# Materia C2 Estructuras de Datos

## Clase 5 / Lunes 25 de marzo:

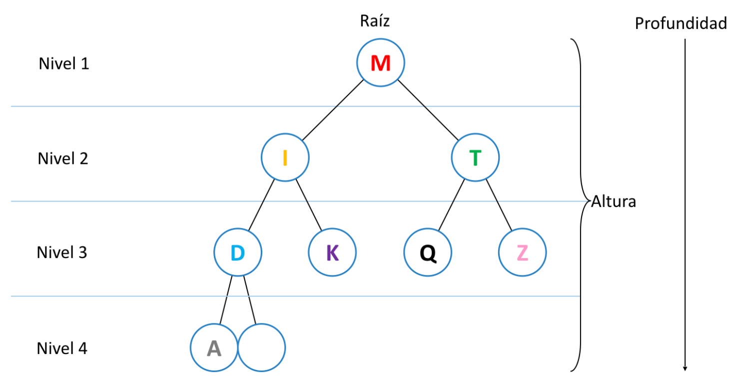
**El conteo de votos**

Como la cantidad de candidatos será variable, un arreglo no nos sirve. Para este problema usamos una lista ligada. En esta estructura como tal, no sabemos donde están almacenados los datos, por lo tanto cada vez tenemos que recorrer toda la lista.

**El diccionario:** permite asociar un **valor** a una **clave,** o actualizarlo. Y obtener dicho valor asociado a la clave. También permite eliminar claves.

Los **árboles binarios** guardan un puntero con la posición del dato de al medio. En este caso, la letra M es la que está en la mitad. Notar la relación aquí con binary search.

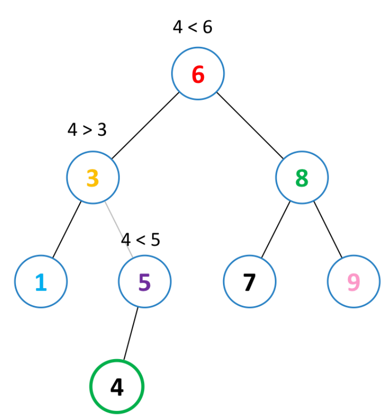
**Árbol binario de búsqueda (abb):** son árboles binarios donde cada partición es otro abb.




— insertar código búsqueda en árboles binarios —

Cuando busco un dato que no existe, entonces se **inserta** en la posición que debería haber estado. En la siguiente imagen podemos ver como se inserta el número 4 después de no ser encontrado.

Para **eliminar** nodos hay que realizar más trabajo. Sobre todo cuando el nodo tiene 2 hijos. Para esto nos sirve encontrar el **sucesores** o el **antecesor.**



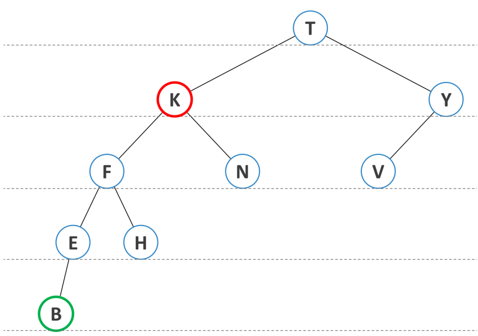
## Clase 6 / Miércoles 27 de marzo:

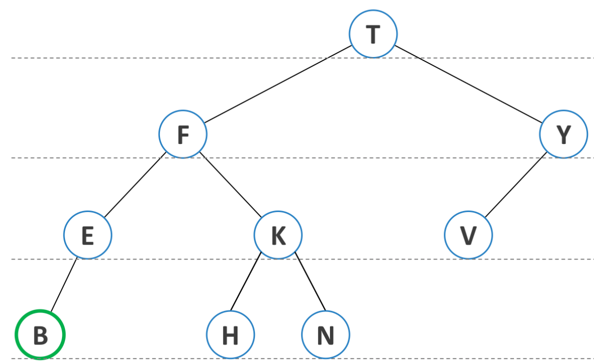
**Balancear árboles**

Existen varias formas para ordenar árboles. Las siguientes tres nos aseguran que la altura del árbol no será mayor que $O(\log n)$. Los árboles están balanceados, a su vez cada nodo está balanceado para abajo. Nos interesa que estén balanceados recursivamente.

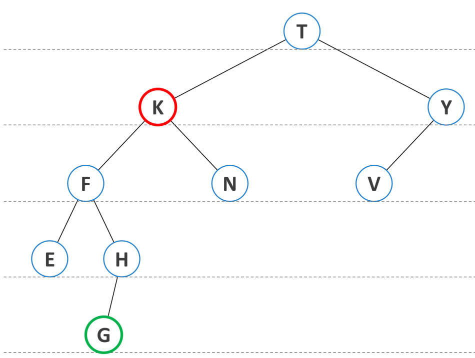
- **AVL:** Las alturas de sus hijos no difieren en más que 1 entre ellas. Cada hijo a su vez está avl-balanceado. Con cada inserción re-balanceamos. Para esto vamos mirando de abajo hacia arriba por la rama en la que llegamos. Cuando lleguemos a un nodo con diferencia de dos o más niveles lo arreglamos. Esto lo podemos ver con el "factor de balance", sólo puede ser -1, 0 o 1. Para calcularlo hacemos Izq - Der. En el peor caso vamos a necesitar hacer 2 rotaciones.

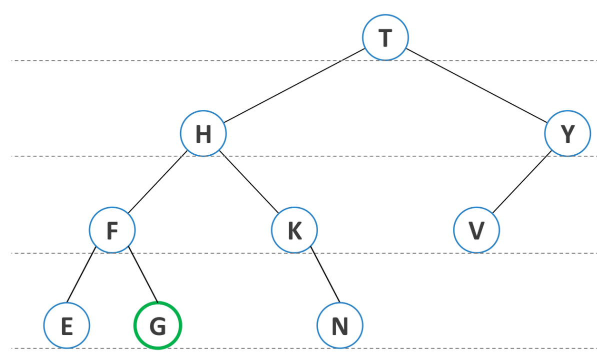
Rotación a la derecha en K-F



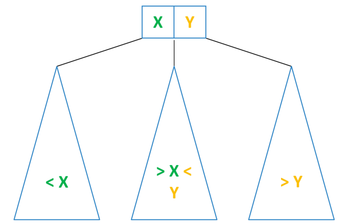

Ahora una doble rotación:

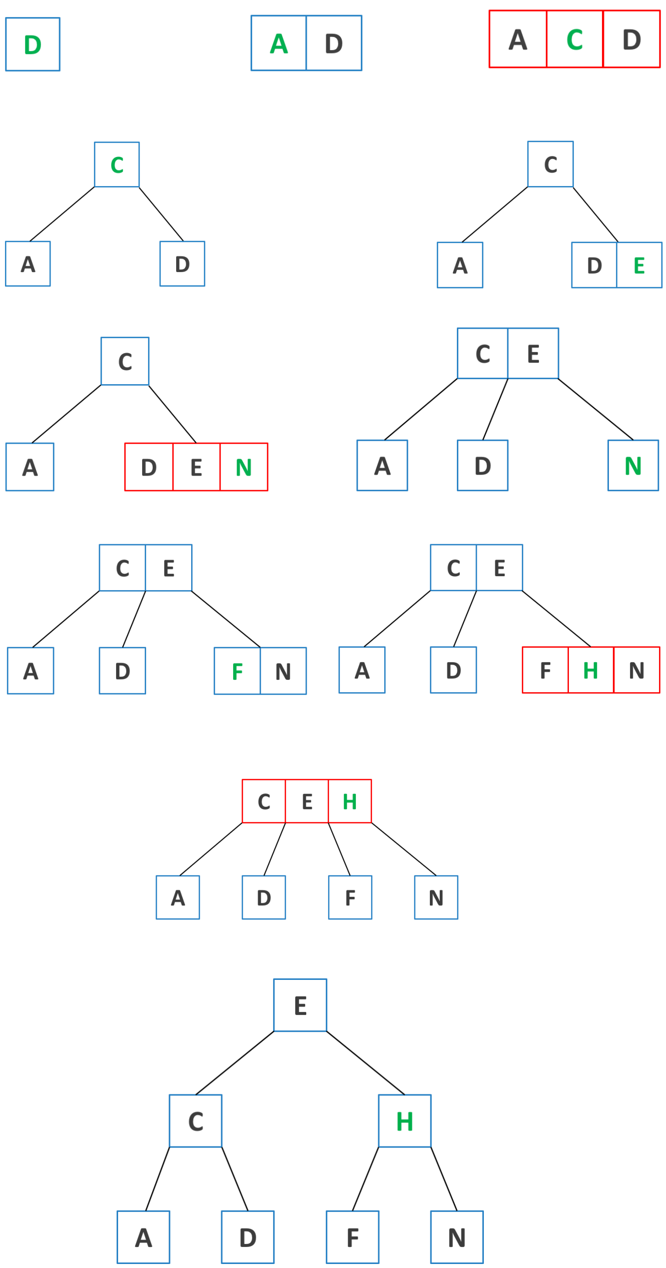


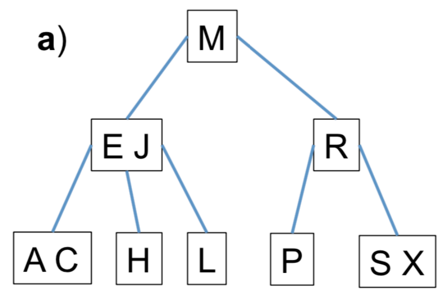
## Clase 7 / Lunes 1 de abril:

- **Árboles 2-3:** tiene dos tipos de nodos. por cada nivel hacemos como máximo 2 comparaciones. La altura varía entre $O(log_2 n)$ y $O(log_3 n)$, el peor caso son sólo nodos 2. Tienen mucho *overhead*, ya que deben cambiar mucho de nodo.
  - **Nodo 2:** una clave y 2 hijos, tal que ambos son nulos (hoja) o ambos son raíces de subárboles 2-3
  - **Nodo 3:** dos claves distintas y tres hijos, tal que tal que todos son nulos (hoja) o los tres son raíces de subárboles 2-3

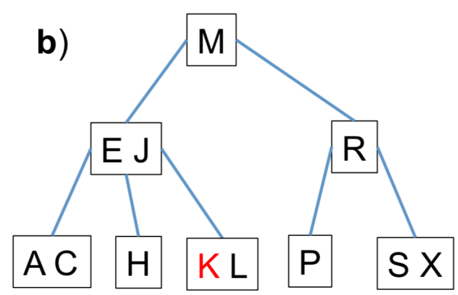


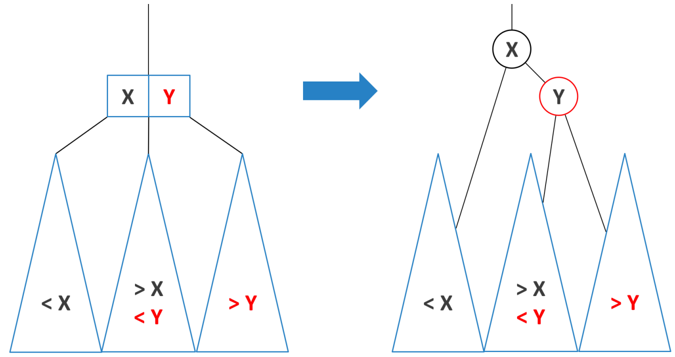
a)



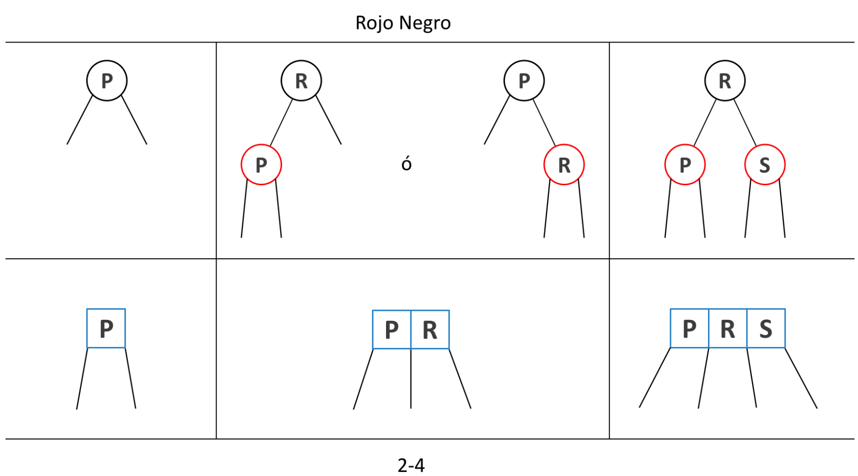

b)



- **Árboles rojo-negro:** Para solucionar el problema del *overhead,* los árboles 2-3 se pueden representar como árboles binarios. 4 Propiedades:
  - Cada nodo es rojo o **negro**
  - La raíz es **negra**
  - Si un nodo es rojo, sus hijos deben ser **negros**
  - La cantidad de nodos **negros** camino a cada hoja es siempre la misma[1]




No todos los árboles rojo-negro tienen un árbol 2-3 equivalente, pero si tienen un árbol 2-4 equivalente.




[1] Las hojas nulas se consideran como **negras**