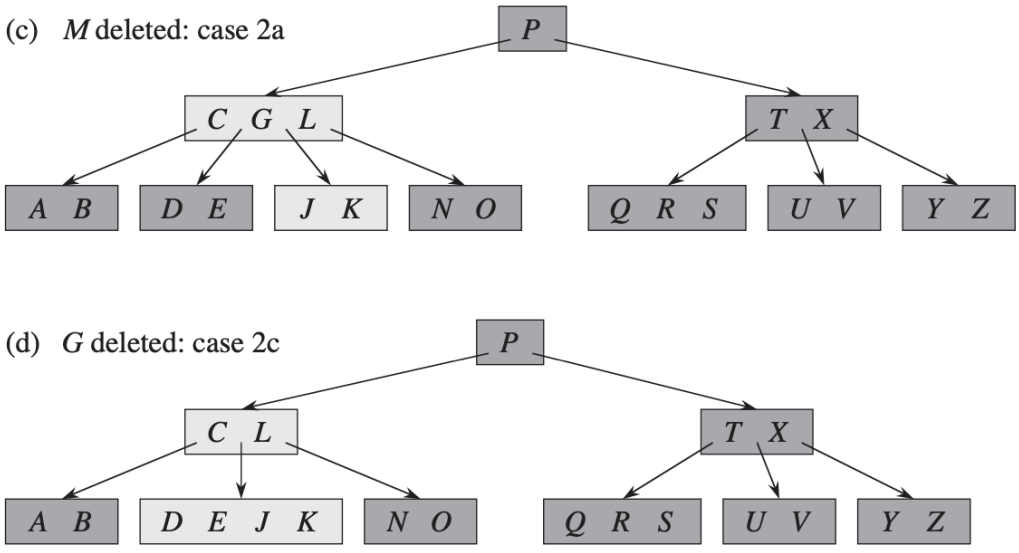
(b) es el árbol original, (c) es el árbol luego de borrar M.

Case 2B

Lo mismo que **Case 2A** pero con el sucesor del key que se quiere borrar.

Case 2C

Ni el predecesor ni el sucesor tienen más que el número mínimo de keys (o simplemente el nodo no tiene predecesores ni sucesores). En ese caso, el key baja y se hace merge entre X y Y. Luego, recursivamente se borra empezando desde ese nuevo nodo creado.



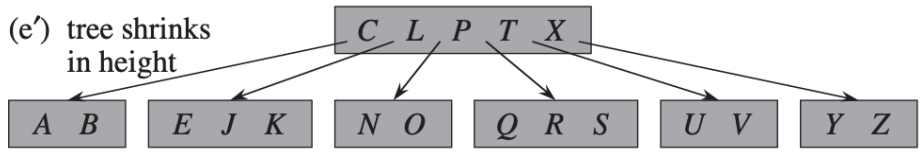
(c) es el árbol original, (d) es el árbol luego de borrar G.

▼ Case 3 (no encontrado)

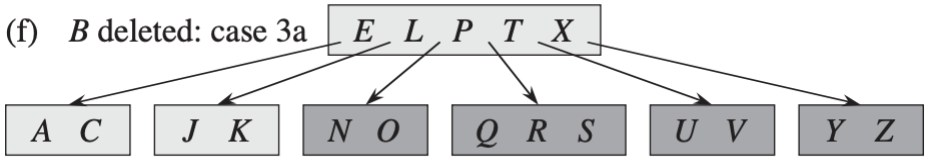
Este caso sirve para mantener la invariante anteriormente mencionada: que dado cualquier nodo visitado al buscar el key a borrar, se asegura que siempre tendrá más que el número mínimo de keys.

Case 3A

Toca hacer recursión por el nodo $x.c_i$, si tiene el número mínimo de keys y alguno de sus dos hermanos inmediatos tiene más que el número mínimo, entonces que uno de sus hermanos les done un key mediante el parent. Luego de la donación, se borra recursivamente en el nodo que recibió la donación, osea $x.c_i$.



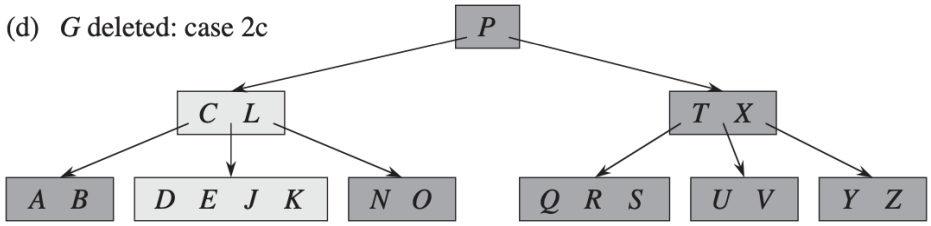
Árbol original. El nodo E J K es el que va a donar E al padre y el padre va a donar C al nodo necesitado.



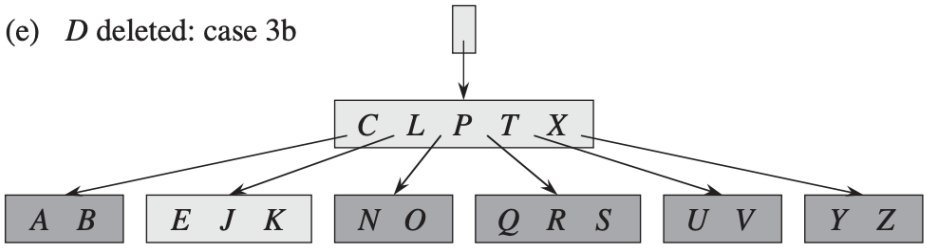
Árbol luego de borrar B.

Case 3B

Si ambos hermanos inmediatos de $x.c_i$ tienen el número mínimo de keys, toca hacer merge entre cualquiera de los hermanos inmediatos y $x.c_i$, lo cual implica que x baje un key para que se una al nodo nuevo.



Árbol original. Notar que la búsqueda empieza en P y continua en C L, pero C L tiene el número mínimo de keys, entonces toca hacer los pasos necesarios para que se mantenga la invariante que simplifica bastante el desarrollo del delete y aumenta su eficiencia.



Árbol luego de borrar D.