UNIVERSIDAD DEL BÍO-BÍO

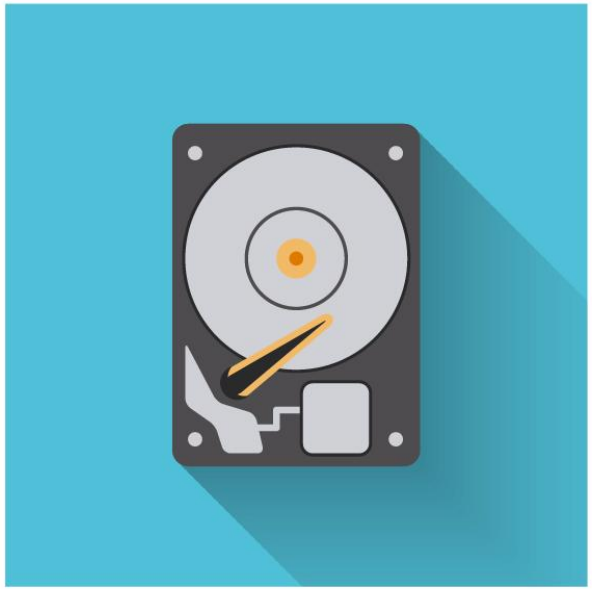
Estructura de Datos

Árbol B y sus variantes

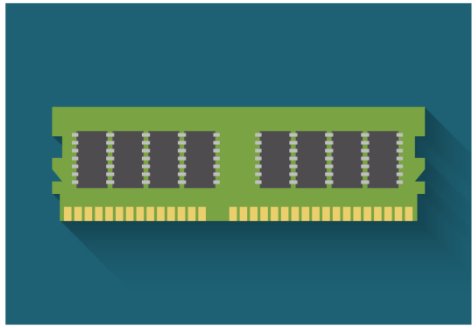
Introducción

Cuando se trabaja con grandes conjuntos de datos, a menudo no es posible o deseable mantener toda la estructura en el almacenamiento primario (RAM/Cache). Una solución podría ser leer directamente desde la memoria secundaría, pero se produce un nuevo problema.

Tiempo de Acceso al Disco es muy lento



Vs



Designed by macrovector / Freepik

En resumen

1. Los árboles B son árboles equilibrados que están optimizados para situaciones en las que parte o todo el árbol debe mantenerse en un almacenamiento secundario, como un disco magnético.

2. Los accesos a disco son costosos en tiempo, por lo que se debe evitar realizar muchas lecturas.

3. Es mejor que los arboles de búsqueda binaria cuando los datos son almacenados en memoria externa, pero los árboles B no son apropiados para almacenar datos en memoria principal.

En resumen

1. Los árboles B son árboles equilibrados que están optimizados para situaciones en las que parte o todo el árbol debe mantenerse en un almacenamiento secundario, como un disco magnético.

2. Los accesos a disco son costosos en tiempo, por lo que se debe evitar realizar muchas lecturas.

3. Es mejor que los arboles de búsqueda binaria cuando los datos son almacenados en memoria externa, pero los árboles B no son apropiados para almacenar datos en memoria principal.

En resumen

1. Ideal para búsqueda en base de datos ya que es más importante minimizar los intercambios de paginas que las comparaciones.
2. Son árboles 100% balanceados en su estructura, lo cual repercute en búsquedas eficientes y en accesos mínimos al disco.

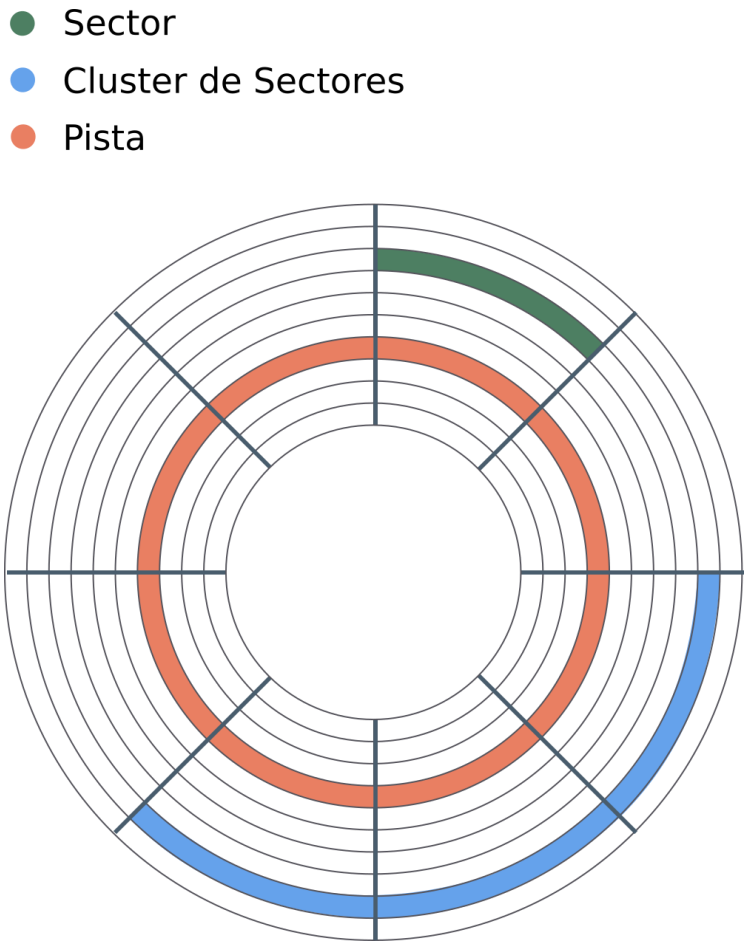
Designed by Photoroyalty / Freepik

Principio básico

Los datos en disco están almacenados en “trozos” (paginas, bloques, unidades de asignación) y el lector del disco duro lee o escribe un mínimo de una pagina cada vez. Para optimizar un algoritmo para acceder memoria externa debemos:

1. Minimizar los accesos a discos
2. Leer y escribir en múltiples tamaños de paginas
3. Tamaño de bloque varía, usualmente en potencia de 2.

Estructura física del contenido Disco Duro



1. **Pista:** Unidad de disco de camino circular en la superficie del disco magnético.

2. **Sector :** Las pistas a su vez están divididas en trozos de arco llamados sectores. En estos tramos es donde se almacenan los bloques de datos.

3. **Cluster:** También llamado unidad de asignación, es una agrupación de sectores. Cada archivo ocupará un determinado número de clusters, y ningún otro archivo podrá estar almacenado en un determinado cluster.

Ejemplo - Directorio telefónico

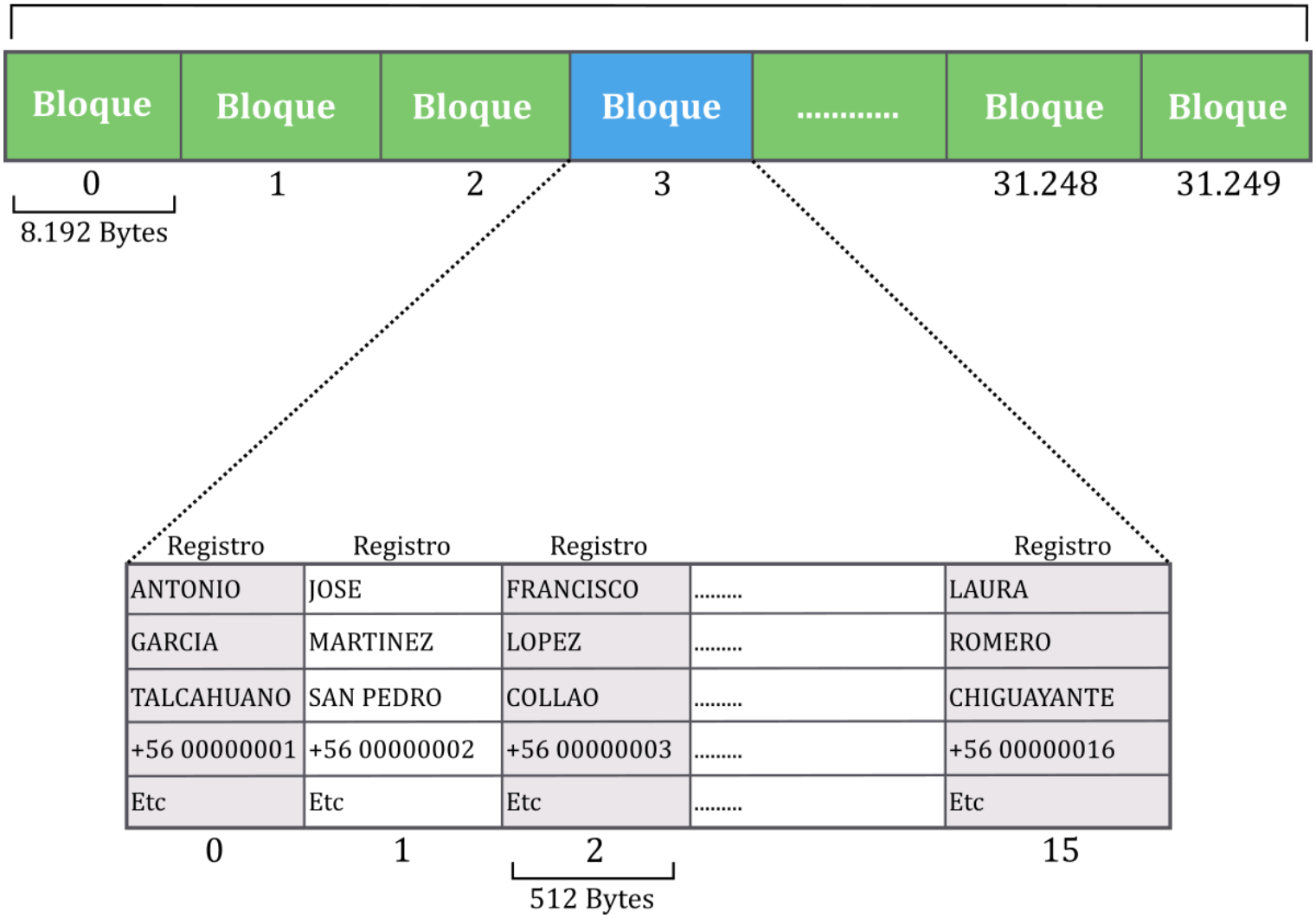
Un Bloque es de **8.192 B (2^13)**
Base de datos tiene una capacidad de **256.000.000 B**

¿Cuántos bloques hay en la Base de datos?
R: 256.000.000 B/ 8.192 B = 31.250 Bloques

Si el tamaño de un registro es de **512 B**, ¿Cuántos registros hay por bloque?
R: 8.192 B / 512 B = 16 Registros

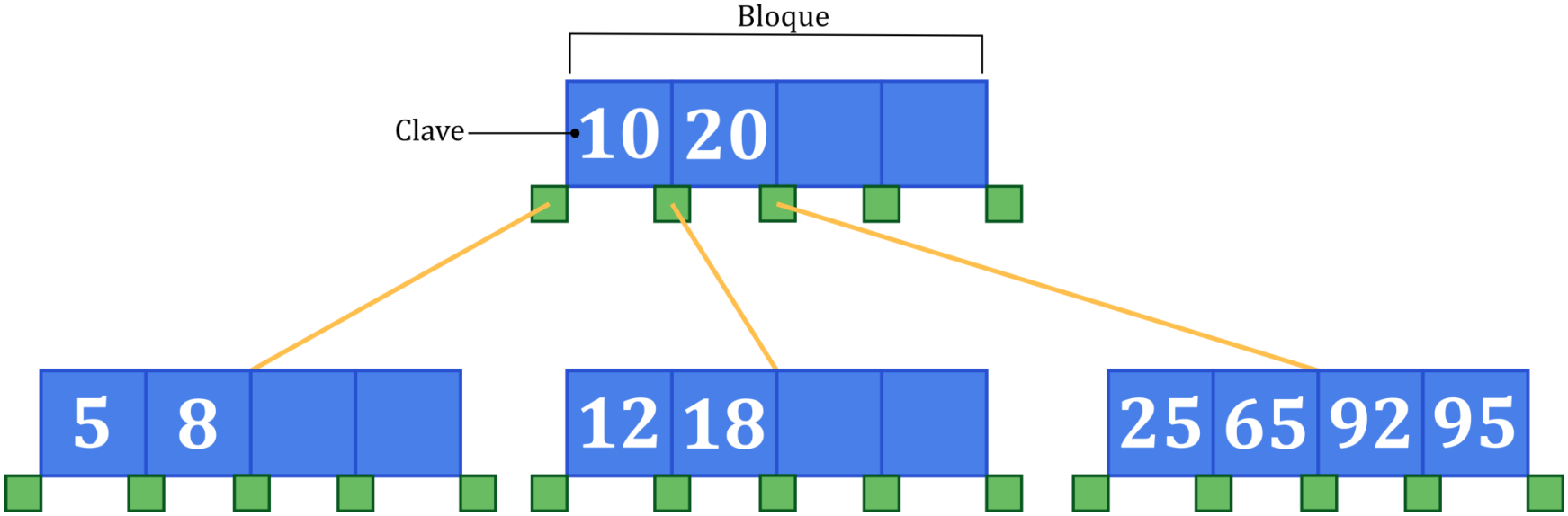
¿Cuántos registros hay en total?
R: 31.250 Bloques x 16 Registros = 500.000 Registros

Archivo de 256.000.000 Bytes



Árbol B

1. Cada nodo debería corresponder a un bloque de datos.
2. Un numero de bloque se puede almacenar en un atributo **int** de 4 bytes.
3. Son árboles de búsqueda multi-vías, ya que cada nodo almacena más de un elemento y tiene referencias a muchos sucesores (hijos).
4. Tiene pocos niveles pero la búsqueda por un ítem requiere de más comparaciones por nivel



Características

Un Árbol B de orden **n** es aquel que:

1. Todas las hojas del árbol están en el nivel inferior.
2. Cada nodo contiene entre **n** y **2n** elementos, excepto el nodo raíz, que puede tener entre **1** y 2n elementos.
3. Si un nodo tiene **m** elementos, el nodo siempre contendrá **m + 1** hijos si no es un nodo hoja.

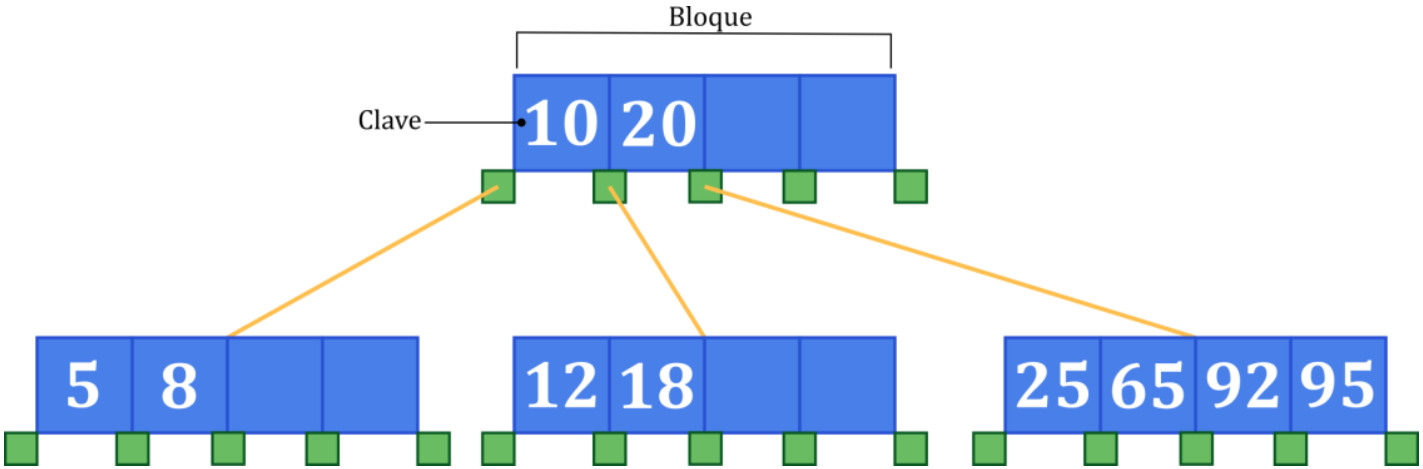
Árbol B

Si un árbol B es de orden 3.

1. Cuántos elementos máximo puede guardar cada nodo del árbol?
   R : Puede guardar como máximo 6 elementos.
2. ¿Cuántos elementos como mínimo puede guardar cada nodo del árbol?
   R: 1 si el la raíz, 3 cualquier otro nodo.
3. ¿Cuántos hijos como máximo puede tener un nodo?
   R: Máximo 7 hijos.
4. ¿Cuántos hijos como mínimo puede tener un nodo?
   R: 0 si es hoja, 2 si es raíz, 4 cualquier otro nodo.

Ejemplo

¿De qué orden es este árbol?

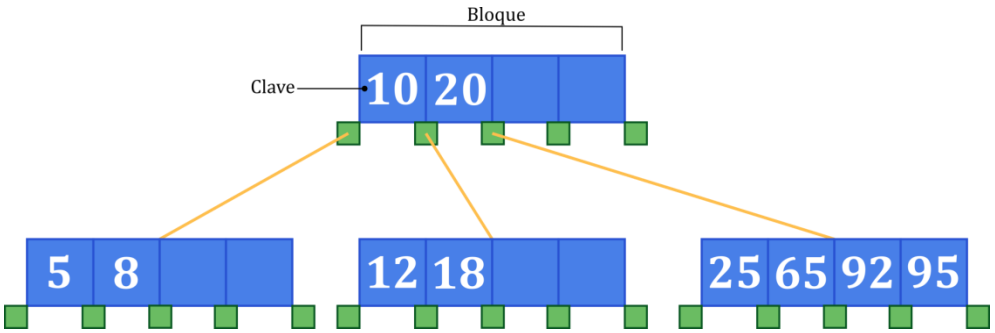


R: Es de orden 2.
Puede almacenar como máximo 2n elementos = 4
Puede tener como máximo 5 Hijos.
Puede tener como mínimo 2 elementos cada bloque, excepto la raíz que puede tener como mínimo un 1 elemento.

Más características

Un árbol B de orden **n** es aquél en el que:

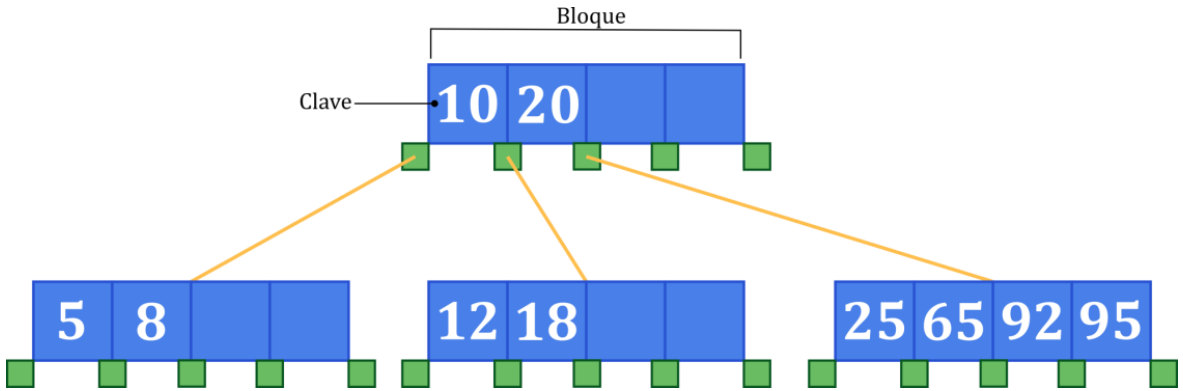
1. Los elementos de un nodo están ordenados linealmente.
2. Los elementos están organizados de tal forma que se cumple la regla de la búsqueda binaria: a la izquierda menores, a la derecha mayores.



Búsqueda en un Árbol B

La búsqueda por un registro para una llave **x**:

1. Cargar bloque que contiene raíz en memoria principal
2. Iterar a través de los registros (k(i), r(i)) ... (k(m) , r(m)) Buscando x.
   a. Si x = k(i) retornar r(i)
   b. Si x < k(i), cargar bloque correspondiente al nodo hijo(i) e iterar nuevamente sobre los registros
   c. Si x > k(m), leer bloque correspondiente al nodo hijo(m + 1) e iterar nuevamente sobre los registros



Inserción

1. Buscar el nodo hoja en donde se debería agregar el elemento.
2. Si hay espacio disponible en el nodo, agregar el elemento y terminar.
3. Si el nodo hoja **NO** tiene capacidad de almacenar el elemento, se deberá crear un nuevo nodo al mismo nivel de la hoja y distribuir a los 2n+1 elementos de la siguiente forma:
   a. El nuevo nodo recibe a los 'n' elementos más grandes.
   b. El nodo existente se queda con los 'n' elementos más pequeños.
   c. La mediana se insertará en el nodo padre siguiendo la misma lógica de inserción. En caso de no haber nodo padre, se creará un nuevo nodo que pasará a ser la nueva raíz.

Inserción

Inserte los siguientes valores: **100, 25, 50**, 0, 75, 20, 60, 70, 90, 150.
Orden: 1.

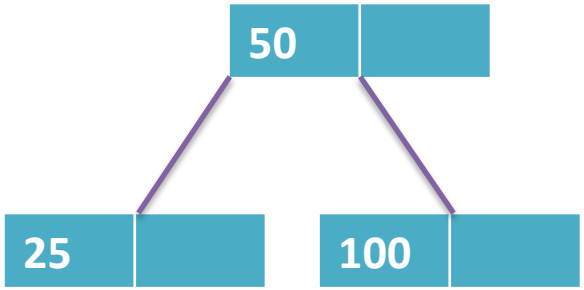
Insertando 100

Insertando 25

Insertando 50

Límite máximo por bloque superado. Es necesario sub-dividir.

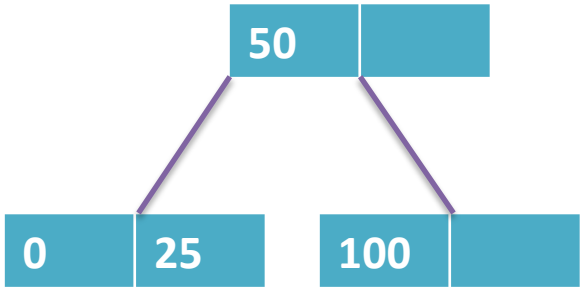
Se selecciona el elemento central (Mediana) del conjunto y se asigna como padre de los demás



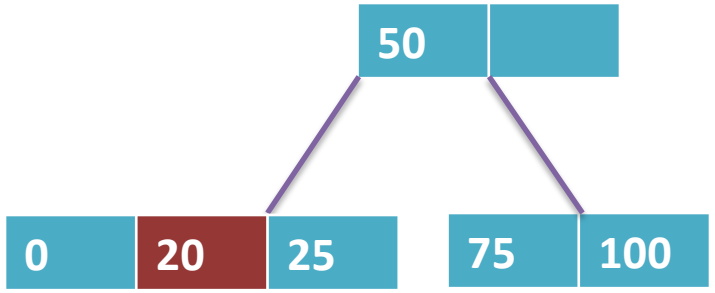
Inserción

Inserte los siguientes valores: 100, 25, 50, **0, 75, 20**, 60, 70, 90, 150.
Orden: 1.

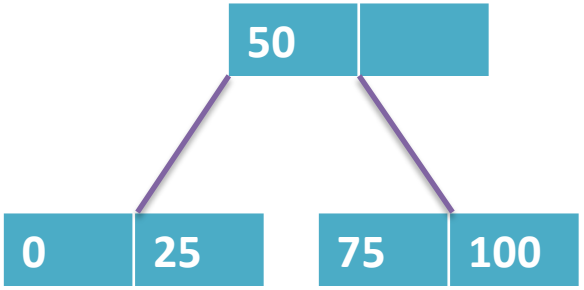
Insertando 0



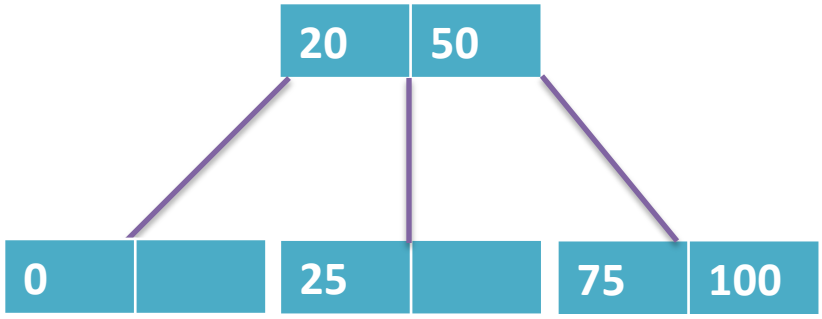
Insertando 20 (Límite alcanzado.)



Insertando 75



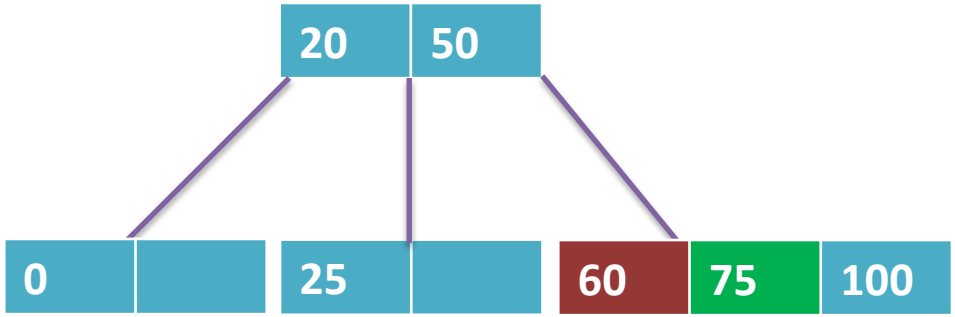
Se selecciona el elemento central (Mediana) del conjunto y se asigna como padre de los demás (Subiendo un nivel).



Inserción

Inserte los siguientes valores: 100, 25, 50, 0, 75, 20, **60,** 70, 90, 150.
Orden: 1.

Insertando 60 (Límite alcanzado.)

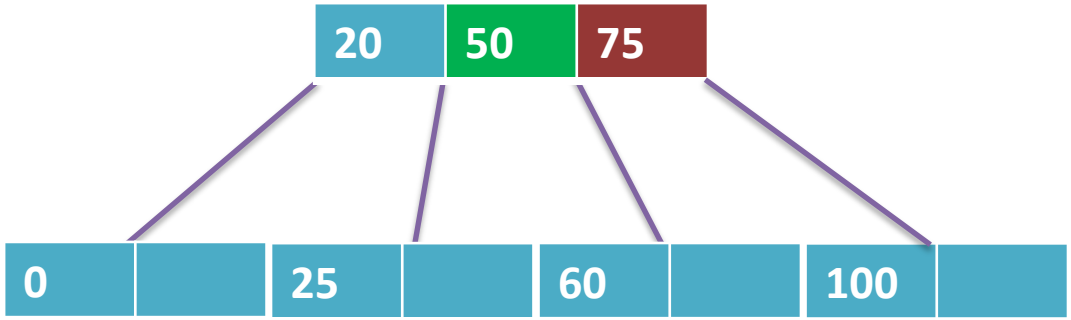


Se selecciona el elemento central (Mediana) del conjunto y se asigna como padre de los demás (Subiendo un nivel).

Inserción

Inserte los siguientes valores: 100, 25, 50, 0, 75, 20, 10, **60,** 70, 90, 150.
Orden: 1.

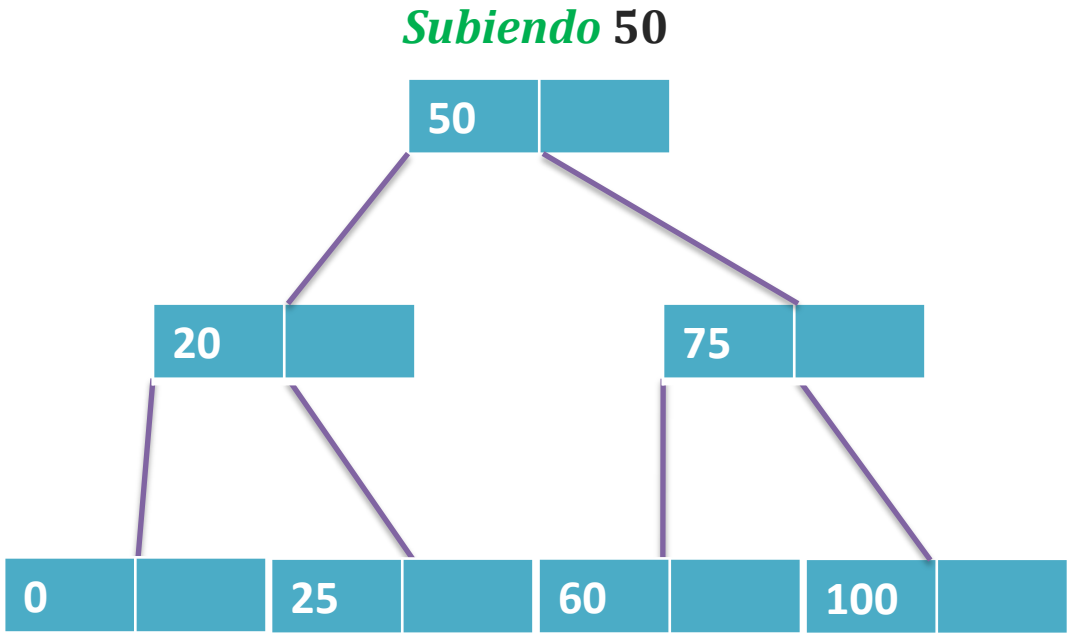
Subiendo **75(Límite alcanzado.)**



Se selecciona el elemento central (Mediana) del conjunto y se asigna como padre de los demás (Subiendo un nivel).

Inserción

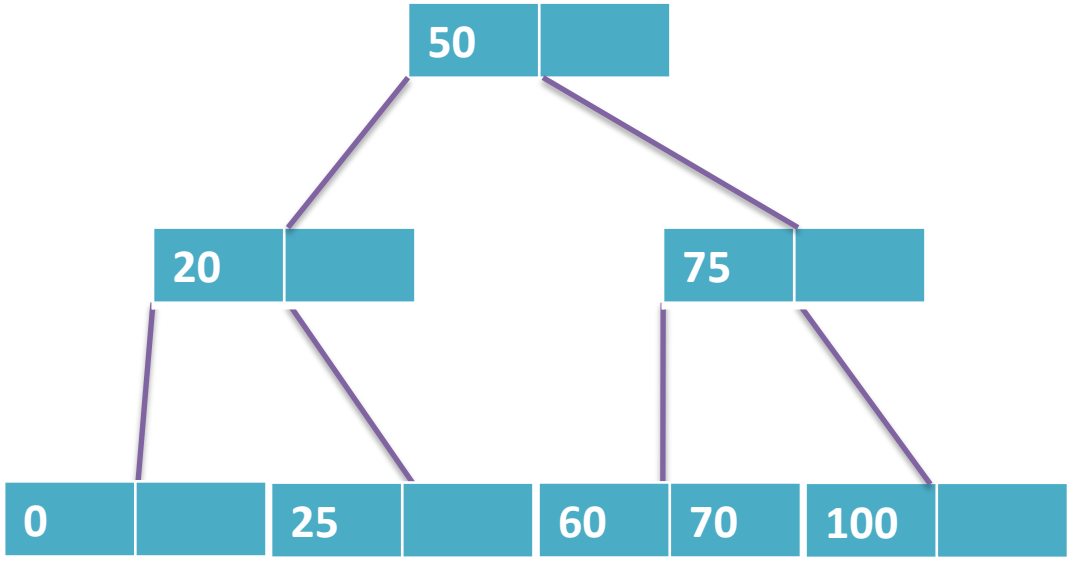
Inserte los siguientes valores: 100, 25, 50, 0, 75, 20, 10, **60,** 70, 90, 150.
Orden: 1.



Inserción

Inserte los siguientes valores: 100, 25, 50, 0, 75, 20, 10, 60**, 70**, 90, 150.
Orden: 1.

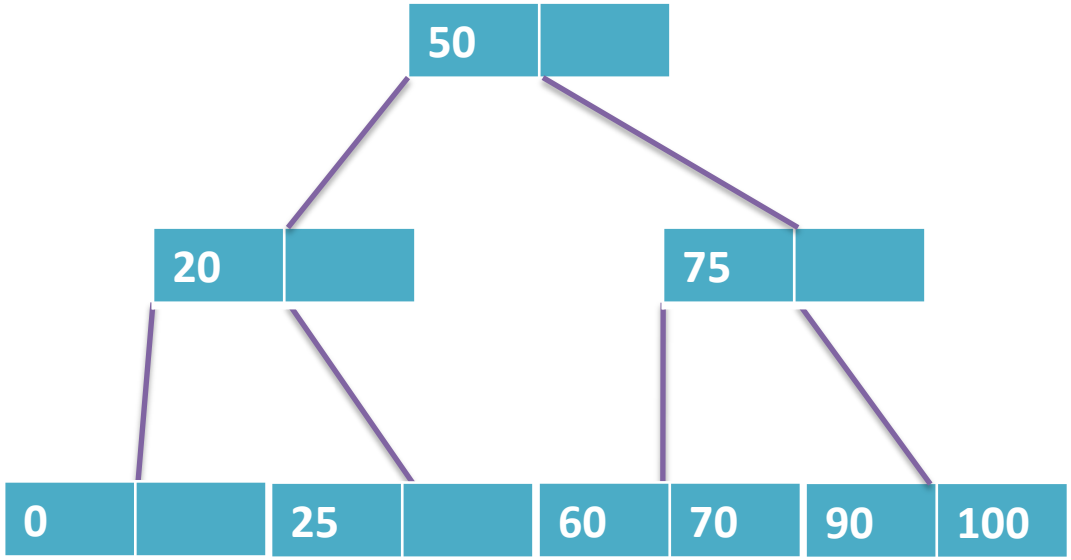
Insertando 70



Inserción

Inserte los siguientes valores: 100, 25, 50, 0, 75, 20, 10, 60, 70, **90**, 150.
Orden: 1.

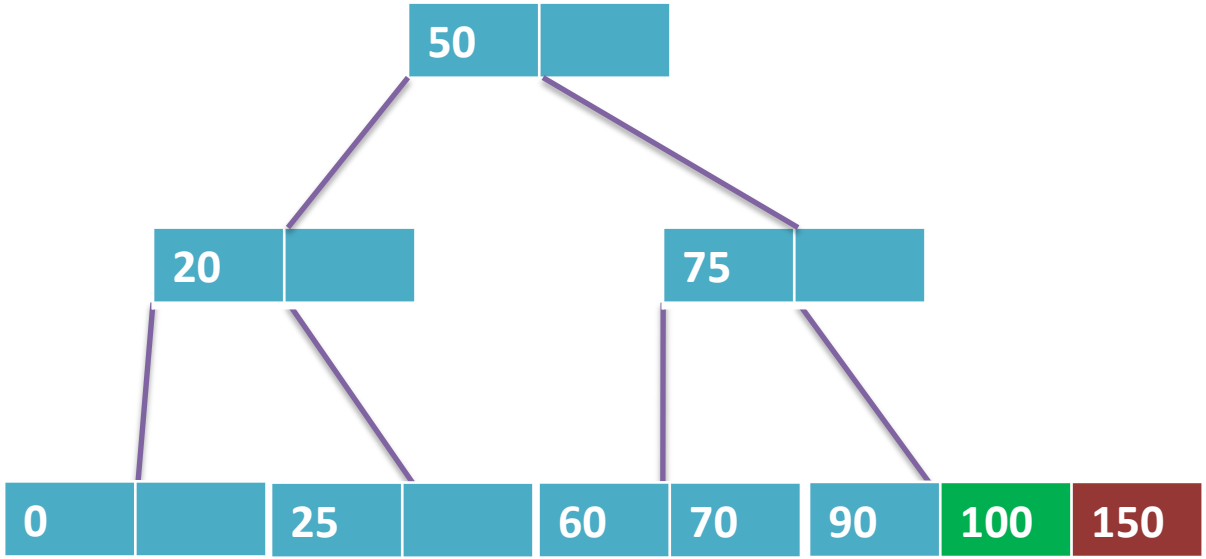
Insertando 90



Inserción

Inserte los siguientes valores: 100, 25, 50, 0, 75, 20, 10, 60, 70, 90, **150**.
Orden: 1.

Insertando 150 (Límite alcanzado.)

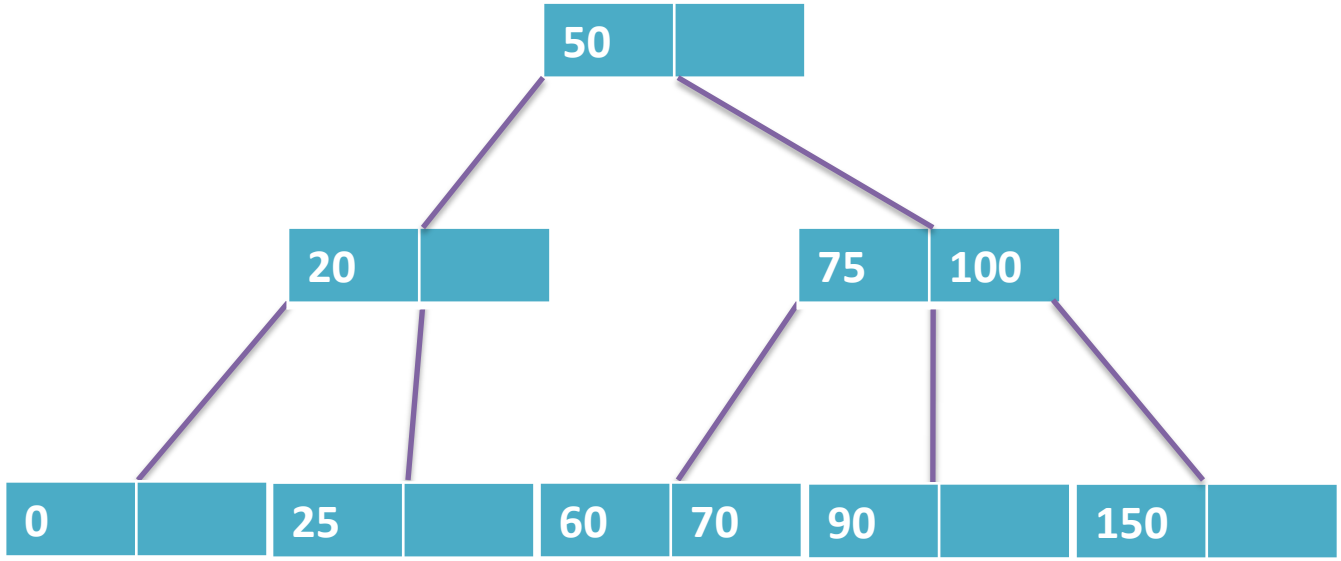


Se selecciona el elemento central (Mediana) del conjunto y se asigna como padre de los demás (Subiendo un nivel).

Inserción

Inserte los siguientes valores: 100, 25, 50, 0, 75, 20, 10, 60, 70, 90, **150**.
Orden: 1.

Subiendo 100



Eliminación

1. Localizar y eliminar el ítem, luego reestructurar el árbol para recuperar sus invariantes
2. Hay dos casos especiales a tener en cuenta cuando se elimina un ítem:
   a. El ítem es un nodo interno que puede ser un separador para sus nodos hijos
   b. La eliminación de un ítem puede poner el nodo bajo el número mínimo de elementos e hijos (nodo deficiente)

Eliminación

1. Caso 1: (Nodo Interno)
   - Elija un nuevo separador(ya sea el elemento más grande en el subárbol izquierdo o el elemento más pequeño en el subárbol derecho), retire nodo separador (hoja) y reemplace el elemento que desea eliminar con este.
2. Caso 2: (Nodo Hoja)
   - Si el valor se encuentra en un nodo de hoja, simplemente elimínelo del nodo. Tal vez deje el nodo con muy pocos ítems; por lo que serán necesarios algunos cambios adicionales en el árbol.

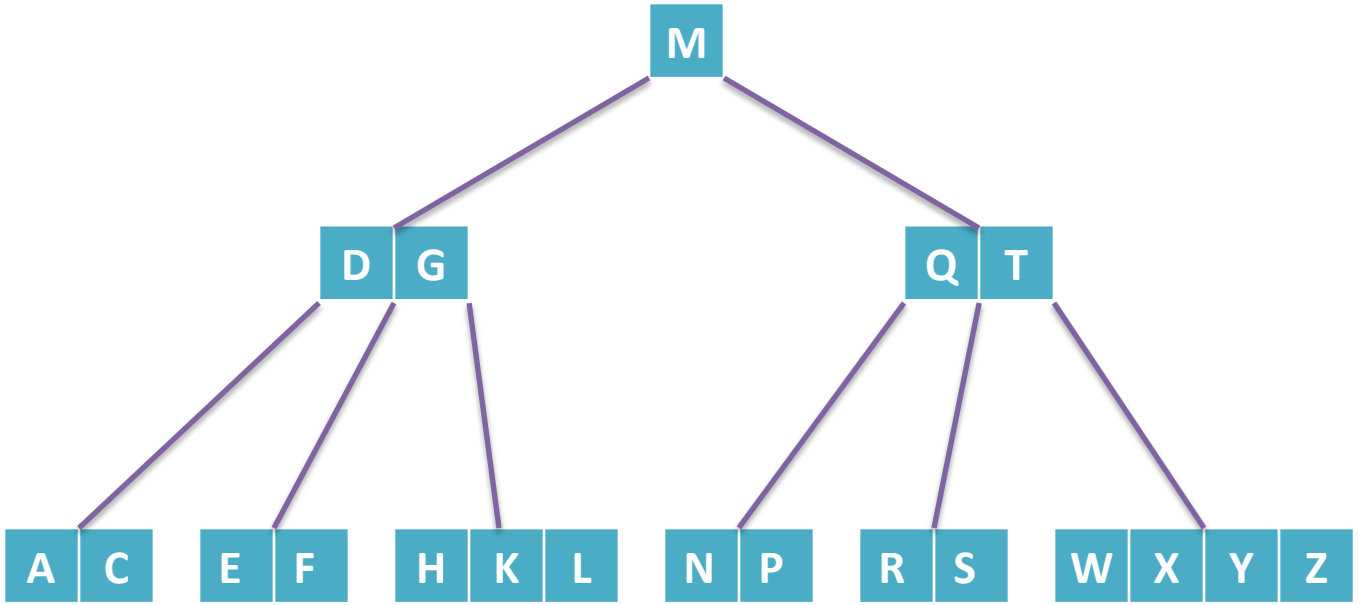
Eliminación

Cambios adicionales – Rebalanceo después de eliminar

1. Si el hermano derecho tiene más del número mínimo de ítems
   - Pedir prestado uno, ajustar el separador.
2. Sino, revisar el hermano izquierdo, si tiene más del número mínimo de ítems
   - Pedir prestado uno, ajuste el separador.
3. Si ambos hermano inmediatos tienen sólo el mínimo número de elementos
   - Crear un nuevo nodo con todos los ítems del nodo deficiente, todos los ítems de uno de sus hermanos, y el separador en el padre entre los dos nodos hermanos combinados.
   - Retirar el separador de los padres, y reemplazar a los dos hijos que se separaron con el nodo combinado.
   - Si esto provoca que el número de ítems del padre sea menor al mínimo, repita estos pasos con el nuevo nodo deficiente, a menos que sea la raíz, ya que la raíz puede ser deficiente.

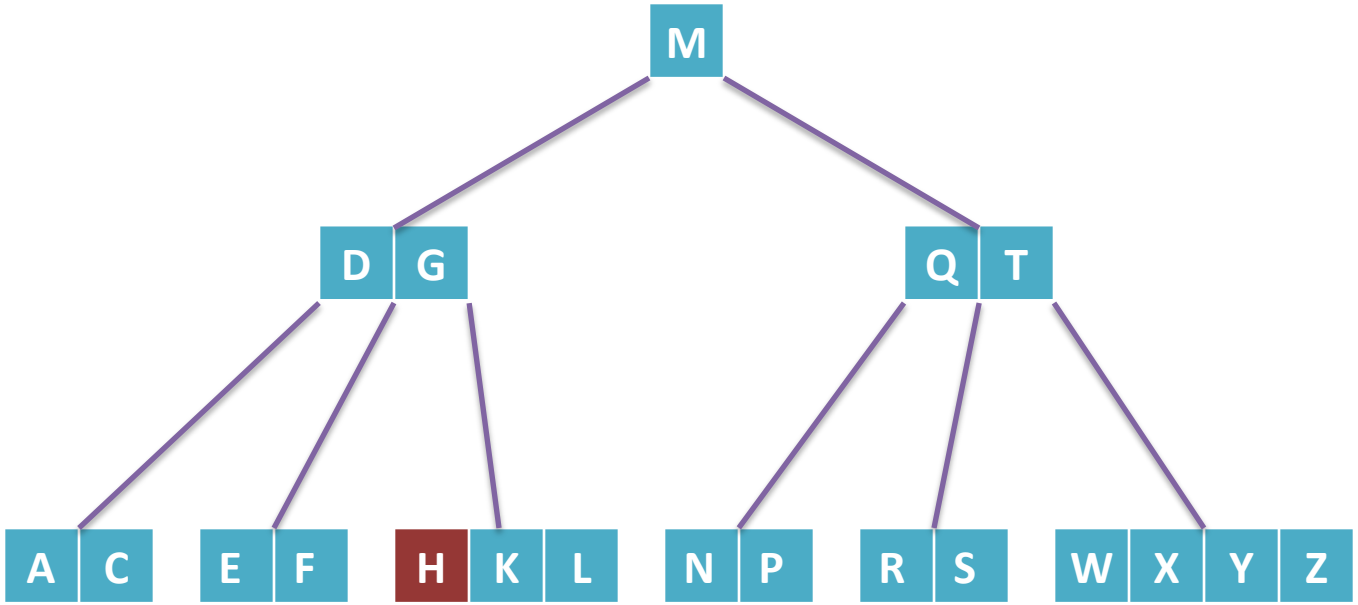
Eliminación

1. ¿Orden?
   - R = 2
2. ¿Mínimo de elementos por Nodo? (Excepto Raíz)
   - R = 2

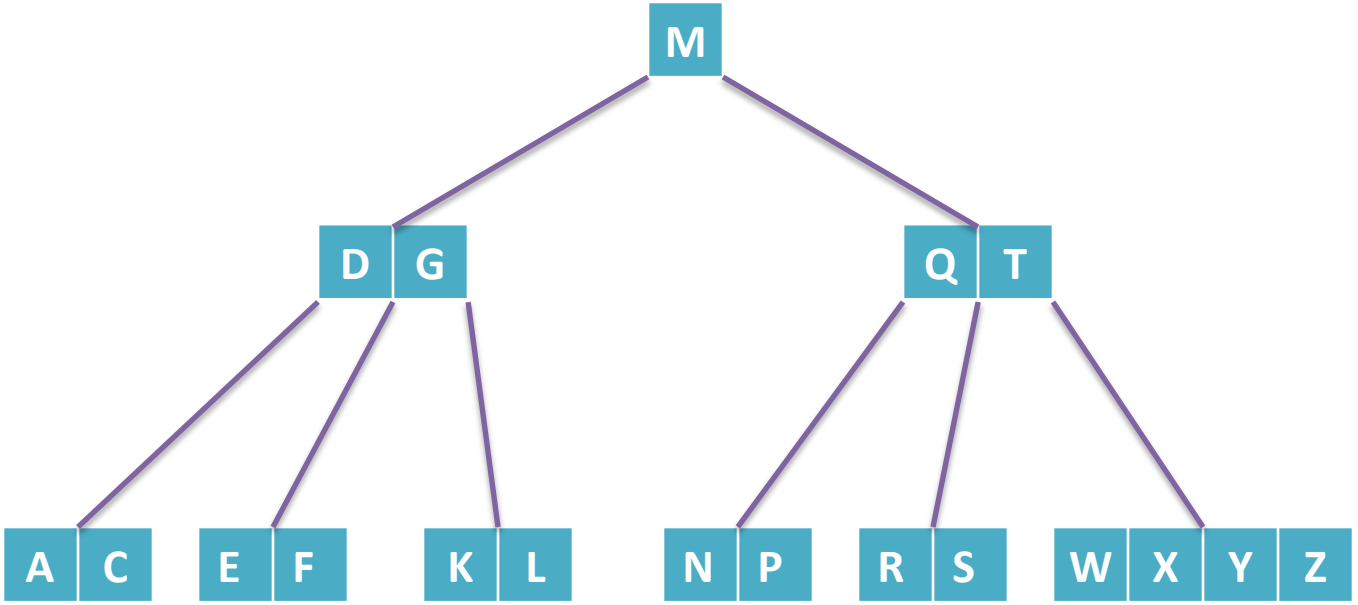


Eliminación

Eliminar H.
H es Nodo Hoja, si es eliminado aún se obtiene la cantidad mínima permitida.
Acción: Eliminación simple. (Solamente quitar)

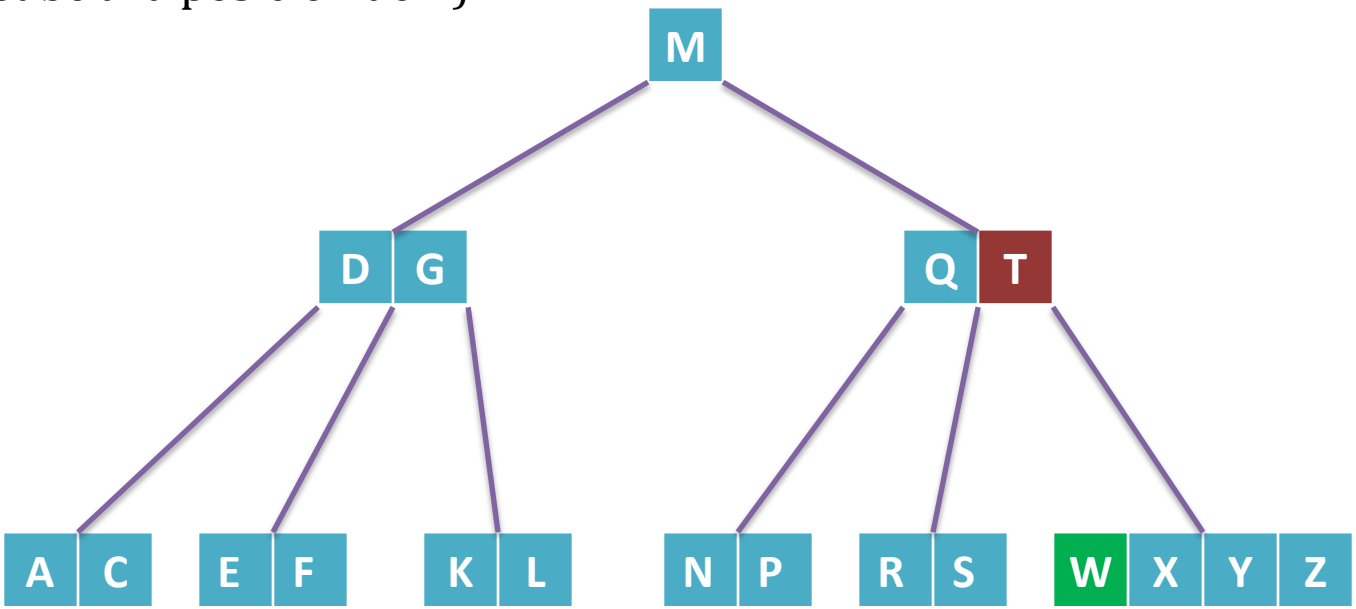


Eliminación

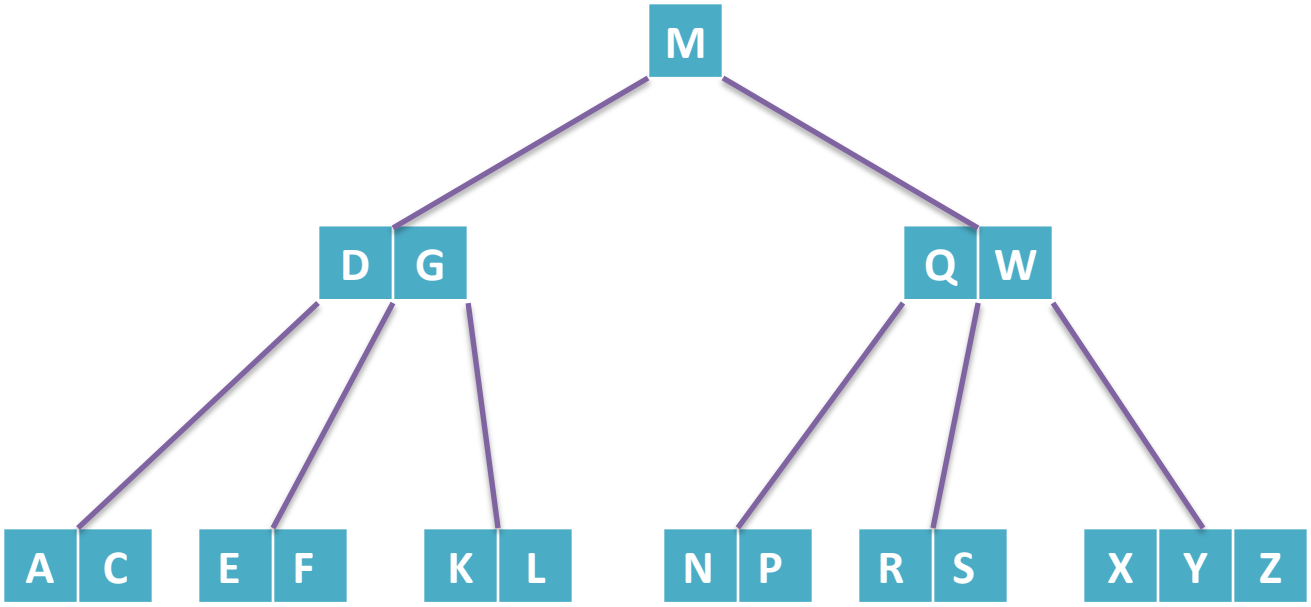


Eliminación

Eliminar T.
T es Nodo Interno.
Acción: Seleccionar el menor ítem del sub árbol derecho que reemplace a T.
(**W** sube a la posición de T)

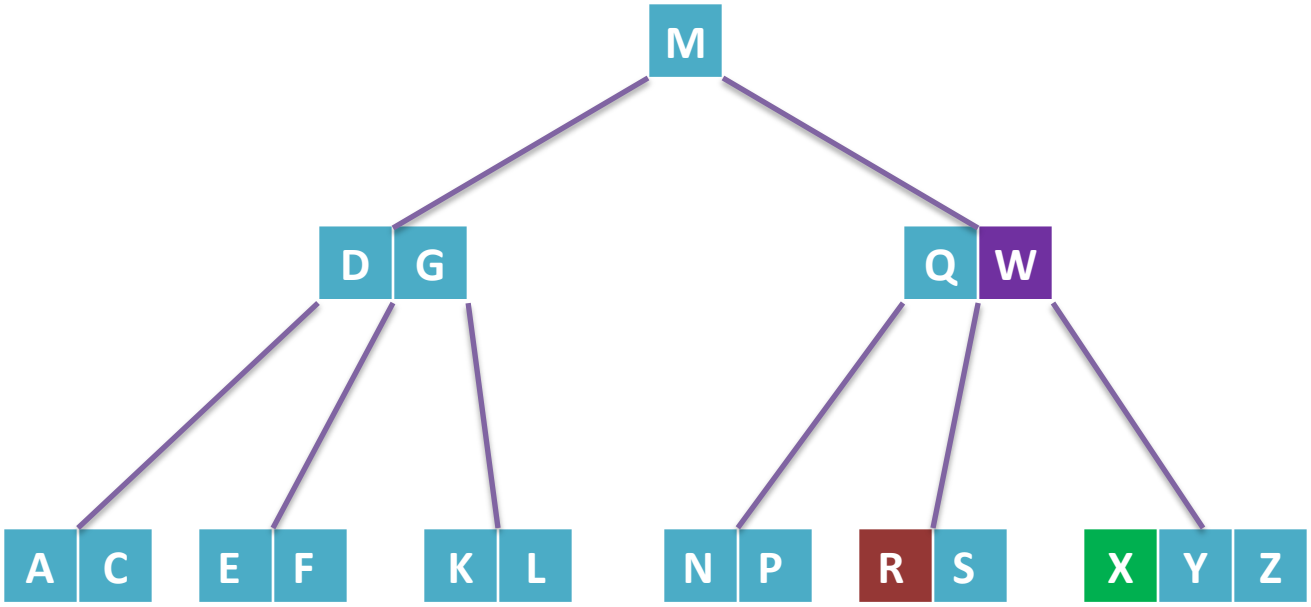


Eliminación

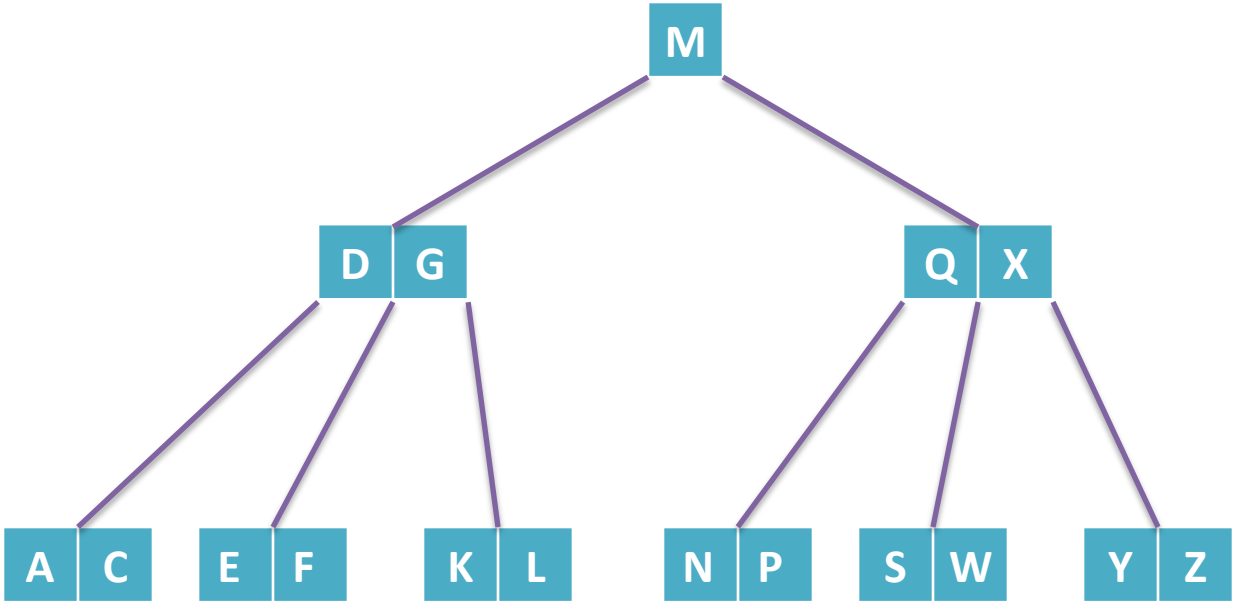


Eliminación

Eliminar R.
R es Nodo Hoja, si es eliminado romperá la regla de la cantidad mínima.
Acción: Eliminar R y pedir prestado una llave del hermano derecho, ajustar el separador:
Baje **W**, combínelo con **S**, suba **X** al padre.



Eliminación



Eliminación

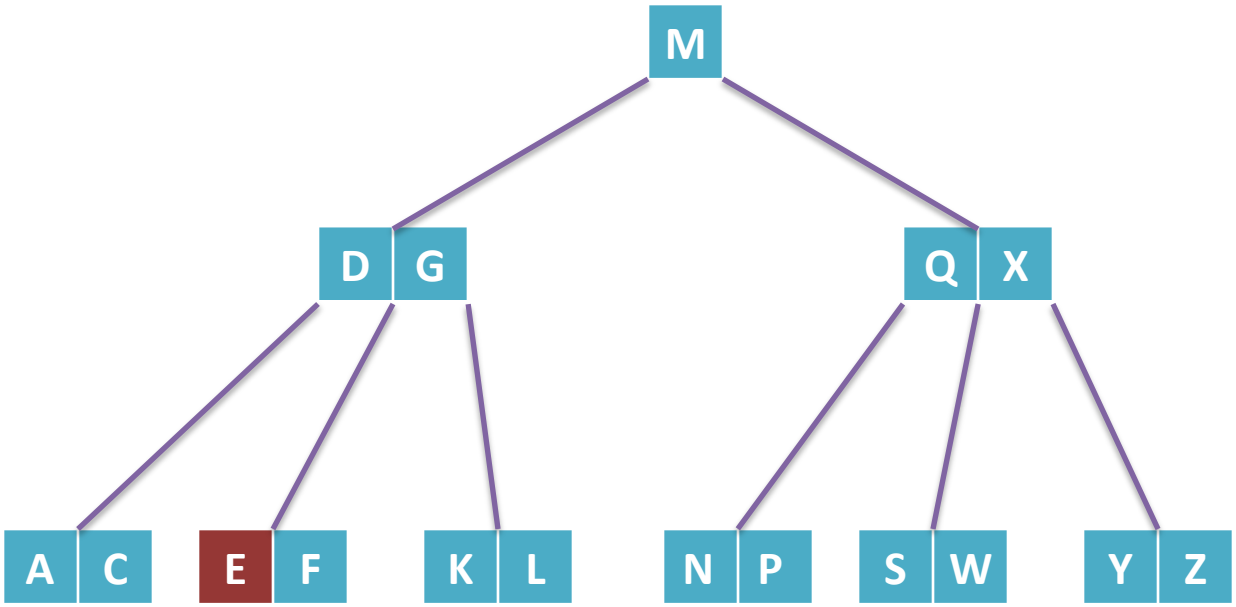
Eliminar E.

E es Nodo Hoja, si es eliminado romperá la regla de la cantidad mínima.

No puede pedir prestado un elemento a su hermano Derecho ya que rompería la regla para su Nodo.

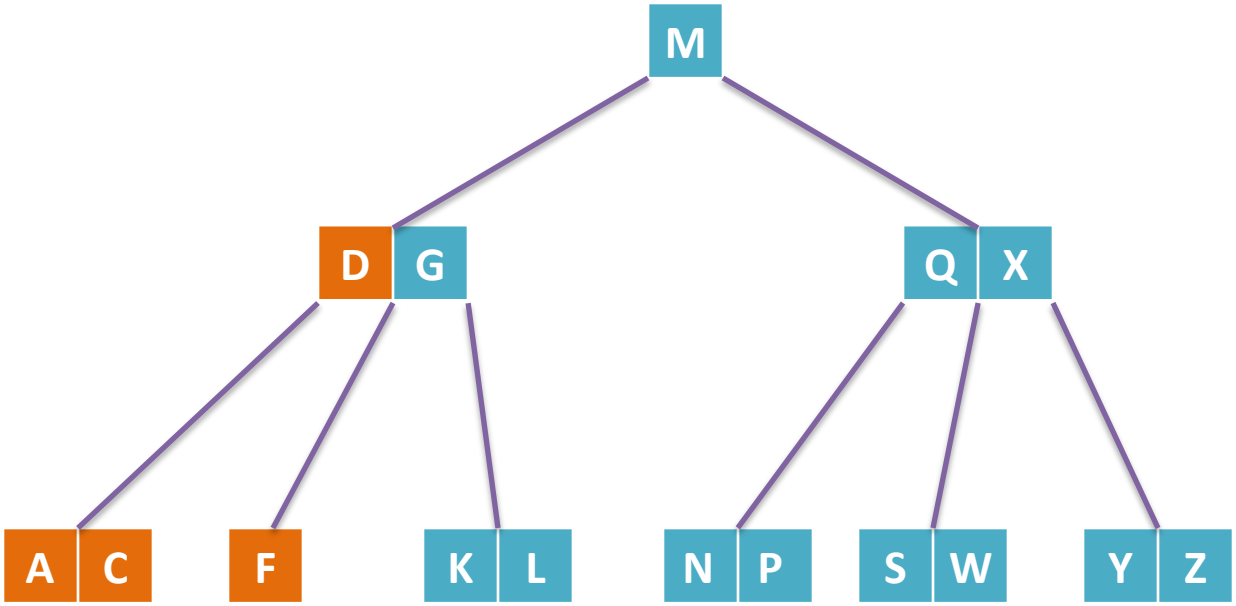
Se intenta pedir al hermano izquierdo, pero si presta un elemento **romperá la regla** para su Nodo.

Acción: Crea un nuevo nodo que combina el hermano izquierdo, el separador desde el padre y el nodo deficiente.



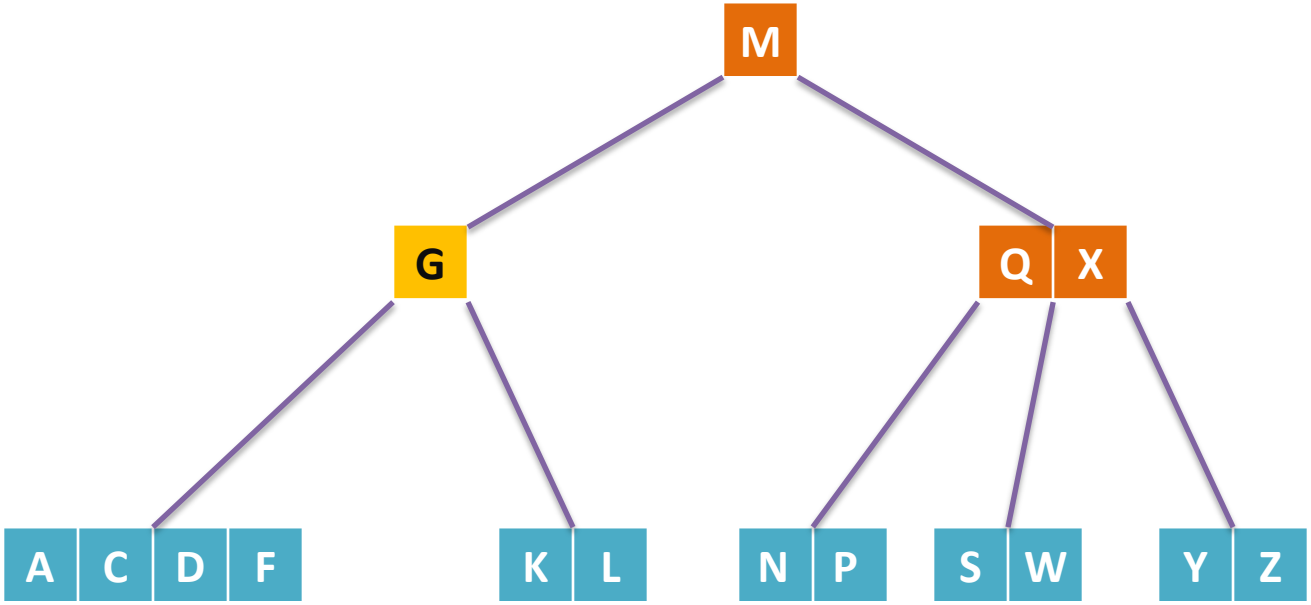
Eliminación

A, C, D, F se fusionan.

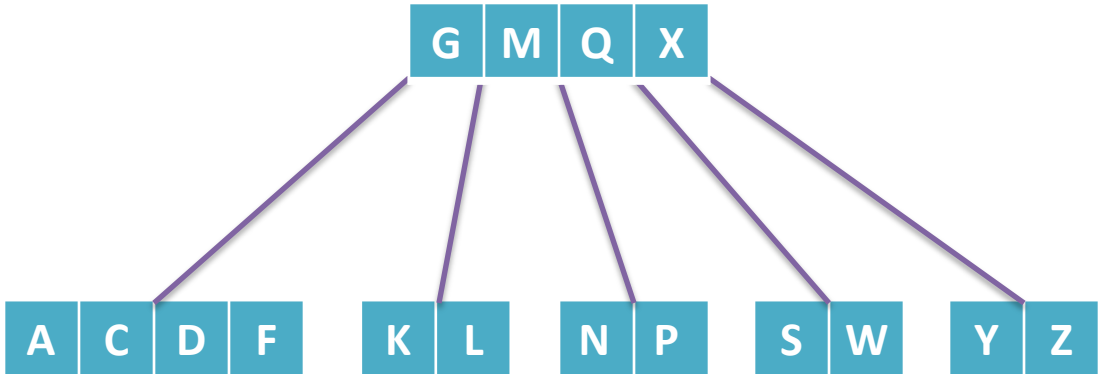


Eliminación

Ahora G, no cumple la regla…
Se intenta pedir un elemento al hermano Derecho, pero no se puede ya que rompería la regla para su Nodo.
Acción: Crea un nuevo nodo que combina el **hermano derecho**, el **separador desde el padre** y el **nodo deficiente**.



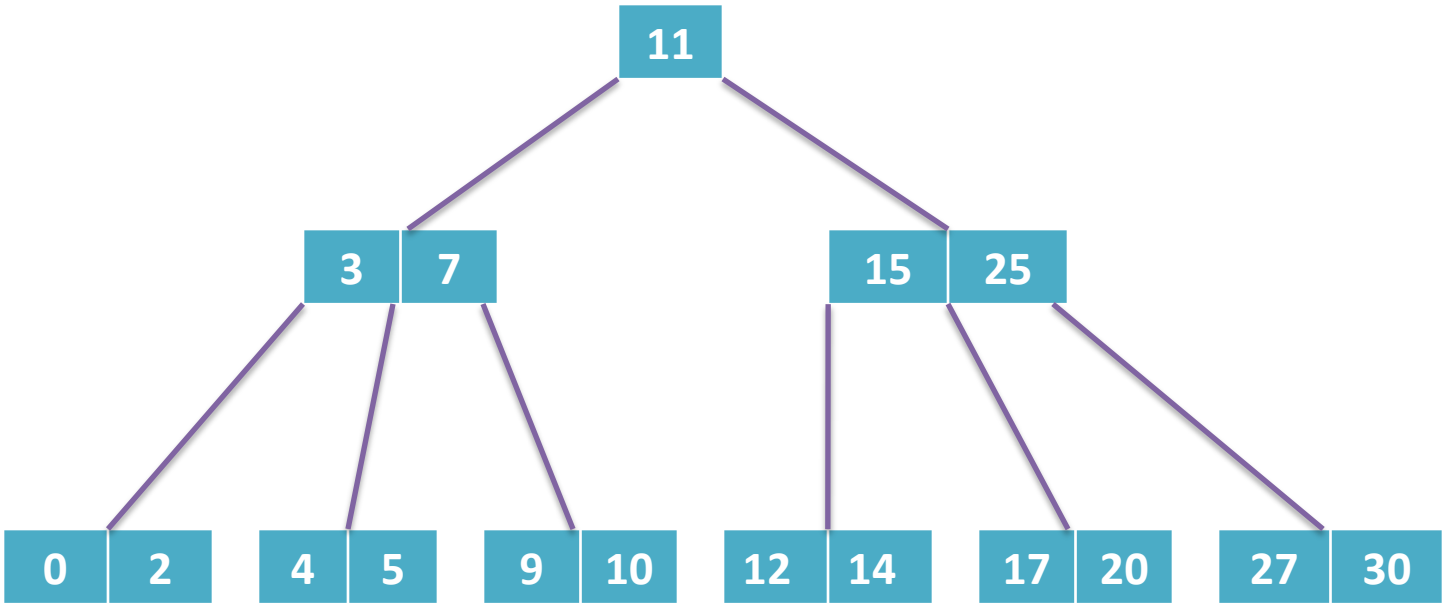
Eliminación



Árbol B - Ejercicio - 1

Inserte los siguientes valores: 10, 15, 7, 4, 5, 2, 0, 3, 14, 17, 20, 25, 12, 11, 9, 30, 27.
Orden: 2.

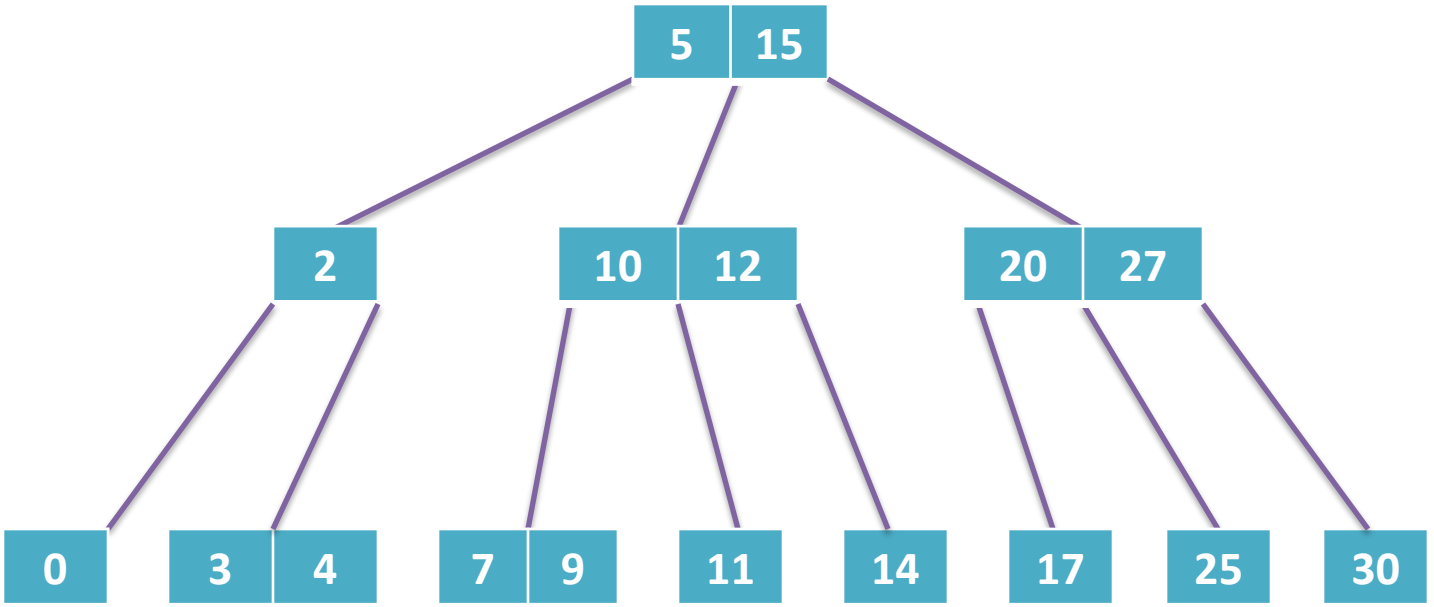
Solución



Árbol B - Ejercicio - 2

Inserte los siguientes valores: 10, 15, 7, 4, 5, 2, 0, 3, 14, 17, 20, 25, 12, 11, 9, 30, 27.
Orden: 1.

Solución



Variantes de Árbol B – B+

1. El árbol B+, es un árbol B, en el que los datos se almacenan sólo en las hojas, nunca en niveles superiores.
2. Esto hace que los nodos padres sean más compactos (sólo contienen llaves), a costa de algo de redundancia: las llaves de los padres se duplican en las hojas.
3. Tener en cuenta que esto cambia la eliminación de las llaves en los padres, ya que los nodos a la derecha de una llave ahora son mayores que o igual a la llave.
4. Esto complica en algo la implementación, ya que los nodos hoja son diferentes a los otros nodos.

Variantes de Árbol B – B*

B * - propuesto por D. Knuth
Es una variante del árbol B donde todos los nodos, excepto la raíz, están obligados a estar por lo menos 2/3 lleno.
La implementación es más compleja, ya que:

- Al insertar en vez de hacer una división de un nodo en dos, tienes que dividir dos nodos en tres.
- Sin embargo, la operación de división se retrasa con el fin de redistribuir las llaves entre los vecinos cuando el nodo se llena. Así, en lugar de una división, primero se debe tratar de redistribuir las llaves en el nodo, su vecino y la llave en el nodo principal.

UNIVERSIDAD DEL BÍO-BÍO

Para continuar aprendiendo visitar Biblioteca
Nro. de Pedido :005.73 C123 2002
Autor : Cairó, Osvaldo
Título : Estructuras de datos

Para practicar inserción y eliminación de una manera visual, visitar:
https://www.cs.usfca.edu/~galles/visualization/BTree.html