Árboles B

Estructura de Datos

Árboles B

- Los árboles B son estructuras no lineales que fueron introducidos por R. Bayer y E. McCreight en 1972, con el principal objetivo de mejorar el tiempo de acceso en estructuras de datos manejadas en memoria externa.

- Los árboles B son una generalización de los árboles balanceados, con una estructura jerárquica que beneficia considerablemente la búsqueda de un elemento en específico, reduciendo el número de nodos o archivos accesados.

Árboles B

- **En un árbol B se debe procurar:**
  - Conservar la altura del árbol al mínimo, por lo que no deben existir subárboles vacíos al interior del árbol.
  - Que todos los nodos, excepto la raíz, tengan un mínimo número de llaves (quizás la mitad del máximo).
  - Que todas las hojas se mantengan al mismo nivel, garantizando así que las búsquedas se consigan con aproximadamente el mínimo número de accesos.

- **Luego, un árbol B de "orden d" se puede formalizar de la siguiente forma:**
  - Cada página (nodo), exceptuando la raíz contiene entre d y 2d elementos.
  - Cada página (nodo), excepto la página raíz y las páginas hojas, tienen entre d+1 y 2d+1 descendientes. Se utiliza m para expresar el número de elementos pos cada página.
  - La página raíz posee al menos dos descendientes.
  - Las páginas hojas están todas al mismo nivel.

Árboles B

- La idea tras los árboles-B es que los nodos internos deben tener un número variable de nodos hijo dentro de un rango predefinido.
- Cuando se inserta o se elimina un dato de la estructura, la cantidad de nodos hijo varía dentro de un nodo.
- Para que siga manteniéndose el número de nodos dentro del rango predefinido, los nodos internos se juntan o se parten.
- Dado que se permite un rango variable de nodos hijo, los árboles-B no necesitan rebalancearse tan frecuentemente como los árboles binarios de búsqueda auto-balanceables.
- Pero, por otro lado, pueden desperdiciar memoria, porque los nodos no permanecen totalmente ocupados.
- Los límites (uno superior y otro inferior) en el número de nodos hijo son definidos para cada implementación en particular.

Árboles B

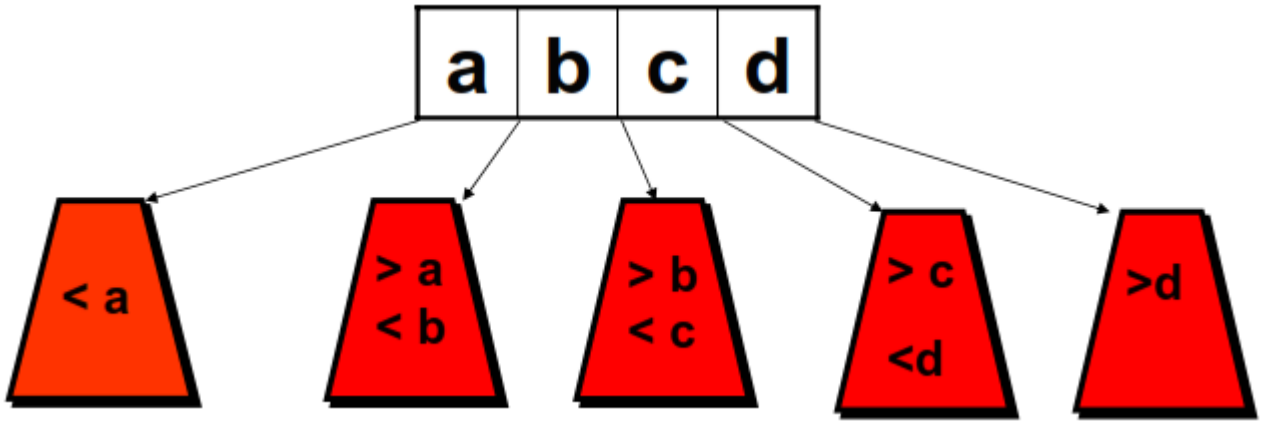
- Un árbol-B se mantiene balanceado porque requiere que todos los nodos hoja se encuentren a la misma altura.
- Los árboles B tienen ventajas sustanciales sobre otras implementaciones cuando el tiempo de acceso a los nodos excede al tiempo de acceso entre nodos.
- Este caso se da usualmente cuando los nodos se encuentran en dispositivos de almacenamiento secundario como los discos rígidos.
- Al maximizar el número de nodos hijo de cada nodo interno, la altura del árbol decrece, las operaciones para balancearlo se reducen, y aumenta la eficiencia.
- Usualmente este valor se coloca de forma tal que cada nodo ocupe un bloque de disco, o un tamaño análogo en el dispositivo.

Definición técnica

- B-árbol es un árbol de búsqueda que puede estar vacío o aquel cuyos nodos pueden tener varios hijos, existiendo una relación de orden entre ellos.
- Un árbol-B de orden M (el máximo número de hijos que puede tener cada nodo) es un árbol que satisface las siguientes propiedades:
  - Cada nodo tiene como máximo M hijos.
  - Cada nodo (excepto raíz) tiene como mínimo (M)/2 claves.
  - La raíz tiene al menos 1 hijos si no es un nodo hoja. (según M)
  - Todos los nodos hoja aparecen al mismo nivel.
  - Un nodo no hoja con k hijos contiene k-1 elementos almacenados.
  - Los hijos que cuelgan de la raíz (r1, ⋯, rm) tienen que cumplir ciertas condiciones:
    - El primero tiene valor menor que r1.
    - El segundo tiene valor mayor que r1 y menor que r2, etc.
    - El último hijo tiene valor mayor que rm.

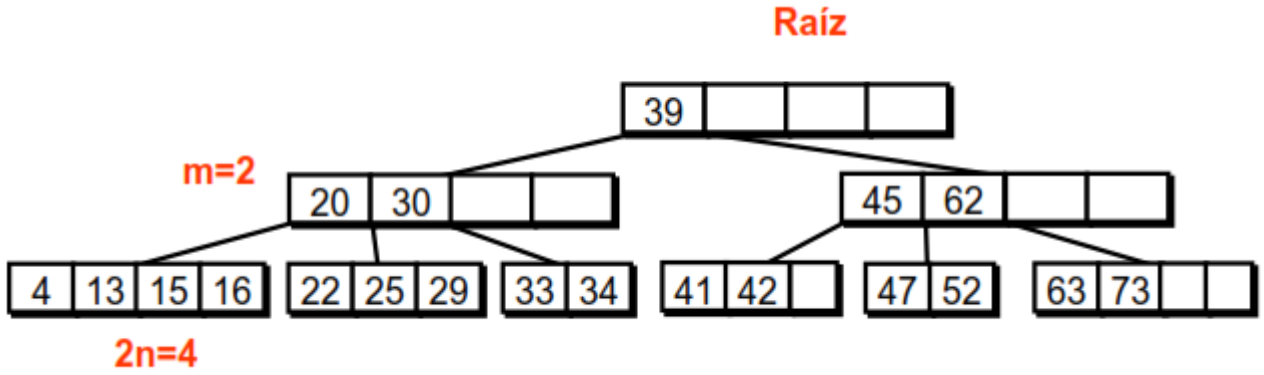
Características

- Las claves dividen el espacio como en un AVL
- Ejemplo: 2n=4=2*2
  - Número máximo por página: 4 claves y 5 ramas
  - Número mínimo por página: 2 claves y 3 ramas.
- Se rastrea el camino de búsqueda similar al ABB.



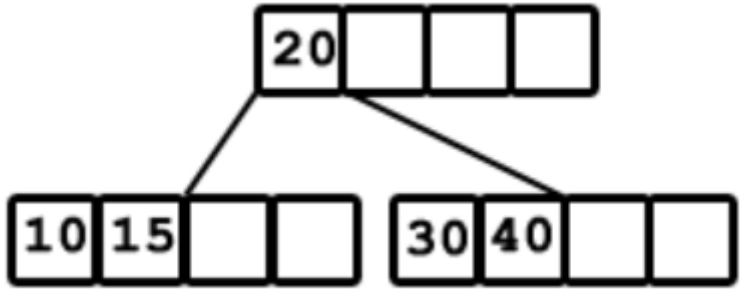
Características

- Ejemplo:
  - Máximo número de claves: 2n=2*2
  - Mínimo n=2
  - Raíz: una clave

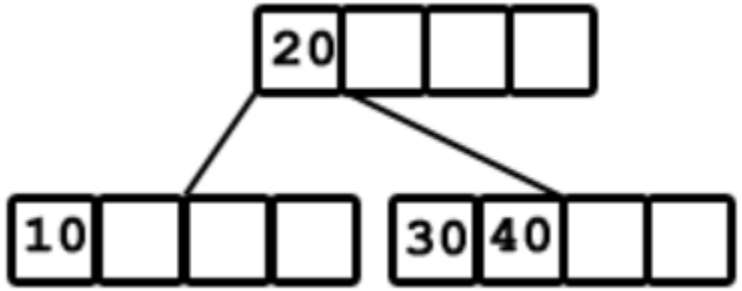


Ejemplos

Este es un árbol B correcto, ya que cumple todas las reglas en cuanto a su estructura y al orden.



En cambio, este no es un árbol B, ya que a pesar de mantener el orden, hay una página (que no es la raíz) que tiene menos de *n* elementos (en este caso menos de 2 elementos).



Estructura de los nodos

- Cada elemento de un nodo interno actúa como un valor separador, que lo divide en subárboles.

- Por ejemplo, si un nodo interno tiene tres nodos hijo, debe tener dos valores separadores o elementos a1 y a2.

- Todos los valores del subárbol izquierdo deben ser menores a a1, todos los valores del subárbol del centro deben estar entre a1 y a2, y todos los valores del subárbol derecho deben ser mayores a a2.

Estructura de los nodos

- Los nodos internos de un árbol B, es decir los nodos que no son hoja, usualmente se representan como un conjunto ordenado de elementos y punteros a los hijos.
- Cada nodo interno contiene un máximo de U hijos y, con excepción del nodo raíz, un mínimo de L hijos.
- Para todos los nodos internos exceptuando la raíz, el número de elementos es uno menos que el número de punteros a nodos. El número de elementos se encuentra entre L-1 y U-1.
- El número U debe ser 2L o 2L-1, es decir, cada nodo interno está por lo menos a medio llenar.
- Esta relación entre U y L implica que dos nodos que están a medio llenar pueden juntarse para formar un nodo legal, y un nodo lleno puede dividirse en dos nodos legales (si es que hay lugar para subir un elemento al nodo padre).
- Estas propiedades hacen posible que el árbol B se ajuste para preservar sus propiedades ante la inserción y eliminación de elementos.

Estructura de los nodos

- Los nodos hoja tienen la misma restricción sobre el número de elementos, pero no tienen hijos, y por tanto carecen de punteros.

- El nodo raíz tiene límite superior de número de hijos, pero no tiene límite inferior. Por ejemplo, si hubiera menos de L-1 elementos en todo el árbol, la raíz sería el único nodo del árbol, y no tendría hijos.

- Un árbol B de altura n+1 puede contener U veces por elementos más que un árbol B de profundidad n, pero el costo en la búsqueda, inserción y eliminación crece con la altura del árbol. Como todo árbol balanceado, el crecimiento del costo es más lento que el del número de elementos.

- Algunos árboles balanceados guardan valores sólo en los nodos hoja, y por lo tanto sus nodos internos y nodos hoja son de diferente tipo.

- Los árboles B guardan valores en cada nodo, y pueden utilizar la misma estructura para todos los nodos. Sin embargo, como los nodos hoja no tienen hijos, una estructura especial para éstos mejora el funcionamiento.

Construcción inicial

- En aplicaciones, es frecuentemente útil construir un árbol-B para representar un gran número de datos existentes y después actualizarlo de forma creciente usando operaciones estándar de los árboles-B.

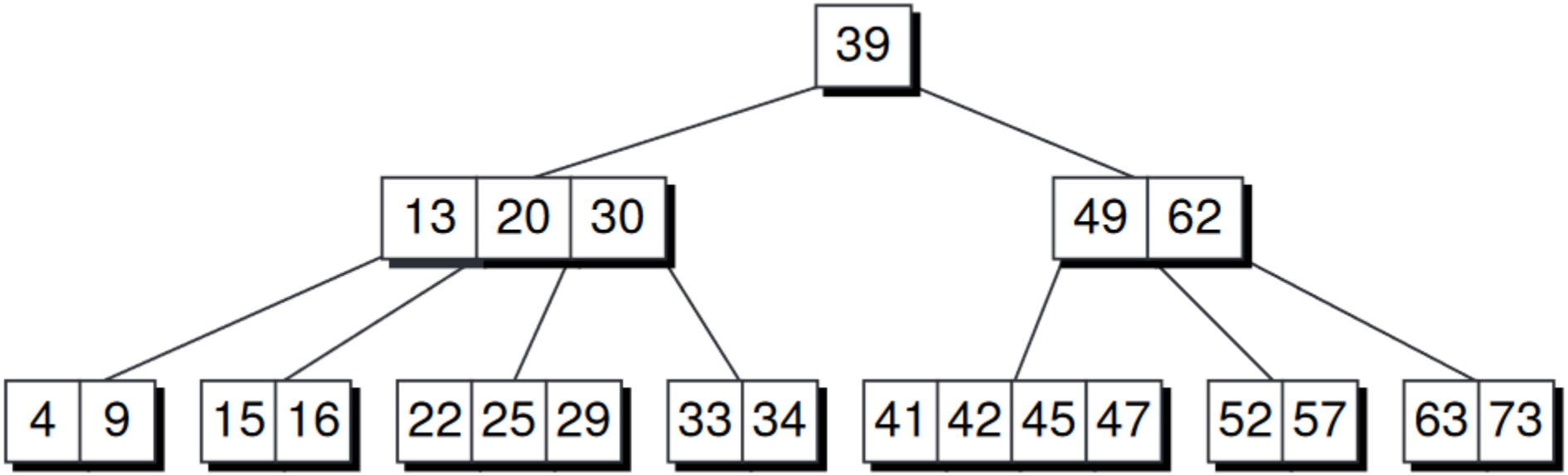
- En este caso, el modo más eficiente para construir el árbol-B inicial no sería insertar todos los elementos en el conjunto inicial sucesivamente, sino construir el conjunto inicial de nodos hoja directamente desde la entrada, y después construir los nodos internos a partir de este conjunto. Inicialmente, todas las hojas excepto la última tienen un elemento más, el cual será utilizado para construir los nodos internos.

Construcción inicial

Por ejemplo

- Construir un árbol B con los siguientes valores.

4, 9, 13, 15, 16, 20, 22, 25, 29, 30, 33, 34, 39, 41, 42, 45, 47, 49, 62, 52, 57, 63, 73.



Búsqueda

- La búsqueda es similar a la de los árboles binarios. Se empieza en la raíz, y se recorre el árbol hacia abajo, escogiendo el sub-nodo de acuerdo a la posición relativa del valor buscado respecto a los valores de cada nodo. Típicamente se utiliza la búsqueda binaria para determinar esta posición relativa.

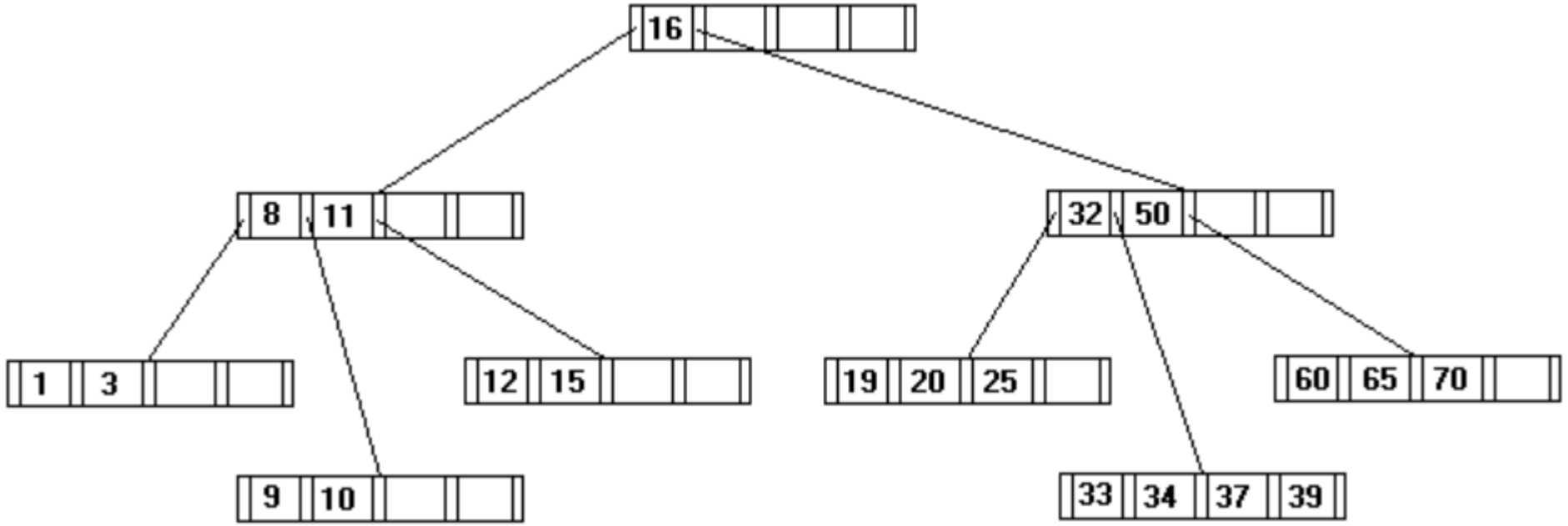
- **Procedimiento**

Situarse en el nodo raíz.

- (*) Comprobar si contiene la clave a buscar.
  - Encontrada fin de procedimiento.
  - No encontrada:
    - Si es hoja no existe la clave.
    - En otro caso el nodo actual es el hijo que corresponde:
      - La clave a buscar k < k1: hijo izquierdo.
      - La clave a buscar k > ki y k < ki+1 hijo iésimo.
      - Volver a paso (*).

Búsqueda

- Buscar el valor 15

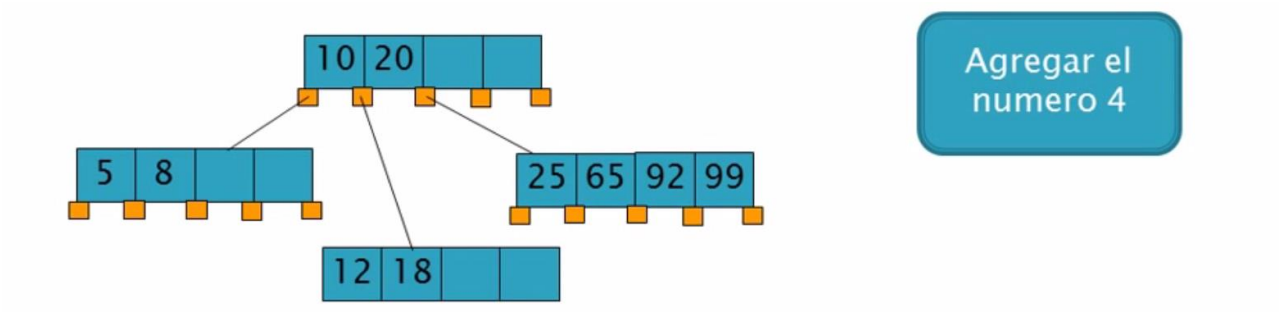


Insertar

- Las inserciones se hacen en los nodos hoja.

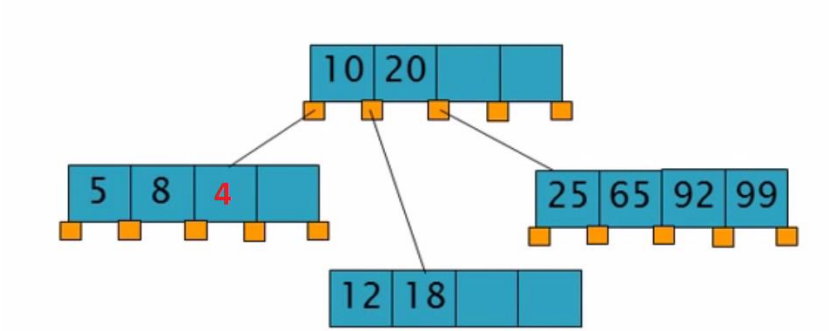
- Realizando una búsqueda en el árbol, se halla el nodo hoja en el cual debería ubicarse el nuevo elemento.
- Si el nodo hoja tiene menos elementos que el máximo número de elementos legales, entonces hay lugar para uno más. Inserte el nuevo elemento en el nodo, respetando el orden de los elementos.
- De otra forma, el nodo debe ser dividido en dos nodos. La división se realiza de la siguiente manera:
  - Se escoge el valor medio entre los elementos del nodo y el nuevo elemento.
  - Los valores menores que el valor medio se colocan en el nuevo nodo izquierdo, y los valores mayores que el valor medio se colocan en el nuevo nodo derecho; el valor medio actúa como valor separador.
  - El valor separador se debe colocar en el nodo padre, lo que puede provocar que el padre sea dividido en dos, y así sucesivamente.

Insertar

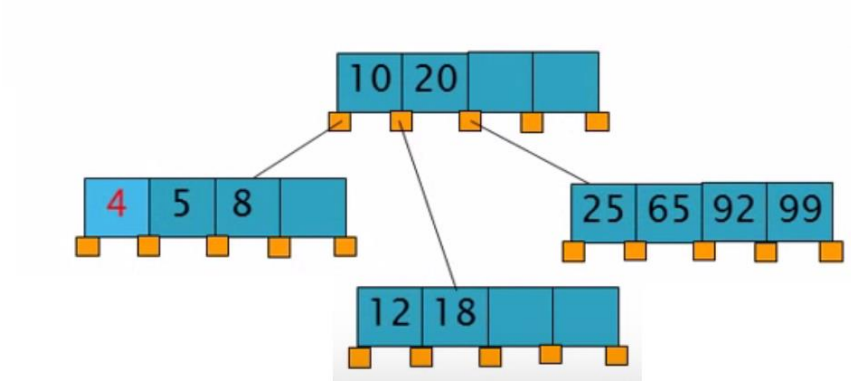


Agregar el numero 4

Paso 1

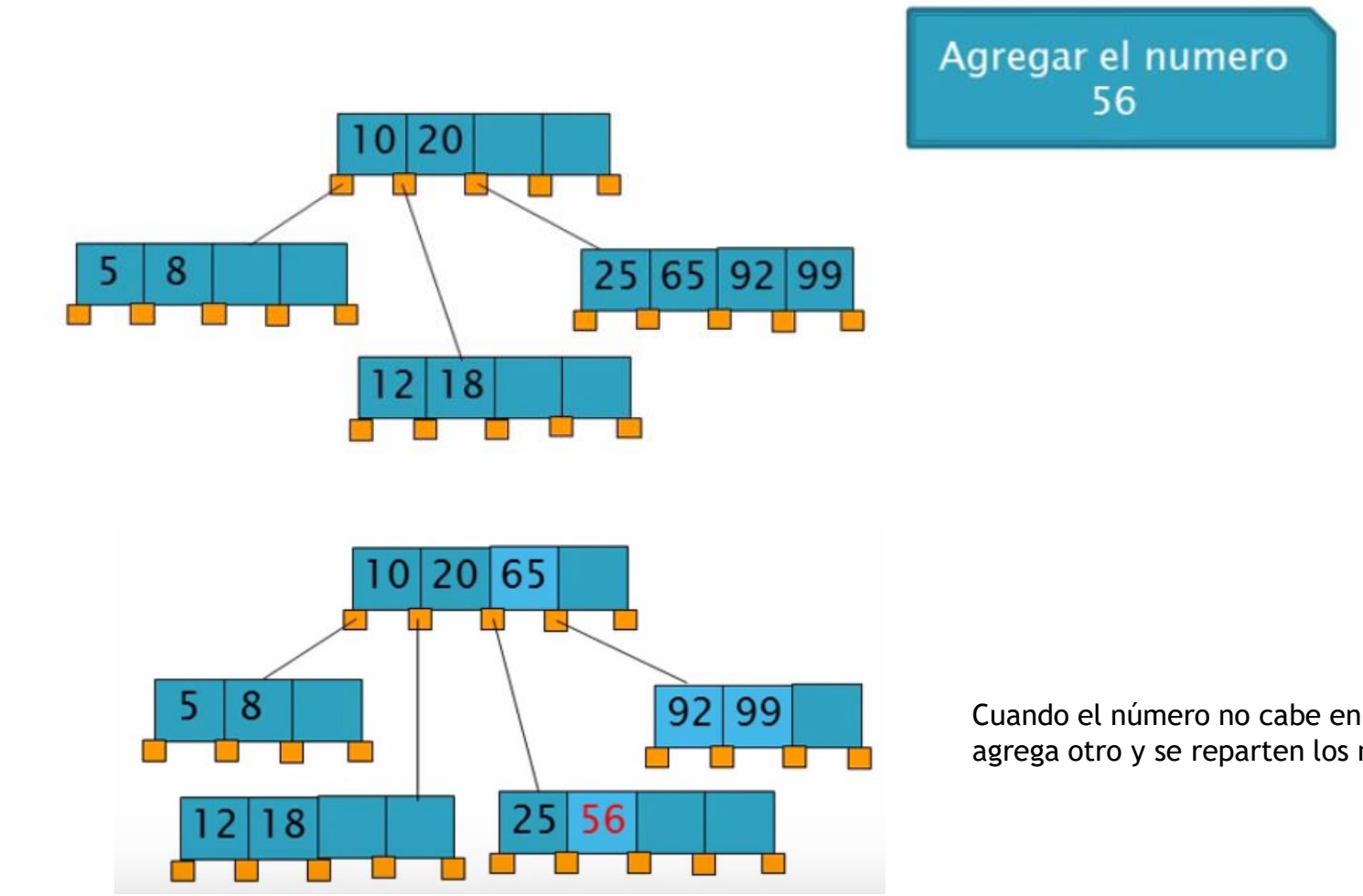


Paso 2



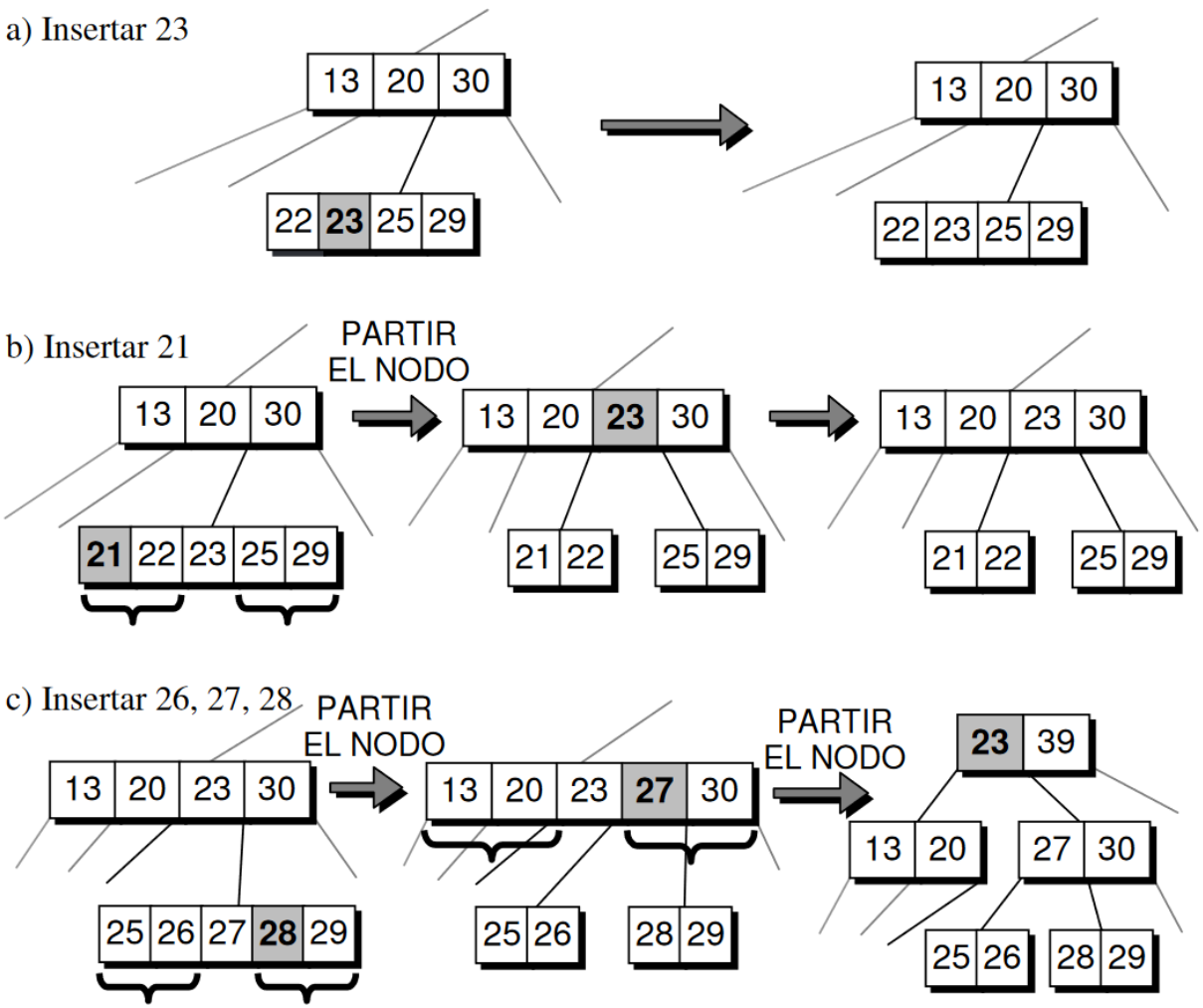
Si existe espacio se agrega el valor y se acomodan de menor a mayor

Insertar



Cuando el número no cabe en el nodo, se agrega otro y se reparten los números

Insertar


a) Insertar 23
13 20 30
22 23 25 29
13 20 30
22 23 25 29
b) Insertar 21
PARTIR
EL NODO
13 20 30
21 22 23 25 29
13 20 23 30
21 22
25 29
13 20 23 30
21 22
25 29
c) Insertar 26, 27, 28
PARTIR
EL NODO
13 20 23 30
25 26 27 28 29
13 20 23 27 30
25 26
28 29
PARTIR
EL NODO
23 39
13 20
27 30
25 26
28 29

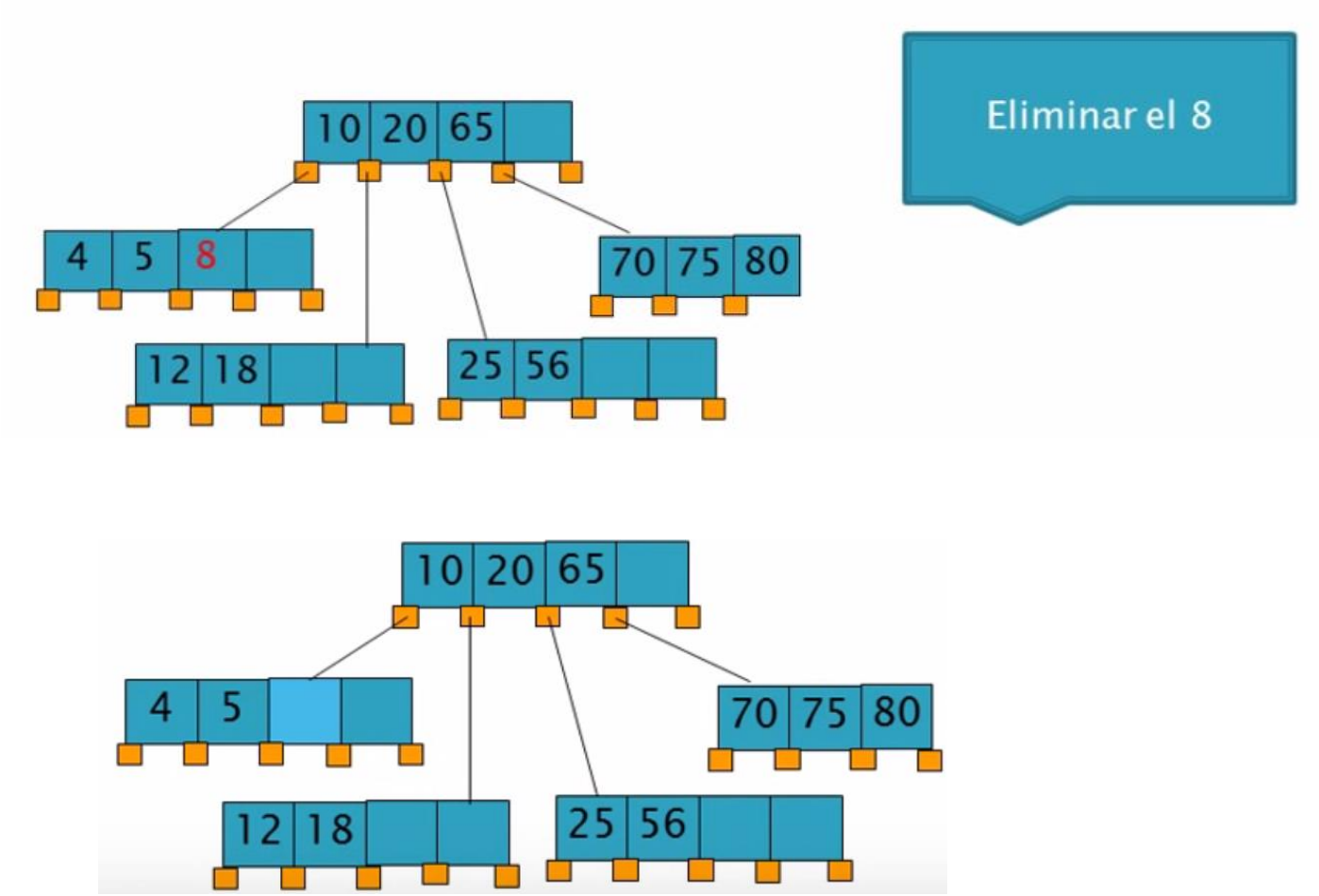
Eliminar

- La eliminación de un elemento es directa si no se requiere corrección para garantizar sus propiedades. Hay dos estrategias populares para eliminar un nodo de un árbol B.
  - Localizar y eliminar el elemento, y luego corregir, o
  - Hacer una única pasada de arriba a abajo por el árbol, pero cada vez que se visita un nodo, reestructurar el árbol para que cuando se encuentre el elemento a ser borrado, pueda eliminarse sin necesidad de continuar reestructurando
- Se pueden dar dos problemas al eliminar elementos. Primero, el elemento puede ser un separador de un nodo interno. Segundo, puede suceder que al borrar el elemento número de elementos del nodo quede debajo de la cota mínima. Estos problemas se tratan a continuación en orden.

Eliminar

- **Eliminación en un nodo hoja**
  - Busque el valor a eliminar.
  - Si el valor se encuentra en un nodo hoja, se elimina directamente la clave, posiblemente dejándolo con muy pocos elementos; por lo que se requerirán cambios adicionales en el árbol.

Eliminar



Cuando el nodo tiene más elementos que el mínimo, se elimina el elemento y se termina el proceso

Eliminar

- **Eliminación en un nodo interno**
  - En el segundo caso, uno de los dos nodos hijos tienen un número de elementos mayor que el mínimo.
  - Entonces izquierdo o el menor elemento del nuevo separador.
    - Como se ha eliminado un elemento de un nodo hoja, se trata este caso de manera equivalente.

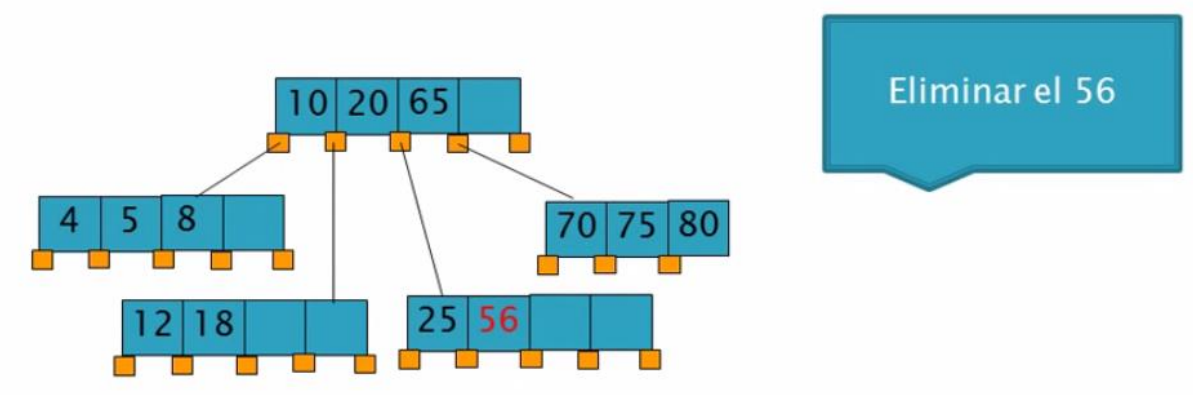
Eliminar

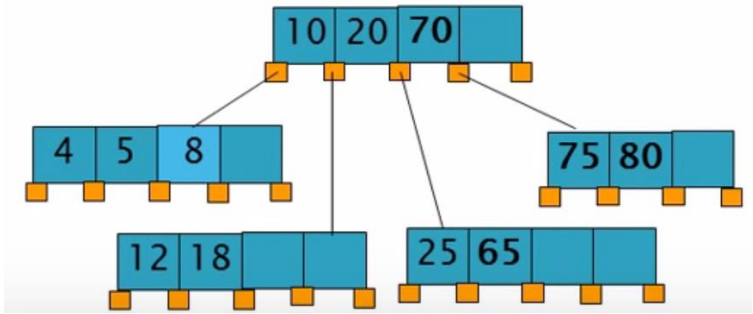
- **Rebalanceo después de la eliminación**
  - Si al eliminar un elemento de un nodo hoja el nodo se ha quedado con menos elementos que el mínimo permitido, algunos elementos se deben redistribuir.
  - En algunos casos el cambio lleva la deficiencia al nodo padre, y la redistribución se debe aplicar iterativamente hacia arriba del árbol, quizá incluso hasta a la raíz. Dado que la cota mínima en el número de elementos no se aplica a la raíz, el problema desaparece cuando llega a ésta.

Eliminar

- El proceso para eliminar es el siguiente:
  - Buscar el elemento a eliminar
  - Si el elemento a eliminar está en un nodo hoja, se lo eliminar y termina el proceso.
  - Si el elemento a eliminar no se encuentra en una hoja, se busca al sustituto más apropiado.

Eliminar





Paso 1: Eliminar el 56

Paso 2: Sube el 70

Paso 3: Baja el 65

Cuando el nodo tiene el mínimo se toma un elemento de los hermanos

GRACIAS